ZMotion PLC 编程手册

Version 2.0.0

第一章 ZPLC 编程入门

1.1 ZPLC 特点

1.1.1 PLC 执行原理

PLC 梯形图执行从左侧的母线开始，从左至右，从上至下依次扫描，从第一行程序开始顺序扫描到 END 为一个扫描周期，然后又开始新一轮程序扫描，直到程序被停止。

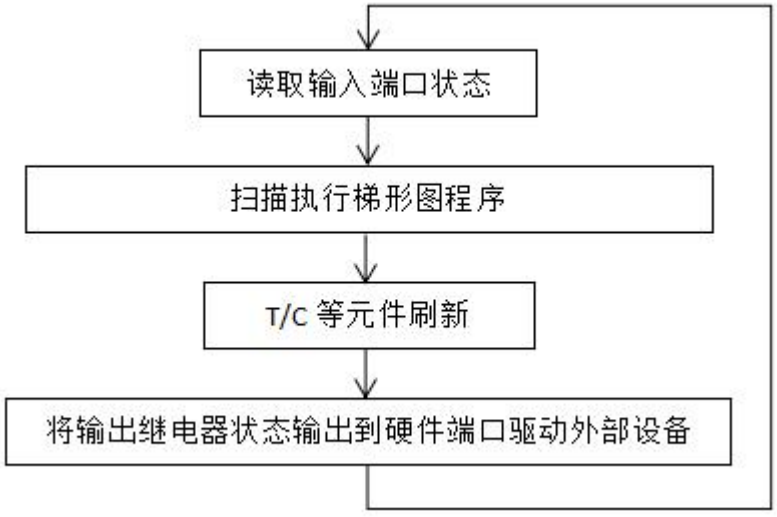
PLC 运行时，主要经过自检测、通讯处理、输入检测、程序执行、输出刷新这五个阶段。扫描程序之前，先执行故障检测与程序检查，发现异常停机显示出错信息，打印程序编写问题，再执行与其他设备的通信响应。

一个扫描周期包括输入检测、程序执行、输出刷新三个阶段。

输入检测：PLC 以扫描方式顺序读入各输入端子的通断状态，并写入相应的输入状态寄存器，即刷新输入，接着转入程序执行阶段。一般来说，输入信号的宽度要大于一个扫描周期，否则可能导致信号丢失。

程序执行：PLC 按从左至右，从上至下的顺序对每条梯形图指令进行扫描，并将相应的运算和处理结果保存在输出状态寄存器中。在程序执行的过程中，若输入信号状态发生改变，但此时状态寄存器内的输入状态没有改变，直到下一扫描开始时再读入输入信号状态。

输出刷新：在所有指令执行完毕后，输出状态寄存器的通断状态写入输出端子，驱动相应的输出设备。



扫描周期主要取决于程序的长短、指令的类型、CPU 执行指令的速度。扫描周期的时间可以通过特殊寄存器 D8010（扫描时间）、D8011（扫描最小时间）、D8012（扫描最大时间）来查看。

1.1.2 用户程序

用户程序是用户编写的程序代码，下载到控制器后执行控制功能，由主程序、子程序、中断程序三类程序体组成，其中子程序和中断程序不是必要的。

主程序

主程序是用户程序的主体和框架，系统处于运行状态时，主程序循环执行，PLC 一般只有一个主程序。

中断程序和子程序一般放于主程序后方，使用指令触发运行，主程序以 FEND 指令结束，FEND 指令下方程序调用时才扫描执行。

中断程序

中断程序是处理系统中断事件的程序，中断信号的请求能及时的响应处理，是中断功能

的主要特点，当系统中断发生时，主程序被打断，程序自动跳转到该中断执行，中断执行完返回到主程序断点处继续往下执行。

中断使用需要满足两个条件，一是使用 EI 指令开启中断总开关，二是满足中断触发条件。EI 指令前面必须加触点，不能与母线直接相连，通过 LBL 指令调用中断函数、定义中断条件。

中断分为三种：掉电中断、外部中断、定时器中断。

掉电中断在 PLC 掉电时触发，执行时间有限，只能写入较少语句，程序将关注的数据存储到 VR。

外部中断指令的输入口编号必须和触发编号一致，且输入口必须是控制器自带的编号，否则该外部中断无法触发，外部输入有效时才触发外部中断执行一个扫描周期后返回主程序，再次触发后再导通一个扫描周期。外部中断一般最多支持 32 个，编号 0~31，与 Basic 编程外部输入中断相同。

定时器中断达到定时器设定的时间后触发执行，支持的定时器数量使用?*max 打印，查看 max_timer 参数确定。

各类型中断优先级一致。

中断函数：

EI：允许中断，同 BASIC 命令，INT_ENABLE=1 开启

DI：禁止中断，同 BASIC 命令，INT_ENABLE=0 关闭

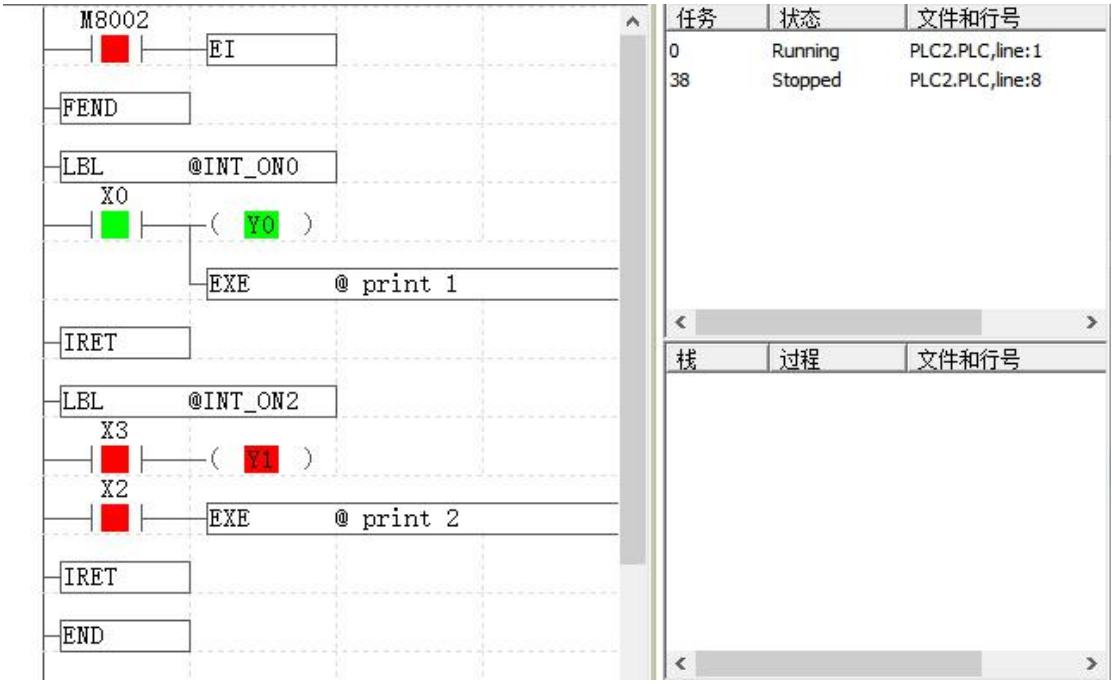
ONPOWEROFF：掉电中断

INT_ONn：外部输入中断，上升沿有效，n-输入口编号

INT_OFFn：外部输入中断，下降沿有效，n-输入口编号

ONTIMERn：定时器中断，n-定时器编号

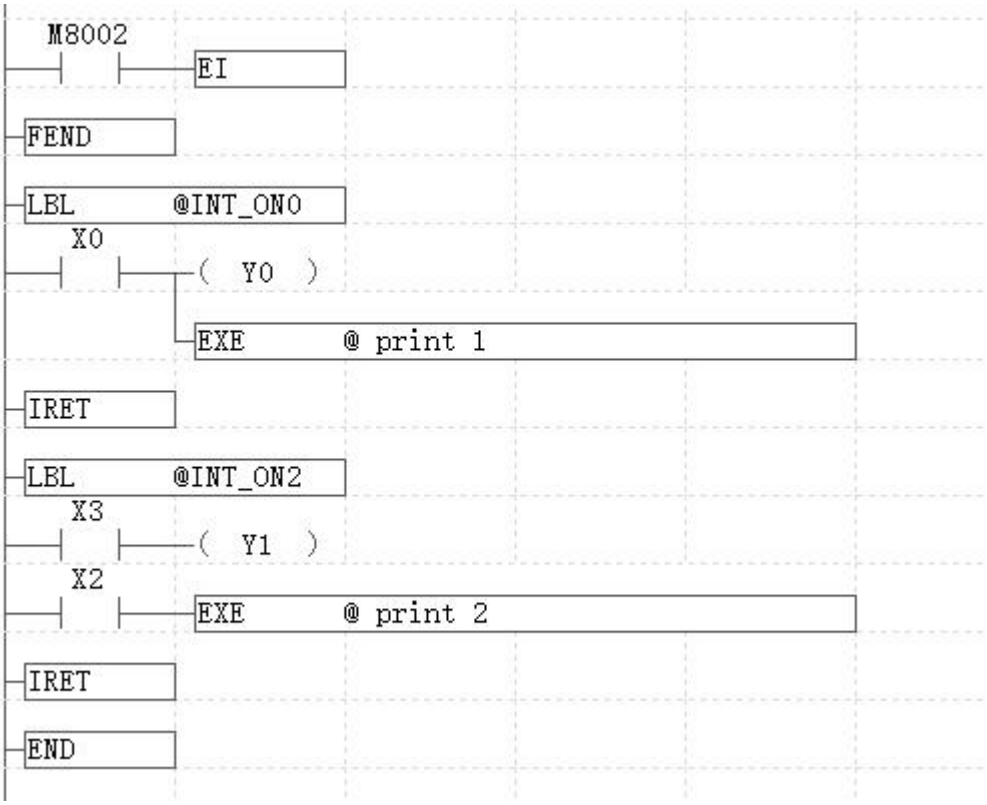
如下图，中断触发后，中断程序以一个任务运行，并提示中断处于哪个文件内，注意中断程序不宜过长，否则会导致该中断堵塞，影响其他中断程序执行，中断程序较长的建议单独开启任务去执行。



例：上电后 M8002 导通一个扫描周期，中断开启，闭合 X0，外部上升沿中断 0 触发，中断子程序执行一次，Y0 一直导通（中断子程序保留最后一个扫描周期的特性，导通后不

管 X0 什么状态，Y0 保持），打印一次数字 1，外部上升沿中断 0 结束返回。

X2 未闭合前，先闭合 X3，Y1 不导通，原因在于 X2 不闭合该中断程序未触发执行，PLC 不扫描该中断。接通 X2，外部上升沿中断 2 触发，中断子程序执行一次，打印一次数字 2 后中断结束返回。



子程序

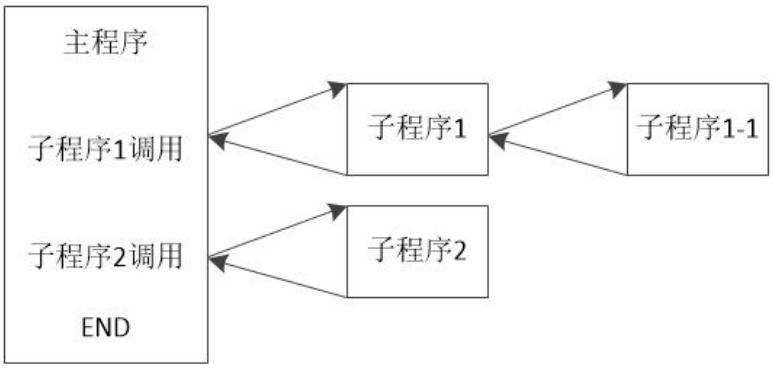
子程序功能是独立的，可以被其他程序调用。子程序可以选择是否带返回参数。

用户程序没有子程序，也可以设置多个子程序。

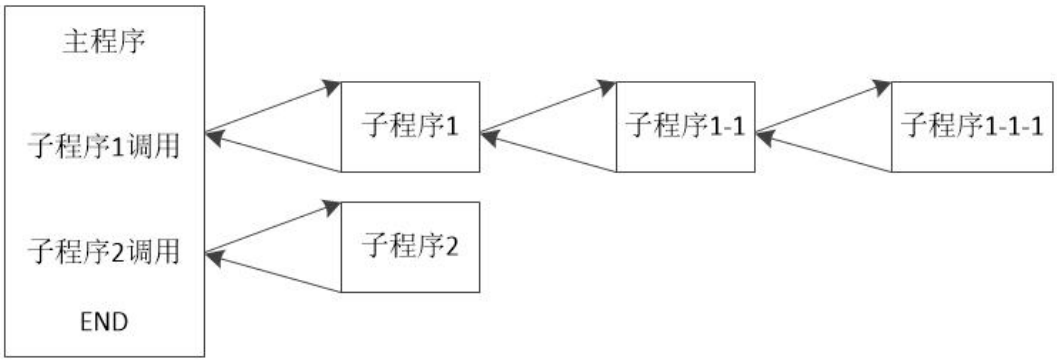
子程序和中断程序均使用 LBL 标明，一般放于主程序 FEND 之后，子程序只有在 CALL 指令调用时才执行，执行时循环扫描、SRET 子程序返回指令返回到主程序。

子程序中还可以嵌套其他子程序，不能调用自身，嵌套调用的深度不能超过八层。

子程序调用示意图：



子程序两层嵌套示意图

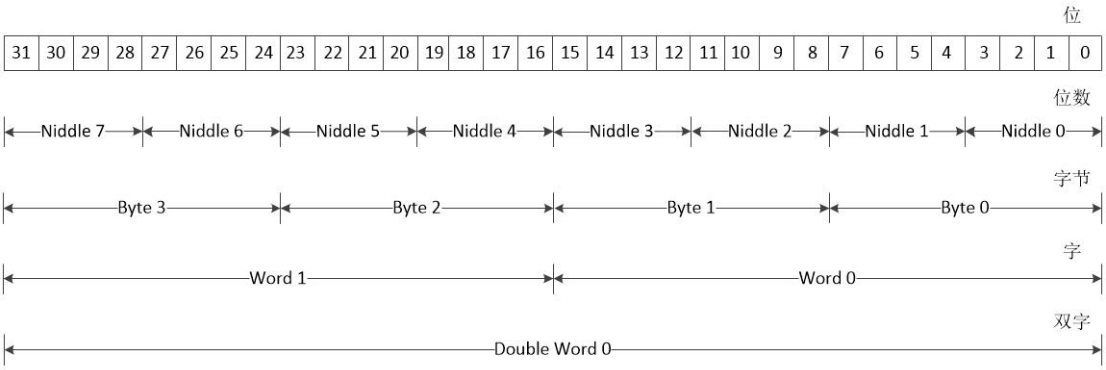


子程序三层嵌套示意图

1.1.3 数据类型

二进制数据

位（bit）为二进制数值最基本的单位，其状态非 1 即 0；4 个连续的位（bit）组合成 1 个位数（Nibble）；8 个连续的位（bit）组合成 1 个字节（Byte）；16 个连续的位（bit）组合成 1 个字（Word）；32 个连续的位（bit）组合成双字（Double Word）。



PLC 常用数据类型

常数 K

K 表示 10 进制整数的符号，主要用于计数器或定时器的设定值，或应用指令操作数中的数值。16 位指令中，K 的取值范围是-32768 - +32767；32 位指令中，K 的取值范围是-2147483648 - +2147483647。

常数 H

H 表示十六进制数的符号，主要用于指定应用指令的操作数的数值。16 位指令中，常数 H 的取值范围是 0000-FFFFh；32 位指令中，H 的取值范围是 0000 0000-FFFF FFFFh。

八进制

八进制数由三位二进制数组成，每个数字位范围 0-7，不存在 8 和 9，在 PLC 中，输入继电器 X 与输出继电器 Y 采用八进制。

BCD 码

BCD 码用四位二进制数表示一位十进制，每个数字位的范围为 0-9。

BIN 码

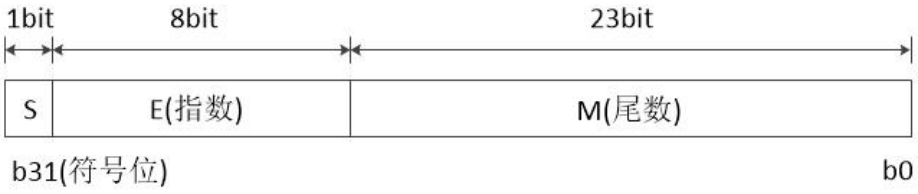
BIN 为二进制编码，每个数字位只能为 0 或 1。

浮点数

ZPLC 内浮点数均为 32 位单精度浮点数，只能使用 32 位浮点数指令处理浮点数数据。

浮点数以小数点的方式来表示，用 32 位的寄存器长度表示浮点数，表示方法采用 IEEE754 标准，标准格式如下：

最高位 b31 为符号位，b30~b23 共 8 位为指数位，b22~b0 共 23 位为尾数位。



表达式：$(-1)^S \times 2^{E-B} \times 1.M$（其中 S 为符号位，0 为正数，1 为负数，B=127，E 的取值范围为 1~254），因此单精度浮点数的取值范围为$\pm 2^{-126}$~$\pm 2^{+127}$。

例：将浮点数 23.0000 保存到数据寄存器 D0，此时 D1D0 组成 32 位空间用以存储浮点数。

将 23 转换成二进制数 10111。

$(10111)_2=1.0111 \times 2^4$，其中 0111 为尾数，4 为指数。根据表达式得 E-B=4，E=4+B=4+127=131。

131 转换成二进制数为 10000011。

组合符号位、指数位、尾数得出浮点数 0 10000011 01110000000000000000000，转换为十六进制为 41B80000，此时寄存器的值为 D1=16842，D0=0。

有符号数

PLC 内部数据进行四则和函数运算时，都按有符号数进行运算操作。

符号位为 0 时，表示为正数。符号位为 1 时表示负数，负数在计算机中以补码的形式存储。

16 位数据寄存器 D 中的最高位 bit15 为符号位，故数据寄存器 D 的取值范围是-32768~+32767。32 位数据使用两个连续的 16 位数据空间表示，此时最高位 bit31 为符号位，数据寄存器 D 的取值范围是-2147483648 - +2147483647。

进行数值较大的运算时，要注意符号的处理，尤其是出现进位或借位操作时，要进行进位标志、借位标志的判断与相应处理，否则导致计算结果出错。

无符号数

没有符号位，默认均为正数，对于 16 位寄存器，其取值范围是 0~65535，在计时与计数的应用场合，只有正数，故按照无符号数处理。

当进行逻辑运算时（逻辑与、逻辑或、逻辑异或等），操作数当无符号数处理，符号位与其他位同等参与逻辑运算。

进制转换举例：

二进制 BIN	八进制 OCT	十进制 DEC	十六进制 HEX	BCD 码	二进制 BIN	八进制 OCT	十进制 DEC	十六进制 HEX	BCD 码
0000	0	0	0	0000	1000	10	8	8	1000
0001	1	1	1	0001	1001	11	9	9	1001
0010	2	2	2	0010	1010	12	10	A	0001 0000
0011	3	3	3	0011	1011	13	11	B	0001 0001
0100	4	4	4	0100	1100	14	12	C	0001 0010
0101	5	5	5	0101	1101	15	13	D	0001 0011
0110	6	6	6	0110	1110	16	14	E	0001 0100
0111	7	7	7	0111	1111	17	15	F	0001 0101

主要用途：

10 进制数 DEC	8 进制数 OTC	16 进制数 HEX	二进制数 BIN	BCD 码
常数 K 以及输入输出继电器以外的内部软元件编号	输入继电器、输出继电器的软元件编号	常数 H 等	PLC 内部数据处理	BCD 数字式开关，7 段码显示器

1.1.4 指令特点

ZPLC 是 ZMotion 运动控制器支持的 PLC 语言，可以使用梯形图和语句表。

指令格式说明

指令操作数释义：

例如：SMOV 指令格式：[SMOV　S　m1　m2　D　n]

S：源操作数，执行指令前后 S 的值不变，有多个源操作数用 S1，S2 表示。

D：目标操作数，执行指令后，D 的值发生变化，有多个目标操作数用 D1，D2 表示。

m/n：不符合上述两种操作数的其他操作数，有多个时用 m1，m2，n1，n2 表示。

所有软元件的使用必须指定编号，一般来说，软元件的使用次数没有限制。

指令形式

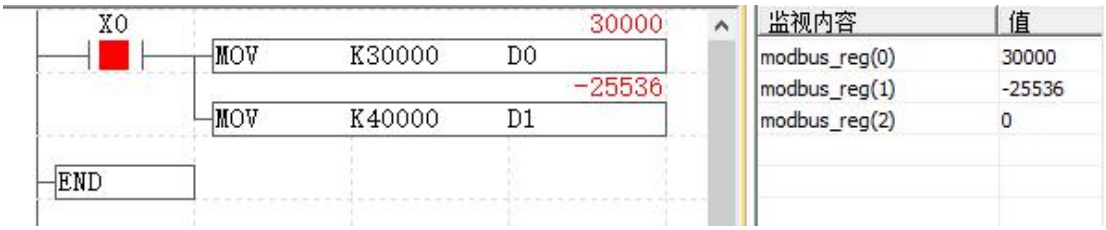
处理数值的应用指令中，根据数值的位长分为 16 位指令和 32 位指令两种，16 位数据和 32 位数据处理采用不同的指令，除了数据长度不同外，二者其他方面均相同，处理数据类型均为有符号数。

16 位指令

传送的数值范围：-32768 -+32767。

执行 16 位数据的传送，若源操作数 S 大于 16 位，低 16 位数据保存在目标操作数 D 中，溢出数据舍弃。

例：第二条 MOV 指令传送数据 40000 超过指令处理范围，溢出数据舍弃。

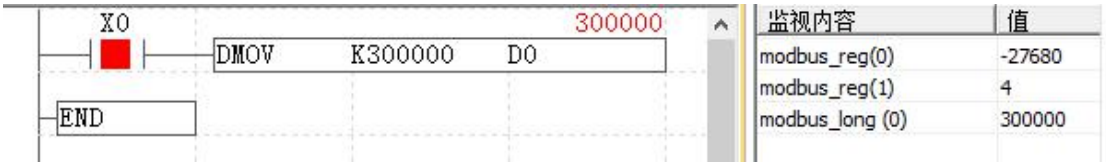


32 位指令

传送的数值范围：-2147483648 - +2147483647。

16 位指令上添加 D 符号表示 32 位指令。每个 32 位数据使用连续两个 16 位空间存储，数据若超出 32 位，溢出部分舍弃。

例：常数 300000 在 16 位与 32 位之间，使用 D1D0 两个连续寄存器空间存储数据，D1 为高 16 位，D0 为低 16 位。



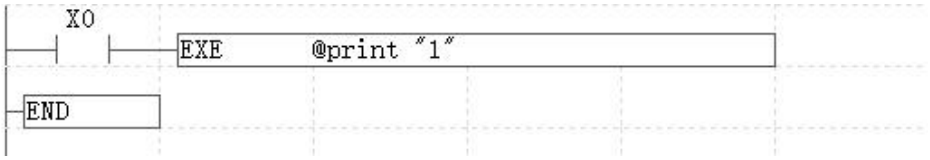
执行方式

根据指令的执行方式的不同，分为连续执行型和脉冲执行型两种。

连续执行

满足条件，每个扫描周期都执行一次。

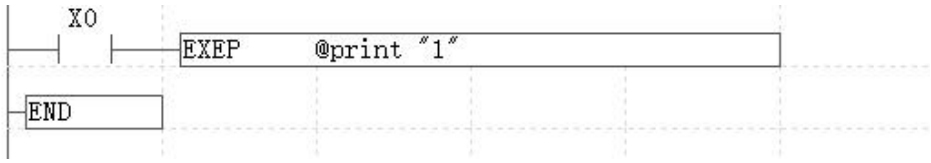
例：闭合 X0，循环不断的打印出数字“1”。



脉冲执行

指令添加 P 符号表示脉冲执行型指令，即给一个脉冲仅执行一次（状态由 OFF 变为 ON 时）。不需要一直执行的情况建议使用脉冲执行型指令。

例：闭合 X0，打印一次数字“1”，X0 每闭合一次执行一次打印。



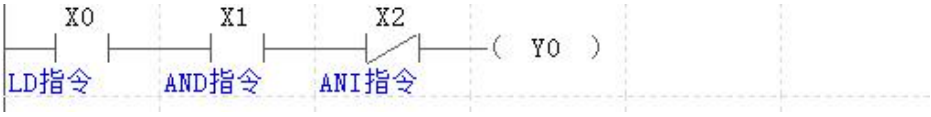
触点类型

触点按连接方式可划分为串联和并联两大类。

串联触点

AND、ANI 指令是执行串联连接的触点，串联触点指令可以连续多次使用。

在梯形图编程方式下，常开触点可输入 LD 指令或 AND 指令，得出的梯形图软元件相同，如下图 X0、X1 为串联。

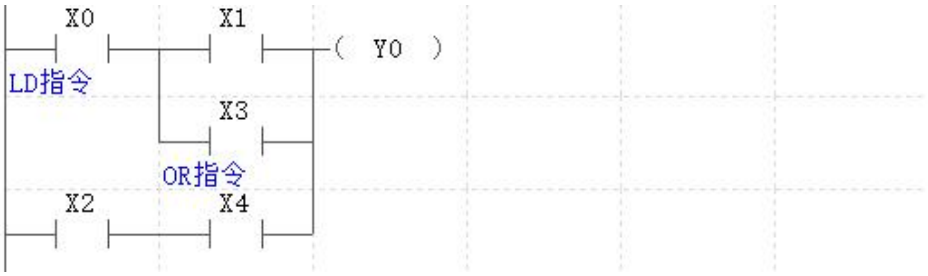


梯形图对应语句表编程如下。

```
LD X0
AND    X1
ANI    X2
OUT Y0
```

并联触点

OR、ORI 指令是执行并联连接的触点，并联触点指令可以连续多次使用。如下图 X1 与 X3 为并联。



梯形图对应语句表编程如下，ANB 为块串联指令，将触点 X1 和 X3 组成的块与触点 X0 串联；ORB 为块并联指令，将触点 X2 和 X4 组成的块与上方触点并联。

```
LD  X0
LD      X1
OR      X3
ANB
LD      X2
AND     X4
ORB
OUT y0
```

1.1.5 多任务运行

一个正在运行的程序称为一个任务，多任务运行指 PLC 可以支持多个 PLC 任务同时运行，且任务之间互不干扰。

PLC 主任务：第一个设置自动运行的 PLC 任务为主任务，PLC 循环的相关状态参数由这个任务得到，PLC 推荐设置一个自动运行主任务（若设置两个或两个以上 PLC 自动运行主任务，下载会提示 WARN 信息，但可以正常运行），其他任务由主任务调用。

控制器支持多 PLC 任务同时运行，也可以 PLC 和 Basic 任务一起运行。

当中断任务运行时，PLC 主任务暂停运行，其它 PLC 任务不受影响，与中断任务同时运行。

1.1.6 掉电存储

各个寄存器有部分是掉电保持的，与 VR 类似。可以通过调用 Basic 的 SETCOM 命令的 variable 参数寄存器选择，配置寄存器 D 是否掉电保持。

当需要掉电保持时，推荐 variable=3；不需要掉电保持时，使 variable=2。

variable 参数是全局的设置，所有的端口共一个。

variable 值	描述
0	VR，此时一个 VR 映射到一个 MODBUS_REG
1	TABLE，此时一个 TABLE 数据映射到一个 MODBUS_REG（不推荐）
2（缺省）	系统 MODBUS 寄存器，此时 VR 与 MODBUS 寄存器是两片独立区间
3	VR_INT 模式，此时一个 VR_INT 映射到两个 MODBUS_REG

1.1.7 PLC 步数

不同型号控制器的程序容量不一样，空间和 ZBasic 程序共享，没有 ZBasic 程序时，大部分控制器总步数在 40K 步上下。

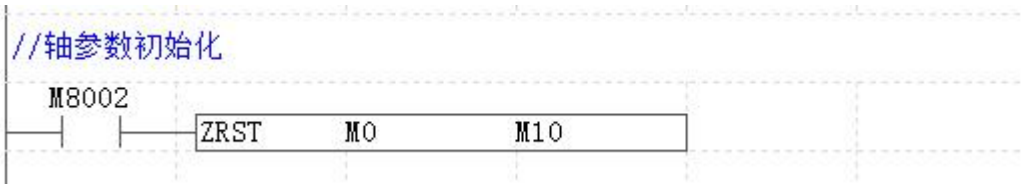
每个指令占用的步数是寄存器个数+1,@寄存器多占用一步。

1.1.8 注释

PLC 梯形图添加注释方式有三种，使用 PLC 注释指令、PLC 右键菜单“编辑批注”、菜单栏“视图”-“注释”窗口。

注释指令“//”

使用该指令注释占一行，一般在某个功能模块前添加注释，补充说明该功能的作用等相关信息。



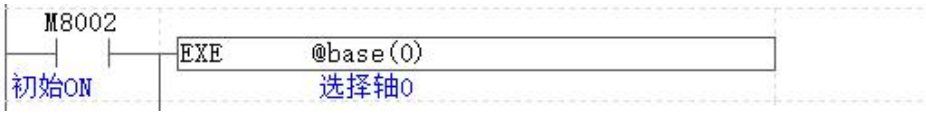
编辑批注

此功能在梯形图界面下可对软元件进行注释。

批注方法：选中要批注的软元件，点击右键选择批注编辑。

弹出批注编辑窗口之后，在批注下方双击左键，输入注释内容，点击确定。

完成操作成功显示批注，如下图，批注内容默认显示，不希望显示批注时不勾选“显示批注”这一项即可隐藏批注。



注释窗口

注释窗口主要用于对 PLC 元件进行批注。

其中“系统注释”主要针对 PLC 特殊继电器 M 和特殊寄存器 D，可快速对所有特殊元件进行注释。

“项目注释”自定义输入寄存器名和注释，与编辑批注使用类似，支持将此窗口的项目注释导入导出。

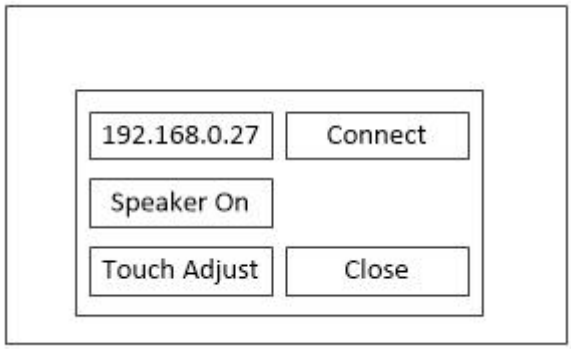
1.1.9 连接人机界面

与 ZHD 示教盒连接

ZHD300(X)系列串口示教盒直接使用配发的连接线接到控制器的 232 接口上即可通讯。

ZHD400(X)系列网口示教盒直接使用配发的连接线接到控制器的网口上，然后在屏上四个角，如下图，按画 Z 字顺序点击左上角、右上角、左下角、右下角，连续 2 次，弹出可以弹出设置窗口，可以进行触摸校正，控制器 IP 修改等。

设置窗口如下，连好控制器触摸屏自动获取控制器 IP，点击 Connect 即可连接使用。



ZHD300 使用 ZDevelopHD 软件开发，需要手动画出每个控件，不建议使用。

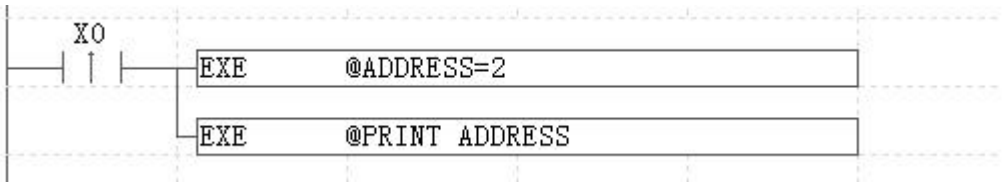
ZHD300X 和 ZHD400X 使用 ZDevelop 软件的 Zhmi 模式开发，程序保存在控制器中，不需要下载到触摸屏内，控制器下好程序连接触摸屏即可使用。

ZHD400 也使用 ZDevelop 软件的 Zhmi 模式开发，但是程序需要单独下载到示教盒内，与控制器程序分离。

与其他厂家人机界面连接

控制器支持 MODBUS-RTU 协议或 MODBUS-TCP 协议连接人机界面，即支持这两种通讯协议的人机界面均可与控制器通讯。

调用 Basic 的 ADDRESS 参数设置协议站号，缺省 1。可以在 Basic 中设置，也可以在 PLC 中调用 ADDRESS 指令来实现。

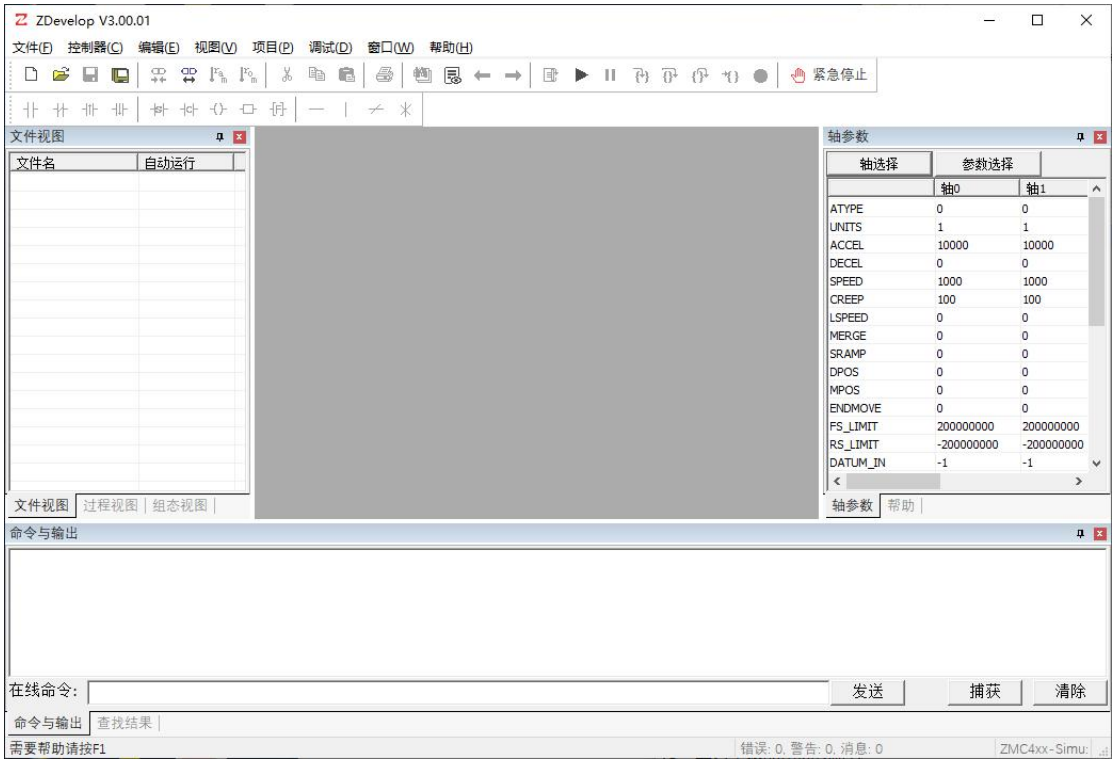


串口连接时配置好对应串口号，以太网连接时配置人机界面的 PLC 端 IP 为控制器的 IP，端口号 502。控制器 IP 默认为 192.168.0.11，可通过 Basic 的 IP_ADDRESS 指令查看/修改 IP，无控制器时，人机界面可以连接到仿真器，仿真器只能通过网口连接，IP 为 127.0.0.1。

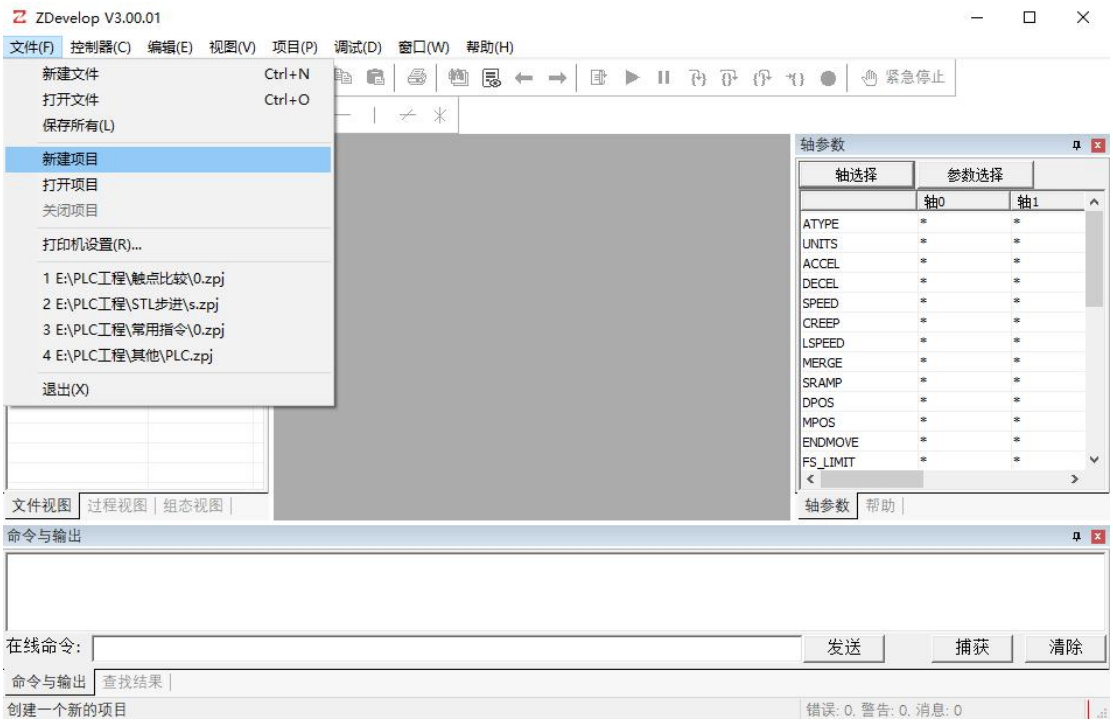
1.2 PLC 编程软件使用

1.2.1 新建工程

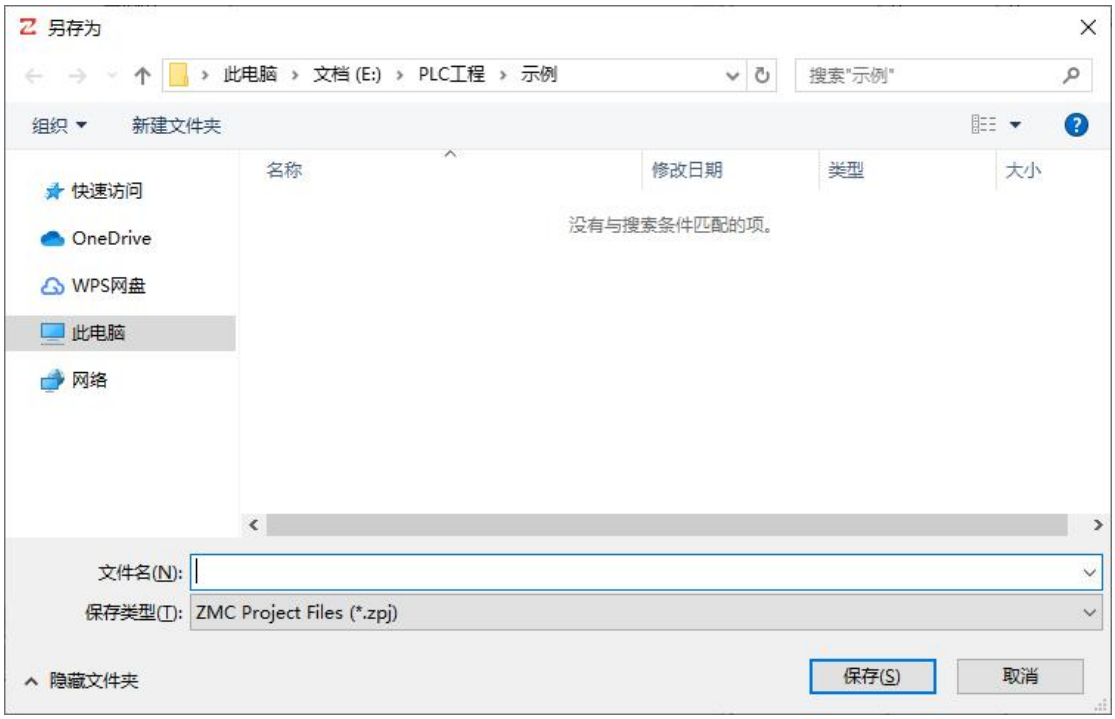
在正运动官方网站 www.zmotion.com.cn 下载 ZDevelop 编程软件最新版本安装。
当前示例版本为 ZDevelop3.00.01，进入 ZDevelop 编程界面。



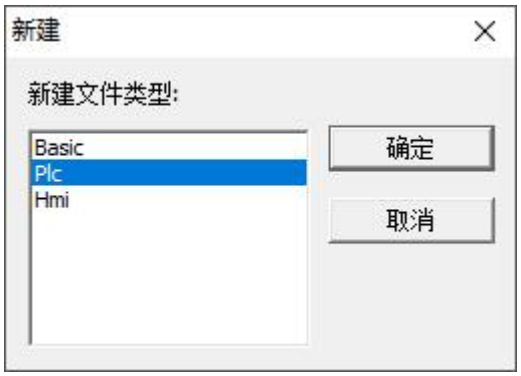
1、新建项目：点击菜单栏“文件”-“新建项目”。



弹出另存为界面，给 ZPJ 文件命名，将文件保存（最好新建一个文件夹，将项目文件均存入此文件夹下）。

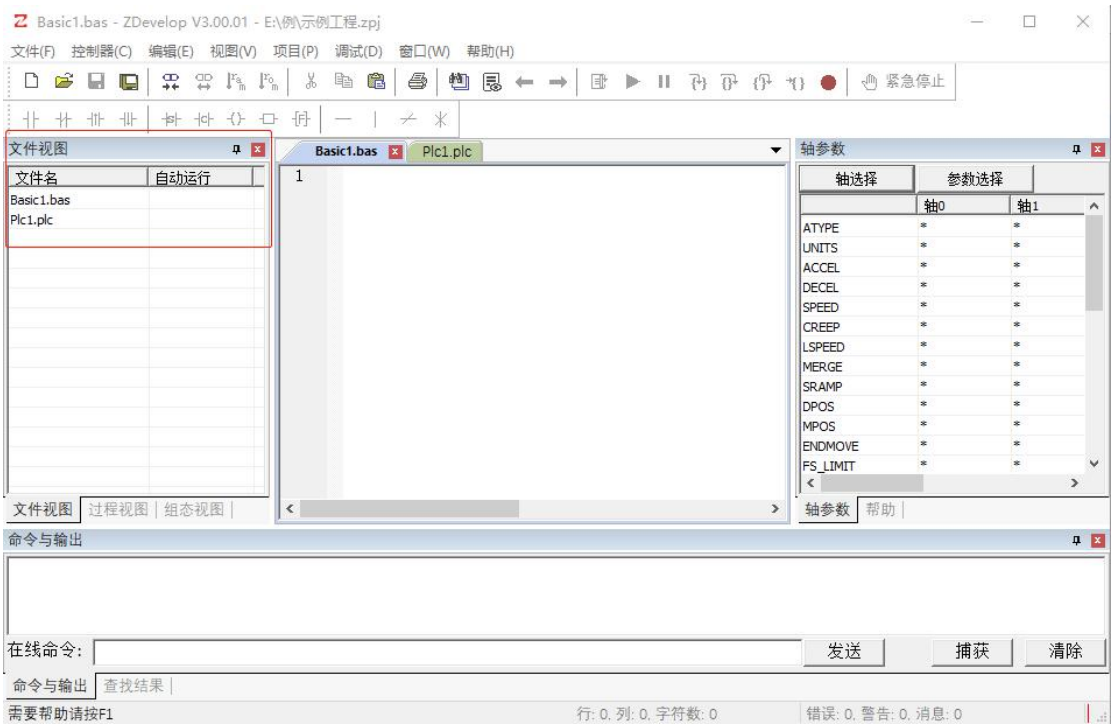


2、新建文件：在点击菜单栏“文件”-“新建文件”，弹出如下文件类型选择窗口，软件支持如下三种编程方式，此处以 PLC 文件为例（Basic、Hmi 新建工程步骤与 PLC 相同）。



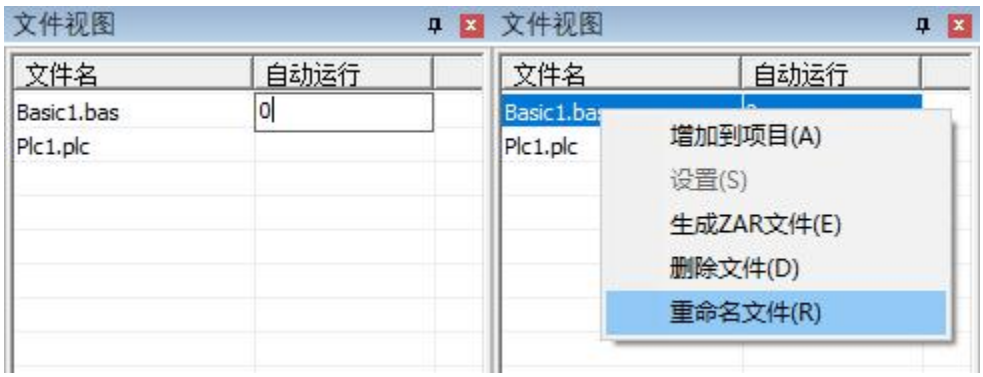
3、保存文件：确认后文件自动添加到项目，如下图“文件视图”所示。

PLC 文件和之前建立的 ZPJ 文件必须位于同一文件夹下，一个项目里可新建多个 PLC 文件，或其他编程方式的文件（例如 Basic 和 PLC 混合编程），只需将文件保存在同一个文件夹下。

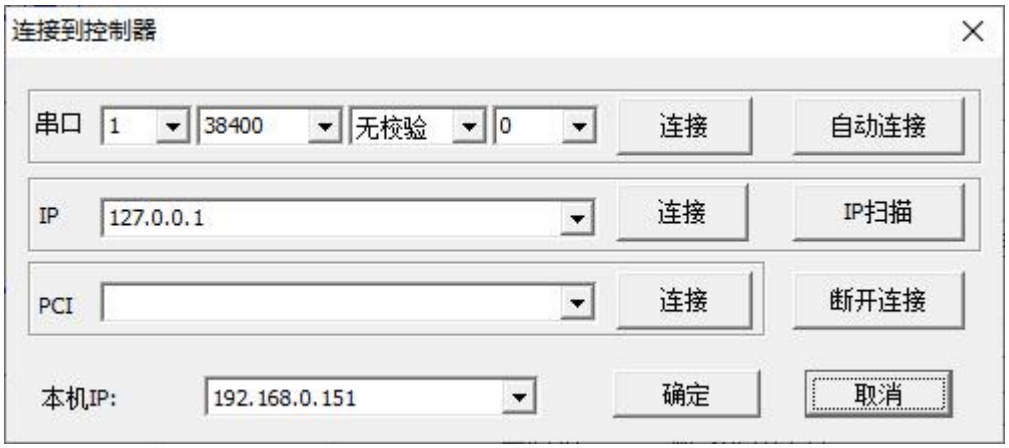


4、设置自动运行任务号：点击文件左边自动运行处，设置自动运行任务号 0（数字 0 仅表示任务编号）。

文件名称可自定义，在文件处点击鼠标“右键”-“重命名文件”修改。

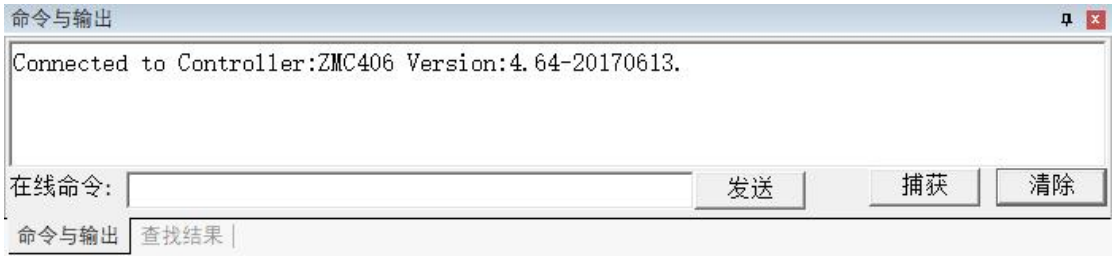


5、下载运行程序：在右方程序编辑区域写好程序之后，点击 按钮（或菜单栏“控制器”-“连接”），弹出连接到控制器窗口，选择串口或网口连接参数后连接控制器，串口参数按程序 SETCOM 指令的参数设置，网口连接在 IP 下拉框里选择控制器 IP 地址直接连接，详细连接方法参见下节“连接控制器”。

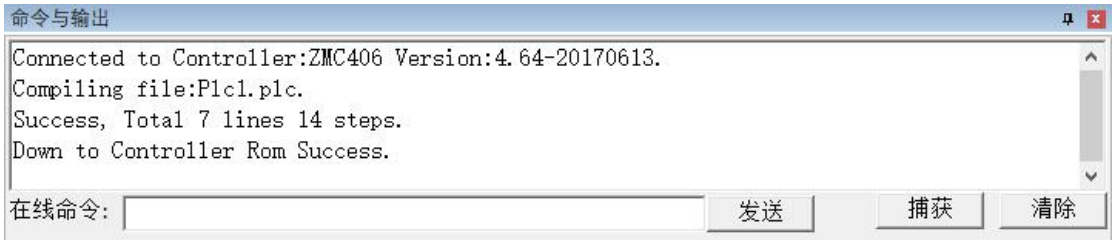


没有控制器时，可连接到仿真器，菜单栏“控制器”-“连接到仿真器”即可连上。

连接成功命令与输出打印如下信息，包含控制器型号和固件版本号。



点击菜单栏按钮 “下载到 RAM”或按钮 “下载到 ROM”，下载成功命令和输出窗口会有提示“Down to Controller Rom Success.”，同时程序下载到控制器并自动运行。RAM 下载掉电后程序不保存，ROM 下载掉电后程序保存。下载到 ROM 的程序下次连接上控制器之后程序会自动运行。



程序下载到控制器之后就可以执行运动指令，控制轴运动。

注意：

不建立项目的时候，只有 Bas 文件无法下载到控制器。

打开工程项目时，选择打开项目 ZPJ 文件，只打开其中的 Bas 文件无法下载到控制器。

自动运行的数字 0 表示任务编号，以任务 0 运行程序。一般一个 PLC 工程仅设置一个自动运行任务号，采用指令调用的方式开启多任务。

若整个工程项目内的文件都不设置任务编号，下载到控制器时，系统提示如下信息。

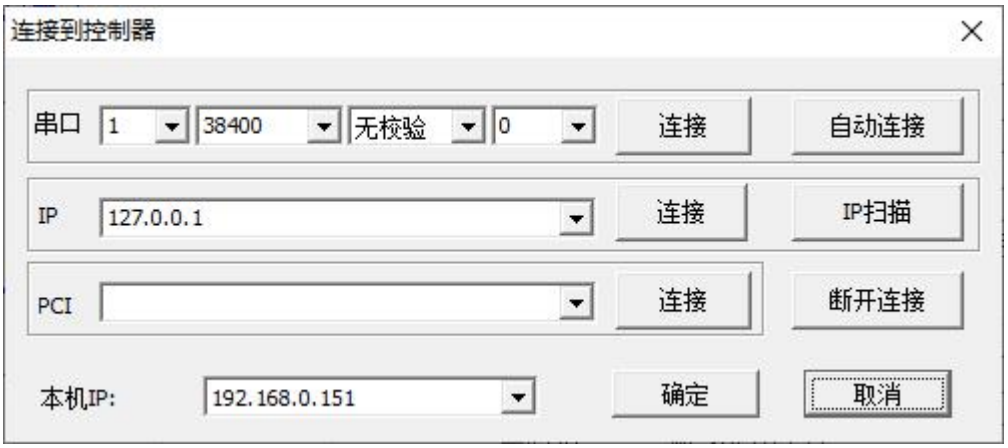
WARN:no program set autorun.

1.2.2 连接控制器

ZDevelop 软件连接到控制器有两种方式，串口连接和网口连接。

串口连接方式

将控制器与需要连接的设备进行物理连接，选择合适的串口对应连接起来，物理连接完成后打开 ZDevelop 软件，点击菜单栏的“连接”，弹出下面窗口：



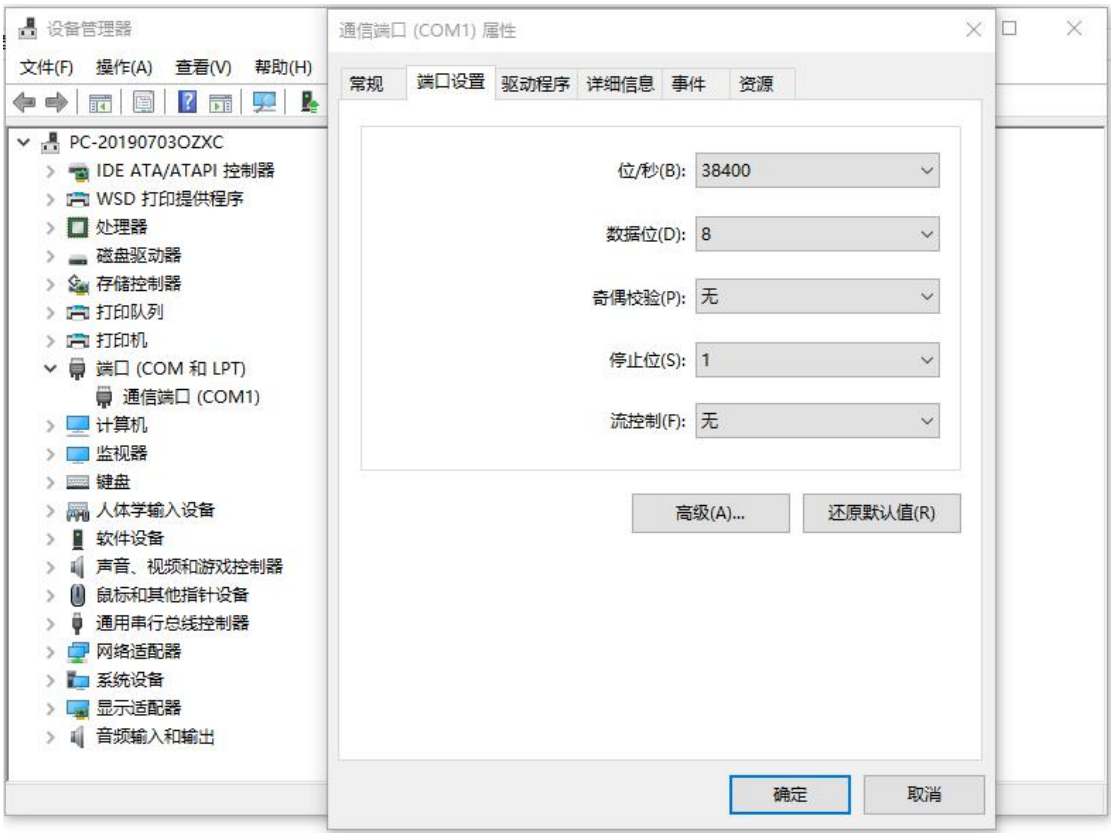
第一行为串口连接方式，选择需要连接的串口编号、设置波特率、校验位、停止位之后，点击连接，连接是否成功会在软件输出窗口自动打印出相应信息。

若连接失败，按下面方法依次排查：

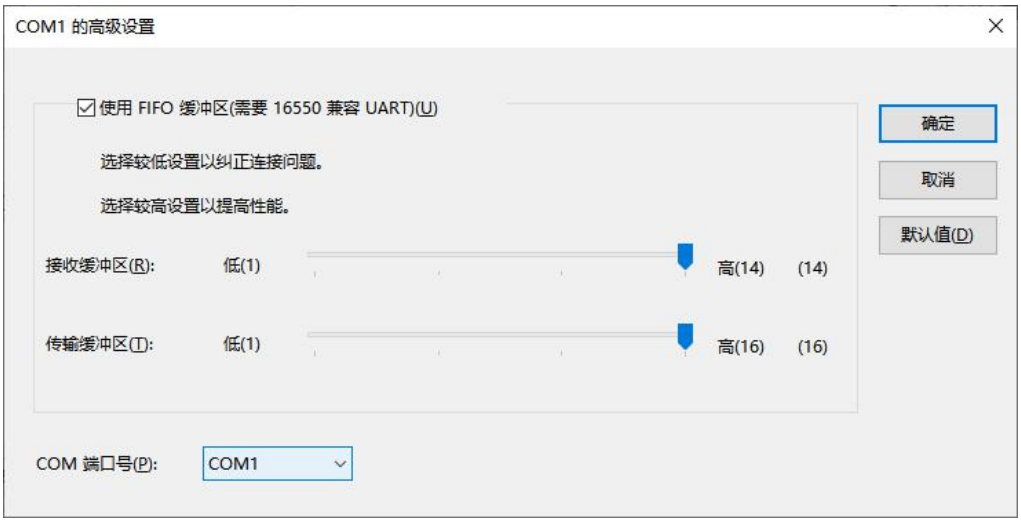
1．查看串口连接线是否为交叉线。

2．“连接到控制器”里的 COM 口编号、参数是否选择正确。

3．打开电脑“设备管理器”-“端口”-“通信端口（COM）”-“端口设置”，查看 COM 口设置是否正确，控制器串口默认参数 波特率 38400，数据位 8，停止位 1，校验位无。



在“端口设置”-“高级”选项中可更改 com 端口号，通过下拉列表选择。



4．当通过串口连接到控制器时，对应的控制器串口必须配置为 MODBUS 从协议模式（缺省模式），断电重启即可恢复。

5．COM 口是否已被其他程序占用，如串口调试助手等。

6．PC 端是否有足够的串口硬件。

更换串口线/电脑测试。

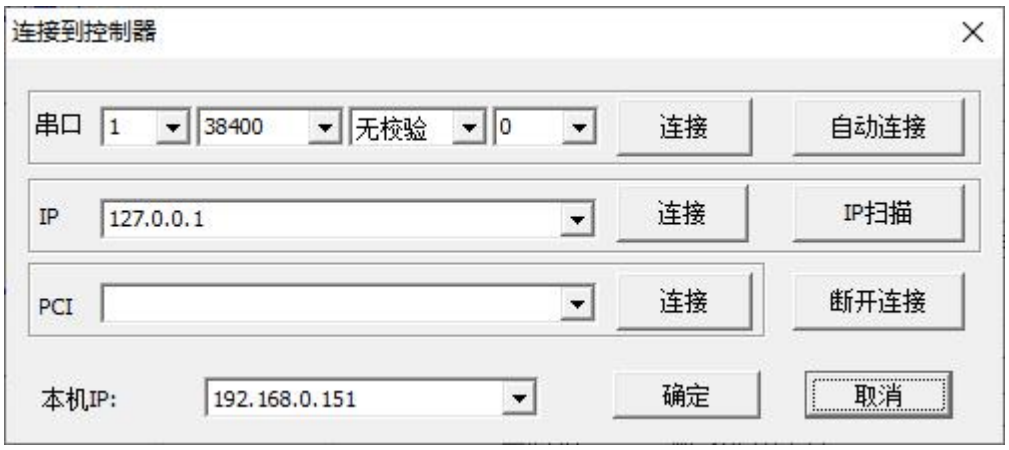
网口连接方式

IP 地址列表下拉选择时，会自动查找当前局域网可用的控制器 IP 地址。

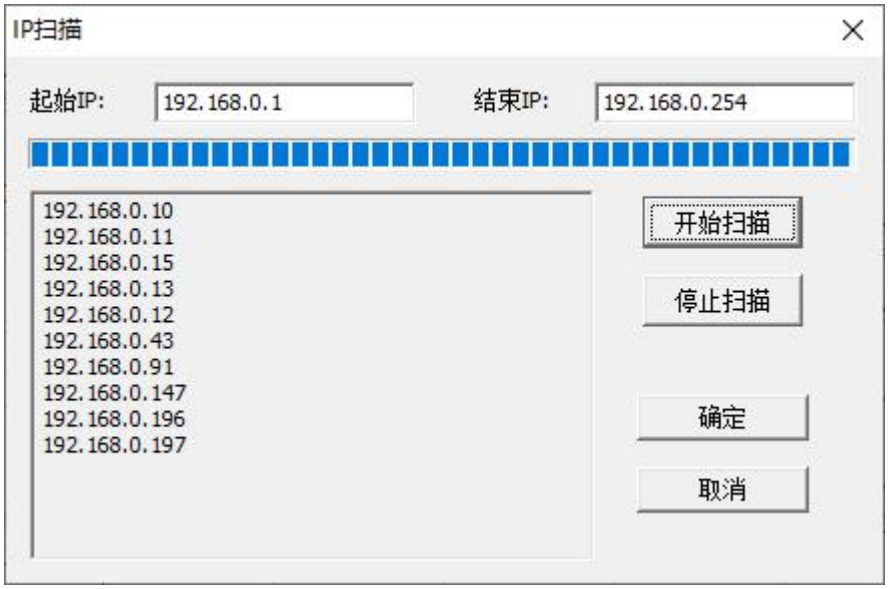
控制器出厂的缺省 IP 地址为 192.168.0.11，“连接到控制器”窗口能显示出本机 IP

地址，电脑需要设置 IP 地址与控制器 IP 处于同一网段才能连接，即四段的前三段要相同，最后一段不同才可通讯。

如下图通过“连接到控制器”窗口，可以快速查看本机 IP，请注意设置有线网卡与无线网卡各自的 IP。

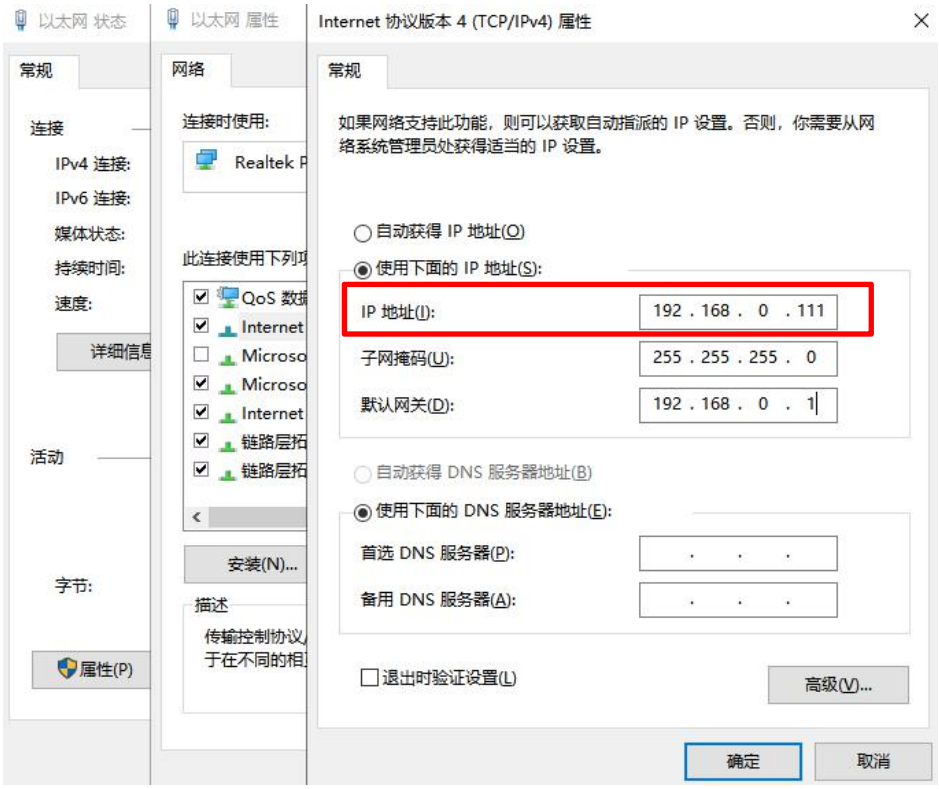


同一个网络有多个控制器的时候，可以采取 IP 扫描来查看。



PC 端 IP 地址修改方法：

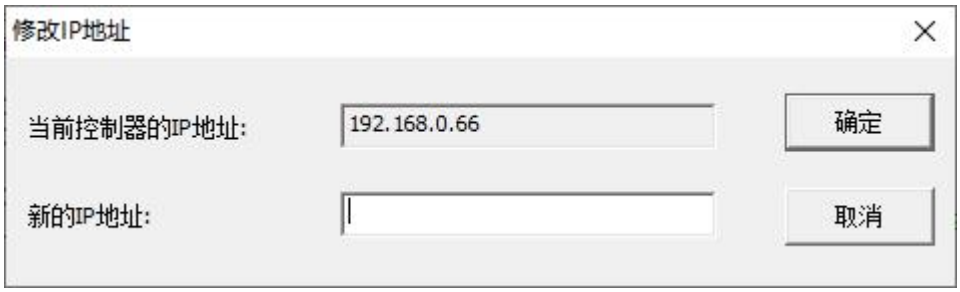
查看电脑本地 IP 协议版本 4 地址是否为 192.168.0.xxx，前三段与控制器一致，最后一段不能一样，控制器出厂默认 IP 192.168.0.11。如果 IP 地址的第三段不一样，则需要把对应的子网掩码改为 0。设置好之后再进行软件连接。



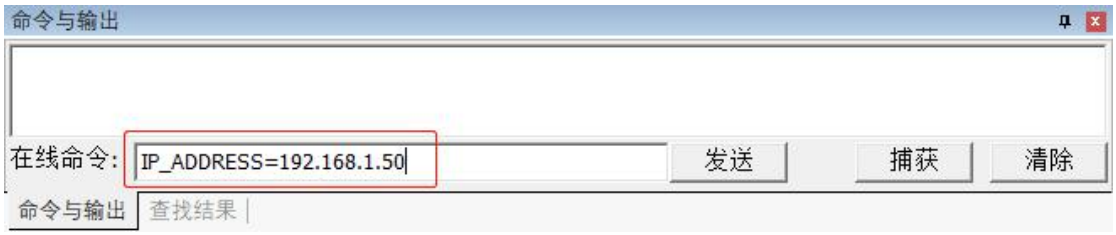
控制器 IP 地址修改方法：

如果控制器 IP 被修改，不处于 192.168.0.XXX 这个网段，此时只能先通过串口连接控制器，获取控制器 IP 地址，然后修改本机 IP 或控制器 IP 使二者处于同一网段。

修改控制器 IP 地址方法有多种，可点击菜单栏“控制器”-“修改 IP 地址”，弹出如下窗口，此时会显示当前控制器 IP，在窗口可直接输入新的 IP 地址。



或在 ZDevelop 菜单栏“控制器”-“控制器状态”查看或在线命令获取控制器 IP 地址，控制器 IP 用指令 IP_ADDRESS 修改。



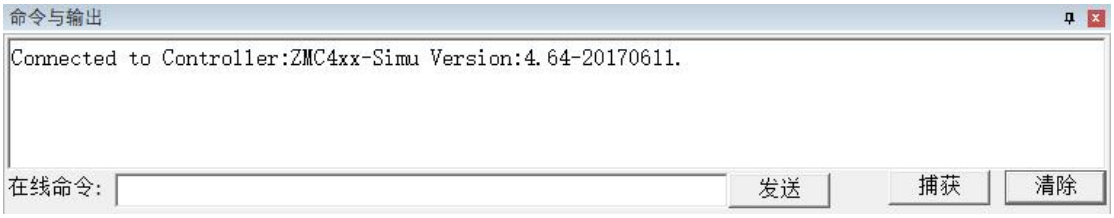
修改 IP 后，控制器与 ZDevelop 的连接会断开，此时再次选择新设置的 IP 地址连接即可。

1.2.3 连接仿真器

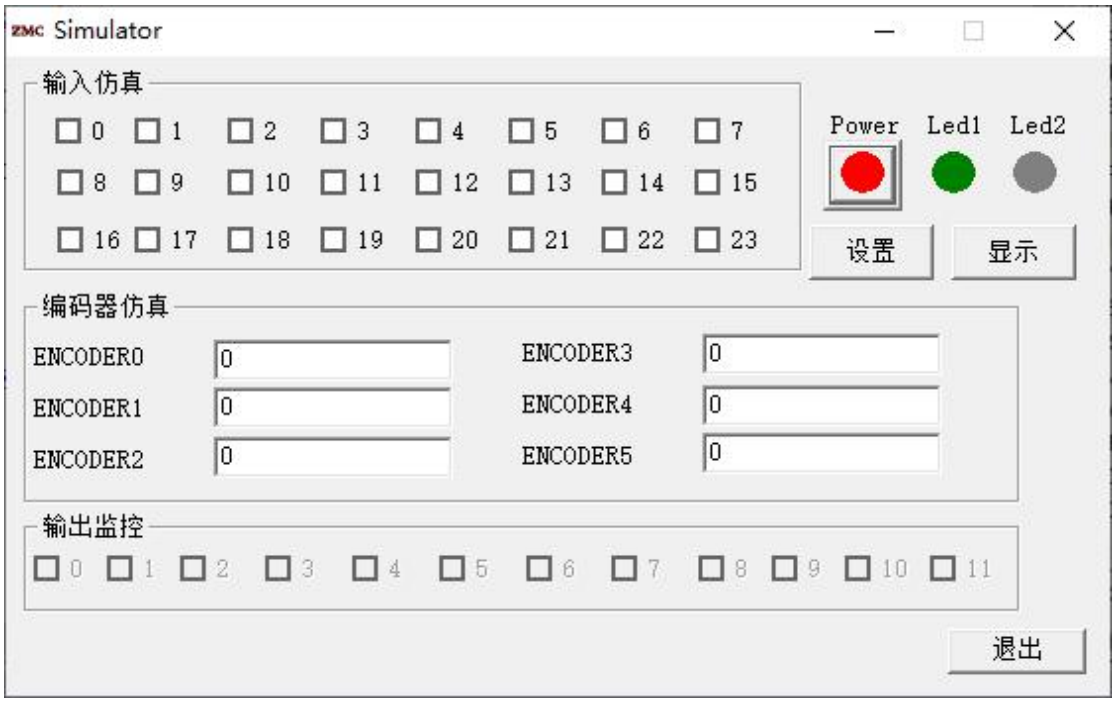
没有控制器时可以连接到仿真器调试程序。

编辑好程序后点击菜单栏“控制器”-“连接到仿真器”。

此时命令与输出打印如下连接信息。表示连接成功，仿真器对应的 IP 地址为 127.0.0.1。



连接成功弹出如下窗口，勾选输入口可模拟 IN 输入。

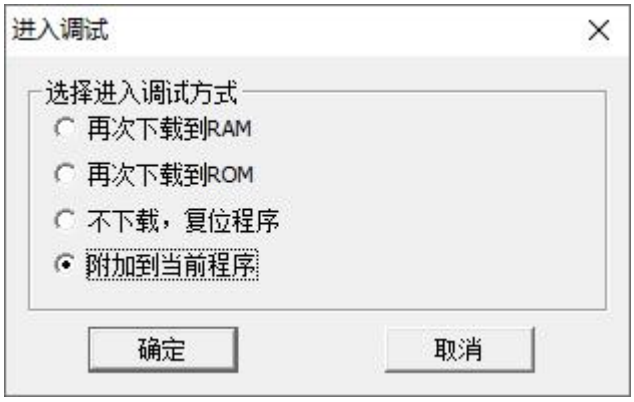


1.2.4 程序调试

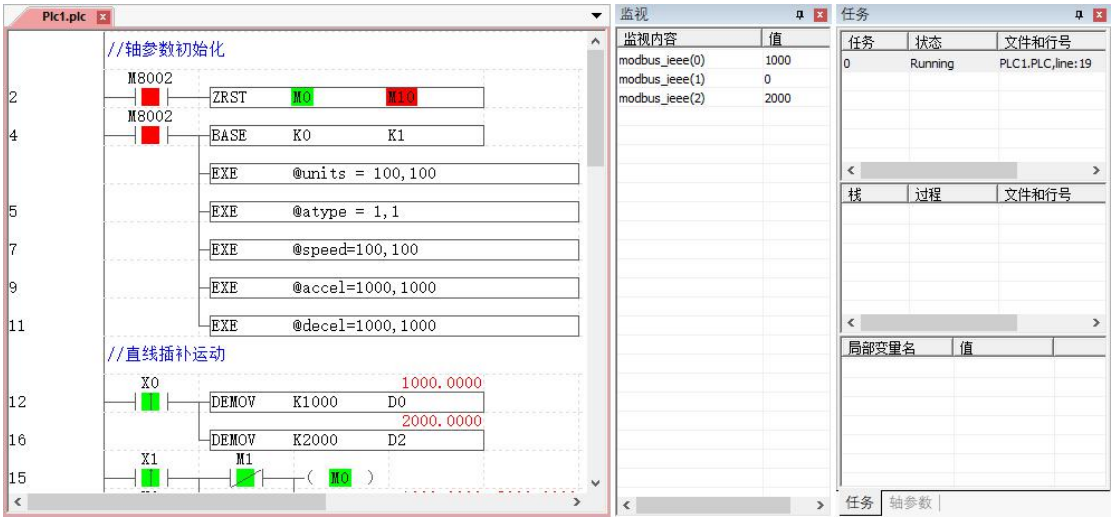
调试机器要注意安全！请务必在机器中设计有效的安全保护装置，并在软件中加入出错处理程序，否则所造成的损失，正运动技术公司没有义务或责任对此负责。

请保证 PC 端程序文件和控制器的程序文件一致，否则可能导致光标位置错误。调试只能在控制器 UNLOCK 状态时进行程序调试。

ZDevelop 连接控制器后，从菜单栏选择“调试”-“启动/停止调试”，弹出如下窗口，选择“再次下载到 RAM”表示程序再次下载到 RAM 运行，“再次下载到 ROM”表示程序再次下载到 ROM 运行，“不下载，复位程序”表示不下载程序，仅重新运行之前下载的程序，“附加到当前程序”表示此时程序不下载，仅在窗口显示目前的运行状态。



如下图，此时可以查看各任务运行情况、监视内容、子函数堆栈调用过程、子函数局部变量值。进入调试模式，位软元件会出现红绿色提示，绿色表示导通，红色表示关断，字软元件上方红色字体显示当前数值。

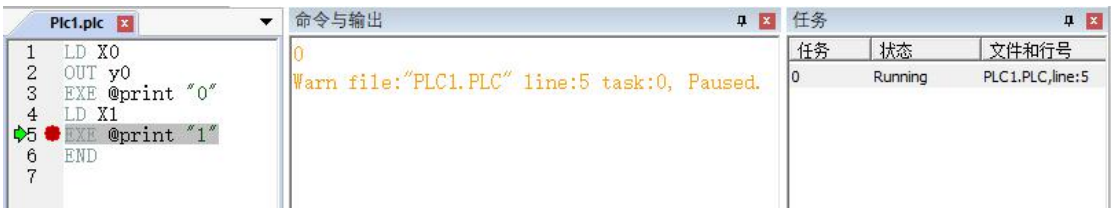


PLC 程序梯形图界面不支持断点调试，语句表界面支持断点调试。

断点调试可以查看程序运行的具体过程，主要用于判断程序逻辑错误。配合监视内容和轴参数变化情况可以查看程序每执行一步对寄存器、变量、数组等的影响。断点快捷键 F9 添加，或“调试”-“增删断点”，断点可以添加多个。

与 Basic 相同，断点调试时 PLC 从第一行扫描执行到断点处停止，断点行及之后的程序均不执行，调试完需要将断点全部清除。

如下图 PLC 指令表程序，上电运行并接通 X0，X1，仅打印出 0，不打印 1，因为该行设置了断点，程序停止在断点处后，断点行不执行，此时可以进行逐步调试，快捷键 F11，按一次程序向下执行一步。



如果断点是设置在循环中，那么下次循环运行到断点处时还是会停止程序。

通过断点列表窗口可以对当前所有的断点进行编辑。双击断点信息跳转到断点行，可以选择移除一个或多个断点，移除后点击确认生效。

程序调试完成后，需要清除所有断点才能下载到控制器运行，断点不清除就下载程序到控制器，命令与输出区域会打印如下警告信息：Warn file:"Basic1.BAS" line:11 task:0, Paused.

1.2.5 示波器使用

示波器属于程序调试与运行中极其重要的一个部分，ZDevelop2.61 以上版本支持示波器功能，在“视图”-示波器中打开。示波器必须先启动后触发才能成功采样，打开示波器设置好之后点击启动，可手动触发，也可在程序里加入“TRIGGER”指令自动触发示波器采样。

编号：选择需要采集信息的轴号、数字 IO 编号、模拟量 IO 编号、TABLE 编号、VR 编号、MODBUS 数据编号等

数据源：选择抓取的数据类型，下拉菜单选择

偏移：波形上下偏移

刻度：纵轴刻度

水平刻度：横轴刻度

XY 模式：勾选是切换成 XY 平面显示，不勾选时时间为横轴，纵轴为各数据轴显示。

设置：设置示波器相关参数，以下以 ZDevelop3.00.01 版本为例说明。

若要设置示波器参数，如轴编号、数据源以及启动设置，要先停止示波器再设置，点击“设置”按钮，弹出如下所示“示波器”设置窗口。

通道数：要采样的数据通道

深度：总共采样的数据次数，深度越大采样范围越大。

间隔：采样时间间隔，单位为系统周期，与控制器固件版本有关，一般默认 1ms，指令 SERVO_PERIOD 查看。一般来说，间隔越小，采样数据越准确，相同时间内数据量越大。

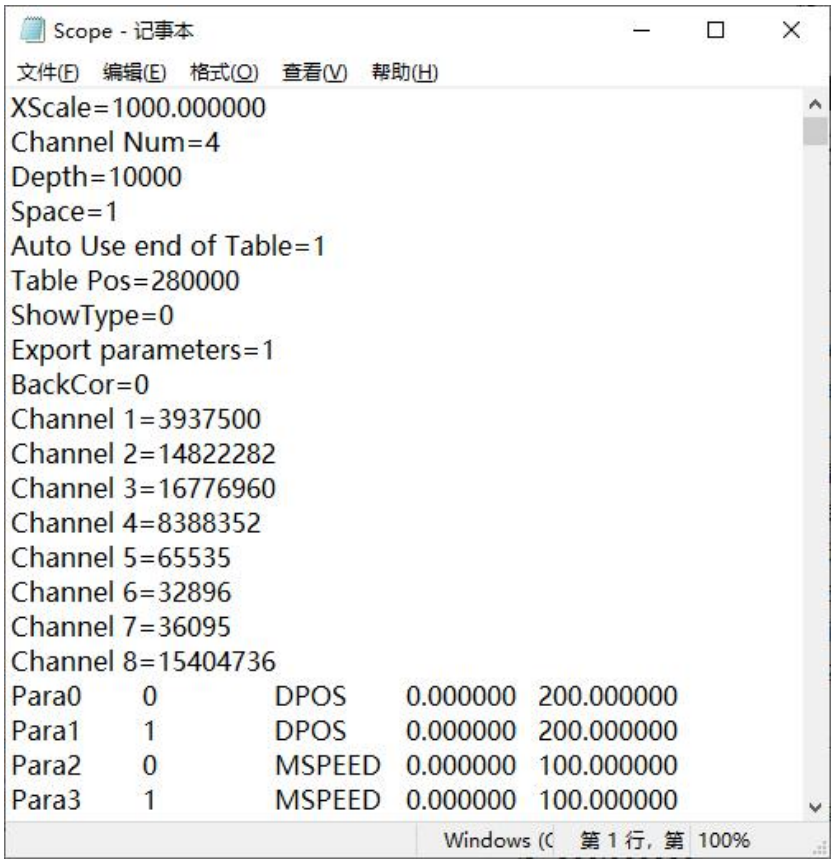
TABLE 位置：设置抓取数据存放的位置，一般默认自动使用 TABLE 数据末尾空间，也可以自己配置，但是设置时注意不要与程序使用的 TABLE 数据区域重合。

背景颜色/通道颜色：设置背景与每个通道波形对应的颜色。

显示类型：点和线段两种曲线类型可选。线段更容易发现异常点的数据显示。

导出参数显示数据更加完整，通道配置类型、采样时间等也一并记录下来导出。

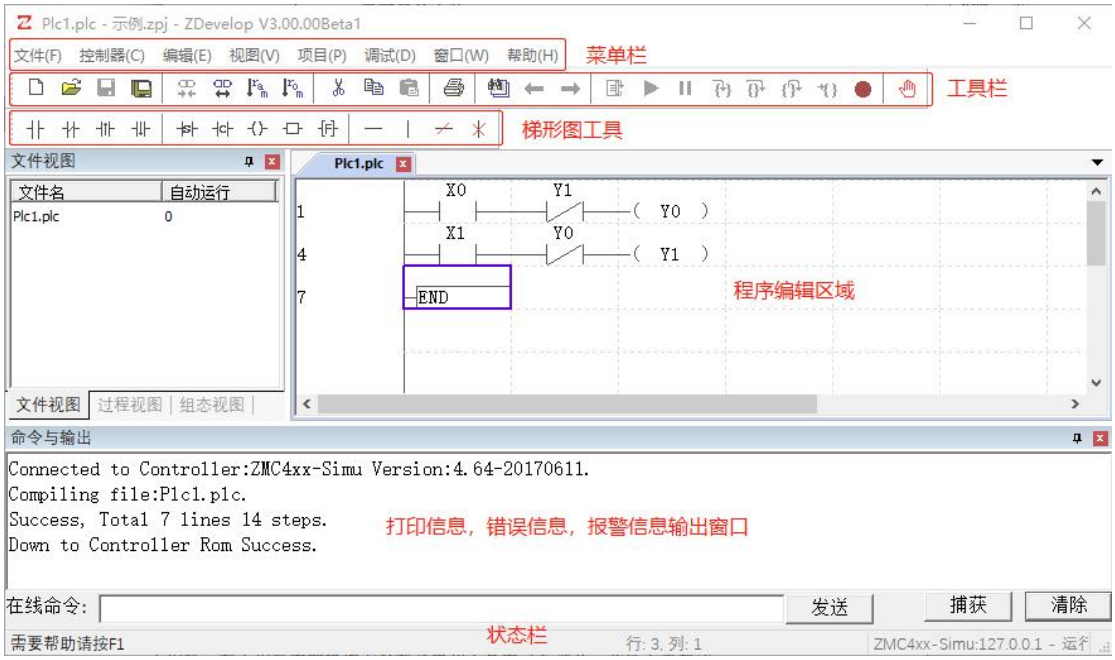
导出采样数据方法：现在设置里勾选“导出参数”，启动示波器采样，采样完成后点击“导出”，选择文件夹保存示波器数据。



1.3 PLC 编程软件说明

1.3.1 PLC 编程界面

界面基本组成如下，更多功能在菜单栏查找。



1.3.2 梯形图工具说明

点击梯形图工具快捷按钮可以快速选择对应功能输入。

符号	名称	动作特性	对应输入
┤├	常开触点	X：外部输入信号闭合，状态保持为 ON，断开为 OFF Y：输出线圈得电保持为 ON，失电为 OFF M：辅助继电器得电保持为 ON，失电为 OFF S：当 S 作为辅助继电器使用时，线圈得电保持为 ON，失电为 OFF C：当对应的计数器达到设定的数值，状态保持为 ON，否则为 OFF T：当对应的时间继电器线圈得电，且计时时间达到设定的数值，状态保持为 ON，否则为 OFF D：Dn.b，n 为寄存器编号，b 为位编号，当指定位的值为 1 时，线圈得电保持为 ON，指定位的值为 0 时失电为 OFF @：值为 0 线圈失电为 OFF，值不为 0 时得电保持为 ON	LD
┤/├	常闭触点	与常开触点相反，可用触点类型 X、Y、M、S、T、C、D、@	LDI

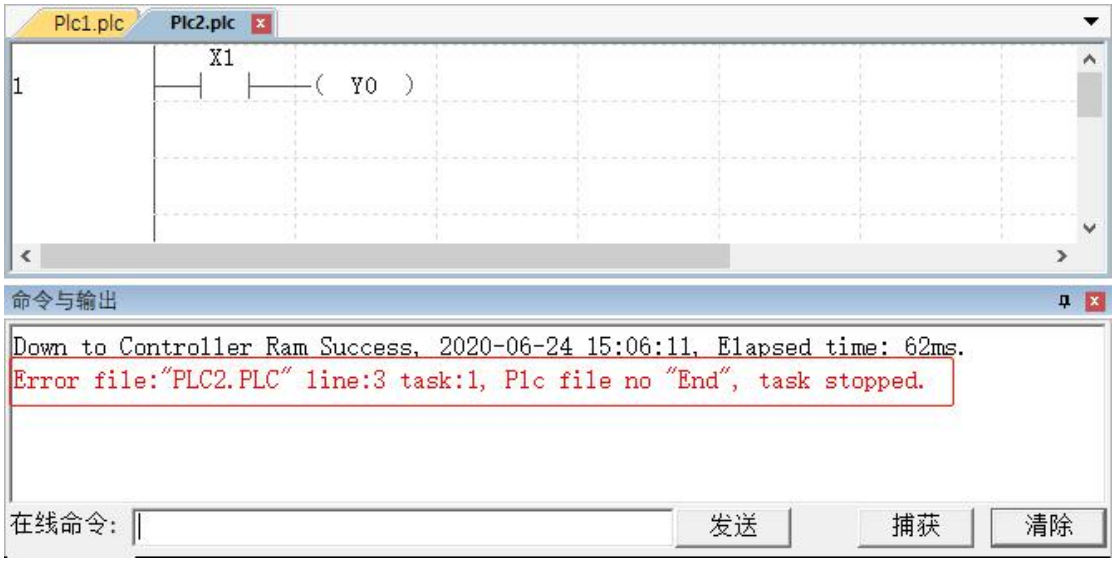
	取脉冲上升沿	当触点元件的状态由 OFF 变为 ON 时，触点导通一个扫描周期后变为 OFF	LDP
	取脉冲下降沿	当触点元件的状态由 ON 变为 OFF 时，触点导通一个扫描周期后变为 OFF	LDF
	步进触点	步进指令状态的转移	STL
	比较触点	对指令内的操作数数值进行比较，符合条件接通，不符合条件断开	LD*
	输出线圈	Y 或 M 的线圈得电时，其常开触点动作闭合，常闭触点动作断开，失电后恢复默认状态 T 线圈得电时开始计时，当计时时间到达设定值时，其常开触点动作闭合，常闭触点动作断开，失电后恢复默认状态 C 线圈每得电一次，计数值增加 1，当计数值到达设定值时，其常开触点动作闭合，常闭触点动作断开，接通后若要改变线圈状态需要使用指令对 C 复位 D：Dn.b，n 为寄存器编号，b 为位编号，满足输出条件将指定的位置 1，不满足条件保持为原来的数值	OUT
	选取指令	快速打开指令输入窗口，选取指令	任意指令
	子函数标记	快速定义子函数	LBL
	增加横线	给梯形图增加连接线	快捷键 F8
	增加竖线	给梯形图增加连接线	快捷键 F9
	删除横线	删除梯形图连接线	快捷键 Ctrl+F8
	删除竖线	删除梯形图连接线	快捷键 Ctrl+F9

1.3.3 编程方式

ZPLC 是 ZMotion 运动控制器支持的 PLC 语言，可以使用梯形图和语句表这两种编程方式。

PLC 指令均不分大小写。

程序结尾一定要包含 END 程序结束指令，否则报错，无法下载到控制器执行。

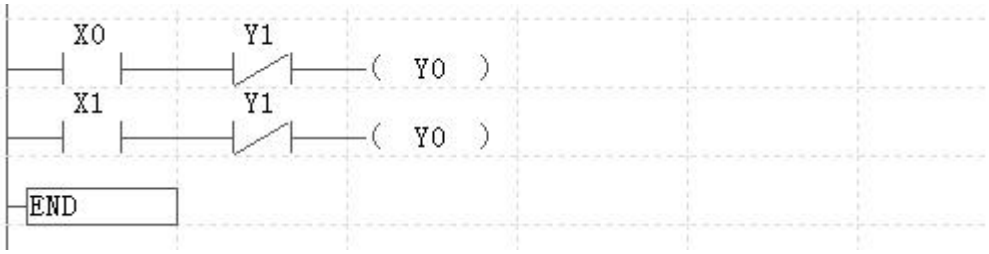


梯形图

梯形图编程方式就是使用顺序符号和软元件编号在编程界面上画出顺控梯形图的方式，由于顺控回路是通过触点符号和线圈符号来表现的，显示更加直观，程序的内容更易理解。

在梯形图显示状态下程序监控与调试更为便捷。

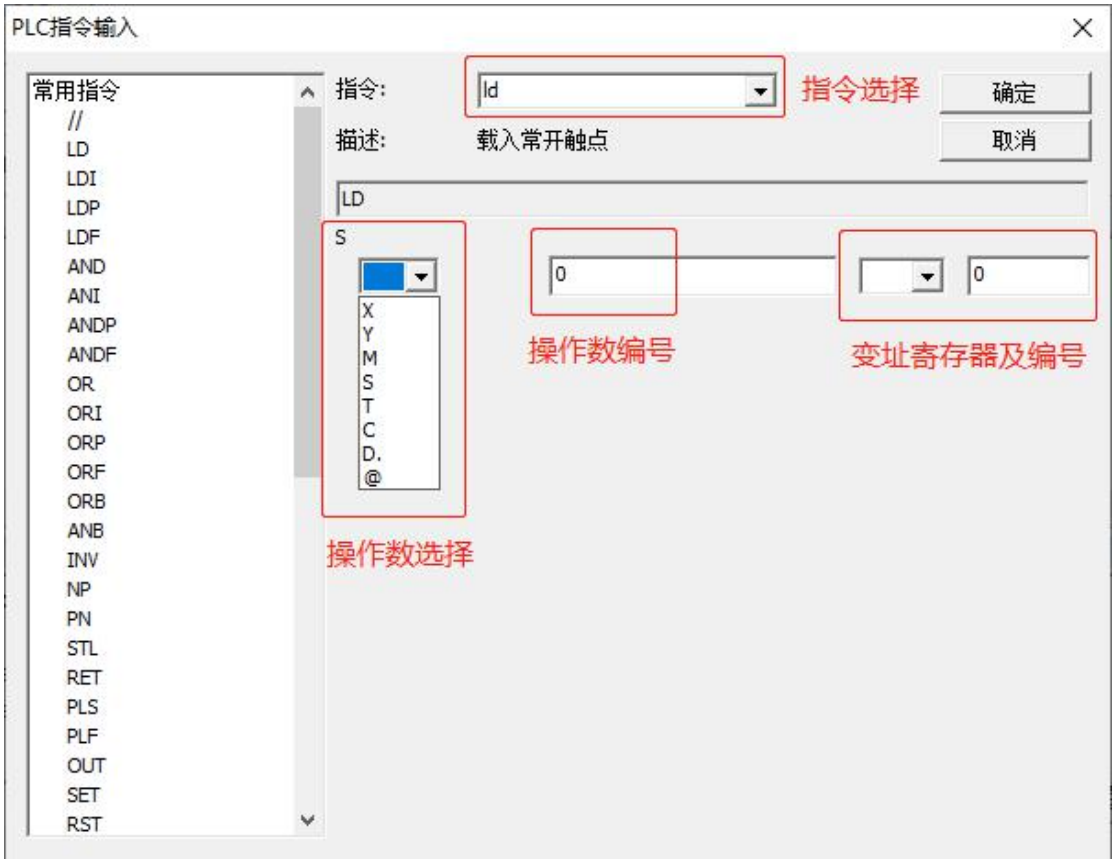
梯形图显示示例：



梯形图界面编程方法

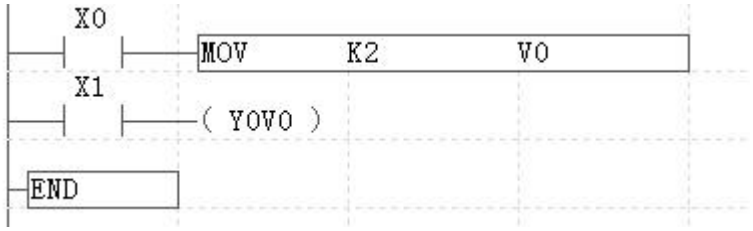
方式一：选中窗格，在梯形图工具栏里直接点击符号，弹出如下“PLC 指令输入窗口”，输入指令、操作数、操作数编号后确认。

指令输入可选择直接输入、在下拉列表选择、或在左侧常用指令列表选择。

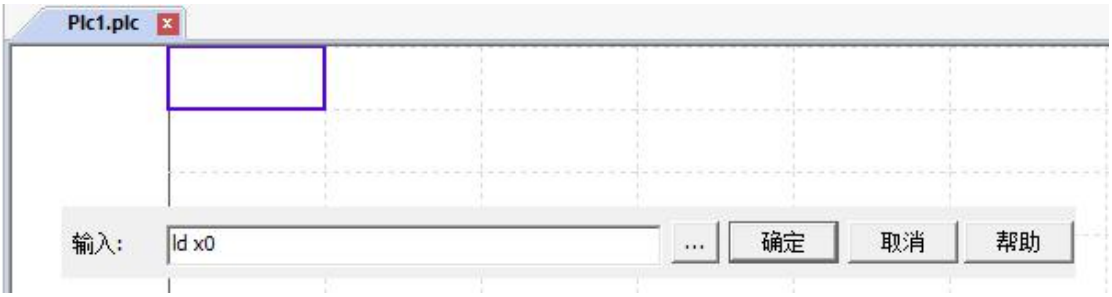


“PLC 指令输入”窗口变址寄存器可选 V、Z、LV，与软元件组合使用时，用于改变软元件编号或数值，可能变址的软元件有：X、Y、M、S、T、C、D、K。

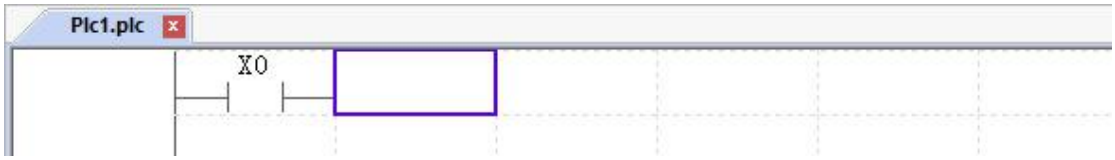
如下图，按下 X0，常数 2 传送到 V0，按下 X1，驱动 Y2 线圈，Y0V0=Y(0+V0)=Y2。



方式二：直接输入，在英文输入法下输入内容可自动弹出下图灰色输入框，直接输入指令、操作数、操作数编号后确认。



指令格式正确成功输入梯形图，光标自动后移一格，若指令格式错误会弹窗提示。

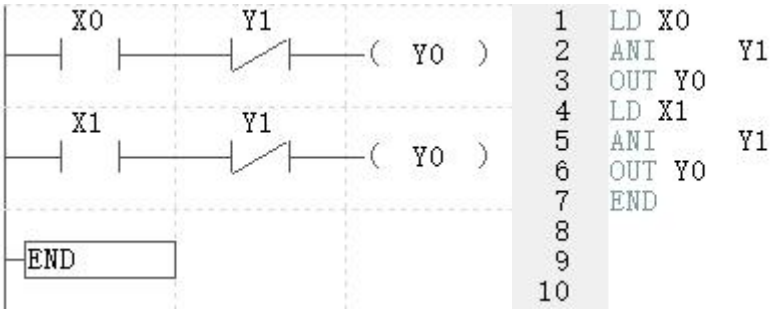


方式三：选择输入，在蓝色框上双击鼠标左键或单击回车，弹出灰色指令输入框，点击 ... 按钮，弹出“PLC 指令输入”窗口，参见方式一，输入指令、操作数、操作数编号后确认。

语句表

语句表编程方式就是通过 LD、AND、OUT 等 PLC 指令语言输入顺控指令的方式。该方式是顺控程序中基本的输入形态。

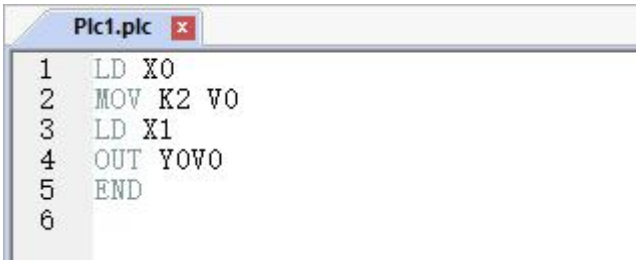
梯形图对应语句表显示示例：



两种编程方式可在 ZDevelop 软件菜单栏“编辑”里选择切换。

语句表界面编程

此编程方式适合熟悉 PLC 的有经验的编程人员，按指令语法格式一行输入一个指令。



编程界面差异

除了编程方式有所差异，编程界面的右键菜单的功能也不尽相同。

语句表界面特有“断点调试”功能。

梯形图界面下特有“批注”和“写入值”功能，批注支持给元件编辑批注，写入值窗口立即修改寄存器的值。

PLC 梯形图右键菜单

PLC 语句表右键菜单

断点调试

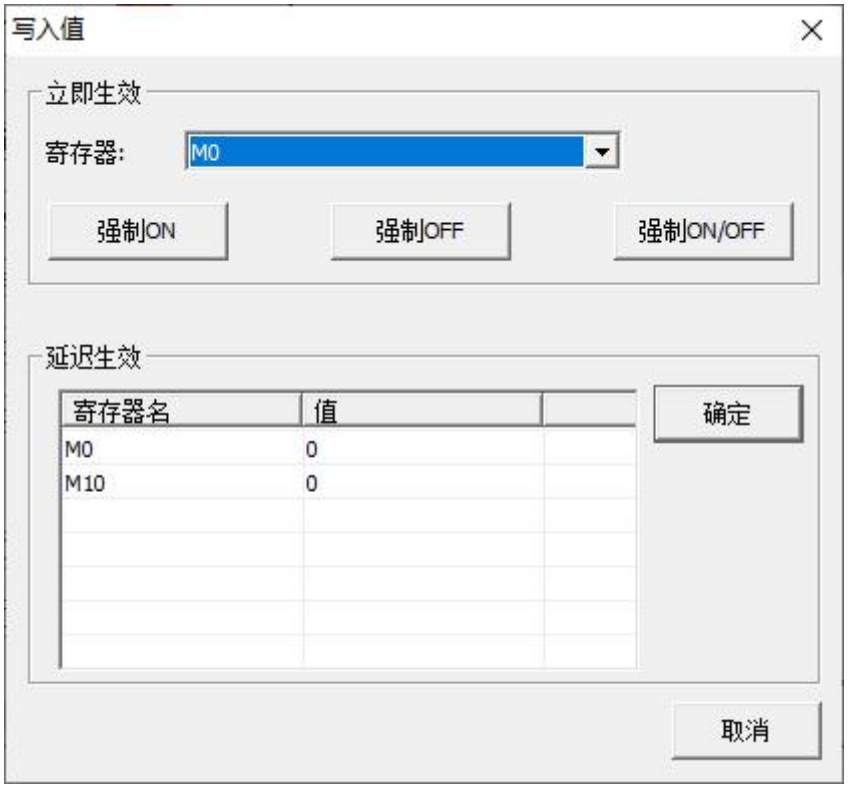
主要区别是梯形图不支持断点调试，而语句表界面下支持，断点调试方法参见程序调试章节。

批注编辑

给软元件添加批注，参见“注释”章节。

写入值

仅在梯形图界面调试状态下可以使用，选中窗格，点击右键打开写入值窗口，快速对位变量 M 或字节变量 D 的值进行编辑。



1.3.4 主要功能窗口

寄存器窗口

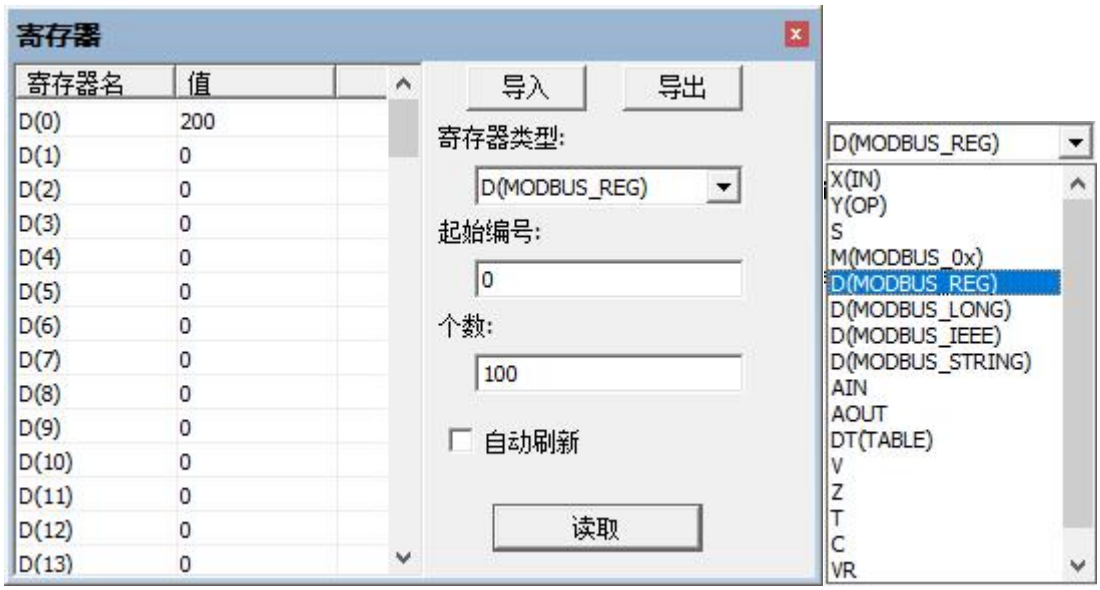
通过菜单栏“视图”-“寄存器”打开。

通过这个窗口可以查看控制器的寄存器、以及数值，可以选择查看不同类型的寄存器（支持 PLC 功能的控制器才支持此功能）。

使用方法：选择要读取的寄存器的类型，选取读取数据范围后点击“读取”即可在窗口显示出数据，注意个数选择不要超过寄存器空间，否则无法读出数据，双击寄存器的值可立即修改，点击“导入”/“导出”可快速上传/下载寄存器数据。

导出数据方便客户把自己关注的部分寄存器导出到文本保存。

导入数据方便客户把关注部分的已保存的数据直接更改到控制器内部。



寄存器类型：

X(IN)：输入继电器，由外部开关信号驱动，对应 IN

Y(OP)：输出继电器，能直接驱动外部负载，对应 OP

S：状态继电器，用于对工序步进控制

M：辅助继电器，不能直接驱动外部负载，对应 MODBUS_BIT

D：数据寄存器，存贮数据，对应 MODBUS_REG

AIN：模拟量输入

AOUT：模拟量输出

DT(TABLE)：浮点寄存器，长度 32 位，对应 TABLE

V：变址寄存器，长度 16 位

Z：变址寄存器，长度 16 位

T：定时器，单位为 1ms

C：计数器

VR：掉电保存寄存器，32 位浮点型

VR_INT：掉电保存寄存器，32 位整型

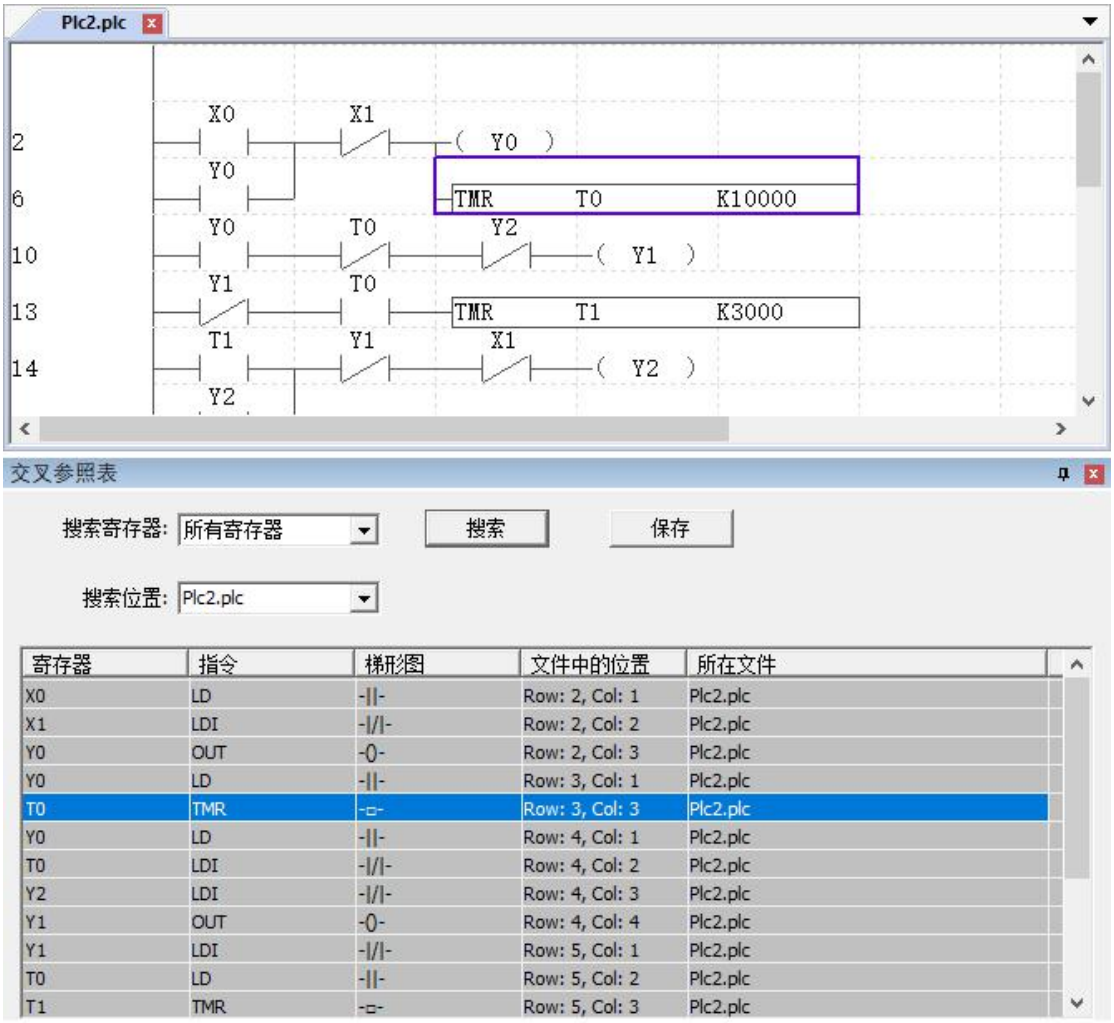
导出数据示例：

交叉参数表

适用于 PLC 编程方式，“交叉参照表”查看已使用的寄存器的具体信息，如下图，双击寄存器行快速定位到程序中的对应窗格位置。

搜索寄存器要带上寄存器编号。

搜索位置下拉框可选择在哪个 PLC 文件内搜索。

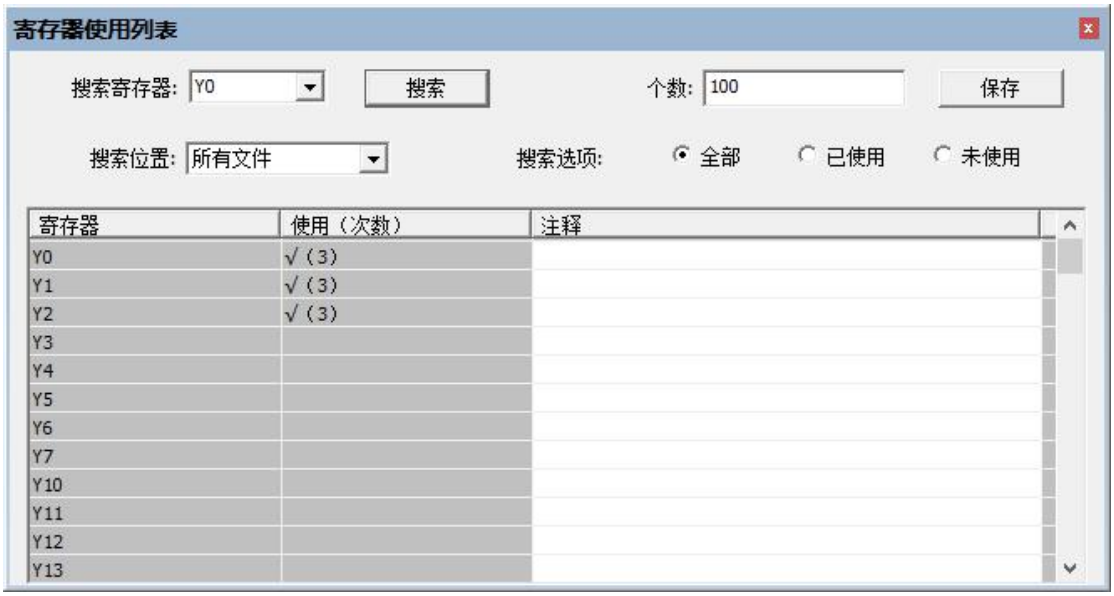


寄存器使用列表

适用于 PLC 编程方式“寄存器使用列表”便于查看寄存器的使用情况，查看哪些寄存器用了，哪些没用，在后续的程序中好规划。

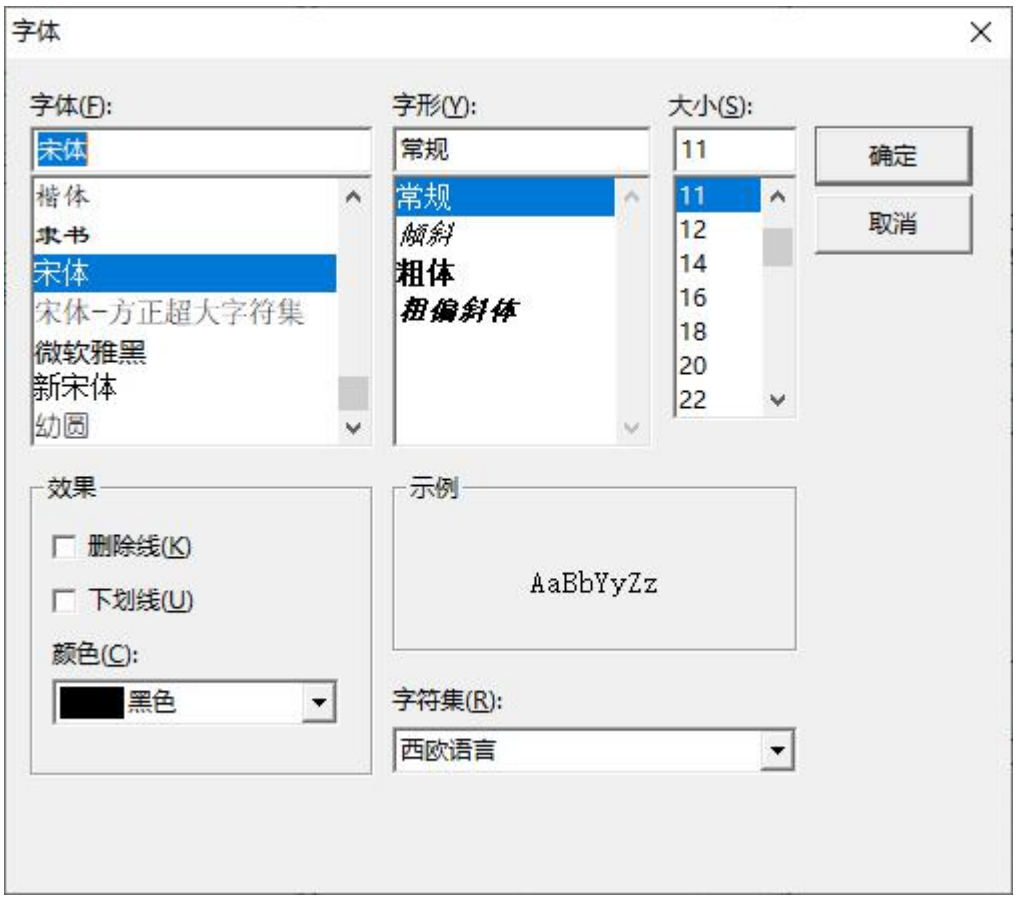
显示使用了的寄存器可在后方“注释”一栏双击添加注释，未使用的寄存器不支持添加注释。搜索的个数大于超过寄存器的编号。

双击寄存器灰色部分，可以立即打开交叉参照表，并自动搜索出点击的寄存器的信息。



字体

PLC 程序的字体大小可以在菜单栏的“视图”-“字体”中调整。



1.3.5 快捷操作说明

快捷键	名称	说明
Ctrl+A	全选	选择当前文件下的所有程序
Ctrl+Z	撤销	在当前编辑的文件中取消当前编辑的程序，返回上一步编辑的程序
Ctrl+Y	重做	将撤销的程序再复原回来
Del	删除	删除选取的程序
Ctrl+X	剪切	剪切选取的程序
Ctrl+C	复制	复制选取的程序
Ctrl+V	粘贴	粘贴复制的程序
Ctrl+F	查找	查找当前程序中的各类寄存器，软元件
Ctrl+H	替换	PLC 梯形图不支持替换功能，需要将梯形图转换成语句表将当前查找内容的各类寄存器和软元件替换成另外一个寄存器和软元件
Shift+F4	整个项目中查找	在整个项目中查找内容，并显示到“查找结果”窗口
Shift+Ins	插入一行	在当前选取的程序行上面增加一行空白的程序行。后面的程序自动往后面移动
Shift+Del	删除一行	删除当前选中的一行程序。选中多行时，一次操作只能删除第一行
Ctrl+F5	启动/停止调试	快速下载程序程序调试或退出调试状态
Ctrl+F2	设置/取消书签	在当前选取的程序行左侧加入一个书签标记 M。
F1	打开帮助手册	选中指令按下 F1 即可快速打开当前指令说明帮助文档

查找窗口

打开如下“查找”窗口，查找程序内容。

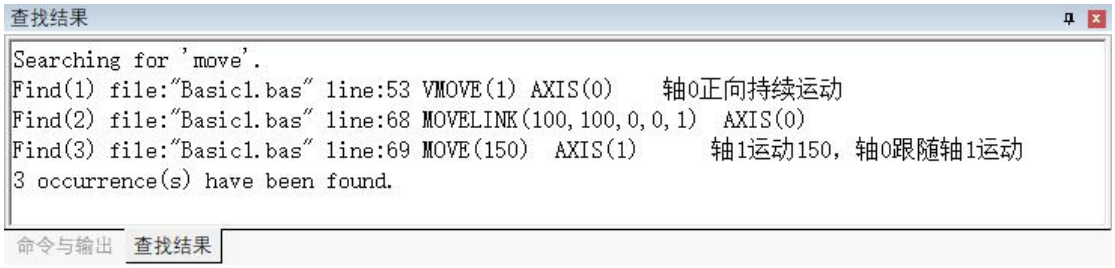
替换窗口

打开如下“替换”窗口，查找程序内容并替换为目标内容。

整个项目中查找

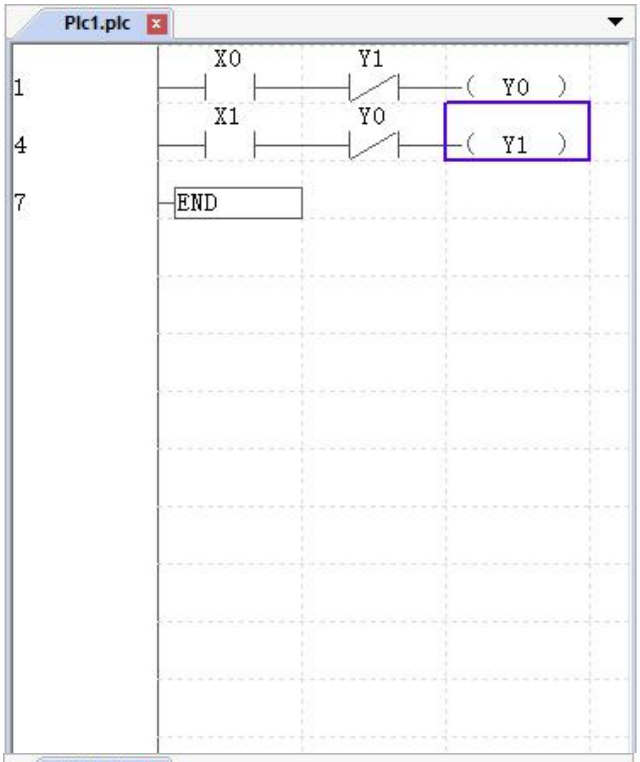
打开如下“整个项目中查找”窗口，在整个项目中查找内容，并将所有结果显示到“查找结果”窗口。可查找 basic 文件内容、PLC 软元件、寄存器等。

搜索结果如下图所示，双击内容可控制光标跳转到对应行。

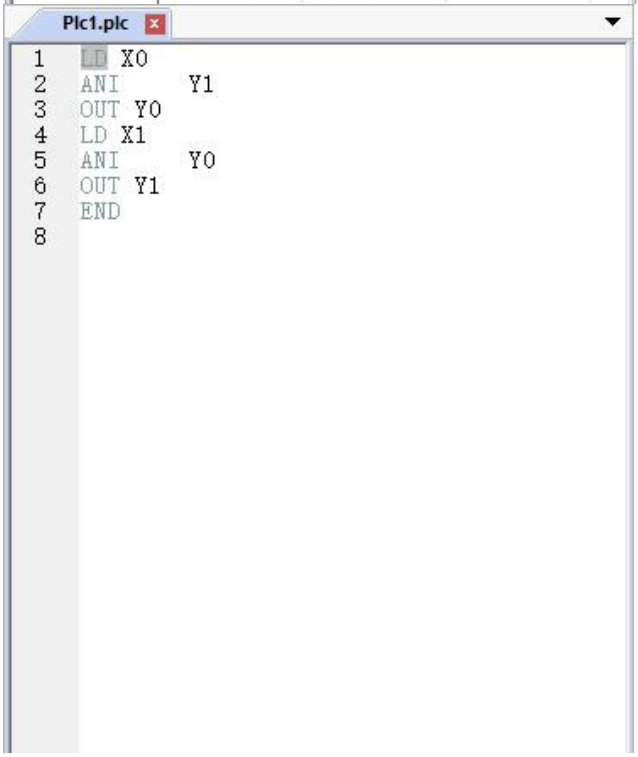


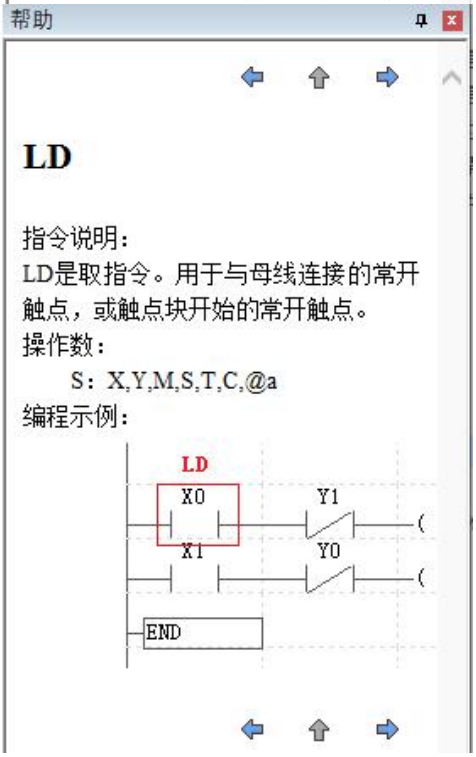
F1 快捷键

在梯形图和指令表中都可以使用 F1 快捷键打开指令帮助文档。
将光标置于帮助文档上，可以使用如下快捷键查看帮助内容。
HOME 键跳到当前页的行首。
END 键跳到当前页的行尾。
PAGEUP 和 PAGEDOWN 分别为上、下翻页。
Ctrl+滚轮为调整格子的缩放页面内容。


Plc1.plc
X0
Y1
Y0
1
X1
Y0
Y1
4
END
7


帮助
OUT
指令说明：
OUT是输出指令。是对软元件线圈驱动的指令
操作数：
S：Y,M,S,T,C
编程示例：当X0接通时，OUT指令输出Y0。
X0
X1
OUT
Y0
Y0


Plc1.plc
1 LD X0
2 ANI Y1
3 OUT Y0
4 LD X1
5 ANI Y0
6 OUT Y1
7 END
8


帮助
LD
指令说明：
LD是取指令。用于与母线连接的常开触点，或触点块开始的常开触点。
操作数：
S：X,Y,M,S,T,C,@a
编程示例：
LD
X0
Y1
X1
Y0
END

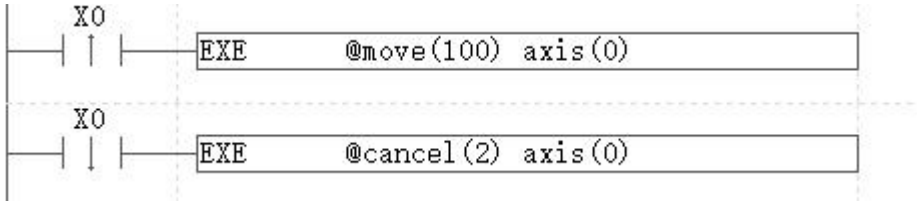
1.4 PLC 和 Basic 混合编程

1.4.1 PLC 调用 BASIC 指令

PLC 可以通过 EXE 指令或 EXEP 指令调用 Basic 标准指令。EXEP 指令是 EXE 指令的脉冲形式，仅在驱动输入由 OFF 变为 ON 后，才调用 Basic 标准指令。

语法格式如下：

“EXE @BASIC 指令”等价于“BASIC 指令”

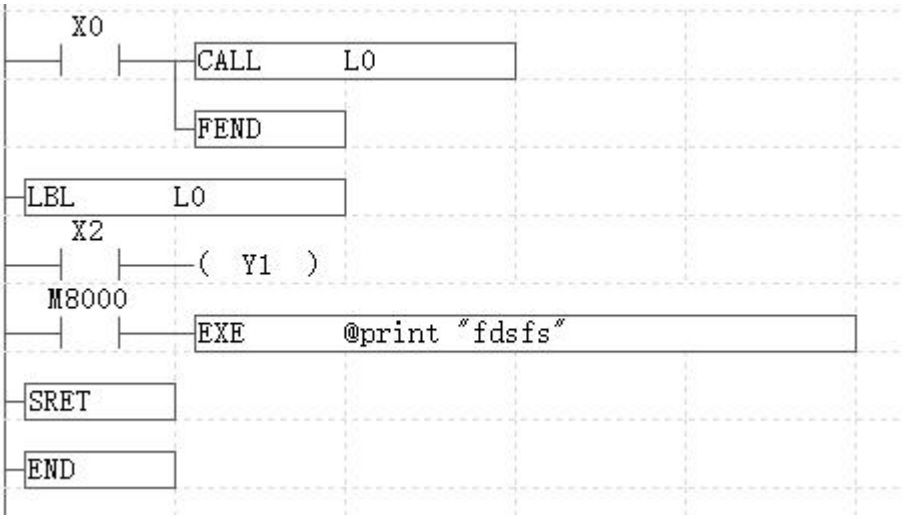


1.4.2 PLC 调用 BASIC 和 PLC 程序

PLC 使用 CALL 指令调用子程序。程序文件必须在同一项目文件（.ZPJ）内才可调用。

PLC 调用 PLC 子程序

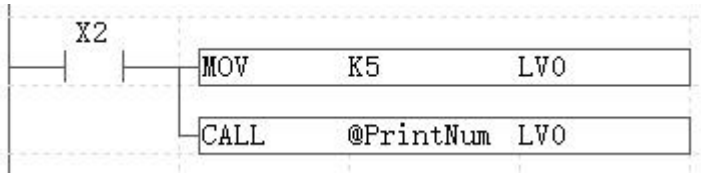
CALL 调用子程序后，跳转到对应 LBL 行执行，执行到 SRET 指令处返回。



PLC 调用 Basic 子程序

PLC 调用 Basic 子程序时，需要在 SUB 函数名前加上@，调用时可根据实际情况选择是否传入参数，如下图的参数 LV0。

PLC 梯形图程序：



被调用的 Basic 子程序：

```
GLOBAL SUB PrintNum(s1)
    print s1
END SUB
```

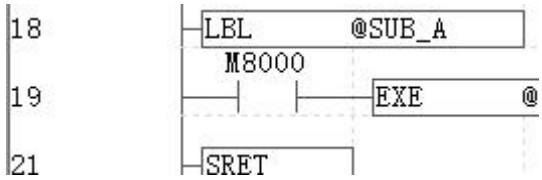
1.4.3 BASIC 调用 PLC 文件

程序文件必须在同一项目文件（.ZPJ）内才可调用。

Basic 启动 PLC 任务

Basic 里可使用语句“RUN“xxx.plc”,任务编号”来启动 PLC 文件任务。

```
'Basic开启任务调用PLC文件
run "Plc1.plc",6
'Basic开启任务调用PLC子函数
runtask 2,SUB_A
END
```



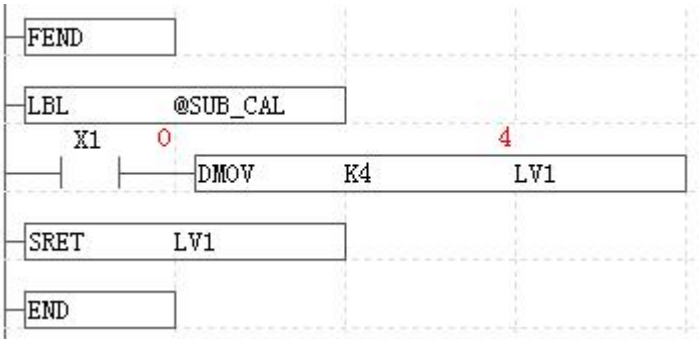
PLC 子程序 SRET 子程序返回指令可以带上返回值，返回值存储在 LV 局部寄存器中，返回值是浮点格式，调用者通过 RETURN 指令获取返回值。

Basic 调用 PLC 子程序

Basic 里使用“CALL SUB_FUNC”或“RUNTASK 任务号,SUB_FUNC”来调用 PLC 子程序 LBL。

如下图，Basic 程序执行到 CALL 语句后跳转到 PLC 程序的 LBL 子程序执行，执行完子程序 SRET 返回参数 LV1 保存在 RETURN 中。

```
while 1
    local b
    delay(1000)
    call SUB_CAL
    b = return
    trace "b = " b
wend
end
```



1.5 ZPLC 指令与三菱 PLC 指令的区别

1.5.1 三菱没有的指令

触点逻辑运算指令

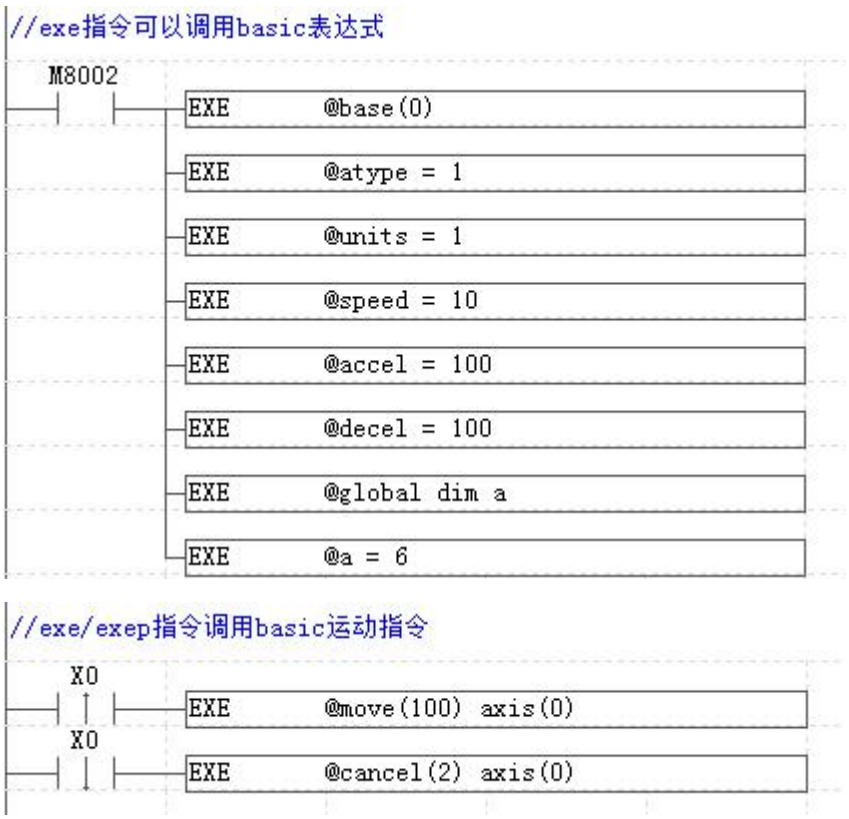
LD&、LD|、LD^、LDD&、LDD|、LDD^
AND&、AND|、AND^、ANDD&、ANDD |、ANDD ^
OR&、OR|、OR^、ORD&、ORD|、ORD^
指令用法，请查阅触点比较指令章节。

浮点数触点比较指令

FLD= 、FLD> 、FLD< 、FLD< > 、FLD<= 、FLD>=
FAND=、FAND>、FAND<、FAND< >、FAND<=、FAND>=
FOR= 、FOR> 、FOR< 、FOR< > 、FOR<= 、FOR>=
指令用法，请查阅触点比较指令章节。

调用 BASIC 指令

EXE、EXEP
指令用法：EXE @Basic 标准指令



详细指令用法，请查阅常用指令章节。

注释指令//

在 ZPLC 中可使用 ZDevelop 软件的批注功能，还可使用 ZPLC 梯形图的注释指令“//”。

编辑批注是对单个软元件的注释（这个保存在.zpj 文件里），注释指令“//”是在程序里面的空白行添加注释文段（保存在.plc 文件里）。

其它指令

BASE、MOVE、MOVEABS、VMOVE、DATUM。

指令用法，请查阅其他指令章节。

1.5.2 计数器指令的差异

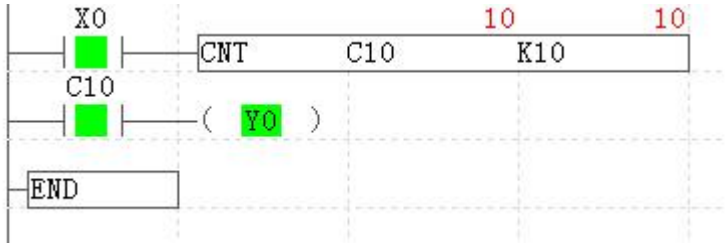
ZPLC

ZMC 控制器的 PLC 程序中使用计数器，可以使用计数器指令 CNT，也可以不使用计数器指令，用 OUT 指令输出。不同的是，使用计数器指令，在调试时显示个数，同时用绿色表示导通，红色表示断开；不用计数器指令，调试时不显示个数，仅用红绿色表示状态。

1) 使用计数器指令

下例中，计数器 C10 显示的值表示个数，当 X0 闭合一次，计数器 C10 计数为 1，当 X0 闭合 10 次时，计数器 C10 的计数值为 10，位软元件 C10 闭合，驱动 Y0 导通。后面 X0 每闭合一次，计数器 C10 继续加 1，位软元件 C10 始终闭合。

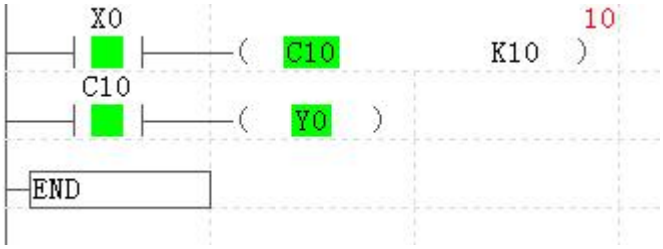
C10 一旦导通，断开 X0，位软元件 C10 仍保持导通，可使用 ZRST 指令复位计数器。



2) 不使用计数器指令

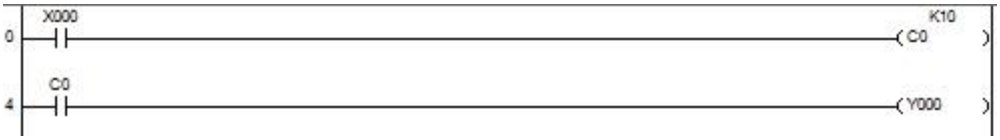
下例中，计数器 C10 不显示的当前计数值，当 X0 闭合 10 次后，位软元件 C10 闭合，无论后面 X0 闭合多少次，C10 依旧为闭合状态。

C10 一旦导通，断开 X0，位软元件 C10 仍保持导通，可使用 ZRST 指令复位计数器。



三菱 PLC

三菱 FX 系列 PLC 程序中，使用计数器不需要计数器指令。



1.5.3 定时器指令的差异

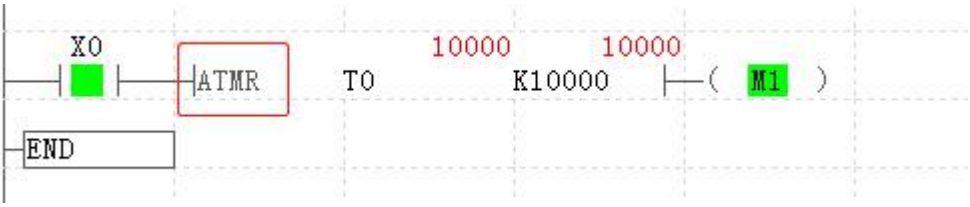
ZPLC

同计数器一样，ZMC 控制器在 PLC 程序中要使用定时器，可以使用定时器指令 TMR，也可以不使用定时器指令，用 OUT 指令输出。不同的是，使用计数器指令，在调试时显示当前毫秒数，同时用绿色表示导通，红色表示断开；不用计数器指令，调试时不显示毫秒数，仅用红绿色表示状态。

1) 使用定时器指令

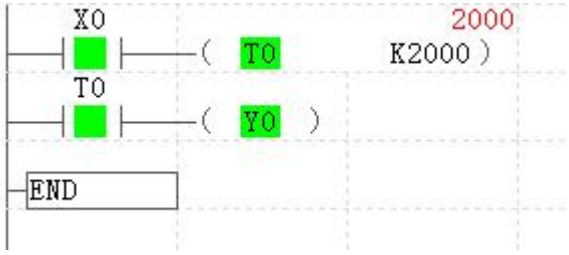
下例中，定时器 T0 显示的值表示毫秒数，当 X0 闭合时，T0 计时到 2000 毫秒停止计时，位软元件 T0 闭合，当 X0 断开时，T0 复位，值为 0。

在 ZPLC 中还有一个定时器触点指令 ATMR，相当于将 AND 与 TMR 指令组合，指令后方可以添加触点，指令特性与 TMR 相同。



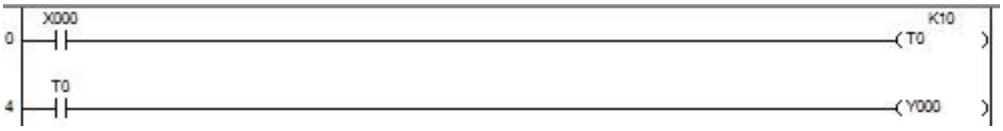
2) 不使用定时器指令

下例中，定时器 T0 不显示当前毫秒数，当 X0 闭合时，计时到 2000 毫秒，位软元件 T0 闭合，当 X0 断开时，T0 复位，值为 0。



三菱 PLC

三菱 FX 系列 PLC 程序中，使用定时器不需要定时器指令。



1.5.4 循环跳转指令的差异

在三菱 FX 系列 PLC 中，没有 LBL 指令，三菱在执行 CJ、CALL 等指令时，用指针编号(P)来标记跳转目标。

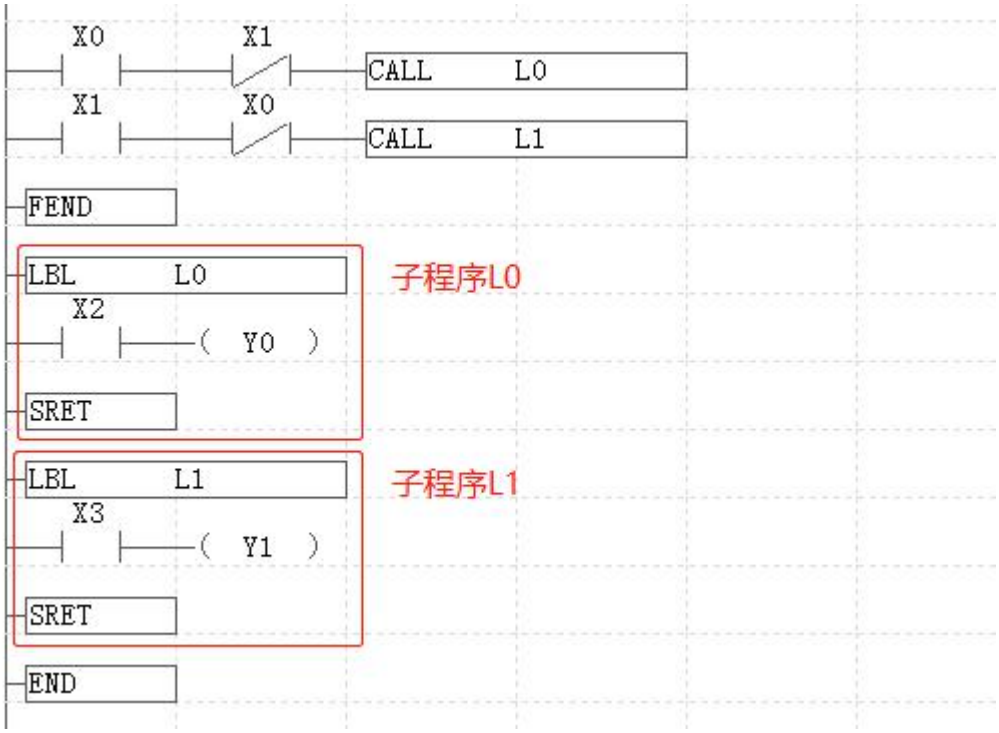
在 ZPLC 中 LBL 指令的作用是 BASIC 调用 PLC 子程序时，给 PLC 子程序标明名称，或在 PLC 调用 PLC 子程序时，给 PLC 子程序编号。在执行 CJ、CALL 等指令时，LBL 指令与 L 寄存器联合使用，定义跳转或调用条件，与三菱的指针编号(P)的功能类似。

ZPLC

CALL 指令用 L 寄存器来标明子程序编号，调用子程序没有指针(P)，而是用 LBL 指令定义 L 寄存器，来实现子程序的开启运行。

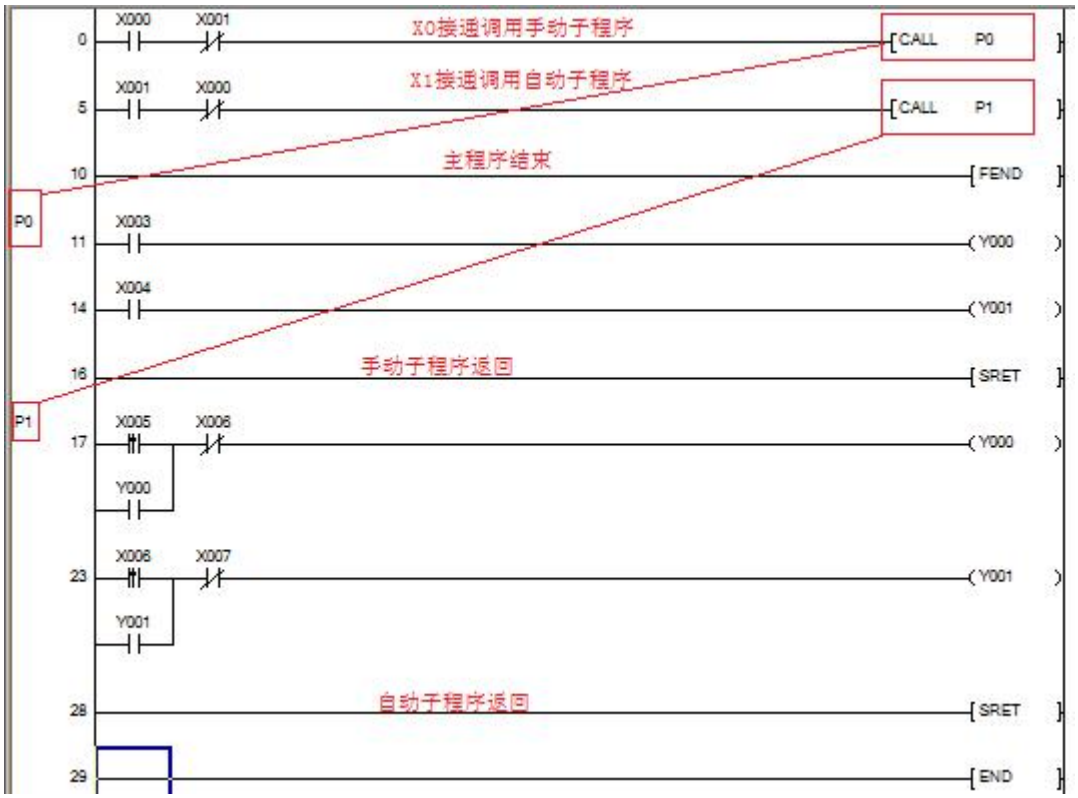
例：X0 接通调用子程序 L0，L0 执行完成由 SRET 指令返回到主程序的 CALL L0 调用语句的下一行继续扫描，扫描到 FEND 子程序结束指令后，返回第一行循环扫描。

X1 若不导通，则子程序 L1 不被调用，不执行，所以 X3 无法控制 Y1 的通断。



三菱 PLC

CALL 指令调用子程序，通过指针（P）标记跳转目标。



1.5.5 中断指令的差异

在三菱 FX 系列 PLC 中，没有 LBL 指令，三菱在执行 EI 中断时，用指针编号(I)来触发某个中断条件，会立刻暂停当前正在执行的用户程序（主程序或子程序），然后直接执行调用由中断条件产生的中断子程序。

在 ZPLC 中 LBL 指令的作用是调用中断函数。在开启中断指令 EI 后，与 L 寄存器联合使用，定义中断条件，当中断条件触发时，会立刻暂停当前正在执行的用户程序（主程序或子程序），然后直接执行调用由中断条件产生的中断子程序。

在三菱 PLC 中，EI 是直接与母线连接的，而 ZPLC 的 EI 前面必须加触点。

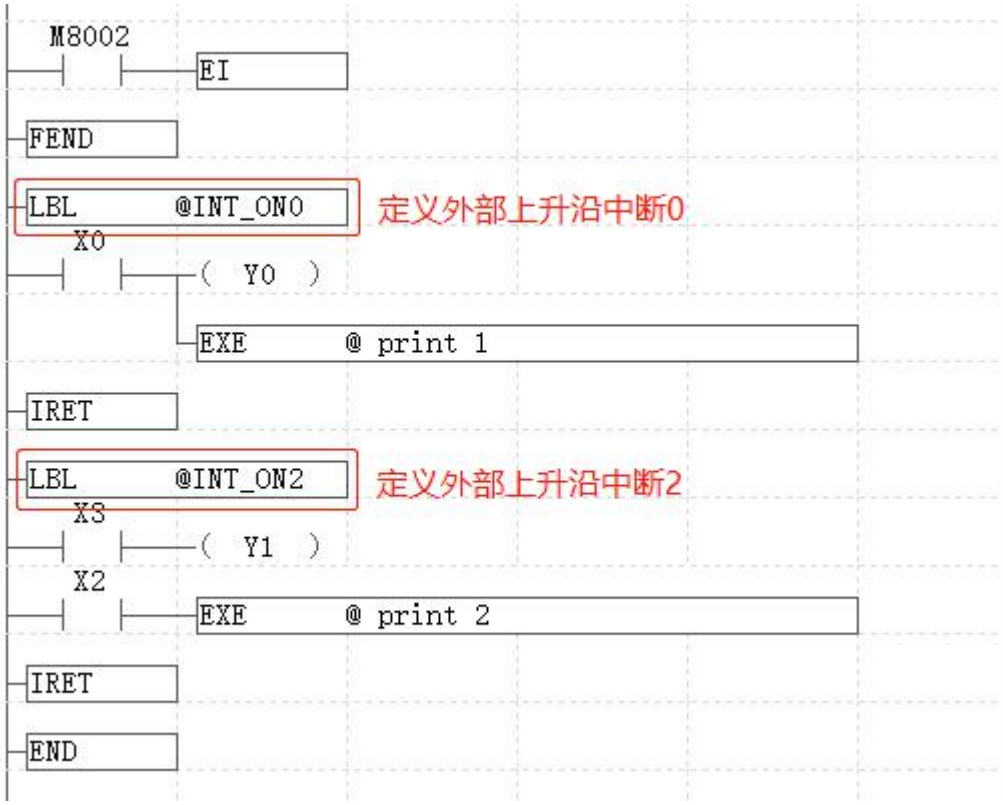
下面是 ZPLC 中断子程序输入中断指令和三菱 PLC 中断子程序输入中断的异。

ZPLC

EI 前面必须加触点，不能与母线直接相连，通过 LBL 指令调用中断函数，定义中断条件。

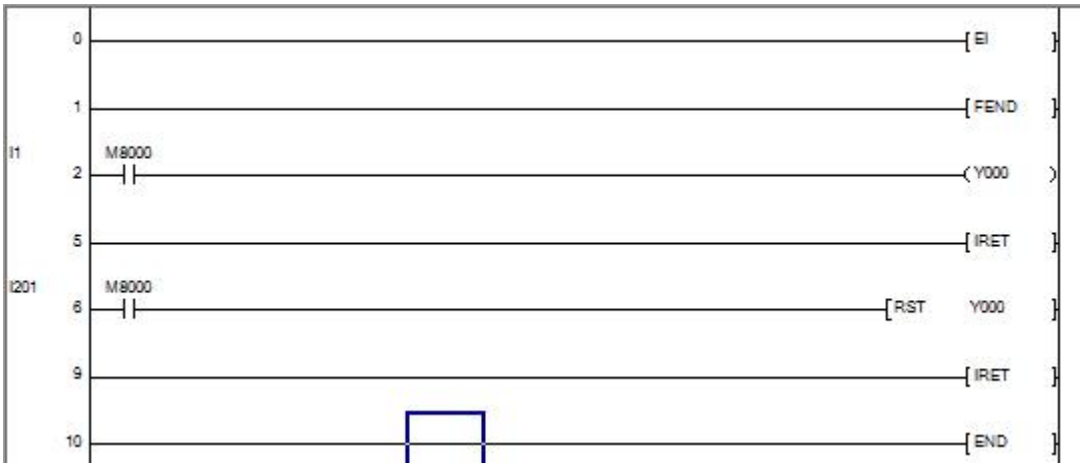
例：上电后 M8002 导通一个扫描周期，中断开启，闭合 X0，中断子程序执行一次后返回主程序，Y0 一直导通（中断子程序保留最后一个扫描周期的特性，导通后不管 X0 什么状态，Y0 保持），打印一次数字 1，外部上升沿中断 0 结束返回。

X2 未为闭合前，先闭合 X3，Y1 不导通，原因在于 X2 不闭合该中断程序不会触发执行。接通 X2，打印一次数字 2，外部上升沿中断 2 结束返回。



三菱 PLC

EI 直接与母线相连，通过 I 指针来定义中断条件。



第二章 寄存器（软元件）类型

软元件一览表：

软元件符号	软元件名称	编号	标记	处理数据类型
X	输入继电器	本地编号+扩展编号	八进制	布尔型
Y	输出继电器	本地编号+扩展编号	八进制	布尔型
M	辅助继电器	0-4095，掉电保持 2048-2175	十进制	布尔型
S	状态继电器	0-999，掉电保持 0-127	十进制	布尔型
D	数据寄存器	0-2047	十进制	16 位整型
C	计数器	0-127，掉电保持 100-107	十进制	16/32 位整型
T	定时器	0-127，掉电保持 100-103	十进制	16/32 位整型
V	变址寄存器	0-7	十进制	16 位整型
Z	变址寄存器	0-7	十进制	16 位整型
DT	浮点寄存器	0-(TSIZE-1)	十进制	32 位浮点数
LV	局部寄存器	0-7	十进制	32 位浮点数
L	Label 寄存器	0-63	十进制	/

计数器 C 和定时器 T 的数据类型与访问时使用的指令有关，通过 16 位指令访问时自动使用低 16 位，通过 32 位指令访问时使用 32 位。

PLC 与 Basic 相关寄存器对应关系：

<table>
<tr><th colspan="2">PLC</th><th colspan="2">BASIC</th></tr>
<tr><td rowspan="4">输入继电器 X</td><td>X0~X7</td><td rowspan="4">输入口 IN</td><td>IN(0)~IN(7)</td></tr>
<tr><td>X10~X17</td><td>IN(8)~IN(15)</td></tr>
<tr><td>X20~X27</td><td>IN(16)~IN(23)</td></tr>
<tr><td>X1770~X1777</td><td>IN(1016)~IN(1023)</td></tr>
<tr><td rowspan="4">输出继电器 Y</td><td>Y0~Y7</td><td rowspan="4">输出口 OP</td><td>OP(0)~OP(7)</td></tr>
<tr><td>Y10~Y17</td><td>OP(8)~OP(15)</td></tr>
<tr><td>Y20~Y27</td><td>OP(16)~OP(23)</td></tr>
<tr><td>Y1770~Y1777</td><td>OP(1016)~OP(1023)</td></tr>
<tr><td rowspan="3">辅助继电器 M</td><td>M0</td><td rowspan="3">MODBUS_BIT</td><td>MODBUS_BIT(0)</td></tr>
<tr><td>M1</td><td>MODBUS_BIT(1)</td></tr>
<tr><td>M1023</td><td>MODBUS_BIT(1023)</td></tr>
<tr><td rowspan="3">特殊继电器 D</td><td>D0</td><td rowspan="3">MODBUS_REG</td><td>MODBUS_REG(0)</td></tr>
<tr><td>D1</td><td>MODBUS_REG(1)</td></tr>
<tr><td>D1023</td><td>MODBUS_REG(1023)</td></tr>
<tr><td rowspan="3">浮点寄存器 DT</td><td>DT0</td><td rowspan="3">TABLE</td><td>TABLE(0)</td></tr>
<tr><td>DT1</td><td>TABLE(1)</td></tr>
<tr><td>DT1023</td><td>TABLE(1023)</td></tr>
<tr><td colspan="2">EXE @BASIC 指令</td><td colspan="2">BASIC 指令</td></tr>
</table>

2.1 Basic 寄存器@

此寄存器可以把 BASIC 表达式或执行语句嵌入到 PLC 中，在不同的指令中会自动进行格式变换以匹配。

举例：

```
LD   @(a+b)
EXEP   @MOVEABS(100)   AXIS(0)
```

单个@寄存器字符数不要超过 1000 字节。

2.2 输入继电器 X

在控制器中，输入端子是从外部的开关接收信号的窗口。不能用程序来驱动。

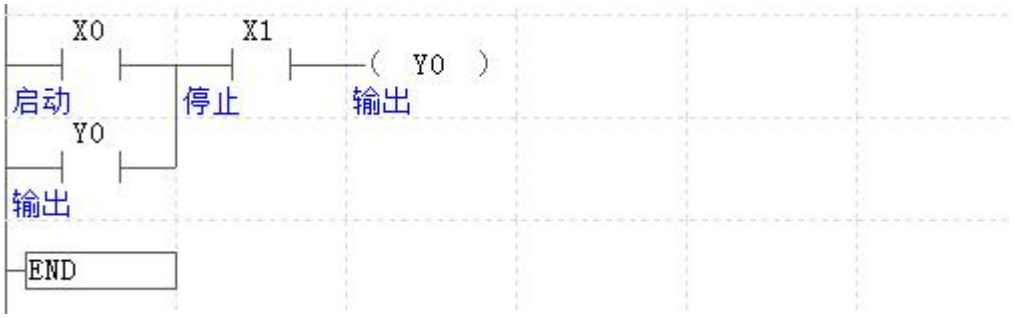
输入继电器 X 输入点（IN）可连接外部输入信号，如感应器、限位开关、按钮。

PLC 编程下，X 为 8 进制（X0~X7，X10~X17，…），而控制器的输入口 IN 为 10 进制，编写程序时要特别注意，做进制转换，比如，IN8 对应 X10，IN20 口对应 X24。

PLC 调用 Basic 程序时还是按 10 进制。

X 可用编号范围依据控制器自身 IO 点数加所连接的扩展模块 IO 点数确定。

例：在一段起保停程序中 X0 作启动信号，X1 作停止信号。



2.3 输出继电器 Y

在控制器中，输出端子是向外部的负载发出信号的窗口。不能用程序来驱动。

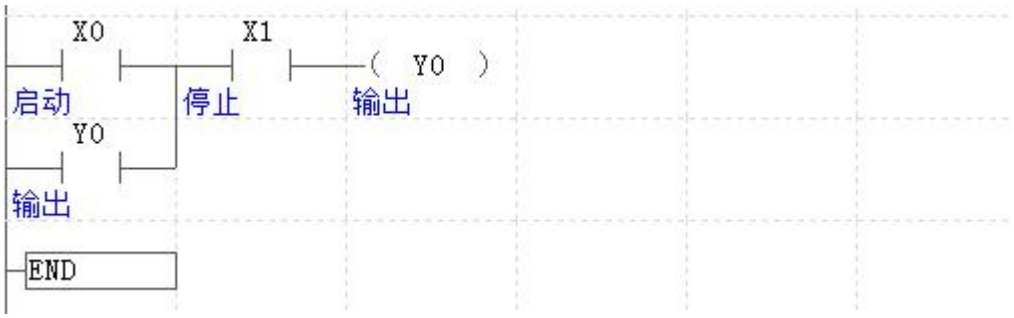
输出继电器 Y 输出点（OUT）输出给外部继电器、电磁阀。

PLC 编程下，Y 为 8 进制（Y0~Y7，Y10~Y17，…），而控制器的输出口 OUT 为 10 进制，编写程序时要特别注意，做进制转换，比如，OUT8 对应 Y10，OUT20 口对应 Y24。

PLC 调用 Basic 程序时还是按 10 进制。

Y 可用编号范围依据控制器自身 IO 点数加所连接的扩展模块 IO 点数确定。

例：在起保停程序中，Y0 作输出信号，也与 X0 自锁。



2.4 辅助继电器 M

辅助继电器的线圈与输出继电器一样，可由控制器内的软元件的触点来驱动。

辅助继电器不能直接驱动外部负载，外部负载必须要通过输出继电器来进行驱动。辅助继电器是控制器内部的软元件，没有实际的物理端子，不会坏。

此软元件对应 MODBUS_BIT（0x 寄存器）。

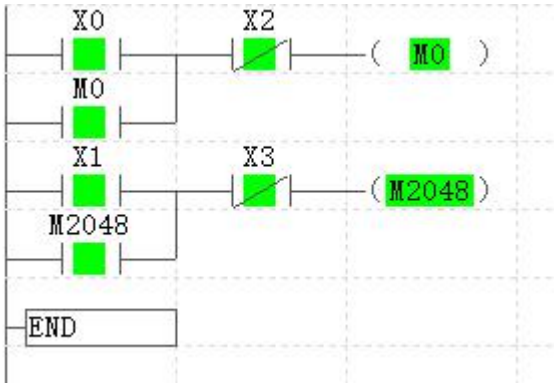
掉电保持型辅助继电器能记忆电源中断瞬间的状态，并在重新通电后再现其状态。

编号：0~4095

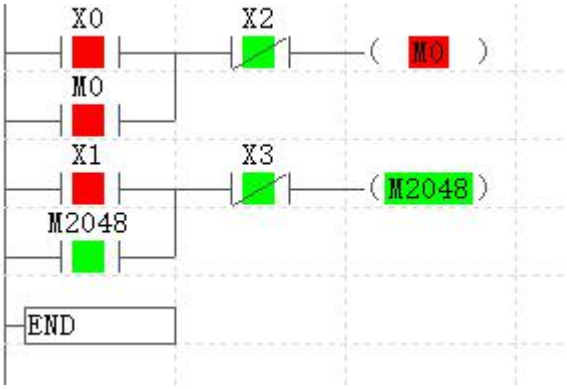
掉电保持：2048~2175

例：在这段程序中，M0 作辅助控制功能，与 X0 自锁和接通定时器。

例：M2048 为掉电保持型，闭合 X0 和 X1，M0 和 M2048 均导通。M0 与 X0 自锁，M2048 与 X1 自锁。



给控制器断电，再次上电后，M0 断开，M2048 仍导通。但再次运行时如果 X3 常闭触点为开路，此时 M2048 不动作。



2.5 状态继电器 S

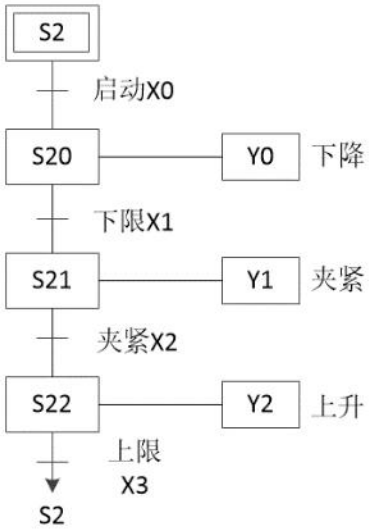
状态 S 是对工序步进控制简易编程的重要软元件，经常与步进梯形图指令 STL 结合使用，不用步进梯形图指令时，状态继电器 S 可作为辅助继电器 M 在程序中使用。

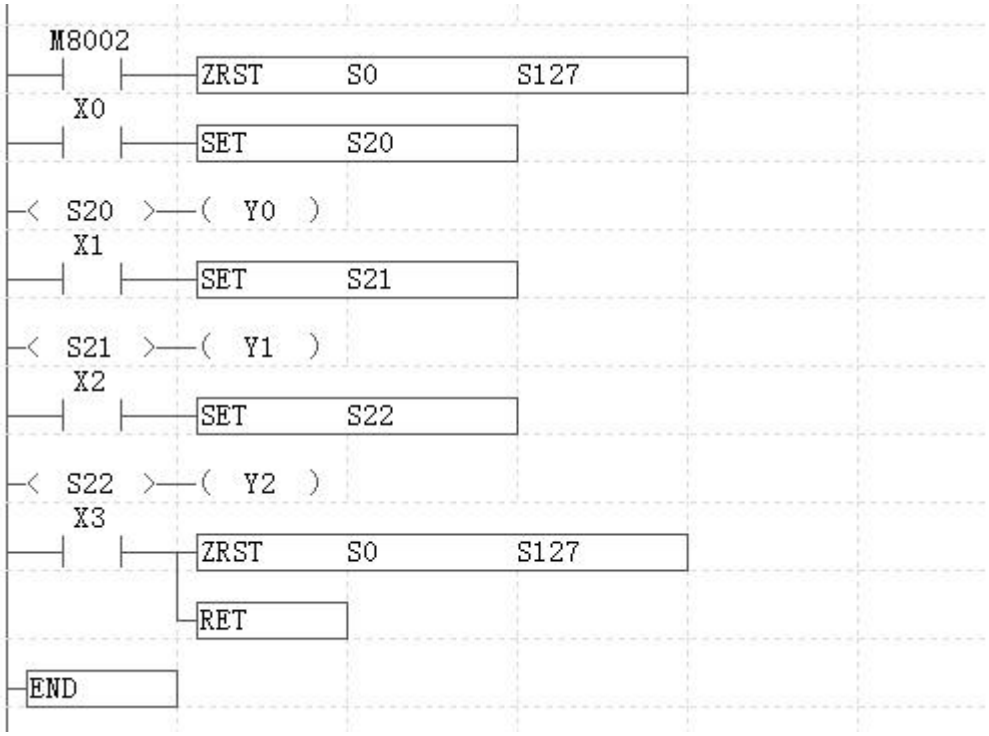
一个程序可以包含多个步进流程，步进编号 S 不能重复，每一个步进流程结束时，一定要写入 RET 步进结束指令。

编号：0~999

掉电保持：0~127

例：下图中，当起动信号 X0 有效时，机械手下降，到下降限位 X1 开始夹紧工件，夹紧到位信号 X2 为 ON 时，机械手上升到上限 X3 则停止。整个过程可分为三步，每一步都用一个状态继电器 S20、S21、S22 记录。每个状态器都有各自的置位和复位信号(如 S21 由 X1 置位，X2 复位)，并且要做各自的操作（驱动 Y0、Y1、Y2）。从起动开始由上之下随着状态动作的转移，下一状态动作，则上面状态自动恢复原状。必须为顺序执行。





初始化过程上电执行一次，将 S20、S21、S22 均置零。

X0 闭合，S20 导通，输出 Y0。

X1 闭合，转移到步进点 S21，S21 导通，输出 Y1，S20 断开，Y0 断开。

X2 闭合，转移到步进点 S22，S22 导通，输出 Y2，S21 断开，Y1 断开。

X3 闭合，将 S0-S127 置零，RET 指令返回，等待执行下一步进流程。

2.6 数据寄存器 D

控制器中的数据寄存器用于存储模拟量控制、位置量控制、数据 I/O 所需的数据及工作参数。每一个数据寄存器都是 16 位（最高位为符号位），可以用两个数据寄存器合并起来存放 32 位数据（最高位为符号位）。

在 32 位指令中使用数据寄存器时，会使用两个连续的 MODBUS_REG 空间来存储 32 位数据，使用时注意编号，以免数据发生覆盖导致错误。

此寄存器对应 MODBUS_REG，（4x 寄存器）,通过 D.可以按位访问。

16 位数据，占用一个 MODBUS_REG 空间，数据范围-32768 ~ +32767。

Dn+1															
b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0

32 位数据，占用连续两个 MODBUS_REG 空间，数据范围-2147483648 - +2147483647。

Dn+1																Dn															
b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0	b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0

编号：0~2047

根据 SETCOM 的配置来确定映射到 VR 的方式(参见 ZBASIC 手册 SETCOM 指令)。

当需要掉电保持时，推荐 variable=3

variable 参数是全局的设置，所有的端口共一个。

variable 值	描述
0	VR，此时一个 VR 映射到一个 MODBUS_REG
1	TABLE，此时一个 TABLE 数据映射到一个 MODBUS_REG（不推荐）
2（缺省）	系统 MODBUS 寄存器，此时 VR 与 MODBUS 寄存器是两片独立区间
3	VR_INT 模式，此时一个 VR_INT 映射到两个 MODBUS_REG

数据寄存器的数值读出与写入一般采用应用指令，此外，也可以从数据存取单元（显示器）与编程设备直接读出/写入。

2.7 字软元件的位指定（Dn.b）

Dn.b 指定字软元件的位，可以将其作为位数据使用。

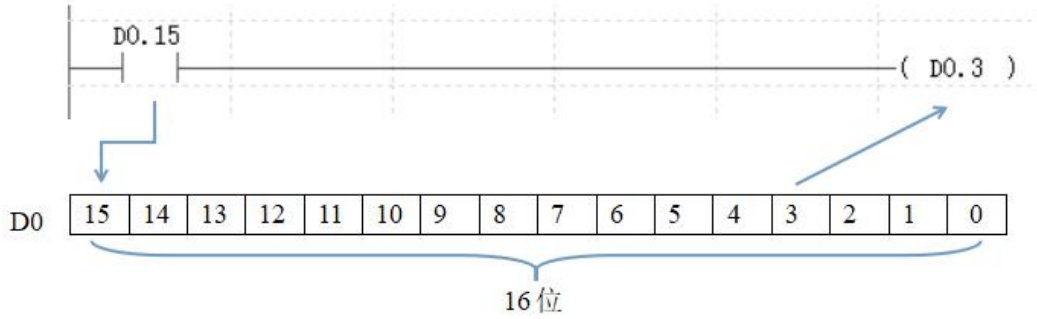
Dn.b 指定字软元件的位时，请使用字软元件编号 n 和位编号 b（16 进制数）进行设定。（例如：D1.2····表示数据寄存器 D1 的第 2 位）

在软元件编号，位编号中不能执行变址修正。

对象的字软元件：数据寄存器或特殊数据寄存器

字软元件编号 n：0~8000

位编号 b：0~15（16 进制）



2.8 位的位数指定（KnXn）

仅处理 ON/OFF 信息的软元件被称为位软元件。位软元件包括 X,Y,M,S 等。

T,C,D 等处理数值的软元件被称为字软元件。字软元件包括 K,KnX,KnY KnM,KnS,T,C,D,V,Z 等。

即使是位软元件，通过组合使用后也可以处理数值，这种情况下，以位数 Kn 和起始软元件的编号的组合来显示，Kn 中的 n 为 1-8 的整数，起始位为 Xn。

位数为 4 位单位的 K1~K4（16 位数据），K1~K8（32 位数据），例如：

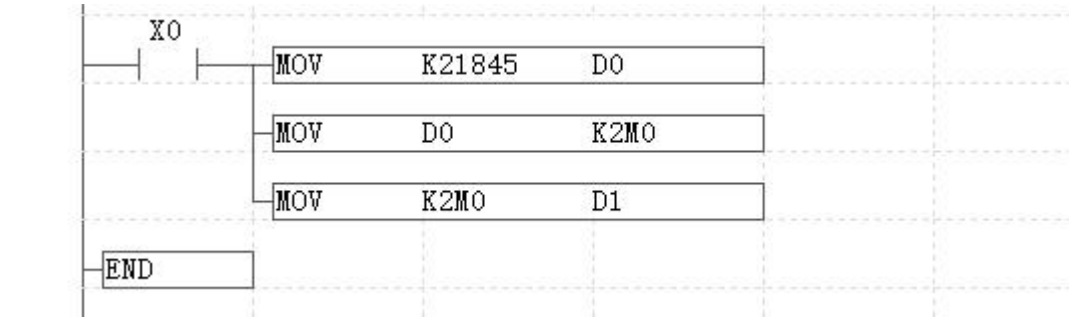
K1M0 表示位软元件 M0、M1、M2、M3 共四个位组合使用；

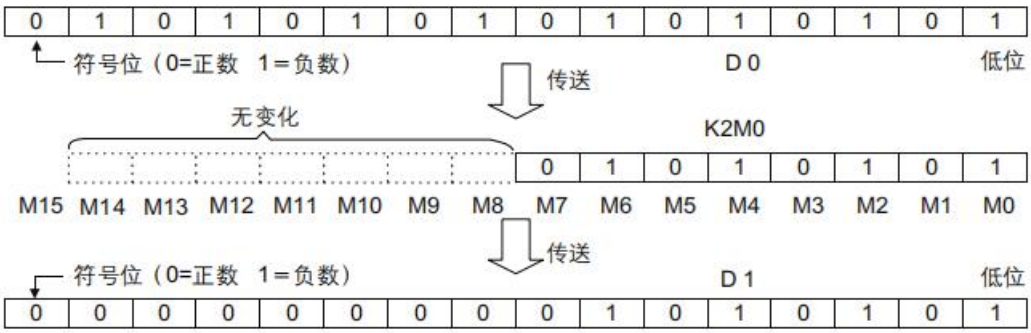
K4M0 表示位软元件 M0、M1、M2、…、M15 共 16 个位组合成 16 位数据使用；

K8M0 表示位软元件 M0、M1、M2、…、M31 共 32 个位组合成 32 位数据使用。

例如：K2M0，由于是 M0~M7，所以是 2 位数的数据。

D0 向 K2M0 传送 16 位数据，低 8 位数据存入 M0~M7 后，数据长度不足的高位部分不被传送（32 位数据的情况相同）。





使用位数 Kn 指定时避开特殊继电器 M 和特殊继电器 D 的编号范围，以免发生错误。

2.9 计数器 C

计数器是用来计数的，当计数值达到预设值时执行动作。计数器是在执行扫描操作时对内部信号（如 X、Y、M、S、T 等）进行计数。

寄存器长度 32 位，当通过 16 位指令访问时自动使用低 16 位。

处理数值范围（十进制）：

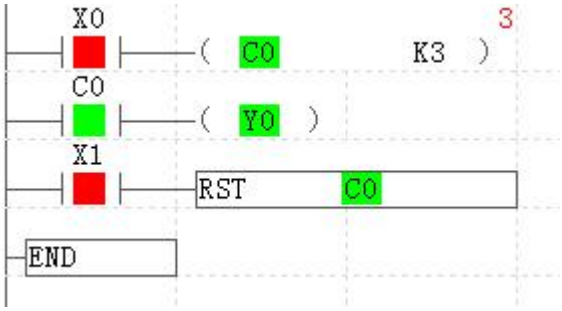
16 位：-32768~32767

32 位：-2147483648~ +2147483647

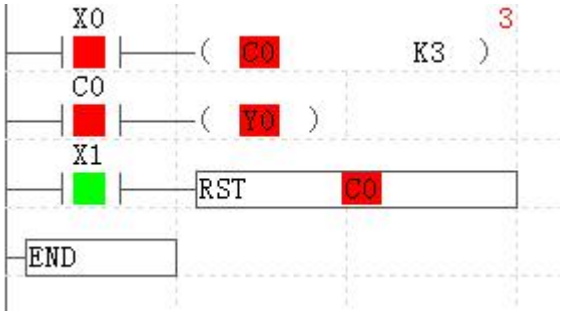
编号：0~127

掉电保持：100~107

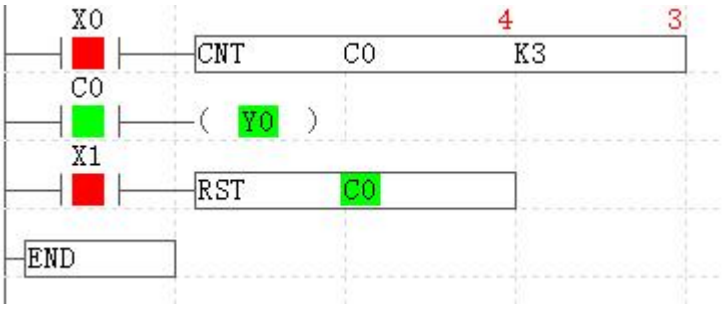
例：下图中，当 X0 每接通一次，计数器 C0 则计数一次，当所计次数与预设值（C0=3）相同时，C0 的常开触点接通，则 Y0 有输出。C0 接通后，X0 无论是断开还是闭合，C0 仍然导通。



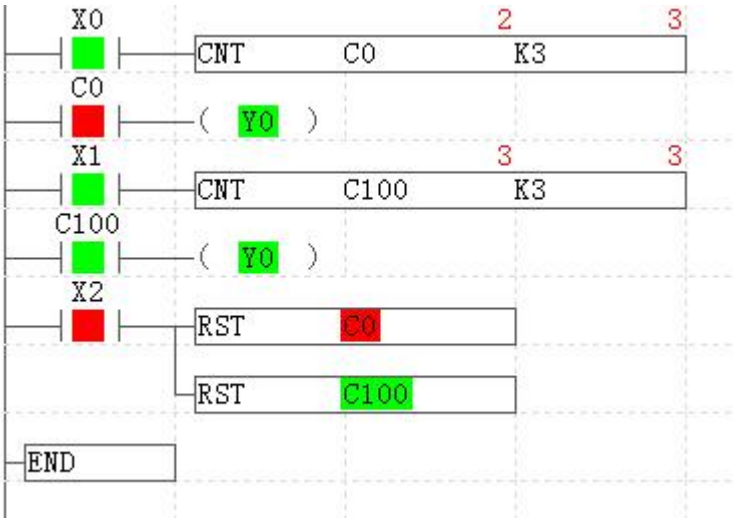
当 X1 接通时，复位 C0，C0 清零。



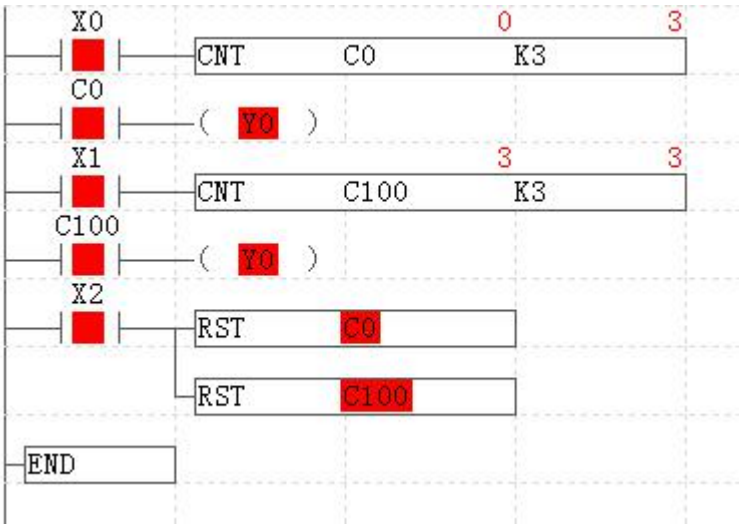
例：使用计数器指令时，与不使用计数器指令不同之处在于会记录下计数器导通次数。



例：如下图，C100 为掉电保持型，此时给控制器断电。

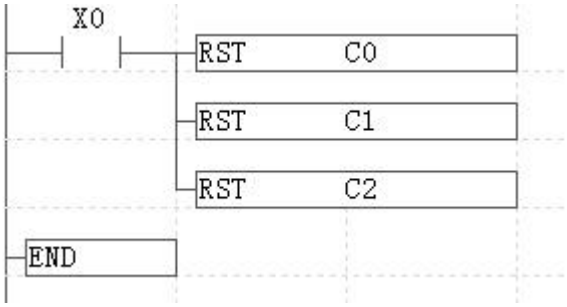


再次给控制器上电后，C0 清零，C100 仍保持断电前的数值，接通 X1，C100 导通。

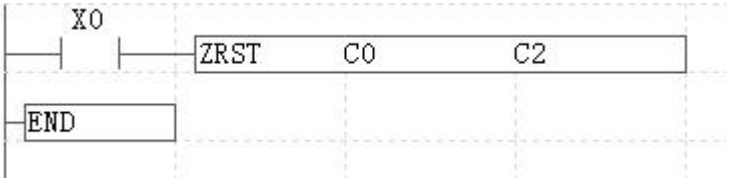


掉电保持软元件初始化方法

RST 指令复位：一次只能复位一个软元件，若需复位多个软元件，可连续使用 RST 指令。可复位的软元件：Y, M, S, T,C,D, V,Z,DT,LV



ZRST 指令成批复位：一次操作复位多个连续编号软元件。可复位的软元件：Y, M, S,T,C,D, V,Z,DT,LV



2.10 定时器 T

定时器的编号按十进制分配，单位为 1ms，它的作用是当达到所设定的时间输出触点就会动作，相当于继电器电路中的时间继电器。

定时器采用程序储存器内的常数 K 作设定值，也用数据寄存器 D 的内容进行间接制定。

寄存器长度 32 位，当通过 16 位指令访问时自动使用低 16 位。

同 BASIC 的 TIMER 命令。

处理数值范围（十进制）：

16 位：0~32767

32 位：0~ +2147483647

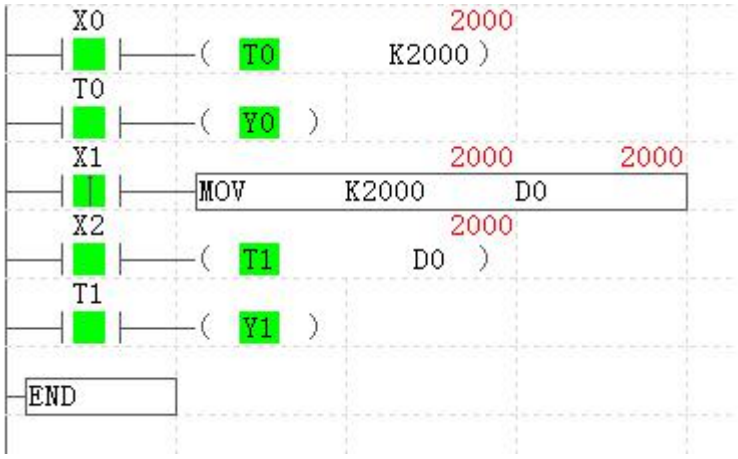
编号：0~127

掉电保持：100~127，掉电保持型失电后计时值不会清零，下次开启时直接触发。

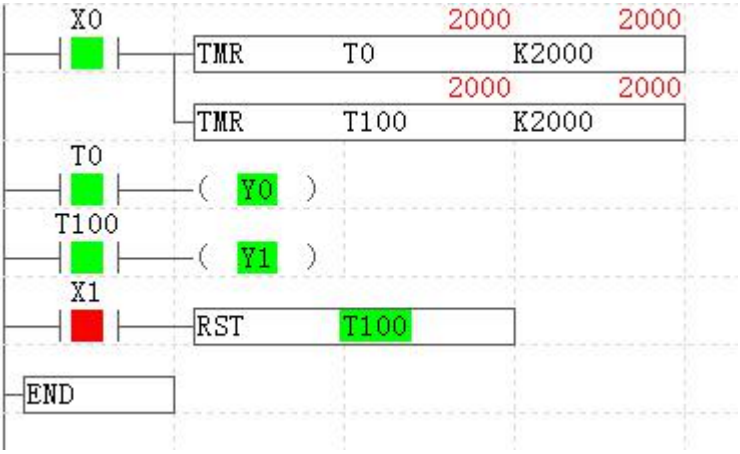
例：在下图中，T0 的值直接指定，T1 的值经由寄存器间接指定。

按下 X0，接通 T0 定时器开始计时，2S 时间到，T0 闭合，输出 Y0。按下 X0，若 2S 时间未到时便断开 X0，此时 T0 的值变为 0。

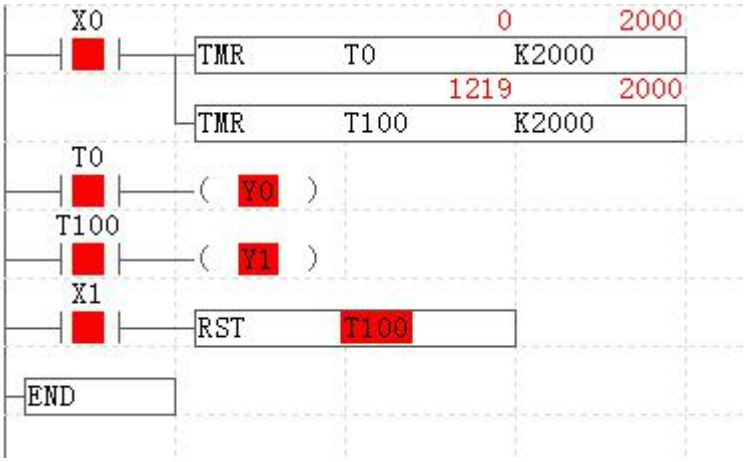
按下 X1 时，将常数 2000 传送到寄存器 D0，按下 X2，接通 T1 定时器。开始计时，2S 时间到，T1 闭合，输出 Y1。



例：使用定时器指令，与不使用定时器指令不同之处在于会显示当前计时时间。



例：T100 为掉电保持型，当计时为达到预设值时，断开 X0，T0 清零，T100 保持当前计时数值，X0 再次导通，T100 从当前值开始计时。X0 断开，T0、T100 均失电，Y0、Y1 断开。



控制器断电后再次上电，掉电保持性定时器数值仍为断电前的值，普通定时器数值清零。

2.11 变址寄存器 V

变址寄存器 V 与普通的数据寄存器一样，进行数值数据读入、写出的 16 位数据寄存器，V0~V7 共 8 个。

这种变址寄存器除了和普通的数据寄存器有相同的使用方法外，在应用指令的操作数中，还可以和其他的软元件编号或数值组合使用，可在程序中改变软元件编号或数值内容的一个特殊寄存器。

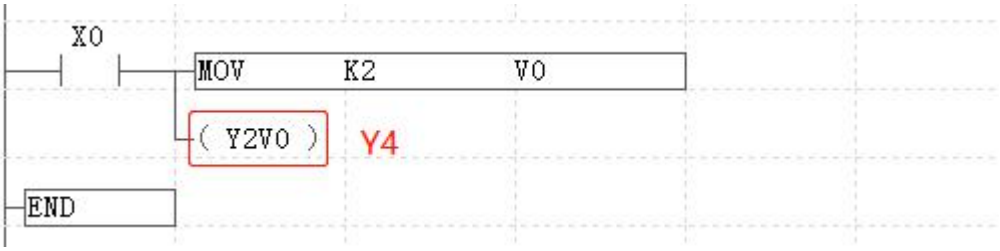
可能变址的软元件有：X、Y、M、S、T、C、D、K

寄存器长度 16 位。

编号：0~7

使用变址寄存器 V 变址软元件时，避开特殊继电器 M 和特殊继电器 D 的编号范围，以免发生错误。

例：在下图中，按下 X0，传送常数 2 到 V0，驱动 Y4 线圈（Y2V0 就等于 Y（2+V0））



2.12 变址寄存器 Z

变址寄存器 Z 的用法同变址寄存器 V 一样，进行数值数据读入、写出的 16 位数据寄存器，V0~V7 共 8 个。但在处理 32 位应用指令中的软元件或处理超过 16 位范围的数值时，必须用 Z0~Z7。

控制器将 Z 作为 32 位寄存器的低位侧进行动作，而 V 为高位侧。

16 位数据寄存器	16 位数据寄存器	32 位数据寄存器组合方式

V0	Z0	V0（高 16 位）	Z0（低 16 位）
V1	Z1	V1（高 16 位）	Z1（低 16 位）
V2	Z2	V2（高 16 位）	Z2（低 16 位）
V3	Z3	V3（高 16 位）	Z3（低 16 位）
V4	Z4	V4（高 16 位）	Z4（低 16 位）
V5	Z5	V5（高 16 位）	Z5（低 16 位）
V6	Z6	V6（高 16 位）	Z6（低 16 位）
V7	Z7	V7（高 16 位）	Z7（低 16 位）

这种变址寄存器除了和普通的数据寄存器有相同的使用方法外，在应用指令的操作数中，还可以和其他的软元件编号或数值组合使用，可在程序中改变软元件编号或数值内容的一个特殊寄存器。

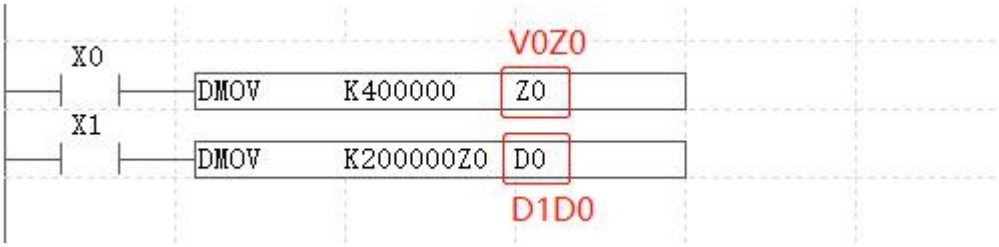
可能变址的软元件有：X、Y、M、S、T、C、D、K

寄存器长度 16 位。

编号：0~7

使用变址寄存器 V 变址软元件时，避开特殊继电器 M 和特殊继电器 D 的编号范围，以免发生错误。

例：按下 X0 时，传送的常量 K400000 超过 16 位范围的数值，需要占 V0Z0 两个位。



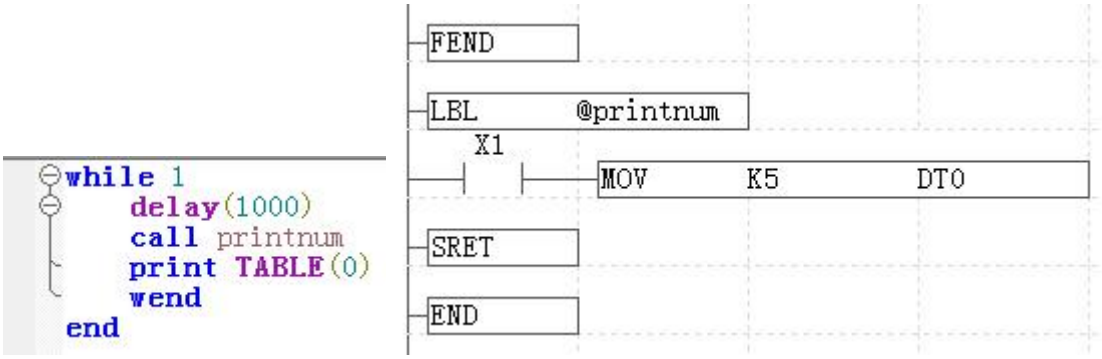
2.13 浮点寄存器 DT

此寄存器为 32 位浮点数，采用其他格式访问时会自动转换类型，转成整数时自动去掉小数部分。

此寄存器与 ZBasic 的 TABLE 为同一个，可以用于与 ZBasic 之间传输数据。

编号：0~TSIZE-1

例：控制器上电，BASIC 程序设为自动运行，调用 PLC 子函数@printnum，打印输出函数 TABLE(0)为 0。X1 闭合，将 K5 传送到 DT0，DT0 等同于 TABLE(0)，所以打印输出函数 TABLE(0)为 5。



2.14 局部寄存器 LV

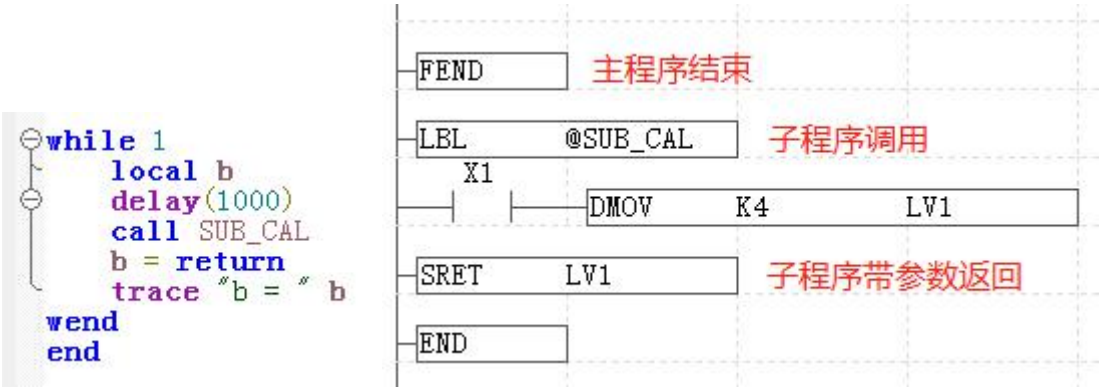
在函数内局部有效的寄存器，每个函数的每个运行实例使用不同的数据；每个函数有 8 个 LV，编号 0-7；此寄存器为 32 位浮点数。

此寄存器与 ZBASIC 的 LOCAL 类似，ZBASIC 调用 ZPLC 时，调用参数自动传入 LV 寄存器。

LV 存储器可以同时作为变址寄存器。

编号：0~7

例：控制器上电，BASIC 程序运行，调用 PLC 子函数@SUB_CAL，此时 LV1 的值为 0，由 SRET 指令给定作返回值，RETURN 的返回值为 0，打印输出函数 b=0。当 X1 闭合时，LV1 为 4，由 SRET 指令给定作返回值，RETURN 的返回值为 4，打印输出函数 b=4。



2.15 Label 寄存器 L

此寄存器通过 LBL 指令定义，一共有 64 个 L 寄存器，L 寄存器可以通过 CALL/JUMP 指令来调用或跳转。

编号：0~63

2.16 特殊继电器 M

M8000：只要 PLC 处于运行状态，M8000 的常开触点一直保持闭合。可以作为需要一直驱动程序的输入条件以及作为控制器的运行状态的显示来使用。

M8001：只要 PLC 处于运行状态，M8001 的常开触点一直断开。

M8002：PLC 从停止到运行的时候，M8002 仅在第一个扫描周期内导通一次，一般用于初始化参数和程序复位，或者批量赋值。

M8003：PLC 从停止到运行的时候，M8003 断开第一个扫描周期，然后恢复常闭状态。

M8011：是以 10 毫秒为周期时间，在这 10 毫秒内 M8011 的触点接通 5MS，断开 5 毫秒，不断重复循环。

M8012：是以 100 毫秒为周期时间，在这 100 毫秒内 M8011 的触点接通 50MS，断开 50 毫秒，不断重复循环。

M8013：是以 1 秒为周期时间，在这 1 秒内 M8011 的触点接通 500MS，断开 500 毫秒，不断重复循环。

M8014：是以 1 分钟为周期时间，在这 1 分钟内 M8011 的触点接通 30 秒，断开 30 秒，不断重复循环。

M8020：加减运算结果为零时，M8020 就会置位，反之则为复位状态。

M8021：减法运算结果小于负的最大值时，借位标志位动作。

M8022：加法运算结果大于正的最大值时，进位标志位动作。

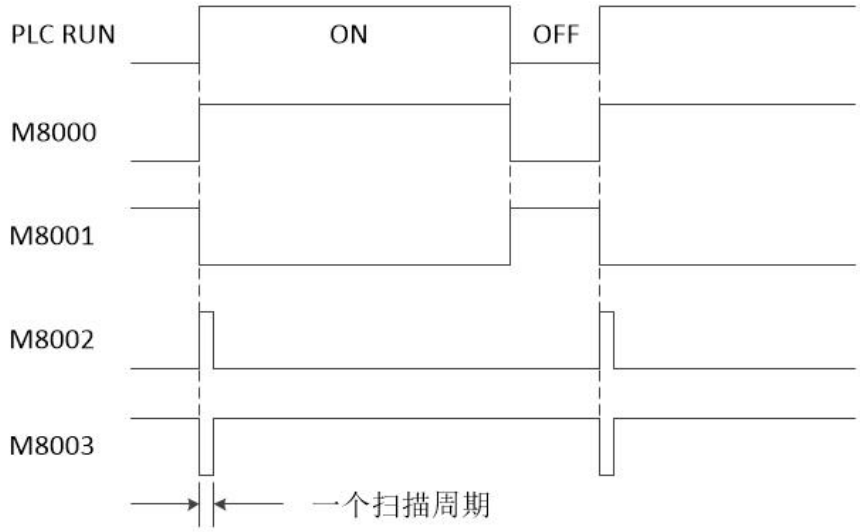
M8023：小数点标志。

M8100-M8199：轴 0-99 的 IDLE 标志

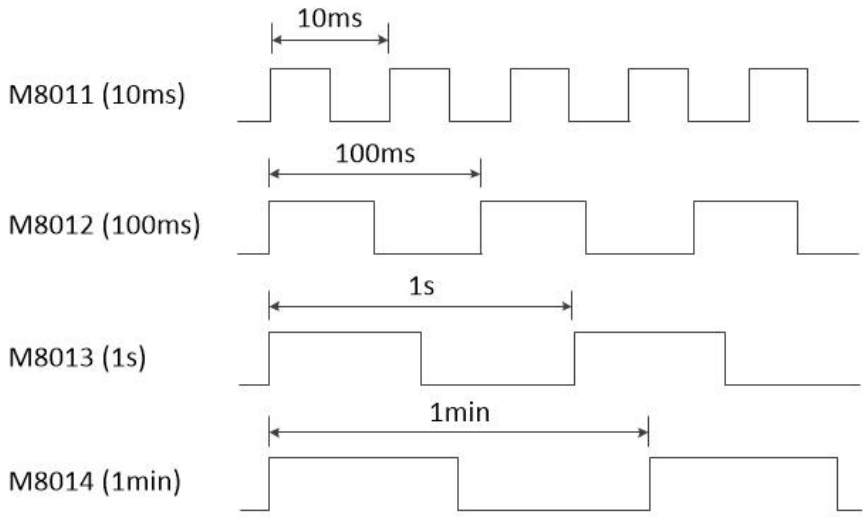
M8200-M8299：轴 0-99 的 BUFFER 剩余标志

M10000~：IN(0)对应 M10000 地址，IN(1)对应 M10001 地址，以此类推。

M20000~：OP(0)对应 M20000 地址，OP(1)对应 M20001 地址，以此类推。



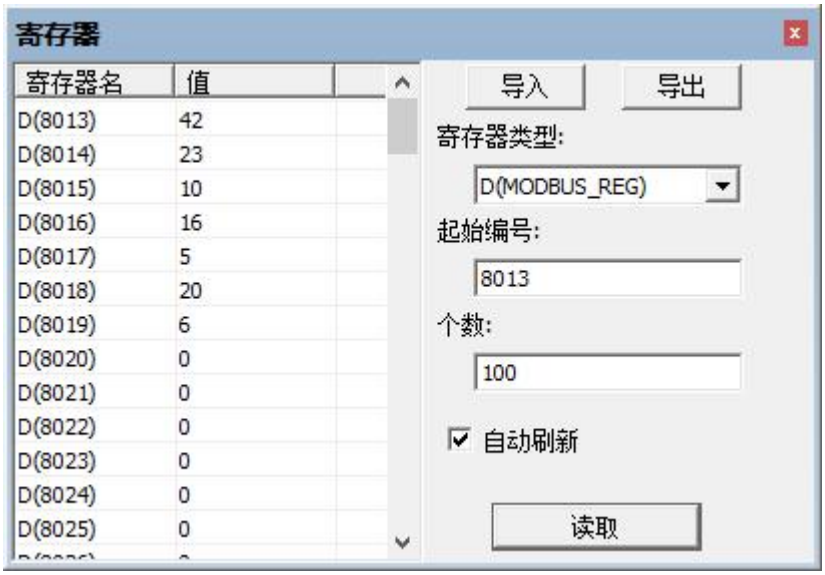
当 PLC 处于 run 状态时，特殊继电器 M8011~M8014 的电平变化如下图，当 PLC 停止时，均不不动作。



2.17 特殊寄存器 D

D8001：版本
D8004：错误
D8010：扫描时间
D8011：扫描最小时间
D8012：扫描最大时间
D8013：秒
D8014：分
D8015：时
D8016：日
D8017：月
D8018：年
D8019：星期
D10000-D10198：DPOS，浮点方式，每个轴占 1 个 MOBBUS_IEEE。
D11000-D11198：MPOS，浮点方式，每个轴占 1 个 MOBBUS_IEEE。
D12000-D12198：VPSPEED，浮点方式，每个轴占 1 个 MOBBUS_IEEE。
D13000-D13128：AOUT，模拟量输出，每个通道占 1 个 MOBBUS_REG。
D14000-D14256：AIN，模拟量输入，每个通道占 1 个 MOBBUS_REG。

以上特殊寄存器除 D8000 和 D8004，其他均为只读，可通过寄存器窗口查看特殊寄存器 D 的值，如下图，显示当前时间为：星期 6，2020 年 5 月 16 日 10 时 23 分 42 秒。


寄存器
寄存器名
值
D(8013) 42
D(8014) 23
D(8015) 10
D(8016) 16
D(8017) 5
D(8018) 20
D(8019) 6
D(8020) 0
D(8021) 0
D(8022) 0
D(8023) 0
D(8024) 0
D(8025) 0
导入
导出
寄存器类型:
D(MODBUS_REG)
起始编号:
8013
个数:
100
自动刷新
读取

第三章 指令

3.1 常用指令

常用指令一览表

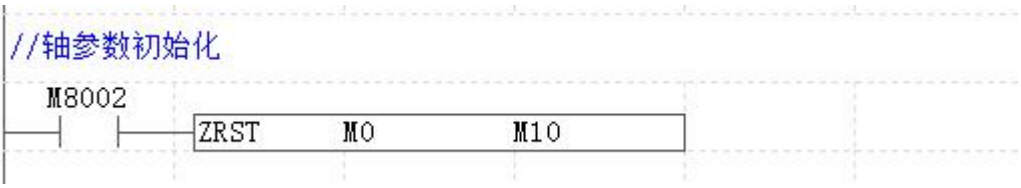
指令	功能	操作数类型
LD	取指令，常开触点	X,Y,M,S,T,C,@a
LDI	取反指令，常闭触点	X,Y,M,S,T,C,@a
OUT	输出指令	Y,M,S,T,C
LDP	取脉冲上升沿指令	X,Y,M,S,T,C,@a
LDF	取脉冲下降沿指令	X,Y,M,S,T,C,@a
AND	与指令，常开触点	X,Y,M,S,T,C,@a
ANI	与非指令，常闭触点	X,Y,M,S,T,C,@a
ANB	并联模块串联指令	X,Y,M,S,T,C,@a
ANDP	与脉冲上升沿指令	X,Y,M,S,T,C,@a
ANDF	与脉冲下降沿指令	X,Y,M,S,T,C,@a
OR	或指令，常开触点	X,Y,M,S,T,C,@a
ORI	或指令，常闭触点	X,Y,M,S,T,C,@a
ORB	串联模块并联指令	X,Y,M,S,T,C,@a
ORP	或脉冲上升沿指令	X,Y,M,S,T,C,@a
ORF	或脉冲下降沿指令	X,Y,M,S,T,C,@a
PLS	上升沿微分输出指令	Y,M
PLF	下降沿微分输出指令	Y,M
STL	步进开始指令	S
RET	步进结束指令	无
SET	置位指令	Y,M,S,T
RST	复位指令	Y,M,S,T,C,D,S,V,Z,DT,LV
NOP	空操作指令	无
EU	累积寄存器上升沿触发指令	无
ED	累积寄存器下降沿触发指令	无
INV	运算结果取反指令	无
EXE	调用 Basic 标准指令	Basic 标准指令
EXEP	调用 Basic 标准指令，脉冲执行	Basic 标准指令
MPS	存入堆栈指令	无
MRD	读出堆栈指令，不删除结果	无
MPP	读出堆栈指令，删除结果	无

指令	功能	操作数类型
TMR	16 位定时器指令	S1：T0~T127 S2：KnX,KnY,KnM,KnS,T,C,D,K,H,V,Z,LV,DT,@
DTMR	32 位定时器指令	S1：T0~T127 S2：KnX,KnY,KnM,KnS,T,C,D,K,H,V,Z,LV,DT,@
ATMR	16 位定时器串联指令	S1：T0~T127 S2：KnX,KnY,KnM,KnS,T,C,D,K,H,V,Z,LV,DT,@
CNT	16 位计数器指令	S1：C0~C127 S2：KnX,KnY,KnM,KnS,T,C,D,K,H,V,Z,LV,DT,@
DCNT	32 位计数器指令	S1：C0~C127 S2：KnX,KnY,KnM,KnS,T,C,D,K,H,V,Z,LV,DT,@
MC	主控起始指令	N
MCR	主控结束指令	N

//注释

“//”表示此行为注释，以文字方式来说明相关程序块，注释总是占用一整行。

由于注释会占用整行空间，因此如果某行已经存在其他元件，则此行不能再输入注释；同理，已经被注释占用的行，也不能再输入其他任何元件。



LD

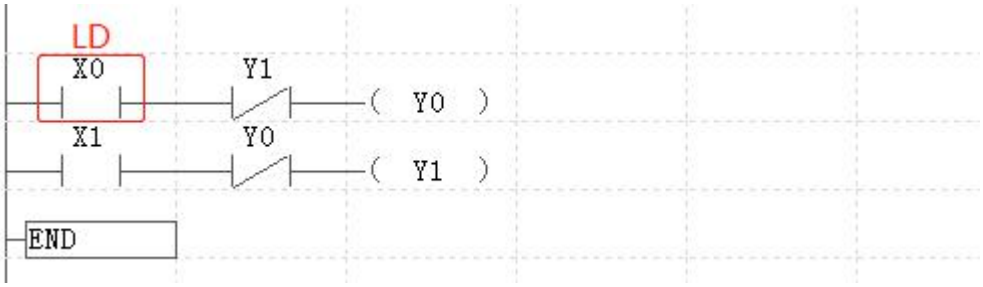
指令说明：

LD 是取指令。用于与母线连接的常开触点，或触点块开始的常开触点。

操作数：

S：X,Y,M,S,T,C,@a

编程示例：



LDI

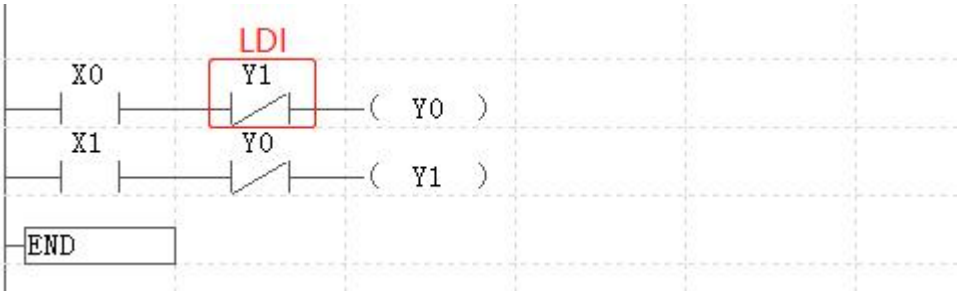
指令说明：

LDI 是取反指令。用于与母线连接的常闭触点，或触点块开始的常闭触点。

操作数：

S：X,Y,M,S,T,C,@a

编程示例：



LDP

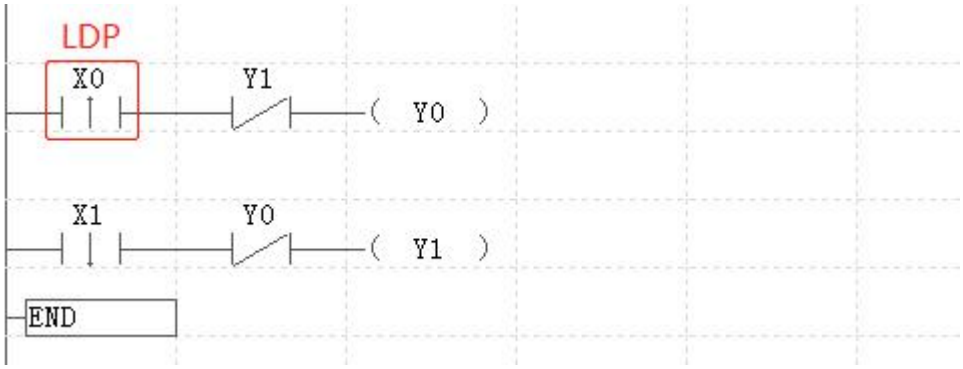
指令说明：

LDP 是取脉冲上升沿指令。用来进行上升沿检测的指令，仅在指定位软元件的上升沿时（由 OFF→ON 变化时）接通一个扫描周期，又称上升沿微分触点指令。

操作数：

S：X,Y,M,S,T,C,@a

编程示例：



LDF

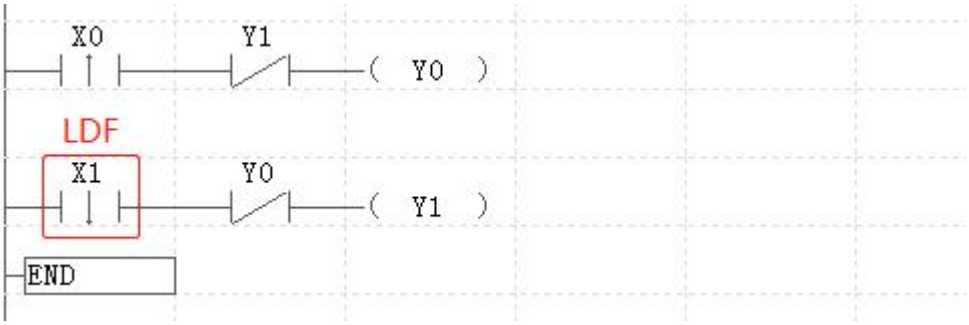
指令说明：

LDF 是取脉冲下降沿指令。用来进行下降沿检测的指令，仅在指定位软元件的下降沿时（由 ON→OFF 变化时）接通一个扫描周期，又称下降沿微分触点指令。

操作数：

S：X,Y,M,S,T,C,@a

编程示例：



OUT

指令说明：

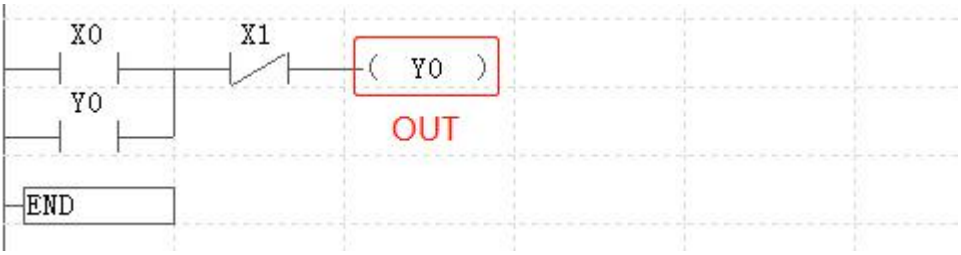
OUT 是输出指令。是对软元件线圈驱动的指令。

操作数：

S：Y,M,S,T,C

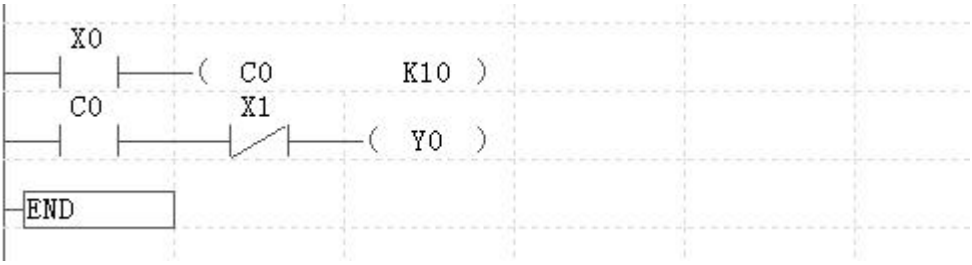
编程示例：

当 X0 接通时，OUT 指令输出 Y0。



OUT 指令驱动定时器，当 X0 接通时，定时器 T0 开始计时，2s 后位软元件 T0 接通，输出 M0。

OUT 指令驱动计数器，当 X0 接通 10 次后，计数器 C0 达到预设值，位软元件 C0 接通，输出 Y0。



AND

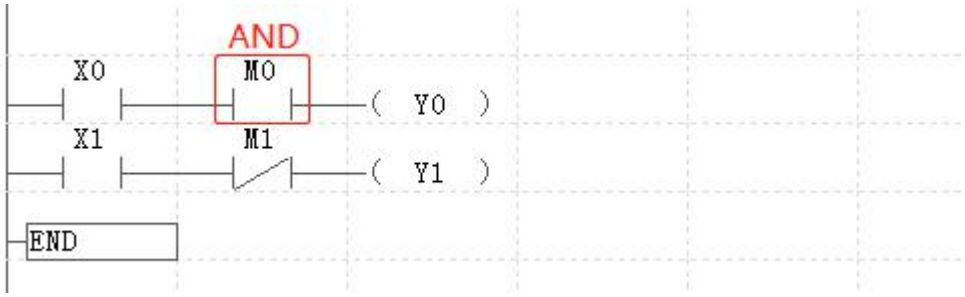
指令说明：

AND 是与指令。用于单个常开触点的串联连接。

操作数：

S：X,Y,M,S,T,C,@a

编程示例：



ANI

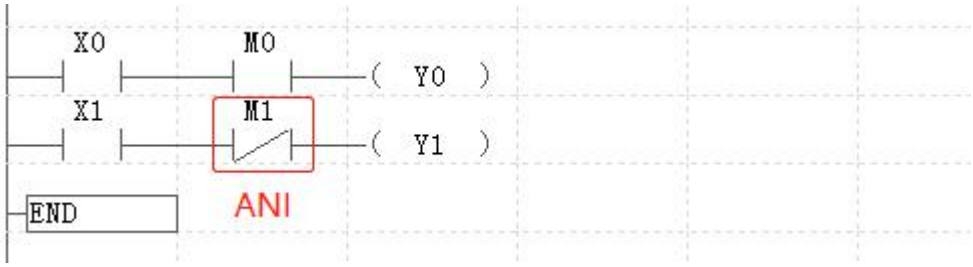
指令说明：

ANI 是与非指令。用于单个常闭触点的串联连接。

操作数：

S：X,Y,M,S,T,C,@a

编程示例：



ANB

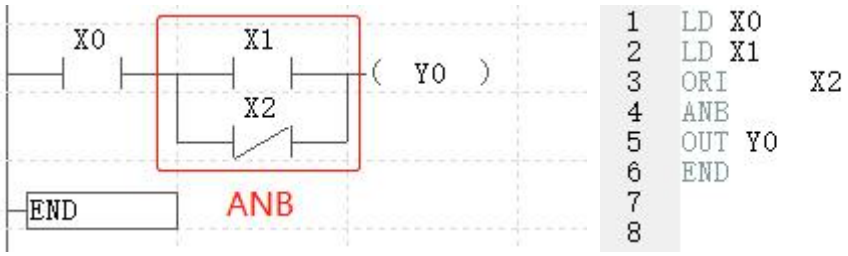
指令说明：

ANB 指令用于将并联回路块与前面的回路串联连接。有多个并联回路的时候，对每个回路快使用 ANB 指令，从而连接到一起。

操作数：

无

编程示例：



ANDP

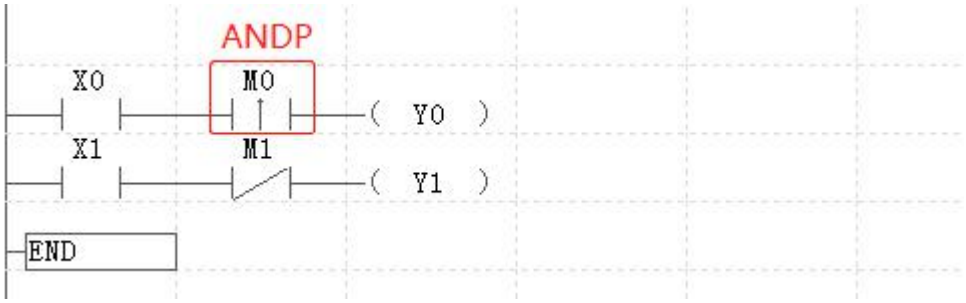
指令说明：

ANDP 是与脉冲上升沿指令。用于单个上升沿触点的串联连接，仅在指定位软元件的上升沿时（由 OFF→ON 变化时）接通一个扫描周期。

操作数：

S：X,Y,M,S,T,C,@a

编程示例：



ANDF

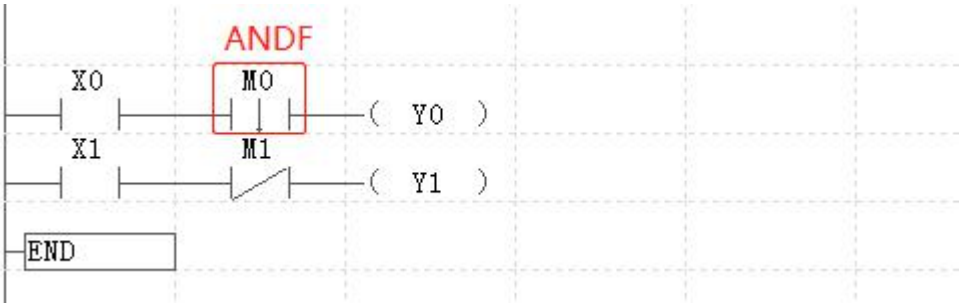
指令说明：

ANDF 是与脉冲下降沿指令。用于单个下降沿触点的串联连接，仅在指定位软元件的下降沿时（由 ON→OFF 变化时）接通一个扫描周期。

操作数：

S：X,Y,M,S,T,C,@a

编程示例：



OR

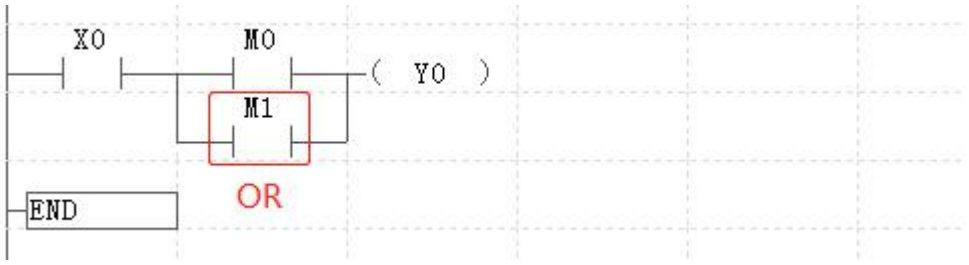
指令说明：

OR 是或指令。用于单个常开触点的并联连接。

操作数：

S：X,Y,M,S,T,C,@a

编程示例：



ORI

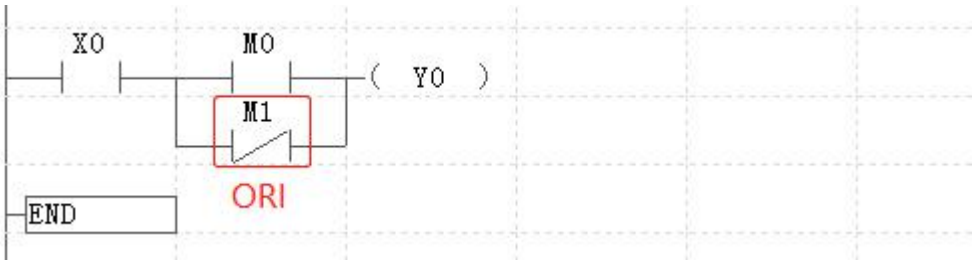
指令说明：

ORI 是或指令。用于单个常闭触点的并联连接。

操作数：

S：X,Y,M,S,T,C,@a

编程示例：



ORB

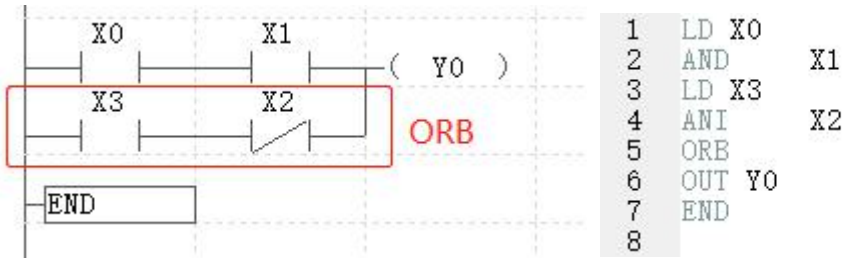
指令说明：

ORB 指令用于并联连接串联回路块，由 2 个以上的触点串联连接的回路称为串联回路块。有多个并联回路时，在每个回路块中使用 ORB 指令，将回路连接在一起。

操作数：

无

编程示例：



ORP

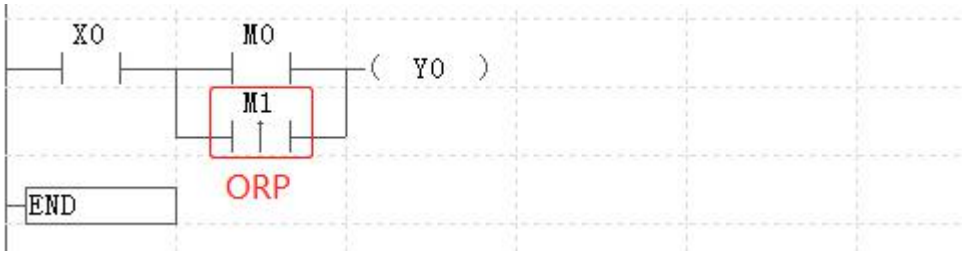
指令说明：

ORP 是或脉冲上升沿指令。用于单个上升沿触点的并联连接，仅在指定位软元件的上升沿时（由 OFF→ON 变化时）接通一个扫描周期。。

操作数：

S：X,Y,M,S,T,C,@a

编程示例：



ORF

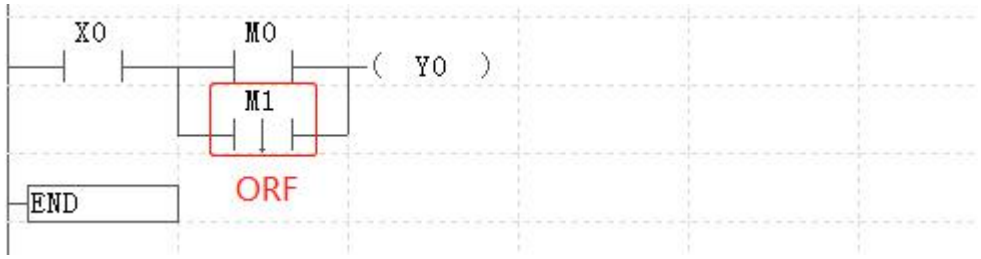
指令说明：

ORF 是或脉冲下降指令。用于单个下降沿触点的并联连接，仅在指定位软元件的下降沿时（由 ON→OFF 变化时）接通一个扫描周期。。

操作数：

S：X,Y,M,S,T,C,@a

编程示例：



PLS

指令说明：

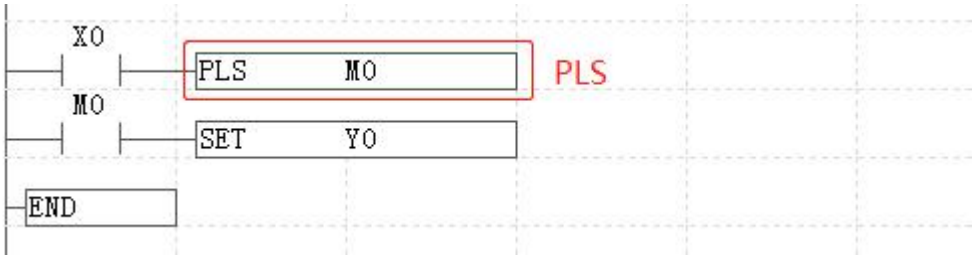
PLS 是上升沿微分输出指令。其作用是在驱动条件成立时，输入信号的上升沿使辅助继电器 M 或输出继电器 Y 接通一个扫描周期。

操作数：

S：Y,M

编程示例：

X0 从断开到接通时，M0 会导通一个扫描周期。



PLF

指令说明：

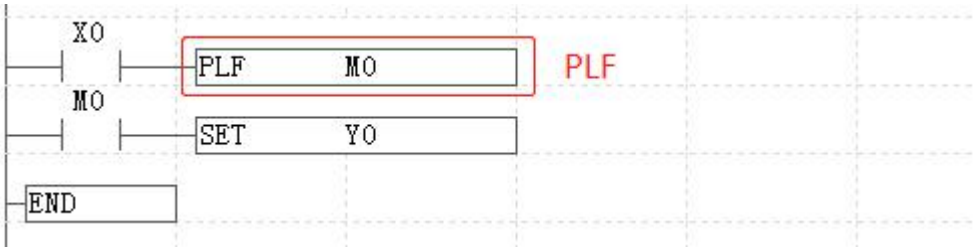
PLF 是下降沿微分输出指令。其作用是在驱动条件成立时，输入信号的下降沿使辅助继电器 M 或输出继电器 Y 接通一个扫描周期。

操作数：

S：Y,M

编程示例：

当 X0 从接通到断开时，M0 会导通一个扫描周期。



STL

指令说明：

步进开始指令，使用步进梯形图指令的程序，以机械的动作为基础，对各工序分配状态 S，作为连接在状态触点（STL 触点）中的回路，对输入条件和输出控制的顺序进行编程。

当前状态（S0）向下一状态（S1）转移时，该扫描周期两个状态内的动作均得到执行；下一扫描周期执行时，当前状态（S0）被下一状态（S1）所复位，当前状态（S0）内的动

作不被执行，所以 OUT 元件的输入均被断开。

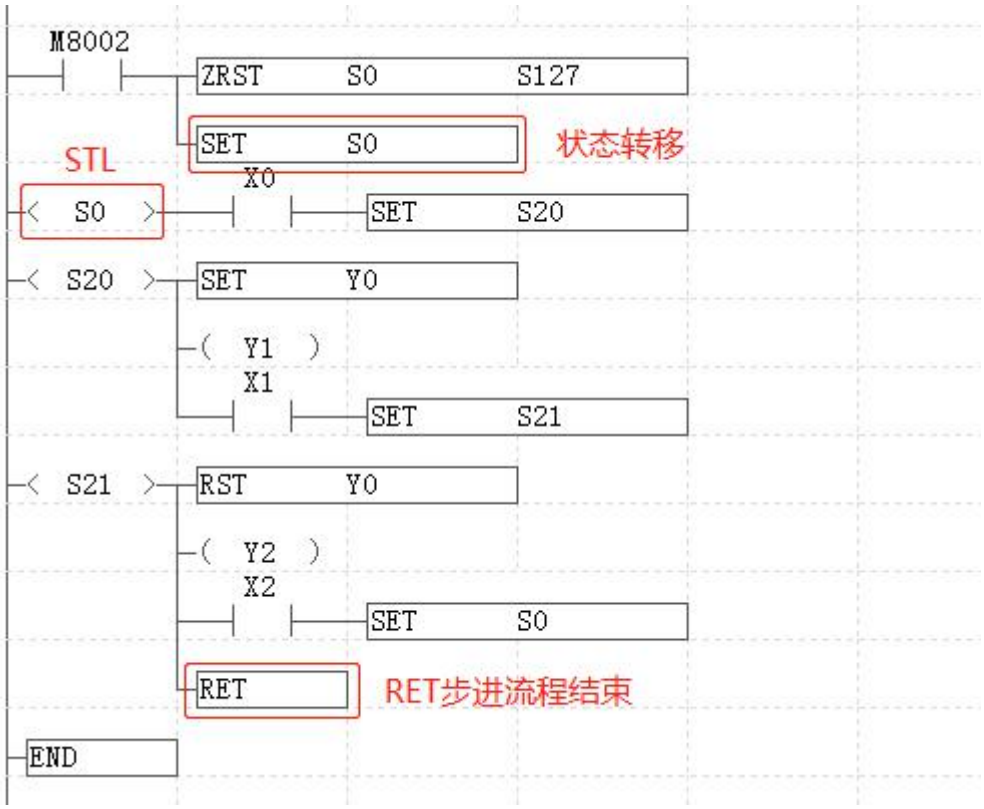
状态的转移只能使用 SET Sn，若使用 OUT，则 Sn 作为普通的辅助继电器使用；步进程序编写完成后需要使用 RET 结束程序。

操作数：

S：Sn

编程示例：

M8002 上电导通一个扫描周期，将 S0-S127 清零后，再置位 S0，步进状态转移。X0 闭合后，S20 导通，输出 Y0 和 Y1，同时 S0 被复位。X1 闭合后，S21 导通，清零 Y0，输出 Y2，同时 S20 被复位。X2 闭合后，置位 S0，步进流程结束，同时 S21 被复位。



RET

指令说明：

步进结束指令，RET 指令代表一个步进流程的结束，用于返回主程序的指令，所以一连串步进点的最后一定要有 RET 指令。

操作数：

无

编程示例：

参见 STL 指令。

SET

指令说明：

SET 是置位指令。其作用是使被操作的目标软元件置位并动作保持。对于同一软元件，SET、RST 可多次使用，顺序也可随意，但最后执行者有效。

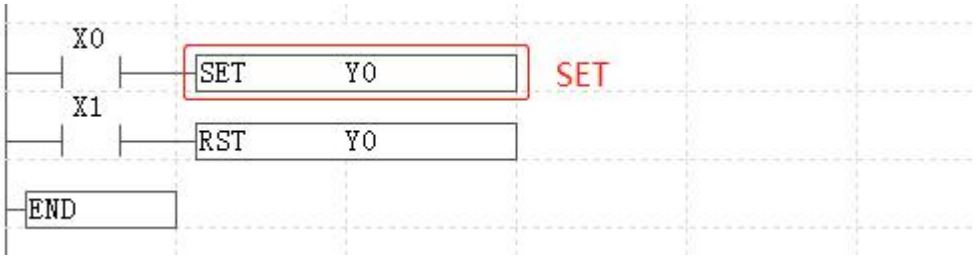
SET 指令一次只能置位一个软元件，想成批复位可多次使用 SET。

操作数：

S：Y,M,S,T

编程示例：

当 X0 接通时，Y0 保持输出。即使 X0 断开，Y0 仍保持输出，直到 X1 接通，Y0 复位。



RST

指令说明：

RST 是复位指令。其作用是使被操作的目标软元件复位并消除动作保持，清除当前值及寄存器值清零，对于同一软元件，SET、RST 可多次使用，顺序也可随意，但最后执行者有效。此外，要使数据寄存器 D，变址寄存器 V、Z 的内容清零时，也可使用 RST 指令。

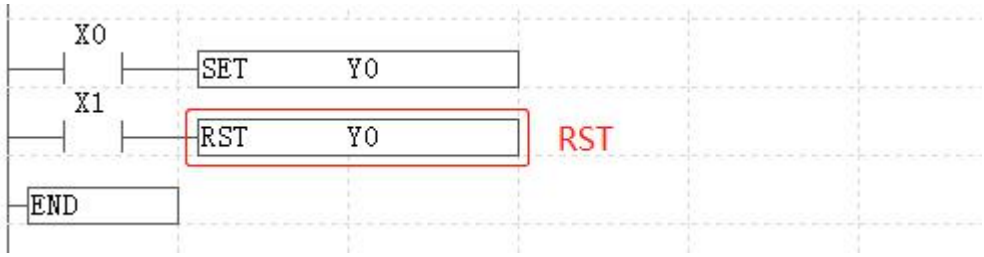
RST 指令一次只能复位一个软元件，想成批复位可多次使用 RST 或使用 ZRST 成批复位指令。

操作数：

S：Y,M,T,C,D,S,V,Z,DT,LV

编程示例：

当 X1 一旦接通，即使 X1 断开，Y0 仍保持断开。此时如要接通 Y0，需使 X1 断开后闭合 X0。



NOP

指令说明：

NOP 指令为空操作的指令，在程序中不做任何运算，因此执行后仍会保持原逻辑运算结果，更改现有的程序，改写成 NOP 指令时，等同于执行删除指令的操作。使用时如下：想要删除某一指令，而又不想改变程序长度，则可以 NOP 指令取代。

语句表中插入 NOP 指令后无法转换为梯形图。

操作数：

无

INV

指令说明：

INV 指令是将 INV 指令之前的逻辑运算结果取反的指令，无需指定软元件的编号。

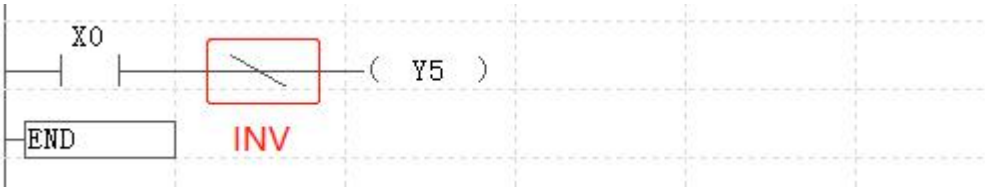
INV 指令不要直接与母线相连，会导致后方触点无法执行。

操作数：

无

编程示例：

X0=OFF 时，输出 Y0；X0=ON 时，Y0 断开。



EXE

指令说明：

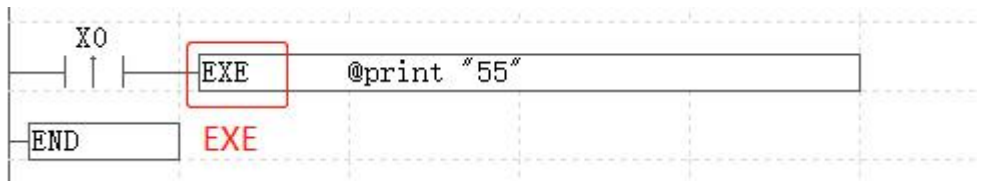
PLC 可以通过 EXE 指令调用 Basic 标准指令。EXE 指令为连续执行型指令。

操作数：

Basic 标准指令

编程示例：

调用 Basic 中的 print 函数，当 X0 由 OFF 变为 ON 时，打印一次 55。



EXEP

指令说明：

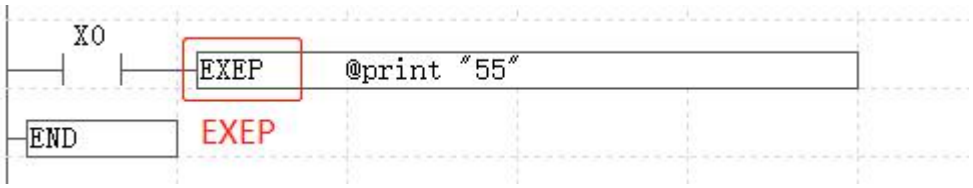
EXEP 指令用法与 EXE 相同，是 EXE 指令的脉冲形式。仅在驱动输入由 OFF 变为 ON 后，才调用一次 Basic 标准指令。

操作数：

Basic 标准指令

编程示例：

调用 Basic 中的 print 函数，当 X0 接通时，打印出一次 55。



MPS

指令说明：

MPS、MRD、MPP 指令都无操作数，这 3 个指令所占程序步数均为 1。

嵌入式 PLC 中有 11 个栈空间，也就是说可以压栈的最大深度为 11 级。每使用一次 MPS 将当前结果压入第一段存储，以前压入的结果依次移入下一段。MPP 指令将第一段读出，并且删除它，同时以下的单元依次向前移。MRD 指令读出第一段，但并不删除它。其他单元保持不变。使用这 3 条指令可以方便多分支编程。

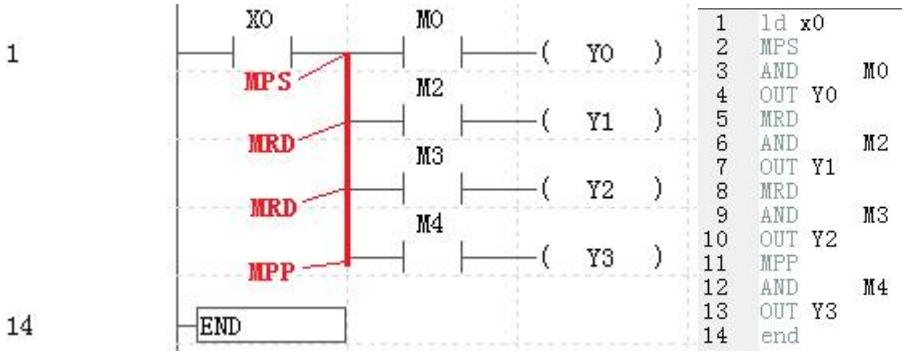
在进行多分支编程时，MPS 保存前面的计算结果，以后的分支可以利用 MRD,MPP 从栈中读出前面的计算结果，再进行后面的计算。最后一个分支必须用 MPP，保证 MPS、MPP 使用的次数相同。 注意，使用 MPP 以后，就不能使用 MRD 读出运算结果，也就是 MPP 必须放在最后的分支使用。

MRD 指令可以使用多次，没有限制。MPS 连续使用的最多次数为 11，但是可以多次使用。每个 MPS 指令都有一个 MPP 指令对应，MPP 的个数不能多于 MPS 的个数。

操作数：

无

编程示例：



MRD

参见 MPS 指令。

MPP

参见 MPS 指令。

TMR

指令说明：

当 TMR 指令执行时，其所指定的定时器线圈受电，定时器开始计时，当到达定时器所指定的定时值，定时器所对应的触点动作。

寄存器长度 16 位。

定时器一共 128 个，分别是 T0~T127，其中 T100~T127 为掉电保持型。

处理数值范围（十进制）：0-32767

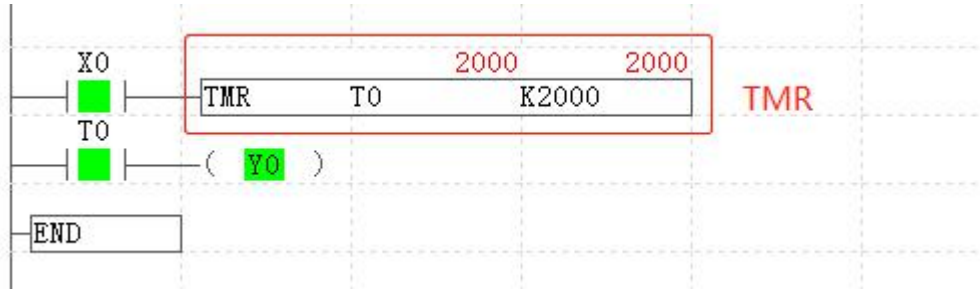
操作数：

S1：T0~T127

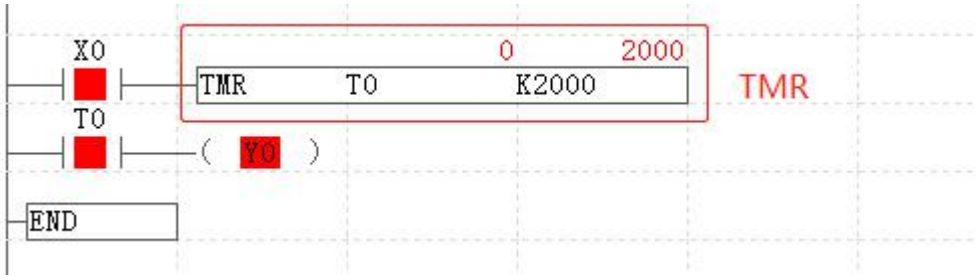
S2：KnX,KnY,KnM,KnS,T,C,D,K,H,V,Z,LV,DT,@

编程示例：

X0 接通后定时器 T0 开始计时，2 秒后 T0 常开触点闭合，线圈 Y0 导通。



当 X0 断开时，T0 失电，当前值清零，Y0 断开。



DTMR

指令说明：

当 TMR 指令执行时，其所指定的定时器线圈受电，定时器开始计时，当到达定时器所指定的定时值，定时器所对应的触点动作。

寄存器长度 32 位，当通过 16 位指令访问时自动使用低 16 位。

定时器一共 128 个，分别是 T0~T127，其中 T100~T127 为掉电保持型。

处理数值范围（十进制）：-2147483648 - +2147483647

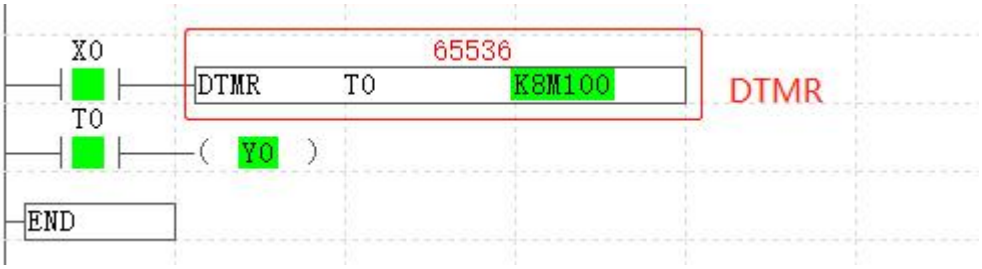
操作数：

S1：T0~T127

S2：KnX,KnY,KnM,KnS,T,C,D,K,H,V,Z,LV,DT,@

编程示例：

X0 接通后定时器 T0 开始计时，65536 毫秒后 T0 常开触点闭合，线圈 Y0 导通。



ATMR

指令说明：

ATMR 指令相当于 AND + TMR 指令之组合，其前面触点导通时，此指定的定时器将开始计时，当计时值到达时（计时值 >= 设定值），其触点闭合；当前面接点不成立时，则 ATMR 自动清除计时值。

寄存器长度 16 位。

定时器一共 128 个，分别是 T0~T127，其中 T100~T103 为掉电保持型。

处理数值范围（十进制）：0-32767

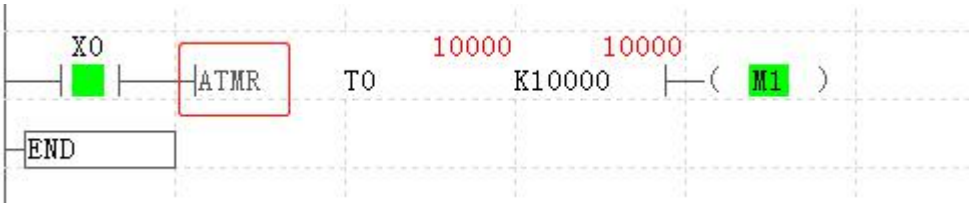
操作数：

S1：T0~T127

S2：KnX,KnY,KnM,KnS,T,C,D,K,H,V,Z,LV,DT,@

编程示例：

当 M0 接通时，定时器触点指令执行，T0 的数值计时达到 10000 时，该触点导通，M1 接通。当 M0 断开时，T0 失电，当前值清零，M0 断开。



CNT

指令说明：

该指令为 16 位计数指令，当 CNT 指令由 OFF 到 ON 执行，表示指定的计数器线圈由失电到受电，则该计数器计数值加 1，当计数到达所指定的数值时，计数器的触点动作。

寄存器长度 32 位，当通过 16 位指令访问时自动使用低 16 位。

计数器一共 128 个，分别是 C0~C127，其中 C100~C107 为掉电保持型。

处理数值范围（十进制）：0 – 32767

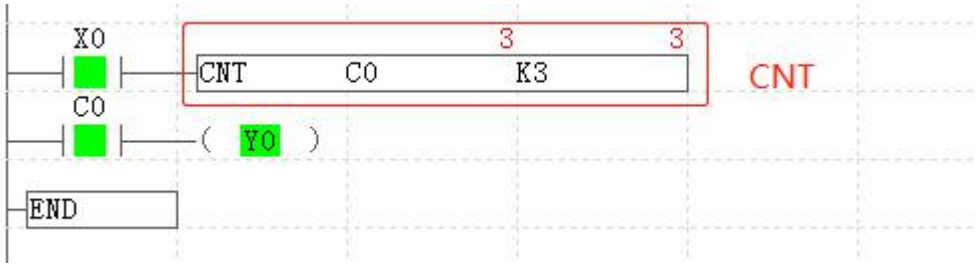
操作数：

S1：C0~C127

S2：KnX,KnY,KnM,KnS,T,C,D,K,H,V,Z,LV,DT,@

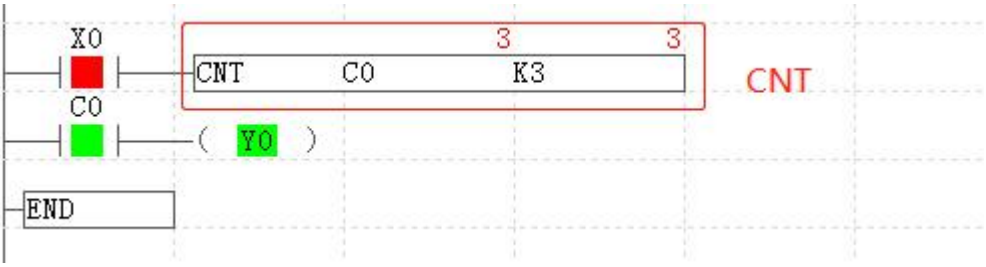
编程示例：

X0 由 OFF 变为 ON 则计数器 C0 数值加 1，当 C0 数值为 3 时，C0 常开触点闭合，Y0 导通。



C0 一旦导通再断开 X0，C0 保持闭合状态。

C0 达到计数值导通后，再接通 X0，C0 数值继续加 1。



DCNT

指令说明：

该指令为 32 位计数指令，当 DCNT 指令由 OFF 到 ON 执行，表示指定的计数器线圈由失电到受电，则该计数器计数值加 1，当计数到达所指定的数值时，计数器的触点动作。

计数器一共 128 个，分别是 C0~C127，其中 C100~C107 为掉电保持型。

处理数值范围（十进制）：-2147483648 - +2147483647

操作数：

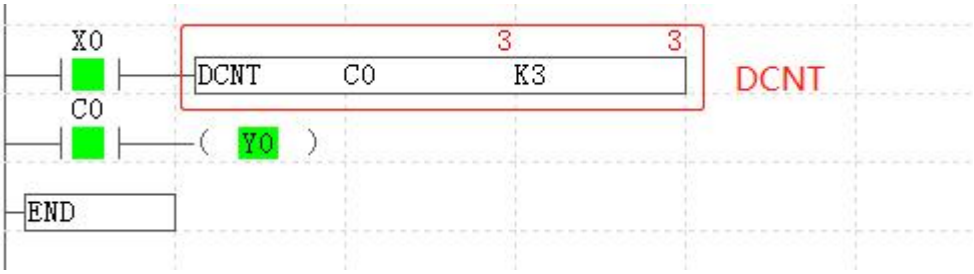
S1：C0~C127

S2：KnX,KnY,KnM,KnS,T,C,D,K,H,V,Z,LV,DT,@

编程示例：

X0 由 OFF 变为 ON 则计数器 C0 数值加 1，当 C0 数值为 10 时，C0 常开触点闭合，Y0 导通。C0 一旦导通再断开 X0，C0 保持闭合状态。

C0 达到计数值导通后，再接通 X0，C0 数值继续加 1。



MC

指令说明：

MC 为主控起始指令，当 MC 指令执行时，位于 MC 与 MCR 指令之间的指令照常执行。当 MC 指令 OFF 时，位于 MC 与 MCR 指令之间的指令动作如下所示：

一般定时器：计时值归零，线圈失电，接点不动作。

子程序用定时器：计时值归零，线圈失电，接点不动作。

累计型定时器：线圈失电，计时值及接点保持目前状态。

计数器：线圈失电，计数值及接点保持目前状态。

SET、RST 指令驱动的元件：保持目前状态。

Basic 调用指令：全部不动作。

MCR 为主控结束指令，置于主控程序最后，在 MCR 指令之前不可有接点指令，直接与母线相连。

操作数：

S：N

编程示例：

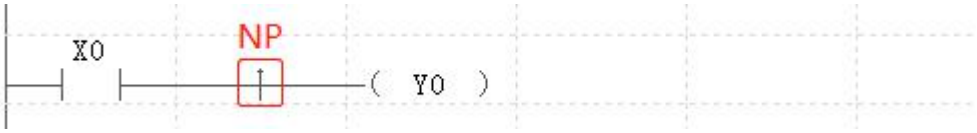
X0 闭合时，主控程序启动，仅执行 MC 与 MCR 之间的指令，X0 断开时，MC 与 MCR 之间的指令均不执行。

MCR

MCR 为主控结束指令，置于主控程序最后，与 MC 主控指令配合使用，在 MCR 指令之前不可有接点指令，直接与母线相连。
参见 MC 指令。

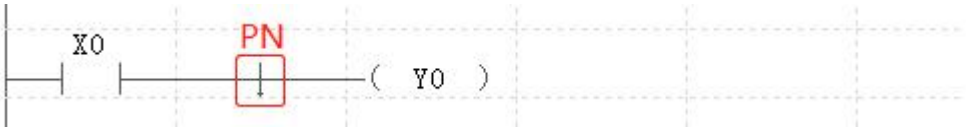
EU

指令说明：
EU(NP)指令当累积寄存器由 0 变为 1 时，此指令将使累积寄存器维持一次扫描周期的 1，然后第二次扫描周期之后，自动将累积寄存器改为 0。
操作数：
无
编程示例：



ED

指令说明：
ED(PN)指令当累积寄存器由 1 变为 0 时，此指令将使累积寄存器维持一次扫描周期的 1，然后第二次扫描周期之后，自动将累积寄存器改为 0。
操作数：
无
编程示例：



3.2 触点比较指令

以下指令均为两个数据之间的数据比较，将操作数 S1、S2 按指定条件进行比较，满足条件触点导通，不满足条件触点闭合。
指令格式均相同，示例：[LD= S1 S2]，操作数 S1、S2 类型相同。
指令均可与母线直接相连。

分类	指令	类型	功能	操作数类型
LD 指令	LD=	16 位	S1=S2 导通	KnX,KnY,KnM,KnS,T,C,D,K,H,V,Z,LV,DT,@

	LDD=	32 位		
	LD>	16 位	S1>S2 导通	
	LDD>	32 位		
	LD<	16 位	S1<S2 导通	
	LDD<	32 位		
	LD<>	16 位	S1≠S2 导通	
	LDD<>	32 位		
	LD<=	16 位	S1<=S2 导通	
	LDD<=	32 位		
	LD>=	16 位	S1>=S2 导通	
	LDD>=	32 位		
AND 指令	AND=	16 位	S1=S2 导通	KnX,KnY,KnM,KnS,T,C,D,K,H,V,Z,LV,DT,@
	ANDD=	32 位		
	AND>	16 位	S1>S2 导通	
	ANDD>	32 位		
	AND<	16 位	S1<S2 导通	
	ANDD<	32 位		
	AND<>	16 位	S1≠S2 导通	
	ANDD<>	32 位		
	AND<=	16 位	S1<=S2 导通	
	ANDD<=	32 位		
	AND>=	16 位	S1>=S2 导通	
	ANDD>=	32 位		
OR 指令	OR=	16 位	S1=S2 导通	KnX,KnY,KnM,KnS,T,C,D,K,H,V,Z,LV,DT,@
	ORD=	32 位		
	OR>	16 位	S1>S2 导通	
	ORD>	32 位		
	OR<	16 位	S1<S2 导通	
	ORD<	32 位		
	OR<>	16 位	S1≠S2 导通	
	ORD<>	32 位		
	OR<=	16 位	S1<=S2 导通	
	ORD<=	32 位		
	OR>=	16 位	S1>=S2 导通	
	ORD>=	32 位		
LD 逻辑运算	LD&	16 位	S1&S2≠0 导通	KnX,KnY,KnM,KnS,T,C,D,K,H,V,Z,LV,DT,@
	LDD&	32 位		
	LD|	16 位	S1|S2≠0 导通	
	LDD|	32 位		

	LD^	16 位	S1^S2≠0 导通	
	LDD^	32 位		
AND 逻辑运算	AND&	16 位	S1&S2≠0 导通	KnX,KnY,KnM,KnS,T,C,D,K,H,V,Z,LV,DT,@
	ANDD&	32 位		
	AND|	16 位	S1|S2≠0 导通	
	ANDD|	32 位		
	AND^	16 位	S1^S2≠0 导通	
	ANDD^	32 位		
OR 逻辑运算	OR&	16 位	S1&S2≠0 导通	KnX,KnY,KnM,KnS,T,C,D,K,H,V,Z,LV,DT,@
	ORD&	32 位		
	OR|	16 位	S1|S2≠0 导通	
	ORD|	32 位		
	OR^	16 位	S1^S2≠0 导通	
	ORD^	32 位		
LD 浮点运算	FLD=	32 位浮点型	S1=S2 导通	T,C,D,LV,DT,@
	FLD>	32 位浮点型	S1>S2 导通	
	FLD<	32 位浮点型	S1<S2 导通	
	FLD<>	32 位浮点型	S1≠S2 导通	
	FLD<=	32 位浮点型	S1<=S2 导通	
	FLD>=	32 位浮点型	S1>=S2 导通	
AND 浮点运算	FAND=	32 位浮点型	S1=S2 导通	T,C,D,LV,DT,@
	FAND>	32 位浮点型	S1>S2 导通	
	FAND<	32 位浮点型	S1<S2 导通	
	FAND<>	32 位浮点型	S1≠S2 导通	
	FAND<=	32 位浮点型	S1<=S2 导通	
	FAND>=	32 位浮点型	S1>=S2 导通	
OR 浮点运算	FOR=	32 位浮点型	S1=S2 导通	T,C,D,LV,DT,@
	FOR>	32 位浮点型	S1>S2 导通	

	FOR<	32 位浮点型	S1<S2 导通	
	FOR<>	32 位浮点型	S1≠S2 导通	
	FOR<=	32 位浮点型	S1<=S2 导通	
	FOR>=	32 位浮点型	S1>=S2 导通	

LD=

指令说明：

LD=是连接在母线上的 16 位触点比较指令，其作用是将 S1、S2 的内容按指定的条件进行比较，根据其结果来控制触点的导通或是不导通。

其实触点比较我们可以把它看成一个触点，当 S1=S2，触点是闭合状态。若 S1<S2 或 S1>S2，触点是断开状态。

比较指令	导通条件	不导通条件	条件成立	条件不成立
[= S1 S2]	S1 = S2	S1 < S2 S1 > S2	触点闭合	触点断开

操作数：

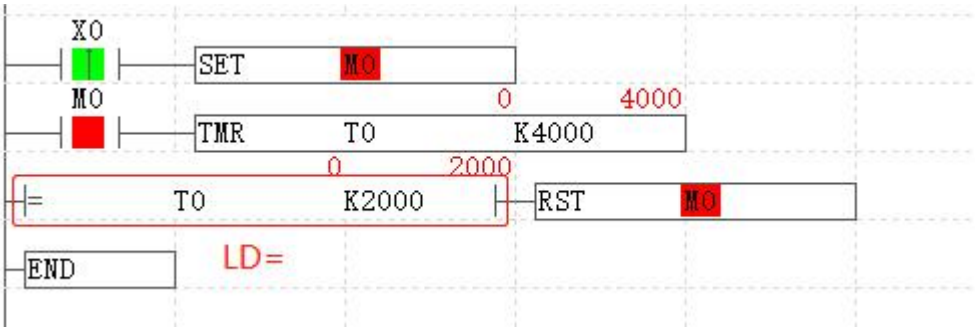
S1：KnX,KnY,KnM,KnS,T,C,D,K,H,V,Z,LV,DT,@

S2：KnX,KnY,KnM,KnS,T,C,D,K,H,V,Z,LV,DT,@

指令格式：[LD= S1 S2]

编程示例：

X0 闭合，M0 置位，T0 开始计时，当 T0 计时值等于 2000 时，LD=触点导通，M0 复位，定时器当前值清零。



LDD=

指令说明：

LDD=是连接在母线上的 32 位触点比较指令，其作用是将 S1、S2 的内容按指定的条件进行比较，根据其结果来控制触点的导通或是不导通。

LDD=指令同 LD=的用法一样，但比较数据范围超过 32767 时，就要用 32 位的比较指令 LDD=。

比较指令	导通条件	不导通条件	条件成立	条件不成立
[D = S1 S2]	S1 = S2	S1 < S2 S1 > S2	触点闭合	触点断开

操作数：

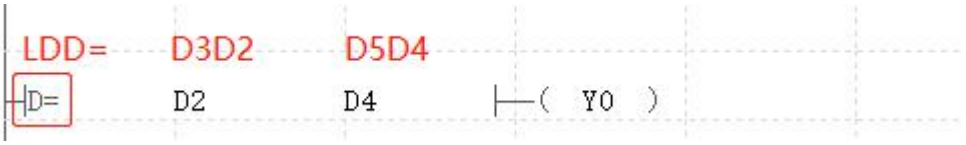
S1：KnX,KnY,KnM,KnS,T,C,D,K,H,V,Z,LV,DT,@

S2：KnX,KnY,KnM,KnS,T,C,D,K,H,V,Z,LV,DT,@

指令格式：[LDD= S1 S2]

编程示例：

32 位指令使用连续两个 16 位数据寄存器 D 空间，当 D3D2 的 32 位数据与 D5D4 相等时，LDD=触点导通，Y0 输出。



LD>

指令说明：

LD>是连接在母线上的 16 位触点比较指令，其作用是将 S1、S2 的内容按指定的条件进行比较，根据其结果来控制触点的导通或是不导通。

其实触点比较我们可以把它看成一个触点，当 S1>S2，触点是闭合状态。若 S1<S2 或 S1=S2，触点是断开状态。

比较指令	导通条件	不导通条件	条件成立	条件不成立
[> S1 S2]	S1 > S2	S1 = S2 S1 < S2	触点闭合	触点断开

操作数：

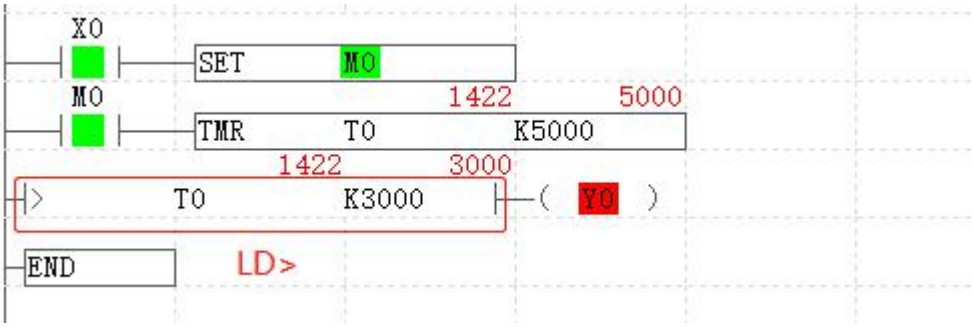
S1：KnX,KnY,KnM,KnS,T,C,D,K,H,V,Z,LV,DT,@

S2：KnX,KnY,KnM,KnS,T,C,D,K,H,V,Z,LV,DT,@

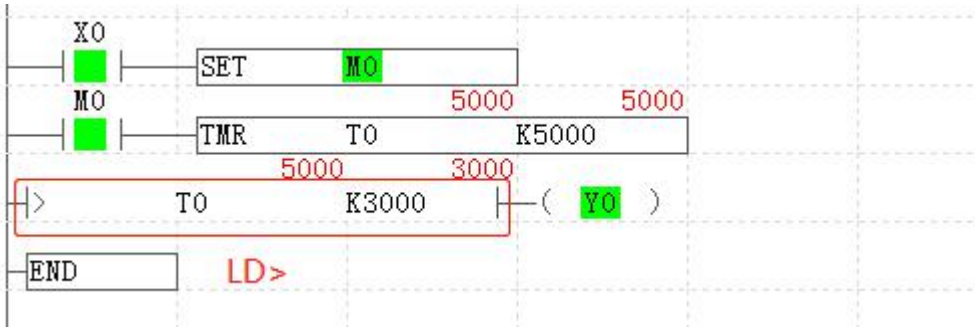
指令格式：[LD> S1 S2]

编程示例：

X0 闭合，M0 置位，T0 开始计时。不满足条件时，LD>触点不导通。



当 T0 的值大于 3000 时，比较触点导通，输出 Y0。



LDD>

指令说明：

LDD>是连接在母线上的 32 位触点比较指令，其作用是将 S1、S2 的内容按指定的条件进行比较，根据其结果来控制触点的导通或是不导通。

LDD>指令同 LD>的用法一样，但比较的数据范围超过 32767 时，就要用 32 位的比较指令 LDD>。

比较指令	导通条件	不导通条件	条件成立	条件不成立
[D> S1 S2]	S1 > S2	S1 = S2 S1 < S2	触点闭合	触点断开

操作数：

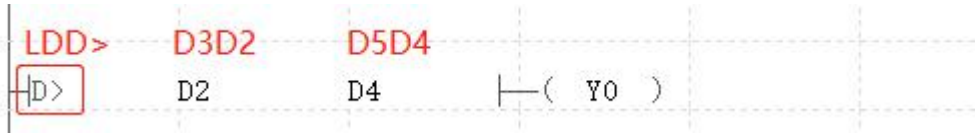
S1：KnX,KnY,KnM,KnS,T,C,D,K,H,V,Z,LV,DT,@

S2：KnX,KnY,KnM,KnS,T,C,D,K,H,V,Z,LV,DT,@

指令格式：[LDD> S1 S2]

编程示例：

32 位指令使用连续两个 16 位数据寄存器 D 空间，当 D3D2 的数据大于 D5D4 时，该触点导通，Y0 输出。



LD<

指令说明：

LD<是连接在母线上的 16 位触点比较指令，其作用是将 S1、S2 的内容按指定的条件进行比较，根据其结果来控制触点的导通或是不导通。

其实触点比较我们可以把它看成一个触点，当 S1<S2，触点是闭合状态。若 S1=S2 或是 S1>S2，触点是断开状态。

比较指令	导通条件	不导通条件	条件成立	条件不成立
[< S1 S2]	S1 < S2	S1 = S2	触点闭合	触点断开

		S1 > S2		

操作数：

S1：KnX,KnY,KnM,KnS,T,C,D,K,H,V,Z,LV,DT,@

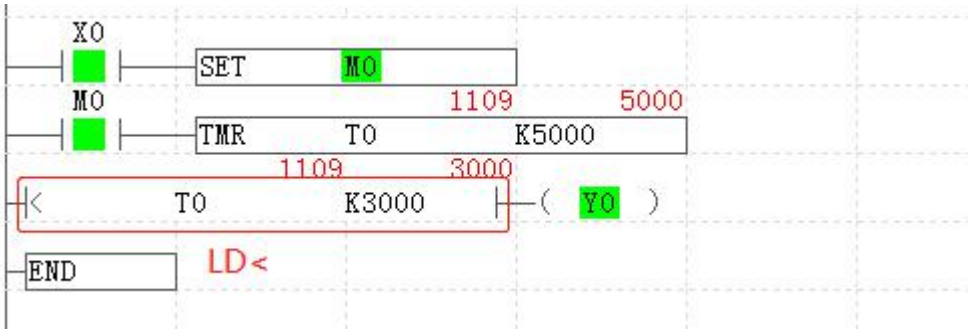
S2：KnX,KnY,KnM,KnS,T,C,D,K,H,V,Z,LV,DT,@

指令格式：[LD< S1 S2]

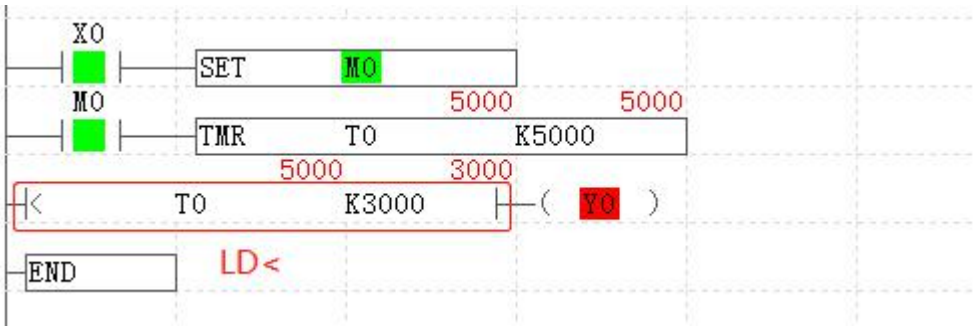
编程示例：

上电后，X0 断开，M0=OFF，T0 未启动，值为 0，LD<条件成立，触点闭合，Y0 有输出。

将 X0 闭合，M0 导通，T0 开始计时，计时值小于 3000 时，LD<条件仍成立，触点为闭合状态。



当计时值到达 3000 时，LD<条件不成立，触点断开，Y0 断开。



LDD<

指令说明：

LDD<是连接在母线上的 32 位触点比较指令，其作用是将 S1、S2 的内容按指定的条件进行比较，根据其结果来控制触点的导通或是不导通。

LDD<指令同 LD<的用法一样，但比较的数据范围超过 32767 时，就要用 32 位的比较指令 LDD<。

比较指令	导通条件	不导通条件	条件成立	条件不成立
[D< S1 S2]	S1 < S2	S1 = S2 S1 > S2	触点闭合	触点断开

操作数：

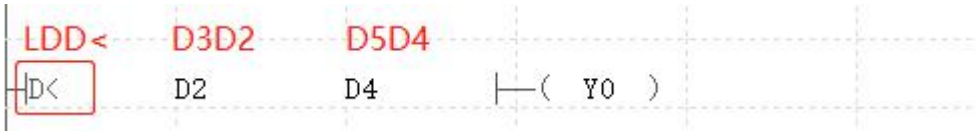
S1：KnX,KnY,KnM,KnS,T,C,D,K,H,V,Z,LV,DT,@

S2：KnX,KnY,KnM,KnS,T,C,D,K,H,V,Z,LV,DT,@

指令格式：[LDD< S1 S2]

编程示例：

32 位指令使用连续两个 16 位数据寄存器 D 空间，当 D3D2 的数据小于 D5D4 时，该触点导通，Y0 输出。



LD<>

指令说明：

LD<>是连接在母线上的 16 位触点比较指令，其作用是将 S1、S2 的内容按指定的条件进行比较，根据其结果来控制触点的导通或是不导通。

其实触点比较我们可以把它看成一个触点，当 S1<S2 或 S1>S2 时，触点闭合。若 S1=S2 时，触点断开。

比较指令	导通条件	不导通条件	条件成立	条件不成立
[< > S1 S2]	S1 < S2 S1 > S2	S1 = S2	触点闭合	触点断开

操作数：

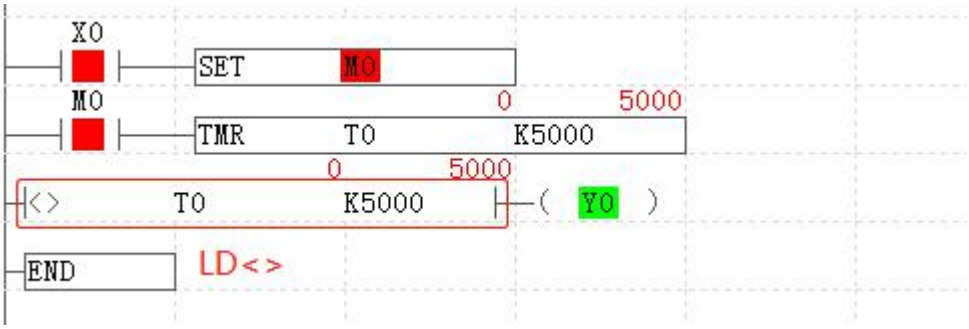
S1：KnX,KnY,KnM,KnS,T,C,D,K,H,V,Z,LV,DT,@

S2：KnX,KnY,KnM,KnS,T,C,D,K,H,V,Z,LV,DT,@

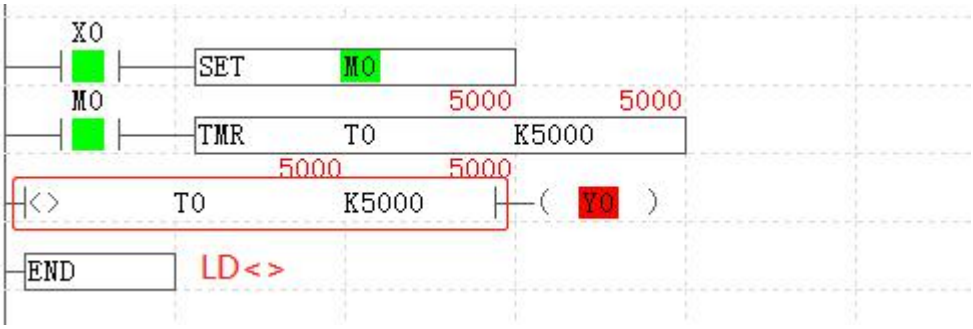
指令格式：[LD< > S1 S2]

编程示例：

X0 未接通时，T0 的值为 0，LD<>的比较触点在 T0≠5000 时，一直导通，Y0 输出。



X0 接通，置位 M0，常开触点 M0 导通，T0 开始计时，当 T0 计时等于 5000 时，LD<>的比较触点断开，Y0 无输出。



LDD<>

指令说明：

LDD<>是连接在母线上的 32 位触点比较指令，其作用是将 S1、S2 的内容按指定的条件进行比较，根据其结果来控制触点的导通或是不导通。

LDD<>指令同 LD<>的用法一样，但比较的数据范围超过 32767 时，就要用 32 位的比较指令 LDD<>。

比较指令	导通条件	不导通条件	条件成立	条件不成立
[D< > S1 S2]	S1 < S2 S1 > S2	S1 = S2	触点闭合	触点断开

操作数：

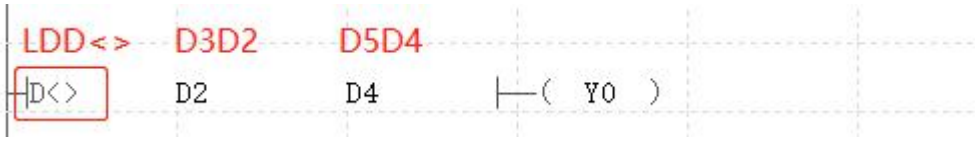
S1：KnX,KnY,KnM,KnS,T,C,D,K,H,V,Z,LV,DT,@

S2：KnX,KnY,KnM,KnS,T,C,D,K,H,V,Z,LV,DT,@

指令格式：[LDD<> S1 S2]

编程示例：

32 位指令使用连续两个 16 位数据寄存器 D 空间，当 D3D2 的数值不等于 D5D4 时，该触点导通，Y0 输出。



LD<=

指令说明：

LD<=是连接在母线上的 16 位触点比较指令，其作用是将 S1、S2 的内容按指定的条件进行比较，根据其结果来控制触点的导通或是不导通。

其实触点比较我们可以把它看成一个触点，当 S1<S2 或 S1=S2 时，触点闭合。若 S1>S2 时，触点断开。

比较指令	导通条件	不导通条件	条件成立	条件不成立
[< = S1 S2]	S1 < S2 S1 = S2	S1 > S2	触点闭合	触点断开

操作数：

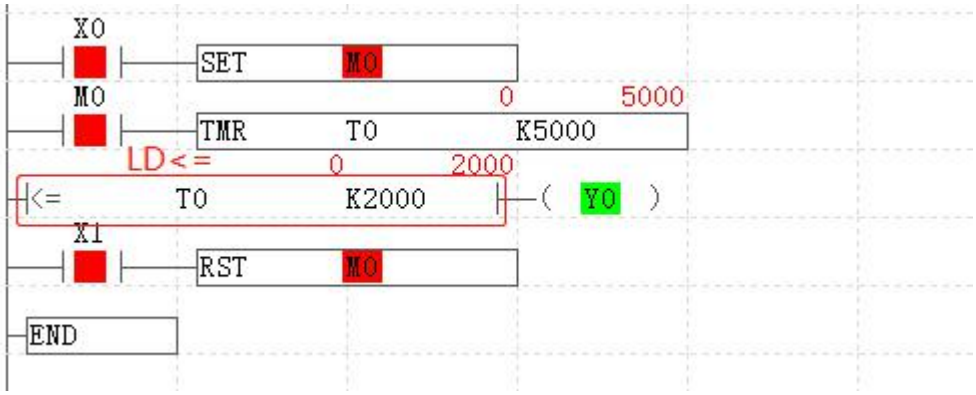
S1：KnX,KnY,KnM,KnS,T,C,D,K,H,V,Z,LV,DT,@

S2：KnX,KnY,KnM,KnS,T,C,D,K,H,V,Z,LV,DT,@

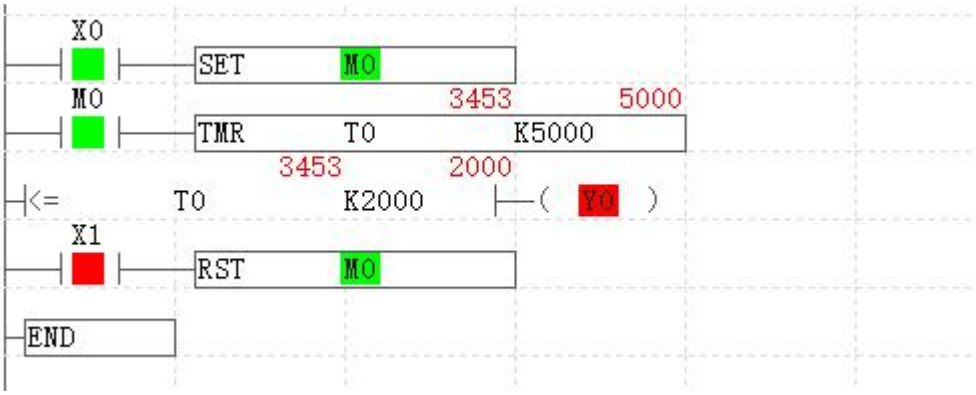
指令格式：[LD<= S1 S2]

编程示例：

X0 未接通时，T0 的值为 0，比较触点在 T0<=2000 时，一直导通，Y0 输出。



X0 接通，置位 M0，常开触点 M0 导通，T0 开始计，当 T0>2000 时，比较触点断开，Y0 无输出。



LDD<=

指令说明：

LDD<=是连接在母线上的 32 位触点比较指令，其作用是将 S1、S2 的内容按指定的条件进行比较，根据其结果来控制触点的导通或是不导通。

LDD<=指令同 LD<=的用法一样，但比较的数据范围超过 32767 时，就要用 32 位的比较指令 LDD<=。

比较指令	导通条件	不导通条件	条件成立	条件不成立
[D< = S1 S2]	S1 < S2 S1 = S2	S1 > S2	触点闭合	触点断开

操作数：

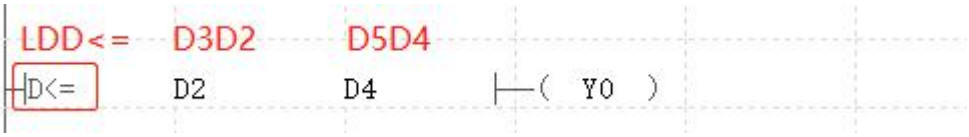
S1：KnX,KnY,KnM,KnS,T,C,D,K,H,V,Z,LV,DT,@

S2：KnX,KnY,KnM,KnS,T,C,D,K,H,V,Z,LV,DT,@

指令格式：[LDD<= S1 S2]

编程示例：

32 位指令使用连续两个 16 位数据寄存器 D 空间，当 D3D2 的数值小于或等于 D5D4 时，该触点导通，Y0 输出。



LD>=

指令说明：

LD>=是连接在母线上的 16 位触点比较指令，其作用是将 S1、S2 的内容按指定的条件进行比较，根据其结果来控制触点的导通或是不导通。

其实触点比较我们可以把它看成一个触点，当 S1>S2 或 S1=S2 时，触点闭合。若 S1<S2 时，触点断开。

比较指令	导通条件	不导通条件	条件成立	条件不成立
[>= S1 S2]	S1 > S2 S1 = S2	S1 < S2	触点闭合	触点断开

操作数：

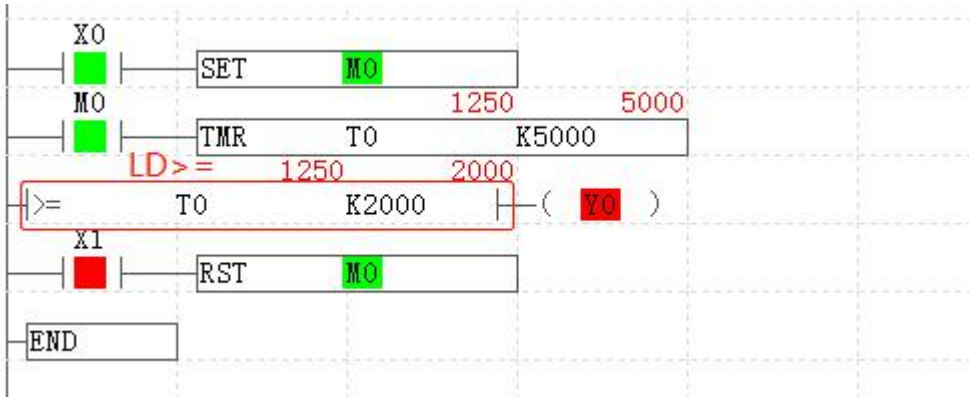
S1：KnX,KnY,KnM,KnS,T,C,D,K,H,V,Z,LV,DT,@

S2：KnX,KnY,KnM,KnS,T,C,D,K,H,V,Z,LV,DT,@

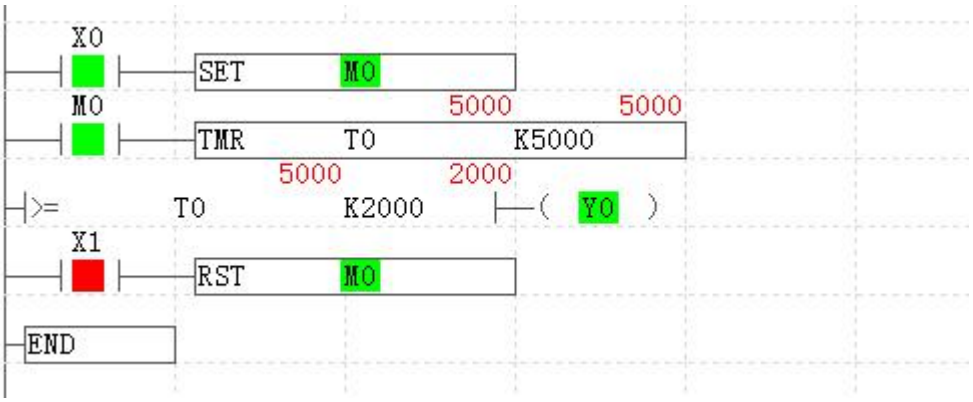
指令格式：[LD>= S1 S2]

编程示例：

X0 接通，置位 M0，常开触点 M0 导通。T0 开始计时，比较触点在 T0<2000 时，LD>=比较触点不满足条件，触点断开，Y0 无输出。



但当 T0>=2000 时，LD>=比较触点满足条件，触点闭合，Y0 输出。



LDD>=

指令说明：

LDD>=是连接在母线上的 32 位触点比较指令，其作用是将 S1、S2 的内容按指定的条件进行比较，根据其结果来控制触点的导通或是不导通。

LDD>=指令同 LD>=的用法一样，但比较的数据范围超过 32767 时，就要用 32 位的比较指令 LDD>=。

比较指令	导通条件	不导通条件	条件成立	条件不成立
[D>= S1 S2]	S1 > S2 S1 = S2	S1 < S2	触点闭合	触点断开

操作数：

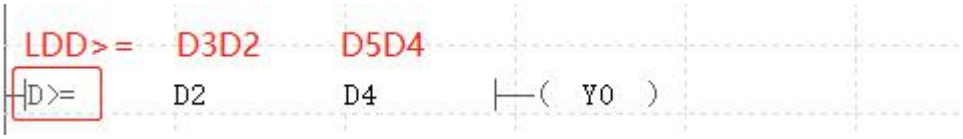
S1：KnX,KnY,KnM,KnS,T,C,D,K,H,V,Z,LV,DT,@

S2：KnX,KnY,KnM,KnS,T,C,D,K,H,V,Z,LV,DT,@

指令格式：[LDD>= S1 S2]

编程示例：

32 位指令使用连续两个 16 位数据寄存器 D 空间，当 D3D2 的数值大于或等于 D5D4 时，该触点导通，Y0 输出。



AND=

指令说明：

AND=是与其它触点串联的 16 位触点比较指令，其作用是将 S1、S2 的内容按指定的条件进行比较，根据其结果来控制触点的导通或是不导通。

其实触点比较我们可以把它看成一个触点，当 S1=S2 时，触点闭合。若 S1<S2 或 S1>S2 时，触点断开。

比较指令	导通条件	不导通条件	条件成立	条件不成立
[= S1 S2]	S1 = S2	S1 < S2 S1 > S2	触点闭合	触点断开

操作数：

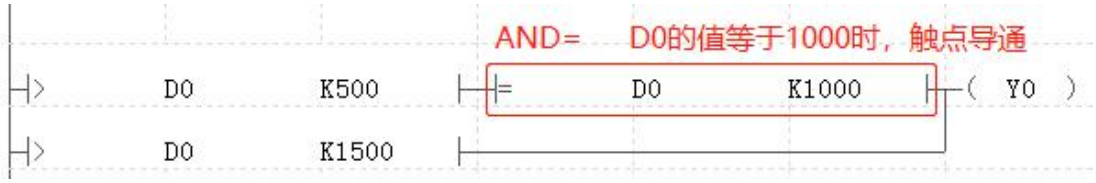
S1：KnX,KnY,KnM,KnS,T,C,D,K,H,V,Z,LV,DT,@

S2：KnX,KnY,KnM,KnS,T,C,D,K,H,V,Z,LV,DT,@

指令格式：[AND= S1 S2]

编程示例：

当 D0 的值等于 1000 时，AND=触点导通，否则不导通。



ANDD=

指令说明：

ANDD=是与其它触点串联的 32 位触点比较指令，其作用是将 S1、S2 的内容按指定的条件进行比较，根据其结果来控制触点的导通或是不导通。

ANDD=指令同 AND=的用法一样，但比较的数据范围超过 32767 时，就要用 32 位的比较指令 ANDD=。

比较指令	导通条件	不导通条件	条件成立	条件不成立
[D= S1 S2]	S1 = S2	S1 < S2 S1 > S2	触点闭合	触点断开

操作数：

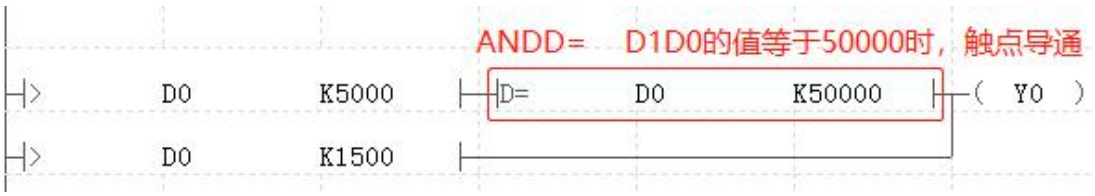
S1：KnX,KnY,KnM,KnS,T,C,D,K,H,V,Z,LV,DT,@

S2：KnX,KnY,KnM,KnS,T,C,D,K,H,V,Z,LV,DT,@

指令格式：[ANDD= S1 S2]

编程示例：

当 D1D0 的值为 50000 时，ANDD=触点导通，否则不导通。



AND>

指令说明：

AND>是与其它触点串联的 16 位触点比较指令，其作用是将 S1、S2 的内容按指定的条件进行比较，根据其结果来控制触点的导通或是不导通。

其实触点比较我们可以把它看成一个触点，当 S1>S2 时，触点闭合。若 S1<S2 或 S1=S2 时，触点断开。

比较指令	导通条件	不导通条件	条件成立	条件不成立
[> S1 S2]	S1 > S2	S1 < S2 S1 = S2	触点闭合	触点断开

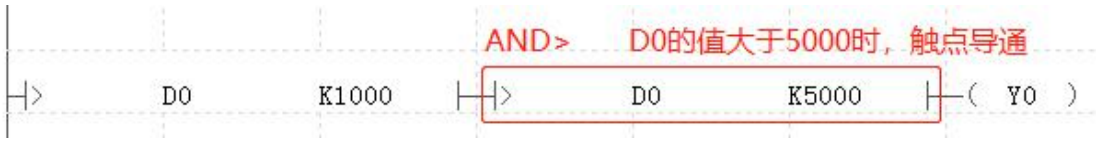
操作数：

S1：KnX,KnY,KnM,KnS,T,C,D,K,H,V,Z,LV,DT,@

S2：KnX,KnY,KnM,KnS,T,C,D,K,H,V,Z,LV,DT,@

指令格式：[AND> S1 S2]

编程示例：



ANDD>

指令说明：

ANDD>是与其它触点串联的 32 位触点比较指令，其作用是将 S1、S2 的内容按指定的条件进行比较，根据其结果来控制触点的导通或是不导通。

ANDD>指令同 AND>的用法一样，但比较的数据范围超过 32767 时，就要用 32 位的比较指令 ANDD>。

比较指令	导通条件	不导通条件	条件成立	条件不成立
[D> S1 S2]	S1 > S2	S1 < S2 S1 = S2	触点闭合	触点断开

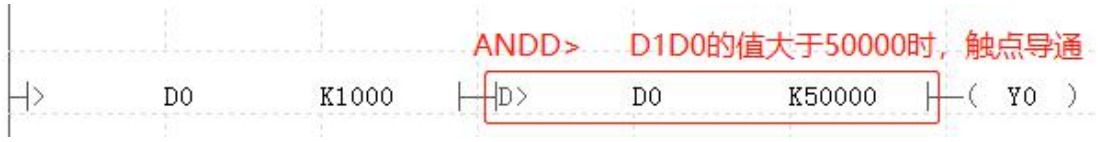
操作数：

S1：KnX,KnY,KnM,KnS,T,C,D,K,H,V,Z,LV,DT,@

S2：KnX,KnY,KnM,KnS,T,C,D,K,H,V,Z,LV,DT,@

指令格式：[ANDD> S1 S2]

编程示例：



AND<

指令说明：

AND<是与其它触点串联的 16 位触点比较指令，其作用是将 S1、S2 的内容按指定的条件进行比较，根据其结果来控制触点的导通或是不导通。

其实触点比较我们可以把它看成一个触点，当 S1<S2 时，触点闭合。若 S1>S2 或 S1=S2 时，触点断开。

比较指令	导通条件	不导通条件	条件成立	条件不成立
[< S1 S2]	S1 < S2	S1 > S2 S1 = S2	触点闭合	触点断开

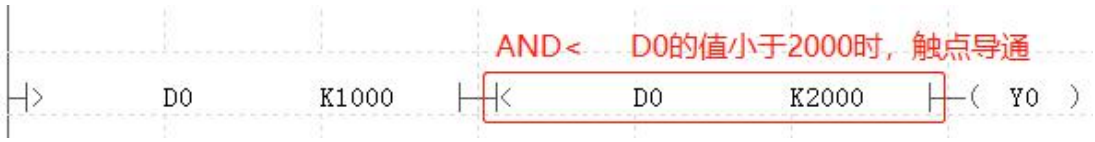
操作数：

S1：KnX,KnY,KnM,KnS,T,C,D,K,H,V,Z,LV,DT,@

S2：KnX,KnY,KnM,KnS,T,C,D,K,H,V,Z,LV,DT,@

指令格式：[AND< S1 S2]

编程示例：



ANDD<

指令说明：

ANDD<是与其它触点串联的 32 位触点比较指令，其作用是将 S1、S2 的内容按指定的条件进行比较，根据其结果来控制触点的导通或是不导通。

ANDD<指令同 AND<的用法一样，但比较的数据范围超过 32767 时，就要用 32 位的比较指令 ANDD<。

比较指令	导通条件	不导通条件	条件成立	条件不成立
[D< S1 S2]	S1 < S2	S1 > S2 S1 = S2	触点闭合	触点断开

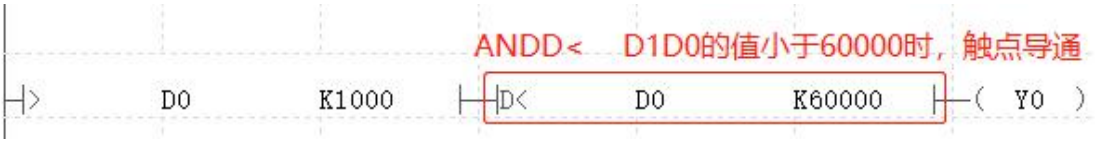
操作数：

S1：KnX,KnY,KnM,KnS,T,C,D,K,H,V,Z,LV,DT,@

S2：KnX,KnY,KnM,KnS,T,C,D,K,H,V,Z,LV,DT,@

指令格式：[ANDD< S1 S2]

编程示例：



AND<>

指令说明：

AND<>是与其它触点串联的 16 位触点比较指令，其作用是将 S1、S2 的内容按指定的条件进行比较，根据其结果来控制触点的导通或是不导通。

其实触点比较我们可以把它看成一个触点，当 S1<S2 或 S1>S2 时，触点闭合。若 S1=S2 时，触点断开。

比较指令	导通条件	不导通条件	条件成立	条件不成立
[< > S1 S2]	S1 < S2 S1 > S2	S1 = S2	触点闭合	触点断开

操作数：

S1：KnX,KnY,KnM,KnS,T,C,D,K,H,V,Z,LV,DT,@

S2：KnX,KnY,KnM,KnS,T,C,D,K,H,V,Z,LV,DT,@

指令格式：[AND< > S1 S2]

编程示例：

ANDD<>

指令说明：

ANDD<>是与其它触点串联的 32 位触点比较指令，其作用是将 S1、S2 的内容按指定的条件进行比较，根据其结果来控制触点的导通或是不导通。

ANDD<>指令同 AND<>的用法一样，但比较的数据范围超过 32767 时，就要用 32 位的比较指令 ANDD<>。

比较指令	导通条件	不导通条件	条件成立	条件不成立
[D < > S1 S2]	S1 < S2 S1 > S2	S1 = S2	触点闭合	触点断开

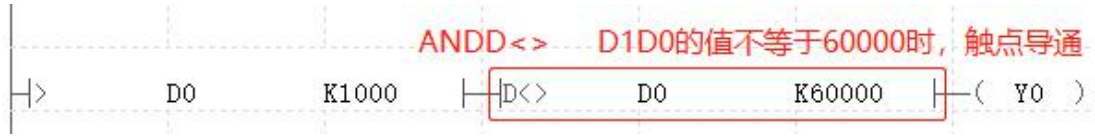
操作数：

S1：KnX,KnY,KnM,KnS,T,C,D,K,H,V,Z,LV,DT,@

S2：KnX,KnY,KnM,KnS,T,C,D,K,H,V,Z,LV,DT,@

指令格式：[ANDD< > S1 S2]

编程示例：



AND<=

指令说明：

AND<=是与其它触点串联的 16 位触点比较指令，其作用是将 S1、S2 的内容按指定的

条件进行比较，根据其结果来控制触点的导通或是不导通。

其实触点比较我们可以把它看成一个触点，当 S1<S2 或 S1=S2 时，触点闭合。若 S1>S2 时，触点断开。

比较指令	导通条件	不导通条件	条件成立	条件不成立
[< = S1 S2]	S1 < S2 S1 = S2	S1 > S2	触点闭合	触点断开

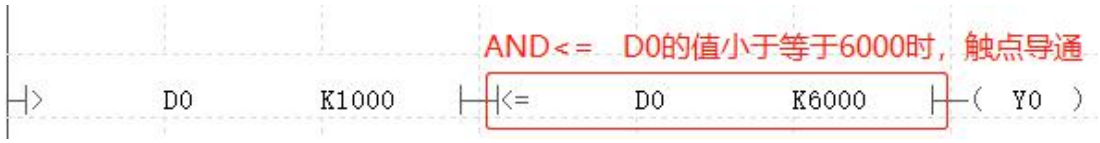
操作数：

S1：KnX,KnY,KnM,KnS,T,C,D,K,H,V,Z,LV,DT,@

S2：KnX,KnY,KnM,KnS,T,C,D,K,H,V,Z,LV,DT,@

指令格式：[AND<= S1 S2]

编程示例：



ANDD<=

指令说明：

ANDD<=是与其它触点串联的 32 位触点比较指令，其作用是将 S1、S2 的内容按指定的条件进行比较，根据其结果来控制触点的导通或是不导通。

ANDD<=指令同 AND<=的用法一样，但比较的数据范围超过 32767 时，就要用 32 位的比较指令 ANDD<=。

比较指令	导通条件	不导通条件	条件成立	条件不成立
[D < = S1 S2]	S1 < S2 S1 = S2	S1 > S2	触点闭合	触点断开

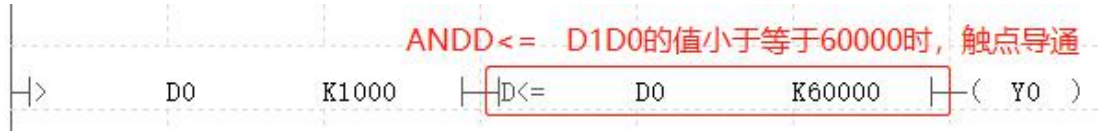
操作数：

S1：KnX,KnY,KnM,KnS,T,C,D,K,H,V,Z,LV,DT,@

S2：KnX,KnY,KnM,KnS,T,C,D,K,H,V,Z,LV,DT,@

指令格式：[ANDD<= S1 S2]

编程示例：



AND>=

指令说明：

AND>=是与其它触点串联的 16 位触点比较指令，其作用是将 S1、S2 的内容按指定的

条件进行比较，根据其结果来控制触点的导通或是不导通。

其实触点比较我们可以把它看成一个触点，当 S1>S2 或 S1=S2 时，触点闭合。若 S1<S2 时，触点断开。

比较指令	导通条件	不导通条件	条件成立	条件不成立
[> = S1 S2]	S1 >S2 S1 = S2	S1 < S2	触点闭合	触点断开

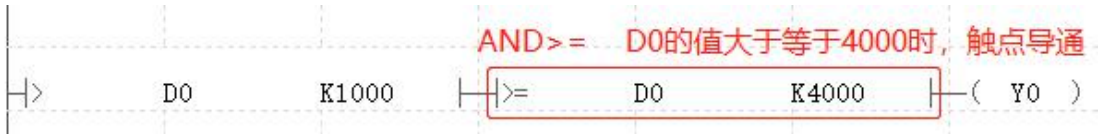
操作数：

S1：KnX,KnY,KnM,KnS,T,C,D,K,H,V,Z,LV,DT,@

S2：KnX,KnY,KnM,KnS,T,C,D,K,H,V,Z,LV,DT,@

指令格式：[AND>= S1 S2]

编程示例：



ANDD>=

指令说明：

AND>=是与其它触点串联的 32 位触点比较指令，其作用是将 S1、S2 的内容按指定的条件进行比较，根据其结果来控制触点的导通或是不导通。

ANDD>=指令同 AND>=的用法一样，但比较的数据范围超过 32767 时，就要用 32 位的比较指令 ANDD>=。

比较指令	导通条件	不导通条件	条件成立	条件不成立
[D > = S1 S2]	S1 > S2 S1 = S2	S1 < S2	触点闭合	触点断开

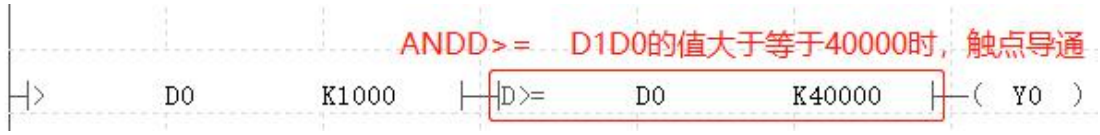
操作数：

S1：KnX,KnY,KnM,KnS,T,C,D,K,H,V,Z,LV,DT,@

S2：KnX,KnY,KnM,KnS,T,C,D,K,H,V,Z,LV,DT,@

指令格式：[ANDD>= S1 S2]

编程示例：



OR=

指令说明：

OR=是与其它触点并联的 16 位触点比较指令，其作用是将 S1、S2 的内容按指定的条

件进行比较，根据其结果来控制触点的导通或是不导通。

其实触点比较我们可以把它看成一个触点，当 S1=S2 时，触点闭合。若 S1<S2 或 S1>S2 时，触点断开。

比较指令	导通条件	不导通条件	条件成立	条件不成立
[= S1 S2]	S1 = S2	S1 < S2 S1 > S2	触点闭合	触点断开

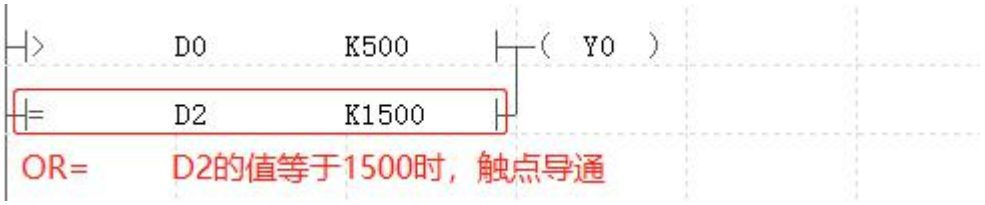
操作数：

S1：KnX,KnY,KnM,KnS,T,C,D,K,H,V,Z,LV,DT,@

S2：KnX,KnY,KnM,KnS,T,C,D,K,H,V,Z,LV,DT,@

指令格式：[OR= S1 S2]

编程示例：



ORD=

指令说明：

ORD=是与其它触点并联的 32 位触点比较指令，其作用是将 S1、S2 的内容按指定的条件进行比较，根据其结果来控制触点的导通或是不导通。

ORD=指令同 OR=的用法一样，但比较的数据范围超过 32767 时，就要用 32 位的比较指令 ORD=。

比较指令	导通条件	不导通条件	条件成立	条件不成立
[D = S1 S2]	S1 = S2	S1 < S2 S1 > S2	触点闭合	触点断开

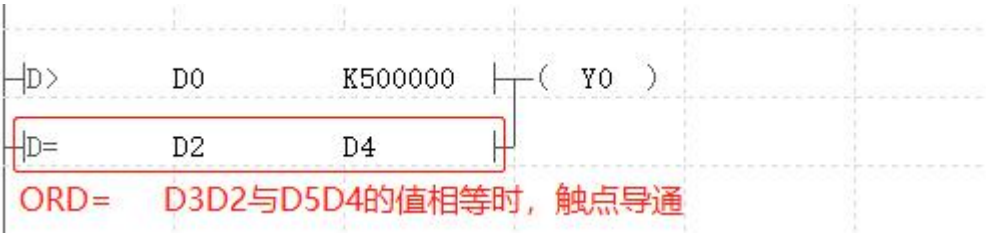
操作数：

S1：KnX,KnY,KnM,KnS,T,C,D,K,H,V,Z,LV,DT,@

S2：KnX,KnY,KnM,KnS,T,C,D,K,H,V,Z,LV,DT,@

指令格式：[ORD= S1 S2]

编程示例：



OR>

指令说明：

OR>是与其它触点并联的 16 位触点比较指令，其作用是将 S1、S2 的内容按指定的条件进行比较，根据其结果来控制触点的导通或是不导通。

其实触点比较我们可以把它看成一个触点，当 S1>S2 时，触点闭合。若 S1<S2 或 S1=S2 时，触点断开。

比较指令	导通条件	不导通条件	条件成立	条件不成立
[> S1 S2]	S1 > S2	S1 < S2 S1 = S2	触点闭合	触点断开

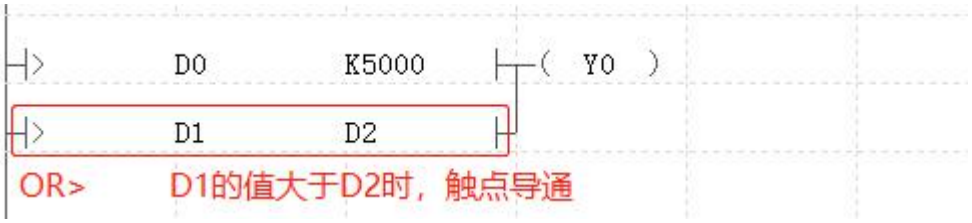
操作数：

S1：KnX,KnY,KnM,KnS,T,C,D,K,H,V,Z,LV,DT,@

S2：KnX,KnY,KnM,KnS,T,C,D,K,H,V,Z,LV,DT,@

指令格式：[OR> S1 S2]

编程示例：



ORD>

指令说明：

ORD>是与其它触点并联的 32 位触点比较指令，其作用是将 S1、S2 的内容按指定的条件进行比较，根据其结果来控制触点的导通或是不导通。

ORD>指令同 OR>的用法一样，但比较的数据范围超过 32767 时，就要用 32 位的比较指令 ORD>。

比较指令	导通条件	不导通条件	条件成立	条件不成立
[> S1 S2]	S1 > S2	S1 < S2 S1 = S2	触点闭合	触点断开

操作数：

S1：KnX,KnY,KnM,KnS,T,C,D,K,H,V,Z,LV,DT,@

S2：KnX,KnY,KnM,KnS,T,C,D,K,H,V,Z,LV,DT,@

指令格式：[ORD> S1 S2]

编程示例：

OR<

指令说明：

OR<是与其它触点并联的 16 位触点比较指令，其作用是将 S1、S2 的内容按指定的条件进行比较，根据其结果来控制触点的导通或是不导通。

其实触点比较我们可以把它看成一个触点，当 S1<S2 时，触点闭合。若 S1>S2 或 S1=S2 时，触点断开。

比较指令	导通条件	不导通条件	条件成立	条件不成立
[< S1 S2]	S1 < S2	S1 > S2 S1 = S2	触点闭合	触点断开

操作数：

S1：KnX,KnY,KnM,KnS,T,C,D,K,H,V,Z,LV,DT,@

S2：KnX,KnY,KnM,KnS,T,C,D,K,H,V,Z,LV,DT,@

指令格式：[OR< S1 S2]

编程示例：

ORD<

指令说明：

ORD<是与其它触点并联的 32 位触点比较指令，其作用是将 S1、S2 的内容按指定的条件进行比较，根据其结果来控制触点的导通或是不导通。

ORD<指令同 OR<的用法一样，但比较的数据范围超过 32767 时，就要用 32 位的比较指令 ORD<。

比较指令	导通条件	不导通条件	条件成立	条件不成立
[< S1 S2]	S1 < S2	S1 > S2 S1 = S2	触点闭合	触点断开

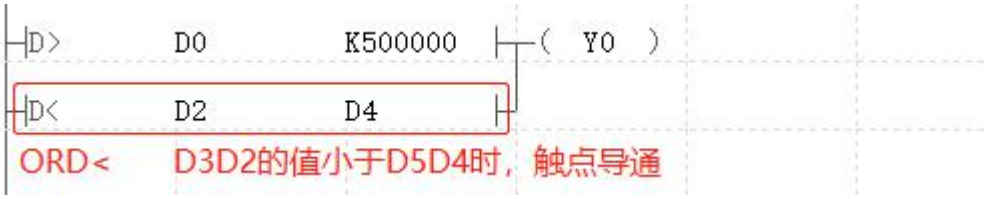
操作数：

S1：KnX,KnY,KnM,KnS,T,C,D,K,H,V,Z,LV,DT,@

S2：KnX,KnY,KnM,KnS,T,C,D,K,H,V,Z,LV,DT,@

指令格式：[ORD< S1 S2]

编程示例：



OR<>

指令说明：

OR<>是与其它触点并联的 16 位触点比较指令，其作用是将 S1、S2 的内容按指定的条件进行比较，根据其结果来控制触点的导通或是不导通。

其实触点比较我们可以把它看成一个触点，当 S1<S2 或 S1>S2 时，触点闭合。若 S1=S2 时，触点断开。

比较指令	导通条件	不导通条件	条件成立	条件不成立
[< > S1 S2]	S1 < S2 S1 > S2	S1 = S2	触点闭合	触点断开

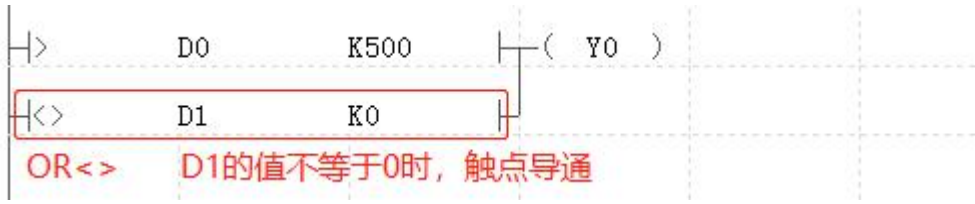
操作数：

S1：KnX,KnY,KnM,KnS,T,C,D,K,H,V,Z,LV,DT,@

S2：KnX,KnY,KnM,KnS,T,C,D,K,H,V,Z,LV,DT,@

指令格式：[OR<> S1 S2]

编程示例：



ORD<>

指令说明：

ORD<>是与其它触点并联的 32 位触点比较指令，其作用是将 S1、S2 的内容按指定的条件进行比较，根据其结果来控制触点的导通或是不导通。

ORD<>指令同 OR<>的用法一样，但比较的数据范围超过 32767 时，就要用 32 位的比较指令 ORD<>。

比较指令	导通条件	不导通条件	条件成立	条件不成立
[D < > S1 S2]	S1 < S2 S1 > S2	S1 = S2	触点闭合	触点断开

操作数：

S1：KnX,KnY,KnM,KnS,T,C,D,K,H,V,Z,LV,DT,@

S2：KnX,KnY,KnM,KnS,T,C,D,K,H,V,Z,LV,DT,@

指令格式：[ORD<>　S1　S2]

编程示例：

OR<=

指令说明：

OR<=是与其它触点并联的 16 位触点比较指令，其作用是将 S1、S2 的内容按指定的条件进行比较，根据其结果来控制触点的导通或是不导通。

其实触点比较我们可以把它看成一个触点，当 S1<S2 或 S1=S2 时，触点闭合。若 S1>S2 时，触点断开。

比较指令	导通条件	不导通条件	条件成立	条件不成立
[<=　S1　S2]	S1 < S2 S1 = S2	S1 > S2	触点闭合	触点断开

操作数：

S1：KnX,KnY,KnM,KnS,T,C,D,K,H,V,Z,LV,DT,@

S2：KnX,KnY,KnM,KnS,T,C,D,K,H,V,Z,LV,DT,@

指令格式：[OR<=　S1　S2]

编程示例：

ORD<=

指令说明：

ORD<=是与其它触点并联的 32 位触点比较指令，其作用是将 S1、S2 的内容按指定的条件进行比较，根据其结果来控制触点的导通或是不导通。

ORD<=指令同 OR<=的用法一样，但比较的数据范围超过 32767 时，就要用 32 位的比较指令 ORD<=。

比较指令	导通条件	不导通条件	条件成立	条件不成立
[D <=　S1　S2]	S1 < S2 S1 = S2	S1 > S2	触点闭合	触点断开

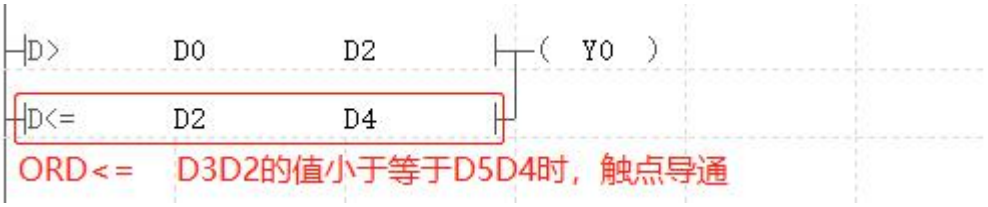
操作数：

S1：KnX,KnY,KnM,KnS,T,C,D,K,H,V,Z,LV,DT,@

S2：KnX,KnY,KnM,KnS,T,C,D,K,H,V,Z,LV,DT,@

指令格式：[ORD<= S1 S2]

编程示例：



OR>=

指令说明：

OR>=是与其它触点并联的 16 位触点比较指令，其作用是将 S1、S2 的内容按指定的条件进行比较，根据其结果来控制触点的导通或是不导通。

其实触点比较我们可以把它看成一个触点，当 S1>S2 或 S1=S2 时，触点闭合。若 S1<S2 时，触点断开。

比较指令	导通条件	不导通条件	条件成立	条件不成立
[>= S1 S2]	S1 > S2 S1 = S2	S1 < S2	触点闭合	触点断开

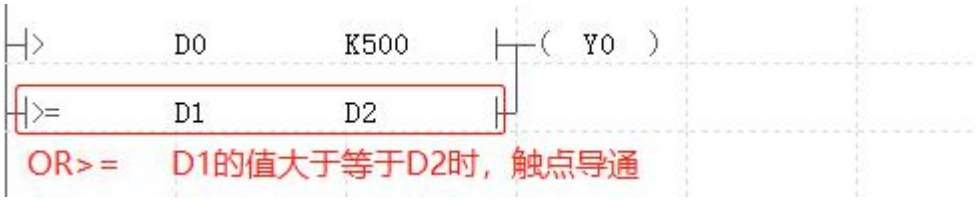
操作数：

S1：KnX,KnY,KnM,KnS,T,C,D,K,H,V,Z,LV,DT,@

S2：KnX,KnY,KnM,KnS,T,C,D,K,H,V,Z,LV,DT,@

指令格式：[OR>= S1 S2]

编程示例：



ORD>=

指令说明：

ORD>=是与其它触点并联的 32 位触点比较指令，其作用是将 S1、S2 的内容按指定的条件进行比较，根据其结果来控制触点的导通或是不导通。

ORD>=指令同 OR>=的用法一样，但比较的数据范围超过 32767 时，就要用 32 位的比较指令 ORD>=。

比较指令	导通条件	不导通条件	条件成立	条件不成立
[D >= S1 S2]	S1 > S2	S1 < S2	触点闭合	触点断开

	S1 = S2			

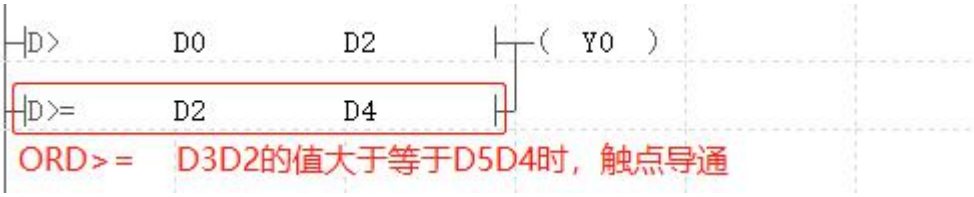
操作数：

S1：KnX,KnY,KnM,KnS,T,C,D,K,H,V,Z,LV,DT,@

S2：KnX,KnY,KnM,KnS,T,C,D,K,H,V,Z,LV,DT,@

指令格式：[ORD>= S1 S2]

编程示例：



LD#

指令说明：

触点逻辑运算指令 LD&、LDD&、LD|、LDD|、LD^、LDD^，S1 与 S2 的内容作位运算的指令，结果不为 0 时该指令导通，结果为 0 时该指令不导通。

16bit 指令	32bit 指令	导通条件	不导通条件
LD&	LDD&	S1 & S2 ≠ 0	S1 & S2 = 0
LD|	LDD|	S1 | S2 ≠ 0	S1 | S2 = 0
LD^	LDD^	S1 ^ S2 ≠ 0	S1 ^ S2 = 0

操作数：

S1：K, H, KnX, KnY, KnM, T, C, D,LV,DT,@

S2：K, H, KnX, KnY, KnM, T, C, D,LV,DT,@

逻辑运算规则：

运算	& 与				| 或				^ 异或			
操作数	0	0	1	1	0	0	1	1	0	0	1	1
	0	1	0	1	0	1	0	1	0	1	0	1
结果	0	0	0	1	0	1	1	1	0	1	1	0

LD&

指令说明：

LD&是直接与母线相连的 16 位连续执行型触点逻辑‘与’运算指令。是将 S1 和 S2 的二进制形式的每一个位上的二进制数字进行按位与运算之后的结果，如果结果不等于零时导通。

操作数：

S1：K, H, KnX, KnY, KnM, T, C, D,LV,DT,@

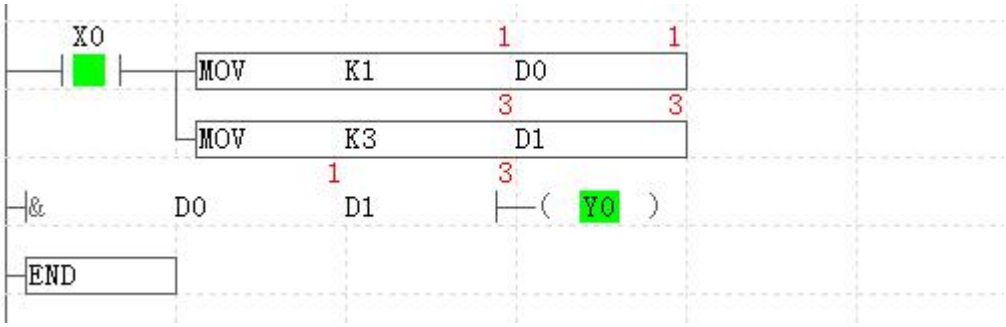
S2：K, H, KnX, KnY, KnM, T, C, D,LV,DT,@

指令格式：[LD& S1 S2]

编程示例：

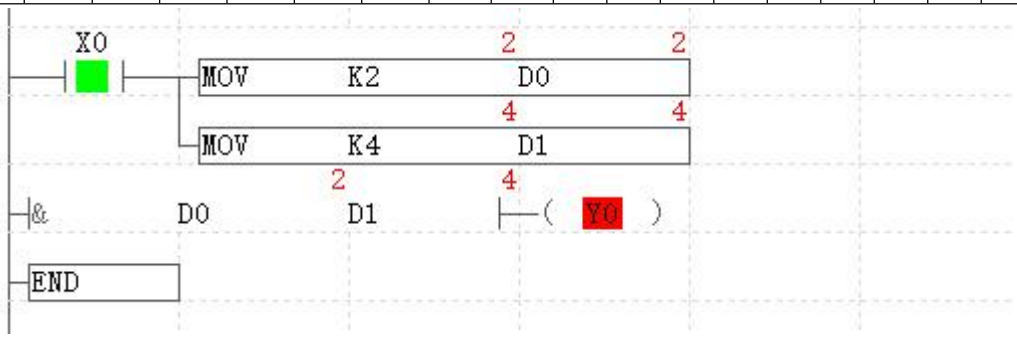
假设 D0 为 1，D1 为 3，那么 D0 的二进制数中的 b0 位为 1，其余位为 0，D1 的二进制数中的 b0 位为 1，b1 位为 1，其余位为 0。因为 D0 与 D1 的 b0 位都为 1，所以与运算后，该触点导通，Y0 有输出。

软元件	b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0	值
D0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1
D1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	3
与运算	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1



假设 D0 为 2，D1 为 4，那么 D0 的二进制数中的 b1 位为 1，其余位为 0，D1 的二进制数中的 b2 位为 1，其余位为 0。 因为 D0 与 D1 无相同的位都为 1，所以与运算后，该触点断开，Y0 无输出。

软元件	b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0	值
D0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	2
D1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	4
与运算	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0



LDD&

指令说明：

LDD&是直接与母线相连的 32 位连续执行型触点逻辑‘与’运算指令。是将 S1 和 S2 的二进制形式的每一个位上的二进制数字进行按位与运算之后的结果，如果结果不等于零时导通。

用法与 16 位指令 LD&相同，当运算的数据超过 16 位时，就要使用 32 位的与运算指令。

操作数：

S1：K, H, KnX, KnY, KnM,KnS, T, C, D,LV,DT,@

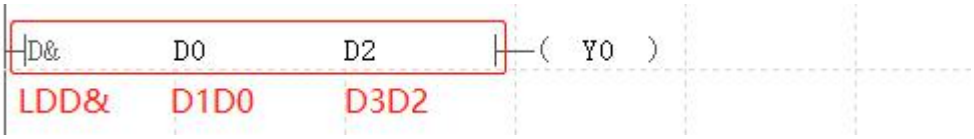
S2：K, H, KnX, KnY, KnM,KnS, T, C, D,LV,DT,@

指令格式：[LDD&　S1　S2]

编程示例：

若 D1D0 和 D3D2 表示的 32 位二进制数，若有任意一个相同的位值为 1，进行与运算后，结果不为 0，LDD&触点导通，输出 Y0。

当 D1D0 和 D3D2 没有相同位的值为 1 时，进行与运算结果为 0，LDD&触点不导通。



LD|

指令说明：

LD|是直接与母线相连的 16 位连续执行型触点逻辑‘或’运算指令。是将 S1 和 S2 的二进制形式的每一个位上的二进制数字进行按位或运算之后的结果，如果其十进制数结果不为 0 时导通。

操作数：

S1：K, H, KnX, KnY, KnM,KnS, T, C, D,LV,DT,@

S2：K, H, KnX, KnY, KnM,KnS, T, C, D,LV,DT,@

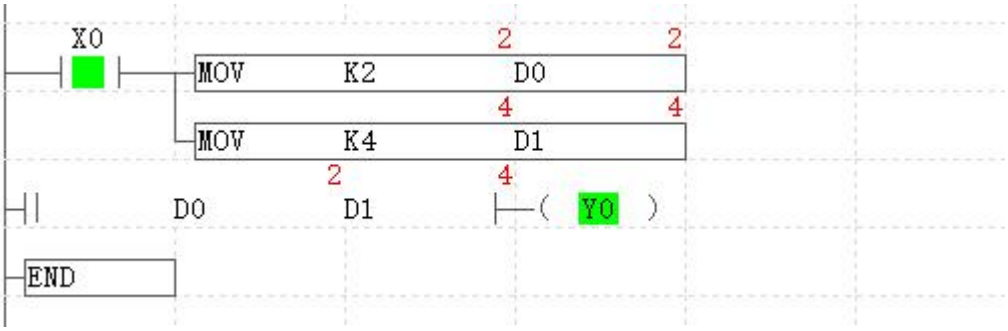
指令格式：[LD|　S1　S2]

编程示例：

下例中，D0 或者是 D1 任意一个寄存器的值不为 0，它们的二进制数中至少有一个位不为 0，进行或运算后，该触点导通，Y0 有输出。

D0 的值为 2，D1 的值为 4，或运算后，结果为 6，LD|触点导通，输出 Y0。

软元件	b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0	值
D0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	2
D1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	4
或运算	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	6



LDD|

指令说明：

LDD|是直接与母线相连的 32 位连续执行型触点逻辑‘或’运算指令。是将 S1 和 S2 的二进制形式的每一个位上的二进制数字进行按位或运算之后的结果，如果其十进制数结果不为 0 时导通。

用法与 16 位指令 LD|相同，当运算的数据超过 16 位时，就要使用 32 位的与运算指令。

操作数：

S1：K, H, KnX, KnY, KnM,KnS, T, C, D,LV,DT,@

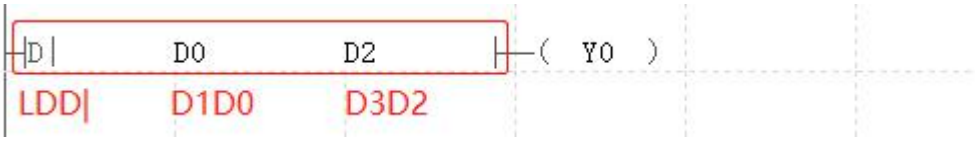
S2：K, H, KnX, KnY, KnM,KnS, T, C, D,LV,DT,@

指令格式：[LDD|　S1　S2]

编程示例：

若 D1D0 和 D3D2 任意一个寄存器的值不为 0，它们的二进制数中至少有一个位不为 0，进行或运算后，LDD|触点导通，Y0 有输出。

仅当 D1D0 和 D3D2 的值均为 0 时，进行或运算结果也为 0，LDD|触点不导通。



LD^

指令说明：

LD^是直接与母线相连的 16 位连续执行型触点逻辑‘异或’运算指令。如果 S1、S2 两个值不相同，异或结果为 1，该触点导通。如果 S1、S2 两个值相同，异或结果为 0，该触点不导通。

操作数：

S1：K, H, KnX, KnY, KnM,KnS, T, C, D,LV,DT,@

S2：K, H, KnX, KnY, KnM,KnS, T, C, D,LV,DT,@

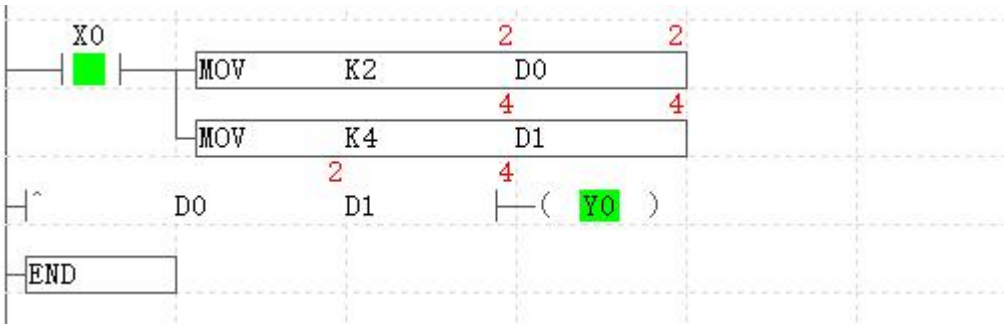
指令格式：[LD^　S1　S2]

编程示例：

下例中，D0 和 D1 的值若不相同，它们的二进制数也不相同，异或结果不为 0，LD^触点导通。

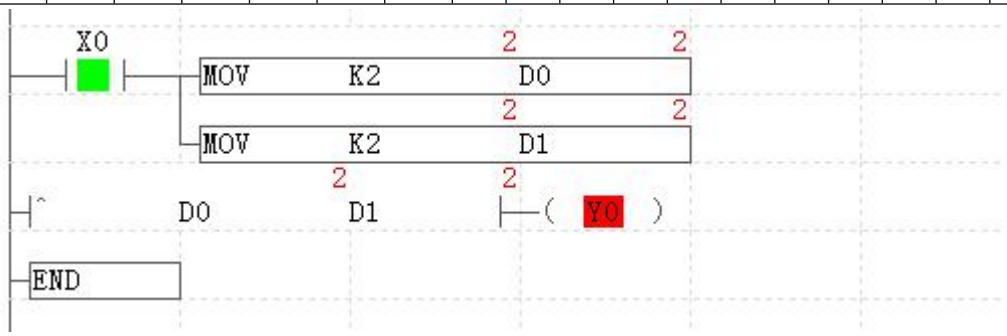
D0 的值为 2，D1 的值为 4，它们的值不相同，异或运算后结果不为 0，该触点导通，Y0 有输出。

软元件	b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0	值
D0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	2
D1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	4
异或运算	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	6



如果它们的值相同，异或运算后结果为 0，该触点不导通，Y0 无输出。

软元件	b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0	值
D0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	2
D1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	2
异或运算	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0



LDD^

指令说明：

LDD^是直接与母线相连的 32 位连续执行型触点逻辑‘异或’运算指令。如果 S1、S2 两个值不相同，异或结果为 1，该触点导通。如果 S1、S2 两个值相同，异或结果为 0，该触点不导通。

用法与 16 位指令 LD^相同，当运算的数据超过 16 位时，就要使用 32 位的与运算指令。

操作数：

S1：K, H, KnX, KnY, KnM,KnS, T, C, D,LV,DT,@

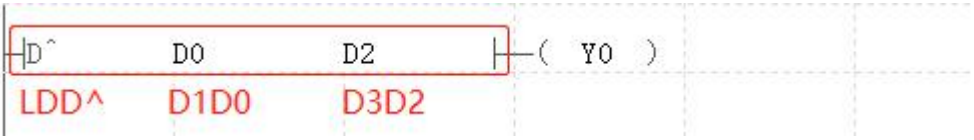
S2：K, H, KnX, KnY, KnM,KnS, T, C, D,LV,DT,@

指令格式：[LDD^ S1 S2]

编程示例：

若 D1D0 和 D3D2 寄存器的值不相同，它们的二进制数也不相同，进行异或运算后，LDD^触点导通，Y0 有输出。

仅当 D1D0 和 D3D2 的值相同时，进行异或运算结果为 0，LDD^触点不导通。



AND#

指令说明：

触点逻辑运算指令 AND&、AND|、AND^、ANDD&、ANDD|、ANDD^，S1 与 S2 的内容作位运算的指令，比较结果不为 0 时则该指令导通，比较结果为 0 时则该指令不导通。

16bit 指令	32bit 指令	导通条件	不导通条件
AND&	ANDD&	S1 & S2 ≠ 0	S1 & S2 = 0
AND|	ANDD|	S1 | S2 ≠ 0	S1 | S2 = 0
AND^	ANDD^	S1 ^ S2 ≠ 0	S1 ^ S2 = 0

操作数：

S1：K, H, KnX, KnY, KnM, T, C, D,LV,DT,@

S2：K, H, KnX, KnY, KnM, T, C, D,LV,DT,@

逻辑运算规则：

运算	& 与				| 或				^ 异或			
操作数	0	0	1	1	0	0	1	1	0	0	1	1
	0	1	0	1	0	1	0	1	0	1	0	1
结果	0	0	0	1	0	1	1	1	0	1	1	0

AND&

指令说明：

AND&是与触点串联的 16 位连续执行型触点逻辑‘与’运算指令。是将 S1 和 S2 的二进制形式的每一个位上的二进制数字进行按位与运算之后的结果，如果结果不等于零时导通。

操作数：

S1：K, H, KnX, KnY, KnM,KnS, T, C, D,LV,DT,@

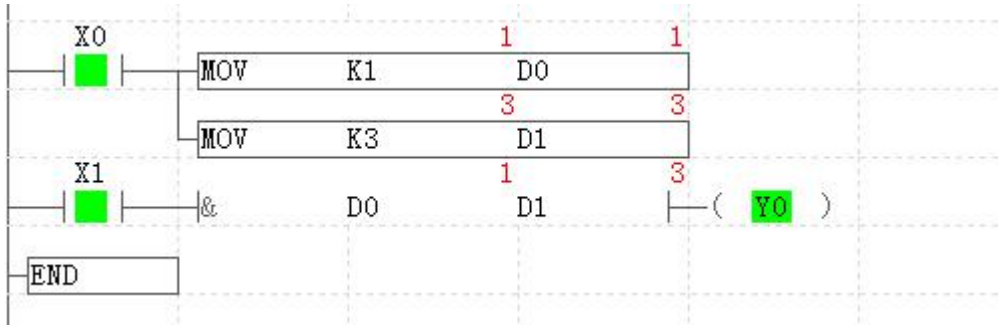
S2：K, H, KnX, KnY, KnM,KnS, T, C, D,LV,DT,@

指令格式：[AND& S1 S2]

编程示例：

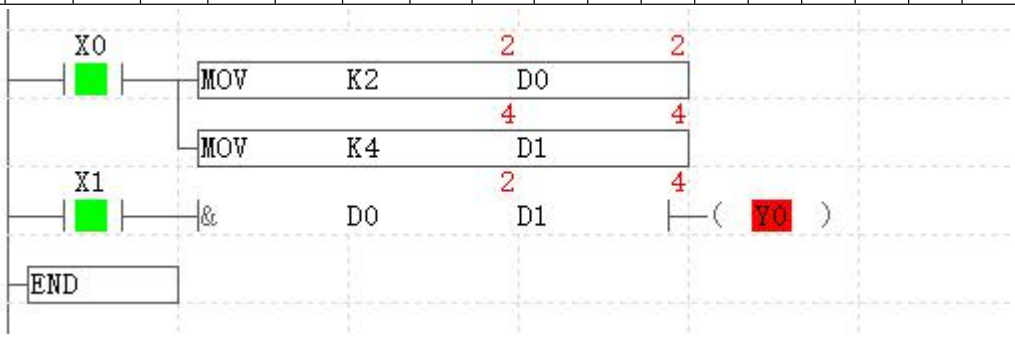
假设 D0 为 1，D1 为 3，那么 D0 的二进制数中的 b0 位为 1，其余位为 0，D1 的二进制数中的 b0 位为 1，b1 位为 1，其余位为 0。因为 D0 与 D1 的 b0 位都为 1，所以与运算后，该触点导通，Y0 有输出。

软元件	b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0	值
D0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1
D1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	3
与运算	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1



假设 D0 为 2，D1 为 4，那么 D0 的二进制数中的 b1 位为 1，其余位为 0，D1 的二进制数中的 b2 位为 1，其余位为 0。因为 D0 与 D1 无相同的位都为 1，所以与运算后，该触点断开，Y0 无输出。

软元件	b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0	值
D0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	2
D1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	4
与运算	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0



ANDD&

指令说明：

ANDD&是与触点串联的 32 位连续执行型触点逻辑‘与’运算指令。是将 S1 和 S2 的二进制形式的每一个位上的二进制数字进行按位与运算之后的结果，如果结果不等于零时导通。

用法与 16 位指令 AND&相同，当运算的数据超过 16 位时，就要使用 32 位的与运算指令。

操作数：

S1：K, H, KnX, KnY, KnM,KnS, T, C, D,LV,DT,@

S2：K, H, KnX, KnY, KnM,KnS, T, C, D,LV,DT,@

指令格式：[ANDD&　S1　S2]

编程示例：

若 D1D0 和 D3D2 表示的 32 位二进制数，若有任意一个相同的位值为 1，进行与运算后，结果不为 0，ANDD&触点导通，输出 Y0。

当 D1D0 和 D3D2 没有相同位的值为 1 时，进行与运算结果为 0，ANDD&触点不导通。

AND|

指令说明：

AND|是与触点串联的 16 位连续执行型触点逻辑‘或’运算指令。是将 S1 和 S2 的二进制形式的每一个位上的二进制数字进行按位或运算之后的结果，如果其十进制数结果不为 0 时导通。

操作数：

S1：K, H, KnX, KnY, KnM,KnS, T, C, D,LV,DT,@

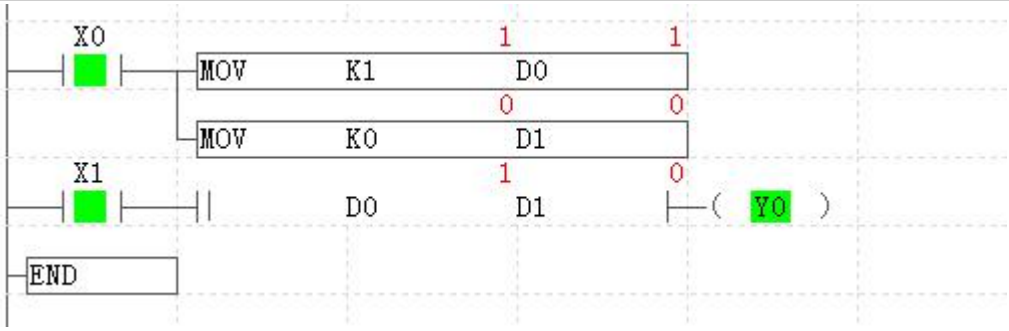
S2：K, H, KnX, KnY, KnM,KnS, T, C, D,LV,DT,@

指令格式：[AND| S1 S2]

编程示例：

下例中，D0 或者是 D1 任意一个寄存器的值不为 0，它们的二进制数中至少有一个位不为 0，进行或运算后，该触点导通，Y0 有输出。

软元件	b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0	值
D0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1
D1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
或运算	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1



ANDD|

指令说明：

ANDD|是与触点串联的 32 位连续执行型触点逻辑‘或’运算指令。是将 S1 和 S2 的二进制形式的每一个位上的二进制数字进行按位或运算之后的结果，如果其十进制数结果不为 0 时导通。

用法与 16 位指令 AND|相同，当运算的数据超过 16 位时，就要使用 32 位的与运算指令。

操作数：

S1：K, H, KnX, KnY, KnM,KnS, T, C, D,LV,DT,@

S2：K, H, KnX, KnY, KnM,KnS, T, C, D,LV,DT,@

指令格式：[ANDD| S1 S2]

编程示例：

若 D1D0 和 D3D2 任意一个寄存器的值不为 0，它们的二进制数中至少有一个位不为 0，进行或运算后，ANDD|触点导通，Y0 有输出。

仅当 D1D0 和 D3D2 的值均为 0 时，进行或运算结果也为 0，ANDD|触点不导通。

AND^

指令说明

AND^是与触点串联的 16 位连续执行型触点逻辑‘异或’运算指令。如果 S1、S2 两个值不相同，异或结果为 1，该触点导通。如果 S1、S2 两个值相同，异或结果为 0，该触点不导通。

操作数：

S1：K, H, KnX, KnY, KnM,KnS, T, C, D,LV,DT,@

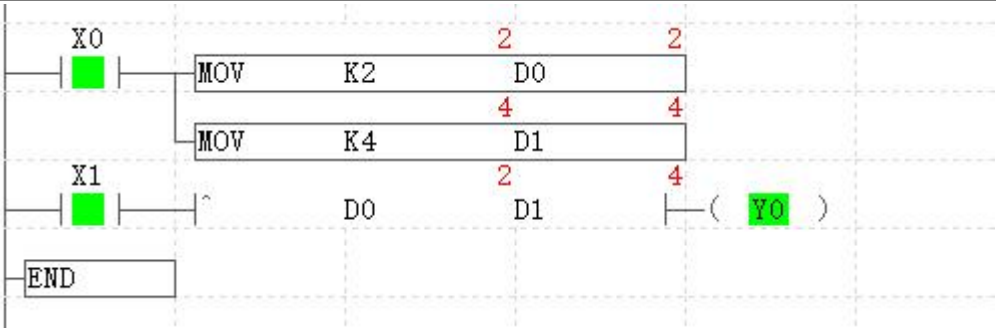
S2：K, H, KnX, KnY, KnM,KnS, T, C, D,LV,DT,@

指令格式：[AND^ S1 S2]

编程示例：

下例中，D0 的值为 2，D1 的值为 4，它们的值不相同，异或运算后结果不为 0，该触点导通，Y0 有输出。

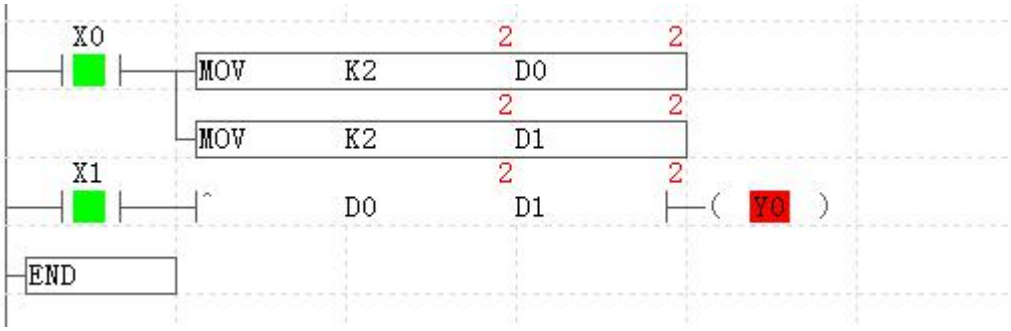
软元件	b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0	值
D0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	2
D1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	4
异或运算	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	6



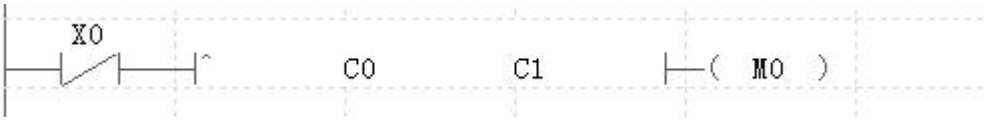
如果它们的值相同，异或运算后结果为 0，该触点不导通，Y0 无输出。

软元件	b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0	值
D0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	2

D1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	2
异或运算	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0



当 C0 或 C1 的值不同时，触点导通，输出 M0；当 C0 或 C1 的值相同时，触点断开，M0 断开。



ANDD^

指令说明：

ANDD^是直接与母线串联的 32 位连续执行型触点逻辑‘异或’运算指令。如果 S1、S2 两个值不相同，异或结果为 1，该触点导通。如果 S1、S2 两个值相同，异或结果为 0，该触点不导通。当运算的数据超过 16 位时，就要使用 32 位的与运算指令。

操作数：

S1：K, H, KnX, KnY, KnM,KnS, T, C, D,LV,DT,@

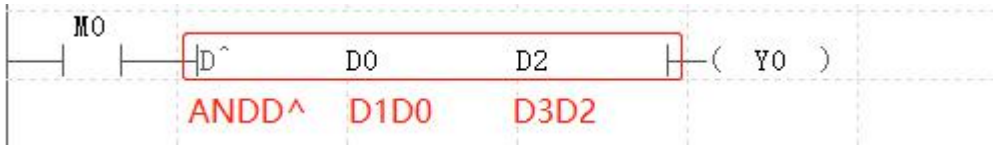
S2：K, H, KnX, KnY, KnM,KnS, T, C, D,LV,DT,@

指令格式：[ANDD^ S1 S2]

编程示例：

若 D1D0 和 D3D2 寄存器的值不相同，它们的二进制数也不相同，进行异或运算后，ANDD^触点导通，Y0 有输出。

仅当 D1D0 和 D3D2 的值相同时，进行异或运算结果为 0，ANDD^触点不导通。



OR#

指令说明：

触点逻辑运算指令 OR&、OR|、OR^、ORD&、ORD|、ORD^，S1 与 S2 的内容作位运算的指令，比较结果不为 0 时则该指令导通，比较结果为 0 时则该指令不导通。

16bit 指令	32bit 指令	导通条件	不导通条件
OR&	ORD&	S1 & S2 ≠ 0	S1 & S2 = 0
OR|	ORD|	S1 | S2 ≠ 0	S1 | S2 = 0
OR^	ORD^	S1 ^ S2 ≠ 0	S1 ^ S2 = 0

操作数：

S1：K, H, KnX, KnY, KnM, T, C, D,LV,DT,@

S2：K, H, KnX, KnY, KnM, T, C, D,LV,DT,@

逻辑运算规则：

运算	& 与				| 或				^ 异或			
操作数	0	0	1	1	0	0	1	1	0	0	1	1
	0	1	0	1	0	1	0	1	0	1	0	1
结果	0	0	0	1	0	1	1	1	0	1	1	0

OR&

指令说明：

OR&是与触点并联的 16 位连续执行型触点逻辑‘与’运算指令。是将 S1 和 S2 的二进制形式的每一个位上的二进制数字进行按位与运算之后的结果，如果结果不等于零时导通。

操作数：

S1：K, H, KnX, KnY, KnM,KnS, T, C, D,LV,DT,@

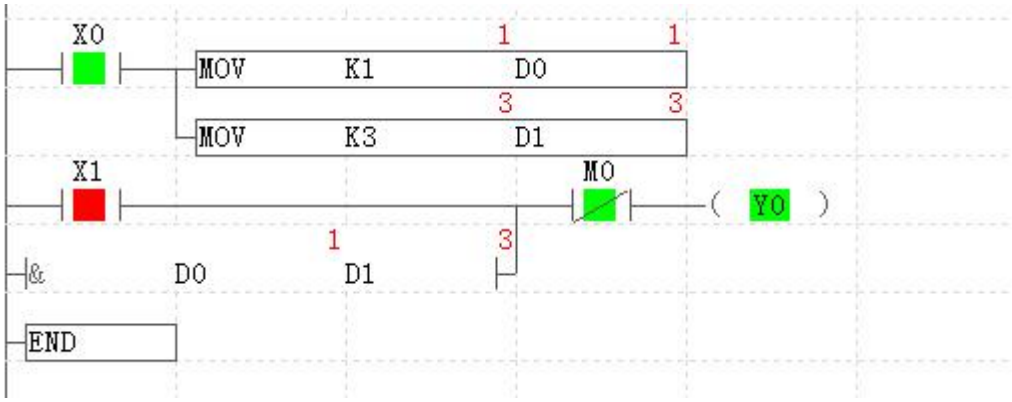
S2：K, H, KnX, KnY, KnM,KnS, T, C, D,LV,DT,@

指令格式：[OR&　S1　S2]

编程示例：

假设 D0 为 1，D1 为 3，那么 D0 的二进制数中的 b0 位为 1，其余位为 0，D1 的二进制数中的 b0 位为 1，b1 位为 1，其余位为 0。因为 D0 与 D1 的 b0 位都为 1，所以与运算后，该触点导通，Y0 有输出。

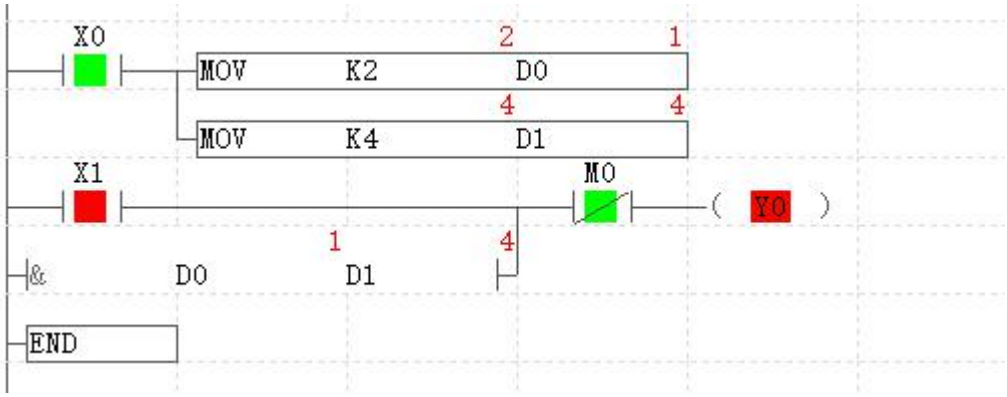
软元件	b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0	值
D0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1
D1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	3
与运算	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1



假设 D0 为 2，D1 为 4，那么 D0 的二进制数中的 b1 位为 1，其余位为 0，D1 的二进

制数中的 b2 位为 1，其余位为 0。 因为 D0 与 D1 无相同的位都为 1，所以与运算后，该触点断开，Y0 无输出。

软元件	b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0	值
D0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	2
D1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	4
与运算	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0



ORD&

指令说明：

ORD&是与触点并联的 32 位连续执行型触点逻辑‘与’运算指令。是将 S1 和 S2 的二进制形式的每一个位上的二进制数字进行按位与运算之后的结果，如果结果不等于零时导通。

用法与 16 位指令 OR&相同，当运算的数据超过 16 位时，就要使用 32 位的与运算指令。

操作数：

S1：K, H, KnX, KnY, KnM,KnS, T, C, D,LV,DT,@

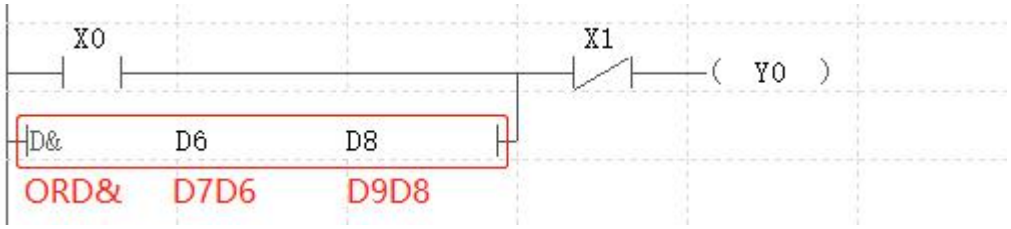
S2：K, H, KnX, KnY, KnM,KnS, T, C, D,LV,DT,@

指令格式：[ORD&　S1　S2]

编程示例：

若 D1D0 和 D3D2 表示的 32 位二进制数，若有任意一个相同的位值为 1，进行与运算后，结果不为 0，ORD&触点导通，输出 Y0。

当 D1D0 和 D3D2 没有相同位的值为 1 时，进行与运算结果为 0，ORD&触点不导通。



OR|

指令说明：

OR|是与触点并联的 16 位连续执行型触点逻辑‘或’运算指令。是将 S1 和 S2 的二进制形式的每一个位上的二进制数字进行按位或运算之后的结果，如果其十进制数结果不为 0 时导通。

操作数：

S1：K, H, KnX, KnY, KnM,KnS, T, C, D,LV,DT,@

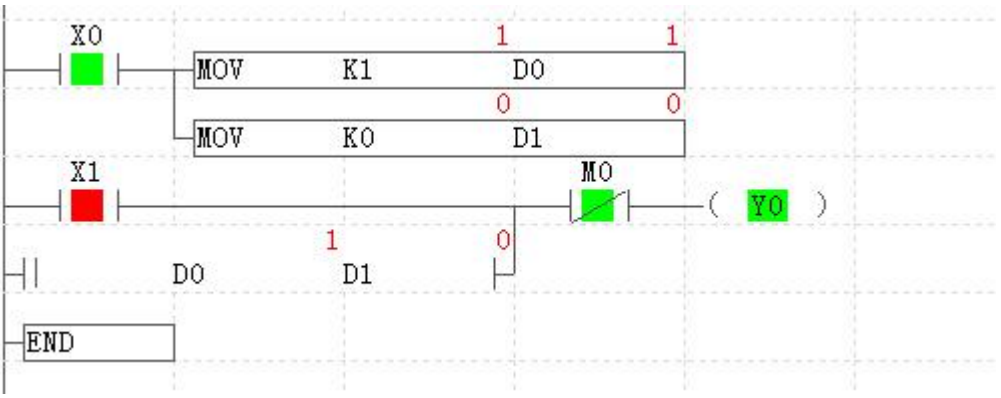
S2：K, H, KnX, KnY, KnM,KnS, T, C, D,LV,DT,@

指令格式：[OR|　S1　S2]

编程示例：

下例中，D0 或者是 D1 任意一个寄存器的值不为 0，它们的二进制数中至少有一个位不为 0，进行或运算后，该触点导通，Y0 有输出。

软元件	b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0	值
D0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1
D1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
或运算	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1



ORD|

指令说明：

ORD|是与触点并联的 32 位连续执行型触点逻辑‘或’运算指令。是将 S1 和 S2 的二进制形式的每一个位上的二进制数字进行按位或运算之后的结果，如果其十进制数结果不为 0 时导通。当运算的数据超过 16 位时，就要使用 32 位的与运算指令。

操作数：

S1：K, H, KnX, KnY, KnM,KnS, T, C, D,LV,DT,@

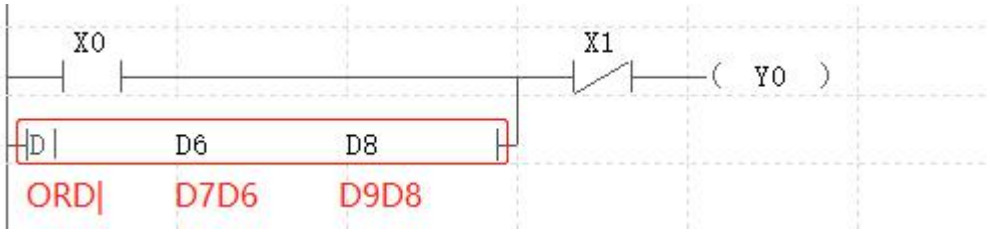
S2：K, H, KnX, KnY, KnM,KnS, T, C, D,LV,DT,@

指令格式：[ORD|　S1　S2]

编程示例：

若 D1D0 和 D3D2 任意一个寄存器的值不为 0，它们的二进制数中至少有一个位不为 0，进行或运算后，ORD|触点导通，Y0 有输出。

仅当 D1D0 和 D3D2 的值均为 0 时，进行或运算结果也为 0，ORD|触点不导通。



OR^

指令说明

OR^是与触点并联的 16 位连续执行型触点逻辑‘异或’运算指令。如果 S1、S2 两个值不相同，异或结果为 1，该触点导通。如果 S1、S2 两个值相同，异或结果为 0，该触点不导通。

操作数：

S1：K, H, KnX, KnY, KnM,KnS, T, C, D,LV,DT,@

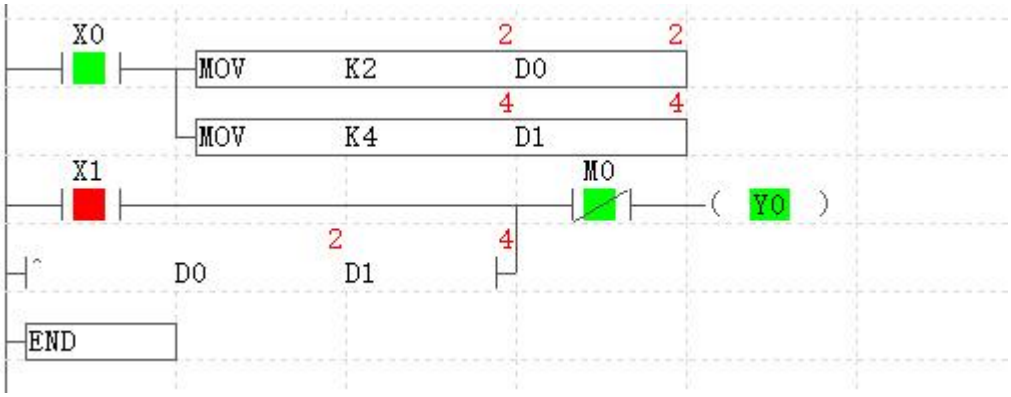
S2：K, H, KnX, KnY, KnM,KnS, T, C, D,LV,DT,@

指令格式：[OR^ S1 S2]

编程示例：

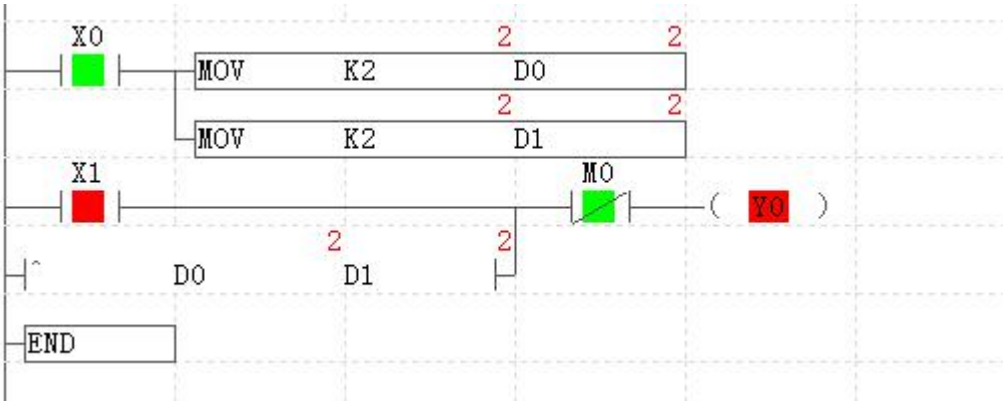
下例中，D0 的值为 2，D1 的值为 4，它们的值不相同，异或运算后结果不为 0，该触点导通，Y0 有输出。

软元件	b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0	值
D0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	2
D1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	4
异或运算	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	6



如果它们的值相同，异或运算后结果为 0，该触点不导通，Y0 无输出。

软元件	b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0	值
D0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	2
D1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	2
异或运算	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0



ORD^

指令说明：

ORD^是直接与母线并联的 32 位连续执行型触点逻辑‘异或’运算指令。如果 S1、S2 两个值不相同，异或结果为 1，该触点导通。如果 S1、S2 两个值相同，异或结果为 0，该触点不导通。当运算的数据超过 16 位时，就要使用 32 位的与运算指令。

操作数：

S1：K, H, KnX, KnY, KnM,KnS, T, C, D,LV,DT,@

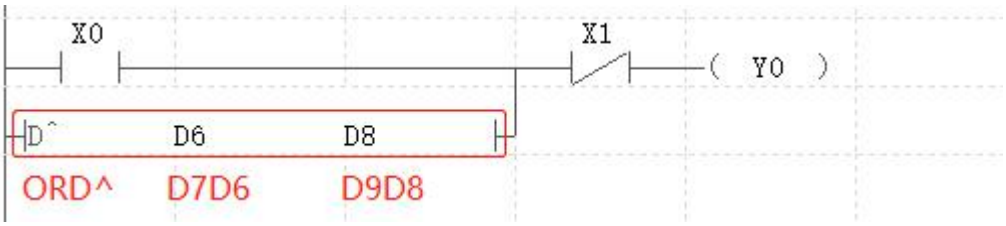
S2：K, H, KnX, KnY, KnM,KnS, T, C, D,LV,DT,@

指令格式：[ORD^ S1 S2]

编程示例：

若 D1D0 和 D3D2 寄存器的值不相同，它们的二进制数也不相同，进行异或运算后，ORD^触点导通，Y0 有输出。

仅当 D1D0 和 D3D2 的值相同时，进行异或运算结果为 0，ORD^触点不导通。



FLD=

指令说明：

FLD=是连接在母线上的 32 位浮点数触点比较指令，其作用是将 S1、S2 的内容按指定的条件进行比较，根据其结果来控制触点的导通或是不导通。

其实浮点数触点比较我们可以把它看成一个触点，当 S1=S2，触点是闭合状态。若 S1<S2 或 S1>S2，触点是断开状态。

比较指令	导通条件	不导通条件	条件成立	条件不成立

[F=　S1　S2]	S1 = S2	S1 ≠ S2	触点闭合	触点断开

操作数：

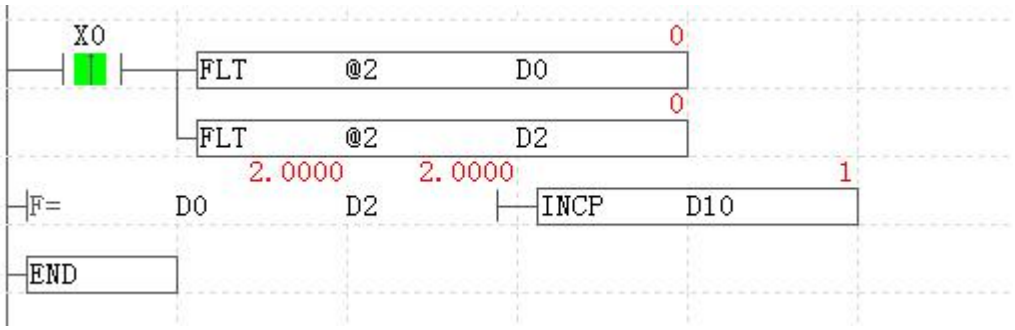
S1：T,C,D,LV,DT,@

S2：T,C,D,LV,DT,@

指令格式：[FLD=　S1　S2]

编程示例：

X0 从 OFF 变为 ON，FLT 指令执行，将整数 2 转换成二进制浮点数，由于是 32 位浮点数，占用连续两个数据寄存器 D 空间，数据存储到 D1D0、D3D2 中，当 D1D0 和 D3D2 的数值相等时，触点导通，D10 自增 1。



FLD>

指令说明：

FLD>是连接在母线上的 32 位浮点数触点比较指令，其作用是将 S1、S2 的内容按指定的条件进行比较，根据其结果来控制触点的导通或是不导通。

其实浮点数触点比较我们可以把它看成一个触点，当 S1>S2，触点是闭合状态。若 S1<S2 或 S1=S2，触点是断开状态。

比较指令	导通条件	不导通条件	条件成立	条件不成立
[F>　S1　S2]	S1 > S2	S1 ≤ S2	触点闭合	触点断开

操作数：

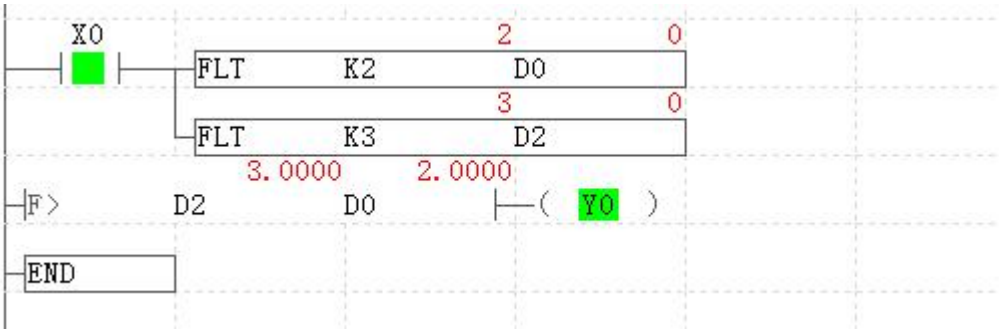
S1：T,C,D,LV,DT,@

S2：T,C,D,LV,DT,@

指令格式：[FLD>　S1　S2]

编程示例：

X0 闭合，FLT 指令连续执行，将整数 2 转换成二进制浮点数，由于是 32 位浮点数，占用连续两个数据寄存器 D 空间，数据存储到 D1D0，整数 3 转换成二进制浮点数存储到 D3D2 中，此时 D3D2>D1D0，触点导通，输出 Y0。



FLD<

指令说明：

FLD<是连接在母线上的 32 位浮点数触点比较指令，其作用是将 S1、S2 的内容按指定的条件进行比较，根据其结果来控制触点的导通或是不导通。

其实浮点数触点比较我们可以把它看成一个触点，当 S1<S2，触点是闭合状态。若 S1>S2 或 S1=S2，触点是断开状态。

比较指令	导通条件	不导通条件	条件成立	条件不成立
[F< S1 S2]	S1 < S2	S1 ≥ S2	触点闭合	触点断开

操作数：

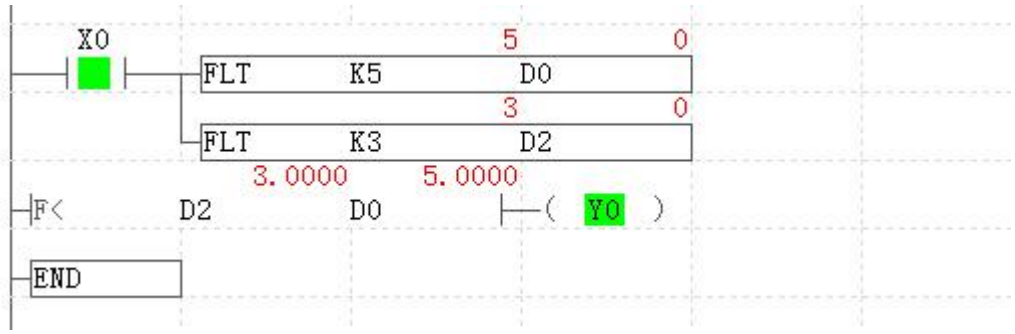
S1：T,C,D,LV,DT,@

S2：T,C,D,LV,DT,@

指令格式：[FLD< S1 S2]

编程示例：

X0 闭合，FLT 指令执行，将整数 5 转换成二进制浮点数，由于是 32 位浮点数，占用连续两个数据寄存器 D 空间，数据存储到 D1D0，整数 3 转换成二进制浮点数存储到 D3D2 中，此时 D3D2<D1D0，触点导通，输出 Y0。



FLD<>

指令说明：

FLD<>是连接在母线上的 32 位浮点数触点比较指令，其作用是将 S1、S2 的内容按指定的条件进行比较，根据其结果来控制触点的导通或是不导通。

其实浮点数触点比较我们可以把它看成一个触点，当 S1≠S2，触点是闭合状态。若 S1=S2，触点是断开状态。

比较指令	导通条件	不导通条件	条件成立	条件不成立
[F<>　S1　S2]	S1 ≠ S2	S1 = S2	触点闭合	触点断开

操作数：

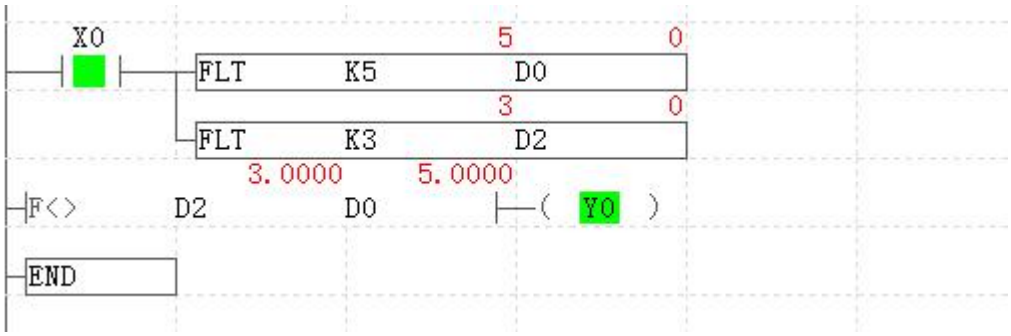
S1：T,C,D,LV,DT,@

S2：T,C,D,LV,DT,@

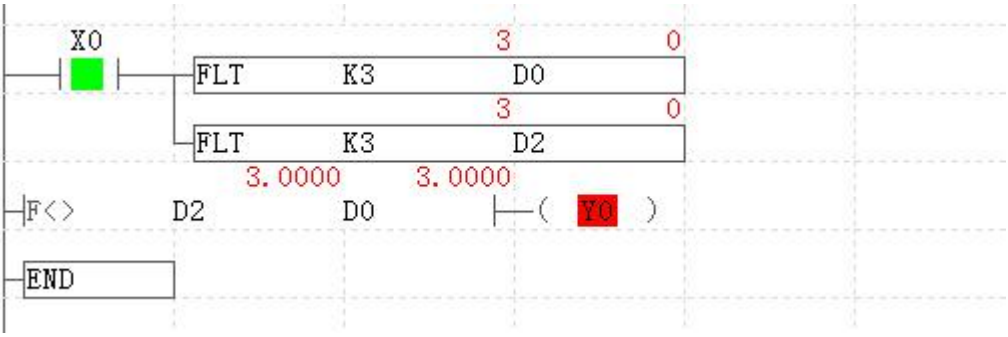
指令格式：[FLD<>　S1　S2]

编程示例：

X0 闭合，FLT 指令执行，将整数 5 转换成二进制浮点数，由于是 32 位浮点数，占用连续两个数据寄存器 D 空间，数据存储到 D1D0，整数 3 转换成二进制浮点数存储到 D3D2 中，此时 D3D2 不等于 D1D0，触点导通，输出 Y0。



当 D3D2=D1D0 时，触点断开，Y0 断开。



FLD<=

指令说明：

FLD<=是连接在母线上的 32 位浮点数触点比较指令，其作用是将 S1、S2 的内容按指定的条件进行比较，根据其结果来控制触点的导通或是不导通。

其实浮点数触点比较我们可以把它看成一个触点，当 S1≤S2，触点是闭合状态。若 S1>S2，触点是断开状态。

比较指令	导通条件	不导通条件	条件成立	条件不成立
[F< =　S1　S2]	S1 ≤ S2	S1 > S2	触点闭合	触点断开

操作数：

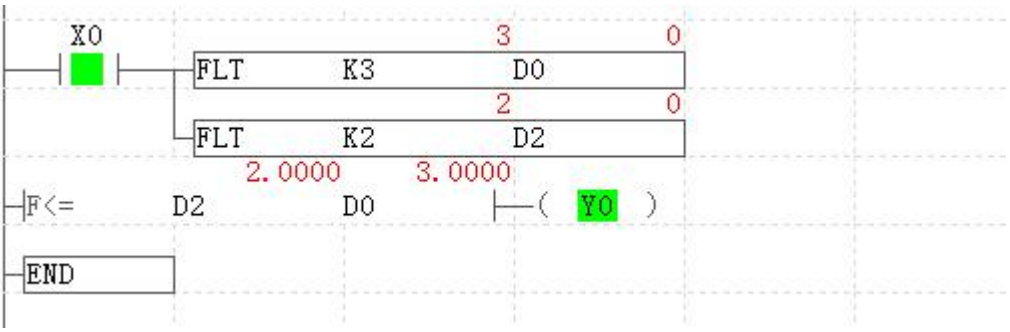
S1：T,C,D,LV,DT,@

S2：T,C,D,LV,DT,@

指令格式：[FLD<=　S1　S2]

编程示例：

X0 闭合，FLT 指令执行，将整数 3 转换成二进制浮点数，由于是 32 位浮点数，占用连续两个数据寄存器 D 空间，数据存储到 D1D0，整数 2 转换成二进制浮点数存储到 D3D2 中，此时 D3D2<D1D0，触点导通，输出 Y0。



FLD>=

指令说明：

FLD>=是连接在母线上的 32 位浮点数触点比较指令，其作用是将 S1、S2 的内容按指定的条件进行比较，根据其结果来控制触点的导通或是不导通。

其实浮点数触点比较我们可以把它看成一个触点，当 S1≥ S2，触点是闭合状态。若 S1<S2，触点是断开状态。

比较指令	导通条件	不导通条件	条件成立	条件不成立
[F >=　S1　S2]	S1 ≥ S2	S1 < S2	触点闭合	触点断开

操作数：

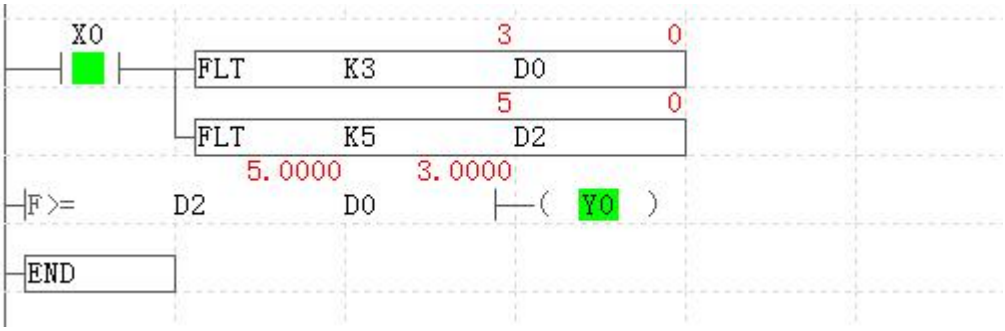
S1：T,C,D,LV,DT,@

S2：T,C,D,LV,DT,@

指令格式：[FLD>=　S1　S2]

编程示例：

X0 闭合，FLT 指令执行，将整数 3 转换成二进制浮点数，由于是 32 位浮点数，占用连续两个数据寄存器 D 空间，数据存储到 D1D0，整数 5 转换成二进制浮点数存储到 D3D2 中，此时 D3D2>D1D0，触点导通，输出 Y0。



FAND=

指令说明：

FAND=是与触点串联的 32 位浮点数触点比较指令，其作用是将 S1、S2 的内容按指定的条件进行比较，根据其结果来控制触点的导通或是不导通。

其实浮点数触点比较我们可以把它看成一个触点，当 S1=S2，触点是闭合状态。若 S1≠S2，触点是断开状态。

比较指令	导通条件	不导通条件	条件成立	条件不成立
[F=　S1　S2]	S1 = S2	S1 ≠ S2	触点闭合	触点断开

操作数：

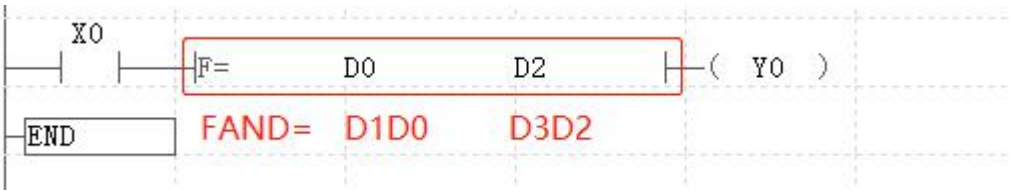
S1：T,C,D,LV,DT,@

S2：T,C,D,LV,DT,@

指令格式：[FAND=　S1　S2]

编程示例：

X0 闭合，当 D1D0=D3D2 时，触点闭合，输出 Y0。当 D1D0 不等于 D3D2 时，触点断开，Y0 无输出。



FAND>

指令说明：

FAND>是与触点串联的 32 位浮点数触点比较指令，其作用是将 S1、S2 的内容按指定的条件进行比较，根据其结果来控制触点的导通或是不导通。

其实浮点数触点比较我们可以把它看成一个触点，当 S1>S2，触点是闭合状态。若 S1<S2 或 S1=S2，触点是断开状态。

比较指令	导通条件	不导通条件	条件成立	条件不成立
[F>　S1　S2]	S1 > S2	S1 ≤ S2	触点闭合	触点断开

操作数：

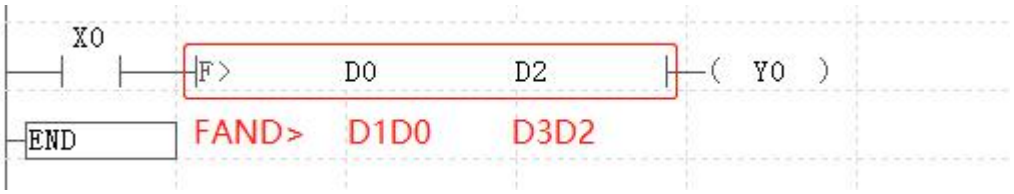
S1：T,C,D,LV,DT,@

S2：T,C,D,LV,DT,@

指令格式：[FAND>　S1　S2]

编程示例：

X0 闭合，当 D1D0> D3D2 时，触点闭合，输出 Y0。当 D1D0<= D3D2 时，触点断开，Y0 无输出。



FAND<

指令说明：

FAND<是与触点串联的 32 位浮点数触点比较指令，其作用是将 S1、S2 的内容按指定的条件进行比较，根据其结果来控制触点的导通或是不导通。

其实浮点数触点比较我们可以把它看成一个触点，当 S1<S2，触点是闭合状态。若 S1>S2 或 S1=S2，触点是断开状态。

比较指令	导通条件	不导通条件	条件成立	条件不成立
[F< S1 S2]	S1 < S2	S1 ≥ S2	触点闭合	触点断开

操作数：

S1：T,C,D,LV,DT,@

S2：T,C,D,LV,DT,@

指令格式：[FAND< S1 S2]

编程示例：

X0 闭合，当 D1D0< D3D2 时，触点闭合，输出 Y0。当 D1D0>= D3D2 时，触点断开，Y0 无输出。

FAND<>

指令说明：

FAND<>是与触点串联的 32 位浮点数触点比较指令，其作用是将 S1、S2 的内容按指定的条件进行比较，根据其结果来控制触点的导通或是不导通。

其实浮点数触点比较我们可以把它看成一个触点，当 S1≠S2，触点是闭合状态。若 S1=S2，触点是断开状态。

比较指令	导通条件	不导通条件	条件成立	条件不成立
[F<> S1 S2]	S1 ≠ S2	S1 = S2	触点闭合	触点断开

操作数：

S1：T,C,D,LV,DT,@

S2：T,C,D,LV,DT,@

指令格式：[FAND<>　S1　S2]

编程示例：

X0 闭合，当 D1D0 不等于 D3D2 时，触点闭合，输出 Y0。当 D1D0= D3D2 时，触点断开，Y0 无输出。

FAND<=

指令说明：

FAND<=是与触点串联的 32 位浮点数触点比较指令，其作用是将 S1、S2 的内容按指定的条件进行比较，根据其结果来控制触点的导通或是不导通。

其实浮点数触点比较我们可以把它看成一个触点，当 S1≤S2，触点是闭合状态。若 S1>S2，触点是断开状态。

比较指令	导通条件	不导通条件	条件成立	条件不成立
[F< =　S1　S2]	S1 ≤ S2	S1 > S2	触点闭合	触点断开

操作数：

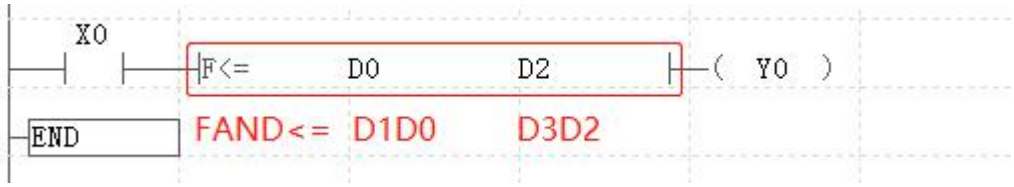
S1：T,C,D,LV,DT,@

S2：T,C,D,LV,DT,@

指令格式：[FAND<=　S1　S2]

编程示例：

X0 闭合，当 D1D0<= D3D2 时，触点闭合，输出 Y0。当 D1D0> D3D2 时，触点断开，Y0 无输出。



FAND>=

指令说明：

FAND>=是与触点串联的 32 位浮点数触点比较指令，其作用是将 S1、S2 的内容按指定的条件进行比较，根据其结果来控制触点的导通或是不导通。

其实浮点数触点比较我们可以把它看成一个触点，当 S1≥ S2，触点是闭合状态。若 S1<S2，触点是断开状态。

比较指令	导通条件	不导通条件	条件成立	条件不成立

[F >= S1 S2]	S1 ≥ S2	S1 < S2	触点闭合	触点断开

操作数：

S1：T,C,D,LV,DT,@

S2：T,C,D,LV,DT,@

指令格式：[FAND>= S1 S2]

编程示例：

X0 闭合，当 D1D0>= D3D2 时，触点闭合，输出 Y0。当 D1D0< D3D2 时，触点断开，Y0 无输出。

FOR=

指令说明：

FOR=是与触点并联的 32 位浮点数触点比较指令，其作用是将 S1、S2 的内容按指定的条件进行比较，根据其结果来控制触点的导通或是不导通。

其实浮点数触点比较我们可以把它看成一个触点，当 S1=S2，触点是闭合状态。若 S1≠S2，触点是断开状态。

比较指令	导通条件	不导通条件	条件成立	条件不成立
[F= S1 S2]	S1 = S2	S1 ≠ S2	触点闭合	触点断开

操作数：

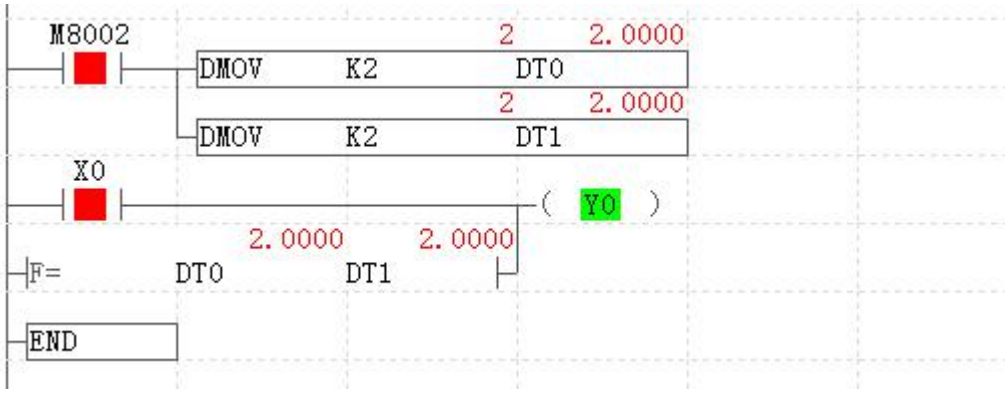
S1：T,C,D,LV,DT,@

S2：T,C,D,LV,DT,@

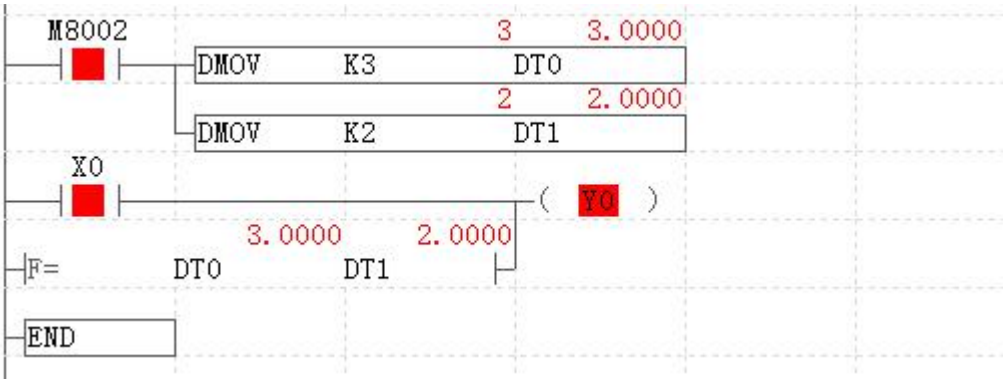
指令格式：[FOR= S1 S2]

编程示例：

上电后，M8002 接通一个扫描周期，DMOV 数据传送指令执行，比较指令条件满足触点导通，输出 Y0。



不满足条件时触点断开，Y0 不输出。



FOR>

指令说明：

FOR>是与触点并联的 32 位浮点数触点比较指令，其作用是将 S1、S2 的内容按指定的条件进行比较，根据其结果来控制触点的导通或是不导通。

其实浮点数触点比较我们可以把它看成一个触点，当 S1>S2，触点是闭合状态。若 S1<S2 或 S1=S2，触点是断开状态。

比较指令	导通条件	不导通条件	条件成立	条件不成立
[F>　S1　S2]	S1 > S2	S1 ≤ S2	触点闭合	触点断开

操作数：

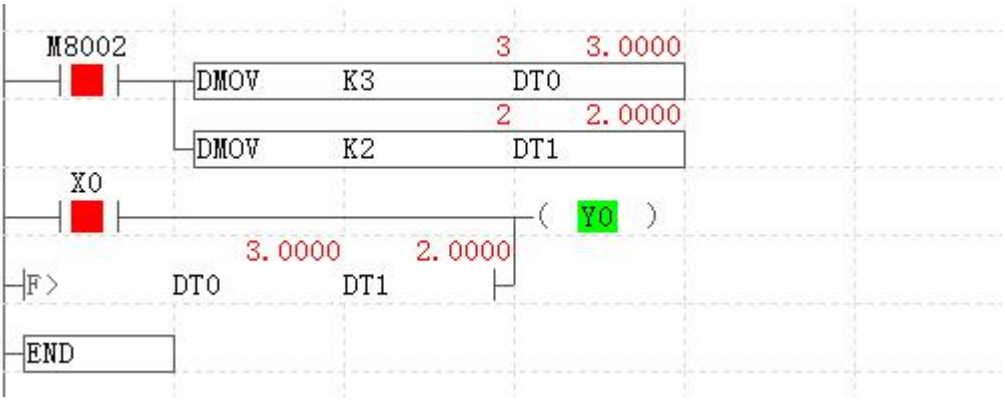
S1：T,C,D,LV,DT,@

S2：T,C,D,LV,DT,@

指令格式：[FOR>　S1　S2]

编程示例：

上电后，M8002 接通一个扫描周期，DMOV 数据传送指令执行，比较指令条件满足触点导通，输出 Y0。



FOR<

指令说明：

FOR<是与触点并联的 32 位浮点数触点比较指令，其作用是将 S1、S2 的内容按指定的条件进行比较，根据其结果来控制触点的导通或是不导通。

其实浮点数触点比较我们可以把它看成一个触点，当 S1<S2，触点是闭合状态。若 S1>S2 或 S1=S2，触点是断开状态。

比较指令	导通条件	不导通条件	条件成立	条件不成立
[F< S1 S2]	S1 < S2	S1 ≥ S2	触点闭合	触点断开

操作数：

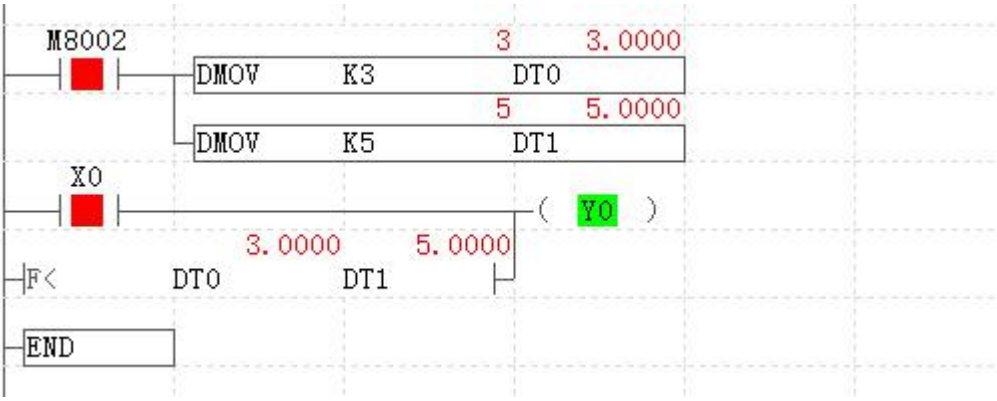
S1：T,C,D,LV,DT,@

S2：T,C,D,LV,DT,@

指令格式：[FOR< S1 S2]

编程示例：

上电后，M8002 接通一个扫描周期，DMOV 数据传送指令执行，比较指令条件满足触点导通，输出 Y0。



FOR<>

指令说明：

FOR<>是与触点并联的 32 位浮点数触点比较指令，其作用是将 S1、S2 的内容按指定的条件进行比较，根据其结果来控制触点的导通或是不导通。

其实浮点数触点比较我们可以把它看成一个触点，当 S1≠S2，触点是闭合状态。若 S1=S2，触点是断开状态。

比较指令	导通条件	不导通条件	条件成立	条件不成立
[F<> S1 S2]	S1 ≠ S2	S1 = S2	触点闭合	触点断开

操作数：

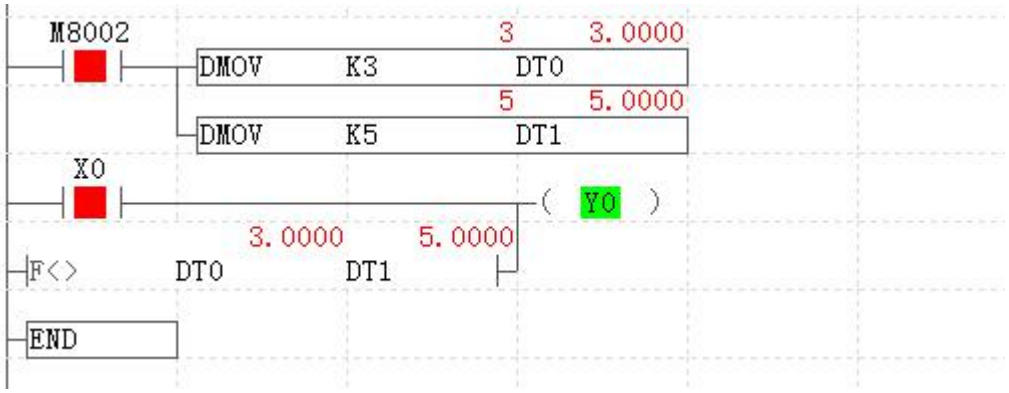
S1：T,C,D,LV,DT,@

S2：T,C,D,LV,DT,@

指令格式：[FOR<> S1 S2]

编程示例：

上电后，M8002 接通一个扫描周期，DMOV 数据传送指令执行，比较指令条件满足触点导通，输出 Y0。



FOR<=

指令说明：

FOR<=是与触点并联的 32 位浮点数触点比较指令，其作用是将 S1、S2 的内容按指定的条件进行比较，根据其结果来控制触点的导通或是不导通。

其实浮点数触点比较我们可以把它看成一个触点，当 S1≤S2，触点是闭合状态。若 S1>S2，触点是断开状态。

比较指令	导通条件	不导通条件	条件成立	条件不成立
[F< = S1 S2]	S1 ≤ S2	S1 > S2	触点闭合	触点断开

操作数：

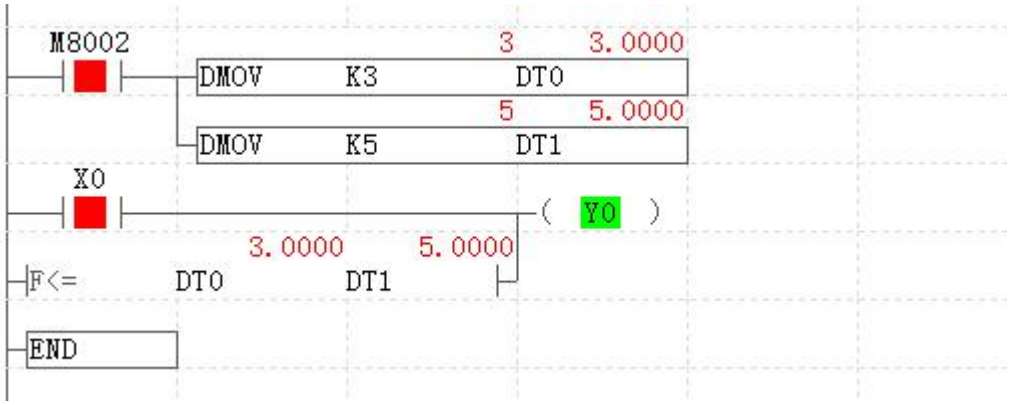
S1：T,C,D,LV,DT,@

S2：T,C,D,LV,DT,@

指令格式：[FOR<= S1 S2]

编程示例：

上电后，M8002 接通一个扫描周期，DMOV 数据传送指令执行，比较指令条件满足触点导通，输出 Y0。



FOR>=

指令说明：

FOR>=是与触点并联的 32 位浮点数触点比较指令，其作用是将 S1、S2 的内容按指定的条件进行比较，根据其结果来控制触点的导通或是不导通。

其实浮点数触点比较我们可以把它看成一个触点，当 S1≥ S2，触点是闭合状态。若 S1<S2，触点是断开状态。

比较指令	导通条件	不导通条件	条件成立	条件不成立
[F >= S1 S2]	S1 ≥ S2	S1 < S2	触点闭合	触点断开

操作数：

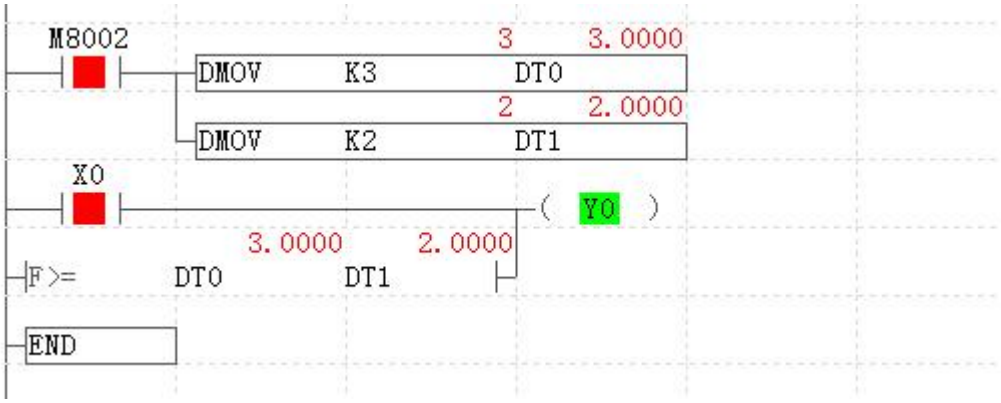
S1：T,C,D,LV,DT,@

S2：T,C,D,LV,DT,@

指令格式：[FOR>= S1 S2]

编程示例：

上电后，M8002 接通一个扫描周期，DMOV 数据传送指令执行，比较指令条件满足触点导通，输出 Y0。



3.3 传送和比较指令

后缀字母“P”为脉冲型指令，前缀字母“D”为 32 位指令。

以下指令均不可与母线直接相连。

分类	指令	位数	脉冲	功能	操作数类型
CMP 比较	CMP	16 位	-	[CMP S1 S2 D] S1>S2，Dn 为 ON S1=S2，Dn+1 为 ON S1<S2，Dn+2 为 ON	S1/S2：KnX,KnY,KnM,T,C,D,K,H,V,Z,LV,DT,@ D：Y,M,S,@
	CMPP		√		
	DCMP	32 位	-		
	DCMPP		√		
ZCP 连续区间比较	ZCP	16 位	-	[ZCP S1 S2 S D] S<S1，Dn 为 ON S1<=S<=S2，Dn+1 为 ON S>S2，Dn+2 为 ON	S/S1/S2：KnX,KnY,KnM,T,C,D,K,H,V,Z,LV,DT,@ D：Y,M,S,@
	ZCPP		√		
	DZCP	32 位	-		
	DZCPP		√		
MOV 传送	MOV	16 位	-	[MOV S D] 将 S 的值传送给 D	S：KnX,KnY,KnM,KnS,T,C,D,K,H,V,Z,LV,DT,@ D：KnY,KnM,KnS,T,C,
	MOVP		√		
	DMOV	32 位	-		

	DMOVP		√		D,V,Z,LV,DT
SMOV 移动	SMOV	16 位	-	[SMOV S m1 m2 D n] S 转换成 4 位 BCD，然后将 S 从 m1 位数开始的低 m2 位数部分合并到 D 的 n 位处，最后将合并后的数据转换成 BIN 并保存到 D 中	S：KnX,KnY,KnM,KnS, T,C,D,V,Z,LV,DT,@ m1/m2/n：K,H,D,DT,C,T D：KnY,KnM,KnS,T,C, D,V,Z,LV,DT
	SMOVP		√		
CML 反转传送	CML	16 位	-	[CML S D] S 的数据（自动转换成二进制数）逐位取反后，传送到 D	S：KnX,KnY,KnM,KnS, T,C,D,K,H,V,Z,LV,DT,@ D：KnY,KnM,KnS, T,C, D,V,Z,LV,DT,@
	CMLP		√		
	DCML	32 位	-		
	DCMLP		√		
BMOV 成批传送	BMOV	16 位	-	[BMOV S D n] S 开始的 n 个数据成批传送到 D 开始的 n 点软元件中	S： KnX,KnY,KnM,KnS, T,C,D,LV,DT D：KnY,KnM,KnS,T,C, D,LV,DT n：D,K,H
	BMOVP		√		
FMOV 多点传送	FMOV	16 位	-	[FMOV S D n] S 的数据多点传送到 D 开始的 n 点软元件中	S：KnX,KnY,KnM,KnS, T,C,D,K,H,V,Z,LV,DT,@ D：KnY,KnM,KnS,T,C, D,LV,DT n：K,H
	FMOVP		√		
	DFMOV	32 位	-		
	DFMOVP		√		
XCH 交换	XCH	16 位	-	[XCH D1 D2] D1 和 D2 的数据交换	D1：KnY,KnM,KnS,T, C,D,V,Z,LV,DT D2：KnY,KnM,KnS,T, C,D,V,Z,LV,DT
	XCHP		√		
	DXCH	32 位	-		
	DXCHP		√		
BCD 转换	BCD	16 位	-	[BCD S D] 将 S 中的 BIN 码（二进制表示十进制）转换成 BCD 码（4 位二进制表示一位十进制）传送到 D 中	S：KnX,KnY,KnM,KnS, T,C,D,V,Z,LV,DT,@ D：KnY,KnM,KnS,T, C,D,V,Z,LV,DT
	BCDP		√		
	DBCD	32 位	-		
	DBCDP		√		
BIN 转换	BIN	16 位	-	[BIN S D] 将 S 中的 BCD 码转换成 BIN 码传送到 D 中	S：KnX,KnY,KnM,KnS, T,C,D,V,Z,LV,DT,@ D：KnY,KnM,KnS,T, C,D,V,Z,LV,DT
	BINP		√		
	DBIN	32 位	-		
	DBINP		√		

CMP

指令说明：

CMP 为 16 位连续型比较指令，对比较值 S1 和 S2 的数据进行比较，根据其结果（小于、相等、大于），使 Dn、Dn_{+1}、Dn_{+2} 其中一个为 ON。

条件 / 比较指令	S1 > S2	S1 = S2	S1 < S2
[CMP S1 S2 Dn]	Dn 为 ON	D_{n+1} 为 ON	D_{n+2} 为 ON

CMP 指令的导通条件导通后再断开时，目标操作数 D 保持在 CMP 导通时的状态。

操作数：

S1：KnX, KnY, KnM, T,C,D,K,H,V,Z,LV,DT,@

S2：KnX, KnY, KnM, T,C,D,K,H,V,Z,LV,DT,@

D： Y,M,S,@

指令格式：[CMP S1 S2 D]

编程示例：

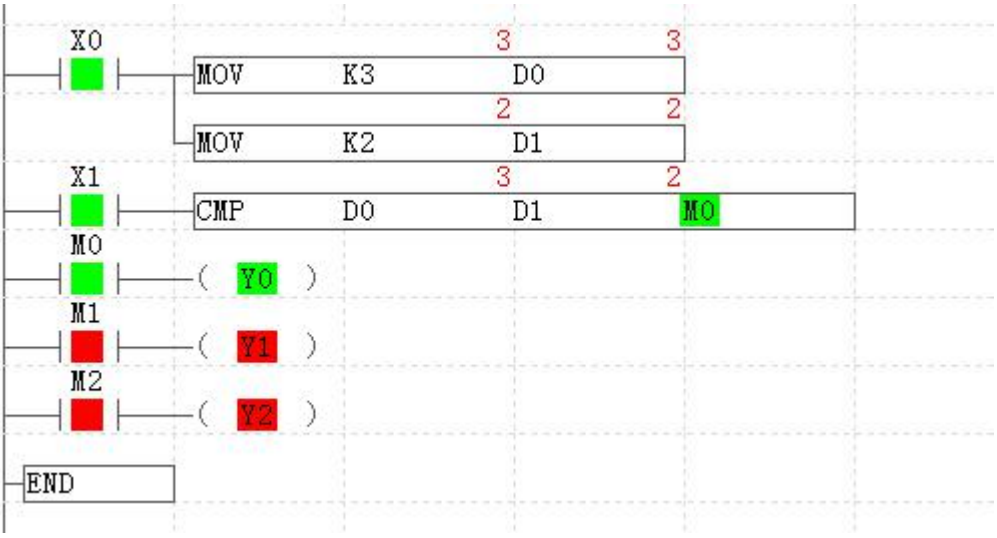
X0 接通，比较 D0 跟 D1 里面的数据，结果会影响 M0 开始的 3 个地址。

D0>D1 时，M0=1，M1=0，M2=0Y0=1；

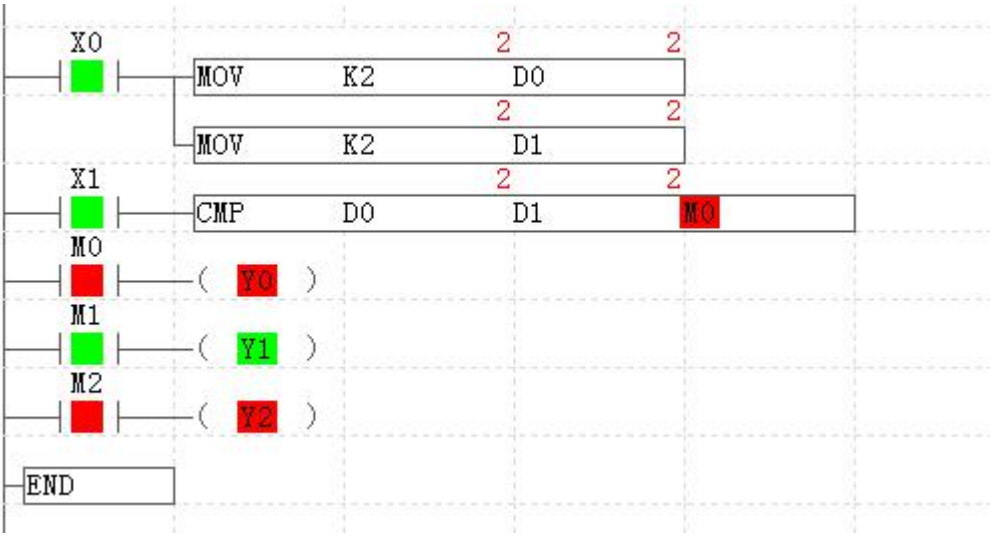
D0=D1 时，M0=0，M1=1，M2=0Y1=1；

D0<D1 时，M0=0，M1=0，M2=1Y2=1。

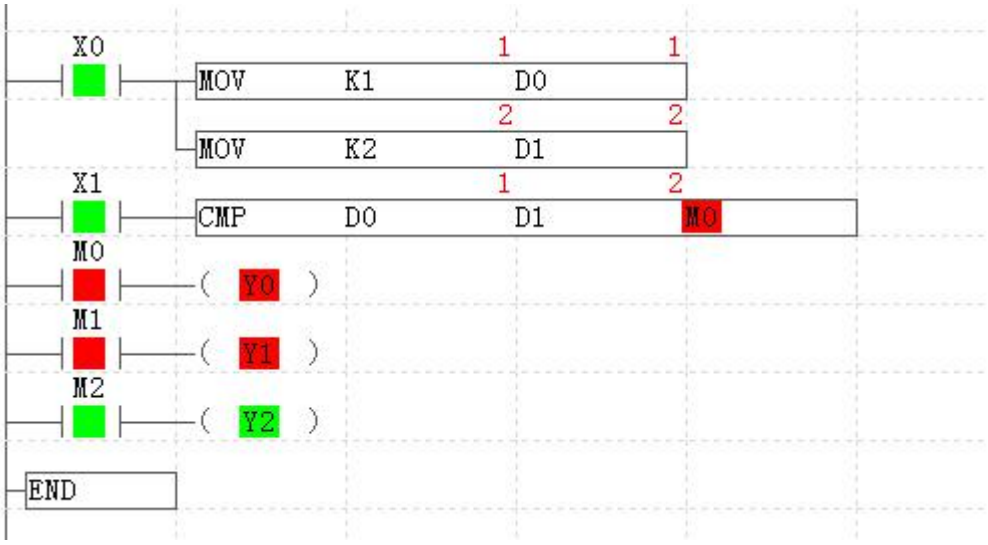
当 D0>D1 时，导通 M0。



当 D0=D1 时，导通 M1。

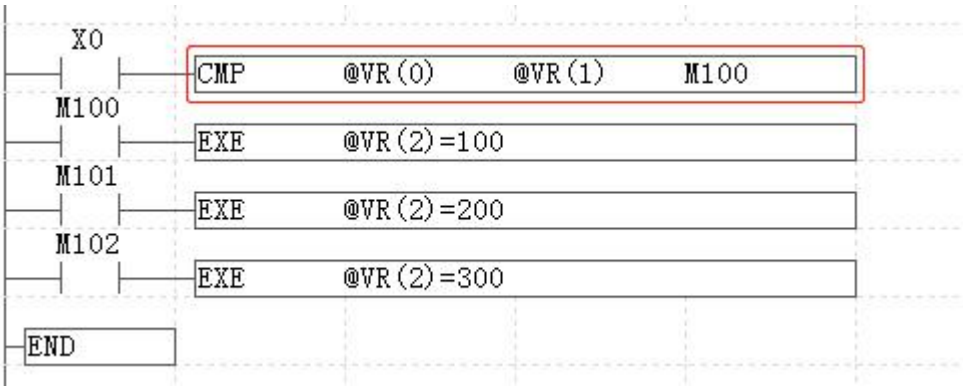


当 D0<D1 时，导通 M2。



CMP 指令导通后，断开 X1，M0 仍保持导通状态。

如下，X10 接通，比较 VR(0)跟 VR(1)里面的数据，结果会影响 M100 开始的 3 个地址。



CMPP

指令说明：

CMPP 为 16 位脉冲型比较指令，对比较值 S1 和 S2 的数据进行比较，根据其结果（小于、相等、大于），使 Dn、Dn_{+1}、Dn_{+2} 其中一个为 ON。

条件 / 比较指令	S1 > S2	S1 = S2	S1 < S2
[CMPP S1 S2 Dn]	Dn 为 ON	Dn_{+1} 为 ON	Dn_{+2} 为 ON

CMPP 指令的导通条件导通后再断开时，目标操作数 D 保持在 CMPP 导通时的状态。

操作数：

S1：KnX, KnY, KnM, T,C,D,K,H,V,Z,LV,DT,@

S2：KnX, KnY, KnM, T,C,D,K,H,V,Z,LV,DT,@

D： Y,M,S,@

指令格式：[CMPP S1 S2 D]

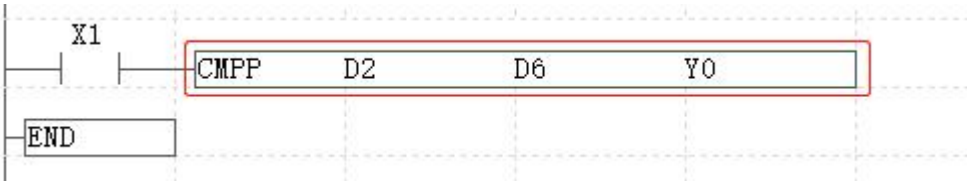
编程示例：

X1 从 OFF 变为 ON 时，比较指令执行一次，比较 D2 跟 D6 里面的数据，结果会影响 Y0，Y1，Y2 的状态。

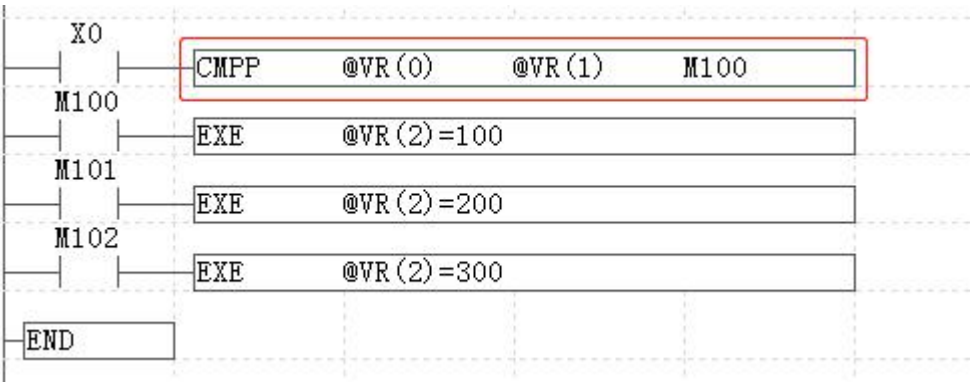
D2>D6 时，Y0 为 ON；

D2=D6 时，Y1 为 ON；

D2<D6 时，Y2 为 ON。



X0 接通，比较 VR(0)跟 VR(1)里面的数据，结果会影响 M100 开始的 3 个地址，当 VR(0)与 VR(1)的数据有变化时，需要再次关闭开启 X0，CMPP 比较指令才会执行。



DCMP

指令说明：

DCMP 为 32 位连续型比较指令，对比较值[S1+1,S1]和[S2+1,S2]的数据进行比较，根据其结果（小于、相等、大于），使 Dn、Dn_{+1}、Dn_{+2} 其中一个为 ON。

条件 / 比较指令	S1 > S2	S1 = S2	S1 < S2
[DCMP S1 S2 Dn]	Dn 为 ON	Dn_{+1} 为 ON	Dn_{+2} 为 ON

※ [S1+1,S1] [S2+1,S2]表示的是双字占据两个相邻的 16 位数据组合，例 D0 表示的就是 D1D0，D1 为高 16 位，D0 为低 16 位，D2 表示 D3D2。

DCMP 指令的导通条件导通后再断开时，目标操作数 D 保持在 DCMP 导通时的状态。

操作数：

S1：KnX, KnY, KnM, T,C,D,K,H,V,Z,LV,DT,@

S2：KnX, KnY, KnM, T,C,D,K,H,V,Z,LV,DT,@

D： Y,M,S,@

指令格式：[DCMP S1 S2 D]

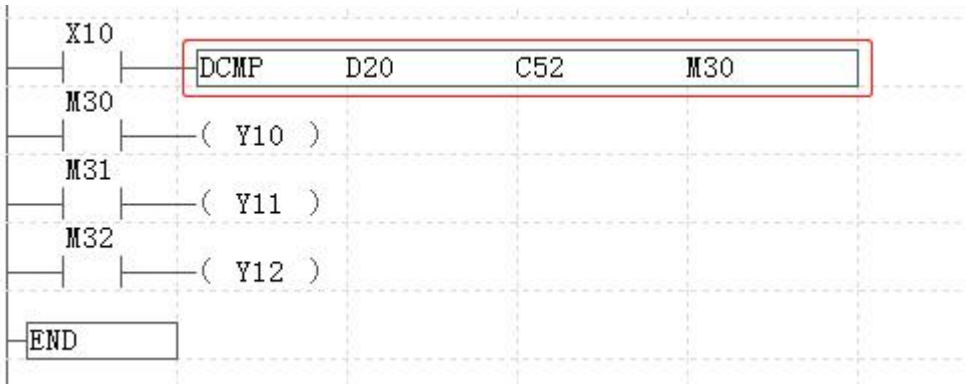
编程示例：

X10 接通，比较 D21D20 组合的 32 数据与 C53C52 组合的数据，三种结果会影响 M30 开始的三个地址。

D21D20>C53C52 时，M30 为 ON，Y10 输出；

D21D20=C53C52 时，M31 为 ON，Y11 输出；

D21D20<C53C52 时，M32 为 ON，Y12 输出。



DCMPP

指令说明：

DCMPP 为 32 位脉冲型比较指令，对比较值[S1+1,S1]和[S2+1,S2]的数据进行比较，根据其结果（小于、相等、大于），使 Dn、Dn_{+1}、Dn_{+2} 其中一个为 ON。

条件 / 比较指令	S1 > S2	S1 = S2	S1 < S2
[DCMPP S1 S2 Dn]	Dn 为 ON	Dn_{+1} 为 ON	Dn_{+2} 为 ON

※ [S1+1,S1] [S2+1,S2]表示的是双字占据两个相邻的 16 位数据组合，例 D0 表示的就是 D1D0，D1 为高 16 位，D0 为低 16 位，D2 表示 D3D2。

DCMPP 指令的导通条件导通后再断开时，目标操作数 D 保持在 DCMPP 导通时的状态。

操作数：

S1：KnX, KnY, KnM, T,C,D,K,H,V,Z,LV,DT,@

S2：KnX, KnY, KnM, T,C,D,K,H,V,Z,LV,DT,@

D： Y,M,S,@

指令格式：[DCMPP S1 S2 D]

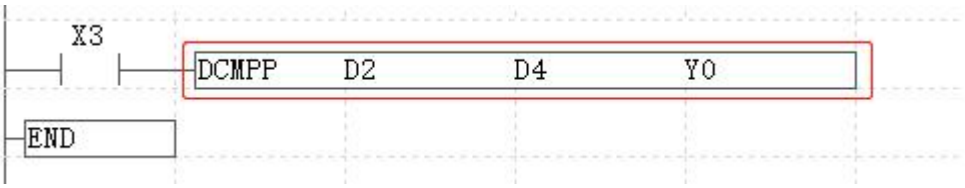
编程示例：

X3 从 OFF 变为 ON 时，32 位比较指令执行一次，比较 D3D2 跟 D5D4 里面的 32 位数据，结果会影响 Y0，Y1，Y2 的状态。

D3D2>D5D4 时，Y0 为 ON；

D3D2=D5D4 时，Y1 为 ON；

D3D2<D5D4 时，Y2 为 ON。



ZCP

指令说明：

ZCP 为 16 位连续型区间比较指令，将比较源 S 的数据与另两个比较源 S1 和 S2 的数据进行比较，比较的结果会影响从 Dn 输出的 3 个位。

条件 / 比较指令	S < S1	S1 <= S <= S2	S > S2
[ZCP S1 S2 S Dn]	Dn 为 ON	Dn_{+1} 为 ON	Dn_{+2} 为 ON

※ S1 的数据不得大于 S2 的数据，如果大于，S2 的数据将会按 S1 的数据来计算。

ZCP 指令的导通条件导通后再断开时，目标操作数 D 保持在 ZCP 导通时的状态。

操作数：

S1：KnX, KnY, KnM, T,C,D,K,H,V,Z ,LV,DT,@

S2：KnX, KnY, KnM, T,C,D,K,H,V,Z ,LV,DT,@

S： KnX, KnY, KnM, T,C,D,K,H,V,Z ,LV,DT,@

D： Y,M,S,@

指令格式：[ZCP S1 S2 S D]

编程示例：

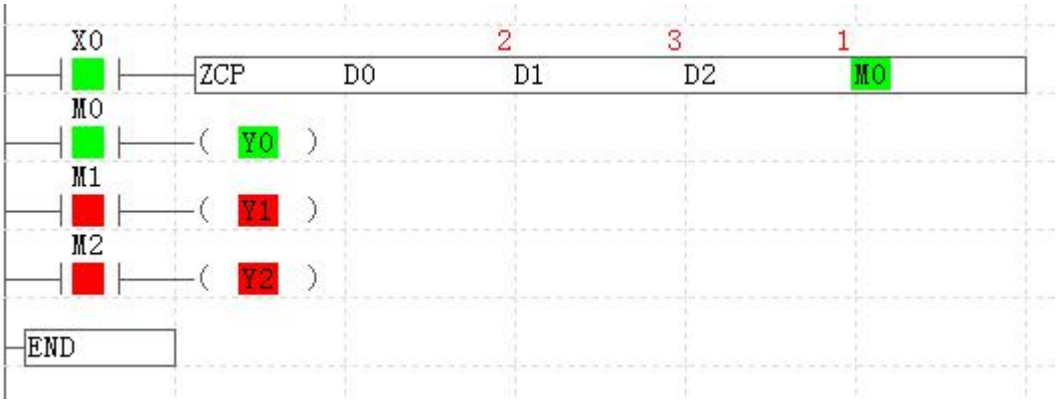
X0 接通，将 D0、D1 的数据跟 D2 进行比较，比较的结果影响从 M0 输出的 3 个位。

D2<D0 时，M0 为 ON，输出 Y0；

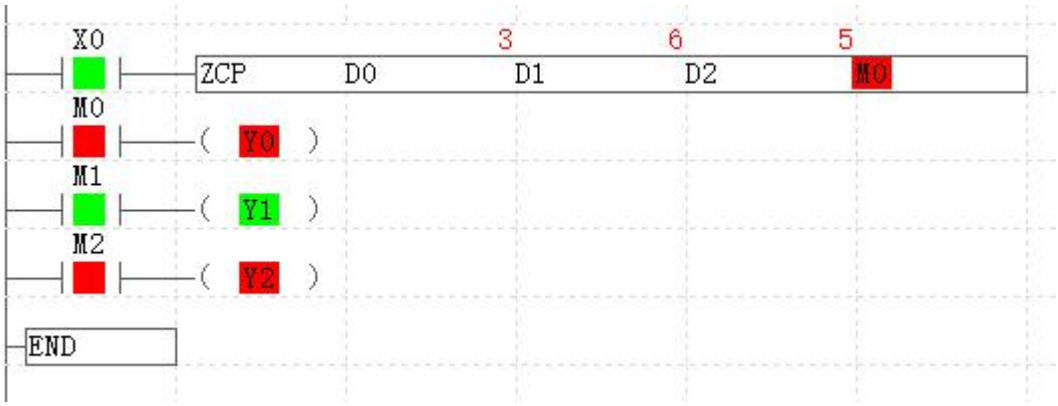
D0<=D2<=D1 时，M1 为 ON，输出 Y1；

D2>D1 时，M2 为 ON，输出 Y2。

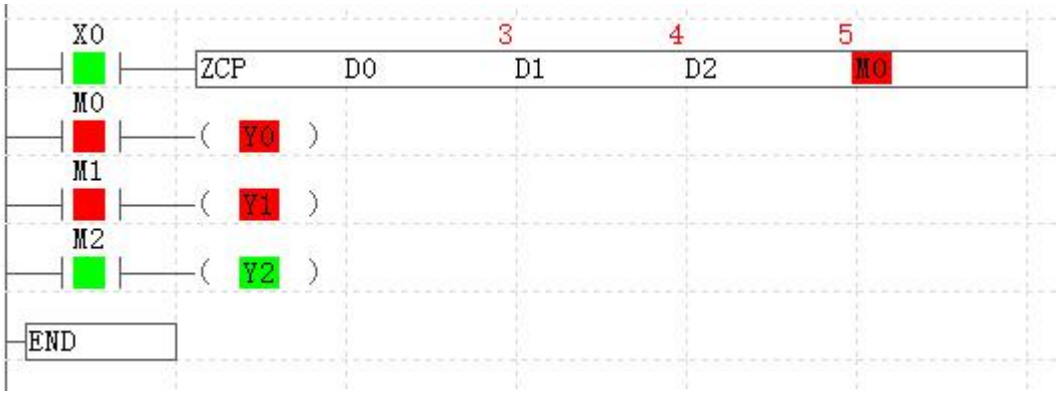
当 D2<D0 时，M0 导通。



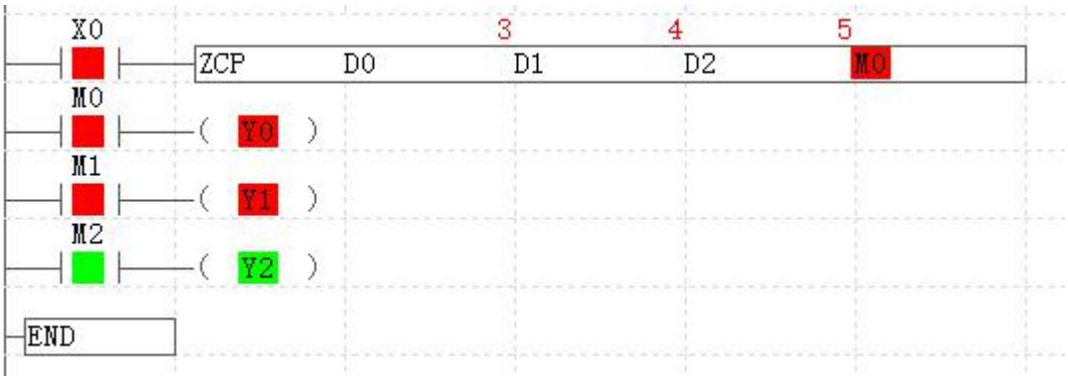
当 D0<=D2<=D1 时，M1 导通。



当 D2>D1 时，M2 导通。



ZCP 指令导通后，断开 X0，M2 仍保持导通状态。

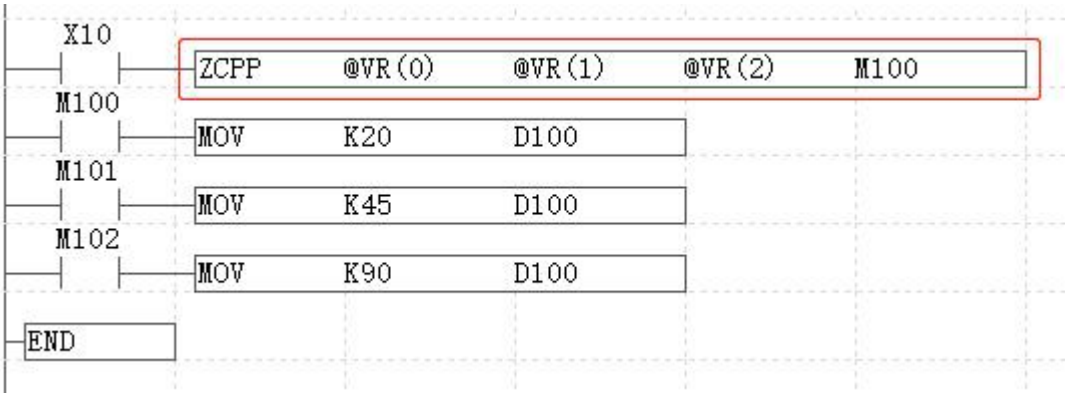


X10 接通，将寄存器 VR(0)、VR(1)的数据跟 VR(2)进行比较，比较的结果影响从 M100 输出的 3 个位。

VR(2)<VR(0)时，M100 导通；

VR(0)<=VR(2)<=VR(1)时，M101 导通；

VR(2)>VR(1)时，M102 导通。



ZCPP

指令说明：

ZCPP 为 16 位脉冲型区间比较指令，将比较源 S 的数据与另两个比较源 S1 和 S2 的数据进行比较，比较的结果会影响从 Dn 输出的 3 个位。

条件 / 比较指令	S < S1	S1 <= S <= S2	S > S2
[ZCPP S1 S2 S Dn]	Dn 为 ON	Dn_{+1} 为 ON	Dn_{+2} 为 ON

※ S1 的数据不得大于 S2 的数据，如果大于，S2 的数据将会按 S1 的数据来计算。

ZCPP 指令的导通条件导通后再断开时，目标操作数 D 保持在 ZCPP 导通时的状态。

操作数：

S1：KnX, KnY, KnM, T,C,D,K,H,V,Z ,LV,DT,@

S2：KnX, KnY, KnM, T,C,D,K,H,V,Z ,LV,DT,@

S： KnX, KnY, KnM, T,C,D,K,H,V,Z ,LV,DT,@

D： Y,M,S,@

指令格式：[ZCPP S1 S2 S D]

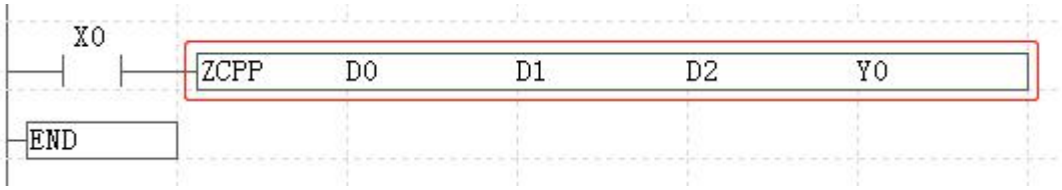
编程示例：

X0 从 OFF 变为 ON 时，区间比较指令执行一次，将 D0、D1 里面的数据跟 D2 进行比较，比较的结果影响从 Y0 输出的 3 个位。

D2<D0 时，Y0 为 ON；

D0<=D2<=D1 时，Y1 为 ON；

D2>D1 时，Y2 为 ON。



DZCP

指令说明：

DZCP 为 32 位连续型区间比较指令，将比较源[S+1,S]的数据与另两个比较源[S1+1，S1]和[S2+1，S2]的数据进行比较，根据其比较的结果，使 Dn，Dn+1，Dn+2 其中一个为 ON。

条件 / 比较指令	S < S1	S1 <= S <= S2	S > S2
[DZCP S1 S2 S Dn]	Dn 为 ON	Dn_{+1} 为 ON	Dn_{+2} 为 ON

※ S1 的数据不得大于 S2 的数据，如果大于，S2 的数据将会按 S1 的数据来计算。

※ [S1+1,S1] [S2+1，S2] [S+1,S]表示的是双字占据两个相邻的 16 位数据组合，例 D0 表示的就是 D1D0，D1 为高 16 位，D0 为低 16 位，D2 表示 D3D2，D4 表示 D5D4。

DZCP 指令的导通条件导通后再断开时，目标操作数 D 保持在 DZCP 导通时的状态。

操作数：

S1：KnX, KnY, KnM, T,C,D,K,H,V,Z ,LV,DT,@

S2：KnX, KnY, KnM, T,C,D,K,H,V,Z ,LV,DT,@

S： KnX, KnY, KnM, T,C,D,K,H,V,Z ,LV,DT,@

D： Y,M,S,@

指令格式：[DZCP S1 S2 S D]

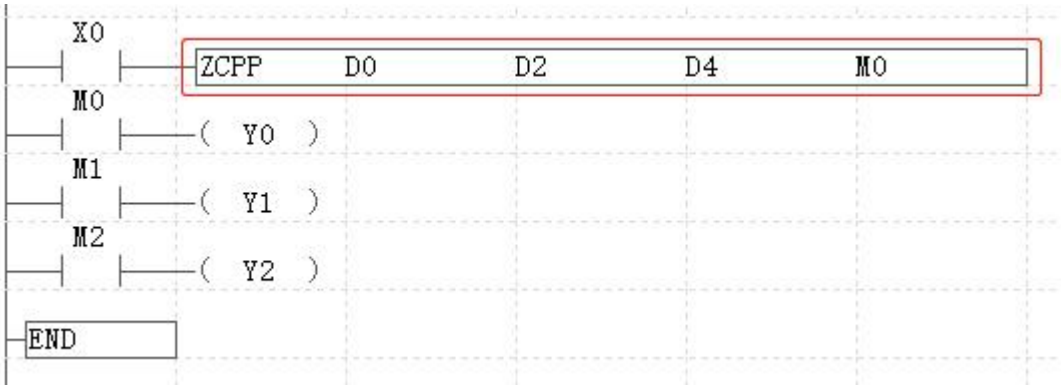
编程示例：

X0 接通，将 D1D0、D3D2 里面的 32 位数据跟 D5D4 进行比较，比较的结果影响从 M0 输出的 3 个位。

D5D4<D1D0 时，M0 为 ON，输出 Y0；

D1D0<=D5D4<=D3D2 时，M1 为 ON，输出 Y1；

D5D4>D3D2 时，M2 为 ON，输出 Y2。



DZCPP

指令说明：

DZCPP 为 32 位脉冲型区间比较指令，将比较源[S+1,S]的数据与另两个比较源[S1+1，S1]和[S2+1，S2]的数据进行比较，根据其比较的结果，使 Dn，Dn+1，Dn+2 其中一个为 ON。

条件 / 比较指令	S < S1	S1 <= S <= S2	S > S2
[DZCPP S1 S2 S Dn]	Dn 为 ON	Dn_{+1} 为 ON	Dn_{+2} 为 ON

※ S1 的数据不得大于 S2 的数据，如果大于，S2 的数据将会按 S1 的数据来计算。

※ [S1+1,S1] [S2+1，S2] [S+1,S]表示的是双字占据两个相邻的 16 位数据组合，例 D0 表示的就是 D1D0，D1 为高 16 位，D0 为低 16 位，D2 表示 D3D2，D4 表示 D5D4。

DZCPP 指令的导通条件导通后再断开时，目标操作数 D 保持在 DZCPP 导通时的状态。

操作数：

S1：KnX, KnY, KnM, T,C,D,K,H,V,Z ,LV,DT,@

S2：KnX, KnY, KnM, T,C,D,K,H,V,Z ,LV,DT,@

S： KnX, KnY, KnM, T,C,D,K,H,V,Z ,LV,DT,@

D： Y,M,S,@

指令格式：[DZCPP S1 S2 S D]

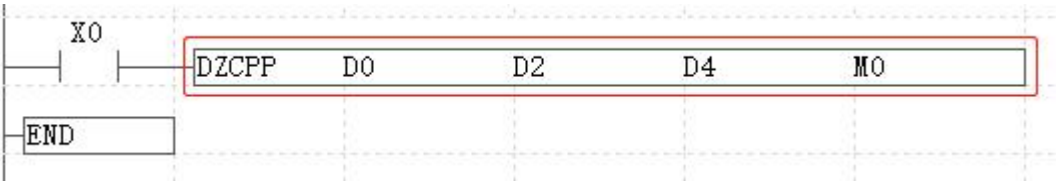
编程示例：

X0 从 OFF 变为 ON 时，32 位区间比较指令执行一次，将 D1D0、D3D2 里面的 32 位数据跟 D5D4 进行比较，比较的结果影响从 M0 输出的 3 个位。

D5D4<D1D0 时，M0 为 ON，输出 Y0；

D1D0<=D5D4<=D3D2 时，M1 为 ON，输出 Y1；

D5D4>D3D2 时，M2 为 ON，输出 Y2。



MOV

指令说明：

MOV 是 16 位连续执行型传送指令，即每个扫描周期传送一次。就是将传送源 S 的内容原封不动地复制到传送目标 D 中，不改变传送源 S 的内容。

软元件传送时指定位软元件的位数的情况，最多传送 16 个（4 的倍数）位软元件。

传送的数值范围：-32768 -+32767

操作数：

S：KnX, KnY, KnM, KnS,T,C,D,K,H,V,Z,LV,DT,@

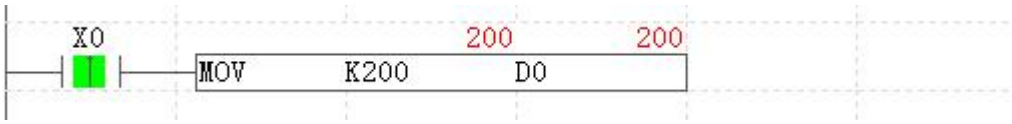
D：KnY, KnM, KnS, T,C,D,V,Z,LV,DT

指令格式：[MOV　S　D]

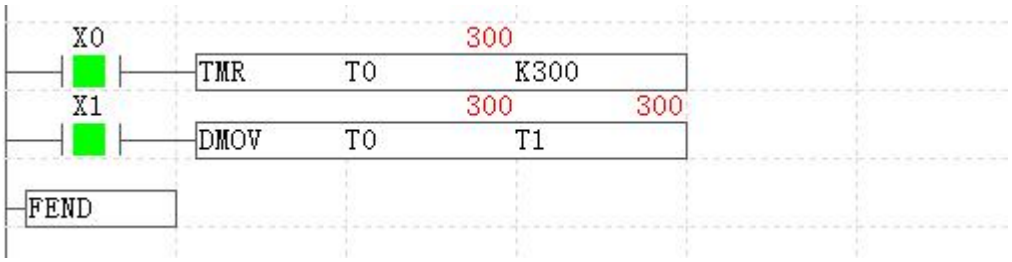
编程示例：

（1）、字软元件传送的情况

条件满足，MOV 指令将常数 200 传送到 D0。

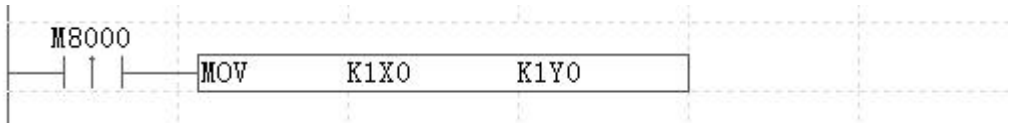


条件满足将 T0 的数据 300 传入 T1。



（2）、位软元件传送的情况

条件满足，MOV 指令将 X0X1X2X3 的位状态依次传送给 Y0Y1Y2Y3。



传送目标软元件的位状态：

执行前	S：K1X0				D：K1Y0			
	X3	X2	X1	X0	Y3	Y2	Y1	Y0
	ON	OFF	ON	OFF	OFF	OFF	OFF	ON
执行后	S：K1X0				D：K1Y0			
	X3	X2	X1	X0	Y3	Y2	Y1	Y0
	ON	OFF	ON	OFF	ON	OFF	ON	OFF

MOVP

指令说明：

MOVP 是 16 位脉冲执行型传送指令，即指令激活一次，执行一次传送。就是将传送源 S 的内容原封不动地复制到传送目标 D 中，不改变传送源 S 的内容。

软元件传送时指定位软元件的位数的情况，最多传送 16 个（4 的倍数）位软元件。

传送的数值范围：-32768 - +32767

操作数：

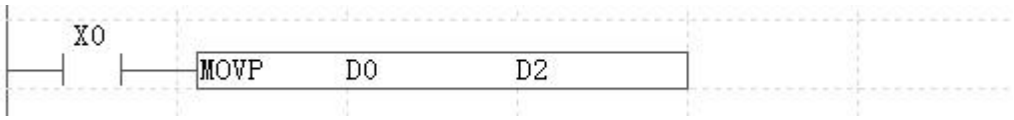
S：KnX, KnY, KnM, KnS,T,C,D,K,H,V,Z,LV,DT,@

D：KnY, KnM, KnS, T,C,D,V,Z,LV,DT

指令格式：[MOVP　S　D]

编程示例：

当 X0 由 OFF 变为 ON 后，只执行一次传送指令，将 D0 的值传送到 D2。如果前面条件是 M8013，那就每秒执行一次传送指令。



DMOV

指令说明：

DMOV 是 32 位连续执行型传送指令，即每个扫描周期传送一次。就是将传送源[Sn+1,Sn]的内容原封不动地复制到传送目标[Dn+1,Dn]中，不改变传送源[Sn+1,Sn]的内容。

软元件传送时指定位软元件的位数的情况，最多传送 32 个（4 的倍数）位软元件。

传送的数值范围：-2147483648 - +2147483647

操作数：

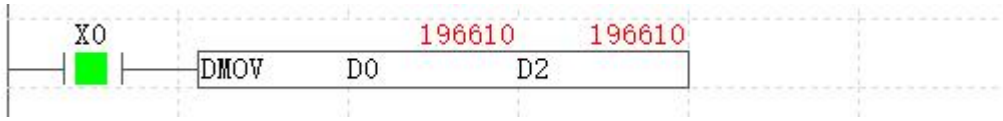
S：KnX, KnY, KnM, KnS,T,C,D,K,H,V,Z,LV,DT,@

D：KnY, KnM, KnS, T,C,D,V,Z,LV,DT

指令格式：[DMOV　S　D]

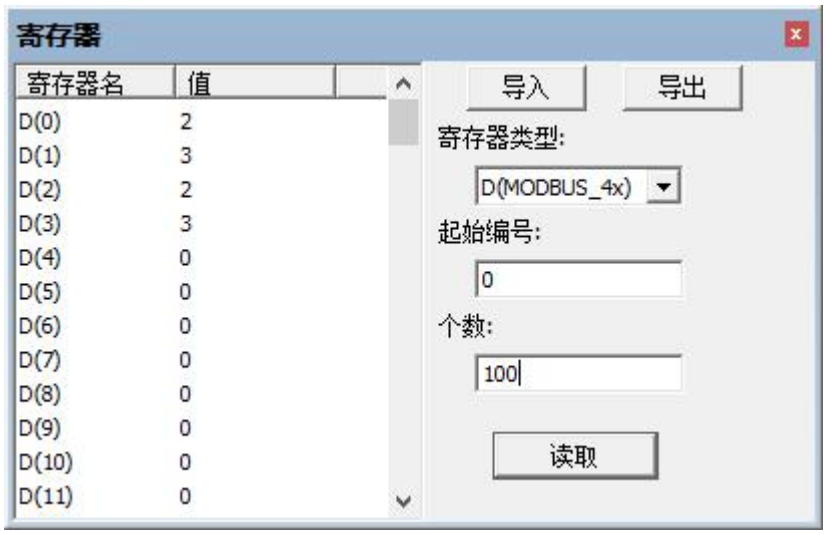
编程示例：

X0 闭合，DMOV 指令执行，将 D1D0 的值传送到 D3D2 中。



执行前：D0=2，D1=3，D2=0，D3=0；

执行后：D0=2，D1=3，D2=2，D3=3。



DMOVP

指令说明：

DMOVP 是 32 位脉冲执行型传送指令，即指令激活一次，执行一次传送。就是将传送源[Sn+1,Sn]的内容原封不动地复制到传送目标[Dn+1,Dn]中，不改变传送源[Sn+1,Sn]的内容。

软元件传送时指定位软元件的位数的情况，最多传送 32 个（4 的倍数）位软元件。

传送的数值范围：-2147483648 - +2147483647

操作数：

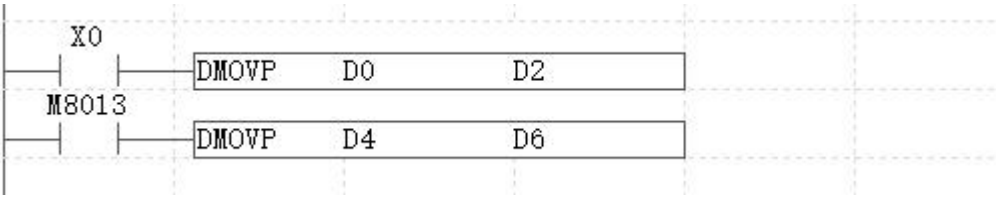
S：KnX, KnY, KnM, KnS,T,C,D,K,H,V,Z,LV,DT,@

D：KnY, KnM, KnS, T,C,D,V,Z,LV,DT

指令格式：[DMOVP　S　D]

编程示例：

当 X0 由常开到常闭后，只执行一次传送指令，X0 不变化，也就不执行传送了。如果前面条件是 M8013，那就每秒执行一次传送指令。



SMOV

指令说明：

SMOV 是 16 位连续执行型位移动指令。以位数为单位（4 位）进行数据的分配合成。

传送源 S 和传送目标先被转换成 4 位的 BCD，然后将 S 中 m1（位数从 1 开始）开始从高位到低位的 m2 个位数部分合并到 D 的第 n 位数开始的位置，最后将合并后的数据转换成 BIN 并保存到 D 中。

操作数：

S ：KnX, KnY, KnM, KnS,T,C,D,V,Z,LV,DT,@

m1：K,H,D,DT,C,T

m2：K,H,D,DT,C,T

D： KnY, KnM, KnS, T,C,D,V,Z,LV,DT

n： K,H,D,DT,C,T

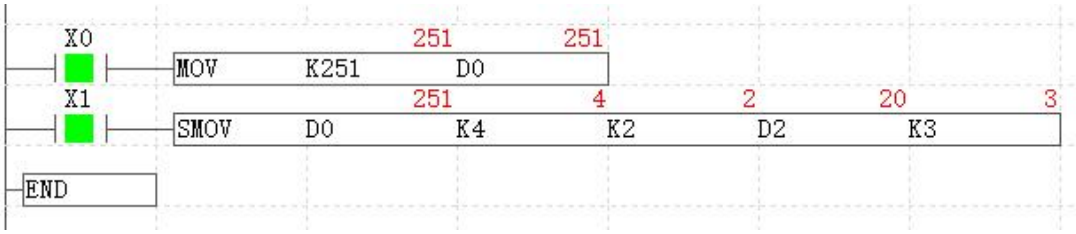
指令格式：[SMOV S m1 m2 D n]

编程示例：

X0 触点闭合，将常数 251 传入 D0，D2 初始值为 0。

X1 触点闭合，执行 SMOV 指令，将 D0 中的数值转换成 4 位 BCD 码为 0251，D2 中的数值转换成 4 位 BCD 码为 0000，将 D0 从第 4 位开始往下取 2 位数据，即 02，传递到 D2 的第 3 位开始的低 2 位，此时 D2 用 BCD 码表示为 0020。

再将 D2 的 BCD 码转换成 BIN 后保存到 D2 中。D2 的第 1 位和第 4 位传送前后数据不变化。

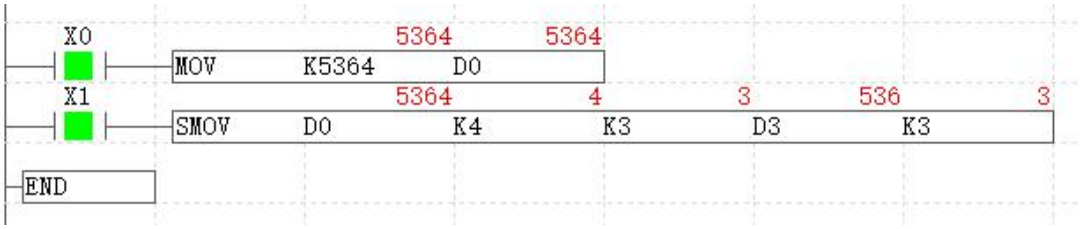


D0、D2 数值变化如下：

	D0 的 BCD 码				D2 的 BCD 码			
执行前	A4	A3	A2	A1	B4	B3	B2	B1
	0	2	5	1	0	0	0	0
	D0 的 BCD 码				D2 的 BCD 码			
执行后	A4	A3	A2	A1	B4	A4	A3	B1
	0	2	5	1	0	0	2	0

将 D0 中的数值转换成 4 位 BCD 码为 5364，D3 中的数值转换成 4 位 BCD 码为 0000，从第 4 位开始往下取 3 位数据，即 536，传递到 D3 的第 3 位开始的低 3 位，此时 D3 用 BCD 码表示为 0536。

再将 D3 的 BCD 码转换成 BIN 后保存到 D3 中。D3 的第 4 位传送前后数据不变化。



	D0 的 BCD 码				D3 的 BCD 码			
执行前	A4	A3	A2	A1	B4	B3	B2	B1
	5	3	6	4	0	0	0	0
执行后	D0 的 BCD 码				D3 的 BCD 码			

	A4	A3	A2	A1	B4	A4	A3	B1
	5	3	6	4	0	5	3	6

SMOVP

指令说明：

SMOVP 是 16 位脉冲执行型位移动指令。即指令激活一次，只执行一次指令。以位数为单位（4 位）进行数据的分配合成。

传送源 S 和传送目标先被转换成 4 位的 BCD，然后将 S 中 m1（位数从 1 开始）开始从高位到低位的 m2 个位数部分合并到 D 的第 n 位数开始的位置，最后将合并后的数据转换成 BIN 并保存到 D 中。

操作数：

S ：KnX, KnY, KnM, KnS,T,C,D,V,Z,LV,DT,@

m1：K,H,D,DT,C,T

m2：K,H,D,DT,C,T

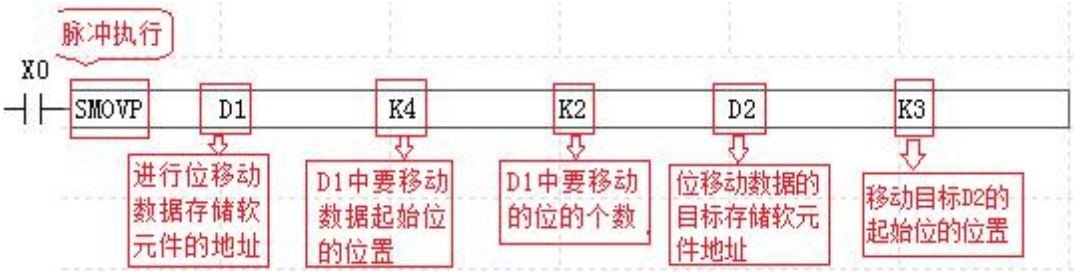
D： KnY, KnM, KnS, T,C,D,V,Z,LV,DT

n： K,H,D,DT,C,T

指令格式：[SMOVP S m1 m2 D n]

编程示例：

具体数值变化参见 SMOV 示例。



CML

指令说明：

CML 为 16 位连续执行型反转传送指令。是以位为单位对数据进行反转并传送复制。是将传送源 S 的数据（自动转换成二进制数）逐位取反（0→1,1→0）后，传送到 D。

在[S]中指定常数（K）时，会自动转换为 BIN。

取反传送值的范围为：-32768 ~ +32767

操作数：

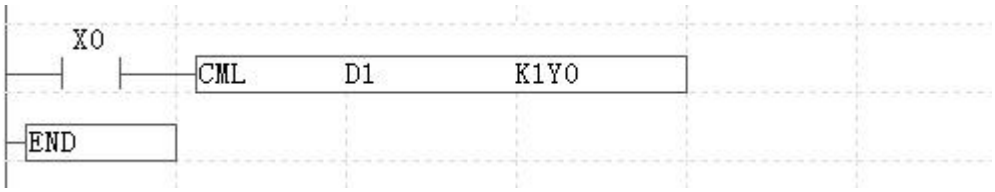
S：KnX, KnY, KnM, KnS,T,C,D,K,H,V,Z,LV,DT,@

D：KnY, KnM, KnS, T,C,D,V,Z,LV,DT,@

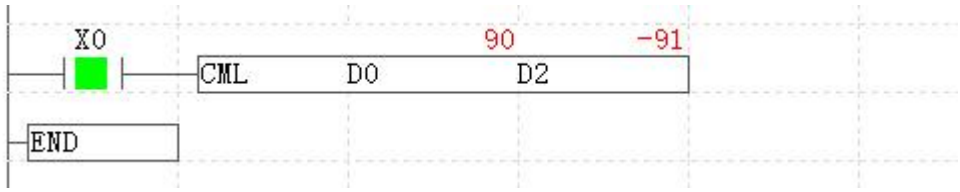
指令格式：[CML S D]

编程示例：

下图中，传送源 D1 为 16 位，而目标地址 K1Y0 仅 4 位软元件，传送时，把 D1 中最低位 4 位（b3~b0）反转传送至 Y3~Y0。



下图中，D0 的数据为 90，转换成二进制数 0000,0000,0101,1010，经过逐位取反后的数据为 1111,1111,1010,0101，复制传送到 D2，D2 的数据为-91。（注意符号位也是要反转的）



CMLP

指令说明：

CMLP 为 16 位脉冲执行型反转传送指令，即指令激活一次，执行一次传送。是以位为单位对数据进行反转并传送复制。是将传送源 S 的数据（自动转换成二进制数）逐位取反（0→1,1→0）后，传送到 D。

在[S]中指定常数（K）时，会自动转换为 BIN。

取反传送值的范围为：-32768 ~ +32767

操作数：

S：KnX, KnY, KnM, KnS,T,C,D,K,H,V,Z,LV,DT,@

D：KnY, KnM, KnS, T,C,D,V,Z,LV,DT,@

指令格式：[CMLP　S　D]

编程示例：

X0 由 OFF 变为 ON 时，上升沿脉冲激活指令 CMLP，指令执行一次，常数 K10 自动转

换为二进制 1010 取反传送给 K1Y0，所以 K1Y0 的二进制为 0101，它们的位状态表示为 Y0=1，Y1=0，Y2=1，Y3=0。

DCML

指令说明：

DCML 为 32 位连续执行型反转传送指令。是以位为单位对数据进行反转并传送复制。是将传送源 S 的数据（自动转换成二进制数）逐位取反（0→1,1→0）后，传送到 D。

在[S]中指定常数（K）时，会自动转换为 BIN。

取反传送值的范围为：-2147483648 ~ +2147483647

操作数：

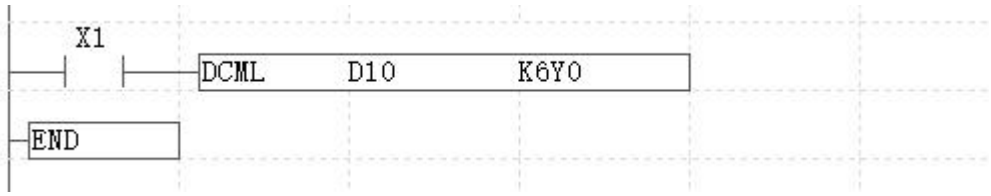
S ： KnX, KnY, KnM, KnS,T,C,D,K,H,V,Z,LV,DT,@

D ： KnY, KnM, KnS, T,C,D,V,Z,LV,DT,@

指令格式：[DCML S D]

编程示例：

满足条件将 32 位 D11D10 的数据低 24 位反转传送到 Y0~Y27 共 24 个位软元件里。



D10	b31	~	b24	b23	b22	b21	b20	~	b7	b6	b5	b4	b3	b2	b1	b0
	1	~	1	0	1	0	1	~	0	0	1	1	1	0	1	0

符号位（0=正数，1=负数） X1=ON

K1Y0	0	~	0	1	0	1	0	~	1	1	0	0	0	1	0	1
	Y37	~	Y30	Y27	Y26	Y25	Y24	~	Y7	Y6	Y5	Y4	Y3	Y2	Y1	Y0

不变化 反转传送的数据

DCMLP

指令说明：

DCMLP 为 32 位脉冲执行型反转传送指令，即指令激活一次，执行一次传送。是以位为

单位对数据进行反转并传送复制。是将传送源 S 的数据（自动转换成二进制数）逐位取反（0→1,1→0）后，传送到 D。

在[S]中指定常数（K）时，会自动转换为 BIN。

取反传送值的范围为：-2147483648 ~ +2147483647

操作数：

S ： KnX, KnY, KnM, KnS,T,C,D,K,H,V,Z,LV,DT,@

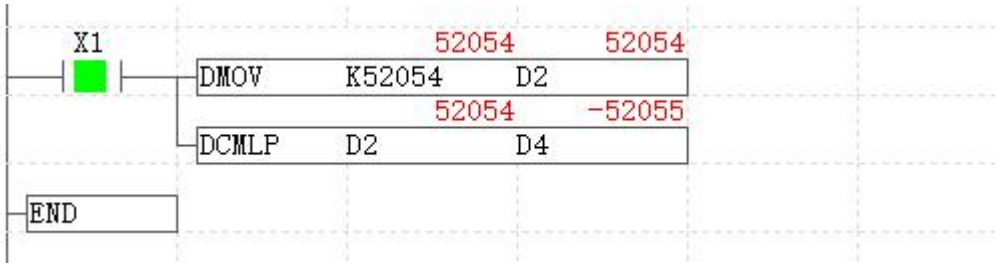
D ： KnY, KnM, KnS, T,C,D,V,Z,LV,DT,@

指令格式：[DCMLP S D]

编程示例：

假设D3D2的值为52054，转换成二进制为0 1100 1011 0101 0110，对其取反的结果D5D4的值为-52055，二进制为1 0011 0100 1010 1001。

解释：对于带符号的数，计算机存储时为其补码形式，因为1 0011 0100 1010 1001除去第一位符号位，其它位取反得其反码，1 1100 1011 0101 0110，再加1得其补码1 1100 1011 0101 0111。最后得到的数据1 1100 1011 0101 0111转化为十进制显示出来的是-52055。



BMOV

指令说明：

BMOV 是 16 位连续执行型成批传送指令。是将传送源[S]开始的 n 个数据成批传送到[D]开始的 n 点的软元件中。

操作数：

S ： KnX, KnY, KnM, KnS,T,C,D,LV,DT

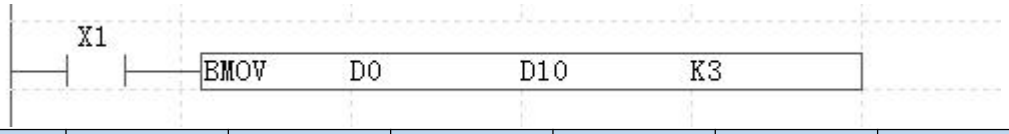
D ： KnY, KnM, KnS, T,C,D,LV,DT

n ： D,K,H

指令格式：[BMOV S D n]

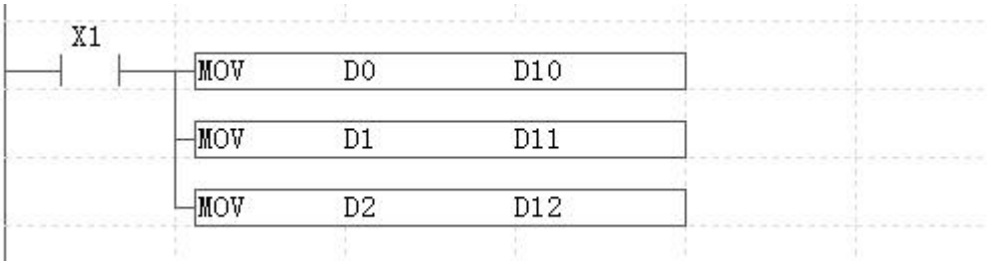
编程示例：

当 X1 接通后，BMOV 指令执行的结果为：D0 的数据至 D10，D1 的数据至 D11，D2 的数据至 D12。



软元件	D0	D1	D2	D10	D11	D12
执行前	30	40	50	0	0	0
执行后	30	40	50	30	40	50

上方例子可用如下 MOV 指令等效替代。



BMOVP

指令说明：

BMOVP 是 16 位脉冲执行型成批传送指令，即指令激活一次，执行一次传送。是将传送源[S]开始的 n 个数据成批传送到[D]开始的 n 点的软元件中。

带有位数指定的位软元件的情况下，[S]的[D]要采用相同的位数。

操作数：

S ： KnX, KnY, KnM, KnS,T,C,D,LV,DT

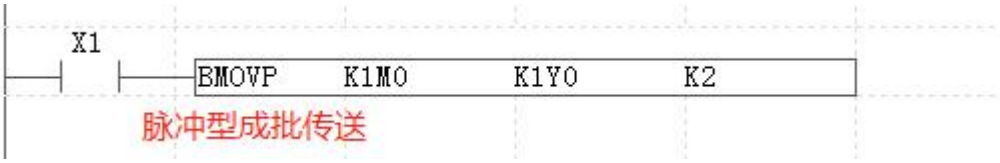
D ： KnY, KnM, KnS, T,C,D,LV,DT

n ： D,K,H

指令格式：[BMOVP S D n]

编程示例：

X1 在断开到闭合的瞬间，产生的上升沿脉冲激活 BMOVP 指令，此时 n 为 2，表示取两个位软元件，将 M0~M7 的位状态传送给 Y0~Y7。



FMOV

指令说明：

FMOV 是 16 位连续执行型多点传送指令。是将传送源[S]的数据多点传送到[D]开始的 n 点软元件中。可以用 FMOV K0 对部分指定的寄存器清零。

※ n 点的软元件内容都相同。

※ 当 n 指定的个数超过了软元件的编号范围时，在可能的范围内传送。

操作数：

S ： KnX, KnY, KnM, KnS,T,C,D,K,H,V,Z,LV,DT,@

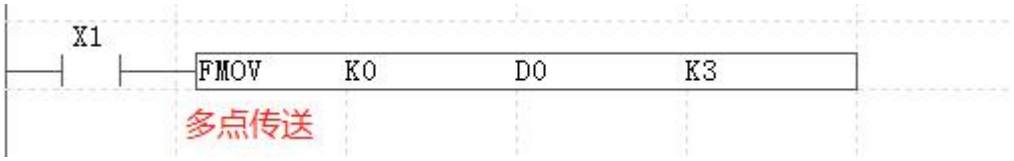
D ： KnY, KnM, KnS, T,C,D,LV,DT

n ： K,H

指令格式：[FMOV S D n]

编程示例：

条件满足后，多点传送指令 FMOV 将常数 0 传送到 D0，D1，D2 中，不管之前 D0~D2 里面的数值是多少，指令执行后 D0~D2 的数值为 0，可以用 FMOV　K0 对部分指定的寄存器清零。



假设执行前 D0~D2 的值	指令 FMOV 执行	执行后 D0~D2 的值
D0=154	K0→D0	D0=0
D1=40	K0→D1	D1=0
D2=3000	K0→D2	D2=0

FMOVP

指令说明：

FMOVP 是 16 位脉冲执行型多点传送指令，即指令激活一次，执行一次传送。是将传送源[S]的数据多点传送到[D]开始的 n 点软元件中。可以用 FMOVP K0　对部分指定的寄存器清零。

※　n 点的软元件内容都相同。

※　当 n 指定的个数超过了软元件的编号范围时，在可能的范围内传送。

操作数：

S ： KnX, KnY, KnM, KnS,T,C,D,K,H,V,Z,LV,DT,@

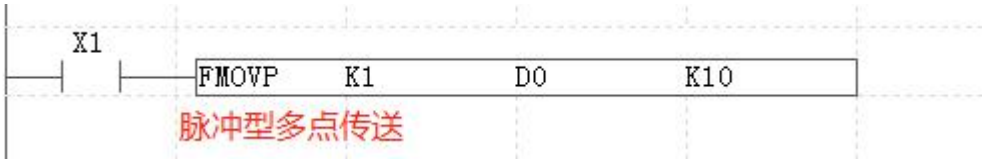
D ： KnY, KnM, KnS, T,C,D,LV,DT

n ： K,H

指令格式：[FMOVP　S　D　n]

编程示例：

当 X0 由断开到闭合的瞬间产生一个上升沿脉冲来激活 FMOVP 指令，执行后的结果为：D0~D9 共 10 个数据寄存器里面的数据全部为 1。



DFMOV

指令说明：

DFMOV 是 32 位连续执行型多点传送指令。是将传送源[Sn+1,S]的数据多点传送到[Dn+1,D]开始的 n 点的 32 位软元件中。可以用 DFMOV K0 对部分指定的寄存器清零。

※　n 点的 32 位软元件内容都相同。

※　当 n 指定的个数超过了软元件的编号范围时，在可能的范围内传送。

操作数：

S ： KnX, KnY, KnM, KnS,T,C,D,K,H,V,Z,LV,DT,@

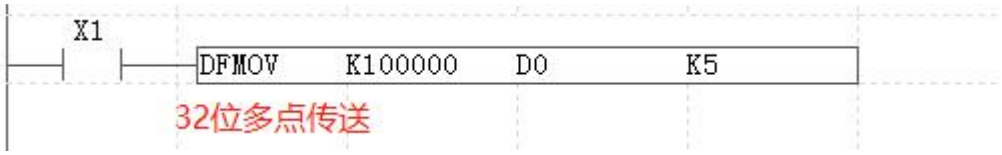
D ： KnY, KnM, KnS, T,C,D,LV,DT

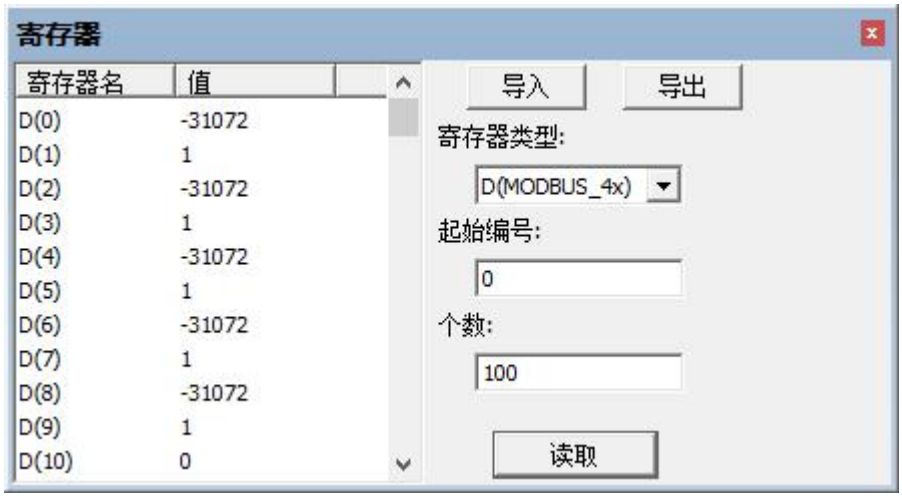
n ： K,H

指令格式：[DFMOV S D n]

编程示例：

由于是 32 位指令，每个数据采用连续两个数据寄存器 D 空间，条件满足后将 32 位的常数值 K100000 传送到 D1D0~D9D8 连续的五组寄存器里面。





DFMOVP

指令说明：

FMOVP 是 32 位脉冲执行型多点传送指令，即指令激活一次，执行一次传送。是将传送源[S]的数据多点传送到[D]开始的 n 点软元件中。可以用 DFMOVP K0 对部分指定的寄存器清零。

※ n 点的 32 位软元件内容都相同。

※ 当 n 指定的个数超过了软元件的编号范围时，在可能的范围内传送。

操作数：

S ： KnX, KnY, KnM, KnS,T,C,D,K,H,V,Z,LV,DT,@

D ： KnY, KnM, KnS, T,C,D,LV,DT

n ： K,H

指令格式：[DFMOVP S D n]

编程示例：

条件满足后，将 D1D0 的数据传送给 D3D2、D5D4 连续 2 个空间。

XCH

指令说明：

XCH 是 16 位连续执行型交换指令，将源操作数 D1 和目标操作数 D2 的数据交换。

交换数据的范围值为：-32768 ~ +32767

操作数：

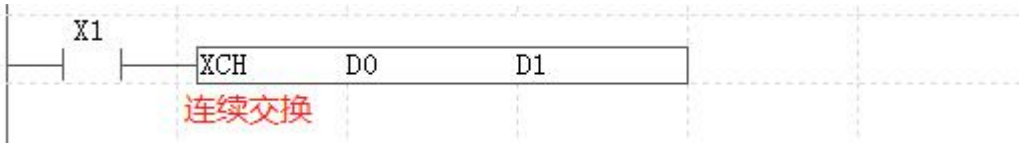
D1：KnY, KnM, KnS,T,C,D,V,Z,LV,DT

D2：KnY, KnM, KnS, T,C,D,V,Z,LV,DT

指令格式：[XCH D1 D2]

编程示例：

X1 处于导通状态时，XCH 指令每个扫描周期将 D0 与 D1 执行一次数据交换，使得目标操作数不停的发生变化。



软元件	执行前	第一个扫描周期	第二个扫描周期
D0	10	36	10
D1	36	10	36

XCHP

指令说明：

XCHP 是 16 位脉冲执行型交换指令，即指令激活一次，执行一次交换，将源操作数 D1 和目标操作数 D2 的数据交换。

交换数据的范围值为：-32768 ~ +32767

操作数：

D1：KnY, KnM, KnS,T,C,D,V,Z,LV,DT,

D2：KnY, KnM, KnS, T,C,D,V,Z,LV,DT

指令格式：[XCHP D1 D2]

编程示例：

闭合 X0 之后，D0 的值为 10，D1 值为 36，X1 闭合产生上升沿脉冲时，XCHP 指令执行一次，将 D0 与 D1 的数据交换一次后 D0 为 36，D1 为 10。

DXCH

指令说明：

DXCH 是 32 位连续执行型交换指令。是将源操作数 D1 和目标操作数 D2 的数据交换。交换数据的范围值为：-2147483648 ~ +2147483647

操作数：

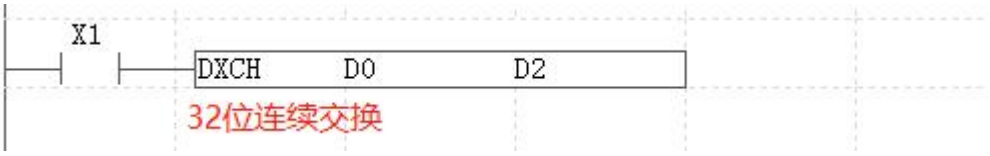
D1：KnY, KnM, KnS,T,C,D,V,Z,LV,DT,

D2：KnY, KnM, KnS, T,C,D,V,Z,LV,DT

指令格式：[DXCH　D1　D2]

编程示例：

X1 处于导通状态时，DXCH 指令每个扫描周期将 D1D0 和 D3D2 执行一次数据交换，使得目标操作数不停的发生变化。



软元件	执行前	第一个扫描周期	第一个扫描周期
D1D0	123456	654321	123456
D3D2	654321	123456	654321

DXCHP

指令说明：

DXCHP 是 32 位脉冲执行型交换指令。是将源操作数 D1 和目标操作数 D2 的数据交换。

交换数据的范围值为：-2147483648 ~ +2147483647：

操作数：

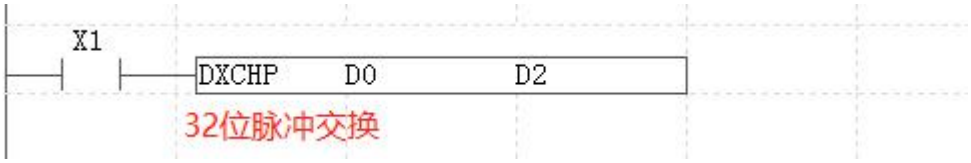
D1：KnY, KnM, KnS,T,C,D,V,Z,LV,DT,

D2：KnY, KnM, KnS, T,C,D,V,Z,LV,DT

指令格式：[DXCHP　D1　D2]

编程示例：

X1 闭合产生上升沿脉冲时，DXCHP 指令执行一次，将 D1D0 与 D3D2 的数据交换一次后，直到 X1 再次产生上升沿脉冲，执行第二次数据交换。



BCD

指令说明：

BCD 是 16 位连续执行型转换指令，是将源操作数[S]中的 BIN 码（二进制数来表示十进制数）转换成 BCD 码（4 位二进制数来表示一位十进制数）后传送到目标操作数[D]中。

BCD 码中十进制数也是和 16 进制数一样用 4 位二进制数来描述。

BCD 码的转换结果超出 0~9999 范围会出错。

操作数：

S ： KnX, KnY, KnM, KnS,T,C,D,V,Z,LV,DT,@

D ： KnY, KnM, KnS, T,C,D,V,Z,LV,DT

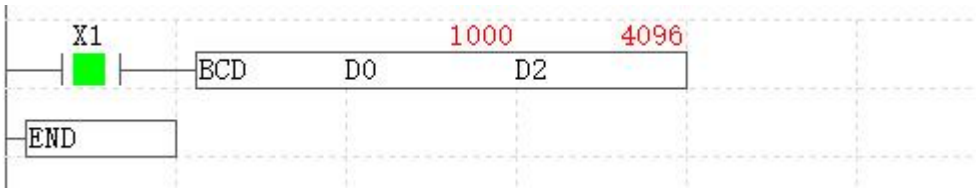
指令格式：[BCD S D]

注意：[S]和[D]指定位数时，可以参考下表。

操作数 D	位数	数据范围
K1Y0	1 位数	0~9
K2Y0	2 位数	0~99
K3Y0	3 位数	0~999
K4Y0	4 位数	0~9999

编程示例：

假设 D0 的值为 K1000，BCD 指令执行后，D0 的值为 BIN 码（二进制数表示十进制数），转换成 BCD 码（每位十进制数表示四位二进制数），也就是 0001,0000,0000,0000，传送到 K4Y0（十进制数），为 K4096。



BCDP

指令说明：

BCDP 是 16 位脉冲执行型转换指令，即指令激活一次，执行一次数据转换传送。是将源操作数[S]中的 BIN（二进制数来表示十进制数）转换成 BCD 码（4 位二进制数来表示一位

十进制数）后传送到目标操作数[D]中。

BCD 码中十进制数也是和 16 进制数一样用 4 位二进制数来描述。

BCD 码的转换结果超出 0-9999 范围会出错。

操作数：

S ： KnX, KnY, KnM, KnS,T,C,D,V,Z,LV,DT,@

D ： KnY, KnM, KnS, T,C,D,V,Z,LV,DT

指令格式：[BCDP S D]

注意：[S]和[D]指定位数时，可以参考下表。

操作数 D	位数	数据范围
K1Y0	1 位数	0~9
K2Y0	2 位数	0~99
K3Y0	3 位数	0~999
K4Y0	4 位数	0~9999

编程示例：

假设 D10 的值为 K1234，转换为 BCD 码表示为 1234，将其拆分为四位二进制，如下：

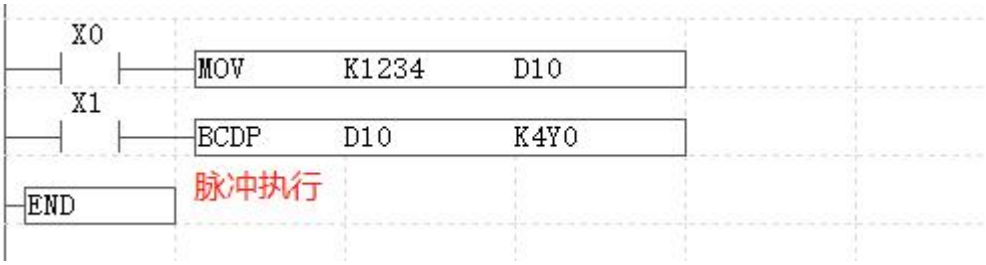
1 的二进制是 0001，对应的是 Y17、Y16、Y15、Y14

2 的二进制是 0010，对应的是 Y13、Y12、Y11、Y10

3 的二进制是 0011，对应的是 Y7、Y6、Y5、Y4

4 的二进制是 0100，对应的是 Y3、Y2、Y1、Y0

将四组二进制组合为 0001,0010,0011,0100，表示的十进制为 K4660，最后传送给 K4Y0。



DBCD

指令说明：

DBCD 是 32 位连续执行型转换指令，是将源操作数[S]中的 BIN（二进制数来表示十进制数）转换成 BCD 码（4 位二进制数来表示一位十进制数）后传送到目标操作数[D]中。

BCD 码中十进制数也是和 16 进制数一样用 4 位二进制数来描述。

BCD 码的转换结果超出 0-99999999 范围会出错

操作数：

S ： KnX, KnY, KnM, KnS,T,C,D,V,Z,LV,DT,@

D ： KnY, KnM, KnS, T,C,D,V,Z,LV,DT

指令格式：[DBCD S D]

注意：[S]和[D]指定位数时，可以参考下表。

操作数 D	位数	数据范围

K1Y0	1 位数	0~9
K2Y0	2 位数	0~99
K3Y0	3 位数	0~999
K4Y0	4 位数	0~9999
K5Y0	5 位数	0~99999
K6Y0	6 位数	0~999999
K7Y0	7 位数	0~9999999
K8Y0	8 位数	0~99999999

编程示例：

假设 D11D10 的值为 K1234567，为 BIN 码（二进制数表示十进制数），转换成 BCD 码（每位十进制数表示四位二进制数）为 0000,0001,0010,0011,0100,0101,0110,0111，传送至 D1D0，为 K19088743。

DBCDP

指令说明：

DBCDP 是 32 位脉冲执行型 BCD 转换指令，即指令激活一次，执行一次数据转换传送。是将源操作数[S]中的 BIN（二进制数来表示十进制数）转换成 BCD 码（4 位二进制数来表示一位十进制数）后传送到目标中操作数[D]中。

BCD 码中十进制数也是和 16 进制数一样用 4 位二进制数来描述。

BCD 码的转换结果超出 0-99999999 范围会出错。

操作数：

S ： KnX, KnY, KnM, KnS,T,C,D,V,Z,LV,DT,@

D ： KnY, KnM, KnS, T,C,D,V,Z,LV,DT

指令格式：[DBCDP S D]

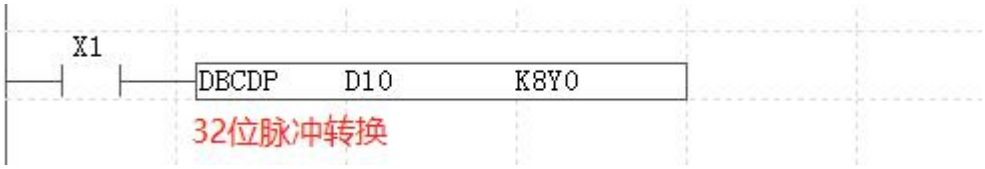
注意：[S]和[D]指定位数时，可以参考下表。

操作数 D	位数	数据范围
K1Y0	1 位数	0~9
K2Y0	2 位数	0~99
K3Y0	3 位数	0~999
K4Y0	4 位数	0~9999
K5Y0	5 位数	0~99999
K6Y0	6 位数	0~999999

K7Y0	7 位数	0~9999999
K8Y0	8 位数	0~99999999

编程示例：

X0 闭合产生一次脉冲，DBCDP 指令执行一次，将 D11D10 的 32 位二进制数据转换成 BCD 码后传送给 k8Y0（从 Y0~Y37 的 32 个位软元件）。



BIN

指令说明：

BIN 是 16 位连续执行型 BIN 转换指令，是将源操作数[S]中的 BCD 码（4 位二进制数来表示一位十进制数）转换成 BIN（二进制数来表示十进制数）后传送到目标操作数[D]中。

源操作数[S]的数值范围为：0~9999，若超过该范围会出错。

操作数：

S ： KnX, KnY, KnM, KnS,T,C,D,V,Z,LV,DT,@

D ： KnY, KnM, KnS, T,C,D,V,Z,LV,DT

指令格式：[BIN S D]

注意：[S]和[D] 指定位数时，可以参考下表。

操作数 D	位数	数据范围
K1Y0	1 位数	0~9
K2Y0	2 位数	0~99
K3Y0	3 位数	0~999
K4Y0	4 位数	0~9999

编程示例：

假设 K4Y0 的值为 K530，BIN 指令执行后，它是一个 BCD 码（4 位二进制数来表示一位十进制数）为 0000,0010,0001,0010，转换成 BIN 码（二进制数表示十进制数），传送到 D0，为 K212。

BINP

指令说明：

BINP 是 16 位脉冲执行型 BIN 转换指令，指令激活一次，执行一次数据转换。是将源操作数[S]中的 BCD 码（4 位二进制数来表示一位十进制数）转换成 BIN（二进制数来表示十进制数）后传送到目标操作数[D]中。

源操作数[S]的数值范围为：0~9999，若超过该范围会出错。

操作数：

S ： KnX, KnY, KnM, KnS,T,C,D,V,Z,LV,DT,@

D ： KnY, KnM, KnS, T,C,D,V,Z,LV,DT

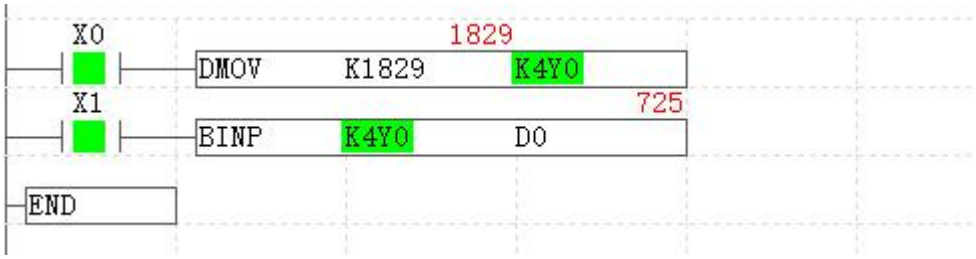
指令格式：[BINP S D]

注意：[S] 和[D] 指定位数时，可以参考下表。

操作数 D	位数	数据范围
K1Y0	1 位数	0~9
K2Y0	2 位数	0~99
K3Y0	3 位数	0~999
K4Y0	4 位数	0~9999

编程示例：

假设 K4Y0 的值为 K1829，BINP 指令执行后，它是一个 BCD 码（4 位二进制数来表示一位十进制数）为 0000,0111,0010,0101，转换成 BIN 码（二进制数表示十进制数），传送到 D0，为 K725。



DBIN

指令说明：

DBIN 是 32 位连续执行型 BIN 转换指令，是将源操作数[S]中的 BCD 码（4 位二进制数来表示一位十进制数）转换成 BIN（二进制数来表示十进制数）后传送到目标操作数[D]中。

源操作数[S]的数值范围为：0~99999999，若超过该范围会出错。

操作数：

S ： KnX, KnY, KnM, KnS,T,C,D,V,Z,LV,DT,@

D ： KnY, KnM, KnS, T,C,D,V,Z,LV,DT

指令格式：[DBIN S D]

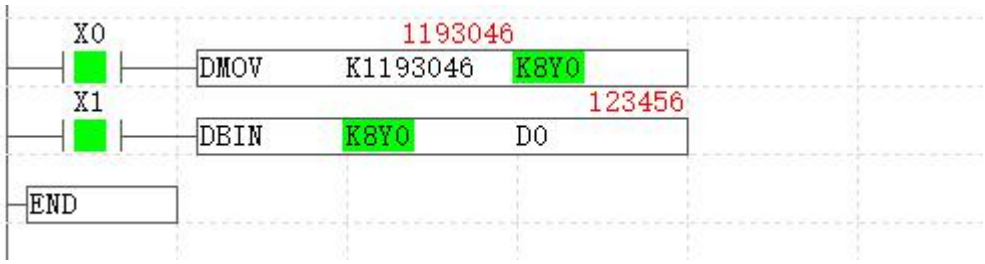
注意：[S] 和[D]指定位数时，可以参考下表。

操作数 D	位数	数据范围
K1Y0	1 位数	0~9

K2Y0	2 位数	0~99
K3Y0	3 位数	0~999
K4Y0	4 位数	0~9999
K5Y0	5 位数	0~99999
K6Y0	6 位数	0~999999
K7Y0	7 位数	0~9999999
K8Y0	8 位数	0~99999999

编程示例：

假设 K8X0 的值为 K1193046，DBIN 指令执行后，它是一个 BCD 码（4 位二进制数来表示一位十进制数）为 0000,0000,0001,0010,0011,0100,0101,0110，转换成 BIN 码（二进制数表示十进制数），传送到 D1D0，为 K123456。



DBINP

指令说明：

DBINP 是 32 位脉冲执行型 BIN 转换指令，指令激活一次，执行一次数据转换。是将源操作数[S]中的 BCD 码（4 位二进制数来表示一位十进制数）转换成 BIN（二进制数来表示十进制数）后传送到目标操作数[D]中。

源操作数[S]的数值范围为：0~99999999，若超过该范围会出错。

操作数：

S ： KnX, KnY, KnM, KnS,T,C,D,V,Z,LV,DT,@

D ： KnY, KnM, KnS, T,C,D,V,Z,LV,DT

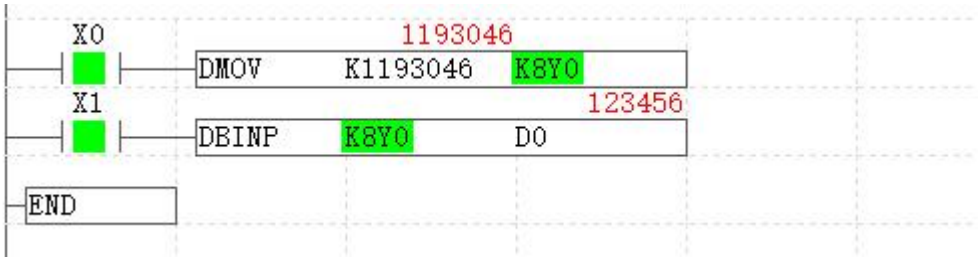
指令格式：[DBINP S D]

注意：[S] 和[D]指定位数时，可以参考下表。

操作数 D	位数	数据范围
K1Y0	1 位数	0~9
K2Y0	2 位数	0~99
K3Y0	3 位数	0~999
K4Y0	4 位数	0~9999
K5Y0	5 位数	0~99999
K6Y0	6 位数	0~999999
K7Y0	7 位数	0~9999999
K8Y0	8 位数	0~99999999

编程示例：

假设 K8X0 的值为 K1193046，X1 闭合一次，DBINP 指令便执行一次，它是一个 BCD 码（4 位二进制数来表示一位十进制数）为 0000,0000,0001,0010,0011,0100,0101,0110，转换成 BIN 码（二进制数表示十进制数），传送到 D1D0，为 K123456。



3.4 循环跳转类

分类	指令	功能	操作数类型
子 程序	LBL	表明子程序和子程序名称	S：@sub_func,ln
	CALL/CALLP	调用 LBL 子程序或 Basic 全局过程	S：@sub_func,ln
	SRET	子程序返回主程序，可选是否带返回值	KnX,KnY,KnM,KnS, T,C,D,K,H,LV,DT,@
强 制跳转	CJ/CJP	强制跳转 LBL 子程序，中间程序不执行	S：Ln,@
	CJEND/CJPEND	强制跳转到 END，中间程序不执行	无
中断	EI	允许中断	无
	DI	禁止中断	无
	IRET	中断子程序返回到主程序	无
循环	FOR	FOR 和 NEXT 之间按指定次数循环执行	S：KnX,KnY,KnM,KnS, T,C,D,K,H,V,Z,LV,DT,@
	NEXT	循环	无
程序结束	FEND	主程序结束	无
	END	程序结束，进入下一扫描周期	无
其他	WDT	对 D8000 设置的看门狗定时器刷新	无

LBL

指令说明：

标明下面是一个子程序，并指明子程序名称。

子程序返回到主程序使用 SRET 指令。

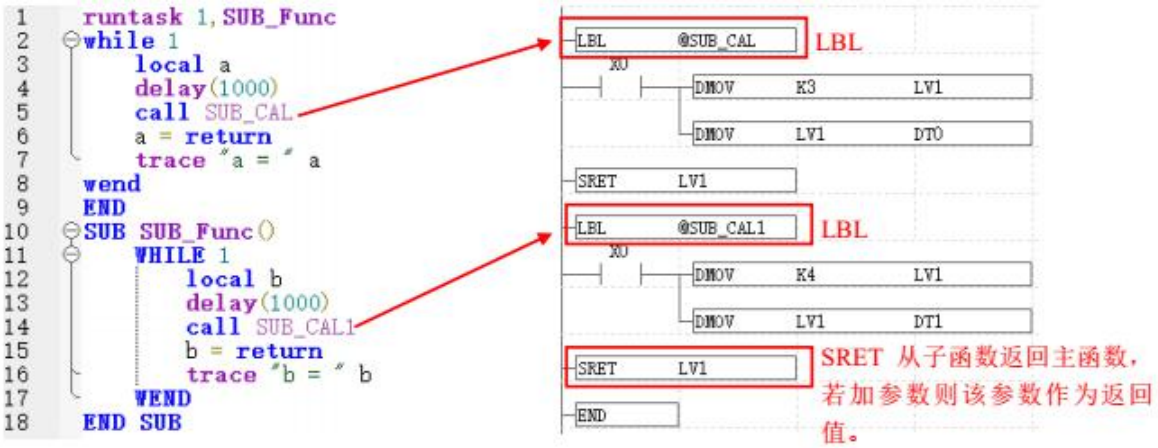
操作数：

S：@sub_func,ln

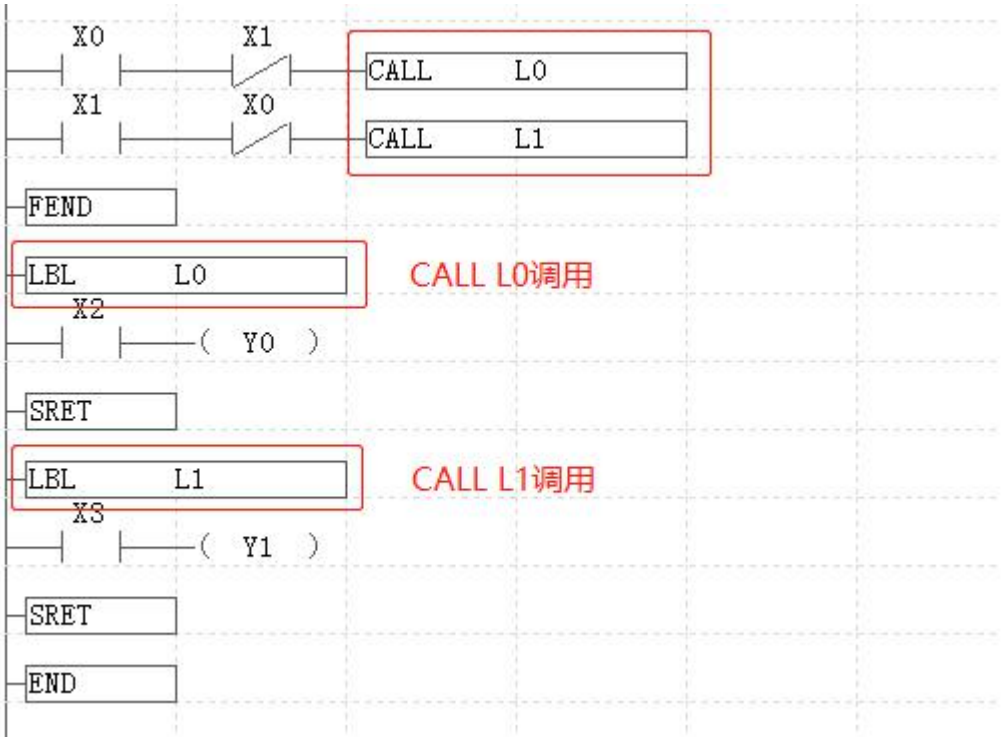
注意：LBL Ln 是给子程序编号，LBL @SUB_FUNC 是指明子程序名称。

编程示例：

1、Basic 调用 PLC 子程序



2、PLC 调用 PLC 子程序



CALL/CALLP

指令说明：

调用子程序指令，可以调用 BASIC 的全局过程，或 LBL 定义的过程。不可直接与母线相连。

操作数：

S：@sub_func,ln

此指令可以传入参数（参数必须为 32 位浮点数，如：全局变量、LV、DT），参数自动传入被调用过程的参数或 LV 寄存器中。

参数自动以 32 位浮点数格式传入。

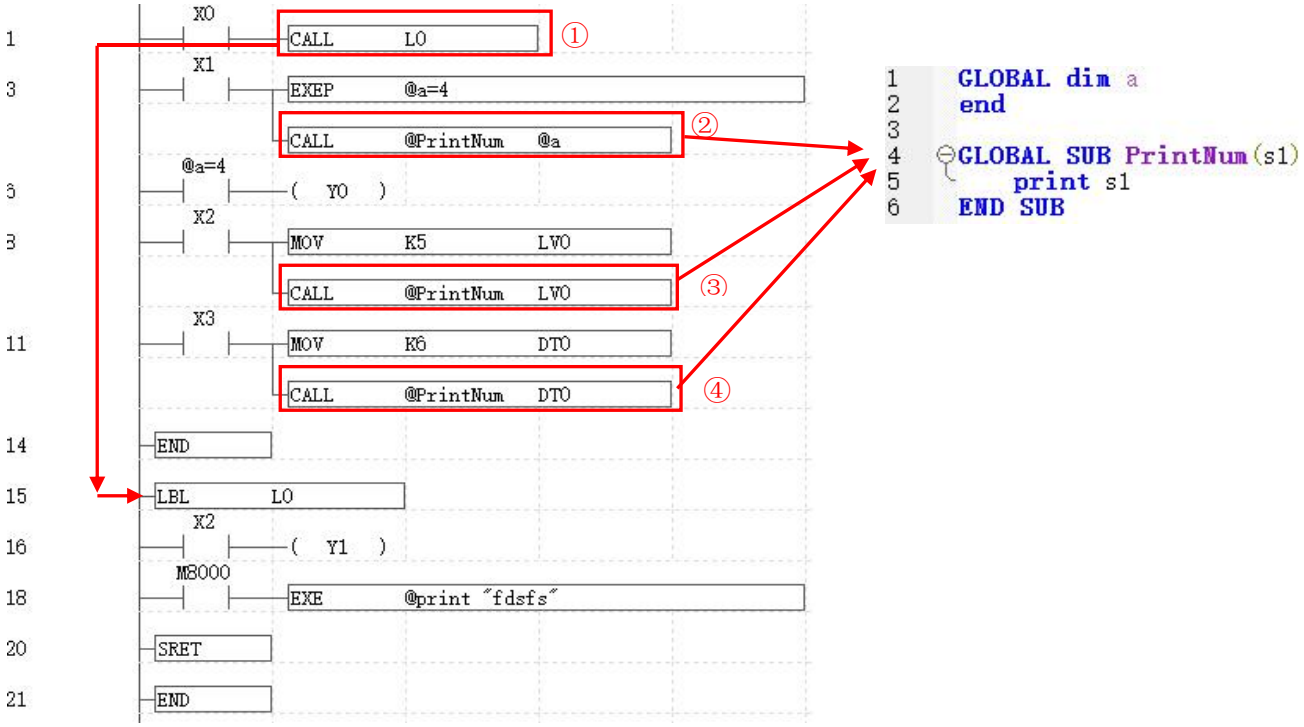
编程示例：

① 为调用 PLC 子函数；

② 为调用 BASIC 的全局过程，参数为全局变量；

③ 为调用 BASIC 的全局过程，参数为 LV；

④ 为调用 BASIC 的全局过程，参数为 DT；



SRET

指令说明：

执行了主程序中的 CALL 指令后，跳转到 LBL 指令标明的子程序，子程序执行完使用 SRET 指令返回到主程序。

操作数：

可以带上返回值，返回值是 32 位浮点格式，调用者通过 RETURN 指令获取返回值。

编程示例：

参见 LBL 指令。

CJ/CJP

指令说明：

使 CJ、CJP（脉冲型）指令开始到 LBL 标记为止的顺序程序不执行的指令。

CJ、CJP 指令跳转到的 LBL 程序不使用 SRET 指令返回，CJ、CJP 指令不可直接与母线相连。

操作数：

S：Ln,@

编程示例：

X0=ON，X1=OFF 时，CJ　L0 执行，跳转到 LBL　L0 行之后顺序扫描执行，但这两条指令之间的程序均不执行，故此时 X2 无法控制 Y0 输出。

当 X0=ON，X1=ON 时，执行过程同上。

当 X0=OFF，X1=ON 时，跳转到 LBL　L1 行之后顺序扫描执行，CJ　L1 和 LBL　L1 之间的程序均不执行。

当 X0=OFF，X1=OFF 时，CJ 指令不执行时，程序在首行和 END 之间顺序循环扫描（若写入 FEND 指令，FEND 之后的程序不扫描，需采用指令调用），此时闭合 X2，Y0 输出，闭合 X3，Y3 输出，闭合 X4，Y4 输出。

CJEND/CJPEND

指令说明：

使 CJEND、CJPEND 指令开始到程序 END 标记为止的顺序程序不执行的指令。指令触发时，程序直接跳转到 END 结束，中间过程程序不执行。

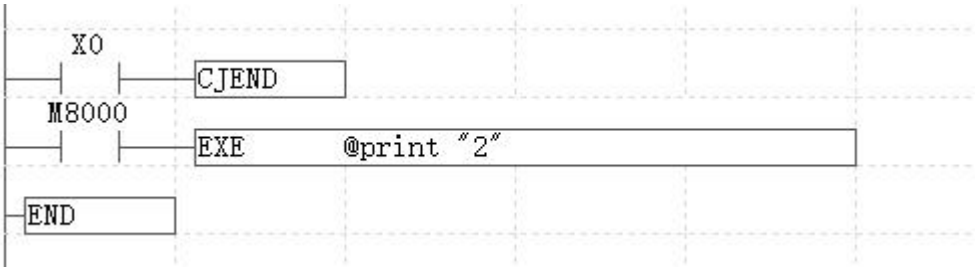
不可直接与母线相连。

操作数：

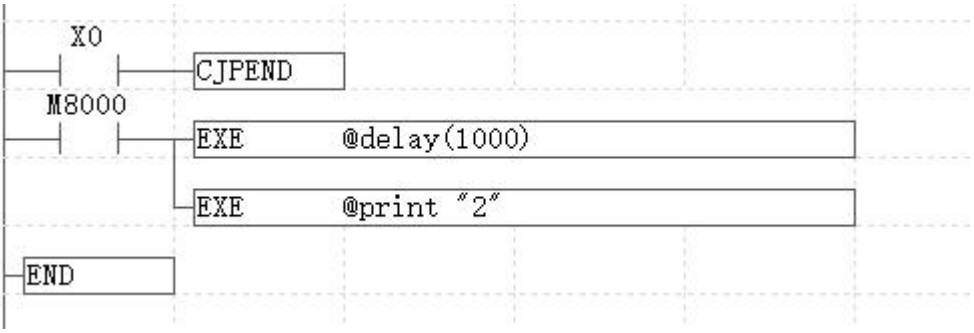
无

编程示例：

1、CJEND 连续执行型指令，X0 为 ON 时一直跳转到 END，Basic 调用打印指令不执行。



2、CJPEND 脉冲执行型指令，X0 从 OFF 变为 ON 时，跳转一次，第二个扫描周期若 X0 未产生上升沿脉冲，CJPEND 指令不执行，此时不跳转，程序顺序扫描，执行 basic 调用指令。



EI

指令说明：

EI 指令使可编程控制器变为允许中断的状态，可编程控制器通常为禁止中断的状态。

中断子程序调用后执行一个扫描周期便返回到主程序，子程序保留最后一个扫描周期的特性。

不可直接与母线相连。

操作数：

无。

编程示例：

上电，开启中断，X1 闭合，定时器 T0 开始计时，到达 2000ms，用 LBL 调用函数 ONTIMER0，定义中断条件，定时器 T0 中断开启，执行中断子程序一个扫描周期，打印“定时器中断”，T0 的常开触点闭合，定时器 T1 导通，常闭触点 T1 断开，T0 重新计时，到达 2000ms 后，再次执行中断，打印“定时器中断”。

用 LBL 调用函数 INT_ON0，INT_ON2，定义中断条件。控制器 IN(0)，IN(2)变有效中断输入口，X0 接通时，执行中断子程序一个扫描周期，Y0 输出，且 Y0 一直导通（子程序保留最后一个扫描周期的特性）。X2 接通时，执行中断子程序一个扫描周期，Y0 复位。

用 LBL 调用函数 ONPOWEROFF，定义中断条件。掉电中断开启，执行中断子程序一个扫描周期，MOV 指令将 K12345 传送到 D0，EXE 调用 BASIC 程序 VR(0)=MODBUS_REG(0)，当控制器断电后，D0 的值保存到 VR(0)中，此时 VR（0）的值为浮点数 12345.000。


M8002
EI
中断允许
X1
T1
TMR T0 K2000
T0
TMR T1 K0
X3
DI
禁止中断
FEND
//外部中断，由控制器上的输入口控制
LBL @INT_ON0
X0
(Y0)
IRET
LBL @INT_ON2
X2
EXE @ print 2
IRET
//定时器中断，定时器到达预设值开启中断
LBL @ontimer0
M8000
EXE @ print "定时器中断"
IRET
//掉电中断，控制器断点后执行此中断
LBL @onpoweroff
M8000
MOV K12345 D0
EXE @ VR(0)=modbus_reg(0)
IRET
END

DI

指令说明：

DI 指令使可编程控制器由允许中断变为禁止中断的状态。

不可直接与母线相连。

操作数：

无。

编程示例：

参见 EI 指令

IRET

指令说明：

IRET 指令使中断子程序返回到主程序。

操作数：

无。

编程示例：

参见 EI 指令。

FOR

指令说明：

从 FOR 指令开始到 NEXT 指令之间的程序按指定次数重复运行。

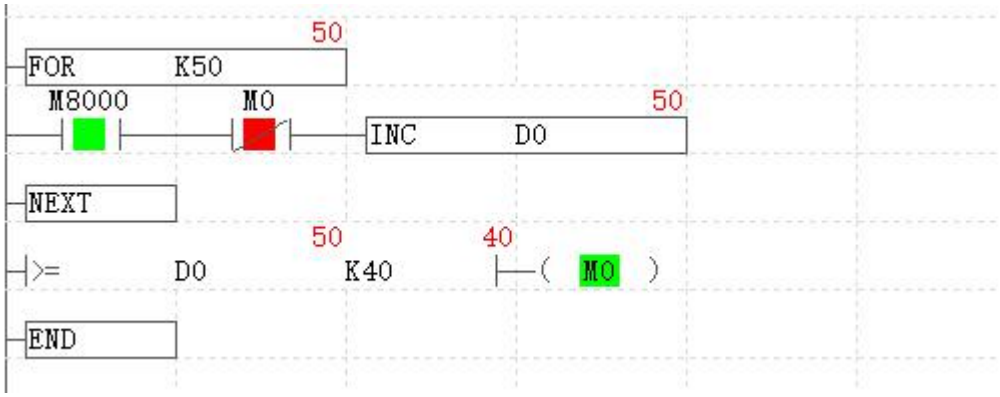
FOR 指令直接与母线相连。

操作数：

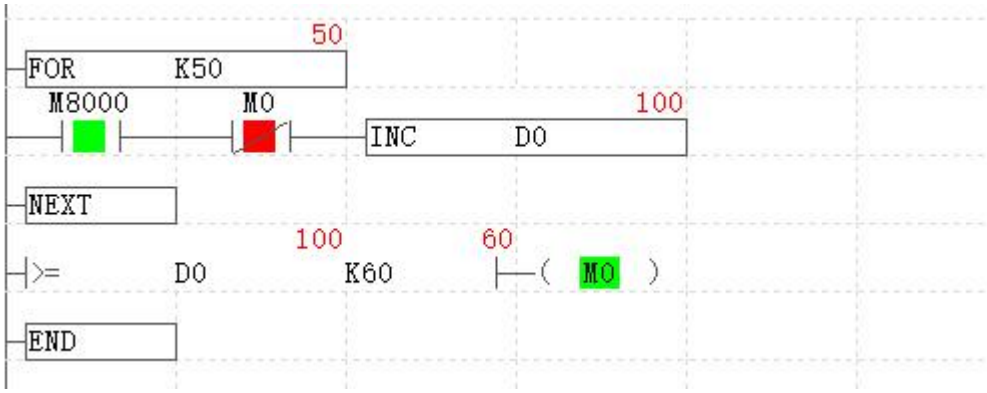
S：KnX, KnY, KnM, KnS,T,C,D,K,H,V,Z,LV,DT,@

编程示例：

FOR 到 NEXT 循环 50 次，M8000 上电后保持导通状态，D0 的值自加为 50，跳出循环，程序接着往下扫描，此时 D0 大于 40，LD>=比较指令导通，M0 线圈接通，常闭触点 M0 变为常开，D0 的值停止自加，所以 D0 的值为 50。



FOR 到 NEXT 循环 50 次，D0 的值为 50，跳出循环，D0 不大于 60，比较指令不导通，M0 线圈不接通。程序扫描到 END 之后从头再次执行 FOR 到 NEXT 再循环 50 次，D0 的值为 100，跳出循环，此时 D0 大于 60，比较指令导通，M0 线圈接通，常闭触点 M0 变为常开，D0 的值停止自加，所以 D0 的值为 100。



NEXT

指令说明：

从 FOR 指令开始到 NEXT 指令之间的程序按指定次数重复运行。

NEXT 指令直接与母线相连。

操作数：

无。

编程示例：

参见 FOR 指令。

FEND

指令说明：

FEND 指令表示主程序结束的指令。

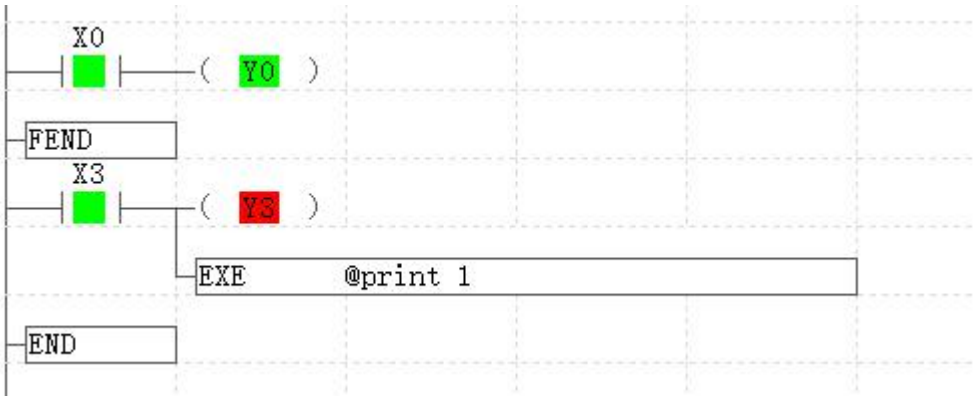
执行 FEND 指令后，会执行与 END 指令相同的输出处理、输入处理、看门狗定时器的刷新，然后返回到 0 步的程序，FEND 指令之后的程序需要使用指令调用。在编写子程序和中断程序时需要使用这个指令。

操作数：

无。

编程示例：

FEND 之后的指令没有指令调用的情况下不扫描，故 Y3 无法输出，且打印指令也不执行。



或参见 EI 指令。

END

指令说明：

END 指令表示程序结束的指令，在程序最后必须写入此指令，PLC 扫描执行到 END 指令之后，再从头扫描执行，对 END 指令之后的程序不做处理。

操作数：

无。

WDT

指令说明：

WDT 指令可以对看门狗定时器进行刷新。

通过设置 D8000 可以设置看门狗定时器的时间。

操作数：

无。

3.5 运算指令

32 位指令的运算使用[S1+1,S1]和 [S2+1,S2]的连续两个存储空间的内容进行二进制运算后传送到目标操作数[D+1,D]中。

分类	指令	位数	脉冲	功能	操作数类型
加 法 指令	ADD	16 位	-	[ADD S1 S2 D] S1 和 S2 的数据进行二进制加法运算后传送到 D 中	S1/S2：KnX, KnY, KnM, KnS, T,C,D,K,H,Z,V,LV,DT,@ D：KnY, KnM, KnS,T,C,D,Z, V,LV,DT
	ADDP		√		
	DADD	32 位	-		
	DADDP		√		
减 法	SUB	16 位	-	[SUB S1 S2 D]	S1/S2：KnX, KnY, KnM, KnS,

指令	SUBP		√	S1 和 S2 的数据进行二进制减法运算后传送到 D 中	T,C,D,K,H,Z,V,LV,DT,@ D：KnY, KnM, KnS,T,C,D,Z, V,LV,DT
	DSUB	32 位	-		
	DSUBP		√		
乘法指令	MUL	16 位	-	[MUL S1 S2 D] S1 和 S2 的数据进行二进制乘法运算后传送到 D 中	S1/S2：KnX, KnY, KnM, KnS, T,C,D,K,H,Z,V,LV,DT,@ D：KnY, KnM, KnS,T,C,D,Z, LV,DT
	MULP		√		
	DMUL	32 位	-		
	DMULP		√		
除法指令	DIV	16 位	-	[DIV S1 S2 D] S1 除以 S2，商传送到 D 中，余数传到 D+1 中	S1/S2：KnX, KnY, KnM, KnS, T,C,D,K,H,Z,V,LV,DT,@ D：KnY, KnM, KnS,T,C,D,Z, V,LV,DT
	DIVP		√		
	DDIV	32 位	-		
	DDIVP		√		
自增 1	INC	16 位	-	[INC D] 将 D 的数据自加一	D：KnY, KnM, KnS,T,C,D,Z, V,LV,DT
	INCP		√		
	DINC	32 位	-		
	DINCP		√		
自减 1	DEC	16 位	-	[DEC D] 将 D 的数据自减一	D：KnY, KnM, KnS,T,C,D,Z, V,LV,DT
	DECP		√		
	DDEC	32 位	-		
	DDECP		√		
逻辑与	WAND	16 位	-	[WAND S1 S2 D] S1 和 S2 的数据按位逻辑与运算后传送到 D 中	S1/S2：KnX, KnY, KnM, KnS, T,C,D,K,H,Z,V,LV,DT,@ D：KnY, KnM, KnS,T,C,D,Z, V,LV,DT
	WANDDP		√		
	DAND	32 位	-		
	DANDP		√		
逻辑或	WOR	16 位	-	[WOR S1 S2 D] S1 和 S2 的数据按位逻辑或运算后传送到 D 中	S1/S2：KnX, KnY, KnM, KnS, T,C,D,K,H,Z,V,LV,DT,@ D：KnY, KnM, KnS,T,C,D,Z, V,LV,DT
	WORDP		√		
	DOR	32 位	-		
	DORP		√		
逻辑异或	WXOR	16 位	-	[ADD S1 S2 D] S1 和 S2 的数据按位逻辑异或运算后传送到 D 中	S1/S2：KnX, KnY, KnM, KnS, T,C,D,K,H,Z,V,LV,DT,@ D：KnY, KnM, KnS,T,C,D,Z, V,LV,DT
	WXORDP		√		
	DXOR	32 位	-		
	DXORP		√		
取补码	WNEG	16 位	-	[NEG D] D 的数据按位取反后再加一，运算的结果传送到 D 中	D：KnY, KnM, KnS,T,C,D,Z, V,LV,DT
	WNEGDP		√		
	DNEG	32 位	-		
	DNEGP		√		

ADD

指令说明：

ADD 是 16 位连续执行型加法运算指令。是将源操作数 S1 和源操作数 S2 的内容进行二进制加法运算后传送到目标操作数 D 中。

各数据位的最高位为正（0）、负（1）的符号位，这些数据以代数方式进行加法运算。

例：7+（-2）=5

S1 和 S2 中指定常数（K）时，会自动进行 BIN（二进制数）转换。

标志位的动作及数值的正负的关系，如下表所示：

软元件	名称	内容
M8020	置位	ON：运算结果为 0 时，M8020 置位 OFF：运算结果为 0 以外时，M8020 复位
M8021	借位	ON：运算结果小于-32768（16 位运算）或-2147483648（32 位运算）时，借位标记位动作 OFF：运算结果大于-32768（16 位运算）或-2147483648（32 位运算）时，借位标记位不动作
M8022	进位	ON：运算结果大于 32767（16 位运算）或 2147483648（32 位运算）时，进位标记位动作 OFF：运算结果小于 32767（16 位运算）或 2147483648（32 位运算）时，进位标记位不动作

操作数：

S1：KnX, KnY, KnM, KnS,T,C,D,K,H,Z,V,LV,DT,@

S2：KnX, KnY, KnM, KnS,T,C,D,K,H,Z,V,LV,DT,@

D：KnY, KnM, KnS, T,C,D,Z,V, LV,DT

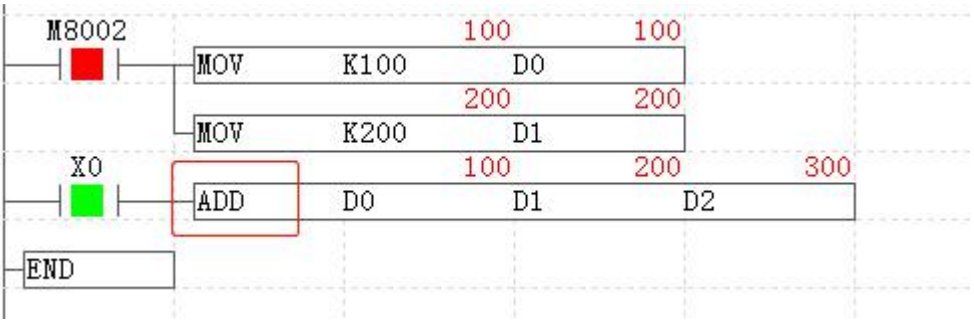
指令格式：[ADD　S1　S2　D]

编程示例：

条件满足后，给 D0、D1 寄存器赋值，D0=100，D1=200

当 X0 接通时候，执行加法运算指令，D0+D1=D2，结果 D2=300，D0、D1 里面的数据保持不变。

执行 16 位储存空间的加法指令，D0、D1、D2 里面的数据最大不能超过 32767。



ADDP

指令说明：

ADDP 是 16 位脉冲执行型加法运算指令，指令激活一次，执行一次加法运算。是将源操作数 S1 和源操作数 S2 的内容进行二进制加法运算后传送到目标操作数 D 中。

各数据位的最高位为正（0）、负（1）的符号位，这些数据以代数方式进行加法运算。

例：7+（-2）=5

S1 和 S2 中指定常数（K）时，会自动进行 BIN（二进制数）转换。

标志位的动作及数值的正负的关系，如下表所示：

软元件	名称	内容
M8020	置位	ON：运算结果为 0 时，M8020 置位 OFF：运算结果为 0 以外时，M8020 复位
M8021	借位	ON：运算结果小于-32768（16 位运算）或-2147483648（32 位运算）时，借位标记位动作 OFF：运算结果大于-32768（16 位运算）或-2147483648（32 位运算）时，借位标记位不动作
M8022	进位	ON：运算结果大于 32767（16 位运算）或 2147483648（32 位运算）时，进位标记位动作 OFF：运算结果小于 32767（16 位运算）或 2147483648（32 位运算）时，进位标记位不动作

操作数：

S1：KnX, KnY, KnM, KnS,T,C,D,K,H,Z,V,LV,DT,@

S2：KnX, KnY, KnM, KnS,T,C,D,K,H,Z,V,LV,DT,@

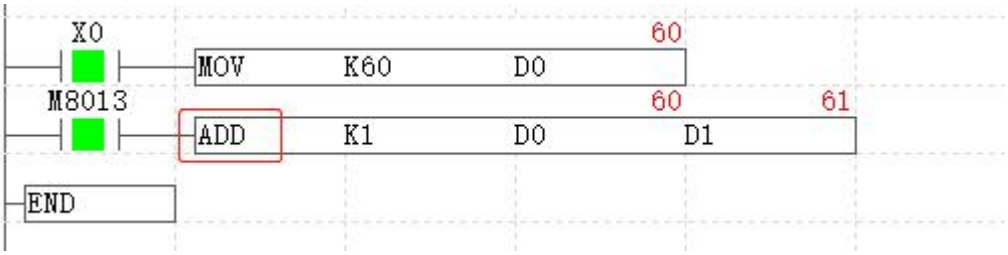
D：KnY, KnM, KnS, T,C,D,Z,V, LV,DT

指令格式：[ADDP　S1　S2　D]

编程示例：

X0 闭合时，将 K60 存入 D0，M8013 以 1S 的时间为周期，来发出脉冲，使 K1 以 1S 为周期和 D0 相加传送到 D1。

执行 16 位储存空间的加法指令，D0、D1、D2 里面的数据最大不能超过 32767。



DADD

指令说明：

DADD 是 32 位连续执行型加法运算指令。是将源操作数[S1+1,S1]和源操作数[S2+1,S2]的内容进行二进制加法运算后传送到目标操作数[D+1,D]中。

各数据位的最高位为正（0）、负（1）的符号位，这些数据以代数方式进行加法运算。

例： 852000+（-1100000）= -248000

[S1+1,S1]、[S2+1,S2]和中指定常数（K）时，会自动进行 BIN（二进制数）转换。

标志位的动作及数值的正负的关系，如下表所示：

软元件	名称	内容
M8020	置位	ON：运算结果为 0 时，M8020 置位 OFF：运算结果为 0 以外时，M8020 复位
M8021	借位	ON：运算结果小于-32768（16 位运算）或-2147483648（32 位运算）时，借位标记位动作 OFF：运算结果大于-32768（16 位运算）或-2147483648（32 位运算）时，借位标记位不动作
M8022	进位	ON：运算结果大于 32767（16 位运算）或 2147483648（32 位运算）时，进位标记位动作 OFF：运算结果小于 32767（16 位运算）或 2147483648（32 位运算）时，进位标记位不动作

操作数：

S1：KnX, KnY, KnM, KnS,T,C,D,K,H,Z,V,LV,DT,@

S2：KnX, KnY, KnM, KnS,T,C,D,K,H,Z,V,LV,DT,@

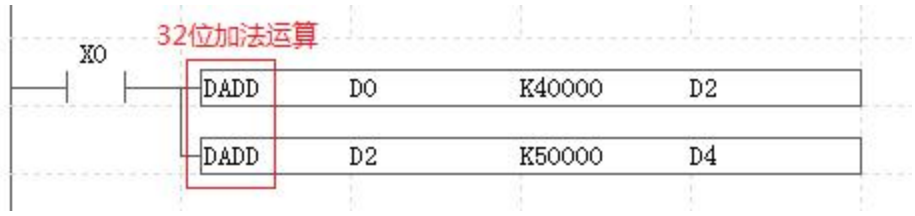
D：KnY, KnM, KnS, T,C,D,Z,V, LV,DT

指令格式：[DADD S1 S2 D]

编程示例：

假设 D1D0 的值为 0，X0 接通先执行 D1D0+40000，D2 的数据为 40000，再执行 D3D2+50000，D5D4 里面的数据为 90000。

32 位储存空间的数据范围我们一般不考虑，平时也用不到这么大的数据。



DADDP

指令说明：

DADDP 是 32 位脉冲执行型加法运算指令，指令激活一次，执行一次加法运算。是将源操作数[S1+1,S1]和源操作数[S2+1,S2]的内容进行二进制加法运算后传送到目标操作数[D+1,D]中。

各数据位的最高位为正（0）、负（1）的符号位，这些数据以代数方式进行加法运算。

例：852000+（-1100000）= -248000

[S1+1,S1]、[S2+1,S2]和中指定常数（K）时，会自动进行 BIN（二进制数）转换。

标志位的动作及数值的正负的关系，如下表所示：

软元件	名称	内容
M8020	置位	ON：运算结果为 0 时，M8020 置位 OFF：运算结果为 0 以外时，M8020 复位

M8021	借位	ON：运算结果小于-32768（16 位运算）或-2147483648（32 位运算）时，借位标记位动作 OFF：运算结果大于-32768（16 位运算）或-2147483648（32 位运算）时，借位标记位不动作
M8022	进位	ON：运算结果大于 32767（16 位运算）或 2147483648（32 位运算）时，进位标记位动作 OFF：运算结果小于 32767（16 位运算）或 2147483648（32 位运算）时，进位标记位不动作

操作数：

S1：KnX, KnY, KnM, KnS,T,C,D,K,H,Z,V,LV,DT,@

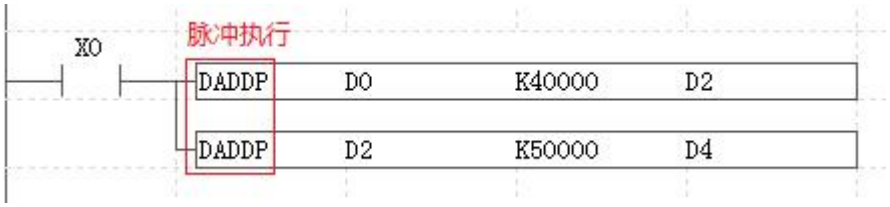
S2：KnX, KnY, KnM, KnS,T,C,D,K,H,Z,V,LV,DT,@

D：KnY, KnM, KnS, T,C,D,Z,V, LV,DT

指令格式：[DADDP　S1　S2　D]

编程示例：

当 X0 闭合时产生一个上升沿脉冲，执行一次加法运算。D1D0+40000=D3D2，D3D2+50000=D5D4，如果 X0 的状态没变化，DADDP 指令也不会再执行。



SUB

指令说明：

SUB 是 16 位连续执行型减法运算指令。是将源操作数 S1 和源操作数 S2 的内容进行二进制减法运算后传送到目标操作数 D 中。

各数据位的最高位为正（0）、负（1）的符号位，这些数据以代数方式进行减法运算。

例：115－15=100

S1 和 S2 中指定常数（K）时，会自动进行 BIN（二进制数）转换。

标志位的动作及数值的正负的关系，如下表所示：

软元件	名称	内容
M8020	置位	ON：运算结果为 0 时，M8020 置位 OFF：运算结果为 0 以外时，M8020 复位
M8021	借位	ON：运算结果小于-32768（16 位运算）或-2147483648（32 位运算）时，借位标记位动作 OFF：运算结果大于-32768（16 位运算）或-2147483648（32 位运算）时，借位标记位不动作
M8022	进位	ON：运算结果大于 32767（16 位运算）或 2147483648（32 位运算）时，进位标记位动作 OFF：运算结果小于 32767（16 位运算）或 2147483648（32 位运算）时，进位标记位不动作

操作数：

S1：KnX, KnY, KnM, KnS,T,C,D,K,H,Z,V,LV,DT,@

S2：KnX, KnY, KnM, KnS,T,C,D,K,H,Z,V,LV,DT,@

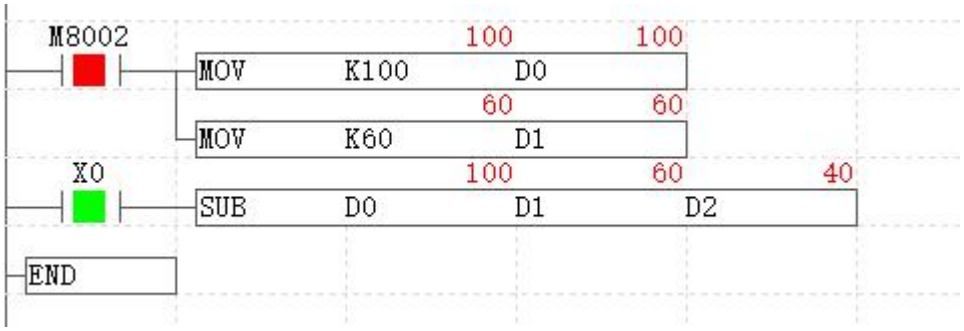
D：KnY, KnM, KnS, T,C,D,Z,V, LV,DT

指令格式：[SUB　S1　S2　D]

编程示例：

X0 接通，SUB 指令执行，D0－D1=D2，也就是 100－60=40。D2 的值为 40。

执行 16 位储存空间的减法指令，D0、D1、D2 里面的数据最大不能超过 32767。



SUBP

指令说明：

SUBP 是 16 位脉冲执行型减法运算指令，即指令激活一次，执行一次减法运算。是将源操作数 S1 和源操作数 S2 的内容进行二进制减法运算后传送到目标操作数 D 中。

各数据位的最高位为正（0）、负（1）的符号位，这些数据以代数方式进行减法运算。

例：115－15=100

S1 和 S2 中指定常数（K）时，会自动进行 BIN（二进制数）转换。

标志位的动作及数值的正负的关系，如下表所示：

软元件	名称	内容
M8020	置位	ON：运算结果为 0 时，M8020 置位 OFF：运算结果为 0 以外时，M8020 复位
M8021	借位	ON：运算结果小于-32768（16 位运算）或-2147483648（32 位运算）时，借位标记位动作 OFF：运算结果大于-32768（16 位运算）或-2147483648（32 位运算）时，借位标记位不动作
M8022	进位	ON：运算结果大于 32767（16 位运算）或 2147483648（32 位运算）时，进位标记位动作 OFF：运算结果小于 32767（16 位运算）或 2147483648（32 位运算）时，进位标记位不动作

操作数：

S1：KnX, KnY, KnM, KnS,T,C,D,K,H,Z,V,LV,DT,@

S2：KnX, KnY, KnM, KnS,T,C,D,K,H,Z,V,LV,DT,@

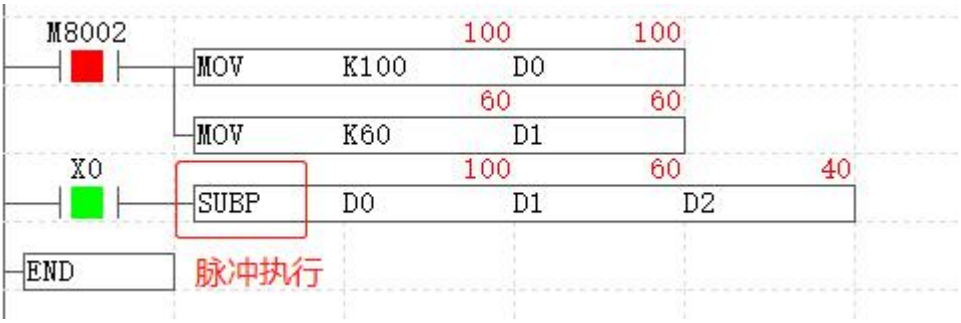
D：KnY, KnM, KnS, T,C,D,Z,V, LV,DT

指令格式：[SUBP　S1　S2　D]

编程示例：

当 X0 闭合时，产生一个上升沿脉冲，使得 SUBP 指令执行一次，D0-D1=D2，D2=50。

执行 16 位储存空间的减法指令，D0、D1、D2 里面的数据最大不能超过 32767。



DSUB

指令说明：

DSUB 是 32 位连续执行型减法运算指令。是将源操作数[S1+1,S1]和源操作数[S2+1,S2]的内容进行二进制减法运算后传送到目标操作数[D+1,D]中。

各数据位的最高位为正（0）、负（1）的符号位，这些数据以代数方式进行减法运算。

例：45000－(-15000)=60000

[S1+1,S1]和[S2+1,S2]中指定常数（K）时，会自动进行 BIN（二进制数）转换。

标志位的动作及数值的正负的关系，如下表所示：

软元件	名称	内容
M8020	置位	ON：运算结果为 0 时，M8020 置位 OFF：运算结果为 0 以外时，M8020 复位
M8021	借位	ON：运算结果小于-32768（16 位运算）或-2147483648（32 位运算）时，借位标记位动作 OFF：运算结果大于-32768（16 位运算）或-2147483648（32 位运算）时，借位标记位不动作
M8022	进位	ON：运算结果大于 32767（16 位运算）或 2147483648（32 位运算）时，进位标记位动作 OFF：运算结果小于 32767（16 位运算）或 2147483648（32 位运算）时，进位标记位不动作

操作数：

S1：KnX, KnY, KnM, KnS,T,C,D,K,H,Z,V,LV,DT,@

S2：KnX, KnY, KnM, KnS,T,C,D,K,H,Z,V,LV,DT,@

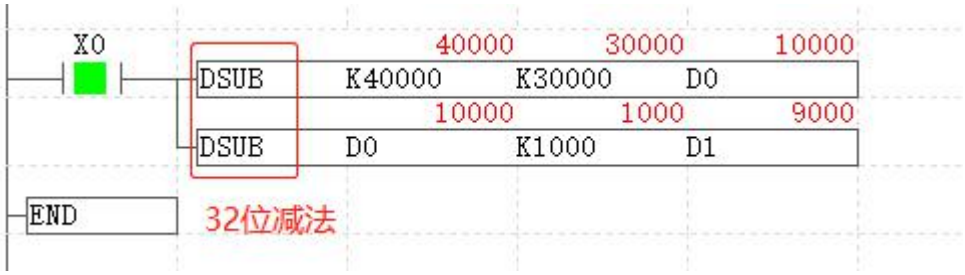
D：KnY, KnM, KnS, T,C,D,Z,V, LV,DT

指令格式：[DSUB　S1　S2　D]

编程示例：

当 X0 闭合时，首先执行 40000－30000=10000，D0 的数据为 10000，再执行 D0-1000，D1 的值为 9000。

使用连续执行型的指令 DSUB 后，每个运算周期减法运算的结果都会变化。



DSUBP

指令说明：

DSUBP 是 32 位脉冲执行型减法运算指令，即指令激活一次，执行一次减法运算。是将源操作数[S1+1,S1]和源操作数[S2+1,S2]的内容进行二进制减法运算后传送到目标操作数[D+1,D]中。

各数据位的最高位为正（0）、负（1）的符号位，这些数据以代数方式进行减法运算。

例：45000－(-15000)=60000

[S1+1,S1]和[S2+1,S2]中指定常数（K）时，会自动进行 BIN（二进制数）转换。

标志位的动作及数值的正负的关系，如下表所示：

软元件	名称	内容
M8020	置位	ON：运算结果为 0 时，M8020 置位 OFF：运算结果为 0 以外时，M8020 复位
M8021	借位	ON：运算结果小于-32768（16 位运算）或-2147483648（32 位运算）时，借位标记位动作 OFF：运算结果大于-32768（16 位运算）或-2147483648（32 位运算）时，借位标记位不动作
M8022	进位	ON：运算结果大于 32767（16 位运算）或 2147483648（32 位运算）时，进位标记位动作 OFF：运算结果小于 32767（16 位运算）或 2147483648（32 位运算）时，进位标记位不动作

操作数：

S1：KnX, KnY, KnM, KnS,T,C,D,K,H,Z,V,LV,DT,@

S2：KnX, KnY, KnM, KnS,T,C,D,K,H,Z,V,LV,DT,@

D：KnY, KnM, KnS, T,C,D,Z,V,LV,DT

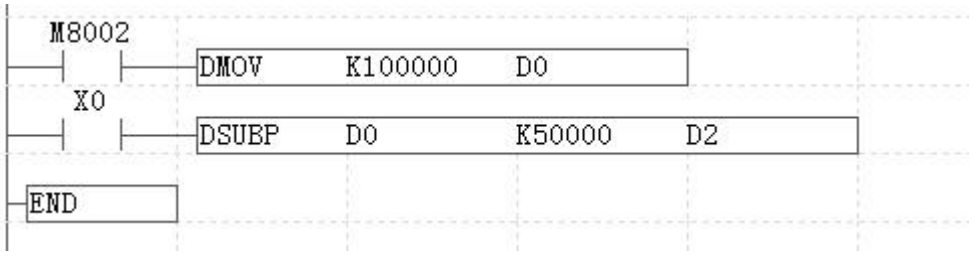
指令格式：[DSUBP　S1　S2　D]

编程示例：

X0 接通，执行一次减法指令，D3D2 的值为 50000。

D1D0 组合的 32 位空间地址为 MODBUS_LONG(0)，D3D2 组合的 32 位空间地址为 MODBUS_LONG(2)。

监视内容	值
modbus_long(0)	100000
modbus_long(2)	50000



MUL

指令说明：

MUL 是 16 位连续执行型乘法运算指令。是将源操作数 S1 和源操作数 S2 的内容进行二进制乘法运算后传送到目标操作数[D+1,D]的 32 位（双字）中。

各数据位的最高位为正（0）、负（1）的符号位，这些数据以代数方式进行乘法运算。

例：500×8=4000

S1 和 S2 中指定常数（K）时，会自动进行 BIN（二进制数）转换。

操作数：

S1：KnX, KnY, KnM, KnS,T,C,D,K,H,V,Z,LV,DT,@

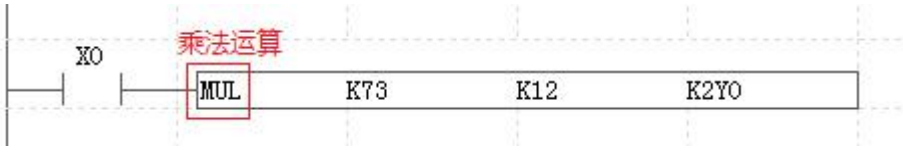
S2：KnX, KnY, KnM, KnS,T,C,D,K,H,V,Z,LV,DT,@

D：KnY, KnM, KnS, T,C,D,Z,V,LV,DT

指令格式：[MUL S1 S2 D]

编程示例：

位软元件指定 K2 时，只能得到乘积（32 位）中的低 8 位



数据的运算如下所示：

K73（H0049）× K12（H000C）= K876（H036C）

K876（H036C）的 32 个数据位（二进制）如下所示：

b32	b31	b30	…	b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
0	0	0	…	0	0	0	0	0	0	1	1	0	1	1	0	1	1	0	0

不输出　　向 K2Y0(Y7~Y0)输出运算结果

MULP

指令说明：

MULP 是 16 位脉冲执行型乘法运算指令，即指令激活一次，执行一次乘法运算。是将源操作数 S1 和源操作数 S2 的内容进行二进制乘法运算后传送到目标操作数[D+1,D]的 32 位（双字）中。

各数据位的最高位为正（0）、负（1）的符号位，这些数据以代数方式进行乘法运算。

例：500×8=4000

S1 和 S2 中指定常数（K）时，会自动进行 BIN（二进制数）转换。

操作数：

S1：KnX, KnY, KnM, KnS,T,C,D,K,H,V,Z,LV,DT,@

S2：KnX, KnY, KnM, KnS,T,C,D,K,H,V,Z,LV,DT,@

D：KnY, KnM, KnS, T,C,D,Z,V,LV,DT

指令格式：[MULP　S1　S2　D]

编程示例：

当 X0 闭合时，MULP 执行一次，D0×D1=D2，D2=20000。

D0　×　D1　=　D3D2

16 位　　16 位　　32 位

DMUL

指令说明：

DMUL 是 32 位连续执行型乘法运算指令。是将源操作数[S1+1,S1]和源操作数[S2+1,S2]的内容进行二进制乘法运算后传送到目标操作数[D+3,D+2,D+1,D]的 64 位（四字）中。

各数据位的最高位为正（0）、负（1）的符号位，这些数据以代数方式进行乘法运算。

例：5500×（-8540）= -46970000

[S1+1,S1]和[S2+1,S2]中指定常数（K）时，会自动进行 BIN（二进制数）转换。

[D+3,D+2,D+1,D]中指定位数为（K1~8）时，只能获得低 32 位的结果。

操作数：

S1：KnX, KnY, KnM, KnS,T,C,D,K,H,V,Z,LV,DT,@

S2：KnX, KnY, KnM, KnS,T,C,D,K,H,V,Z,LV,DT,@

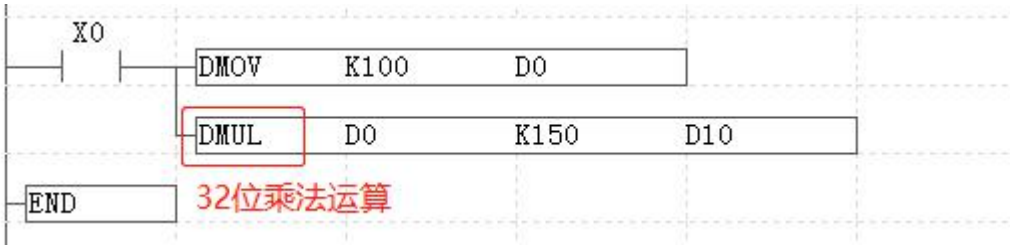
D：KnY, KnM, KnS, T,C,D,Z,V,LV,DT

指令格式：[DMUL　S1　S2　D]

编程示例：

条件满足后执行 DMUL 指令，D1D0 的 100 乘以 150，得到 15000，传送给 D13D12D11D10。

(D1,D0)×K150=(D13,D12,D11,D10)



DMULP

指令说明：

DMULP 是 32 位脉冲执行型乘法运算指令，即指令激活一次，执行一次乘法运算。是将源操作数[S1+1,S1]和源操作数[S2+1,S2]的内容进行二进制乘法运算后传送到目标操作数[D+3,D+2,D+1,D]的 64 位（四字）中。

各数据位的最高位为正（0）、负（1）的符号位，这些数据以代数方式进行乘法运算。

例：5500×（-8540）= -46970000

[S1+1,S1]和[S2+1,S2]中指定常数（K）时，会自动进行 BIN（二进制数）转换。

[D+3,D+2,D+1,D]中指定位数为（K1~8）时，只能获得低 32 位的结果。

操作数：

S1：KnX, KnY, KnM, KnS,T,C,D,K,H,V,Z,LV,DT,@

S2：KnX, KnY, KnM, KnS,T,C,D,K,H,V,Z,LV,DT,@

D：KnY, KnM, KnS, T,C,D,Z,V,LV,DT

指令格式：[DMULP S1 S2 D]

编程示例：

前面条件满足后，100×200= D7D6D5D4，D7D6D5D4=20000

D1D0 × D3D2 = D7D6D5D4

32 位 32 位 64 位

注意：一定要注意储存空间的结构，避免存储地址重叠，而得不到正确的数据。

DIV

指令说明：

DIV 是 16 位连续执行型除法运算指令。是将源操作数 S1 的内容作为被除数，源操作数 S2 的内容作为除数，S1 除以 S2 之后，商传送到 D 中，余数传送到[D+1]中。

各数据位的最高位为正（0）、负（1）的符号位，这些数据以代数方式进行除法运算。

例：36÷5=7（商），1（余数）

运算结果（商、余数），会占用指定 D 开始合计 2 点的软元件，所以请注意不能与其它控制重复。

S1 和 S2 中指定常数（K）时，会自动进行 BIN（二进制数）转换。

操作数：

S1：KnX, KnY, KnM, KnS,T,C,D,K,H,V,Z,LV,DT,@

S2：KnX, KnY, KnM, KnS,T,C,D,K,H,V,Z,LV,DT,@

D：KnY, KnM, KnS, T,C,D,Z,V,LV,DT

指令格式：[DIV　S1　S2　D]

注意：除数 S2 为 0 时，会发生运算出错，其结果为-1，但可以执行命令。运算结果超过 32767（16 位运算）时，出现运算出错。

编程示例：

假设 D0=100，D2=33，当 X0 闭合，执行 DIV 除法指令，将 D0 的 100 除以 D2 的 33，得到的商 3 传送到 D4，得到的余数 1 传送到 D5。

DIVP

指令说明：

DIVP 是 16 位脉冲执行型除法运算指令，即指令激活一次，执行一次除法运算。是将源操作数 S1 的内容作为被除数，源操作数 S2 的内容作为除数，S1 除以 S2 之后，商传送到 D 中，余数传送到[D+1]中。

各数据位的最高位为正（0）、负（1）的符号位，这些数据以代数方式进行除法运算。

例：36÷5=7（商），1（余数）

运算结果（商、余数），会占用指定 D 开始合计 2 点的软元件，所以请注意不能与其它控制重复。

S1 和 S2 中指定常数（K）时，会自动进行 BIN（二进制数）转换。

操作数：

S1：KnX, KnY, KnM, KnS,T,C,D,K,H,V,Z,LV,DT,@

S2：KnX, KnY, KnM, KnS,T,C,D,K,H,V,Z,LV,DT,@

D：KnY, KnM, KnS, T,C,D,Z,V,LV,DT

指令格式：[DIVP　S1　S2　D]

注意：除数 S2 为 0 时，会发生运算出错，其结果为-1，但可以执行命令。运算结果超过 32767（16 位运算）时，出现运算出错。

编程示例：

当 X0 闭合，除法指令执行一次，使 D0 的数据除以 D2，商传到 D4，余数传到 D5。

DDIV

指令说明：

DDIV 是 32 位连续执行型除法运算指令。是将源操作数[S1+1,S1]的内容作为被除数，源操作数[S2+1,S2]的内容作为除数，[S1+1,S1]除以[S2+1,S2]之后，商传送到[D+1,D]中，余数传送到[D+3,D+2]中。

各数据位的最高位为正（0）、负（1）的符号位，这些数据以代数方式进行除法运算。

例：3200÷310=10（商），100（余数）

运算结果（商、余数），会占用指定 D 开始合计 4 点的软元件，所以请注意不能与其它控制重复。

[S1+1,S1]和[S2+1,S2]中指定常数（K）时，会自动进行 BIN（二进制数）转换。

操作数：

S1：KnX, KnY, KnM, KnS,T,C,D,K,H,V,Z,LV,DT,@

S2：KnX, KnY, KnM, KnS,T,C,D,K,H,V,Z,LV,DT,@

D：KnY, KnM, KnS, T,C,D,Z,V,LV,DT

指令格式：[DDIV　S1　S2　D]

注意：除数 S2 为 0 时，会发生运算出错，其结果为-1，但可以执行命令。运算结果超过 2147483647（32 位运算）时，出现运算出错。

编程示例：

假设 D1D0 的存储数据为 100000，D3D2 为 3333，当 X0 闭合后，DDIV 指令将 D1D0 的数据除以 D3D2 的数据，得到的商传至D5D4，余数传至 D7D6。

DDIVP

指令说明：

DDIVP 是 32 位脉冲执行型除法运算指令，即指令激活一次，执行一次除法运算。是将源操作数[S1+1,S1]的内容作为被除数，源操作数[S2+1,S2]的内容作为除数，[S1+1,S1]除以[S2+1,S2]之后，商传送到[D+1,D]中，余数传送到[D+3,D+2]中。

各数据位的最高位为正（0）、负（1）的符号位，这些数据以代数方式进行除法运算。

例：3200÷310=10（商），100（余数）

运算结果（商、余数），会占用指定 D 开始合计 4 点的软元件，所以请注意不能与其它控制重复。

[S1+1,S1]和[S2+1,S2]中指定常数（K）时，会自动进行 BIN（二进制数）转换。

操作数：

S1：KnX, KnY, KnM, KnS,T,C,D,K,H,V,Z,LV,DT,@

S2：KnX, KnY, KnM, KnS,T,C,D,K,H,V,Z,LV,DT,@

D：KnY, KnM, KnS, T,C,D,Z,V,LV,DT

指令格式：[DDIVP　S1　S2　D]

注意：除数 S2 为 0 时，会发生运算出错，其结果为-1，但可以执行命令。运算结果超过 2147483647（32 位运算）时，出现运算出错。

编程示例：

当 X0 闭合时，DDIVP 执行一次，将 D1D0 的 32 位数据除以 D3D2 的 32 位数据，得到的商传送到 D5D4，余数传送到 D7D6。

INC

指令说明：

INC 是 16 位连续执行型加一指令。指令每个扫描周期执行一次，目标操作数[D]里面的数据加一。

操作数：

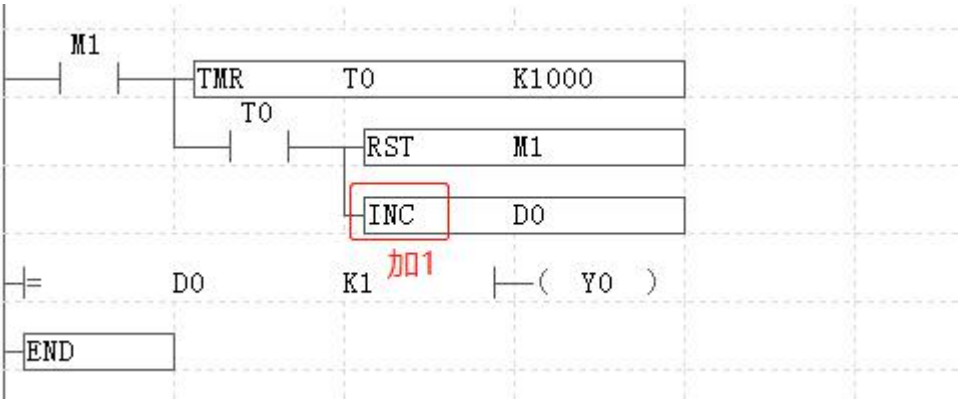
D：KnY, KnM, KnS,T,C,D,V,Z,LV,DT

指令格式：[INC　D]

注意：连续执行型的加 1 运算一定要以脉冲指令的形式来执行。

编程示例：

假设 D0 的初始值为 0，M1 触点导通后，定时器开始计时，计时完成 T0 触点导通，断开 M1 触点的同时让 D0 自增 1，此时 D0 的数据为 1，使得下面的触点比较指令导通，Y0 输出。



INCP

指令说明：

INCP 是 16 位脉冲执行型加一指令，即指令激活一次，执行一次加 1 运算。目标操作数[D]里面的数据加一。

操作数：

D：KnY, KnM, KnS, T,C,D,V,Z,LV,DT

指令格式：[INCP D]

编程示例：

当 X0 从 OFF 变为 ON 时，产生的脉冲上升沿使得 INCP 指令执行一次，D0 自增 1，若 X0 不产生上升沿脉冲，那么 D0 的数据也不会改变。

DINC

指令说明：

DINC 是 32 位连续执行型加一指令。指令每执行一个扫描周期，目标操作数[D+1,D]里面的数据加一。

操作数：

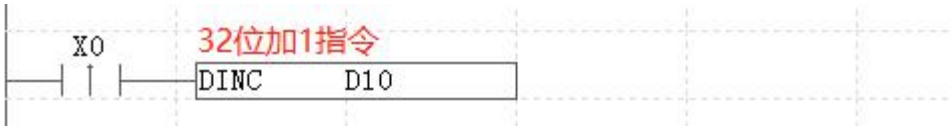
D：KnY, KnM, KnS, T,C,D,V,Z,LV,DT

指令格式：[DINC D]

注意：连续执行型的加 1 运算一定要以脉冲指令的形式来执行。

编程示例：

当 X0 导通时，DINC 指令将 D11D10 的数据内容加一运算后，传送到 D11D10。



DINCP

指令说明：

DINCP 是 32 位脉冲执行型加一指令，即指令激活一次，执行一次加 1 运算。目标操作数[D+1,D]里面的数据加一。

操作数：

D：KnY, KnM, KnS, T,C,D,V,Z,LV,DT

指令格式：[DINCP　D]

编程示例：

当 X0 闭合产生一个上升沿脉冲，DINCP 指令执行一次，D11D10 的数据加 1。

DEC

指令说明：

DEC 是 16 位连续执行型减一指令。指令每执行一个扫描周期，目标操作数[D]里面的数据减一。

操作数：

D：KnY, KnM, KnS, T,C,D,V,Z,LV,DT

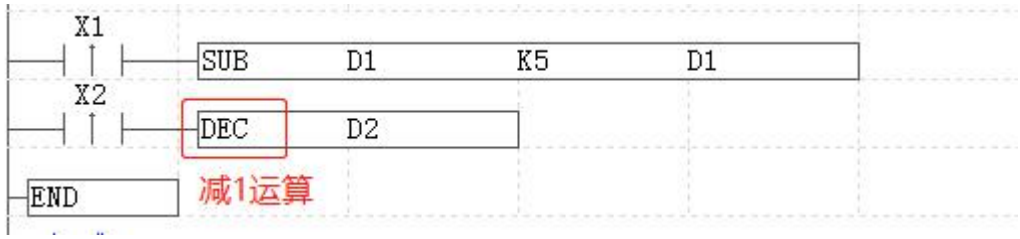
指令格式：[DEC　D]

注意：连续执行型的减 1 运算一定要以脉冲指令的形式来执行。

编程示例：

X1 接通一次，执行一次减法运算，每执行一次后 D1 的数据减 5 再传送到 D1。

X2 接通一次，执行一次自减 1 运算，每执行一次后 D2 的数据自减 1 再传送到 D2



DECP

指令说明：

DECP 是 16 位脉冲执行型减一指令，即指令激活一次，执行一次减 1 运算。目标操作数[D]里面的数据减一。

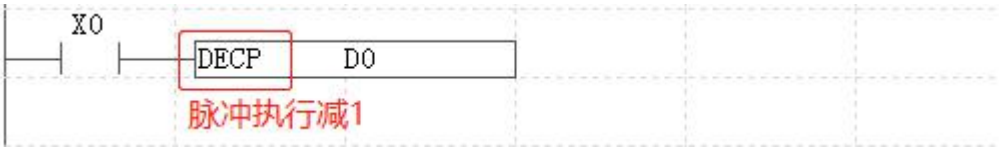
操作数：

D：KnY, KnM, KnS, T,C,D,V,Z,LV,DT

指令格式：[DECP　D]

编程示例：

当 X0 闭合后，产生一个上升沿脉冲，DECP 执行一次，D0 的数据减 1。



DDEC

指令说明：

DDEC 是 32 位连续执行型减一指令。指令每执行一个扫描周期，目标操作数[D+1,D]里面的数据减一。

操作数：

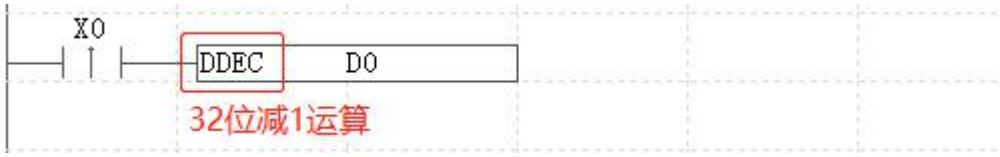
D：KnY, KnM, KnS, T,C,D,V,Z,LV,DT

指令格式：[DDEC　D]

注意：连续执行型的减 1 运算一定要以脉冲指令的形式来执行。

编程示例：

触发 X0 上升沿脉冲指令，DDEC 指令执行一次减 1 运算，D1D0 的数据减 1。



DDECP

指令说明：

DECP 是 32 位脉冲执行型减一指令，即指令激活一次，执行一次减 1 运算。目标操作数[D+1,D]里面的数据减一。

操作数：

D：KnY, KnM, KnS, T,C,D,V,Z,LV,DT

指令格式：[DDECP　D]

编程示例：

当 X0 闭合时，产生的上升沿脉冲使 DDECP 指令执行一次，D1D0 的数据减 1。

WAND

指令说明：

WAND 是 16 位连续执行型逻辑与运算指令，是将源操作数 S1 与源操作数 S2 的数据内容以位为单位进行逻辑与运算后，传送到 D 中。

S1 与 S2 中指定常数（K）时，会自动地进行 BIN 转换。

逻辑与运算以位为单位，如下表中（1&1=1、0&1=0、1&0=0、0&0=0）所示变化。

表中：1=ON，0=OFF

WAND	S1	S2	D
位单位的逻辑与运算	1	1	1
	0	1	0
	1	0	0
	0	0	0

操作数：

S1：KnX, KnY, KnM, KnS,T,C,D,K,H,Z,V,LV,DT,@

S2：KnX, KnY, KnM, KnS,T,C,D,K,H,Z,V,LV,DT,@

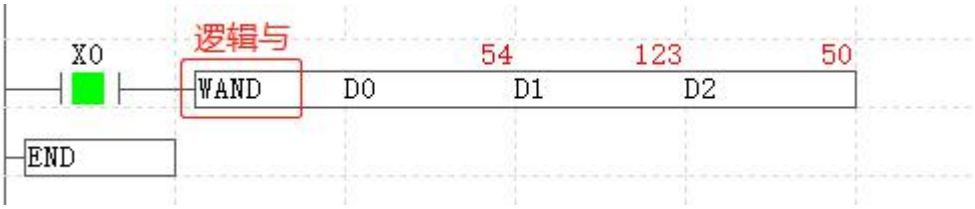
D：KnY, KnM, KnS, T,C,D,Z,V, LV,DT

指令格式：[WAND　S1　S2　D]

编程示例：

假设 D0 为 54，D1 为 123，X0 闭合，执行 WAND 指令，将 D0 的 16 位数据对应 D1 的 16 位数据进行逻辑与运算后，得到 D2 为 50。

b0^b0，b1^b1，b2^b2，…，b15^b15



D0、D1、D2 的 16 位数据位如下表所示：

软元件	b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0	值
D0	0	0	0	0	0	0	0	0	0	0	1	1	0	1	1	0	54
D1	0	0	0	0	0	0	0	0	0	1	1	1	1	0	1	1	123
D2	0	0	0	0	0	0	0	0	0	0	1	1	0	0	1	0	50

WANDP

指令说明：

WANDP 是 16 位脉冲执行型逻辑与运算指令，即指令激活一次，执行一次逻辑与运算。是将源操作数 S1 与源操作数 S2 的数据内容以位为单位进行逻辑与运算后，传送到 D 中。

S1 与 S2 中指定常数（K）时，会自动地进行 BIN 转换。

逻辑与运算以位为单位，如下表中（1&1=1、0&1=0、1&0=0、0&0=0）所示变化。

表中：1=ON，0=OFF

WANDP	S1	S2	D
位单位的逻辑与运算	1	1	1
	0	1	0
	1	0	0
	0	0	0

操作数：

S1：KnX, KnY, KnM, KnS,T,C,D,K,H,Z,V,LV,DT,@

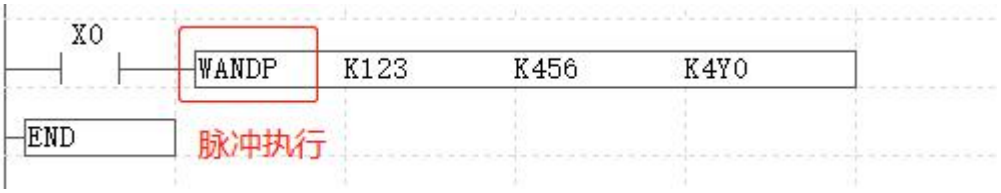
S2：KnX, KnY, KnM, KnS,T,C,D,K,H,Z,V,LV,DT,@

D：KnY, KnM, KnS, T,C,D,Z,V, LV,DT

指令格式：[WANDP　S1　S2　D]

编程示例：

X0 闭合时，WANDP 执行一次，将 K123，K456BIN 转换后，两者的 16 位数据进行逻辑与运算后传送到 K4Y0，K4Y0 的数据为 72。Y3 与 Y6 为 ON。



K123，K456，K4Y0 的 16 位数据位如下表所示：

软元件	b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
K123	0	0	0	0	0	0	0	0	0	1	1	1	1	0	1	1
K456	0	0	0	0	0	0	0	1	1	1	0	0	1	0	0	0
K4Y0	0	0	0	0	0	0	0	0	0	1	0	0	1	0	0	0
(K72)	Y17	Y16	Y15	Y14	Y13	Y12	Y11	Y10	Y7	Y6	Y5	Y4	Y3	Y2	Y1	Y0

DAND

指令说明：

DAND 是 32 位连续执行型逻辑与运算指令，是将源操作数[S1+1,S1]与源操作数[S2+1,S2]的数据内容以位为单位进行逻辑与运算后，传送到[D+1,D]中。

[S1+1,S1]与[S2+1,S2]中指定常数（K）时，会自动地进行 BIN 转换。

逻辑与运算以位为单位，如下表中（1&1=1、0&1=0、1&0=0、0&0=0）所示变化。

表中：1=ON，0=OFF

DAND	[S1+1,S1]	[S2+1,S2]	[D+1,D]
位单位的逻辑与运算	1	1	1
	0	1	0
	1	0	0
	0	0	0

操作数：

S1：KnX, KnY, KnM, KnS,T,C,D,K,H,Z,V,LV,DT,@

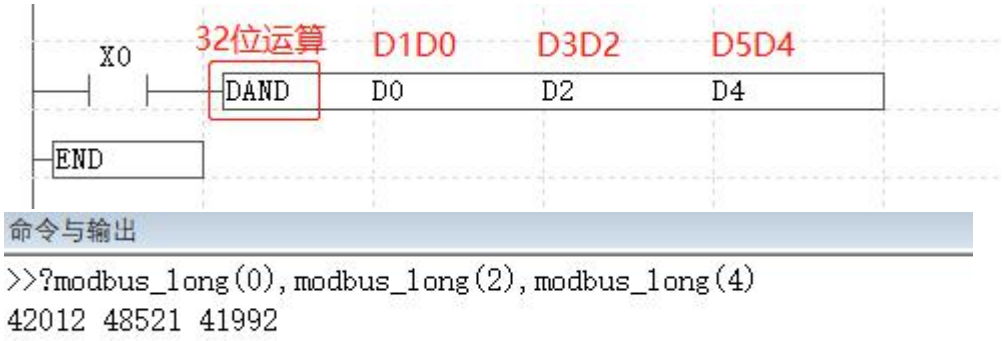
S2：KnX, KnY, KnM, KnS,T,C,D,K,H,Z,V,LV,DT,@

D：KnY, KnM, KnS, T,C,D,Z,V, LV,DT

指令格式：[DAND　S1　S2　D]

编程示例：

假设 D1D0 的数据为 42012，D3D2 为 48521，当 X0 闭合后，执行 DAND 指令，D1D0 的 32 个数据位对应 D3D2 的 32 个数据位进行逻辑与运算，运算后的值传送到 D5D4。D5D4 为 41992。



D1D0、D3D2、D5D4 的 32 位数据位如下表所示：

软元件	b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0	值
D0	1	0	1	0	0	1	0	0	0	0	0	1	1	1	0	0	42012
D1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
D2	1	0	1	1	1	1	0	1	1	0	0	0	1	0	0	1	48521
D3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
D4	1	0	1	0	0	1	0	0	0	0	0	0	1	0	0	0	41992
D5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

DANDP

指令说明：

DANDP 是 32 位脉冲执行型逻辑与运算指令，即指令激活一次，执行一次逻辑与运算。是将源操作数[S1+1,S1]与源操作数[S2+1,S2]的数据内容以位为单位进行逻辑与运算后，传送到[D+1,D]中。

[S1+1,S1]与[S2+1,S2]中指定常数（K）时，会自动地进行 BIN 转换。

逻辑与运算以位为单位，如下表中（1&1=1、0&1=0、1&0=0、0&0=0）所示变化。

表中：1=ON,，0=OFF

DAND 指令	[S1+1,S1]	[S2+1,S2]	[D+1,D]
位单位的逻辑与运算	1	1	1
	0	1	0
	1	0	0
	0	0	0

操作数：

S1：KnX, KnY, KnM, KnS,T,C,D,K,H,Z,V,LV,DT,@

S2：KnX, KnY, KnM, KnS,T,C,D,K,H,Z,V,LV,DT,@

D：KnY, KnM, KnS, T,C,D,Z,V, LV,DT

指令格式：[DANDP　S1　S2　D]

编程示例：

假设 D1D0 的数据为 40000，D3D2 为 50000，当 X0 闭合时，产生一个脉冲，DANDP 指令执行，D1D0 的 32 个数据位对应 D3D2 的 32 个数据位进行逻辑与运算，运算后的数据传送到 K8Y0，K8Y0 为 32832。Y6 和 Y17 为 ON。

D1D0、D3D2、K8Y0 的 32 位数据如下表所示

软元件	b31	…	b16	b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0	值
D1D0	0	…	0	1	0	0	1	1	1	0	0	0	1	0	0	0	0	0	0	40000
D3D2	0	…	0	1	1	0	0	0	0	1	1	0	1	0	1	0	0	0	0	50000
K8Y0	0	…	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	32832
	Y37	…	Y20	Y17	Y16	Y15	Y14	Y13	Y12	Y11	Y10	Y7	Y6	Y5	Y4	Y3	Y2	Y1	Y0	

WOR

指令说明：

WOR 是 16 位连续执行型逻辑或运算指令，是将源操作数 S1 与源操作数 S2 的数据内容以位为单位进行逻辑或运算后，传送到 D 中。

S1 与 S2 中指定常数（K）时，会自动地进行 BIN 转换。

逻辑或运算以位为单位，如下表中（1|1=1、0|1=1、1|0=1、0|0=0）所示变化。

表中：1=ON，0=OFF

WOR	S1	S2	D
位单位的逻辑或运算	1	1	1
	0	1	1
	1	0	1
	0	0	0

操作数：

S1：KnX, KnY, KnM, KnS,T,C,D,K,H,Z,V,LV,DT,@

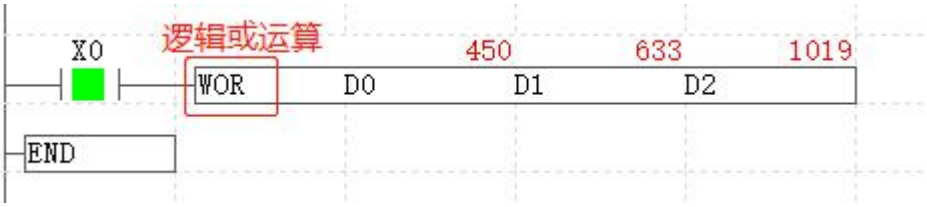
S2：KnX, KnY, KnM, KnS,T,C,D,K,H,Z,V,LV,DT,@

D：KnY, KnM, KnS, T,C,D,Z,V, LV,DT

指令格式：[WOR　S1　S2　D]

编程示例：

假设 D0 的数据为 450，D1 为 633，当 X0 闭合后，WOR 指令执行，将 D0 与 D1 的 16 个数据位一一对应，进行逻辑或运算，运算后的数据传送到 D2，D2 为 1019。



D0、D1、D2 的 16 位数据如下表所示：

软元件	b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0	值
D0	0	0	0	0	0	0	0	1	1	1	0	0	0	0	1	0	450
D1	0	0	0	0	0	0	1	0	0	1	1	1	1	0	0	1	633
D2	0	0	0	0	0	0	1	1	1	1	1	1	1	0	1	1	1019

WORP

指令说明：

WORP 是 16 位脉冲执行型逻辑或运算指令，即指令激活一次，执行一次逻辑或运算。是将源操作数 S1 与源操作数 S2 的数据内容以位为单位进行逻辑或运算后，传送到 D 中。

S1 与 S2 中指定常数（K）时，会自动地进行 BIN 转换。

逻辑或运算以位为单位，如下表中（1|1=1、0|1=1、1|0=1、0|0=0）所示变化。

表中：1=ON，0=OFF

WORP	S1	S2	D
位单位的逻辑或运算	1	1	1
	0	1	1
	1	0	1
	0	0	0

操作数：

S1：KnX, KnY, KnM, KnS,T,C,D,K,H,Z,V,LV,DT,@

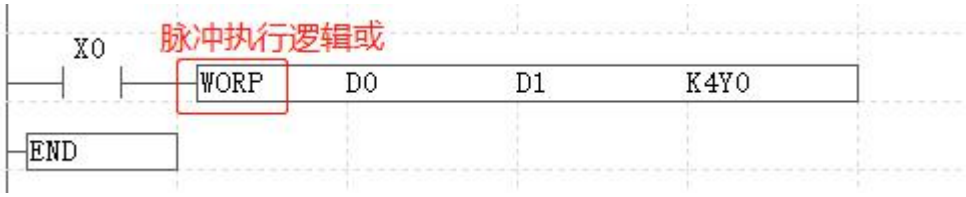
S2：KnX, KnY, KnM, KnS,T,C,D,K,H,Z,V,LV,DT,@

D：KnY, KnM, KnS, T,C,D,Z,V, LV,DT

指令格式：[WORP　S1　S2　D]

编程示例：

假设 D0 的数据为 312，D1 为 455，当 X0 闭合后，产生的上升沿脉冲激活 WORP 指令一次，将 D0 与 D1 的 16 个数据位一一对应，进行逻辑或运算，运算后的数据传送到 K4Y0，K4Y0 为 511。Y0~Y10 为 ON。



D0、D1、K4Y0 的 16 位数据如下表所示：

软元件	b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0	值
D0	0	0	0	0	0	0	0	1	0	0	1	1	1	0	0	0	312
D1	0	0	0	0	0	0	0	1	1	1	0	0	0	1	1	1	455
K4Y0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	511
	Y17	Y16	Y15	Y14	Y13	Y12	Y11	Y10	Y7	Y6	Y5	Y4	Y3	Y2	Y1	Y0	

DOR

指令说明：

DOR 是 32 位连续执行型逻辑或运算指令，是将源操作数[S1+1,S1]与源操作数[S2+1,S2]的数据内容以位为单位进行逻辑或运算后，传送到[D+1,D]中。

[S1+1,S1]与[S2+1,S2]中指定常数（K）时，会自动地进行 BIN 转换。

逻辑或运算以位为单位，如下表中（1|1=1、0|1=1、1|0=1、0|0=0）所示变化。

表中：1=ON，0=OFF

DOR	[S1+1,S1]	[S2+1,S2]	[D+1,D]
位单位的逻辑或运算	1	1	1
	0	1	1
	1	0	1
	0	0	0

操作数：

S1：KnX, KnY, KnM, KnS,T,C,D,K,H,Z,V,LV,DT,@

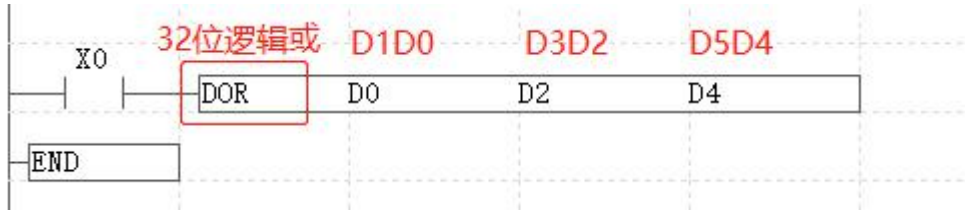
S2：KnX, KnY, KnM, KnS,T,C,D,K,H,Z,V,LV,DT,@

D：KnY, KnM, KnS, T,C,D,Z,V, LV,DT

指令格式：[DOR　S1　S2　D]

编程示例：

假设 D1D0 的数据为 100101，D3D2 为 44525，当 X0 闭合后，DOR 指令执行，将 D1D0 与 D3D2 的 32 个数据位一一对应，进行逻辑或运算，运算后的数据传送到 D5D4，D5D4 的数据为 44525。



D1D0、D3D2、D5D4 的 32 位数据位如下表所示：

软元件	b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0	值

D0	1	0	0	0	0	1	1	1	0	0	0	0	0	1	0	1	100101
D1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	
D2	1	0	1	0	1	1	0	1	1	1	1	0	1	1	0	1	44525
D3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
D4	1	0	1	0	1	1	1	1	1	1	1	0	1	1	0	1	110573
D5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	

DORP

指令说明：

DORP 是 32 位脉冲执行型逻辑或运算指令，即指令激活一次，执行一次逻辑或运算。是将源操作数[S1+1,S1]与源操作数[S2+1,S2]的数据内容以位为单位进行逻辑或运算后，传送到[D+1,D]中。

[S1+1,S1]与[S2+1,S2]中指定常数（K）时，会自动地进行 BIN 转换。

逻辑或运算以位为单位，如下表中（1|1=1、0|1=1、1|0=1、0|0=0）所示变化。

表中：1=ON，0=OFF

DORP	[S1+1,S1]	[S2+1,S2]	[D+1,D]
位单位的逻辑或运算	1	1	1
	0	1	1
	1	0	1
	0	0	0

操作数：

S1：KnX, KnY, KnM, KnS,T,C,D,K,H,Z,V,LV,DT,@

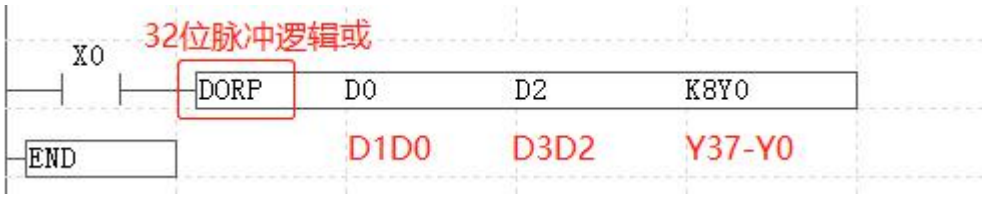
S2：KnX, KnY, KnM, KnS,T,C,D,K,H,Z,V,LV,DT,@

D：KnY, KnM, KnS, T,C,D,Z,V, LV,DT

指令格式：[DORP　S1　S2　D]

编程示例：

假设 D1D0 的数据为 54542，D3D2 为 87873，当 X0 闭合时，产生一个上升沿脉冲，DORP 指令执行一次，将 D1D0 与 D3D2 的 32 个数据位一一对应，进行逻辑或运算，运算后的数据传送到 D5D4，D5D4 的数据为 120655。



D1D0、D3D2、K8Y0 的 32 位数据如下表所示：

软元件	b31	…	b16	b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0	值
D1D0	0	…	0	1	1	0	1	0	1	0	1	0	0	0	0	1	1	1	0	54542
D3D2	0	…	1	0	1	0	1	0	1	1	1	0	1	0	0	0	0	0	1	87873
K8Y0	0	…	1	1	1	0	1	0	1	1	1	0	1	0	0	1	1	1	1	120655
	Y37	…	Y20	Y17	Y16	Y15	Y14	Y13	Y12	Y11	Y10	Y7	Y6	Y5	Y4	Y3	Y2	Y1	Y0	

WXOR

指令说明：

WXOR 是 16 位连续执行型逻辑异或运算指令，是将源操作数 S1 与源操作数 S2 的数据内容以位为单位进行逻辑异或运算后，传送到 D 中。

S1 与 S2 中指定常数（K）时，会自动地进行 BIN 转换。

逻辑异或运算以位为单位，如下表中（1∨1=1、0∨1=1、1∨0=1、0∨0=0）所示变化。

表中：1=ON，0=OFF

WXOR	S1	S2	D
位单位的逻辑异或运算	1	1	0
	0	1	1
	1	0	1
	0	0	0

操作数：

S1：KnX, KnY, KnM, KnS,T,C,D,K,H,Z,V,LV,DT,@

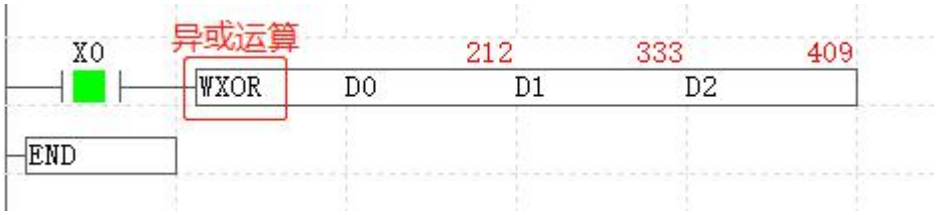
S2：KnX, KnY, KnM, KnS,T,C,D,K,H,Z,V,LV,DT,@

D：KnY, KnM, KnS, T,C,D,Z,V, LV,DT

指令格式：[WXOR　S1　S2　D]

编程示例：

假设 D0 的数据为 212，D1 为 333，当 X0 闭合后，WXOR 指令执行，将 D0 与 D1 的 16 个数据位一一对应，进行异或逻辑运算，运算后的值传送到 D2，D2 为 409。



D0、D1、D2 的 16 位数据如下表所示：

软元件	b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0	值
D0	0	0	0	0	0	0	0	0	1	1	0	1	0	1	0	0	212
D1	0	0	0	0	0	0	0	1	0	1	0	0	1	1	0	1	333
D2	0	0	0	0	0	0	0	1	1	0	0	1	1	0	0	1	409

WXORP

指令说明：

WXORP 是 16 位脉冲执行型逻辑异或运算指令，即指令激活一次，执行一次逻辑异或运算。是将源操作数 S1 与源操作数 S2 的数据内容以位为单位进行逻辑异或运算后，传送到 D 中。

S1 与 S2 中指定常数（K）时，会自动地进行 BIN 转换。

逻辑异或运算以位为单位，如下表中（1∨1=1、0∨1=1、1∨0=1、0∨0=0）所示变化。

表中：1=ON，0=OFF

WXORP	S1	S2	D
位单位的逻辑异或运算	1	1	0
	0	1	1
	1	0	1
	0	0	0

操作数：

S1：KnX, KnY, KnM, KnS,T,C,D,K,H,Z,V,LV,DT,@

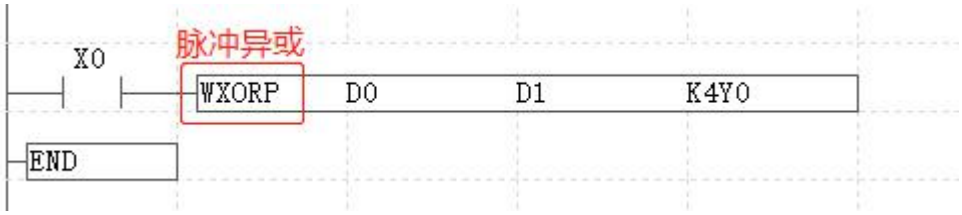
S2：KnX, KnY, KnM, KnS,T,C,D,K,H,Z,V,LV,DT,@

D：KnY, KnM, KnS, T,C,D,Z,V, LV,DT

指令格式：[WXORP　S1　S2　D]

编程示例：

假设 D0 的数据为 155，D1 为 360，当 X0 闭合时，产生一个脉冲使 WXORP 指令执行一次，将 D0 与 D1 的 16 个数据位一一对应，进行逻辑异或运算后传送到 K4Y0，K4Y0 的数据为 499。



D0、D1、K4Y0 的 16 位数据如下表所示：

软元件	b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0	值
D0	0	0	0	0	0	0	0	0	1	0	0	1	1	0	1	1	155
D1	0	0	0	0	0	0	0	1	0	1	1	0	1	0	0	0	360
K4Y0	0	0	0	0	0	0	0	1	1	1	1	1	0	0	1	1	499
	Y17	Y16	Y15	Y14	Y13	Y12	Y11	Y10	Y7	Y6	Y5	Y4	Y3	Y2	Y1	Y0	

DXOR

指令说明：

DXOR 是 32 位连续执行型逻辑异或运算指令，是将源操作数[S1+1,S1]与源操作数[S2+1,S2]的数据内容以位为单位进行逻辑异或运算后，传送到[D+1,D]中。

[S1+1,S1]与[S2+1,S2]中指定常数（K）时，会自动地进行 BIN 转换。

逻辑异或运算以位为单位，如下表中（1∨1=0、0∨1=1、1∨0=1、0∨0=0）所示变化。

表中：1=ON，0=OFF

DXOR	[S1+1,S1]	[S2+1,S2]	[D+1,D]
位单位的逻辑异或运算	1	1	0
	0	1	1
	1	0	1
	0	0	0

操作数：

S1：KnX, KnY, KnM, KnS,T,C,D,K,H,Z,V,LV,DT,@

S2：KnX, KnY, KnM, KnS,T,C,D,K,H,Z,V,LV,DT,@

D：KnY, KnM, KnS, T,C,D,Z,V, LV,DT

指令格式：[DXOR　S1　S2　D]

编程示例：

假设 D1D0 的数据为 85215，D3D2 为 68643，当 X0 闭合，DXOR 指令执行，将 D1D0 与 D3D2 的 32 个数据位一一对应进行逻辑异或运算，运算后的数据传送到 D5D4。D5D4 的数据为 16636。

D1D0、D3D2、D5D4 的 32 位数据位如下表所示：

软元件	b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0	值
D0	0	1	0	0	1	1	0	0	1	1	0	1	1	1	1	1	85215
D1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	
D2	0	0	0	0	1	1	0	0	0	0	1	0	0	0	1	1	68643
D3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	
D4	0	1	0	0	0	0	0	0	1	1	1	1	1	1	0	0	16636
D5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

DXORP

令说明：

DXORP 是 32 位脉冲执行型逻辑异或运算指令，即指令激活一次，执行一次逻辑异或运算。是将源操作数[S1+1,S1]与源操作数[S2+1,S2]的数据内容以位为单位进行逻辑异或运算后，传送到[D+1,D]中。

[S1+1,S1]与[S2+1,S2]中指定常数（K）时，会自动地进行 BIN 转换。

逻辑异或运算以位为单位，如下表中（1∨1=0、0∨1=1、1∨0=1、0∨0=0）所示变化。

表中：1=ON，0=OFF

DXORP	[S1+1,S1]	[S2+1,S2]	[D+1,D]
位单位的逻辑异或运算	1	1	0
	0	1	1
	1	0	1
	0	0	0

操作数：

S1：KnX, KnY, KnM, KnS,T,C,D,K,H,Z,V,LV,DT,@

S2：KnX, KnY, KnM, KnS,T,C,D,K,H,Z,V,LV,DT,@

D：KnY, KnM, KnS, T,C,D,Z,V, LV,DT

指令格式：[DXORP　S1　S2　D]

编程示例：

假设 D1D0 的数据为 74562，D3D2 为 58469，当 X0 闭合时，产生一个上升沿脉冲，激活 DXORP 指令执行一次，将 D1D0 与 D3D2 的 32 个数据位一一对应，进行逻辑异或运算，运算后的数据传送到 K8Y0，K8Y0 的数据为 116519。

D1D0、D3D2、K8Y0 的 32 位数据如下表所示：

软元件	b31	…	b16	b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0	值
D1D0	0	…	1	0	0	1	0	0	0	1	1	0	1	0	0	0	0	1	0	74562
D3D2	0	…	0	1	1	1	0	0	1	0	0	0	1	1	0	0	1	0	1	58469
K8Y0	0	…	1	1	1	0	0	0	1	1	1	0	0	1	0	0	1	1	1	116519
	Y37	…	Y20	Y17	Y16	Y15	Y14	Y13	Y12	Y11	Y10	Y7	Y6	Y5	Y4	Y3	Y2	Y1	Y0	

NEG

指令说明：

NEG 为 16 位连续执行型补码指令，是将操作数 D 的数据内容中的 16 个数据位按位取反（0→1,1→0）后再加一，运算的结果传送到原先操作数 D 中。

因为连续执行型指令则在每一个扫描周期都执行运算，必须要脉冲指令触发条件下使用 NEG 指令，务必引起注意。

处理的数值范围：-32768 ~ +32767

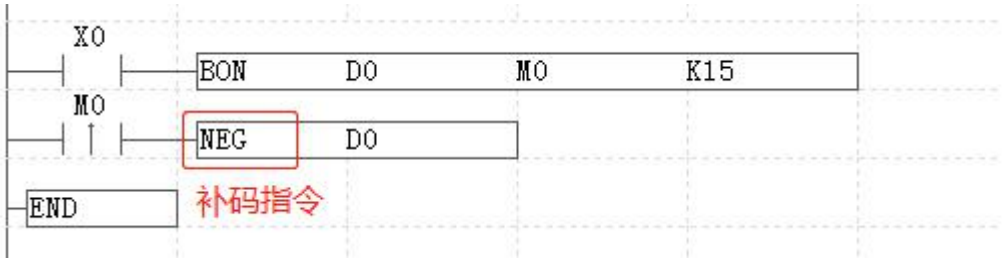
操作数：

D：KnY, KnM, KnS,T,C,D,Z,V,LV,DT

指令格式：[NEG　D]

编程示例：

假设 D0 的数据为-45，X0 导通，BON 指令执行，当 D0 的 16 个数据位（b0~b15）中的 b15 为 1（因为 D0 的值是负数）时，M0 为 ON。当 M0 闭合时，产生一个上升沿脉冲，使 NEG 指令执行一次，得到 D0 的绝对值，D0 为 45。



NEG 指令执行后，D0 数据的变化如下。

D0= -45 时，它的 16 个数据位：

软元件	b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0

D0	1	1	1	1	1	1	1	1	1	1	0	1	0	0	1	1

按位取反后，它的 16 个数据位：

软元件	b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
D0	0	0	0	0	0	0	0	0	0	0	1	0	1	1	0	0

加一后，它的 16 个数据位：

软元件	b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
D0	0	0	0	0	0	0	0	0	0	0	1	0	1	1	0	1

此时，D0=45。

NEGP

指令说明：

NEGP 为 16 位脉冲执行型补码指令，即指令激活一次，执行一次补码运算。是将操作数 D 的数据内容中的 16 个数据位按位取反（0→1,1→0）后再加一，运算的结果传送到原先操作数 D 中。

程序在循环扫描时，因为 NEG 指令具有连续执行型，所以它操作的软元件的数据不断变化，所以在数据的补码运算一般用脉冲执行型的 NEGP 指令。

处理的数值范围：-32768 ~ +32767

操作数：

D：KnY, KnM, KnS,T,C,D,Z,V,LV,DT

指令格式：[NEGP　D]

编程示例：

在下图中，M0 不必为上升沿脉冲指令，因为 NEGP 具有脉冲执行型，假设 D0 为-45，满足 BON 指令条件，M0 闭合，执行 NEGP 指令，D0 为 45。

NEGP 指令执行后，D0 数据的变化如下。

D0= -45 时，它的 16 个数据位：

软元件	b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
D0	1	1	1	1	1	1	1	1	1	1	0	1	0	0	1	1

按位取反后，它的 16 个数据位：

软元件	b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
D0	0	0	0	0	0	0	0	0	0	0	1	0	1	1	0	0

加一后，它的 16 个数据位：

软元件	b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
D0	0	0	0	0	0	0	0	0	0	0	1	0	1	1	0	1

此时，D0=45。

DNEG

指令说明：

DNEG 为 32 位连续执行型补码指令，是将操作数[D+1,D]的数据内容中的 32 个数据位按位取反（0→1,1→0）后再加一，运算的结果传送到原先操作数[D+1,D]中。

因为连续执行型指令则在每一个扫描周期都执行运算，必须要脉冲指令触发条件下使用 DNEG 指令，务必引起注意。

处理的数值范围：-2147483648 ~ +2147483647

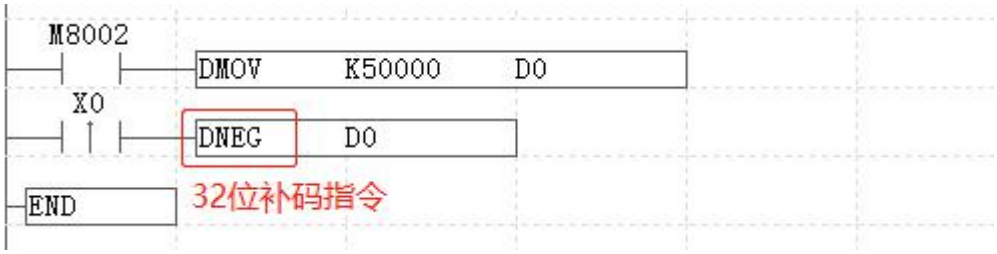
操作数：

D：KnY, KnM, KnS,T,C,D,Z,V,LV,DT

指令格式：[DNEG　D]

编程示例：

控制器上电，M8002 瞬间导通一次，DMOV 指令执行，将常数 50000 传送到 D1D0，当 X0 的上升沿脉冲触点触发时，DNEG 指令执行，将 D1D0 里面的 32 个数据位按位取反后加一，补码后的 D1D0 的数据为-50000，是原来的负数。



D1D0=50000 时，执行 DNEG 指令，D1D0 的数据位变化

D1 的 16 个数据为高位																D0 的 16 个数据为低位															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	1	1	0	1	0	1	0	0	0	0

D1D0 的数据按位取反，它们的数据变化

D1 的 16 个数据为高位																D0 的 16 个数据为低位															
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	1	1	1	1	0	0	1	0	1	0	1	1	1	1

D1D0 取反后的数据再加一，它们的数据变化

D1 的 16 个数据为高位																D0 的 16 个数据为低位															
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	1	1	1	1	0	0	1	0	1	1	0	0	0	0

D1D0 的数据经过取反加一运算后，D1D0= -50000。

DNEGP

指令说明：

DNEGP 为 32 位脉冲执行型补码指令，即指令激活一次，执行一次补码运算。是将操作数[D+1,D]的数据内容中的 32 个数据位按位取反（0→1,1→0）后再加一，运算的结果传送到原先操作数[D+1,D]中。

程序在循环扫描时，因为 DNEG 指令具有连续执行型，所以它操作的软元件的数据不

断变化，所以在数据的补码运算一般用脉冲执行型的 DNEGP 指令。

处理的数值范围：-2147483648 ~ +2147483647

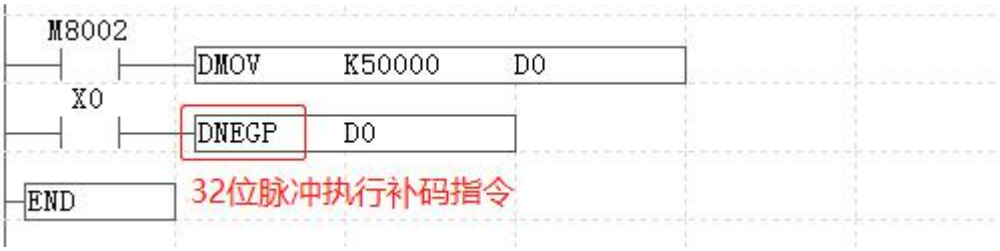
操作数：

D：KnY, KnM, KnS,T,C,D,Z,V,LV,DT

指令格式：[DNEGP　D]

编程示例：

满足条件后，D1D0 的数据为 50000，注意此时的 X0 不是上升沿脉冲触发触点，因为 DNEGP 具有脉冲执行型，当 X0 闭合时，D1D0 的数据也只进行一次补码运算，运算后的数据传送到 D1D0，此时 D1D0 为-50000。



D1D0=50000 时，执行 DNEGP 指令，D1D0 的数据位变化

D1 的 16 个数据为高位																D0 的 16 个数据为低位															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	1	1	0	1	0	1	0	0	0	0

D1D0 的数据按位取反，它们的数据变化

D1 的 16 个数据为高位																D0 的 16 个数据为低位															
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	1	1	1	1	0	0	1	0	1	0	1	1	1	1

D1D0 取反后的数据再加一，它们的数据变化

D1 的 16 个数据为高位																D0 的 16 个数据为低位															
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	1	1	1	1	0	0	1	0	1	1	0	0	0	0

D1D0 的数据经过取反加一运算后，D1D0= -50000。

3.6 移位指令

32 位指令使用连续两个数据寄存器 D 存储空间。

分类	指令	位数	脉冲	功能	操作数类型
循环右移	ROR	16 位	-	[ROR　D　n] 将 D 的数据向右循环移动 n 位。最后移出的位状态存于进位标志 M8022 中	D：KnY, KnM,KnS, T,C, D,Z,V,LV,DT n：D,K,H
	RORP		√		
	DROR	32 位	-		
	DRORP		√		
循环左移	ROL	16 位	-	[ROR　D　n] 将 D 的数据向左循环移动 n 位。最后移出的位状态存于进位标志 M8022 中	D：KnY, KnM,KnS, T,C, D,Z,V,LV,DT n：D,K,H
	ROLP		√		
	DROL	32 位	-		
	DROLP		√		
带进位循	RCR	16 位	-	[RCR　D　n] D 的 16 位+1 位(进位标志 M8022)数	D：KnY, KnM,KnS, T,C, D,Z,V,LV,DT
	RCRP		√		

环右移	DRCR	32 位	-	据向右移动 n 位。最后移出的位值存于进位标志 M8022 中	n：D,K,H
	DRCRP		√		
带进位循环左移	RCL	16 位	-	[RCL D n] D 的 16 位+1 位(进位标志 M8022)数据向左移动 n 位。最后移出的位值存于进位标志 M8022 中	D：KnY, KnM,KnS, T,C, D,Z,V,LV,DT n：D,K,H
	RCLP		√		
	DRCL	32 位	-		
	DRCLP		√		
特殊右移	SFTR	16 位	-	[SFTR S D n1 n2] D 起始的 n1 位数据右移 n2 位。移位后，低位溢出的数据删除，高位缺失的 n2 位由 S 起始的 n2 位补上	S：X,Y,M,S,@ D：Y,M,S,@ n1： K,H,@ ,D,C,T,V,Z n2： K,H,@ ,D,C,T,V,Z
	SFTRP		√		
特殊左移	SFTL	16 位	-	[SFTL S D n1 n2] 将 D 起始的 n1 位数据左移 n2 位。移位后，高位溢出的数据删除，低位缺失的 n2 位由 S 起始的 n2 位补上	S：X,Y,M,S,@ D：Y,M,S,@ n1： K,H,@ ,D,C,T,V,Z n2 ： K,H,@ ,D,C,T,V,Z
	SFTLP		√		
特殊右移	WSFR	16 位	-	[WSFR S D n1 n2] 将 D 起始的 n1 个字软元件右移 n2 个字。移动后，低位溢出的数据删除，高位缺失的 n2 个软元件的数据由 S 起始的 n2 个软元件的数据补上	S： KnX,KnY, KnM, KnS,T,C,D,LV,DT,@ D： KnY, KnM, KnS, T,C,D,LV,DT n1： K,H,@ ,D,C,T,V,Z n2 ： K,H,@ ,D,C,T,V,Z
	WSFRP		√		
特殊左移	WSFL	16 位	-	[WSFL S D n1 n2] 将 D 起始的 n1 个字软元件左移 n2 个字。移动后，高位溢出的数据删除，低位缺失的 n2 个软元件的数据由 S 起始的 n2 个软元件的数据补上	S： KnX,KnY, KnM, KnS,T,C,D,LV,DT,@ D： KnY, KnM, KnS, T,C,D,LV,DT n1： K,H,@ ,D,C,T,V,Z n2： K,H,@ ,D,C,T,V,Z
	WSFLP		√		
移位写入	SFWR	16 位	-	[SFWR S D n] 将 S 的值写入由 D 地址起始，个数为 n 的“先进先出”队列中，当指令执行时，指针 Dn 内容值先加 1，S 数据从 Dn+1 开始存储	S： KnX,KnY, KnM, KnS,T,C,D,LV,DT,@ D： KnY, KnM, KnS, T,C,D, LV,DT n：K,H,@,D,C,T,V,Z
	SFWRP		√		
移位读出	SFRD	16 位	-	[SFWD S D n] 从“先进先出”队列 S 的首项读出到 D 中，当指令执行时，指针 Dn 内容值先减 1，之后 S 的值会写入到 Dn+1	S： KnX,KnY, KnM, KnS,T,C,D,LV,DT,@ D： KnY, KnM, KnS, T,C,D, LV,DT
	SFRDP		√		

				开始的地址	n：K,H,@,D,C,T,V,Z

ROR

指令说明：

ROR 是 16 位连续执行型循环右移指令，即每个扫描周期执行一次数据移位运算。是将不包括进位标志在内的目标操作数[D]的 16 位数据向右循环移动 n 位。最后移出的位状态存于进位标志 M8022 中。

若操作数[D]为 KnY,KnM,KnS 等位数指定软元件时，则只有 K4Y,K4M,K4S 有效。

操作数：

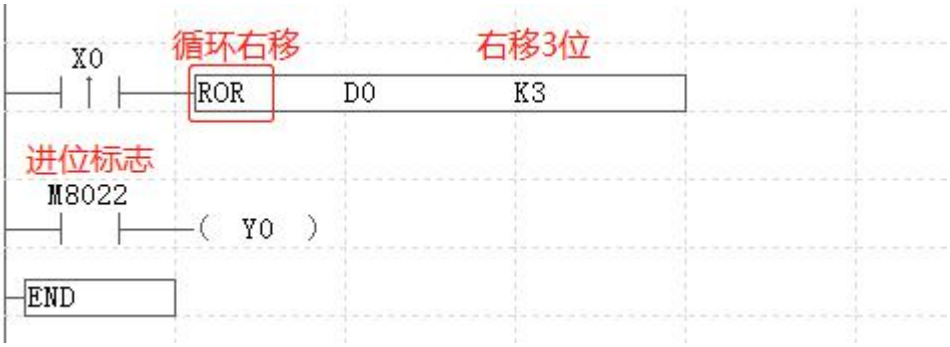
D：KnY, KnM, KnS, T,C,D,Z,V, LV,DT

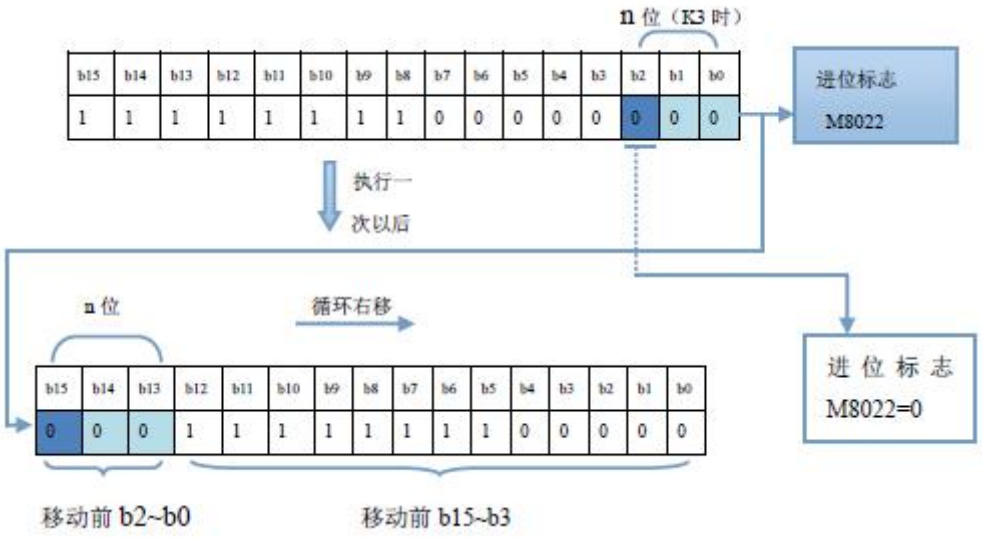
n：D,K,H

指令格式：[ROR　D　n]

编程示例：

假设 D0=-256，当 X0 从 OFF 变 ON 时，ROR 连续右移指令执行一次，将 D0 的 16 个数据位向右移 3 位，最后移出的位状态为 0 时，M8022 进位标志为 OFF，若最后移出的位状态为 1 时，M8022 进位标志为 ON。





RORP

指令说明：

RORP 是 16 位脉冲执行型循环右移指令，即指令激活一次，执行一次数据移位运算。是将不包括进位标志在内的目标操作数[D]的 16 位数据向右循环移动 n 位。最后移出的位状态存于进位标志 M8022 中。

若操作数[D]为 KnY,KnM,KnS 等位数指定软元件时，则只有 K4Y,K4M,K4S 有效。

操作数：

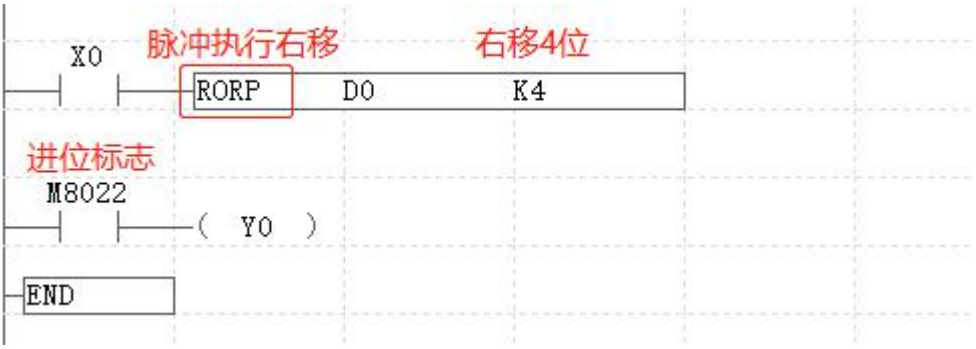
D：KnY, KnM, KnS, T,C,D,Z,V, LV,DT

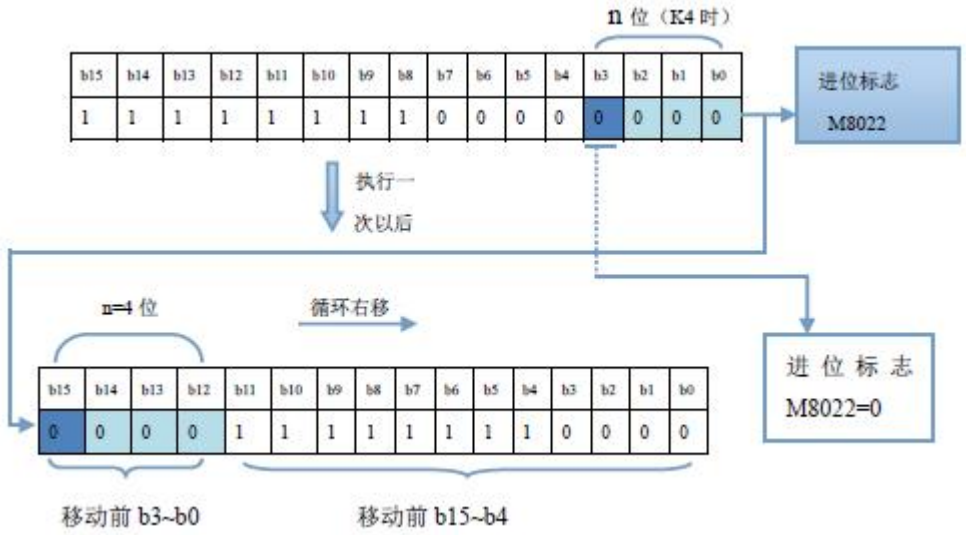
n：D,K,H

指令格式：[RORP　D　n]

编程示例：

当 X0 闭合时，RORP 右移指令执行一次，将 D0 的 16 个数据位向右移 4 位，最后移出的位状态为 0 时，M8022 进位标志为 OFF，b0 移出的状态为 1 时，M8022 进位标志为 ON。





DROR

指令说明：

DROR 是 32 位连续执行型循环右移指令，即每个扫描周期执行一次数据移位运算。是将不包括进位标志在内的目标操作数[D+1,D]的 32 位数据向右循环移动 n 位。最后移出的位状态存于进位标志 M8022 中。

若操作数[D]为指定位数软元件，则只有 K8Y, K8M, K8S 有效。

操作数：

D：KnY, KnM, KnS, T,C,D,Z,V, LV,DT

n：D,K,H

指令格式：[DROR　D　n]

编程示例：

当 X0 闭合时，DROR 指令每个扫描周期都执行右移，当执行一次时，将 K8Y0 的 32 个数据位向右移 4 位，最后移出的位状态为 1 时，进位标志 M8022=ON，若为 0，则 M8022=OFF。

DRORP

指令说明：

DRORP 是 32 位脉冲执行型循环右移指令，即指令激活一次，执行一次数据移位运算。是将不包括进位标志在内的目标操作数[D+1,D]的 32 位数据向右循环移动 n 位。最后移出的位状态存于进位标志 M8022 中。

若操作数[D]为指定位数软元件，则只有 K8Y, K8M, K8S 有效。

操作数：

D：KnY, KnM, KnS, T,C,D,Z,V, LV,DT

n：D,K,H

指令格式：[DRORP　D　n]

编程示例：

当 Y37 闭合后，置位 M100，常开触点 M100 闭合，M8013 触点接使得 DRORP 指令以

1S 为周期执行一次，让 K8Y0 的 32 个数据位循环向右移 1 位，最后移出的位状态为 1 时，进位标志 M8022=ON，若为 0，则 M8022=OFF。

ROL

指令说明：

ROL 是 16 位连续执行型循环左移指令，即每个扫描周期执行一次数据移位运算。是将不包括进位标志在内的目标操作数[D]的 16 位数据向左循环移动 n 位。最后移出的位状态存于进位标志 M8022 中。

若操作数[D]为 KnY,KnM,KnS 等位数指定软元件时，则只有 K4Y,K4M,K4S 有效。

操作数：

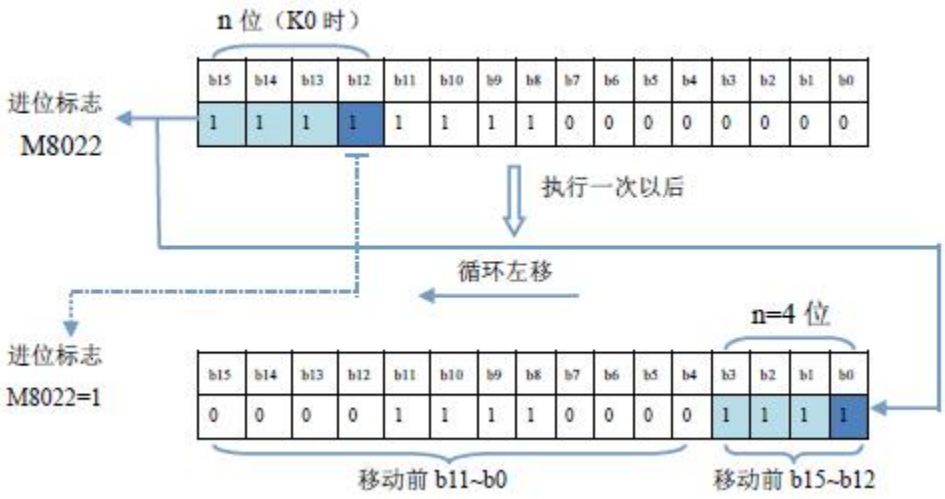
D：KnY, KnM, KnS, T,C,D,Z,V, LV,DT

n：D,K,H

指令格式：[ROL　D　n]

编程示例：

当 X0 闭合时，上升沿脉冲触发 ROL 指令执行一次，将 D0 的 16 个数据位向左移 4 位，最后移出的位状态为 1 时，进位标志 M8022=ON，若为 0，则 M8022=OFF。



ROLP

指令说明：

ROLP 是 16 位脉冲执行型循环左移指令，即指令激活一次，执行一次数据移位运算。是将不包括进位标志在内的目标操作数[D]的 16 位数据向左循环移动 n 位。最后移出的位状态存于进位标志 M8022 中。

若操作数[D]为 KnY,KnM,KnS 等位数指定软元件时，则只有 K4Y,K4M,K4S 有效。

操作数：

D：KnY, KnM, KnS, T,C,D,Z,V, LV,DT

n：D,K,H

指令格式：[ROLP　D　n]

编程示例：

当 X0 闭合时，置位 M100，M100 的上升沿触点触发，将 K1 传送到 K4Y0，M100 的常开触点闭合，M8013 触点接通 0.5 秒，断开 0.5 秒，使得 ROLP 指令以 1S 为周期执行一次，让 K4Y0 的 16 个数据位中 b0 的状态 1 循环左移 1 位，最后移出的位状态为 1 时，进位标志 M8022=ON，若为 0，则 M8022=OFF。

K4Y0 的数据位变化情况：

执行前：

M8022=OFF

Y17	Y16	Y15	Y14	Y13	Y12	Y11	Y10	Y7	Y6	Y5	Y4	Y3	Y2	Y1	Y0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

执行 1 次：
M8022=OFF

Y17	Y16	Y15	Y14	Y13	Y12	Y11	Y10	Y7	Y6	Y5	Y4	Y3	Y2	Y1	Y0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0

执行 2 次：
M8022=OFF

Y17	Y16	Y15	Y14	Y13	Y12	Y11	Y10	Y7	Y6	Y5	Y4	Y3	Y2	Y1	Y0
0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0

执行 15 次：
M8022=ON

Y17	Y16	Y15	Y14	Y13	Y12	Y11	Y10	Y7	Y6	Y5	Y4	Y3	Y2	Y1	Y0
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DROL

指令说明：

DROL 是 32 位连续执行型循环左移指令，即每个扫描周期执行一次数据移位运算。是将不包括进位标志在内的目标操作数[D+1,D]的 32 位数据向左循环移动 n 位。最后移出的位状态存于进位标志 M8022 中。

若操作数[D]为指定位数软元件，则只有 K8Y, K8M, K8S 有效。

操作数：

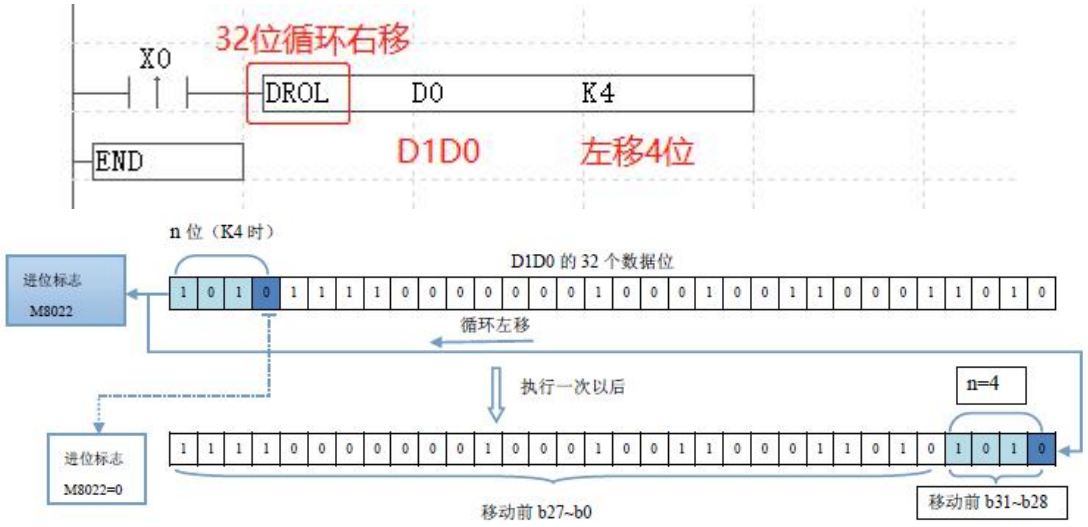
D：KnY, KnM, KnS, T,C,D,Z,V, LV,DT

n：D,K,H

指令格式：[DROL　D　n]

编程示例：

当 X0 闭合时，上升沿脉冲触发 DROR 指令只执行一次，将 D1D0 的 32 个数据位向右移 4 位，最后移出的位状态为 1 时，进位标志 M8022=ON，若为 0，则 M8022=OFF。



DROLP

指令说明：

DROL 是 32 位脉冲执行型循环左移指令，即指令激活一次，执行一次数据移位运算。

是将不包括进位标志在内的目标操作数[D+1,D]的 32 位数据向左循环移动 n 位。最后移出的位状态存于进位标志 M8022 中。

若操作数[D]为指定位数软元件，则只有 K8Y, K8M, K8S 有效。

操作数：

D：KnY, KnM, KnS, T,C,D,Z,V, LV,DT

n：D,K,H

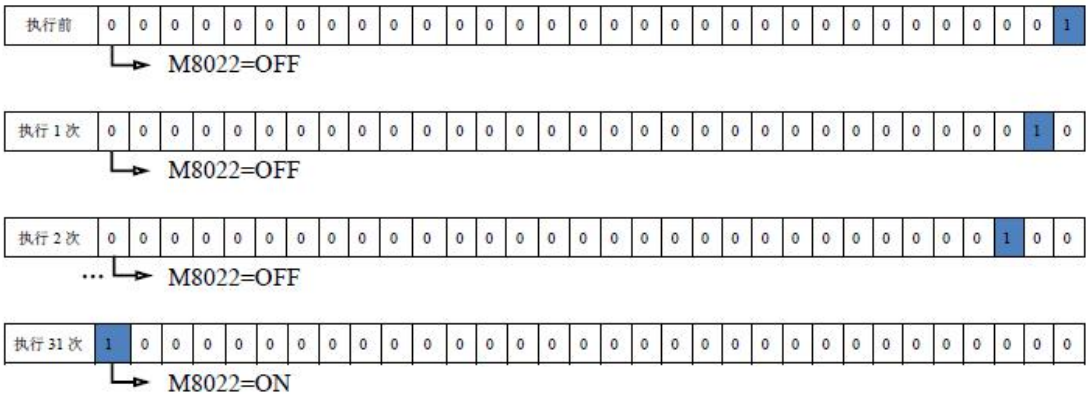
指令格式：[DROLP　D　n]

编程示例：

当 X0 闭合时，置位 M100，M100 的上升沿触点触发，将 K1 传送到 K8Y0。

M100 的常开触点闭合，M8013 触点接通 0.5 秒，断开 0.5 秒，使得 ROLP 指令以 1S 为周期执行一次，让 K8Y0 的 32 个数据位中 b0 的状态 1 循环左移 1 位，最后移出的位状态为 1 时，进位标志 M8022=ON，若为 0，则 M8022=OFF。

Y37~Y0 的变化情况：



RCR

指令说明：

RCR 是 16 位连续执行型带进位循环右移指令，即每个扫描周期执行一次数据移位运算。将包括进位标志在内的目标操作数[D]的 16 位+1 位（进位标志 M8022）数据向右移动 n 位。最后移出的位值存于进位标志 M8022 中。

因为循环回路中有进位标志，所以如果执行循环移位指令之前 M8022 就先 ON 或 OFF，则会被送入目标操作数中。

若操作数[D]为 KnY,KnM,KnS 等位数指定软元件时，则只有 K4Y,K4M,K4S 有效。

操作数：

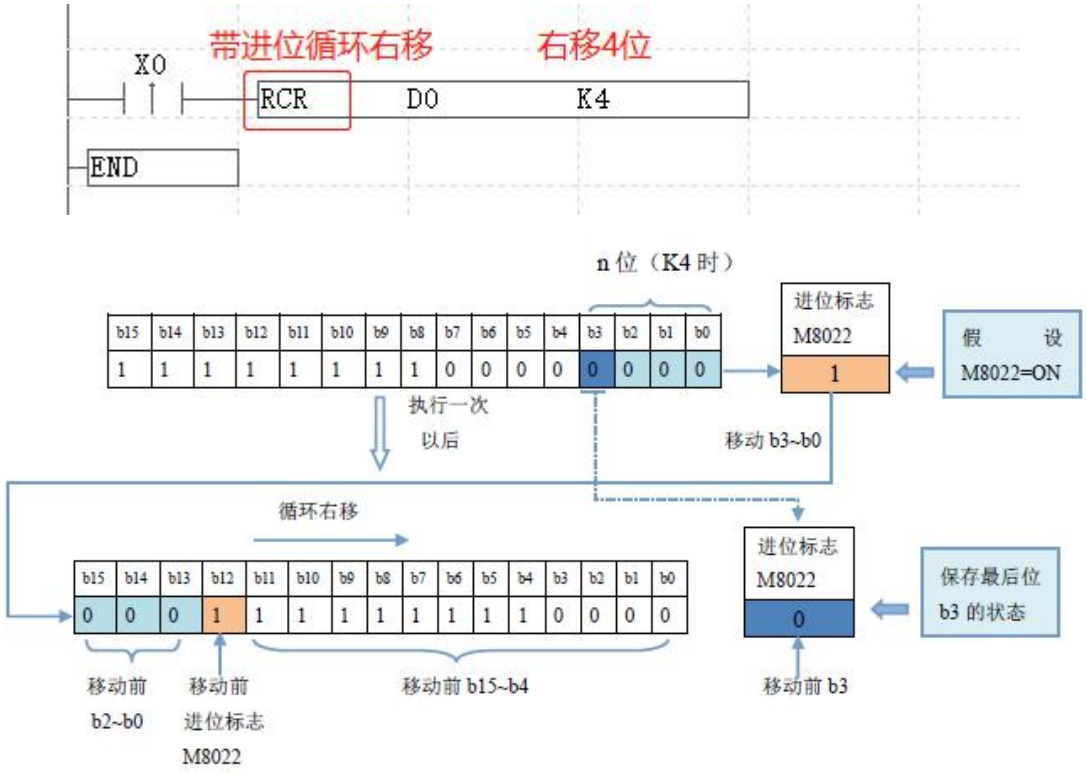
D：KnY, KnM, KnS, T,C,D,Z,V, LV,DT

n ： D,K,H

指令格式：[RCR D n]

编程示例：

当上升沿触点 X0 闭合时，RCR 指令执行一次，将 D0 的 16 个数据位+1 个标志位向右移 4 位。



RCRP

指令说明：

RCRP 是 16 位脉冲执行型带进位循环右移指令，即指令激活一次，执行一次数据移位运算。将包括进位标志在内的目标操作数[D]的 16 位+1 位（进位标志 M8022）数据向右移动 n 位。最后移出的位值存于进位标志 M8022 中。

因为循环回路中有进位标志，所以如果执行循环移位指令之前 M8022 就先 ON 或 OFF，则会被送入目标操作数中。

若操作数[D]为 KnY,KnM,KnS 等位数指定软元件时，则只有 K4Y,K4M,K4S 有效。

操作数：

D：KnY, KnM, KnS, T,C,D,Z,V, LV,DT

n ： D,K,H

指令格式：[RCRP D n]

编程示例：

当 X0 闭合时，置位 M100，M100 的上升沿触点触发，将 1 传送到 K4Y0。M100 的常

开触点闭合，M8013 触点接通 0.5 秒，断开 0.5 秒，使得 RCRP 指令以 1S 为周期执行一次，让 K4Y0 的 16 个数据位加进位标志位 M8022 向右移动 1 位。

K4Y0 数据位的变化情况：

执行前：
M8022=OFF

Y17	Y16	Y15	Y14	Y13	Y12	Y11	Y10	Y7	Y6	Y5	Y4	Y3	Y2	Y1	Y0	进位
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0

执行 1 次：
M8022=ON

Y17	Y16	Y15	Y14	Y13	Y12	Y11	Y10	Y7	Y6	Y5	Y4	Y3	Y2	Y1	Y0	进位
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

执行 2 次：
M8022=OFF

Y17	Y16	Y15	Y14	Y13	Y12	Y11	Y10	Y7	Y6	Y5	Y4	Y3	Y2	Y1	Y0	进位
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DRCR

指令说明：

RCR 是 32 位连续执行型带进位循环右移指令，即每个扫描周期执行一次数据移位运算。将包括进位标志在内的目标操作数[D+1,D]的 32 位+1 位（进位标志 M8022）数据向右移动 n 位。最后移出的位值存于进位标志 M8022 中。

因为循环回路中有进位标志，所以如果执行循环移位指令之前 M8022 就先 ON 或 OFF，则会被送入目标操作数中。

若操作数[D]为指定位数软元件，则只有 K8Y, K8M, K8S 有效。

操作数：

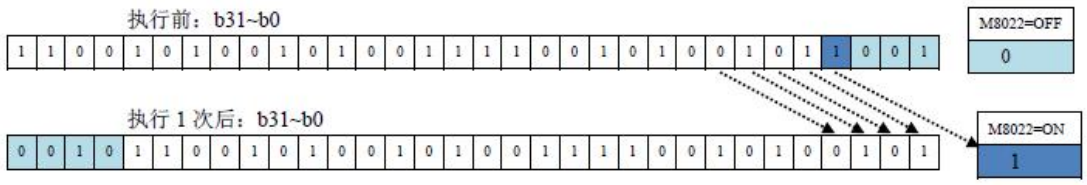
D：KnY, KnM, KnS, T,C,D,Z,V, LV,DT

n ： D,K,H

指令格式：[DRCR D n]

编程示例：

当上升沿触点 X0 闭合时，DRCR 指令执行一次，将 D1D0 的 32 位数据加进位标志位 M8022 向右移动四位。



DRCRP

指令说明：

DRCRP 是 32 位脉冲执行型带进位循环右移指令，即指令激活一次，执行一次数据移位运算。将包括进位标志在内的目标操作数[D+1,D]的 32 位+1 位（进位标志 M8022）数据向右移动 n 位。最后移出的位值存于进位标志 M8022 中。

因为循环回路中有进位标志，所以如果执行循环移位指令之前 M8022 就先 ON 或 OFF，则会被送入目标操作数中。

若操作数[D]为指定位数软元件，则只有 K8Y, K8M, K8S 有效。

操作数：

D：KnY, KnM, KnS, T,C,D,Z,V, LV,DT

n ： D,K,H

指令格式：[DRCRP　D　n]

编程示例：

当 X0 闭合，置位 M100。M100 的上升沿触点触发一次，将 K1 传送到 K8Y0，M100 的常开触点闭合，M8013 以 1S 为周期断开闭合，所以 DRCRP 指令 1S 执行一次，将 K8Y0 的 32 位数据加进位标志位 M8022 向右移动 1 位。

RCL

指令说明：

RCL 是 16 位连续执行型带进位循环左移指令，即每个扫描周期执行一次数据移位运算。将包括进位标志在内的目标操作数[D]的 16 位+1 位（进位标志 M8022）数据向左移动 n 位。最后移出的位值存于进位标志 M8022 中。

因为循环回路中有进位标志，所以如果执行循环移位指令之前 M8022 就先 ON 或 OFF，则会被送入目标操作数中。

若操作数[D]为 KnY,KnM,KnS 等位数指定软元件时，则只有 K4Y,K4M,K4S 有效。

操作数：

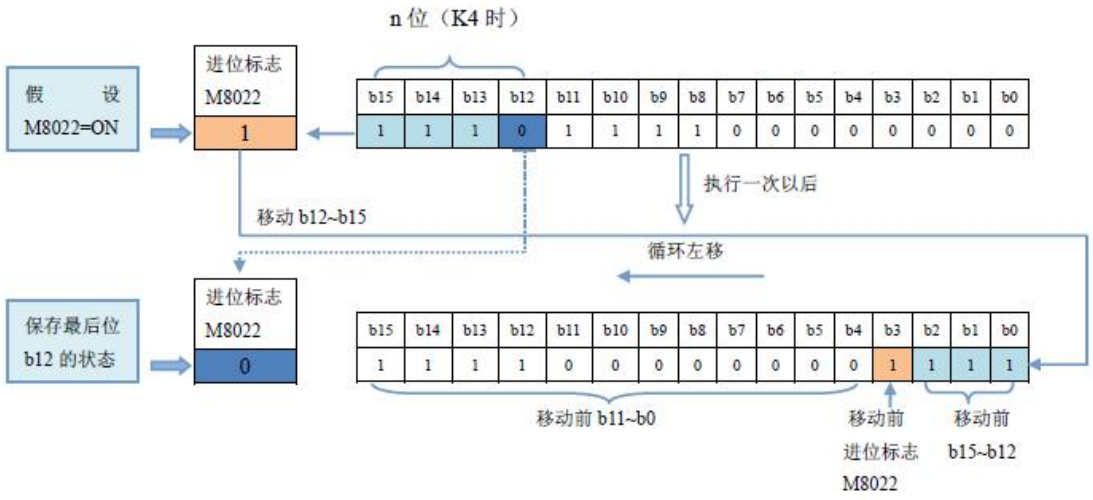
D：KnY, KnM, KnS, T,C,D,Z,V, LV,DT

n ： D,K,H

指令格式：[RCL D n]

编程示例：

当上升沿触点 X0 闭合时，RCL 指令执行一次，将 D0 的 16 个数据位+进位标志位 M8022 向左移 4 位。



RCLP

指令说明：

RCRP 是 16 位脉冲执行型带进位循环左移指令，即指令激活一次，执行一次数据移位运算。将包括进位标志在内的目标操作数[D]的 16 位+1 位（进位标志 M8022）数据向左移动 n 位。最后移出的位值存于进位标志 M8022 中。

因为循环回路中有进位标志，所以如果执行循环移位指令之前 M8022 就先 ON 或 OFF，则会被送入目标操作数中。

若操作数[D]为 KnY,KnM,KnS 等位数指定软元件时，则只有 K4Y,K4M,K4S 有效。

操作数：

D：KnY, KnM, KnS, T,C,D,Z,V, LV,DT

n ： D,K,H

指令格式：[RCLP D n]

编程示例：

当 X0 闭合时，置位 M100，M100 的上升沿触点触发，将 1 传送到 K4Y0。M100 的常开触点闭合，M8013 触点接通 0.5 秒，断开 0.5 秒，使得 RCLP 指令以 1S 为周期执行一次，

让 K4Y0 的 16 个数据位加进位标志位 M8022 向左移动 1 位。

K4Y0 数据位的变化情况：

假设执行前：M8022=OFF

进位	Y17	Y16	Y15	Y14	Y13	Y12	Y11	Y10	Y7	Y6	Y5	Y4	Y3	Y2	Y1	Y0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

执行 1 次：M8022=OFF

进位	Y17	Y16	Y15	Y14	Y13	Y12	Y11	Y10	Y7	Y6	Y5	Y4	Y3	Y2	Y1	Y0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0

执行 15 次：M8022=OFF

进位	Y17	Y16	Y15	Y14	Y13	Y12	Y11	Y10	Y7	Y6	Y5	Y4	Y3	Y2	Y1	Y0
0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

执行 16 次 M8022=ON

进位	Y17	Y16	Y15	Y14	Y13	Y12	Y11	Y10	Y7	Y6	Y5	Y4	Y3	Y2	Y1	Y0
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

执行 17 次 M8022=OFF

进位	Y17	Y16	Y15	Y14	Y13	Y12	Y11	Y10	Y7	Y6	Y5	Y4	Y3	Y2	Y1	Y0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

DRCL

指令说明：

DRCL 是 32 位连续执行型带进位循环左移指令，即每个扫描周期执行一次数据移位运算。将包括进位标志在内的目标操作数[D+1,D]的 32 位+1 位（进位标志 M8022）数据向左移动 n 位。最后移出的位值存于进位标志 M8022 中。

因为循环回路中有进位标志，所以如果执行循环移位指令之前 M8022 就先 ON 或 OFF，则会被送入目标操作数中。

若操作数[D]为指定位数软元件，则只有 K8Y, K8M, K8S 有效。

操作数：

D：KnY, KnM, KnS, T,C,D,Z,V, LV,DT

n ： D,K,H

指令格式：[DRCL D n]

编程示例：

上升沿触点 X0 闭合，DRCL 指令执行一次，将 D1D0 的 32 位数据加进位标志位向左移动四位。

DRCLP

指令说明：

DRCLP 是 32 位脉冲执行型带进位循环左移指令，即指令激活一次，执行一次数据移位运算。将包括进位标志在内的目标操作数[D+1,D]的 32 位+1 位（进位标志 M8022）数据向左移动 n 位。最后移出的位值存于进位标志 M8022 中。

因为循环回路中有进位标志，所以如果执行循环移位指令之前 M8022 就先 ON 或 OFF，则会被送入目标操作数中。

若操作数[D]为指定位数软元件，则只有 K8Y, K8M, K8S 有效。

操作数：

D：KnY, KnM, KnS, T,C,D,Z,V, LV,DT

n ： D,K,H

指令格式：[DRCLP　D　n]

编程示例：

当 X0 闭合，置位 M100。M100 的上升沿触点触发一次，将 K1 传送到 K8Y0，M100 的常开触点闭合，M8013 以 1S 为周期断开闭合，所以 DRCRP 指令 1S 执行一次，将 K8Y0 的 32 位数据加进位标志位 M8022 向左移动 1 位。

SFTR

指令说明：

SFTR 是 16 位连续执行型位右移指令，即每个扫描周期都执行移位运算。是将[D]起始的 n1 位数据右移 n2 位。移位后，低位溢出的数据删除，高位缺失的 n2 位由[S]起始的 n2 位补上。

由于 SFTR 指令具有连续执行型，所以 SFTR 指令前面的触点接通都是以脉冲（上升沿或下降沿）的形式。

补位软元件[S]和移位软元件[D]重复时，发生运算错误。

操作数：

S：X,Y,M,S,@

D：Y,M,S,@

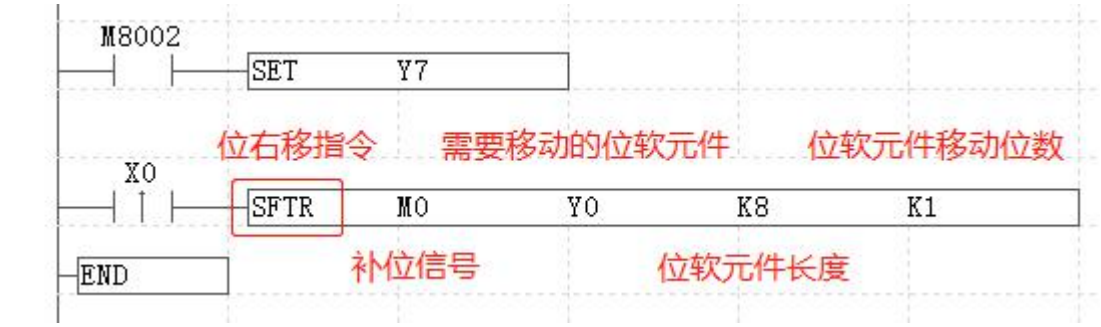
n1：K,H,@,D,C,T,V,Z

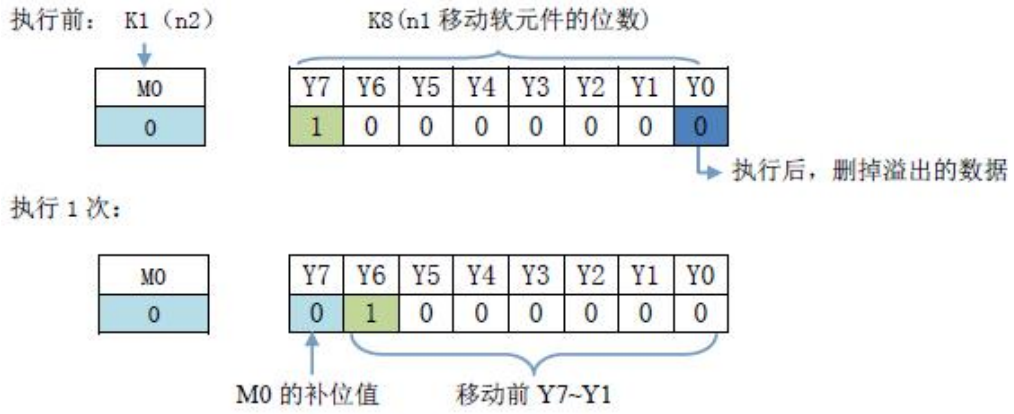
n2：K,H,@,D,C,T,V,Z

指令格式：[SFTR　S　D　n1　n2]

编程示例：

开机上电，M8002 导通一次，置位 Y7，当 X0 的上升沿触点触发，SFTR 指令执行一次，将 Y0 开始的 8 个位右移一位，低位溢出的位值被删除，高位缺失的位值由 M0 补上。





SFTRP

指令说明：

SFTRP 是 16 位脉冲执行型位右移指令，即指令激活一次，执行一次移位运算。是将[D]起始的 n1 位数据右移 n2 位。移位后，低位溢出的数据删除，高位缺失的 n2 位由[S]起始的 n2 位补上。

补位软元件[S]和移位软元件[D]重复时，发生运算错误。

操作数：

S：X,Y,M,S,@

D：Y,M,S,@

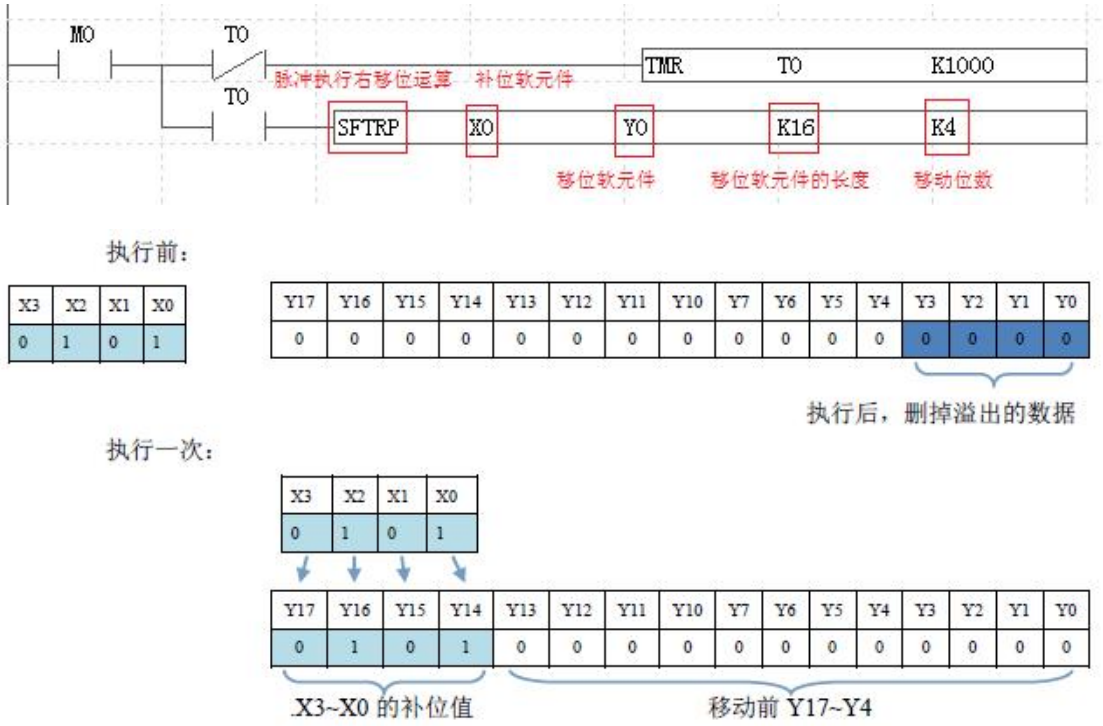
n1：K,H,@ ,D,C,T,V,Z

n2：K,H,@ ,D,C,T,V,Z

指令格式：[SFTRP　S　D　n1　n2]

编程示例：

假设 X0 和 X2 为 ON，X1 和 X3 为 OFF，当 M0 导通，T0 开始计时，1S 后 T0 的常闭触点断开，T0 重新计时。T0 的常开触点闭合，SFTRP 指令执行一次，将 Y0 开始的 16 个位右移 4 位，删除低位溢出的位值，高位缺失的位值由 X0 起始的 4 位补上。



SFTL

指令说明：

SFTL 是 16 位连续执行型位左移指令，即每个扫描周期都执行移位运算。是将[D]起始的 n1 位数据左移 n2 位。移位后，高位溢出的数据删除，低位缺失的 n2 位由[S]起始的 n2 位补上。

由于 SFTL 指令具有连续执行型，所以 SFTL 指令前面的触点接通都是以脉冲（上升沿或下降沿）的形式。

补位软元件[S]和移位软元件[D]重复时，发生运算错误。

操作数：

S：X,Y,M,S,@

D：Y,M,S,@

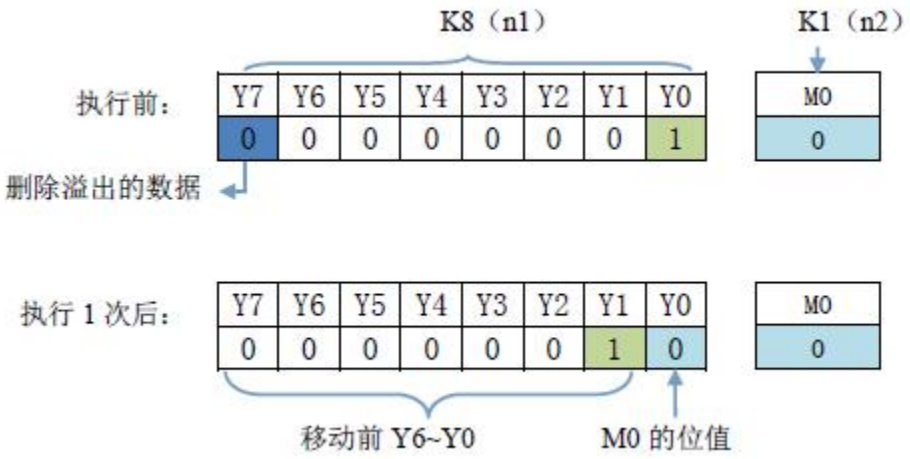
n1：K,H,@ ,D,C,T,V,Z

n2：K,H,@ ,D,C,T,V,Z

指令格式：[SFTL　S　D　n1　n2]

编程示例：

控制器上电运行，M8002 导通一次，置位 Y0，当 X0 的上升沿触点触发，SFTL 指令执行一次，将 Y0 开始的 8 个位左移一位，高位溢出的位值被删除，低位缺失的位值由 M0 补上。



SFTLP

指令说明：

SFTLP 是 16 位脉冲执行型位左移指令，即指令激活一次，执行一次移位运算。是将[D]起始的 n1 位数据左移 n2 位。移位后，高位溢出的数据删除，低位缺失的 n2 位由[S]起始的 n2 位补上。

补位软元件[S]和移位软元件[D]重复时，发生运算错误。

操作数：

S：X,Y,M,S,@

D：Y,M,S,@

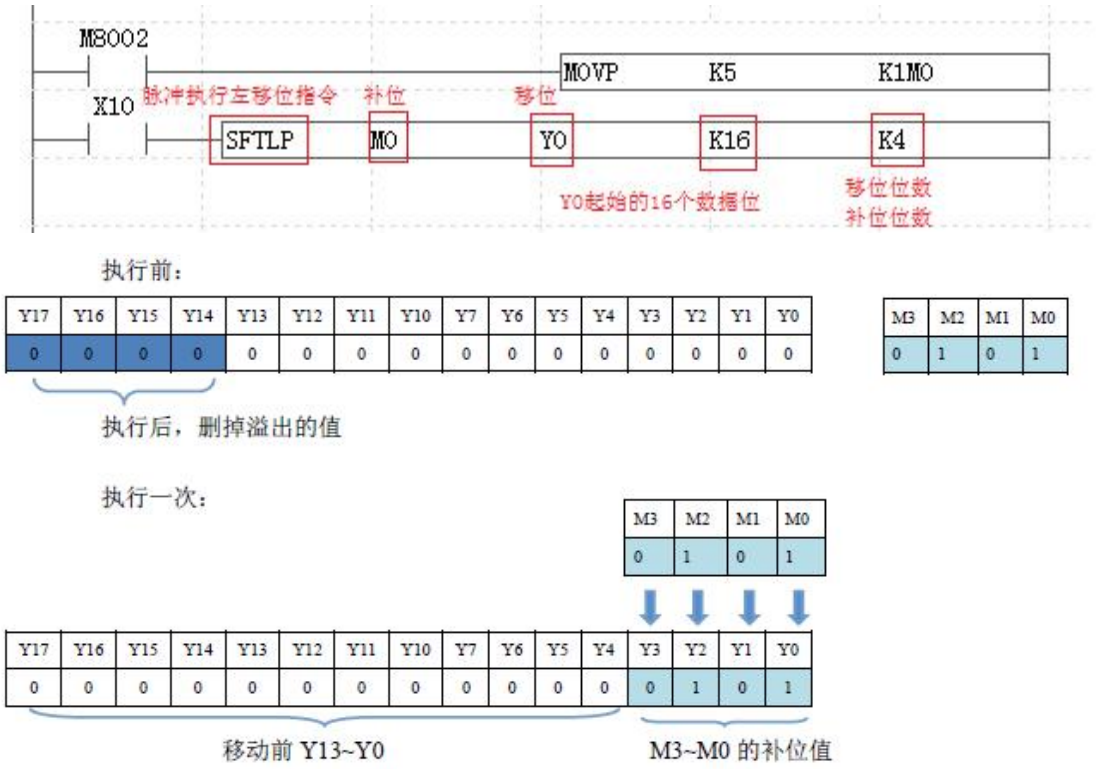
n1：K,H,@ ,D,C,T,V,Z

n2：K,H,@ ,D,C,T,V,Z

指令格式：[SFTLP　S　D　n1　n2]

编程示例：

控制器上电运行，M8002 导通一次，将 K5 传送到 K4M0，M0 和 M2 为 ON，M1 和 M3 为 OFF，SFTLP 指令执行一次，将 Y0 开始的 16 个位左移 4 位，删除高位溢出的位值,低位缺失的位值由 M0 起始的 4 位补上。



WSFR

指令说明：

WSFR 是 16 位连续执行型字右移指令，即每个扫描周期都执行数据移动运算。是将[D]起始的 n1 个字软元件右移 n2 个字。移动后，低位溢出的数据删除，高位缺失的 n2 个软元件的数据由[S]起始的 n2 个软元件的数据补上。

操作数：

S：KnX,KnY, KnM, KnS,T,C,D,LV,DT,@

D：KnY, KnM, KnS,T,C,D,LV,DT

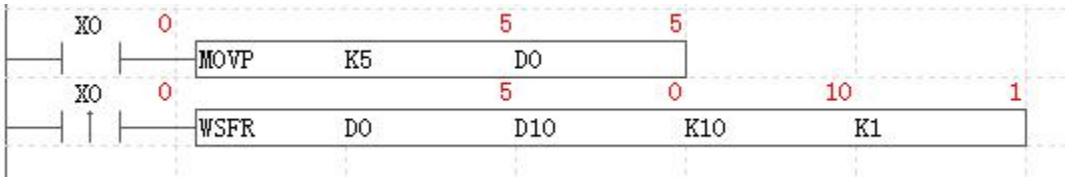
n1：K,H,@ ,D,C,T,V,Z

n2：K,H,@ ,D,C,T,V,Z

指令格式：[WSFR　S　D　n1　n2]

编程示例：

当 X0 导通时，MOVP 指令将 K5 传送到 D0。当 X0 的上升沿触点触发，WSFR 指令执行一次，将 D10 起始的 10 个字软元件向右移动，低位 D10 溢出的数值删除，高位 D19 的值由 D0 的值补上。



执行前:D0, D10~D19 的数据情况

D0	D19	D18	D17	D16	D15	D14	D13	D12	D11	D10
5	0	0	0	0	0	0	0	0	0	0

执行 1 次:D0, D10~D19 的数据情况

D0	D19	D18	D17	D16	D15	D14	D13	D12	D11	D10
5	5	0	0	0	0	0	0	0	0	0

执行 10 次：D0 ,D10~D19 的数据情况

D0	D19	D18	D17	D16	D15	D14	D13	D12	D11	D10
5	5	5	5	5	5	5	5	5	5	5

WSFRP

指令说明：

WSFRP 是 16 位脉冲执行型字右移指令，即指令激活一次，执行一次数据移动运算。是将[D]起始的 n1 个字软元件右移 n2 个字。移动后，低位溢出的数据删除，高位缺失的 n2 个软元件的数据由[S]起始的 n2 个软元件的数据补上。

操作数：

S：KnX,KnY, KnM, KnS,T,C,D,LV,DT,@

D：KnY, KnM, KnS,T,C,D,LV,DT

n1：K,H,@ ,D,C,T,V,Z

n2：K,H,@ ,D,C,T,V,Z

指令格式：[WSFRP　S　D　n1　n2]

编程示例：

当 X0 闭合时，WSFRP 指令执行一次，将 K1Y0 起始的 4 个字软元件（位数指定）右移 2 个字，低位溢出的字软元件数据删掉，高位缺失的 2 个字软元件的数据由 K1X0 起始的 2 个字软元件补上。

K1X0~K1X4，K1Y0~K1Y14 的位值变化情况

执行前：

X7	X6	X5	X4	X3	X2	X1	X0
1	0	1	0	0	1	0	1

Y17	Y16	Y15	Y14	Y13	Y12	Y11	Y10	Y7	Y6	Y5	Y4	Y3	Y2	Y1	Y0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

执行后，溢出的数据删掉

执行一次后：

X7	X6	X5	X4	X3	X2	X1	X0
1	0	1	0	0	1	0	1

Y17	Y16	Y15	Y14	Y13	Y12	Y11	Y10	Y7	Y6	Y5	Y4	Y3	Y2	Y1	Y0
1	0	1	0	0	1	0	1	0	0	0	0	0	0	0	0

移动前 K1X4~K1X0 的补位值　　移动前 K1Y14~K1Y10 的位值

WSFL

指令说明：

WSFL 是 16 位连续执行型字左移指令，即每个扫描周期都执行数据移动运算。是将[D]起始的 n1 个字软元件左移 n2 个字。移动后，高位溢出的数据删除，低位缺失的 n2 个软元件的数据由[S]起始的 n2 个软元件的数据补上。

操作数：

S：KnX,KnY, KnM, KnS,T,C,D,LV,DT,@

D：KnY, KnM, KnS,T,C,D,LV,DT

n1：K,H,@ ,D,C,T,V,Z

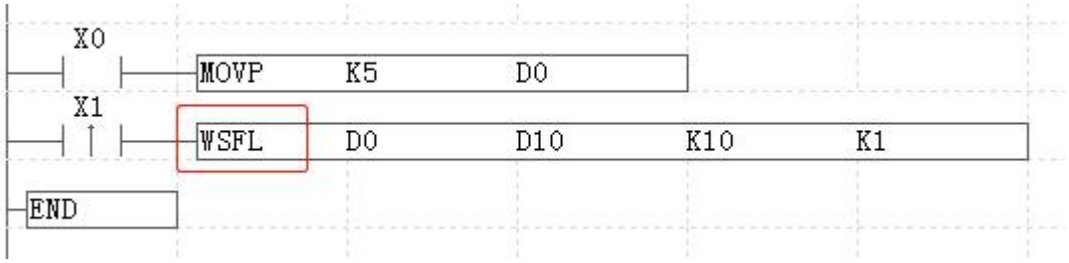
n2：K,H,@ ,D,C,T,V,Z

指令格式：[WSFL　S　D　n1　n2]

编程示例：

当 X0 导通时，MOVP 指令将 K5 传送到 D0。

当 X1 的上升沿触点触发，WSFL 指令执行一次，将 D10 起始的 10 个字软元件向左移动 1 个字，高位 D19 溢出的数值删除，低位 D10 的值由 D0 的值补上。



执行前：D0，D10~D19 的数据情况

D19	D18	D17	D16	D15	D14	D13	D12	D11	D10
0	0	0	0	0	0	0	0	0	0

D0
5

执次 1 次：D0，D10~D19 的数据情况

D19	D18	D17	D16	D15	D14	D13	D12	D11	D10
0	0	0	0	0	0	0	0	0	5

D0
5

执次 10 次：D0，D10~D19 的数据情况

D19	D18	D17	D16	D15	D14	D13	D12	D11	D10
5	5	5	5	5	5	5	5	5	5

D0
5

WSFLP

指令说明：

WSFLP 是 16 位脉冲执行型字左移指令，即指令激活一次，执行一次数据移动运算。是将[D]起始的 n1 个字软元件左移 n2 个字。移动后，高位溢出的数据删除，低位缺失的 n2 个软元件的数据由[S]起始的 n2 个软元件的数据补上。

操作数：

S：KnX,KnY, KnM, KnS,T,C,D,LV,DT,@

D：KnY, KnM, KnS,T,C,D,LV,DT

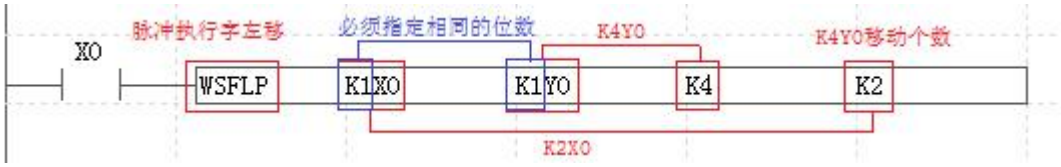
n1：K,H,@ ,D,C,T,V,Z

n2：K,H,@ ,D,C,T,V,Z

指令格式：[WSFLP　S　D　n1　n2]

编程示例：

当 X0 闭合时，WSFLP 指令执行一次，将 K1Y0 起始的 4 个字软元件（位数指定）左移 2 个字，高位溢出的字软元件数据删掉，低位缺失的 2 个字软元件的数据由 K1X0 起始的 2 个字软元件补上。



K2X0，K4Y0 的位值变化情况

执行前：

X7	X6	X5	X4	X3	X2	X1	X0
1	0	1	0	0	1	0	1

Y17	Y16	Y15	Y14	Y13	Y12	Y11	Y10	Y7	Y6	Y5	Y4	Y3	Y2	Y1	Y0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

执行后，溢出的数据删掉

执行一次后：

X7	X6	X5	X4	X3	X2	X1	X0
1	0	1	0	0	1	0	1

Y17	Y16	Y15	Y14	Y13	Y12	Y11	Y10	Y7	Y6	Y5	Y4	Y3	Y2	Y1	Y0
0	0	0	0	0	0	0	0	1	0	1	0	0	1	0	1

移动前 K2Y0 的位值　　移动前 K2X0 的补位值

SFWR

指令说明：

SFWR 是 16 位连续执行型移位写入指令，即每个扫描周期都执行数据移位写入运算。将 S 的值写入由 D 地址起始，个数为 n 的“先进先出”队列中，以第一个编号装置作为指针，当指令执行时，指针内容值先加 1，之后 S 所指定的装置其内容值会写入先入先出 D 数据串列中由指针所指定的位置。

若队列数据已满，则不处理后来的数据，且标志位 M8022 置 ON。

操作数：

S：KnX,KnY, KnM, KnS,T,C,D,LV,DT,@

D：KnY, KnM, KnS,T,C,D, LV,DT

n：K,H,@ ,D,C,T,V,Z

指令格式：[SFWR　S　D　n]

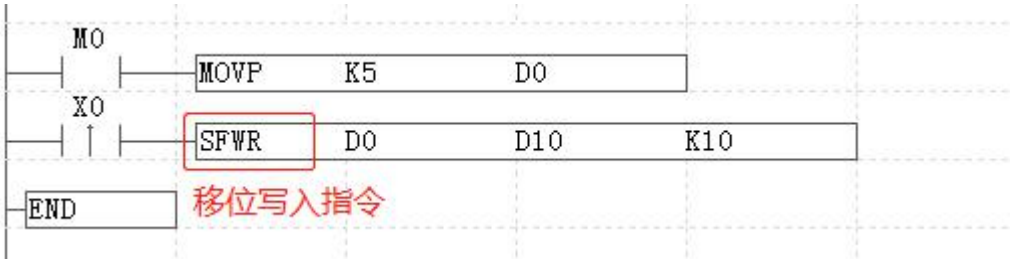
编程示例：

当 M0 闭合，MOVP 指令把 K5 传送到 D0，D0 为 5，D10 作为指针，指令执行一次就从 D11 开始依次写入 D0 数据；

当 X0 闭合 1 次，指针 D10 的值变为 1，D0 的数据写入 D11，D11 的值变为 5；

当 X0 闭合 2 次，指针 D10 的值变为 2，D0 的数据写入 D12，D12 的值变为 5；

当 X0 闭合 9 次，指针 D10 的值变为 9，D0 的数据写入 D19，D19 的值变为 5。



X0 闭合 9 次后，查询寄存器的值如下，若之后 X0 再闭合，D 数据值不会改变。

执行前：D10~D19 的数据变化情况

D0									指针
5	D19	D18	D17	D16	D15	D14	D13	D12	D11	D10
	0	0	0	0	0	0	0	0	0	0

执行 1 次：D10~D19 的数据变化情况

D0									指针
5	D19	D18	D17	D16	D15	D14	D13	D12	D11	D10
	0	0	0	0	0	0	0	0	5	1

执行 2 次：D10~D19 的数据变化情况

D0									指针
5	D19	D18	D17	D16	D15	D14	D13	D12	D11	D10
	0	0	0	0	0	0	0	5	5	2

执行 9 次：D10~D19 的数据变化情况

D0									指针
5	D19	D18	D17	D16	D15	D14	D13	D12	D11	D10
	5	5	5	5	5	5	5	5	5	9

SFWRP

指令说明：

SFWRP 是 16 位脉冲执行型移位写入指令，即指令激活一次，执行一次移位写入运算。将 S 的值写入由 D 地址起始，个数为 n 的“先进先出”队列中,以第一个编号装置作为指针，

当指令执行时，指针内容值先加 1，之后 S 所指定的装置其内容值会写入先入先出 D 数据串列中由指针所指定的位置。

若队列数据已满，则不处理后来的数据，且标志位 M8022 置 ON。

操作数：

S：KnX,KnY, KnM, KnS,T,C,D,LV,DT,@

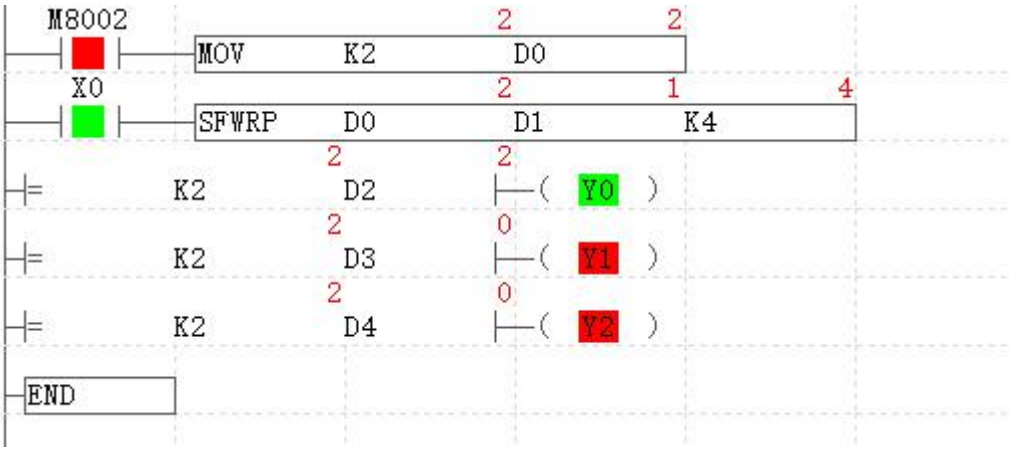
D：KnY, KnM, KnS,T,C,D, LV,DT

n：K,H,@ ,D,C,T,V,Z

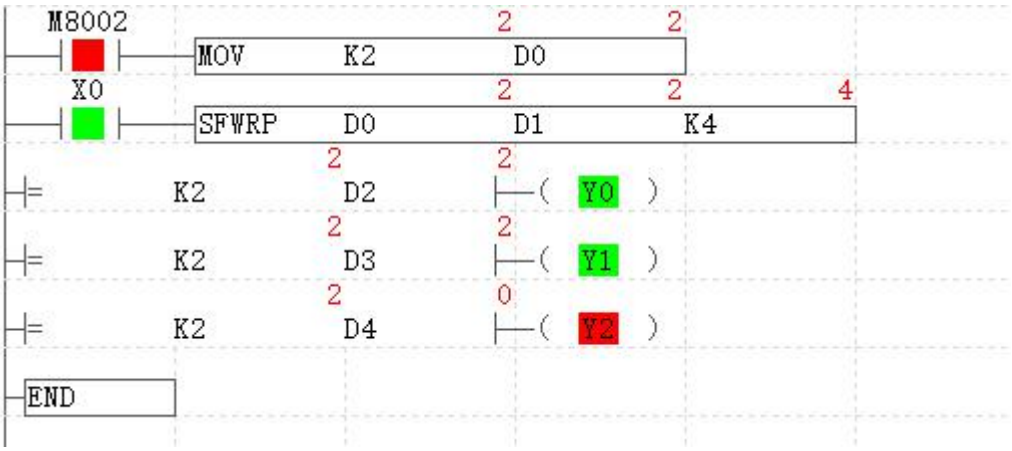
指令格式：[SFWRP　S　D　n]

编程示例：

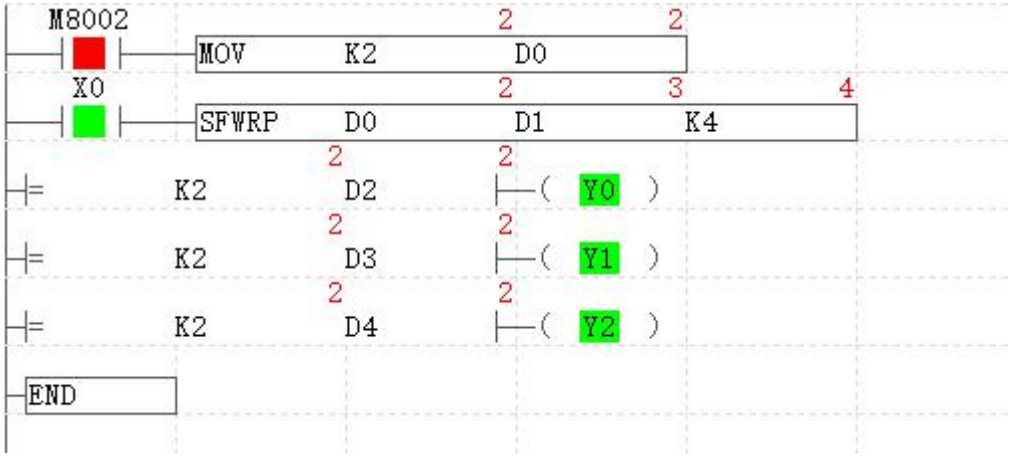
当 X0 断开接通 1 次，SFWRP 指令执行，指针 D1 的值变为 1，将 K2 写入到 D2，D2 的值为 2，第一个比较触点闭合，Y0 输出；



当 X0 第二次接通，SFWRP 指令执行，指针 D1 的值变为 2，将 K2 写入到 D3，D3 的值为 2，第二个比较触点闭合，Y1 输出；



当 X0 第三次接通，SFWRP 指令执行，指针 D1 的值变为 3，将 K2 写入到 D4，D4 的值为 2，第三个比较触点闭合，Y2 输出。



SFRD

指令说明：

SFRD 是 16 位连续执行型移位读出指令，即每个扫描周期都执行数据移位读出运算。即从“先进先出”队列 S 的首项读出到 D 中，然后将队列 S 逐字右移 1 个字，将队列指针递减。以第一个编号装置作为指针，当指令执行时，指针内容值先减 1，之后 S 所指定的装置其内容值会写入先入先出 D 数据串列中由指针所指定的位置。

若指针已经为 0，则指令不处理前述操作，而 0 标志 M8020 会置 1。

※ 由于使用连续执行型指令 SFRD，每个运算周期都移位，因此请使用脉冲执行型指令 SFRDP 编程

操作数：

S：KnY, KnM, KnS,T,C,D, LV,DT,@

D：KnY, KnM, KnS,T,C,D, Z,V,LV,DT

n：K,H,@ ,D,C,T,V,Z

指令格式：[SFRD S D n]

编程示例：

控制器上电运行，成批复位 D0~D20。

写入 1 个值到 D0，上升沿触点 X0 接通 1 次，SFWR 移位写入指令执行一次，然后 D0 值传送到 D11，以此类推，再操作 8 次，8 个不同的值写入到 D12~D19，D10 作为指针记录操作次数。

当上升沿触点 X1 接通 1 次时，SFRD 移位读出指令执行一次，按照先进先出的原则，D11~D19 的值依次传送到 D1，上一个数据传送后，该寄存器被下一个数据占领。操作 9 次后，D10 变为 0，D11~D99 皆为最后位输入的值。

SFWR 指令执行时，D10~D19 的变化情况

写入 D0=11	写入 D0=22	写入 D0=33	写入 D0=44	写入 D0=55	写入 D0=66	写入 D0=77	写入 D0=88	写入 D0=99
按下第 1 次 X0	按下第 2 次 X0	按下第 3 次 X0	按下第 4 次 X0	按下第 5 次 X0	按下第 6 次 X0	按下第 7 次 X0	按下第 8 次 X0	按下第 9 次 X0
D10=1	D10=2	D10=3	D10=4	D10=5	D10=6	D10=7	D10=8	D10=9
D11=11	D12=22	D13=33	D14=44	D15=55	D16=66	D17=77	D18=88	D19=99

SFRD 指令执行时，D10~D19 的变化情况

按下第 1 次 X1	按下第 2 次 X1	按下第 3 次 X1	按下第 4 次 X1	按下第 5 次 X1	按下第 6 次 X1	按下第 7 次 X1	按下第 8 次 X1	按下第 9 次 X1
读出 D1=11	读出 D1=22	读出 D1=33	读出 D1=44	读出 D1=55	读出 D1=66	读出 D1=77	读出 D1=88	读出 D1=99
D10=8	D10=7	D10=6	D10=5	D10=4	D10=3	D10=2	D10=1	D10=0
D11=22	D11=33	D11=44	D11=55	D11=66	D11=77	D11=88	D11=99	D11=99
D12=33	D12=44	D12=55	D12=66	D12=77	D12=88	D12=99	D12=99	D12=99
D13=44	D13=55	D13=66	D13=77	D13=88	D13=99	D13=99	D13=99	D13=99
D14=55	D14=66	D14=77	D14=88	D14=99	D14=99	D14=99	D14=99	D14=99
D15=66	D15=77	D15=88	D15=99	D15=99	D15=99	D15=99	D15=99	D15=99
D16=77	D16=88	D16=99	D16=99	D16=99	D16=99	D16=99	D16=99	D16=99
D17=88	D17=99	D17=99	D17=99	D17=99	D17=99	D17=99	D17=99	D17=99
D18=99	D18=99	D18=99	D18=99	D18=99	D18=99	D18=99	D18=99	D18=99
D19=99	D19=99	D19=99	D19=99	D19=99	D19=99	D19=99	D19=99	D19=99

SFRDP

指令说明：

SFRDP 是 16 位脉冲执行型移位读出指令，即指令激活一次，执行一次数据移位读出运算。即从"先进先出"队列 S 的首项读出到 D 中，然后将队列 S 逐字右移 1 个字，将队列指针递减。以第一个编号装置作为指针，当指令执行时，指针内容值先减 1，之后 S 所指定的装置其内容值会写入先入先出 D 数据串列中由指针所指定的位置。

若指针已经为 0，则指令不处理前述操作，而 0 标志 M8020 会置 1。

※ 由于使用连续执行型指令 SFRD，每个运算周期都移位，因此请使用脉冲执行型指令 SFRDP 编程

操作数：

S：KnY, KnM, KnS,T,C,D, LV,DT,@

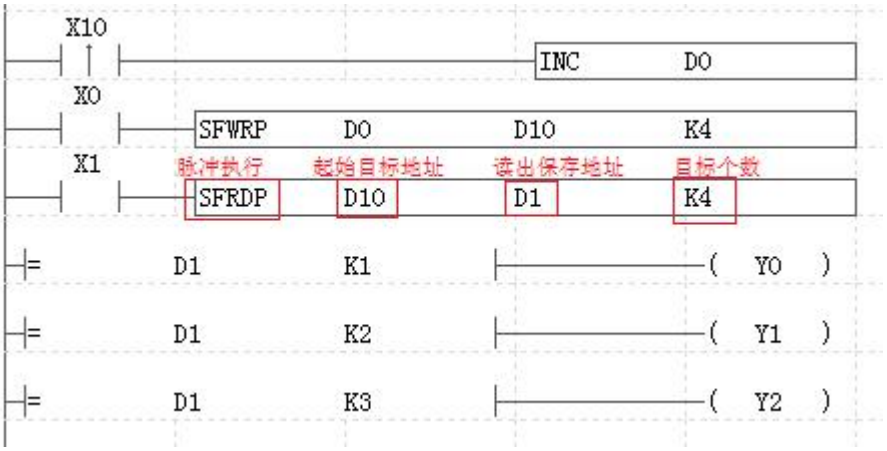
D：KnY, KnM, KnS,T,C,D, Z,V,LV,DT

n：K,H,@ ,D,C,T,V,Z

指令格式：[SFRDP　S　D　n]

编程示例：

X10 接通第一次，D0 自加 1，D0 变为 1，X0 导通一次，将 D0 为 1 的数值传送到 D11，依次类推，X10 和 X0 第二次接通，将 2 传送到 D12，X10 和 X0 第三次接通，3 传送到 D13，然后 X1 导通一次，按照先入先出的原则，将 1,2,3 依次传送到 D1，Y0，Y1，Y2 依次导通。



3.7 数据处理指令

分类	指令	位数	脉冲	功能	操作数类型
成批复位	ZRST	16 位	-	[ZRST　D1　D2] D1 到 D2 之间的软元件全部复位	D1/D2：Y,M,S,T,C,D,V,Z,LV,DT
	ZRSTP		√		
译码	DECO	16 位	-	[DECO　S　D　n] S 的 n 位二进数进行译码，其结果用目标操作数 D 开始的 2^n 个位数置 1 来表示	S：X,Y,M,S,T,C,D,K,H, Z,V,LV,DT D：Y,M,S,T,C,D,LV,DT n：K,H,@ ,D,C,T,V,Z
	DECOP		√		
编码	ENCO	16 位	-	[ENCO　S　D　n] S 开始的 2^n 个位数进行编码，其结果用目标操作数 D 的 n 位二进制数置 1 来表示	S：X,Y,M,S,T,C,D,Z,V, LV,DT D：T,C,D,Z,V,LV, DT n：K,H,@
	ENCOP		√		
ON 位计数	SUM	16 位	-	[SUM　S　D] 对 S 的中为 ON 的位进行计数，计数后的值保存到目标操作数[D]中	S：KnX, KnY, KnM, KnS,T,C,D,K,H,Z,V, LV,DT,@ D：KnY, KnM, KnS, T,C,D,Z,V, LV,DT
	SUMP		√		
	DSUM	32 位	-		
	DSUMP		√		
ON	BON	16 位	-	[BON　S　D　n]	S：KnX, KnY, KnM,

位判定	BONP		√	对 S 数据的 n 位进行 ON 位判定，n 位为 ON 时，目标操作数 D 为 ON，n 位为 OFF 时，目标操作数 D 为 OFF	KnS,T,C,D,K,H,Z,V,LV,DT,@ D：Y, M, S,@ n：K,H,D,@ ,DT
	DBON	32 位	-		
	DBONP		√		
平均值计算	MEAN	16 位	-	[MEAN S D n] 对 S 起始的 n 个软元件的数据相加后的总值，除以 n 得到的平均值，最后保存到 D 中。如果有余数，余数舍去	S：KnX, KnY, KnM, KnS,T,C,D,LV,DT,@ D：KnY, KnM, KnS, T,C, D, Z,V,LV,DT n：D,K,H
	MEANP		√		
	DMEAN	32 位	-		
	DMEANP		√		
信号报警器置位	ANS	16 位	-	[ANS S m D] 指令接通时间超过 S 的判定时间[m×100ms]，置位 D	S：T m：K,H,D,@ D：S
信号报警器复位	ANR	16 位	-	信号报警器复位	无操作数
	ANRP		√		
BIN 开方运算	SQR	16 位	-	[SQR S D] S 开方运算的结果舍去小数点取整数，小数点舍去后生成时，借位标志位 M8021 置 ON	S：D,K,H,LV,DT,@ D：D,LV,DT
	SQRP		√		
	DSQR	32 位	-		
	DSQRP		√		
BIN 转浮点数	FLT	16 位	-	[FLTP S D] S 的 BIN 整数值转换成二进制浮点数（实数）后，保存到[D+1，D]中	S：D,LV,DT,@ D：D,LV,DT,
	FLTP		√		
	DFLT	32 位	-		
	DFLTP		√		
高低字节互换	SWAP	16 位	-	[SWAP S] 将 S 的高八位与低八位数据互换	S：KnX,KnY,KnM, KnS,T,C,D,LV,DT
	SWAPP		√		
	DSWAP	32 位	-		
	DSWAPP		√		

ZRST

指令说明：

ZRST 是 16 位连续执行型成批复位指令，即每个扫描周期都执行一次成批复位运算。是将同一种类的起点操作数[D1]到终点操作数[D2]之间的软元件全部复位。

[D1]的编号必须小于或等于[D2]，如果[D1]的编号大于[D2]，报错，复位不执行。

操作数：

D1：Y, M, S,T,C,D,V,Z,LV,DT

D2：Y, M, S,T,C,D,V,Z,LV,DT

指令格式：[ZRST D1 D2]

编程示例：

当 X0 接通后，位软元件 M0~M20 均被复位成 OFF。

当 X1 接通后，字软元件 D0~D100 均被复位成 0。

当 X2 接通后，字软元件 C0~C10 均被复位成 0。

注意事项：由于 ZRST 具有连续执行型，所以 X0，X1，X2 一直不断开，那么它操作的软元件一直是复位状态。RST 也是复位指令，但一次操作只能复位一个元件。

ZRSTP

指令说明：

ZRSTP 是 16 位脉冲执行型成批复位指令，即指令激活一次，执行一次成批复位运算。是将同一种类的起点操作数[D1]到终点操作数[D2]之间的软元件全部复位。

[D1]的编号必须小于或等于[D2]，如果[D1]的编号大于[D2]，报错，复位不执行。

操作数：

D1：Y, M, S,T,C,D,LV,DT

D2：Y, M, S,T,C,D,LV,DT

指令格式：[ZRSTP　D1　D2]

编程示例：

当 X0 接通后，位软元件 M0~M10 均被复位成 OFF。

当 X1 接通后，字软元件 D0~D10 均被复位成 0。

注意事项：ZRSTP 具有脉冲执行型，就算 X0，X1 不断开，也只对操作的软元件复位一次。

DECO

指令说明：

DECO 是 16 位连续执行型译码指令，即每个扫描周期都执行一次译码运算。是将源操作数[S]的 n 位二进数进行译码，其结果用目标操作数[D]开始的 2^n 个位数置 1 来表示。

目标操作数[D]为位软元件时，n 的取值范围是 1≤n≤8；目标操作数[D]为字软元件时，n 的取值范围是 1≤n≤4，n=0 时不处理，n 在取值范围之外时运算错误标志动作。

操作数：

S：X,Y, M, S,T,C,D,K,H, Z,V,LV,DT

D：Y, M, S,T,C,D, LV,DT

n：K,H,@ ,D,C,T,V,Z

指令格式：[DECO　S　D　n]

编程示例：

当目标地址是位软元件 M0 时，X0 闭合，DECO 指令执行，因为 n 为 3，所以将 D0 二进制数低 3 位的值进行译码后用 M0~M7 的一个位软元件来表示，例如：D0=0，M0=1；D0=1，M1=1；D0=2，M2=1……D0=7，M7=1。

n=K3 时，D0 取三个低位 b0~b2；目标地址为 M0~M7（2^n=8），数据变化如下：

D0 的值	D0		
	b2	b1	b0
0	0	0	0
1	0	0	1
2	0	1	0
3	0	1	1
4	1	0	0
5	1	0	1
6	1	1	0
7	1	1	1

M7	M6	M5	M4	M3	M2	M1	M0
0	0	0	0	0	0	0	1
0	0	0	0	0	0	1	0
0	0	0	0	0	1	0	0
0	0	0	0	1	0	0	0
0	0	0	1	0	0	0	0
0	0	1	0	0	0	0	0
0	1	0	0	0	0	0	0
1	0	0	0	0	0	0	0

DECOP

指令说明：

DECOP 是 16 位脉冲执行型译码指令，即指令激活一次，执行一次译码运算。是将源操作数[S]的 n 位二进数进行译码，其结果用目标操作数[D]开始的 2^n 个位数置 1 来表示。

目标操作数[D]为位软元件时，n 的取值范围是 1≤n≤8；目标操作数[D]为字软元件时，n 的取值范围是 1≤n≤4，n=0 时不处理，n 在取值范围之外时运算错误标志动作。

操作数：

S：X,Y, M, S,T,C,D,K,H, Z,V,LV,DT

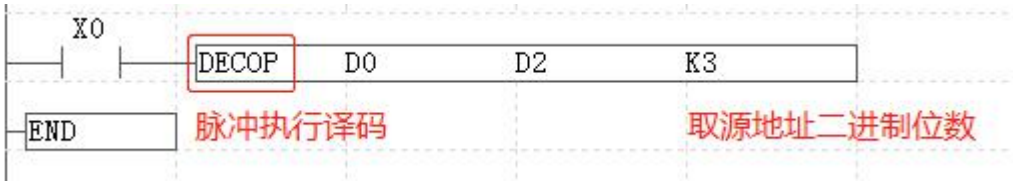
D：Y, M, S,T,C,D, LV,DT

n：K,H,@ ,D,C,T,V,Z

指令格式：[DECOP　S　D　n]

编程示例：

当目标地址是字软元件 D0 时，X0 闭合，DECOP 指令执行，因为 n 为 3，所以将 D0 二进制数低 3 位的值进行译码后用 D2 中的 b7~b0 之间的一个位来表示，例如：当 D0=0 时，D2 的 b0=1，D2=1；当 D0=1 时，D2 的 b1=1，D2=2；当 D0=2 时，D2 的 b2=2，D2=4；…… 当 D0=7 时，D2 的 b7=1，D2=128。



n=K3，源地址 D0 取三个低位 b0~b2；目标地址 D2 为 b0~b7（2^n=8），数据变化如下：

D0 的值	D0		
	b2	b1	b0
0	0	0	0
1	0	0	1
2	0	1	0
3	0	1	1
4	1	0	0
5	1	0	1
6	1	1	0
7	1	1	1

D2								D2 的值
b7	b6	b5	b4	b3	b2	b1	b0	
0	0	0	0	0	0	0	1	1
0	0	0	0	0	0	1	0	2
0	0	0	0	0	1	0	0	4
0	0	0	0	1	0	0	0	8
0	0	0	1	0	0	0	0	16
0	0	1	0	0	0	0	0	32
0	1	0	0	0	0	0	0	64
1	0	0	0	0	0	0	0	128

ENCO

指令说明：

ENCO 是 16 位连续执行型编码指令，即每个扫描周期都执行一次编码运算。是将源操作数[S]开始的 2^n 个位数进行编码，其结果用目标操作数[D]的 n 位二进制数置 1 来表示。

源操作数[S]为位软元件时，n 的取值范围是 1≤n≤8；目标操作数[S]为字软元件时，n 的取值范围是 1≤n≤4，n=0 时不处理，n 在取值范围之外时运算错误标志动作。

[S]的高位为 1 时，忽略低位为 1 的状态。

操作数：

S：X,Y, M, S,T,C,D, Z,V,LV,DT

D：T,C,D, Z,V,LV,DT

n：K,H,@

指令格式：[ENCO　S　D　n]

编程示例：

当源地址是位软元件 M0 时，X0 闭合，ENCO 指令执行，n=K3 时，将 M0 的 2^n 个位数 M7~M0 置 1 的状态进行编码后传送到 D0 的低 3 位 b2~b0 中。

例如 M7=1 时，转换成二进制数的低 3 位 111 传送到 D0 的低 3 位 b2~b0 中，D0 的 b2=1,b1=1,b0=1，D0 为 7。例如 M5=1 时，转换成二进制的低 3 位 101 传送到 D0 的低 3 位 b2~b0 中，所以 b2=1,b1=0,b0=1，D0 为 5。如果 M7 与 M5 都为 1 时，只计算高位 M7 的 ON 位。

ENCOP

指令说明：

ENCOP 是 16 位脉冲执行型编码指令，即指令激活一次，执行一次编码运算。是将源操作数[S]开始的 2^n 个位数进行编码，其结果用目标操作数[D]的 n 位二进制数置 1 来表示。

源操作数[S]为位软元件时，n 的取值范围是 1≤n≤8；目标操作数[S]为字软元件时，n 的取值范围是 1≤n≤4，n=0 时不处理，n 在取值范围之外时运算错误标志动作。

[S]的高位置 1 时，忽略低位置 1 的状态。

操作数：

S：X,Y, M, S,T,C,D, Z,V,LV,DT

D：T,C,D, Z,V,LV,DT

n：K,H,@

指令格式：[ENCOP　S　D　n]

编程示例：

当源地址是字软元件 D2 时，X0 闭合，ENCOP 指令执行，n=K3 时，将 D0 的 2^n 个位数 b7~b0 置 1 的状态进行编码后传送到 D2 的低 3 位 b2~b0 中。

例如 D0 的 b7 位为 1 时（此时 b6~b0 可忽略），进行编码转换成二进制数的低 3 位 111，然后传送到 D2 的低 3 位 b2~b0 中，D2 的 b2=1,b1=1,b0=1，D2 为 7。例如 D0 的 b5 位为 1 时，进行编码转换成二进制数的低 3 位 101 传送到 D2 的低 3 位 b2~b0 中，所以 b2=1,b1=0,b0=1，D2 为 5。如果 D0 的 b7 位与 b5 都为 1 时，只运算高位 b7 的位，忽略低位 b5 的位。

SUM

指令说明：

SUM 是 16 位连续执行型 ON 位计数指令，即每个扫描周期都执行一次 ON 位计数运算。是将源操作数[S]的 16 位数据中为 ON 的位进行计数，计数后的值保存到目标操作数[D]中。

源操作数[S]为 0 时，零位标志位 M8020 为 ON。

操作数：

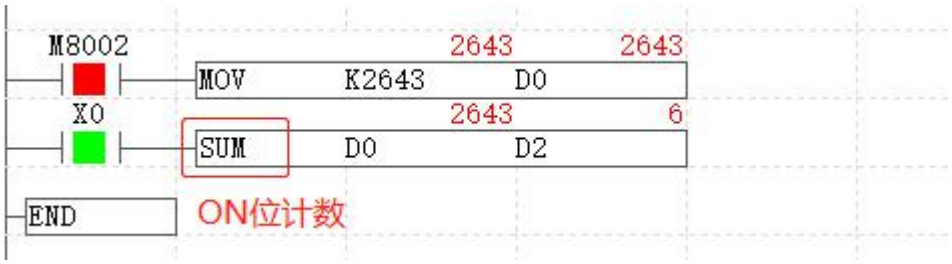
S：KnX, KnY, KnM, KnS,T,C,D,K,H,Z,V,LV,DT,@

D：KnY, KnM, KnS, T,C,D,Z,V, LV,DT

指令格式：[SUM　S　D]

编程示例：

假设 D0 的值为 K2643，X0 闭合，SUM 指令执行，将 D0 为 K2643 的 16 位二进制数 0000,1010,0101,0011 中为 ON 的位进行计数，计数完成后得到 K6，将 K6 保存到 D2 中。



D0，D2 数据变化如下：

D0	b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
K2643	0	0	0	0	1	0	1	0	0	1	0	1	0	0	1	1
计数					6	←	5	←		4	←	3	←		2	← 1

D2	b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
K6	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0

SUMP

指令说明：

SUMP 是 16 位脉冲执行型 ON 位计数指令，即指令激活一次，执行一次 ON 位计数运算。是将源操作数[S]的 16 位数据中为 ON 的位进行计数，计数后的值保存到目标操作数[D]中。

源操作数[S]为 0 时，零位标志位 M8020 为 ON。

操作数：

S：KnX, KnY, KnM, KnS,T,C,D,K,H,Z,V,LV,DT,@

D：KnY, KnM, KnS, T,C,D,Z,V, LV,DT

指令格式：[SUMP　S　D]

编程示例：

假设 D0 的值为 K3737，X0 闭合，SUMP 指令执行一次，将 D0 为 K3737 的 16 位二进制数 0000,1110,1001,1001 中为 ON 的位进行计数，计数完成后得到 K7，将 K7 保存到 K4M0 中，M0~M2 为 ON。

D0，K4M0 数据变化如下：

D0	b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
K3737	0	0	0	0	1	1	1	0	1	0	0	1	1	0	0	1
计数					7 ←	6 ←	5 ←		4 ←			3 ←	2 ←		1	

D2	M15	M14	M13	M12	M11	M10	M9	M8	M7	M6	M5	M4	M3	M2	M1	M0
K7	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1

DSUM

指令说明：

DSUM 是 32 位连续执行型 ON 位计数指令，即每个扫描周期都执行一次 ON 位计数运算。是将源操作数[S+1,S]的 32 位数据中为 ON 的位进行计数，计数后的值保存到目标操作数[D]中，[D+1]中保存 K0。

源操作数[S+1,S]为 0 时，零位标志位 M8020 为 ON。

操作数：

S：KnX, KnY, KnM, KnS,T,C,D,K,H,Z,V,LV,DT,@

D：KnY, KnM, KnS, T,C,D,Z,V, LV,DT

指令格式：[DSUM　S　D]

编程示例：

假设 D1D0 的值为 K56940，X0 闭合，DSUM 指令连续执行，将 D1D0 为 K56940 的 32 位二进制数 0000,0000,0000,0000,1101,1110,0110,1100 中为 ON 的位进行计数，计数完成后得到 K10，将 K10 保存到 D2 中。

DSUMP

指令说明：

DSUMP 是 32 位脉冲执行型 ON 位计数指令，即指令激活一次，执行一次 ON 位计数

运算。是将源操作数[S+1,S]的 32 位数据中为 ON 的位进行计数，计数后的值保存到目标操作数[D]中，[D+1]中保存 K0。

源操作数[S+1,S]为 0 时，零位标志位 M8020 为 ON。

操作数：

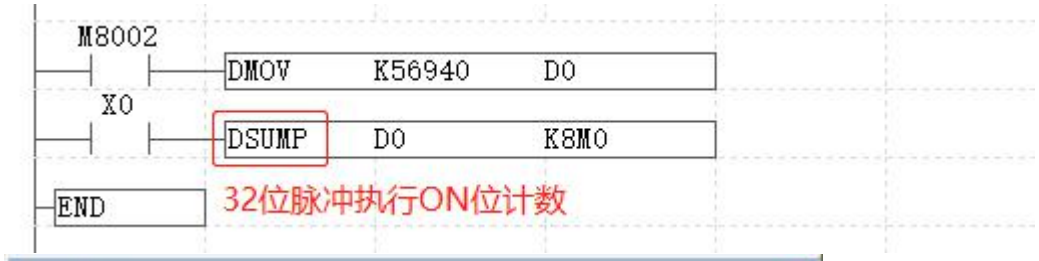
S：KnX, KnY, KnM, KnS,T,C,D,K,H,Z,V,LV,DT,@

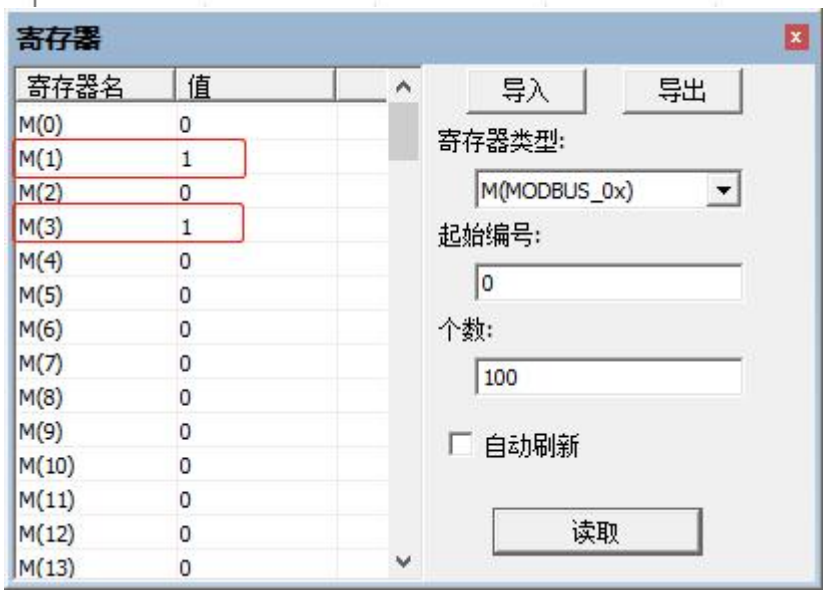
D：KnY, KnM, KnS, T,C,D,Z,V, LV,DT

指令格式：[DSUMP　S　D]

编程示例：

假设 D1D0 的值为 K56940，X0 闭合，DSUMP 指令执行一次，将 D1D0 为 K56940 的 32 位二进制数 0000,0000,0000,0000,1101,1110,0110,1100 中为 ON 的位进行计数，计数完成后得到 K10，将 K10 保存到 K8M0 中，M3 和 M1 为 ON。





BON

指令说明：

BON 是 16 位连续执行型 ON 位判定指令，即每个扫描周期都执行一次 ON 位判定。是将源操作数[S]的 16 位数据的第 n 位进行 ON 位判定，n 位为 ON 时，目标操作数[D]为 ON，n 位为 OFF 时，目标操作数[D]为 OFF。

若源操作数[S]指定了常数 K 时，会自动进行 BIN 转换。

n 的取值范围在：K0~K15

操作数：

S：KnX, KnY, KnM, KnS,T,C,D,K,H,Z,V,LV,DT,@

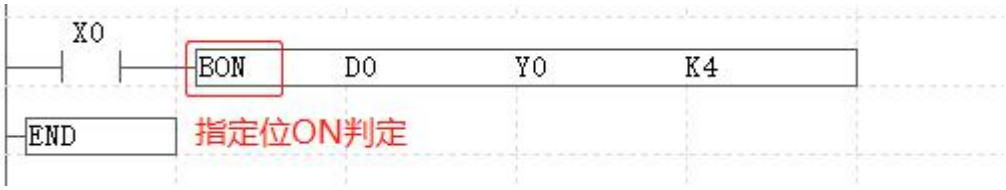
D：Y, M, S,@

n：K,H,D,@,DT

指令格式：[BON　S　D　n]

编程示例：

假设 D0 的值为 K21，X0 闭合，BON 指令执行，将 D0 为 K21 的 16 位二进制数 0000,0000,0001,0101 中的 b4 位进行 ON 位判定，b4 为 1，所以 Y0 为 ON。



BONP

指令说明：

BONP 是 16 位脉冲执行型 ON 位判定指令，即指令激活一次，执行一次 ON 位判定。是将源操作数[S]的 16 位数据的第 n 位进行 ON 位判定，n 位为 ON 时，目标操作数[D]为 ON，n 位为 OFF 时，目标操作数[D]为 OFF。

若源操作数[S]指定了常数 K 时，会自动进行 BIN 转换。

n 的取值范围在：K0~K15

操作数：

S：KnX, KnY, KnM, KnS,T,C,D,K,H,Z,V,LV,DT,@

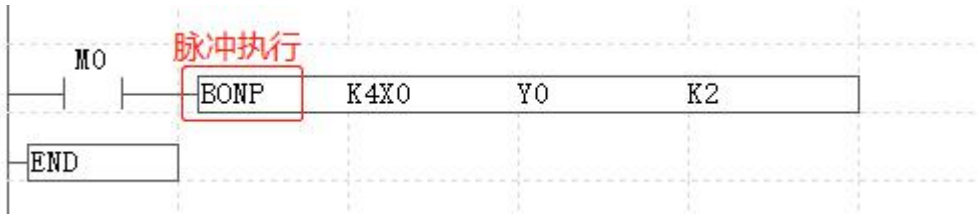
D：Y, M, S,@

n：K,H,D,@ ,DT

指令格式：[BON　S　D　n]

编程示例：

假设 X0~X2 都是 ON 状态，则 K4X0 的值为 K7，M0 闭合，BONP 指令执行一次，将 K4X0 中的 X2 位进行 ON 位判定，X2 为 ON，所以 Y0 为 ON。



DBON

指令说明：

DBON 是 32 位连续执行型 ON 位判定指令，即每个扫描周期都执行一次 ON 位判定。是将源操作数[S+1,S]的 32 位数据的第 n 位进行 ON 位判定，n 位为 ON 时，目标操作数[D]为 ON，n 位为 OFF 时，目标操作数[D]为 OFF。

若源操作数[S+1，S]指定了常数 K 时，会自动进行 BIN 转换。

n 的取值范围在：K0~K31

操作数：

S：KnX, KnY, KnM, KnS,T,C,D,K,H,Z,V,LV,DT,@

D：Y, M, S@

n：K,H,D,@

指令格式：[DBON　S　D　n]

编程示例：

假设 D1D0 的值为 K179210，X0 闭合，DBON 指令执行，将 D1D0 为 K179210 的 32 位二进制数 0000,0000,0000,0010,1011,1100,0000,1010 中的 b17 位进行 ON 位判定，b17 为 1，所以 Y0 为 ON。

DBONP

指令说明：

DBONP 是 32 位脉冲执行型 ON 位判定指令，即指令激活一次，执行一次 ON 位判定。是将源操作数[S+1,S]的 32 位数据的第 n 位进行 ON 位判定，n 位为 ON 时，目标操作数[D]为 ON，n 位为 OFF 时，目标操作数[D]为 OFF。

若源操作数[S+1，S]指定了常数 K 时，会自动进行 BIN 转换。

n 的取值范围在：K0~K31

操作数：

S：KnX, KnY, KnM, KnS,T,C,D,K,H,Z,V,LV,DT,@

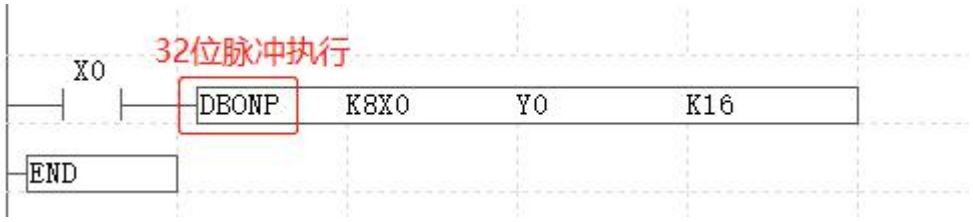
D：Y, M, S,@

n：K,H,D,@

指令格式：[DBONP　S　D　n]

编程示例：

X0 闭合，DBONP 指令执行，将 K8X0 中的 X20（K16 为十进制，需转换成八进制匹配 X）位进行 ON 位判定，X20 为 OFF 时，Y0 为 OFF，X20 为 ON 时，Y0 为 ON。



MEAN

指令说明：

MEAN 是 16 位连续执行型平均值指令，即每个扫描周期都执行一次数据的平均值运算。是将源操作数[S]起始的 n 个软元件的 16 位数据相加后的总值，除以 n 得到的平均值，最后保存到目的操作数[D]中。如果有余数，余数舍去。

操作数：

S：KnX, KnY, KnM, KnS,T,C,D,LV,DT,@

D：KnY, KnM, KnS,T,C,D, Z,V,LV,DT

n：D,K,H

指令格式：[MEAN　S　D　n]

编程示例：

假设 D0 为 K12，D1 为 K143，D2 为 K1076，X1 闭合，MEAN 指令执行，将 D0，D1，D2 的数据相加后除以 K3，得到的商为 K410 保存到 D10，余数为 1 舍去。

MEANP

指令说明：

MEANP 是 16 位脉冲执行型平均值指令，即指令激活一次，执行一次数据的平均值运算。是将源操作数[S]起始的 n 个软元件的 16 位数据相加后的总值，除以 n 得到的平均值，最后保存到目的操作数[D]中。如果有余数，余数舍去。

操作数：

S：KnX, KnY, KnM, KnS,T,C,D,LV,DT,@

D：KnY, KnM, KnS,T,C,D, Z,V,LV,DT

n：D,K,H

指令格式：[MEANP　S　D　n]

编程示例：

假设 D0 为 K240，D1 为 K265，X1 闭合，MEANP 指令执行一次，将 D0，D1 的数据相加后除以 K2，得到的商为 K252 保存到 D10 中，余数为 1 舍去。

DMEAN

指令说明：

DMEAN 是 32 位连续执行型平均值指令，即每个扫描周期都执行一次数据的平均值运算。是将源操作数[S+1,S]起始的 n 个软元件的 32 位数据相加后的总值，除以 n 得到的平均值，最后保存到目的操作数[D+1,D]中。如果有余数，余数舍去。

操作数：

S：KnX, KnY, KnM, KnS,T,C,D,LV,DT,@

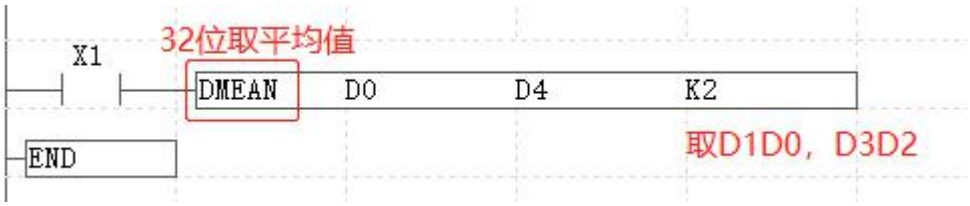
D：KnY, KnM, KnS,T,C,D, Z,V,LV,DT

n：D,K,H

指令格式：[DMEAN　S　D　n]

编程示例：

假设 D1D0 为 K42432，D3D2 为 K26540。X0 闭合，DMEAN 指令执行，将 D1D0 和 D3D2 的数据相加后除以 K2，得到的商为 K34486 保存到 D5D4 中。



DMEANP

指令说明：

DMEANP 是 32 位脉冲执行型平均值指令，即指令激活一次，执行一次数据的平均值运算。是将源操作数[S+1,S]起始的 n 个软元件的 32 位数据相加后的总值，除以 n 得到的平均值，最后保存到目的操作数[D+1,D]中。如果有余数，余数舍去。

操作数：

S：KnX, KnY, KnM, KnS,T,C,D,LV,DT,@

D：KnY, KnM, KnS,T,C,D, Z,V,LV,DT

n：D,K,H

指令格式：[DMEANP　S　D　n]

编程示例：

假设 D1D0 的值为 K30000，D3D2 为 K58600，其余的双字占据寄存器为 0。X1 闭合，DMEANP 指令执行一次，将 D1D0~D9D8 共 5 个 32 位数据相加后除以 K5，得到的商为 K17720 保存到 D11D10 中。

ANS

指令说明：

ANS 是 16 位连续执行型信号报警器置位指令，即每个扫描周期都执行一次运算。前面指令接通超过定时器[S]的判定时间[m×100ms]，置位步进状态器[D]。若不满定时器[S]的判定时间[m×100ms]，则不置位步进状态器[D]。

ANS 指令内的定时器为 100ms 类型。

操作数：

S：T

m：K,H,D,@

D：S

指令格式：[ANS　S　m　D]

编程示例：

X0 闭合，ANS 指令执行，若 X0 闭合的时间超过 1S（10×100ms），则 S0 置位。若 X0 的闭合时间不满 1S，则 S0 不置位。

ANR

ANR 是 16 位连续执行型信号报警器复位指令。

ANRP

ANRP 是 16 位脉冲执行型信号报警器复位指令。

SQR

指令说明：

SQR 是 16 位连续执行型 BIN 开方运算指令，即每个扫描周期都执行一次 BIN 开方运算。操作数[S]在开方运算的结果舍去小数点取整数，小数点舍去后生成时，借位标志位 M8021 置 ON。

该指令在开方结果为 0 时，置位标志位 M8020 置位。

操作数：

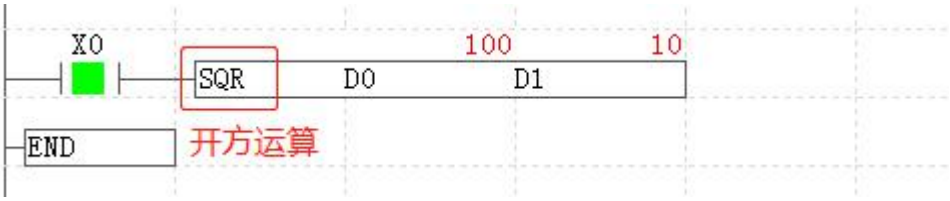
S：D,K,H,LV,DT,@

D：D, LV,DT

指令格式：[SQR　S　D]

编程示例：

假设 D0 为 K100，X0 闭合，SQR 指令执行，计算 D0 的平方根，得到 K10，保存到 D1 中。



SQRP

指令说明：

SQR 是 16 位连续执行型 BIN 开方运算指令，即每个扫描周期都执行一次 BIN 开方运算。操作数[S]在开方运算的结果舍去小数点取整数，小数点舍去后生成时，借位标志位 M8021 置 ON。

该指令在开方结果为 0 时，置位标志位 M8020 置位。

操作数：

S：D,K,H,LV,DT,@

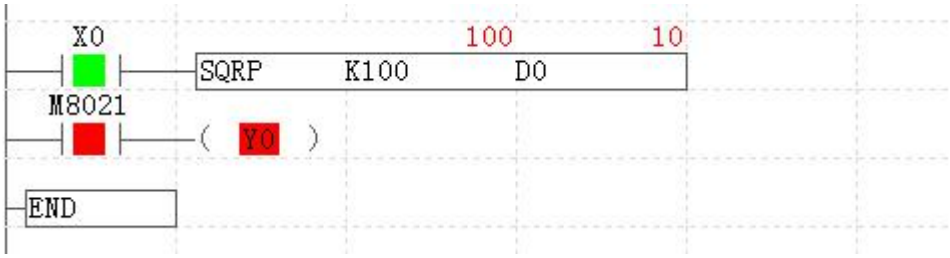
D：D,LV,DT

指令格式：[SQRP　S　D]

编程示例：

X0 闭合，SQRP 指令执行一次，计算 K30 的平方根，舍去小数点，得到 K5，保存到 D1 中，M8021 置 ON。

若开方运算结果为整数，则 M8021 为 OFF。



DSQR

指令说明：

DSQR 是 32 位连续执行型 BIN 开方运算指令，即每个扫描周期都执行一次 BIN 开方运算。计算源操作数[S+1,S]的数据的平方根后，保存到目的操作数[D+1,D]中。操作数[S]在开方运算的结果舍去小数点取整数，小数点舍去后生成时，借位标志位 M8021 置 ON。

操作数：

该指令在开方结果为 0 时，置位标志位 M8020 置位。

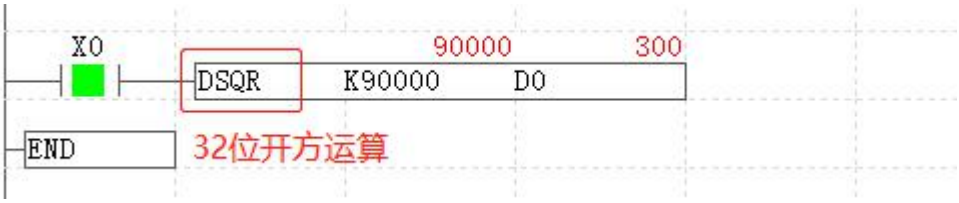
S：D,K,H,LV,DT,@

D：D, LV,DT

指令格式：[DSQR　S　D]

编程示例：

X0 闭合，DSQR 指令执行，计算 K90000 的平方根，得到 K300，保存到 D1D0 中。



DSQRP

指令说明：

DSQRP 是 32 位脉冲执行型 BIN 开方运算指令，即指令激活一次，执行一次 BIN 开方运算。计算源操作数[S+1,S]的数据的平方根后，保存到目的操作数[D+1,D]中。操作数[S]在开方运算的结果舍去小数点取整数，小数点舍去后生成时，借位标志位 M8021 置 ON。

该指令在开方结果为 0 时，置位标志位 M8020 置位。

操作数：

S：D,K,H,LV,DT,@

D：D,LV,DT

指令格式：[DSQRP　S　D]

编程示例：

X0 闭合，DSQRP 指令执行，计算 k90000 的平方根，得到 K300，保存到 D1D0 中。

FLT

指令说明：

FLT 是 16 位连续执行型 BIN 整数转换 2 进制浮点数指令。是将源操作数[S]的 BIN 整数值数据转换成 2 进制浮点数（实数）值后，保存到[D+1,D]中。

操作数：

S：D,LV,DT,@

D：D,LV,DT,

指令格式：[FLT　S　D]

编程示例：

假设 D0 的值为 K200，上电运行后，M8000 闭合，FLT 指令执行，将 D0 的 K200 转换成 200.000 的浮点数传送到 D2D1。

FLTP

指令说明：

FLTP 是 16 位脉冲执行型 BIN 整数转换 2 进制浮点数指令，即指令激活一次，执行一次数据转换运算。是将源操作数[S]的 BIN 整数值数据转换成 2 进制浮点数（实数）值后，保存到[D+1,D]中。

操作数：

S：D,LV,DT,@

D：D, LV,DT,

指令格式：[FLTP　S　D]

编程示例：

注意该指令同 FLT 的执行方式不一样，FLTP 只在 M8000 触点闭合时执行一次，除非再次上电，不然 FLTP 不再运行。

DFLT

指令说明：

FLT 是 32 位连续执行型 BIN 整数转换 2 进制浮点数指令。是将源操作数[S+1,S]的 BIN 整数值数据转换成 2 进制浮点数（实数）值后，保存到[D+1,D]中。

操作数：

S：D,LV,DT,@

D：D, LV,DT,

指令格式：[DFLT　S　D]

编程示例：

假设 D1D0 的值为 K50000，上电运行后，M8000 闭合，DFLT 指令执行，将 D1D0 的 K50000 转换成 50000.000 的浮点数传送到 D3D2。

DFLTP

指令说明：

DFLT 是 32 位脉冲执行型 BIN 整数转换 2 进制浮点数指令，即指令激活一次，执行一次数据转换运算。是将源操作数[S+1,S]的 BIN 整数值数据转换成 2 进制浮点数（实数）值后，保存到[D+1,D]中。

操作数：

S：D,LV,DT,@

D：D, LV,DT,

指令格式：[DFLTP　S　D]

编程示例：

注意该指令同 DFLT 的执行方式不一样，DFLTP 只在 M8000 触点闭合时执行一次，除非再次上电，不然 DFLTP 不再运行。

SWAP

指令说明：

SWAP 是 16 位连续执行型高低字节互换指令，即每个扫描周期都执行一次数据运算。是将源操作数[S]的 16 位数据的高 8 位和低 8 位互换。

操作数：

S：KnY, KnM, KnS,T,C,D,Z,V,LV,DT

指令格式：[SWAP　S]

编程示例：

假设 D0 的值为 K12。X0 闭合，SWAP 指令执行，D0 的值不停地在 K12 与 K3072 两者转换，这是因为 SWAP 具连续执行型，每个扫描周期都会执行互换。

D0 执行一次高低位字节交换的结果：

D0	b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0	值
执行前	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	12
执行后	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	3072

SWAPP

指令说明：

SWAP 是 16 位脉冲执行型高低字节互换指令，即指令激活一次，执行一次数据运算。是将源操作数[S]的 16 位数据的高 8 位和低 8 位互换。

操作数：

S：KnY, KnM, KnS,T,C,D,Z,V,LV,DT

指令格式：[SWAPP　S]

编程示例：

假设 K4Y0 的值为 K3，Y0 与 Y1 导通。X0 闭合，SWAPP 指令执行，将 K4Y0 的低 8 位与高 8 位互换，K4Y0 的值变为 K768，Y8，Y9 导通。

D0	Y17	Y16	Y15	Y14	Y13	Y12	Y11	Y10	Y7	Y6	Y5	Y4	Y3	Y2	Y1	Y0	值
执行前	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	3
执行后	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	768

DSWAP

指令说明：

DSWAP 是 32 位连续执行型高低字节互换指令，即每个扫描周期都执行一次数据运算。是将源操作数[S+1]与[S]的各自的 16 位数据的高 8 位和低 8 位互换。

操作数：

S：KnY, KnM, KnS,T,C,D,Z,V,LV,DT

指令格式：[DSWAP　S]

编程示例：

DSWAP 具连续执行型，每个扫描周期都会执行互换。所以一般用 DSWAPP 脉冲执行型指令。将 D1 的 16 位数据高 8 位和低 8 位互换，同时将 D0 的 16 位数据高 8 位和低 8 位互换。

假设 D1D0 的值为 K50000，它的二进制数是 0000,0000,0000,0000,1100,0011,0101,0000，高 16 位是 D1 的数据，低 16 位是 D0 的数据，X0 闭合，DSWAPP 指令执行一次，将占据 D1 的 16 位数据的高 8 位与低 8 位互换，占据 D0 的 16 位数据的高 8 位与低 8 位互换，互换后 D1D0 的值为 K20675，它的二进制数为 0000,0000,0000,0000,0101,0000,1100,0011。

DSWAPP

指令说明：

DSWAPP 是 32 位脉冲执行型高低字节互换指令，即指令激活一次，执行一次数据运算。是将源操作数[S+1]与[S]的各自的 16 位数据的高 8 位和低 8 位互换。

操作数：

S：KnY, KnM, KnS,T,C,D,Z,V,LV,DT

指令格式：[DSWAPP　S]

编程示例：

X0 每接通一次，DSWAPP 指令执行一次，将 D1 的 16 位数据高 8 位和低 8 位互换，同时将 D0 的 16 位数据高 8 位和低 8 位互换。

假设 D1D0 的值为 K175651，它的二进制数是 0000,0000,0000,0010,1010,1110,0010,0011，高 16 位是 D1 的数据，低 16 位是 D0 的数据，X0 闭合，DSWAPP 指令执行一次，将占据 D1 的 16 位数据的高 8 位与低 8 位互换，占据 D0 的 16 位数据的高 8 位与低 8 位互换，互换后 D1D0 的值为 K33563566，它的二进制数为 0000,0010,0000,0000,0010,0011,1010,1110。

3.8 浮点运算指令

浮点型指令均为 32 位。

分类	指令	脉冲	功能	操作数类型
比较	DECMP	-	[DECMP S1 S2 D] [S1+1,S1]>[S2+1,S2]，Dn 置 ON [S1+1,S1]=[S2+1,S2]，Dn+1 置 ON [S1+1,S1]<[S2+1,S2]，Dn+2 置 ON	S1/S2：D,K,H, LV,DT,@ D：Y, M, S,@
	DECMPP	√		
区间比较	DEZCP	-	[DEZCP S1 S2 S D] [S+1,S]<[S1+1,S1]，Dn 置 ON [S1+1,S1]<[S+1,S]<[S2+1,S2]，Dn+1 置 ON [S+1,S]>[S2+1,S2]，Dn+2 置 ON	S1/S2/S：D,K,H, LV,DT,@ D：Y, M, S,@
	DEZCPP	√		
	DEBCD	-	[DEBCD S D] [S+1,S]的 2 进制浮点数转换成 10 进制浮点数后，传送到[D+1,D]中	S：D,LV,DT,@ D：D,LV,DT
	DEBCDP	√		
	DEBIN	-	[DEBIN S D] [S+1,S]的 10 进制浮点数转换成 2 进制浮点数后，传送到[D+1,D]中	S：D,LV,DT,@ D：D,LV,DT
	DEBINP	√		
加法	DEADD	-	[DEADD S1 S2 D] [S1+1，S1]和[S2+1，S2]的 2 进制浮点数数据进行加法运算，将其运算结果以 2 进制浮点数形式传送到[D+1，D]中	S1：D,K,H,LV,DT,@ S2：D,K,H,LV,DT,@ D：D,LV,DT
	DEADDP	√		
减法	DESUB	-	[DESUB S1 S2 D] [S1+1，S1]和[S2+1，S2]的 2 进制浮点数数据进行减法运算，将其运算结果以 2 进制浮点数形式传送到	S1：D,K,H,LV,DT,@ S2：D,K,H,LV,DT,@ D：D,LV,DT
	DESUBP	√		

			[D+1，D]中	
乘法	DEMUL	-	[DEMUL S1 S2 D] [S1+1，S1]和[S2+1，S2]的 2 进制浮点数数据进行乘法运算，将其运算结果以 2 进制浮点数形式传送到[D+1，D]中	S1：D,K,H,LV,DT,@ S2：D,K,H,LV,DT,@ D：D,LV,DT
	DEMULP	√		
除法	DEDIV	-	[DEADD S1 S2 D] [S1+1，S1]和[S2+1，S2]的 2 进制浮点数数据进行除法运算，将其运算结果以 2 进制浮点数形式传送到[D+1，D]中	S1：D,K,H,LV,DT,@ S2：D,K,H,LV,DT,@ D：D,LV,DT
	DEDIVP	√		
开方	DESQR	-	[DESQR S D] [S+1,S]进行开方（开根号）运算后，将结果传送到[D+1,D]中	S：D,K,H,LV,DT,@ D：D,LV,DT
	DESQRP	√		
转换 BIN 整数	INT	-	[INT S D] 将 S 的 2 进制浮点数转换成 BIN 整数后，传送到 D 中	S：D,LV,DT,@ D：D, LV,DT
	INTP	√		
转换 BIN 整数	DINT	-	[DINT S D] 将[S+1,S]的 2 进制浮点数转换成 BIN 整数后，传送到[D+1,D]中	S：D,LV,DT,@ D：D, LV,DT
	DINTP	√		
角度值转 SIN	DSIN	-	[DSIN S D] [S+1,S]中指定的弧度值（2 进制浮点数）转换成 SIN 值后，传送到[D+1,D]中	S：D,LV,DT,@ D：D, LV,DT
	DSINP	√		
角度值转 COS	DCOS	-	[DCOS S D] [S+1,S]中指定的弧度值（2 进制浮点数）转换成 COS 值后，传送到[D+1,D]中	S：D,LV,DT,@ D：D, LV,DT
	DCOSP	√		
角度值转 TAN	DTAN	-	[DTAN S D] [S+1,S]中指定的弧度值（2 进制浮点数）转换成 TAN 值后，传送到[D+1,D]中	S：D,LV,DT,@ D：D, LV,DT
	DTANP	√		
浮点数传送	DEMOV	-	[DEMOV S D] [S+1,S]的 2 进制浮点数数据传送到[D+1,D]中	S：D,K,H,LV,DT,@ D：D,LV,DT
	DEMOVP	√		
指数运算	DEXP	-	[DEXP S D] [S+1,S]的数据进行指数运算后，结果传送到[D+1,D]中	S：D,K,H,LV,DT,@ D：D,LV,DT
	DEXPP	√		
自然对数运算	DLOGE	-	[DLOGE S D] [S+1,S]的数据执行自然对数运算后，结果传送到[D+1,D]中	S：D,K,H,LV,DT,@ D：D,LV,DT
	DLOGEP	√		
常数对数	DLOG	-	[DEMOV S D] [S+1,S]的数据执行常数对数运算	S：D,K,H,LV,DT,@ D：D,LV,DT
	DLOGP	√		

运算			后，结果传送到 [D+1,D]中	
SIN 转角度值	DASIN	-	[DASIN S D] [S+1,S]的 SIN 值转换成角度值，运算结果保存到[D+1,D]中	S：D,K,H,LV,DT,@ D：D,LV,DT
	DASINP	√		
COS 转角度值	DACOS	-	[DACOS S D] [S+1,S]的 COS 值转换成角度值，运算结果保存到[D+1,D]中	S：D,K,H,LV,DT,@ D：D,LV,DT
	DACOSP	√		
TAN 转角度值	DATAN	-	[DATAN S D] [S+1,S]的 TAN 值转换成角度值，运算结果保存到[D+1,D]中	S：D,K,H,LV,DT,@ D：D,LV,DT
	DATANP	√		
角度转弧度	DRAD	-	[DRAD S D] [S+1,S]的角度值转换成弧度值，运算结果保存到[D+1,D]中	S：D,K,H,LV,DT,@ D：D,LV,DT
	DRADP	√		
弧度转角度	DDEG	-	[DDEG S D] [S+1,S]的弧度值转换成角度值，运算结果保存到[D+1,D]中	S：D,K,H,LV,DT,@ D：D,LV,DT
	DDEGP	√		
浮点数转 SINH	DSINH	-	[DSINH S D] [S+1,S]的浮点数数据转换成 SINH 值，运算结果保存到[D+1,D]中 SINH 值=（$e^{x}-e^{-x}$）/2	S：D,K,H,LV,DT,@ D：D,LV,DT
	DSINHP	√		
浮点数转 COSH	DCOSH	-	[DCOSH S D] [S+1,S]的浮点数数据转换成 COSH 值，运算结果保存到[D+1,D]中 COSH 值=（$e^{x}+e^{-x}$）/2	S：D,K,H,LV,DT,@ D：D,LV,DT
	DCOSHP	√		
浮点数转 TANH	DTANH	-	[DTANH S D] [S+1,S]的浮点数数据转换成 TANH 值，运算结果保存到[D+1,D]中 TANH 值=（$e^{x}-e^{-x}$）/（$e^{x}+e^{-x}$）	S：D,K,H,LV,DT,@ D：D,LV,DT
	DTANHP	√		

DECMP

指令说明：

DECMP 是 32 位连续执行型浮点数比较指令，即每个扫描周期都执行一次浮点数比较运算。是将比较值[S1+1,S1]和比较源[S2+1,S2]作为浮点数数据进行比较，然后根据比较的结果（小于、等于或大于）将[D]、[D+1]、[D+2]中的任意一位置 ON。

[S1+1,S1]、[S2+1,S2]指定了常数（K、H）时，会自动将数值从 BIN 转换成 2 进制浮点数后再处理。

[D]占用 3 点，[D,D+1,D+2]。请注意不要与用于其它用途的软元件重复。

条件 / 比较指令	[S1+1,S1]>[S2+1,S2]	[S1+1,S1]=[S2+1,S2]	[S1+1,S1]<[S2+1,S2]
[DECMP S1 S2 Dn]	Dn 为 ON	Dn_{+1} 为 ON	Dn_{+2} 为 ON

操作数：

S1：D,K,H,LV,DT,@

S2：D,K,H,LV,DT,@

D：Y, M, S,@

指令格式：[DECMP　S1　S2　D]

编程示例：

X0 闭合，DECMP 指令执行，将 D1D0 的浮点数数据与 D3D2 进行比较，如下：

当 D1D0>D3D2 时，M0 置 ON，M1 和 M2 置 OFF，Y0 输出。

当 D1D0=D3D2 时，M1 置 ON，M0 和 M2 置 OFF，Y1 输出。

当 D1D0<D3D2 时，M2 置 ON，M0 和 M1 置 OFF，Y2 输出。

DECMPP

指令说明：

DECMPP 是 32 位脉冲执行型浮点数比较指令，即指令激活一次，执行一次浮点数比较运算。是将比较值[S1+1,S1]和比较源[S2+1,S2]作为浮点数数据进行比较，然后根据比较的结果（小于、等于或大于）将[D]、[D+1]、[D+2]中的任意一位置 ON。

[S1+1,S1]、[S2+1,S2]指定了常数（K、H）时，会自动将数值从 BIN 转换成 2 进制浮点数后再处理。

[D]占用 3 点，[D,D+1,D+2]。请注意不要与用于其它用途的软元件重复。

条件 / 比较指令	[S1+1,S1]>[S2+1,S2]	[S1+1,S1]=[S2+1,S2]	[S1+1,S1]<[S2+1,S2]
[DECMP S1 S2 Dn]	Dn 为 ON	Dn_{+1} 为 ON	Dn_{+2} 为 ON

操作数：

S1：D,K,H,LV,DT,@

S2：D,K,H,LV,DT,@

D：Y, M, S,@

指令格式：[DECMPP S1 S2 D]

编程示例：

当 X0 闭合时，DECMPP 指令执行一次，将 D1D0 的浮点数数据与 D3D2 进行比较，比较的结果，影响 Y0，Y1，Y2 三个位。

当 D1D0>D3D2 时，Y0 输出；

当 D1D0=D3D2 时，Y1 输出；

当 D1D0<D3D2 时，Y2 输出。

DEZCP

指令说明：

DEZCP 是 32 位连续执行型浮点数区间比较指令，即每个扫描周期都执行一次浮点数区间比较运算。是将比较值[S1+1,S1]、[S2+1,S2]和比较源[S+1,S]作为浮点数数据进行比较，然后根据比较结果（小于，等于或大于）将[D]、[D+1]、[D+2]中任意一位置 ON。

[S1+1,S1]、[S2+1,S2]和[S+1,S]指定了常数（K、H）时，会自动将数值从 BIN 转换成 2 进制浮点数后再处理。

[D]占用 3 点，[D,D+1,D+2]。请注意不要与用于其它用途的软元件重复。

条件 / 比较指令	[S+1,S]<[S1+1,S1]	[S1+1,S1]<=[S+1,S]<=[S2+1,S2]	[S+1,S]>[S2+1,S2]
[DEZCP S1 S2 S Dn]	Dn 为 ON	Dn_{+1} 为 ON	Dn_{+2} 为 ON

操作数：

S1：D,K,H,LV,DT,@

S2：D,K,H,LV,DT,@

S：D,K,H,LV,DT,@

D：Y, M, S,@

指令格式：[DEZCP S1 S2 S D]

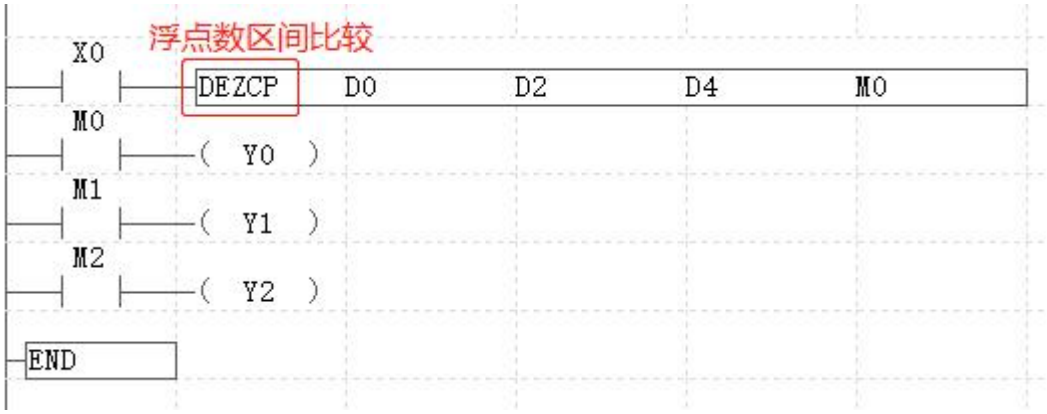
编程示例：

X0 闭合，DECMP 指令执行，将 D1D0、D3D2 的浮点数数据与 D5D4 进行比较，

当 D1D0>D5D4 时，M0 置 ON，M1 和 M2 置 OFF，Y0 输出；

当 D1D0≤D5D4≤D3D2 时，M1 置 ON，M0 和 M2 置 OFF，Y1 输出；

当 D3D2<D5D4 时，M2 置 ON，M0 和 M1 置 OFF，Y2 输出。



DEZCPP

指令说明：

DEZCPP 是 32 位脉冲执行型浮点数区间比较指令，即指令激活一次，执行一次浮点数区间比较运算。是将比较值[S1+1,S1]、[S2+1,S2]和比较源[S+1,S]作为浮点数数据进行比较，然后根据比较结果（小于，等于或大于）将[D]、[D+1]、[D+2]中任意一位置 ON。

[S1+1,S1]、[S2+1,S2]和[S+1,S]指定了常数（K、H）时，会自动将数值从 BIN 转换成 2 进制浮点数后再处理。

[D]占用 3 点，[D,D+1,D+2]。请注意不要与用于其它用途的软元件重复。

条件 / 比较指令	[S+1,S]<[S1+1,S1]	[S1+1,S1]<=[S+1,S]<=[S2+1,S2]	[S+1,S]>[S2+1,S2]
[DEZCPP S1 S2 S Dn]	Dn 为 ON	Dn_{+1} 为 ON	Dn_{+2} 为 ON

操作数：

S1：D,K,H,LV,DT,@

S2：D,K,H,LV,DT,@

S：D,K,H,LV,DT,@

D：Y, M, S,@

指令格式：[DEZCPP S1 S2 S D]

编程示例：

当 X0 闭合时，DEZCPP 指令执行一次，将 D1D0、D3D2 的浮点数数据与 D5D4 进行比较，比较的结果，影响 Y0，Y1，Y2 三个位。

当 D1D0>D5D4 时，Y0 输出；

当 D1D0≤D5D4≤D3D2 时，Y1 输出；

当 D3D2<D5D4 时，Y2 输出。

DEBCD

指令说明：

DEBCD 是 32 位连续执行型 2 进制浮点数转换成 10 进制浮点数的指令，即每个扫描周期都执行一次换算。将源操作数[S+1,S]的 2 进制浮点数转换成 10 进制浮点数后，传送到[D+1,D]中。

操作数：

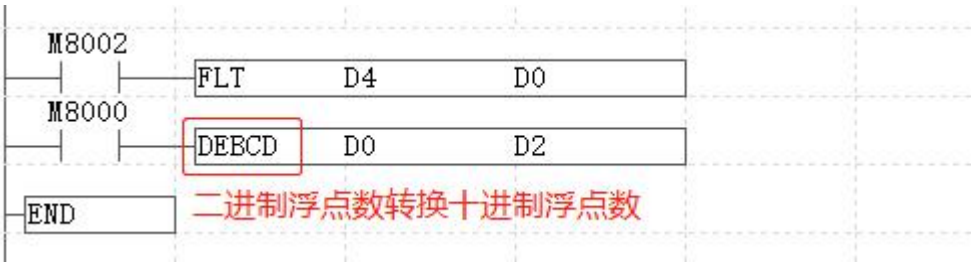
S：D,LV,DT,@

D：D,LV,DT

指令格式：[DEBCD S D]

编程示例：

上电运行后，M8002 导通一个扫描周期，FLT 指令将 D4 的值进行浮点数换算后传送到 D1D0，M8000 上电后一直导通，DEBCD 指令将 D1D0 的 2 进制浮点数转换成 10 进制浮点数传送到 D3D2。



DEBCDP

指令说明：

DEBCDP 是 32 位脉冲执行型 2 进制浮点数转换成 10 进制浮点数的指令，即指令激活一次，执行一次换算。将源操作数[S+1,S]的 2 进制浮点数转换成 10 进制浮点数后，传送到[D+1,D]中。

操作数：

S：D,LV,DT,@

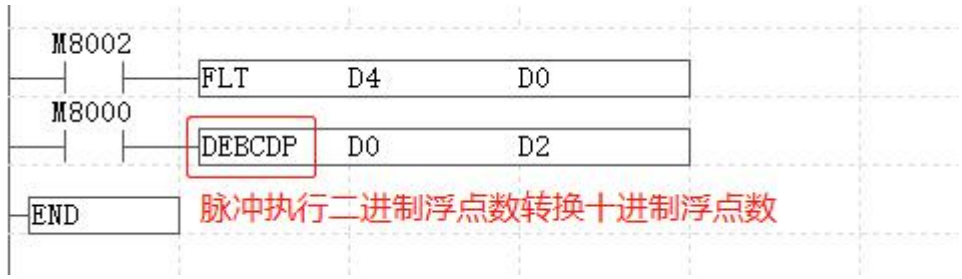
D：D,LV,DT

指令格式：[DEBCDP S D]

编程示例：

该指令同 DEBCD 指令用法相同，只不过是该指令具有脉冲执行型，仅在 M8000 闭合时，产生的上升沿脉冲激活 DEBCDP 指令执行一次。

上电运行后，M8002 导通一个扫描周期，FLT 指令将 D4 的值进行浮点数换算后传送到 D1D0，M8000 上电后一直导通，DEBCDP 指令将 D1D0 的 2 进制浮点数转换成 10 进制浮点数传送到 D3D2。



DEBIN

指令说明：

DEBIN 是 32 位连续执行型 10 进制浮点数转换成 2 进制浮点数的指令，即每个扫描周期都执行一次换算。将源操作数[S+1,S]的 10 进制浮点数转换成 2 进制浮点数后，传送到[D+1,D]中。

操作数：

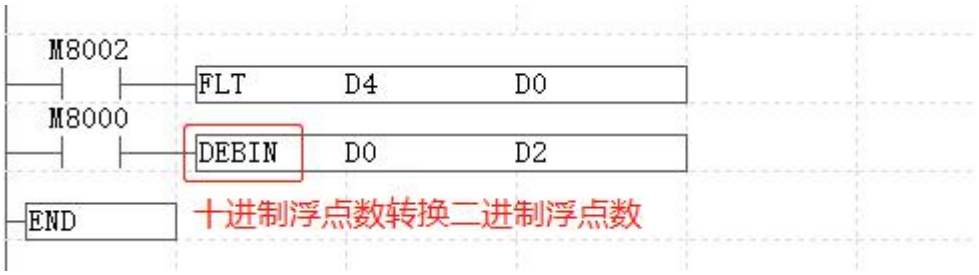
S：D, LV,DT,@

D：D, LV,DT

指令格式：[DEBIN　S　D]

编程示例：

上电运行后，M8002 导通一个扫描周期，FLT 指令将 D4 的值进行浮点数换算后传送到 D1D0，M8000 上电后一直导通，DEBIN 指令将 D1D0 的 10 进制浮点数转换成 2 进制浮点数传送到 D3D2。



DEBINP

指令说明：

DEBINP 是 32 位脉冲执行型 10 进制浮点数转换成 2 进制浮点数的指令，即指令激活一次，执行一次换算。将源操作数[S+1,S]的 10 进制浮点数转换成 2 进制浮点数后，传送到[D+1,D]中。

操作数：

S：D, LV,DT,@

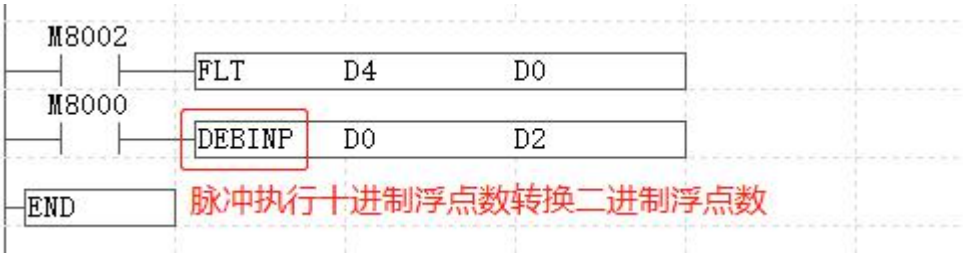
D：D, LV,DT

指令格式：[DEBINP　S　D]

编程示例：

该指令同 DEBIN 指令用法相同，只不过是该指令具有脉冲执行型，仅在 M8000 闭合时，产生的上升沿脉冲激活 DEBINP 指令执行一次。

上电运行后，M8002 导通一个扫描周期，FLT 指令将 D4 的值进行浮点数换算后传送到 D1D0，M8000 上电后一直导通，DEBIN 指令将 D1D0 的 10 进制浮点数转换成 2 进制浮点数传送到 D3D2。



DEADD/DEADDP/DESUB/DESUBP/DEMUL/DEMULP/ DEDIV/DEDIVP/DESQR/DESQRP

指令说明：

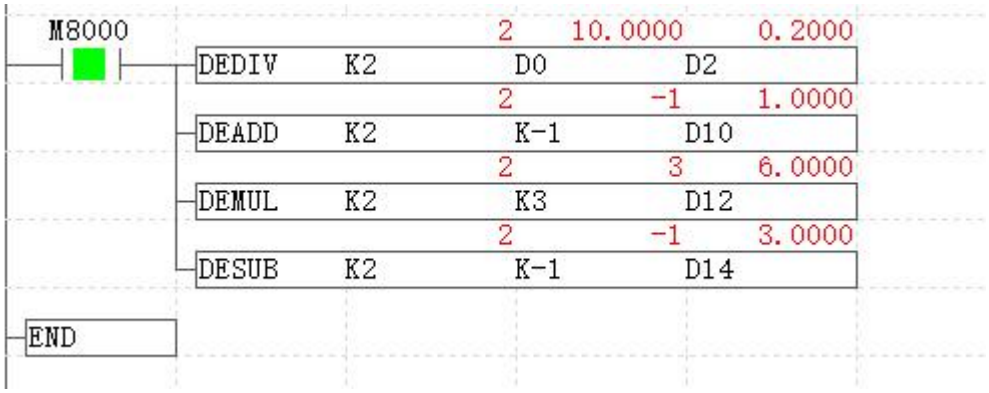
该组指令用于浮点数四则运算。

32 位浮点指令	是否为脉冲型	操作数种类	指令名称
DEADD	否	S1：D,K,H,LV,DT,@ S2：D,K,H,LV,DT,@ D：D,LV,DT	浮点数加法
DEADDP	是		
DESUB	否		浮点数减法
DESUBP	是		
DEMUL	否		浮点数乘法
DEMULP	是		
DEDIV	否		浮点数除法
DEDIVP	是		
DESQR	否		浮点数开方
DESQRP	是		

注意：浮点数指令会将参数中的常数自动转化为浮点数（包含三菱中提到的RAD/DEG/EXP/LOGE）。

浮点数指令中使用数据寄存器时，数据寄存器内的值必须为浮点数或可以自动转化为常数的值。

编程示例：



DEADD

指令说明：

DEADD 是 32 位连续执行型 2 进制浮点数加法运算指令，即每个扫描周期都执行一次运算。是将源操作数[S1+1,S1]和[S2+1,S2]的 2 进制浮点数数据进行加法运算，并将其运算结果以 2 进制浮点数形式传送到目标操作数[D+1,D]中。

在[S1+1,S1]和[S2+1,S2]中指定了常数（K,H）时，数值会自动转换成 2 进制浮点数。

操作数：

S1：D,K,H,LV,DT,@

S2：D,K,H,LV,DT,@

D：D,LV,DT

指令格式：[DEADD　S1　S2　D]

编程示例：

X0 闭合，DEADD 指令执行，将 K20 与 K30 转换成 2 进制浮点数后进行加法运算，运算后得 50.0000 传送到 D1D0；将 D1D0 的 50.0000 与 K10 的浮点数相加，得到的值为 60.0000，传送到 D3D2 中。

DEADDP

指令说明：

DEADDP 是 32 位脉冲执行型 2 进制浮点数加法运算指令，即指令激活一次，执行一次运算。是将源操作数[S1+1,S1]和[S2+1,S2]的 2 进制浮点数数据进行加法运算，并将其运算结果以 2 进制浮点数形式传送到目标操作数[D+1,D]中。

在[S1+1,S1]和[S2+1,S2]中指定了常数（K,H）时，数值会自动转换成 2 进制浮点数。

操作数：

S1：D,K,H,LV,DT,@

S2：D,K,H,LV,DT,@

D：D,LV,DT

指令格式：[DEADDP　S1　S2　D]

编程示例：

假设 D3D2 的初始值为 0，X0 闭合，DEADDP 指令执行一次浮点数自加运算，每接通一次 X0，D3D2 里面的数据自加 2.000。

DESUB

指令说明：

DESUB 是 32 位连续执行型 2 进制浮点数减法运算指令，即每个扫描周期都执行一次运算。是将源操作数[S1+1,S1]和[S2+1,S2]的 2 进制浮点数数据进行减法运算，并将其运算结果以 2 进制浮点数形式传送到目标操作数[D+1,D]中。

在[S1+1,S1]和[S2+1,S2]中指定了常数（K,H）时，数值会自动转换成 2 进制浮点数。

操作数：

S1：D,K,H,LV,DT,@

S2：D,K,H,LV,DT,@

D：D,LV,DT

指令格式：[DESUB　S1　S2　D]

编程示例：

上电运行时，M8002 导通一个扫描周期，DMOV 指令将常数 K2000 传送给 D0，DFLT 指令将 D0 的数据转换为浮点数，存储到 D3D2，X0 闭合，DESUB 指令执行，将 D3D2 的浮点数 2000.0000 减去 K100 的浮点数，运算后得到 1900.0000，传送到 D5D4 中。

DESUBP

指令说明：

DESUBP 是 32 位脉冲执行型 2 进制浮点数减法运算指令，即指令激活一次，执行一次运算。是将源操作数[S1+1,S1]和[S2+1,S2]的 2 进制浮点数数据进行减法运算，并将其运算结果以 2 进制浮点数形式传送到目标操作数[D+1,D]中。

在[S1+1,S1]和[S2+1,S2]中指定了常数（K,H）时，数值会自动转换成 2 进制浮点数。

操作数：

S1：D,K,H,LV,DT,@

S2：D,K,H,LV,DT,@

D：D,LV,DT

指令格式：[DESUBP　S1　S2　D]

编程示例：

上电运行时，M8002 导通一个扫描周期，DMOV 指令将常数 K2000 传送给 D0，DFLT 指令将 D0 的数据转换为浮点数，存储到 D3D2，X0 闭合，DESUB 指令执行，将 D3D2 的浮点数 2000.0000 减去 K100 的浮点数，运算后得到 1900.0000，传送到 D5D4 中。

X1 闭合，DESUBP 指令执行浮点数自减运算，每接通一次 X1，D5D4 里面的数据自减 2.0000。

DEMUL

指令说明：

DEMUL 是 32 位连续执行型 2 进制浮点数乘法运算指令，即每个扫描周期都执行一次运算。是将源操作数[S1+1,S1]和[S2+1,S2]的 2 进制浮点数数据相乘，并将其运算结果以 2 进制浮点数形式传送到目标操作数[D+1，D]中。

在[S1+1,S1]和[S2+1,S2]中指定了常数（K,H）时，数值会自动转换成 2 进制浮点数。

操作数：

S1：D,K,H,LV,DT,@

S2：D,K,H,LV,DT,@

D：D,LV,DT

指令格式：[DEMUL　S1　S2　D]

编程示例：

上电运行时，M8002 导通一个扫描周期，DMOV 指令将常数 K2000 传送给 D0，DFLT 指令将 D0 的数据转换为浮点数，存储到 D3D2，X0 闭合，DEMUL 指令执行，将 D3D2 的浮点数 20.0000 与 K20 的浮点数相乘，得到的乘积为 400.0000，传送到 D5D4 中。

DEMULP

指令说明：

DEMULP 是 32 位脉冲执行型 2 进制浮点数乘法运算指令，即指令激活一次，执行一次运算。是将源操作数[S1+1,S1]和[S2+1,S2]的 2 进制浮点数数据相乘，并将其运算结果以 2

进制浮点数形式传送到目标操作数[D+1，D]中。

在[S1+1,S1]和[S2+1,S2]中指定了常数（K,H）时，数值会自动转换成 2 进制浮点数。

操作数：

S1：D,K,H,LV,DT,@

S2：D,K,H,LV,DT,@

D：D,LV,DT

指令格式：[DEMULP　S1　S2　D]

编程示例：

上电运行时，M8002 导通一个扫描周期，DMOV 指令将常数 K20 传送给 D0，DFLT 指令将 D0 的数据转换为浮点数，存储到 D3D2，X0 闭合，DEMUL 指令执行，将 D3D2 的浮点数 20.0000 与 K30 相乘，得到的乘积为 600.0000，传送到 D5D4 中。

DEDIV

指令说明：

DEDIV 是 32 位连续执行型 2 进制浮点数除法运算指令，即每个扫描周期都执行一次运算。是将源操作数[S1+1,S1]和[S2+1,S2]的 2 进制浮点数数据相除，并将其运算结果以 2 进制浮点数形式传送到目标操作数[D+1,D]中。

在[S1+1,S1]和[S2+1,S2]中指定了常数（K,H）时，数值会自动转换成 2 进制浮点数。

操作数：

S1：D,K,H,LV,DT,@

S2：D,K,H,LV,DT,@

D：D,LV,DT

指令格式：[DEDIV　S1　S2　D]

编程示例：

上电运行时，M8002 导通一个扫描周期，DMOV 指令将常数 K100 传送给 D0，DFLT 指令将 D0 的数据转换为浮点数，存储到 D3D2，X0 闭合，DEDIV 指令执行，将 D3D2 的浮点数 100.0000 除以 K2 的浮点数后得到的商为 50.0000 传送到 D9D8 中。

DEDIVP

指令说明：

DEDIVP 是 32 位脉冲执行型 2 进制浮点数除法运算指令，即指令激活一次，执行一次运算。是将源操作数[S1+1,S1]和[S2+1,S2]的 2 进制浮点数数据相除，并将其运算结果以 2 进制浮点数形式传送到目标操作数[D+1,D]中。

在[S1+1,S1]和[S2+1,S2]中指定了常数（K,H）时，数值会自动转换成 2 进制浮点数。

操作数：

S1：D,K,H,LV,DT,@

S2：D,K,H,LV,DT,@

D：D,LV,DT

指令格式：[DEDIVP　S1　S2　D]

编程示例：

X0 闭合，DMOV 指令将常数 K100 传送给 D1D0，DFLT 指令将 D1D0 的数据转换为浮点数，存储到 D3D2，DMOV 指令将常数 K2000 传送给 D5D4，DFLT 指令将 D5D4 的数据转换为浮点数，存储到 D7D6。

X1 闭合，DEDIVP 指令执行，将 D7D6 的浮点数 2000.0000 除以 D3D2 的浮点数 100 后得到的商为 20.0000 传送到 D9D8 中。

X2 闭合，DEDIVP 指令执行，D9D8 的浮点数 20.0000 数据除以 K2，得到的商为 10.0000 传送到 D11D10 中。

DESQR

指令说明：

DESQR 是 32 位连续执行型 2 进制浮点数开方运算，即每个扫描周期都执行一次运算。是将源操作数[S+1,S]进行开方（开根号）运算后，将其结果传送到目标操作数[D+1,D]中。

在[S+1,S]中指定了常数（K,H）时，数值会自动转换成 2 进制浮点数。

[S+1,S]的内容仅正数有效，若为负数时，运算出错，不执行指令。

该指令在开方结果为 0 时，置位标志位 M8020 置位。

操作数：

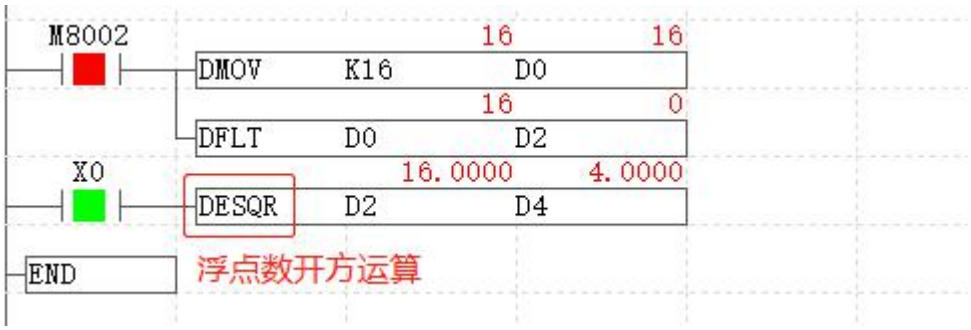
S：D,K,H,LV,DT,@

D：D,LV,DT

指令格式：[DESQR　S　D]

编程示例：

上电运行时，M8002 导通一个扫描周期，DMOV 指令将常数 K16 传送给 D1D0，DFLT 指令将 D1D0 的数据转换为浮点数，存储到 D3D2，X0 闭合，DESQR 指令执行，将 D3D2 的浮点数 16.0000 开方运算，运算后的浮点数值为 4.0000，传送到 D5D4 中。



DESQRP

指令说明：

DESQRP 是 32 位连续执行型 2 进制浮点数开方运算，即指令激活一次，执行一次运算。是将源操作数[S+1,S]进行开方（开根号）运算后，将其结果传送到目标操作数[D+1,D]中。

在[S+1,S]中指定了常数（K,H）时，数值会自动转换成 2 进制浮点数。

[S+1,S]的内容仅正数有效，若为负数时，运算出错，不执行指令。

该指令在开方结果为 0 时，置位标志位 M8020 置位。

操作数：

S：D,K,H,LV,DT,@

D：D,LV,DT

指令格式：[DESQRP　S　D]

编程示例：

上电运行时，M8002 导通一个扫描周期，DMOV 指令将常数 K25 传送给 D1D0，DFLT 指令将 D1D0 的数据转换为浮点数，存储到 D3D2，X0 闭合，DESQRP 指令执行一次，将

D3D2 的浮点数 25.0000 开方运算，运算后的浮点数值为 5.0000，传送到 D5D4 中。

INT

指令说明：

INT 是 16 位连续执行型 2 进制浮点数转换成 BIN 整数的指令，即每个扫描周期都执行一次运算。将源操作数[S+1,S]的 2 进制浮点数转换成 BIN 整数后，传送到目标操作数[D]中。

目标操作数[D]的 16 位 BIN 整数小数点以后舍去。

源操作数[S]里的浮点数大于 16 位寄存器的正限值或小于 16 位寄存器的负限值时，进位标志位 M8022 会置位。

操作数：

S：D, LV,DT,@

D：D, LV,DT

指令格式：[INT　S　D]

编程示例：

X0 闭合，INT 指令执行，将 D3D2 的浮点数值 100.0000 转换成整数值 100 传送到 D4，所以 D4 的值为 K100。

INTP

指令说明：

INT 是 16 位脉冲执行型 2 进制浮点数转换成 BIN 整数的指令，即指令激活一次，执行一次运算。将源操作数[S+1,S]的 2 进制浮点数转换成 BIN 整数后，传送到目标操作数[D]中。

目标操作数[D]的 16 位 BIN 整数小数点以后舍去。

源操作数[S]里的浮点数大于 16 位寄存器的正限值或小于 16 位寄存器的负限值时，进

位标志位 M8022 会置位。

操作数：

S：D, LV,DT,@

D：D, LV,DT

指令格式：[INTP　S　D]

编程示例：

辅助触点 M0 闭合，INTP 指令脉冲执行一次，将 D5D4 的浮点数值 100.000 转换成 100，传送到 D3D2 中。

DINT

指令说明：

DINT 是 32 位连续执行型 2 进制浮点数转换成 BIN 整数的指令，即每个扫描周期都执行一次运算。是将源操作数[S+1,S]的 2 进制浮点数转换成 BIN 整数后，传送到目标操作数[D+1,D]中。

目标操作数[D]的 32 位 BIN 整数小数点以后舍去。

操作数：

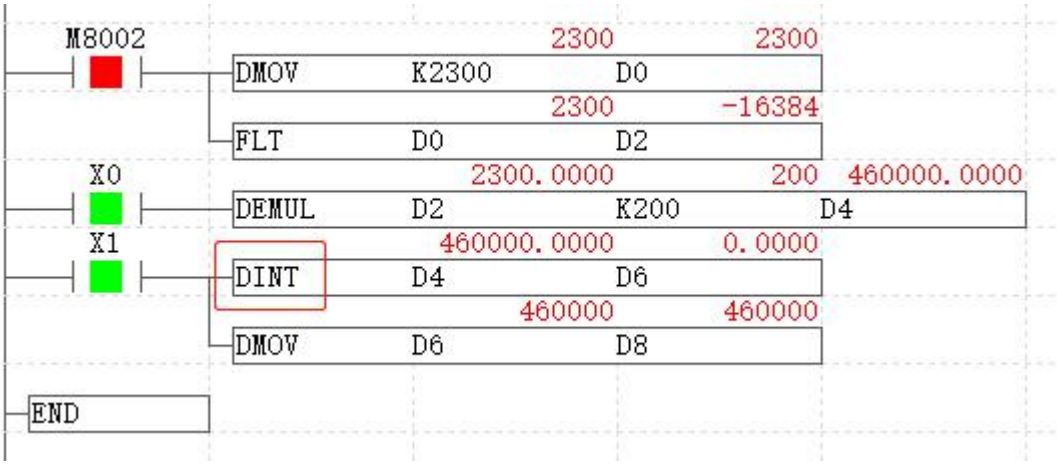
S：D, LV,DT,@

D：D, LV,DT

指令格式：[DINT　S　D]

编程示例：

X1 闭合，DINT 指令执行，将 D5D4 的浮点数值 460000.0000 转换成整数值 460000 后传送到 D7D6 中。



DINTP

指令说明：

DINTP 是 32 位脉冲执行型 2 进制浮点数转换成 BIN 整数的指令，即指令激活一次，执行一次运算。是将源操作数[S+1,S]的 2 进制浮点数转换成 BIN 整数后，传送到目标操作数[D+1,D]中。

目标操作数[D]的 32 位 BIN 整数小数点以后舍去。

操作数：

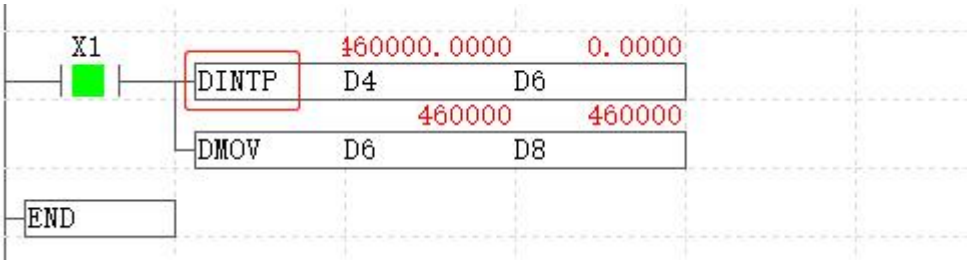
S：D, LV,DT,@

D：D, LV,DT

指令格式：[DINTP　S　D]

编程示例：

X1 闭合，DINTP 指令脉冲执行一次，将 D5D4 的浮点数值 460000.0000 转换成整数值 460000 后传送到 D7D6 中。



DSIN

指令说明：

DSIN 是 32 位连续执行型浮点数 SIN 运算指令，即每个扫描周期都执行一次运算。是将源操作数[S+1,S]中指定的弧度值（2 进制浮点数）转换成 SIN 值后，传送到目标操作数[D+1,D]中。

操作数：

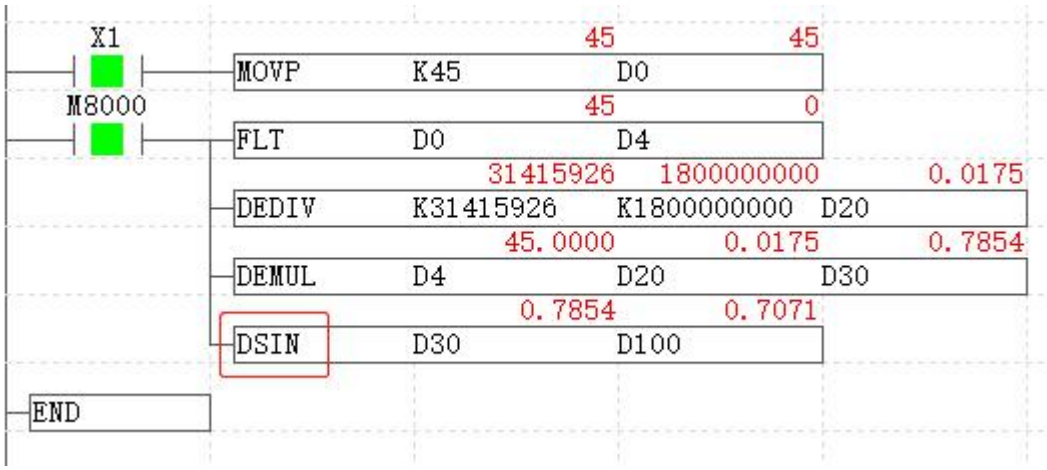
S：D, LV,DT, @

D：D, LV,DT

指令格式：[DSIN　S　D]

编程示例：

X1 闭合，将 K45 传送到 D0。M8000 上电运行后常闭，FLT 指令执行，将 D0 的整数 45 转换成浮点数 45.0000 保存到 D5D4 中。DEDIV 指令将π/180 的浮点数值 0.0175 存入 D21D20。DEMUL 指令将 D5D4 的角度浮点数 45.0000 与 D21D20 的π/180 浮点数 0.0175 相乘后得到弧度值 0.7854，传送到 D31D30。DSIN 指令执行，将弧度值转换成 SIN 值为 0.7071 后，传送到 D101D100 中。



DSINP

指令说明：

DSINP 是 32 位脉冲执行型浮点数 SIN 运算指令，即指令激活一次，执行一次运算。是将源操作数[S+1,S]中指定的弧度值（2 进制浮点数）转换成 SIN 值后，传送到目标操作数[D+1,D]中。

操作数：

S：D, LV,DT, @

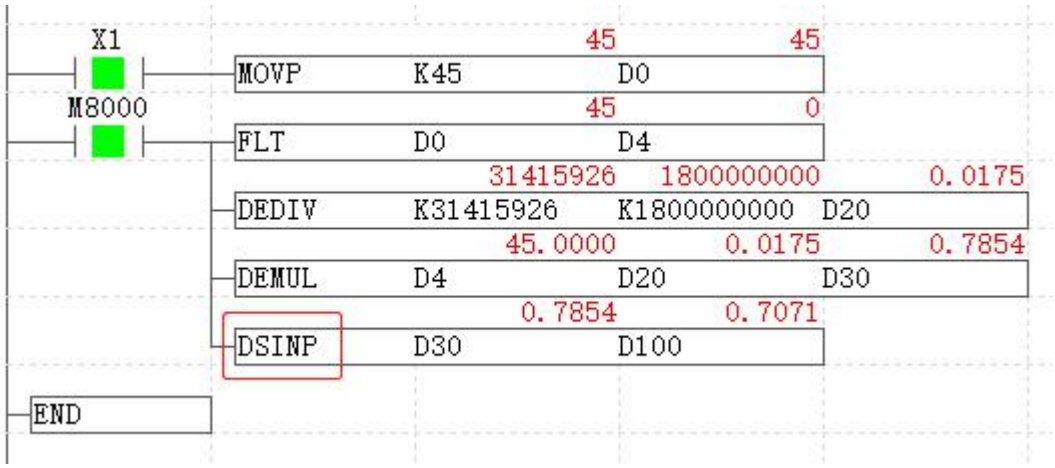
D：D, LV,DT

指令格式：[DSINP　S　D]

编程示例：

DSINP 指令与 DSIN 指令用法相同，但它是脉冲执行方式，DSINP 靠前面的触点闭合时产生的上升沿脉冲激活，执行一次。

X1 闭合，K45 传送到 D0。M8000 上电运行后常闭，FLT 指令执行，将 D0 的整数 45 转换成浮点数 45.0000 保存到 D5D4 中。DEDIV 指令将π/180 的浮点数值 0.0157 存入 D21D20。DEMUL 指令将 D5D4 的角度浮点数 45.0000 与 D21D20 的π/180 浮点数 0.0175 相乘后得到弧度值 0.7854，传送到 D31D30。DSINP 指令执行，将弧度值转换成 SIN 值为 0.7071 后，传送到 D101D100 中。



DCOS

指令说明：

DCOS 是 32 位连续执行型浮点数 COS 运算指令，即每个扫描周期都执行一次运算。是将源操作数[S+1,S]中指定的弧度值（2 进制浮点数）转换成 COS 值后，传送到目标操作数[D+1,D]中。

操作数：

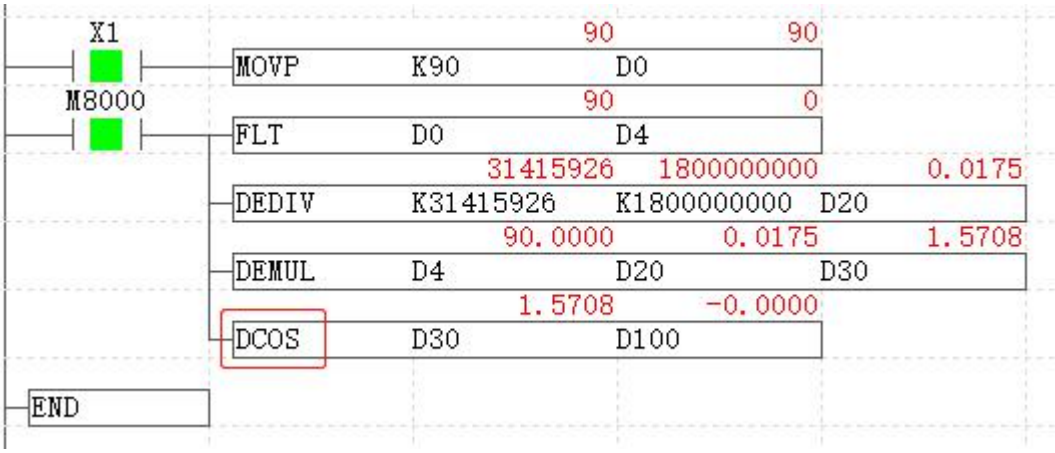
S：D, LV,DT, @

D：D, LV,DT

指令格式：[DCOS S D]

编程示例：

X2 闭合，K90 传送到 D0。M8000 上电运行后常闭，FLT 指令执行，将 D0 的整数 90 转换成浮点数 90.0000 保存到 D5D4 中。DEDIV 指令将π/180 的浮点数值 0.0175 存入 D21D20。DEMUL 指令将 D5D4 的角度浮点数 90.0000 与 D21D20 的π/180 浮点数 0.0175 相乘后得到弧度值 1.5708，传送到 D31D30。DCOS 指令执行，将弧度值转换成 COS 值为 0 后，传送到 D101D100 中。



DCOSP

指令说明：

DCOSP 是 32 位脉冲执行型浮点数 COS 运算指令，即指令激活一次，执行一次运算。是将源操作数[S+1,S]中指定的弧度值（2 进制浮点数）转换成 COS 值后，传送到目标操作数[D+1,D]中。

操作数：

S：D, LV,DT, @

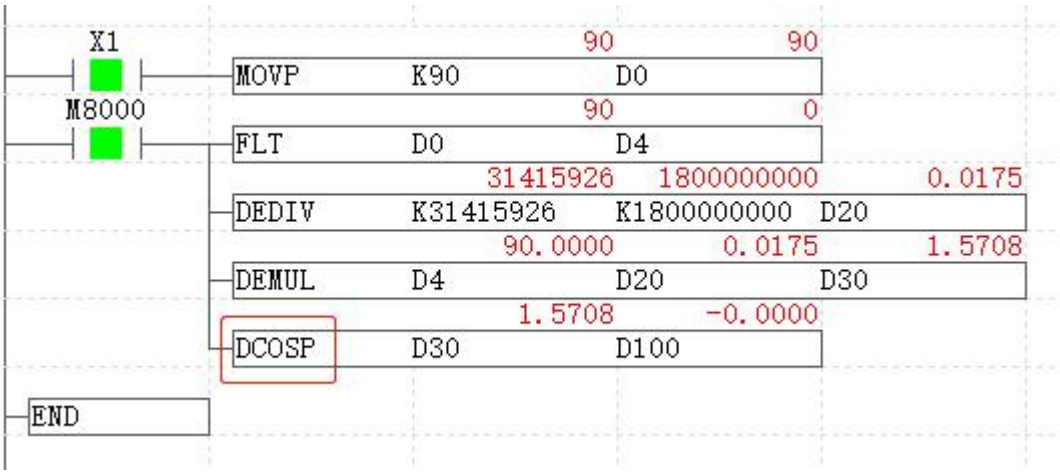
D：D, LV,DT

指令格式：[DCOSP　S　D]

编程示例：

DCOSP 指令与 DCOS 指令用法相同，但它是脉冲执行方式，DCOSP 靠前面的触点闭合时产生的上升沿脉冲激活，执行一次。

X2 闭合，K90 传送到 D0。M8000 上电运行后常闭，FLT 指令执行，将 D0 的整数 90 转换成浮点数 90.0000 保存到 D5D4 中。DEDIV 指令将π/180 的浮点数值 0.0175 存入 D21D20。DEMUL 指令将 D5D4 的角度浮点数 90.0000 与 D21D20 的π/180 浮点数 0.0175 相乘后得到弧度值 1.5708，传送到 D31D30。DCOSP 指令执行，将弧度值转换成 COS 值为 0 后，传送到 D101D100 中。



DTAN

指令说明：

DTAN 是 32 位连续执行型浮点数 TAN 运算指令，即每个扫描周期都执行一次运算。是将源操作数[S+1,S]中指定的弧度值（2 进制浮点数）转换成 TAN 值后，传送到目标操作数[D+1,D]中。

操作数：

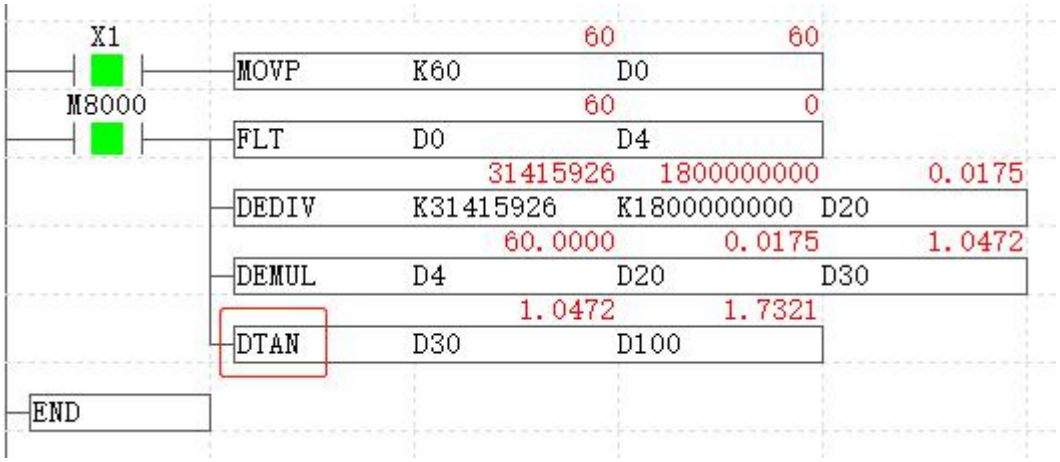
S：D, LV,DT, @

D：D, LV,DT

指令格式：[DTAN　S　D]

编程示例：

X1 闭合，K60 传送到 D0。M8000 上电运行后常闭，FLT 指令执行，将 D0 的整数 60 转换成浮点数 60.0000 保存到 D5D4 中。DEDIV 指令将π/180 的浮点数值 0.0175 存入 D21D20。DEMUL 指令将 D5D4 的角度浮点数 60.0000 与 D21D20 的π/180 浮点数 0.0175 相乘后得到弧度值 1.0472，传送到 D31D30。DTAN 指令执行，将弧度值转换成 TAN 值为 1.7321 后，传送到 D101D100 中。



DTANP

指令说明：

DTANP 是 32 位脉冲执行型浮点数 TAN 运算指令，即指令激活一次，执行一次运算。是将源操作数[S+1,S]中指定的弧度值（2 进制浮点数）转换成 TAN 值后，传送到目标操作数[D+1,D]中。

操作数：

S：D, LV,DT, @

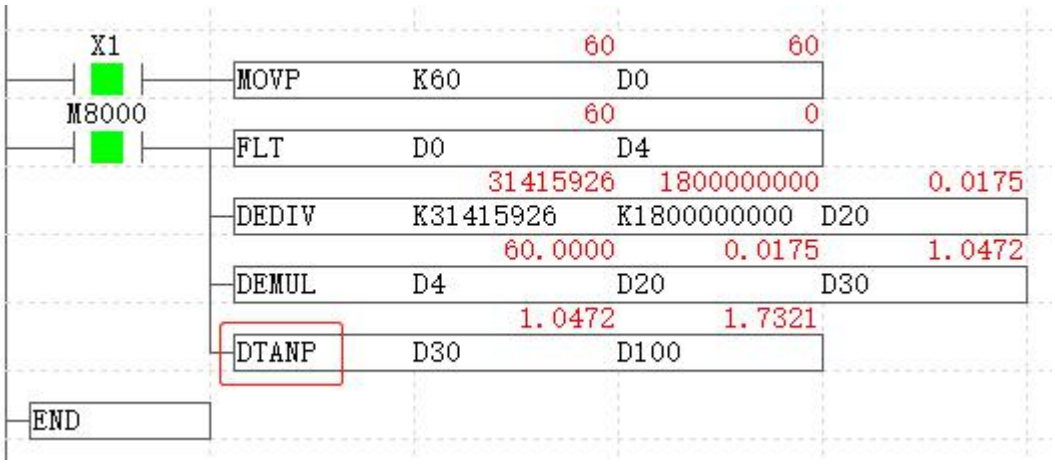
D：D, LV,DT

指令格式：[DTANP　S　D]

编程示例：

DTANP 指令与 DTAN 指令用法相同，但它是脉冲执行方式，DTANP 靠前面的触点闭合时产生的上升沿脉冲激活，执行一次。

X1 闭合，K60 传送到 D0。M8000 上电运行后常闭，FLT 指令执行，将 D0 的整数 60 转换成浮点数 60.0000 保存到 D5D4 中。DEDIV 指令将π/180 的浮点数值 0.0175 存入 D21D20。DEMUL 指令将 D5D4 的角度浮点数 60.0000 与 D21D20 的π/180 浮点数 0.0175 相乘后得到弧度值 1.0472，传送到 D31D30。DTANP 指令执行，将弧度值转换成 TAN 值为 1.7321 后，传送到 D101D100 中。



DEMOV

指令说明：

DEMOV 是 32 位连续执行型 2 进制浮点数数据传送指令，即每个扫描周期都执行一次运算。是将源操作数[S+1,S]的 2 进制浮点数数据传送到目标操作数[D+1,D]中。

在[S+1,S]中指定了常数（K,H）时，数值会自动转换成 2 进制浮点数。

操作数：

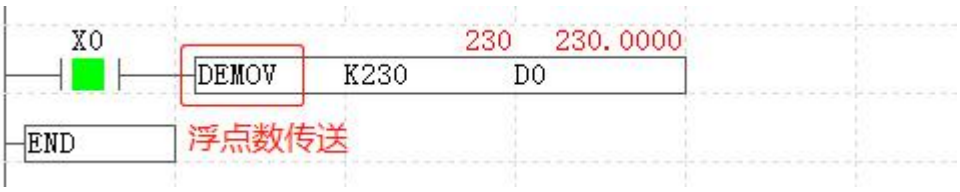
S：D, LV,K,H,DT, @

D：D, LV,DT

指令格式：[DEMOV　S　D]

编程示例：

X0 闭合，DEMOV 指令执行，将 K230 的浮点数 230.0000 传送到 D1D0 中。



DEMOVP

指令说明：

DEMOVP 是 32 位脉冲执行型 2 进制浮点数数据传送指令，即指令激活一次，执行一次运算。是将源操作数[S+1,S]的 2 进制浮点数数据传送到目标操作数[D+1,D]中。

在[S+1,S]中指定了常数（K,H）时，数值会自动转换成 2 进制浮点数。

操作数：

S：D, LV,K,H,DT, @

D：D, LV,DT

指令格式：[DEMOV　S　D]

编程示例：

X0 产生上升沿脉冲触发 DEMOVP 指令执行一次，将 K230 的浮点数值 230.0000 传送到 D1D0 中，D1D0 的值为浮点数 230.0000。

DEXP

指令说明：

DEXP 是 32 位连续执行型 2 进制浮点数指数运算指令，即每个扫描周期都执行一次运算。是将源操作数[S+1,S]为指数做运算，将运算结果保存到目标操作数[D+1,D]中，是以自然常数 2.71828 作为底数。

操作数：

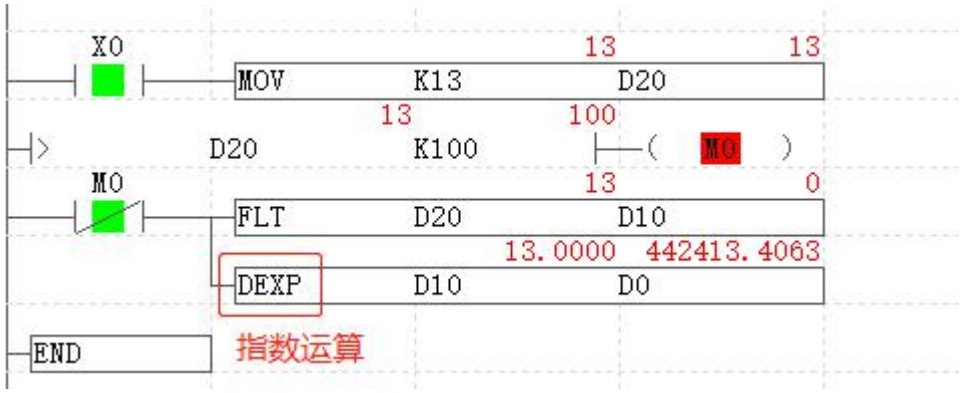
S：D ,K,H LV,DT, @

D：D, LV,DT

指令格式：[DEXP　S　D]

编程示例：

条件满足后，DEXP 指令执行，将 D11D10 里面的浮点数 13.0000 进行指数运算，运算后的浮点数值为 442413.4063，传送到 D1D0。



DEXPP

指令说明：

DEXPP 是 32 位脉冲执行型 2 进制浮点数指数运算指令，即指令激活一次，执行一次运算。是将源操作数[S+1,S]为指数做运算，将运算结果保存到目标操作数[D+1,D]中。是以自然常数 2.71828 作为底数。

操作数：

S：D ,K,H LV,DT, @

D：D, LV,DT

指令格式：[DEXPP　S　D]

编程示例：

X1 产生上升沿脉冲触发 DEXPP 指令执行一次，将 D11D10 的浮点数 4.0000 进行指数运算，运算后的值为 54.5981，传送到 D1D0。

DLOGE

指令说明：

DLOGE 是 32 位连续执行型 2 进制浮点数自然对数运算指令，即每个扫描周期执行一次运算。是将源操作数[S+1,S]执行自然对数（以 e（2.71828）为底时的对数）运算，并将运算结果保存到目的操作数[D+1,D]中。

在[S+1,S]中指定的值，只可以设定正数，负数不能运算。

操作数：

S：D,K,H, LV,DT, @

D：D, LV,DT

指令格式：[DLOGE　S　D]

编程示例：

X1 闭合，K10 传送到 D0 经过浮点数转换后变为 10.000 后传送到 D11D10，X1 闭合时，DLOGE 指令执行，对 D11D10 的浮点数 10.0000 进行自然对数运算，运算后的结果为 2.3026，传送到 D21D20 中。

DLOGEP

指令说明：

DLOGE 是 32 位脉冲执行型 2 进制浮点数自然对数运算指令，即指令激活一次，执行一

次运算。是将源操作数[S+1,S]执行自然对数（以 e（2.71828）为底时的对数）运算，并将运算结果保存到目的操作数[D+1,D]中。

在[S+1,S]中指定的值，只可以设定正数，负数不能运算。

操作数：

S：D,K,H, LV,DT, @

D：D, LV,DT

指令格式：[DLOGEP　S　D]

编程示例：

X1 产生上升沿脉冲触发 DLOGEP 指令执行一次，将 D11D10 的浮点数 10.0000 进行自然对数运算，运算后的值为 2.3026，传送到 D21D20 中。

DLOG

指令说明：

DLOG 是 32 位连续执行型 2 进制浮点数常用对数运算指令，即每个扫描周期执行一次运算。是将源操作数[S+1,S]执行常用对数（以 10 为底时的对数）运算，并将运算结果保存到目的操作数[D+1,D]中。

在[S+1,S]中指定的值，只可以设定正数，负数不能运算。

操作数：

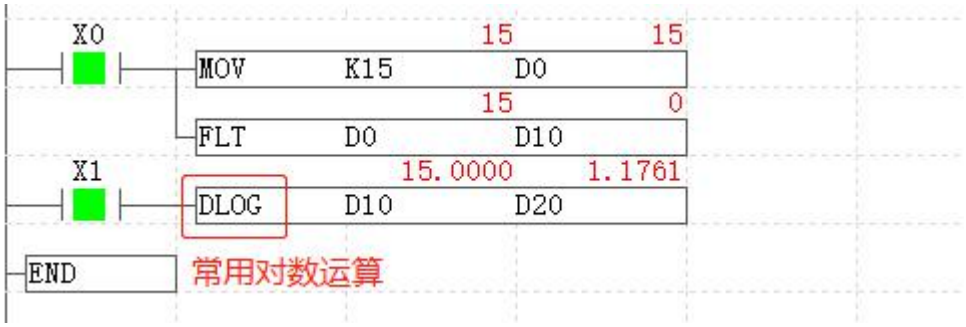
S：D,K,H, LV,DT,@

D：D, LV,DT

指令格式：[DLOG　S　D]

编程示例：

X0 闭合，K15 传送到 D0 经过浮点数转换后变为 15.0000 后传送到 D11D10，X1 闭合，DLOG 指令执行，对 D11D10 的浮点数 15.0000 进行常用对数运算，运算后的结果为 1.1761，传送到 D21D20 中。



DLOGP

指令说明：

DLOG 是 32 位脉冲执行型 2 进制浮点数常用对数运算指令，即指令激活一次，执行一次运算。是将源操作数[S+1,S]执行常用对数（以 10 为底时的对数）运算，并将运算结果保存到目的操作数[D+1,D]中。

在[S+1,S]中指定的值，只可以设定正数，负数不能运算。

操作数：

S：D,K,H, LV,DT,@

D：D, LV,DT

指令格式：[DLOGP　S　D]

编程示例：

X1 产生上升沿脉冲触发 DLOGP 指令执行，对 D11D10 的浮点数 15.0000 进行常用对数运算，运算后的结果为 1.1761，传送到 D21D20 中。

DASIN

指令说明：

DASIN 是 32 位连续执行型 2 进制浮点数 SIN^{-1}（反正弦）运算指令，即每个扫描周期执行一次运算。是将源操作数[S+1,S]的 SIN 值转换成角度值，并将其运算结果保存到目标操作数[D+1,D]中。

[S+1,S]的 SIN 值，可以在-1.0~1.0 的范围内设定。

[D+1,D]中保存的角度（运算结果）是保存弧度（-π/2）~（π/2）的值。

操作数：

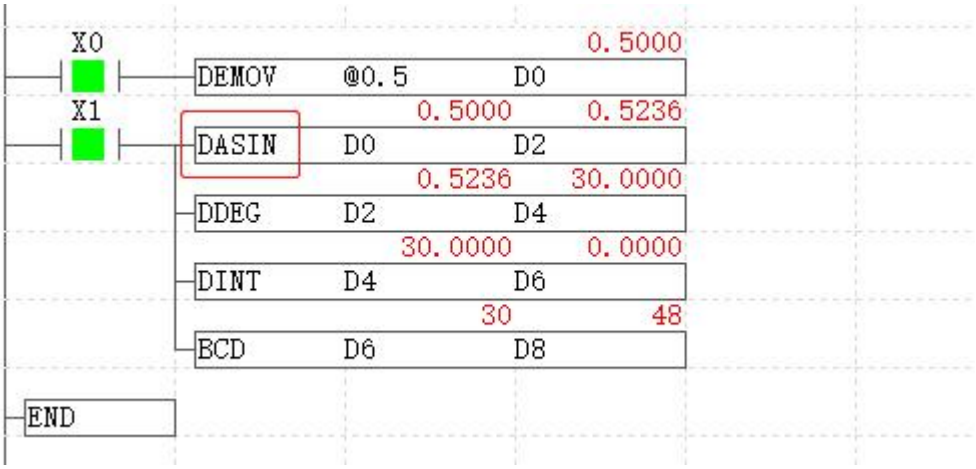
S：D,K,LV,DT,@

D：D,LV,DT

指令格式：[DASIN　S　D]

编程示例：

X1 闭合，DASIN 指令执行，D1D0 的值为 0.5000，通过 SIN^{-1} 运算，算出角度（弧度值）为 0.5236，并保存到 D3D2 中。DDEG 指令执行，将 D3D2 的弧度值转换成角度为 30.0000，并保存到 D5D4 中。DINT 指令执行，将 D5D4 的 2 进制浮点数的角度转换成整数值为 30，并保存到 D7D6 中。BCD 指令执行，将 D7D6 的整数值的角度 30 转换成 BCD 值 48，经过显示器 BIN 自动转换后，在显示器上显示的值为 0030。



DASINP

指令说明：

DASINP 是 32 位脉冲执行型 2 进制浮点数 SIN^{-1}（反正弦）运算指令，即指令激活一次，执行一次运算。是将源操作数[S+1,S]的 SIN 值转换成角度值，并将其运算结果保存到目标操作数[D+1,D]中。

[S+1,S]的 SIN 值，可以在-1.0~1.0 的范围内设定。

[D+1,D]中保存的角度（运算结果）是保存弧度（-π/2）~（π/2）的值。

操作数：

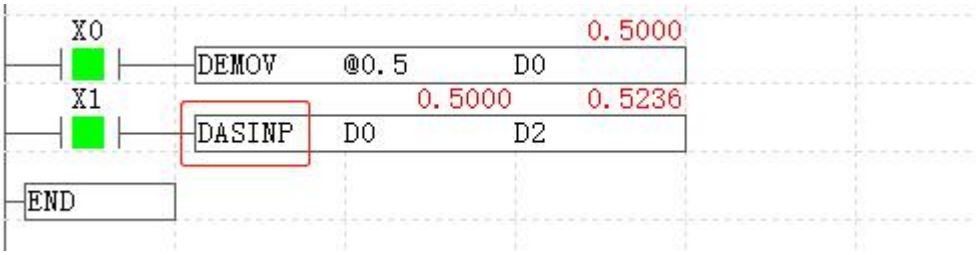
S：D,K,LV,DT,@

D：D,LV,DT

指令格式：[DASINP　S　D]

编程示例：

X1 产生上升沿脉冲触发 DASINP 指令执行一次，将 D1D0 的 SIN 值 0.5000 转换成角度（弧度值）0.5236，并保存到 D3D2 中。



DACOS

指令说明：

DACOS 是 32 位连续执行型 2 进制浮点数 COS^{-1}（反余弦）运算指令，即每个扫描周期执行一次运算。是将源操作数[S+1,S]的 COS 值转换成角度值，并将其运算结果保存到目标操作数[D+1,D]中。

[S+1,S]的 COS 值，可以在-1.0~1.0 的范围内设定。

[D+1,D]中保存的角度（运算结果）是保存弧度 0~π的值。

操作数：

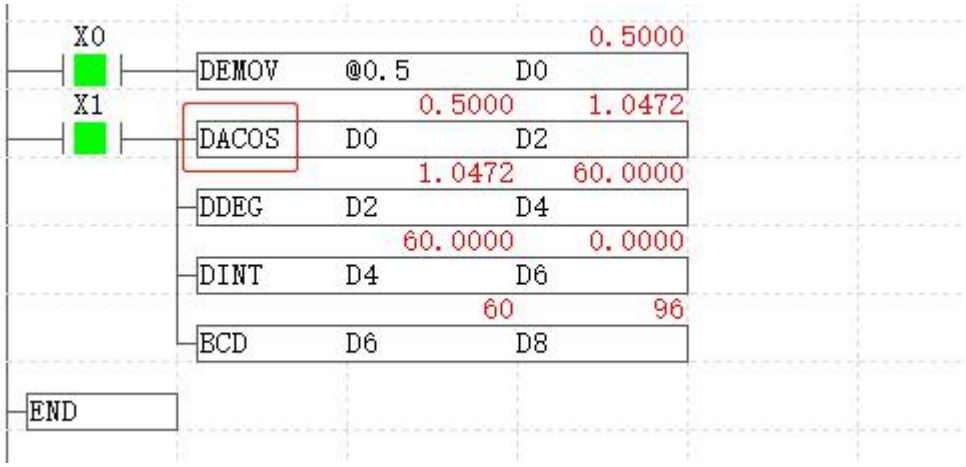
S：D,K,LV,DT, @

D：D,LV,DT

指令格式：[DACOS　S　D]

编程示例：

X1 闭合，DACOS 指令执行，D1D0 的值为 0.5000，通过 COS^{-1}运算，算出角度（弧度值）为 1.0472，并保存到 D3D2 中。DDEG 指令执行，将 D3D2 的弧度值转换成角度为 60.0000，并保存到 D5D4 中。DINT 指令执行，将 D5D4 的 2 进制浮点数的角度转换成整数值为 60，并保存到 D7D6 中。BCD 指令执行，将 D7D6 的整数值的角度 60 转换成 BCD 值 96，经过显示器 BIN 自动转换后，在显示器上显示的值为 0060。



DACOSP

指令说明：

DACOSP 是 32 位脉冲执行型 2 进制浮点数 COS^{-1}（反余弦）运算指令，即指令激活一次，执行一次运算。是将源操作数[S+1,S]的 COS 值转换成角度值，并将其运算结果保存到目标操作数[D+1,D]中。

[S+1,S]的 COS 值，可以在-1.0~1.0 的范围内设定。

[D+1,D]中保存的角度（运算结果）是保存弧度 0~π的值。

操作数：

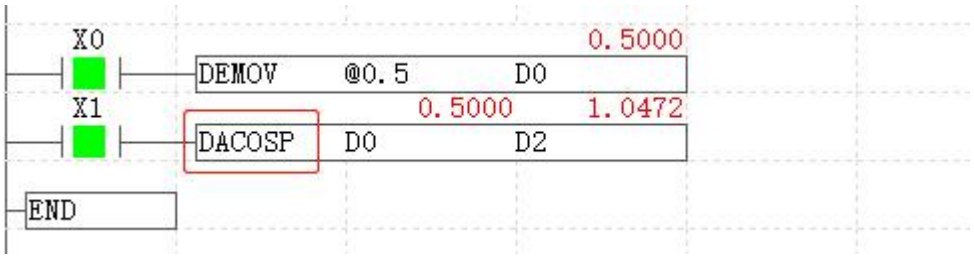
S：D,K,LV,DT,@

D：D,LV,DT

指令格式：[DACOSP　S　D]

编程示例：

X1 产生上升沿脉冲触发 DACOSP 指令执行一次，将 D1D0 的 COS 值 0.5000 转换成角度（弧度值）1.0472，并保存到 D3D2 中。



DATAN

指令说明：

DATAN 是 32 位连续执行型 2 进制浮点数 TAN^{-1}（反正切）运算指令，即每个扫描周期执行一次运算。是将源操作数[S+1,S]的 TAN 值转换成角度值，并将其运算结果保存到目标操作数[D+1,D]中。

[D+1,D]中保存的角度（运算结果）是保存弧度（-π/2）~（π/2）的值。

操作数：

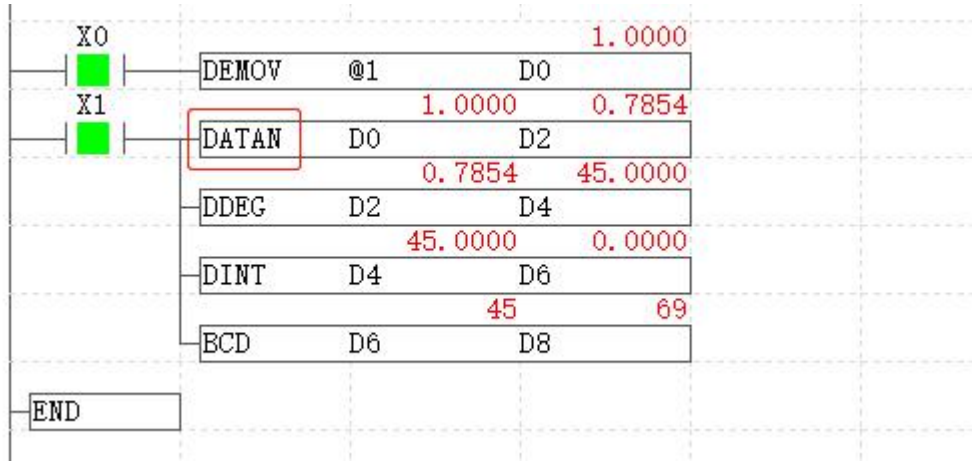
S：D,K,LV,DT,@

D：D,LV,DT

指令格式：[DATAN S D]

编程示例：

X1 闭合，DATAN 指令执行，D1D0 的值为 1.0000，通过 TAN^{-1}运算，算出角度（弧度值）为 0.7854，并保存到 D3D2 中。DDEG 指令执行，将 D3D2 的弧度值转换成角度为 45.0000，并保存到 D5D4 中。DINT 指令执行，将 D5D4 的 2 进制浮点数的角度转换成整数值为 45，并保存到 D7D6 中。BCD 指令执行，将 D7D6 的整数值的角度 45 转换成 BCD 值 69，经过显示器 BIN 自动转换后，在显示器上显示的值为 0045。



DATANP

指令说明：

DATANP 是 32 位脉冲执行型 2 进制浮点数 TAN^{-1}（反正切）运算指令，即指令激活一次，执行一次运算。是将源操作数[S+1,S]的 TAN 值转换成角度值，并将其运算结果保存到

目标操作数[D+1,D]中。

[D+1,D]中保存的角度（运算结果）是保存弧度（-π/2）~（π/2）的值。

操作数：

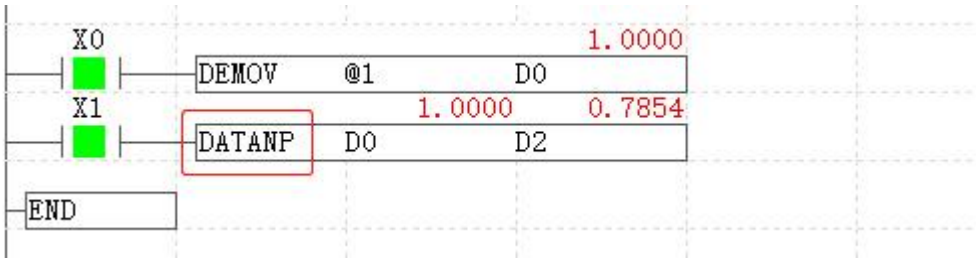
S：D,K,LV,DT,@

D：D,LV,DT

指令格式：[DATANP　S　D]

编程示例：

X1 产生上升沿脉冲触发 DATANP 指令执行一次，将 D1D0 的 TAN 值 1.0000 转换成角度（弧度值）0.7854，并保存到 D3D2 中。



DRAD

指令说明：

DRAD 是 32 位连续执行型 2 进制浮点数角度转换弧度的指令，即每个扫描周期执行一次运算。是将源操作数[S+1,S]的角度值转换成弧度值，并保存到目标操作数[D+1,D]中。

转换公式为弧度=角度×π/180

操作数：

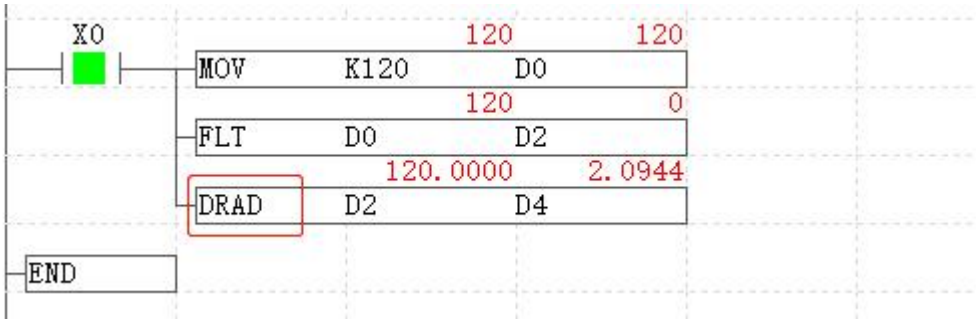
S：D,K,LV,DT,@

D：D,LV,DT

指令格式：[DRAD　S　D]

编程示例：

X0 闭合，MOV 指令将 K120 传送给 D0，FLT 指令执行，将 D0 的整数值 120 转换成浮点数 120.0000，并保存到 D3D2 中。DRAD 指令执行，将 D3D2 的浮点数角度值 120.000 转换成弧度值 2.0944，并保存到 D5D4 中。



DRADP

指令说明：

DRADP 是 32 位脉冲执行型 2 进制浮点数角度转换弧度的指令，即指令激活一次，执行一次运算。是将源操作数[S+1,S]的角度值转换成弧度值，并保存到目标操作数[D+1,D]中。

转换公式为弧度=角度×π/180

操作数：

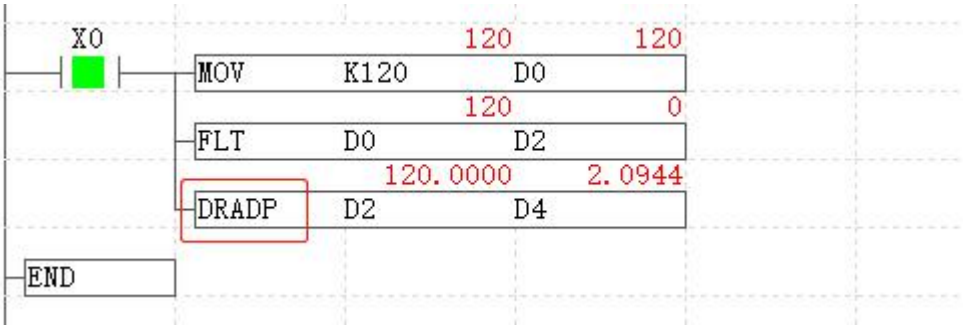
S：D,K,LV,DT,@

D：D,LV,DT

指令格式：[DRADP　S　D]

编程示例：

X0 闭合，MOV 指令将 K120 传送给 D0，FLT 指令执行，将 D0 的整数值 120 转换成浮点数 120.000，并保存到 D3D2 中。X1 产生上升沿脉冲触发 DRADP 指令执行一次，将 D3D2 的浮点数角度值 120.0000 转换成弧度值 2.9044，并保存到 D5D4 中。



DDEG

指令说明：

DDEG 是 32 位连续执行型 2 进制浮点数弧度转换角度的指令，即每个扫描周期执行一次运算。是将源操作数[S+1,S]的弧度值转换成角度值，并保存到目标操作数[D+1,D]中。

转换公式为角度=弧度×180/π

操作数：

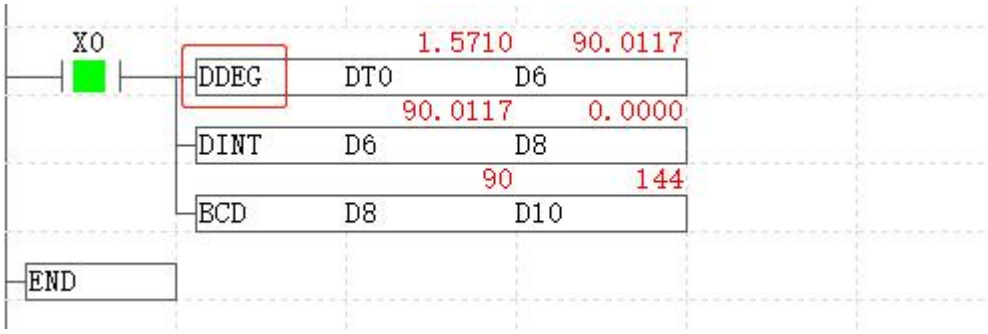
S：D,K,LV,DT,@

D：D,LV,DT

指令格式：[DDEG　S　D]

编程示例：

X0 闭合，DDEG 指令执行，将 DT0 的浮点数弧度值 1.5710 转换成角度值 90.0117，并保存到 D7D6 中。DINT 指令执行，将 D7D6 的浮点数角度值 90.0117 转换成整数绝对值 90，并保存到 D9D8 中。BCD 指令执行，将 D9D8 的 BIN 整数角度值 90 转换成 BCD 值 144，经过显示器 BIN 自动转换后，在显示器上显示的值为 0090。



DDEGP

指令说明：

DDEGP 是 32 位脉冲执行型 2 进制浮点数弧度转换角度的指令，即指令激活一次，执行一次运算。是将源操作数[S+1,S]的弧度值转换成角度值，并保存到目标操作数[D+1,D]中。

转换公式为角度=弧度×180/π

操作数：

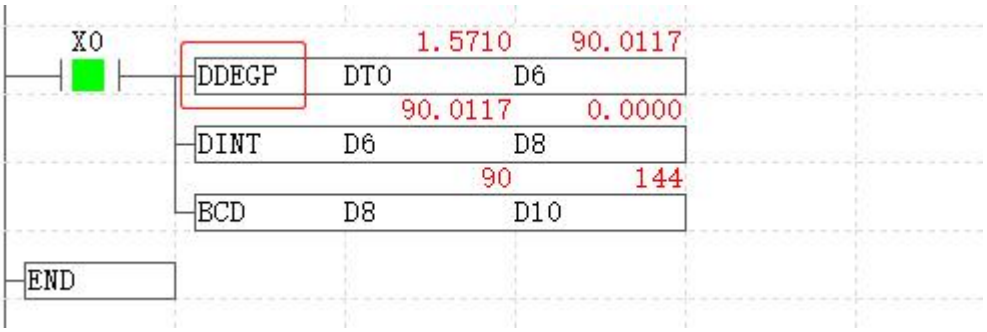
S：D,K,LV,DT,@

D：D,LV,DT

指令格式：[DDEGP　S　D]

编程示例：

X1 产生上升沿脉冲触发 DDEGP 指令执行一次，将 DT0 的浮点数弧度值 1.5710 转换成角度值 90.0117，并保存到 D7D6 中。



DSINH

指令说明：

DSINH 是 32 位连续执行型 2 进制浮点数 SINH 运算指令，即每个扫描周期执行一次运算。是将源操作数[S+1,S]的浮点数数据换算成 SINH 值，并保存到目标操作数[D+1,D]中。

转换公式为 SINH 值=（$e^{x}-e^{-x}$）/2

x 表示操作数[S]，e=2.71828 为底数

操作数：

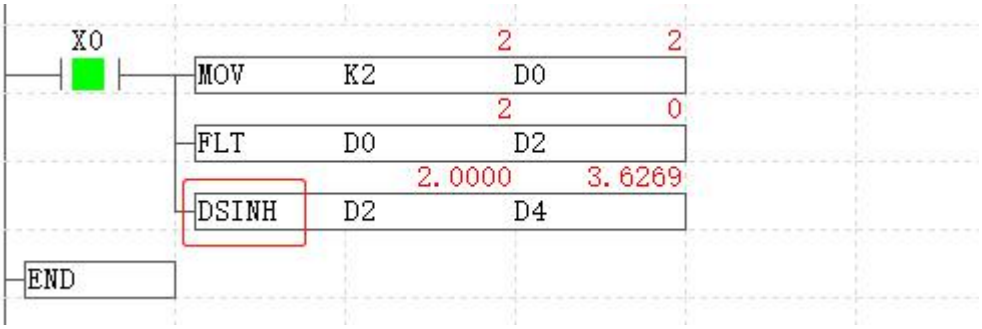
S：D,K,H,LV,DT, @

D：D,LV,DT

指令格式：[DSINH　S　D]

编程示例：

X0 闭合，FLT 指令执行，将 D0 的整数值 2 转换成浮点数 2.0000，并保存到 D3D2 中。DSINH 指令执行，将 D3D2 的浮点数 2.000 进行 SINH 运算后得到的值为 3.6269，并保存到 D5D4 中。



DSINHP

指令说明：

DSINHP 是 32 位脉冲执行型 2 进制浮点数 SINH 运算指令，即指令激活一次，执行一次运算。是将源操作数[S+1,S]的浮点数数据换算成 SINH 值，并保存到目标操作数[D+1,D]中。

转换公式为 SINH 值=（$e^{x}-e^{-x}$）/2

x 表示操作数[S]，e=2.71828 为底数

操作数：

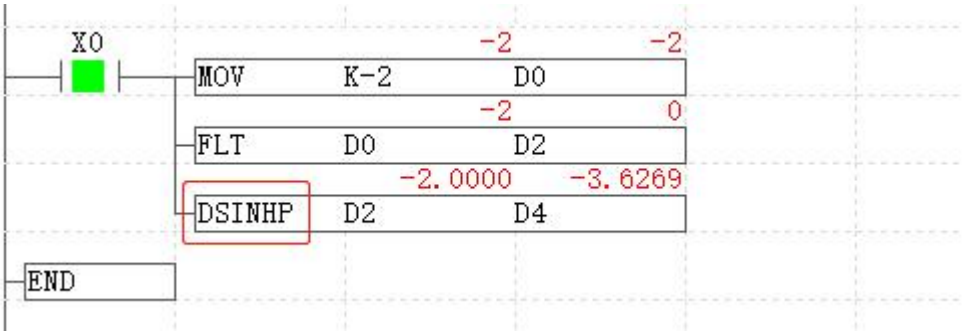
S：D,K,H,LV,DT, @

D：D,LV,DT

指令格式：[DSINHP　S　D]

编程示例：

X0 产生上升沿脉冲触发 DSINHP 指令执行一次，将 D3D2 的浮点数-2.000 经过 SINH 运算后得出-3.6269，并保存到 D5D4 中。



DCOSH

指令说明：

DCOSH 是 32 位连续执行型 2 进制浮点数 COSH 运算指令，即每个扫描周期执行一次

运算。是将源操作数[S+1,S]的浮点数数据换算成 COSH 值，并保存到目标操作数[D+1,D]中。

转换公式为 COSH 值=（e^x+e^{-x}）/2

x 表示操作数[S]，e=2.71828 为底数

操作数：

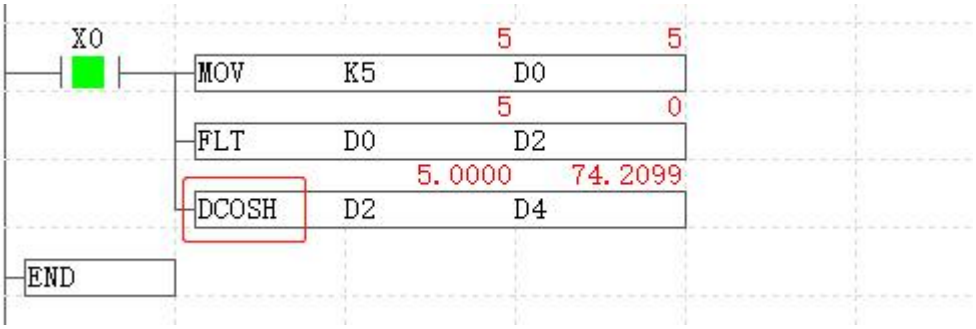
S：D,K,H,LV,DT, @

D：D,LV,DT

指令格式：[DCOSH　S　D]

编程示例：

X0 闭合，FLT 指令执行，将 D0 的整数值 5 转换成浮点数 5.0000，并保存到 D3D2 中。DCOSH 指令执行，将 D3D2 的浮点数 5.0000 进行 COSH 运算后得到的值为 74.2099，并保存到 D5D4 中。



DCOSHP

指令说明：

DCOSH 是 32 位脉冲执行型 2 进制浮点数 COSH 运算指令，即指令激活一次，执行一次运算。是将源操作数[S+1,S]的浮点数数据换算成 COSH 值，并保存到目标操作数[D+1,D]中。

转换公式为 COSH 值=（e^x+e^{-x}）/2

x 表示操作数[S]，e=2.71828 为底数

操作数：

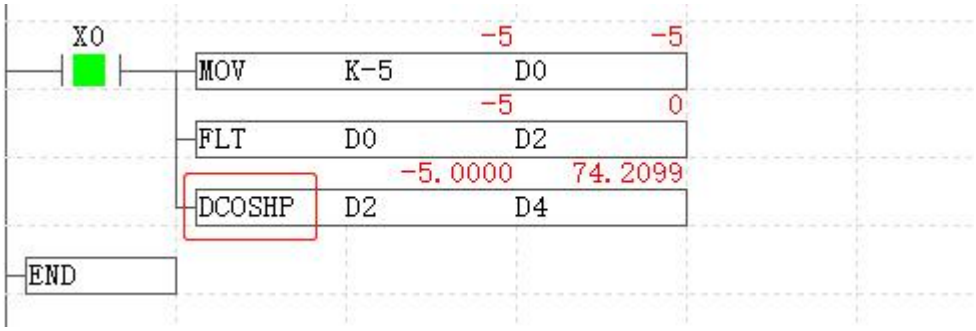
S：D,K,H,LV,DT, @

D：D,LV,DT

指令格式：[DCOSHP　S　D]

编程示例：

X0 产生上升沿脉冲触发 DCOSHP 指令执行一次，将 D3D2 的浮点数-5.0000 经过 COSH 运算后得出 74.2099，并保存到 D5D4 中。



DTANH

指令说明：

DTANH 是 32 位连续执行型 2 进制浮点数 TANH 运算指令，即每个扫描周期执行一次运算。是将源操作数[S+1,S]的浮点数数据换算成 TANH 值，并保存到目标操作数[D+1,D]中。

转换公式为 TANH 值=（$e^x - e^{-x}$）/（$e^x + e^{-x}$）

x 表示操作数[S]，e=2.71828 为底数

操作数：

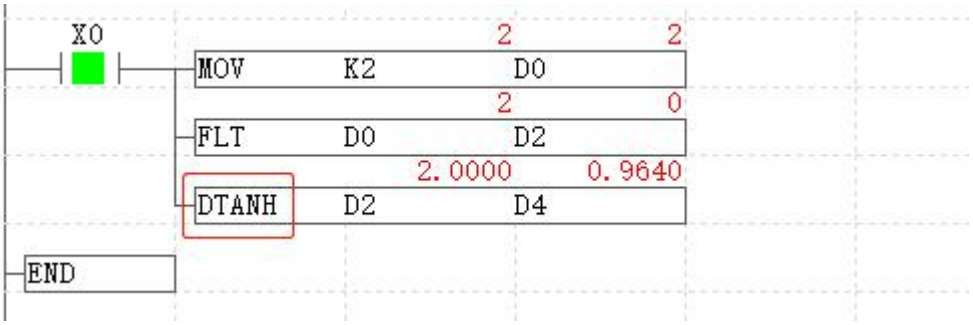
S：D,K,H,LV,DT, @

D：D,LV,DT

指令格式：[DTANH　S　D]

编程示例：

X0 闭合，FLT 指令执行，将 D0 的整数值 2 转换成浮点数 2.0000，并保存到 D3D2 中。DTANH 指令执行，将 D3D2 的浮点数 2.0000 进行 TANH 运算后得到的值为 0.9640，并保存到 D5D4 中。



DTANHP

指令说明：

DTANH 是 32 位脉冲执行型 2 进制浮点数 TANH 运算指令，即指令激活一次，执行一次运算。是将源操作数[S+1,S]的浮点数数据换算成 TANH 值，并保存到目标操作数[D+1,D]中。

转换公式为 TANH 值=（$e^x - e^{-x}$）/（$e^x + e^{-x}$）

x 表示操作数[S]，e=2.71828 为底数

操作数：

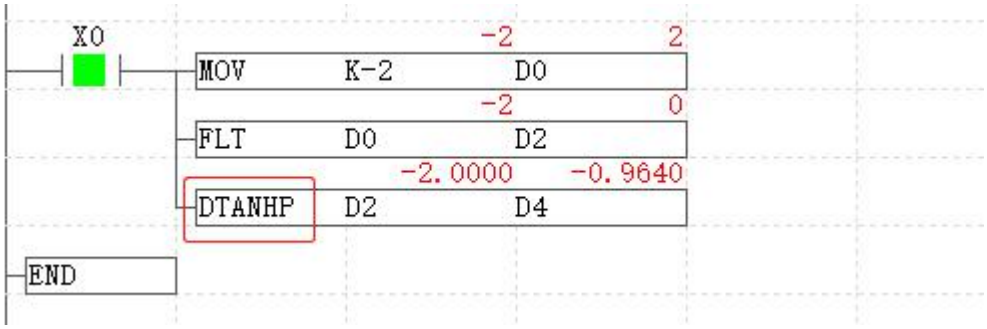
S：D,K,H,LV,DT, @

D：D,LV,DT

指令格式：[DTANHP　S　D]

编程示例：

X1 产生上升沿脉冲触发 DTANHP 指令执行一次，将 D3D2 的浮点数-2.0000 经过 TANH 运算后得出-0.9640，并保存到 D5D4 中。



3.9 其它指令

指令	功能	操作数类型
BASE	指定要参与运动的轴	S：KnX, KnY, KnM, KnS,T,C,D,K,H,Z,V,LV,DT,@
MOVE	直线插补相对运动	S：T,C,D,K,H,Z,V,LV,DT,@
MOVEABS	直线插补绝对运动	S：T,C,D,K,H,Z,V,LV,DT,@
VMOVE	连续向一个方向运动	S：K (-1 为负方向/1 正方向)
DATUM	轴按指定模式回零	S：K,H,D,Z,V,LV,DT,@

BASE

指令说明：

BASE 指令指定要参与运动的轴。一次最多指定 8 个轴。

操作数：

S1：KnX, KnY, KnM, KnS,T,C,D,K,H,Z,V,LV,DT,@

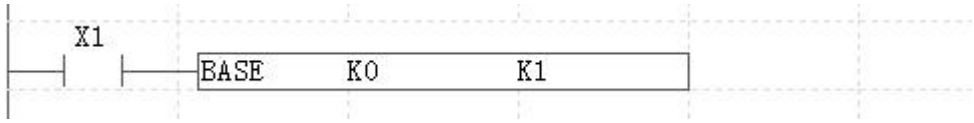
S2：KnX, KnY, KnM, KnS,T,C,D,K,H,Z,V,LV,DT,@

……

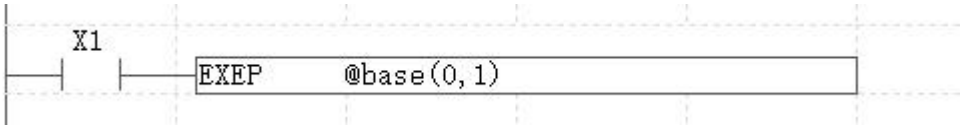
S8：KnX, KnY, KnM, KnS,T,C,D,K,H,Z,V,LV,DT,@

编程示例：

1、直接使用 PLC 指令中的 BASE 指令：



2、在 PLC 中调用 Basic 中的 BASE 指令：



MOVE

指令说明：

MOVE 指令为直线插补相对运动指令。

操作数：

S1： T,C,D,K,H,Z,V,LV,DT,@

S2： T,C,D,K,H,Z,V,LV,DT,@

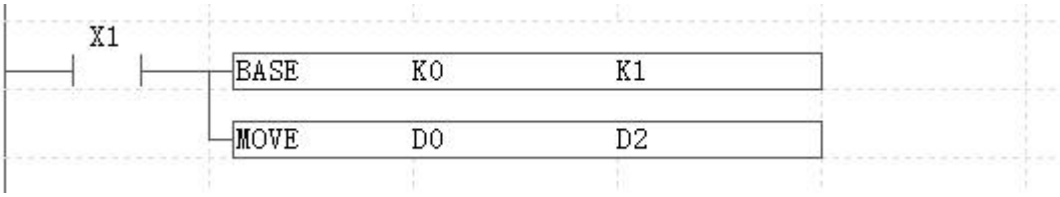
……

S8： T,C,D,K,H,Z,V,LV,DT,@

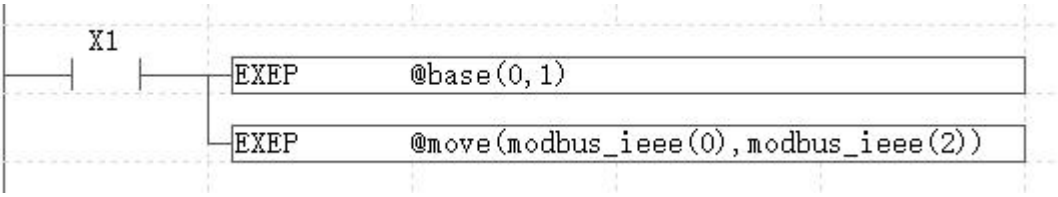
操作数若使用字寄存器，寄存器中的数值需用浮点指令赋值。

编程示例：

1、直接使用 PLC 指令中的 MOVE 指令：



2、在 PLC 中调用 Basic 中的 MOVE 指令：



注：两者对 MOVE 指令的调用存在着差别。使用 PLC 中的 MOVE 指令时，若 MOVE 运动没有完成就断开 X1，则 MOVE 运动立即停止；使用 Basic 中的 MOVE 指令时，即使 MOVE 运动没有完成就断开 X1，该 MOVE 运动也会继续执行直到完成。

MOVEABS

指令说明：

MOVEABS 指令为直线插补绝对运动指令。

操作数：

S1： T,C,D,K,H,Z,V,LV,DT,@

S2： T,C,D,K,H,Z,V,LV,DT,@

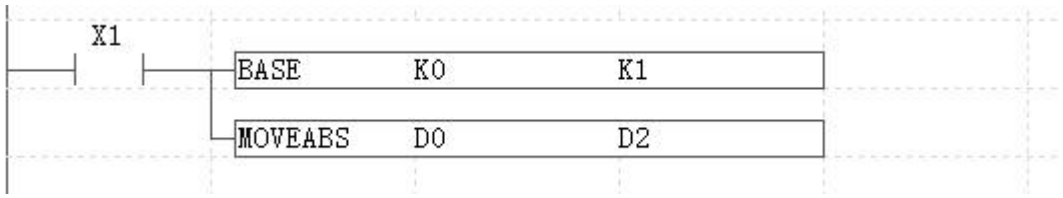
……

S8： T,C,D,K,H,Z,V,LV,DT,@

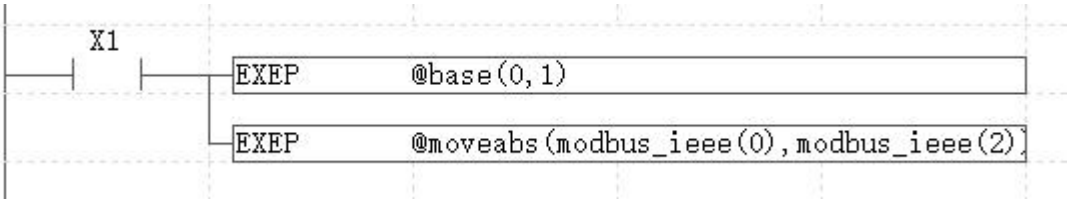
操作数若使用字寄存器，寄存器中的数值需用浮点指令赋值。

编程示例：

1、直接使用 PLC 指令中的 MOVEABS 指令：



2、在 PLC 中调用 Basic 中的 MOVEABS 指令：



注：两者对 MOVEABS 指令的调用存在着差别。使用 PLC 中的 MOVEABS 指令时，若 MOVEABS 运动没有完成就断开 X1，则 MOVEABS 运动立即停止；使用 Basic 中的 MOVEABS 指令时，即使 MOVEABS 运动没有完成就断开 X1，该 MOVEABS 运动也会继续执行直到完成。

VMOVE

指令说明：

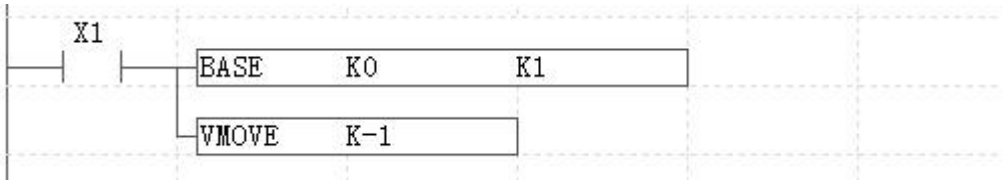
VMOVE 指令为连续运动指令指令，即连续向一个方向运动。

操作数：

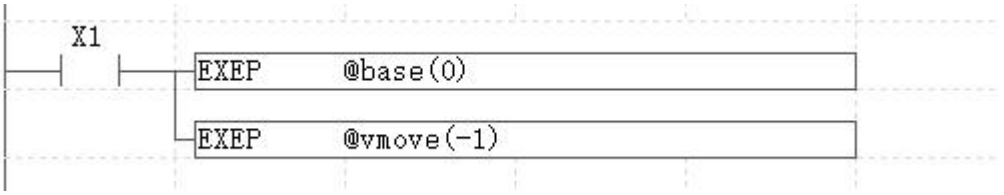
S：K (-1 为负方向/1 正方向)

编程示例：

1、直接使用 PLC 指令中的 VMOVE 指令：



2、在 PLC 中调用 Basic 中的 VMOVE 指令：



注：两者对 VMOVE 指令的调用存在着差别。使用 PLC 中的 VMOVE 指令时，若 VMOVE 运动没有完成就断开 X1，则 VMOVE 运动立即停止；使用 Basic 中的 VMOVE 指令时，即使 VMOVE 运动没有完成就断开 X1，该 VMOVE 运动也会继续执行直到完成。

DATUM

指令说明：

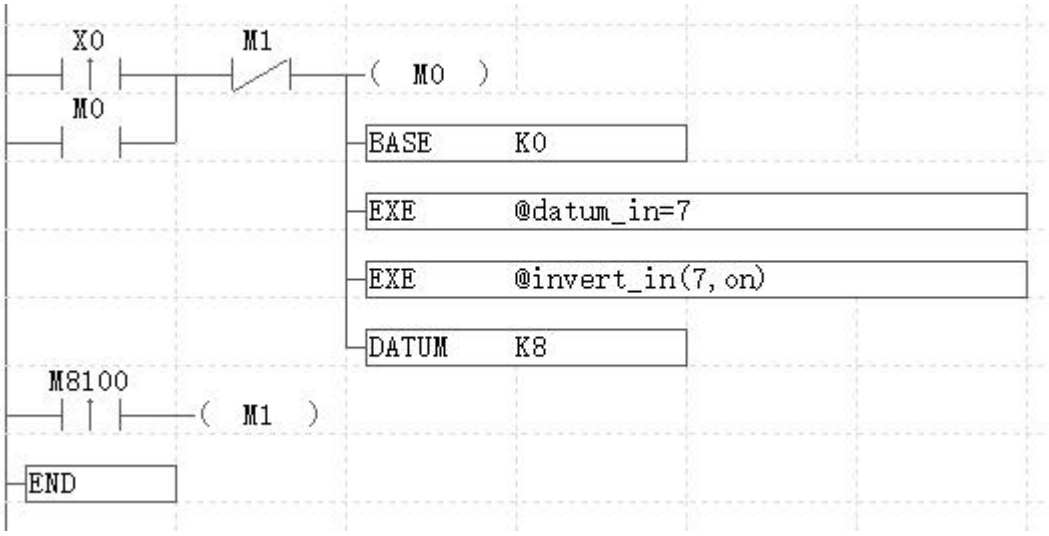
DATUM 指令为轴回零指令，一次操作一个轴回零，多轴回零时，需要对每个轴都使用 DATUM 指令。

操作数：

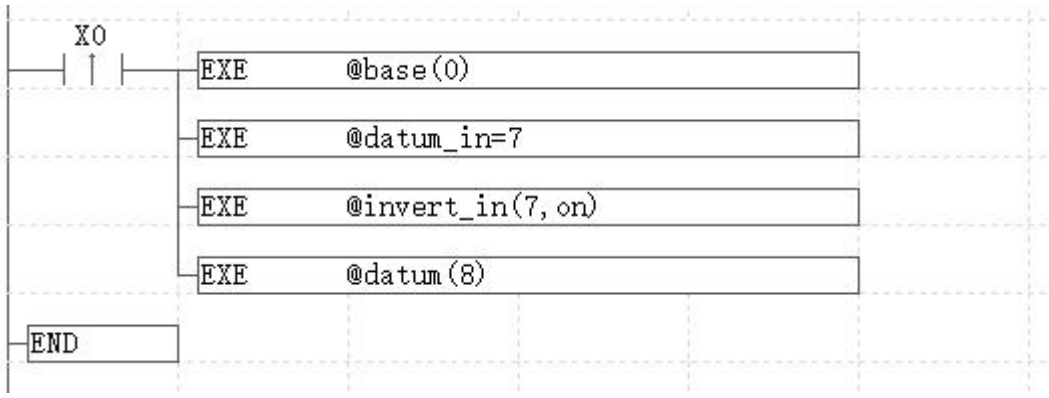
S：D,K,H,Z,V,LV,DT,@

编程示例：

1、 直接使用 PLC 指令中的 DATUM 指令：



2、在 PLC 中调用 Basic 中的 DATUM 指令：



注：两者对 DATUM 指令的调用存在着差别。使用 PLC 中的 DATUM 指令时，若 DATUM 运动没有完成就断开 X1，则 DATUM 运动立即停止；使用 Basic 中的 DATUM 指令时，即使 DATUM 运动没有完成就断开 X1，该 DATUM 运动也会继续执行直到完成。

第四章 模块扩展

当控制器自身的轴资源、IO 资源不够用时，可采用扩展模块来扩展，可以扩展脉冲轴、数字量输入输出、模拟量输入输出这三种类型。只有带脉冲轴接口的扩展模块才支持扩展脉冲轴数，总线轴不可扩展。

可采用 CAN 总线或 EtherCAT 总线两种方式连接扩展模块，扩展模块分为 ZCAN 扩展模块，EtherCAT 扩展模块、ZMIO300 扩展模块三大类。

扩展的资源必须映射到控制器本地资源才可使用。

IO 数字量扩展：4 系列及以上的 ZMC 控制器 IO 点数最多可扩展至 4096 点。

AIO 模拟量扩展：4 系列及以上的 ZMC 控制器 AIO 点数最多可扩展至 520 点。

轴扩展：只能扩展脉冲轴，鉴于成本和使用方面，不建议使用过多轴扩展板，可选用支持脉冲轴数较多的控制器型号。

控制器可扩展的最大 IO 点数可在“命令与输出”窗口输入?*max 打印查看。

4.1 ZCAN 扩展模块

ZCAN 扩展模块包括 ZIO 系列和 ZAIO 系列，型号参数如下。

产品型号	电机轴数	编码器数	总轴数	内部输入/输出口	AD	DA
ZIO0808	-	-	-	8/8	-	-
ZIO0016	-	-	-	0/16	-	-
ZIO1608	-	-	-	16/8	-	-
ZIO1616	-	-	-	16/16	-	-
ZIO1632	-	-	-	16/32	-	-
ZAIO0802	-	-	-	-	8	2
ZIO16082	2	2	2	16/8	-	-

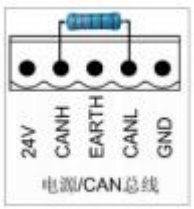
供电方式：只有主电源这一路需要供电的为单电源供电，主电源和 IO 接口均需要供电的为双电源供电。IO 接口电源与主电源分开供电，不得采用同一个电源，以防信号干扰。一个控制网络上的主电源接口可接在一个电源上，如下图所示，IO 电源亦是如此。

判断板块供电方式的方法：查看 OUT 输出端子上有无 24V 接口，有此端子则为双电源供电，否则为单电源供电。

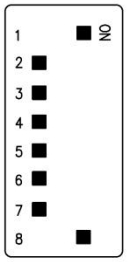
ZIO 系列扩展模块均采用双电源供电，ZAIO 模拟量扩展模块采用单电源供电。

CAN 总线通讯双方必须保证对应 GND 连上或是控制器主电源和扩展模块主电源用同一

个电源，并在扩展模块的 CANL 与 CANH 端并接一个 120 欧姆的电阻，防止扩展模块烧坏。V1.3 以上硬件版本带 8 位拨码开关的扩展模块，板上面集成了 120 欧姆电阻在 CANL 和 CANH 之间，由拨码 8 控制，拨 ON 时电阻接通，只需要把最末端扩展模块的拨码 8 拨 ON，不需要另外在端子外部接 120 欧电阻。

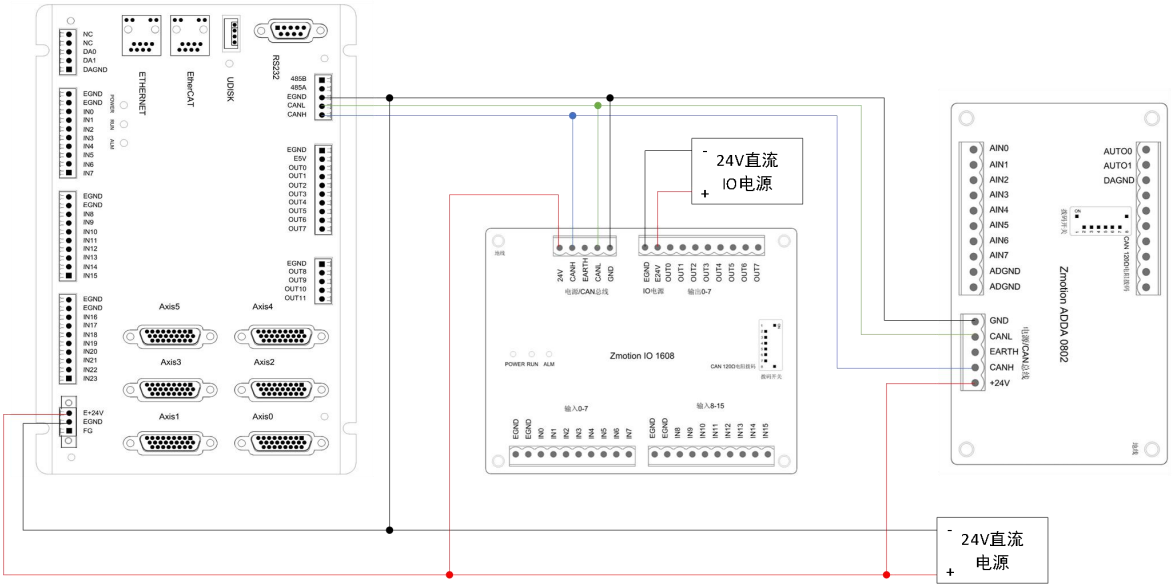


120 欧姆电阻接线



拨码开关第 8 位拨 ON

扩展模块接线参考图如下：



4.2 EtherCAT 总线扩展模块

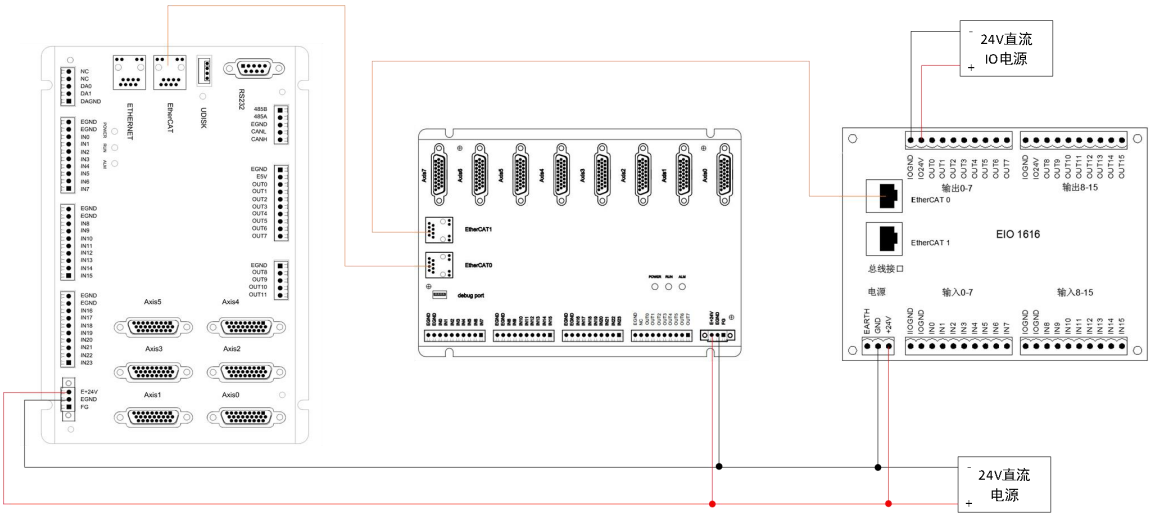
EtherCAT 扩展模块包括 EIO 系列和 ZMIO 系列，EIO 系列型号参数如下。

产品型号	电机轴数	编码器数	总轴数	内部输入/输出口	ECAT IN/OUT
EIO1616	-	-	-	16/16	1/1
EIO24088	8	8	8	24/8	1/1

EtherCAT 扩展接线较为简单，采用 EIO 扩展板，只需将各个模块的 EtherCAT 口相互连接即可，模块电源和 IO 电源的接法与 CAN 总线扩展模块相同。

连线参考如下图所示，控制器为 ZMC432，采用单电源供电，EIO24088 扩展板采用单电源供电，EIO1616 扩展板采用双电源供电。

ZMIO 扩展模块接线方法与 EIO 扩展模块一致。



4.3 扩展资源映射

接线完成还需要对扩展的资源进行映射才可使用。

轴映射方法参见《Zbasic 编程手册》的 AXIS_ADDRESS 轴映射指令。

IO/AIO 资源映射使用分两种。

一：带拨码开关的 CAN 总线扩展模块的 1-4 号拨码开关设置 CAN 地址，拨码每位 OFF 时对应值 0，ON 时对应值 1，组合值=拨码 4×8+拨码 3×4+拨码 2×2+拨码 1，数字量起始 IO 映射编号从 16 开始，按 16 的倍数递增；模拟量起始 IO 映射编号从 8 开始，按 8 的倍数递增，控制器根据 CAN 拨码地址来设定对应 IO 板的 IO 口范围。

拨码 1-4 选择 CAN 地址，数字量 IO 编号分配表：

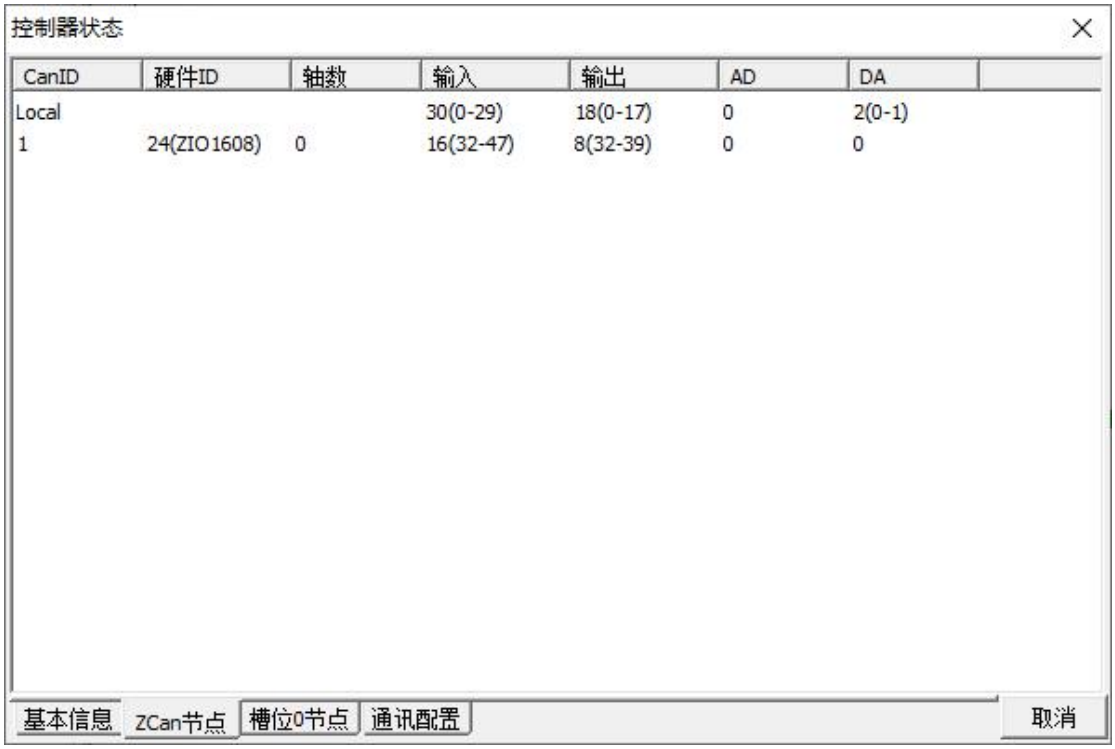
组合值	起始 IO 编号	结束 IO 编号
0	16	31
1	32	47
2	48	63
3	64	79
4	80	95
5	96	111
6	112	127
7	128	143
8	144	159
9	160	175
10	176	191
11	192	207
12	208	223
13	224	239
14	240	255

15	256	271

扩展版拨码开关根据当前已包含 IO 点数（IN 和 OP 最大者），如控制器本身包含 28 个 IN，16 个 OP，那么第一个扩展版设置的起始地址应超过最大值 28，按上表规则应将拨码设置为组合值 1（二进制组合值 0001，从右往左对应拨码 1-4，此时拨码 1 置 ON，其他置 OFF），此时扩展版上的 IO 编号为 32-47，其中，29-31 空缺出来的 IO 编号舍去不用。后续的扩展版则依次按 IO 点数继续确认拨码设置。

当控制器或扩展模块的 IO 编号范围重复时，只有一个有效。建议重新设置拨码使 IO 编号不重复。

例如：ZMC406 控制器连接一个 ZIO1608 扩展板，在软件菜单栏“控制器状态”-“ZCan 节点”查看扩展板连接情况，如下图所示，第一行为控制器本地资源信息，第二行为扩展板映射后的信息。CanID 表示拨码卡关组合值为 1，此时扩展板起始 IO 地址为 32。



二：EtherCAT 总线扩展模块使用 NODE_IO（数字量）、NODE_AIO（模拟量）指令设置设备的 IO 起始编号，设置值只能为 8 的倍数，对需要映射的每个设备进行一次指令设置。

注意映射编号要大于控制器自身 IO 点数，当控制器或扩展模块的 IO 编号范围重复时，只有一个有效，建议调整拨码使得编号不重复。

例如 ZMC432 连接两个 EIO1616 扩展板。

ZMC432 自带 IO 点数为 24 进 12 出，另外 6 个脉冲轴接口内各有 1 进一出，合计共 30 进 18 出，IO 映射点数至少从 32 开始。

映射后的 IO 资源编号如下：

型号	IO 映射	输入点范围	输出点范围
ZMC432	/	0-29	0-17
EIO1616	NODE_IO(0,0)=32	32-47	32-47
EIO1616	NODE_IO(0,1)=48	48-63	48-63

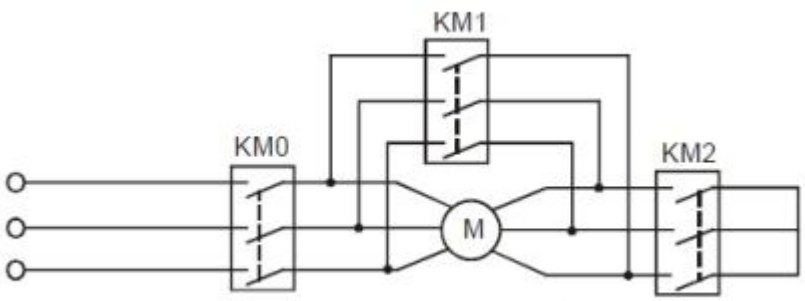
第五章 简易例程

5.1 星三角降压启动

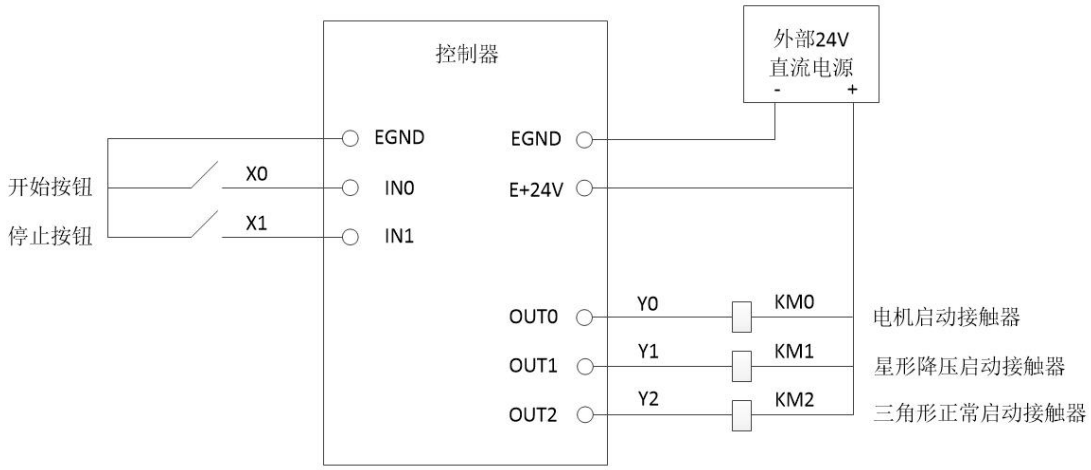
动作要求：

三相交流异步电动机启动时电流较大，一般为额定电流的 5~7 倍，为了减少启动电流对电网的影响，采用星-三角形降压启动方式。

星-三角形降压启动过程：合上总开关后，电机启动接触器 KM0 和星形降压方式启动接触器 KM1 先启动。10 秒延时后，星形降压方式启动接触器断开，再经过 3 秒延时后将三角形正常运行接触器 KM2 接通，电动机主电路接成三角形接法正常运行。采用两级延时的目的是确保星形降压方式启动接触器完全断开后才去接通三角形正常运行接触器。

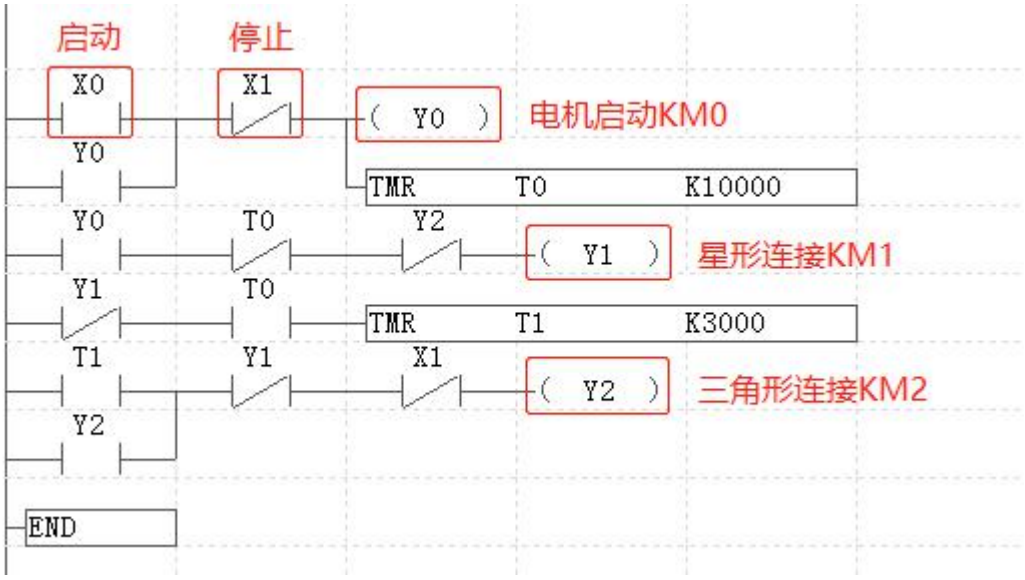


降压启动主电路示意图



控制器外部接线示意图

梯形图控制程序：



程序说明：

按下启动按钮 X0，X1=OFF，Y0（电机启动接触器 KM0）导通并自锁，定时器 T0 开始计时 10s，此时 T0=OFF，Y2=OFF，Y0 导通驱动 Y1（星形接触器 KM1）导通。

待 T0 计时完成后 T0 的常闭触点断开，使 Y1=OFF，T0 的常开触点导通，启动定时器 T1 计时 3s。

T1 计时完成后，T1 的常闭触点导通，驱动 Y2（三角形接触器 KM2）导通并自锁，直到按下停止按钮 X1，无论电动机处于启动状态还是运行状态，Y0、Y1、Y2 都变为 OFF，电机停止运行。

指令表程序如下：

```
LD X0
OR  Y0
ANI X1
OUT Y0
TMR T0 K10000
LD Y0
ANI T0
ANI Y2
OUT Y1
LDI Y1
AND     T0
TMR T1 K3000
LD T1
OR  Y2
ANI Y1
ANI X1
OUT Y2
END
```

5.2 直线插补运动

动作要求：

控制脉冲轴轴 0 和轴 1 直线插补运动。

控制程序：

程序说明：

程序上电初始化时，对轴的各种参数进行初始化。

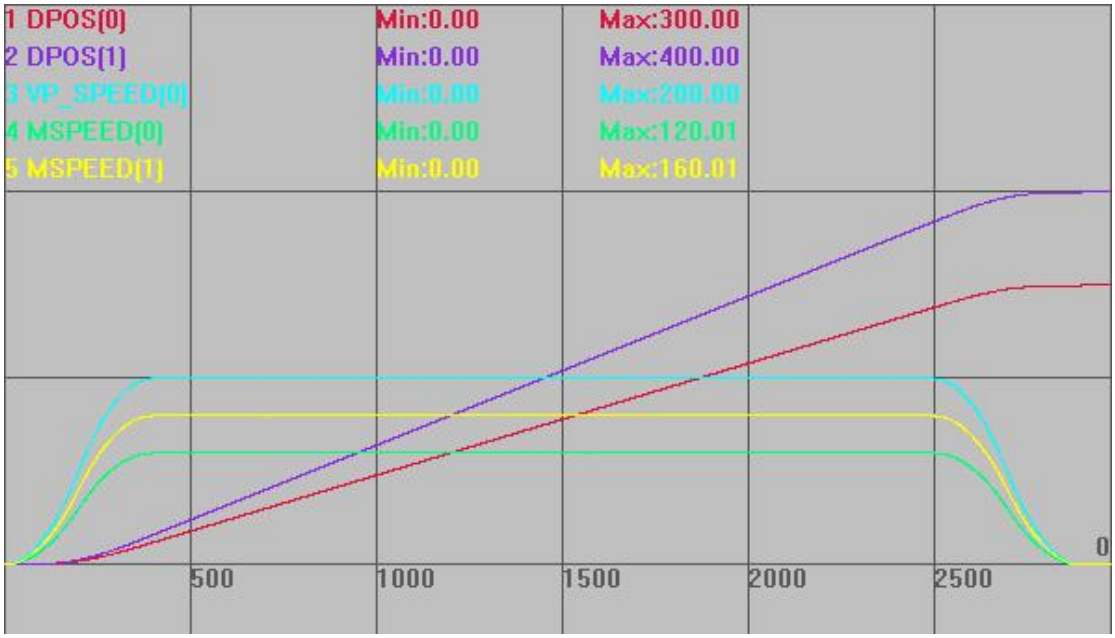
当 X0 上升沿触发时，对存储两个轴运动距离的寄存器 D0、D2 进行赋值，当 X1 上升沿触发启动示波器采样、开启 MOVE 直线插补运动并且 M0 自锁，轴 0 运动距离为 300，轴 1 运动距离为 400。

M8100 为轴 0 的 IDLE 标志，当运动完成时，轴 0 停止，M8100 变为 ON，M1 被置位一个周期，M1 的常闭触点断开一个周期，M0 自锁取消。

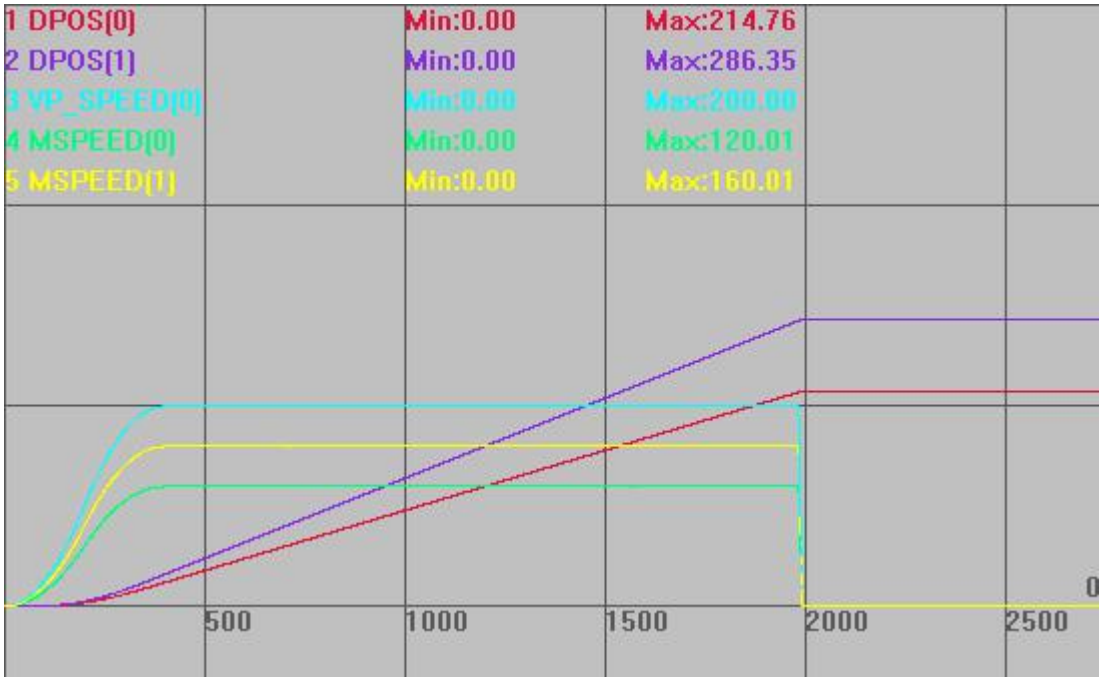
再次按下 X1，MOVE 再次执行轴 0 轴 1 直线插补运动。
X2 为急停按钮，若轴在运动中按下 X2，按 FASTDEC 快减速设置的值快速停下。
指令表程序如下：

```
//轴参数初始化
LD M8002
ZRST M0 M10
LD M8002
BASE k0 k1
EXE @UNITS= 100,100
EXE @ATYPE = 1,1
EXE @SPEED=200,200
EXE @ACCEL=1000,1000
EXE @DECEL=1000,1000
EXE @SRAMP=200,200
EXE @DPOS=0,0
EXE @MPOS=0,0
EXE @FASTDEC=20000,20000
//直线插补运动
LDP X0
DEMOV K300 D0
DEMOV K400   D2
// IN1 上升沿启动运行,IN2 按下运动快速停止
LDP X1
OR  M0
ANI M1
ANI X2
OUT M0
EXEP      @TRIGGER
MOVE D0 D2
LD M8100
PLS M1
END
```

示波器采集的插补运动波形如下所示：
曲线 0：轴 0 位置 DPOS(0)
曲线 1：轴 1 位置 DPOS(1)
曲线 2：轴 0 的插补合成速度 VP_SPEED(0)
曲线 3：轴 0 的分矢量速度 MSPEED(0)
曲线 4：轴 1 的分矢量速度 MSPEED(1)



X2 按下对应的急停曲线：



5.3 三点画圆弧

触摸屏端界面显示如下：

动作要求：

通过触摸屏按键移动 X、Y 轴坐标到合适位置，并定位圆弧中间点和结束点坐标。定位完成后，按启动键来实现指定圆弧动作，停止键可停止圆弧动作。状态栏实时显示当前系统状态。

控制器端程序介绍：

（1）上电运行，PLC 调用 Basic 子程序，初始化参数。

（2）D0，D2 里面储存的是轴 0、轴 1 的 DPOS 的值，也就是当前坐标。

（3）当 D100 的值等于 K1 时，记录圆弧运动的起点坐标；当 D100 的值等于 K2 时，记录圆弧运动的中点坐标；当 D100 的值等于 K3 时，记录圆弧运动的终点坐标。

（4）当轴 0、轴 1 的 IDLE 为-1 时，@的触点导通，D101 为 0，表示待机中；当 D101 不为 1，运动不是自动状态，M0,M1,M2,M3 任意一个导通，D101 为 2，表示运动处于手动状态。

（5）当 D101 不为 1，运动不是自动状态时，导通 M0，轴 0 正转；导通 M1，轴 0 反转；导通 M2，轴 1 正转；导通 M3，轴 1 反转。当 M0,M1,M2,M3 断开时，取消运动。

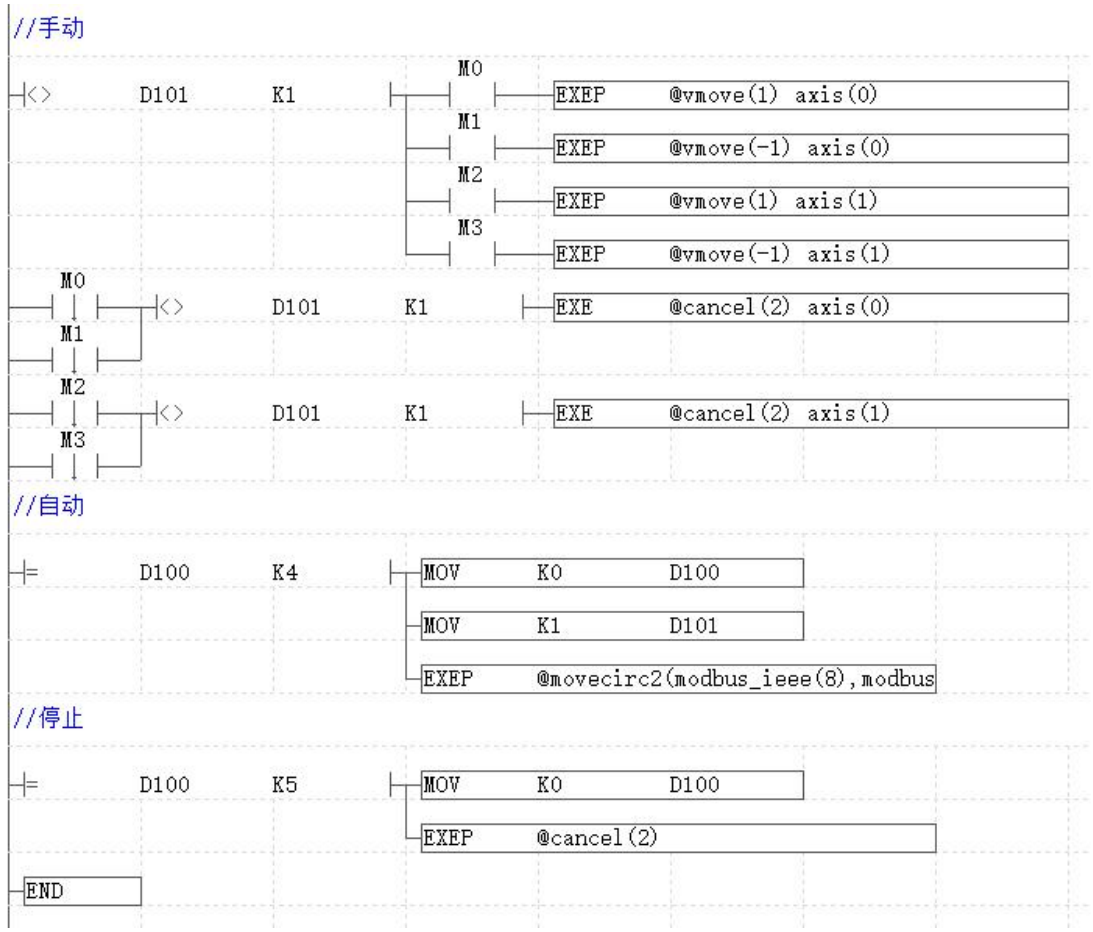
（6）当 D100 的值为 K4 时，D101 为 1，表示运动为自动状态，调用一次 MOVECIRC2，MOVECIRC2 为两轴圆弧插补，三点画弧，相对运动指令。

（7）当 D100 的值为 K5 时，停止轴 0、轴 1 的运动。

程序控制：

PLC 控制程序如下：


//初始化参数
M8002
CALL @InitSub
//显示当前坐标
M8000
DMOV D10000 D0
DMOV D10002 D2
//坐标记录
= D100 K1
MOV K0 D100
DMOV D10000 D4
DMOV D10002 D6
= D100 K2
MOV K0 D100
DMOV D10000 D8
DMOV D10002 D10
= D100 K3
MOV K0 D100
DMOV D10000 D12
DMOV D10002 D14
//状态切换
@idle(0)=-1@idle(1)=-1
MOV K0 D101
M0
<> D101 K1
MOV K2 D101
M1
M2
M3



Basic 初始化程序如下：

```
1   GLOBAL SUB InitSub()
2       base(0,1)
3       atype = 1,1
4       units = 1,1
5       accel = 10000,10000
6       decel = 0,0
7       speed = 1000,1000
8       dpos = 0,0
9       modbus_reg(101)=0
10  END SUB
```

第六章 常见问题排查

控制器在使用过程中，因为接线不对、程序逻辑问题、指令使用错误等各种原因，会出现电机没有按照设定轨迹运行、控制器报错等各种问题。

如何来排查原因和解决问题，第一原则是关闭其他软件，使用 ZDevelop 软件排查，需要熟练使用以下功能：手动运动调试、断点调试、示波器抓取、寄存器查看、在线命令发送、程序打印信息、IO 口快速检测。

现象	排查方法
电机不动	手动运动调试
触摸屏寄存器数据值不对	寄存器查看
没有按照程序预期运行	断点调试+打印程序信息
输入输出口不起作用	IO 口快速检测
机台抖动大	示波器抓取

6.1 问题排查步骤

程序报错先排查程序问题：

当程序运动出错后，ZDevelop 软件会显示出错信息，如果出错信息没有看到，可以通过在线命令行输入?*task 再次查看出错信息，双击出错信息可以自动切换到程序出错位置，或在菜单栏“调试”-“故障诊断”窗口查看。错误信息一般有打印提示，例如下表所示。

问题	可能原因
2043:Unknown function is met	不认识的标识符，控制器不支持的功能
stop of error:2049: Line not ended.	①部分命令必须占用一整行 ②GSUB 调用不需要括号()
stop of error:2033: Unknown label is met.	①未定义的变量或数组 ②未定义的 SUB 过程 ③有定义数组，但是定义的语句没有执行到，可能是对应文件没有设置自动运行
2048:Function can only be used in expression	函数必须返回值，不用在一行的开头地方
2064:Param few	函数参数过少
2063:Param too many	函数参数过多
2072:Need = sign	未写“=”号
2060:Syntax format error	指令语法错误
error:1010	重复暂停
error:1011	没有运动，无法暂停

程序运行错误解决后，仍然不能正常运行，若电机不动，再检查以下设置。

1．驱动器原因：

驱动器的出厂设置一般没有反转 IO 电平，会导致驱动器限位报警，要根据驱动器手册设置限位电平反转。比如松下伺服要将 Pr4.01、Pr4.02 的参数分别设置为 010101h（65793）、020202h（131586）。其他品牌驱动器请根据相关驱动器手册操作。

对应参数	出厂设置值（10 进制）	位置控制/全闭环控制	
		信号名称	逻辑
Pr4.00	00323232h（3289650）	SI-MON5	常开（ON）
Pr4.01	00818181h（8487297）	POT	常闭（NC）
Pr4.02	00828282h（8553090）	NOT	常闭（NC）

信号名称	记号	设定值	
		常开（ON）	常闭（NC）
无效	-	00h	设定不可
正方向驱动禁止输入	POT	01h	81h
负方向驱动禁止输入	NOT	02h	82h

2．程序可能原因：

1）UNITS 设置过小，电机运动极为缓慢，肉眼无法分辨。

2）电机处于异常状态（限位、报警等），无法运动，打印 AXISSTATUS 的值判断。

3）电机接线错误，发出脉冲无法正确传递。

4）轴使能 OP 口未打开（伺服电机才需要打开）。

5）程序处理使得电机无法运动，下载空程序确认。

6）驱动器报警。

以下原因为总线轴才会出现：

7）总线扫描、开启失败，打印返回值确认。

8）未打开 WDOG 总使能、AXIS_ENABLE 轴使能指令。

9）驱动器状态设置错误，具体查看驱动器手册。

3．问题排查步骤：

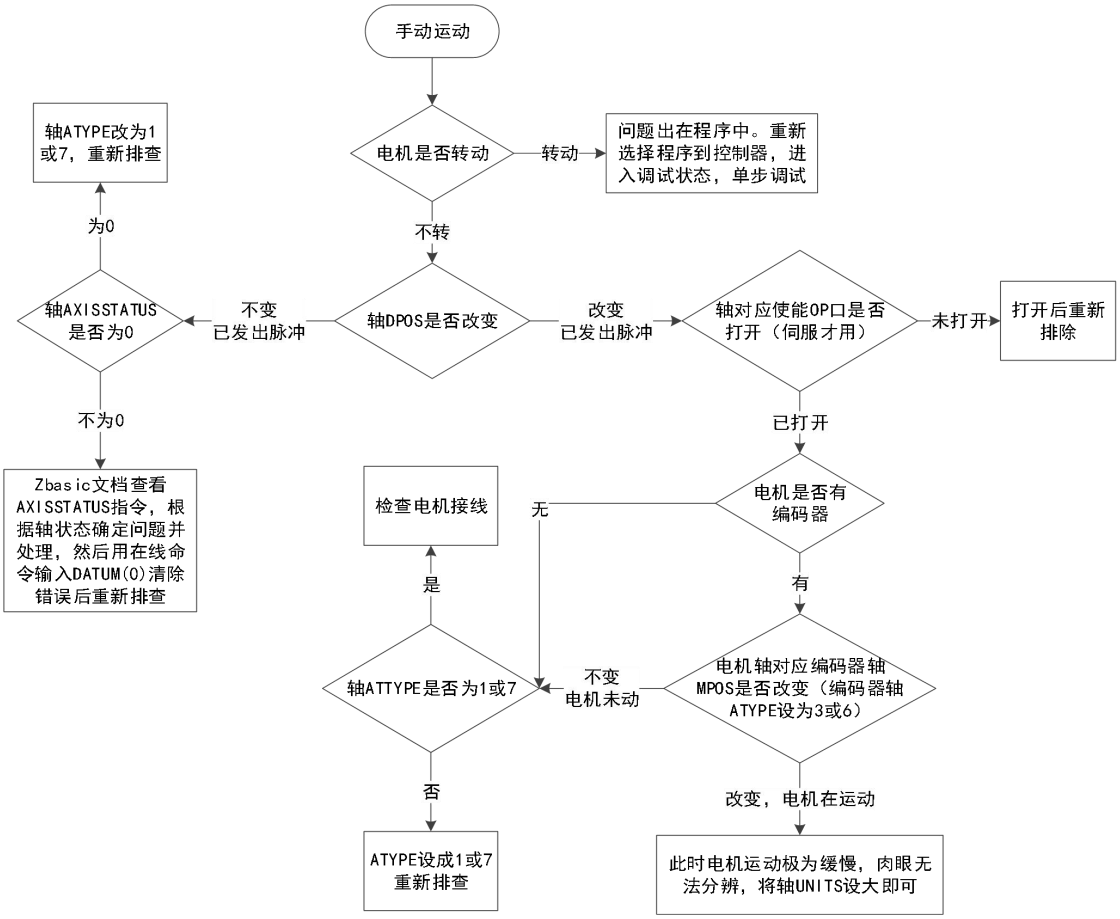
1）使用 ZDevelop 软件进行问题排查。

2）关闭除 ZDevelop 的其他连接控制器的软件、程序等，避免外部因素影响。

3）使用 ZDevelop 下载一个空程序到控制器，避免内部因素影响。

4）打开 ZDevelop 的“视图”-“手动运动”和“视图”-“轴参数”，便于操作和查看。

5）脉冲轴按照下方步骤进行排查。



若电机只能单向运动，可能原因：

1. 电机处于限位状态，查看 AXISSTATUS 确认。
2. 电机控制模式不对，INVERT_STEP 设置为相应模式（双脉冲或脉冲+方向）。
3. 电机接线问题，确认接线。

6.2 问题排查工具

6.2.1 手动运动调试

手动运动可以排查电机接线问题。

关闭所有除 ZDevelop 的软件，同时使用 ZDevelop 连接控制器，下载空程序，并在“视图”-“轴参数”中选择轴号，手动设置好轴类型 ATYPE、脉冲当量 UNITS、、加速度 ACCEL、减速度 DECEL、速度 SPEED，然后打开“视图”-“手动运动”，手动操作电机。（下图为 ZDevelop V3.00.01 版本）

操作方法：按住“左”/“右”不放，电机持续运动，松开停止。“指令位置”显示当前发出的脉冲 DPOS（单位为 UNITS）。填写“距离”参数，点击“运动”，勾选“绝对”时，电机运动到距离参数位置；不勾选“绝对”时，电机按距离参数继续运动。

“左”“右”操作时可能会出现的问题及解决办法：

1．电机不动，但 DPOS 改变。

此时控制器脉冲已发出，检查驱动器有无报警、电机接线。

也可能是 UNITS 设置过小，电机在转动，但是转动不明显。

2．电机只往一个方向转动。

查看电机控制方式，目前控制器脉冲轴只能用双脉冲和脉冲+方向两轴控制模式，不可使用正交脉冲控制。

3．只操作某一边时电机才转动。

检查电机接线。

电机当前控制模式与控制器当前控制模式不同，控制器默认为脉冲+方向控制，使用 INVERT_STEP 指令可以修改。

4．电机不动，DPOS 也不改变。

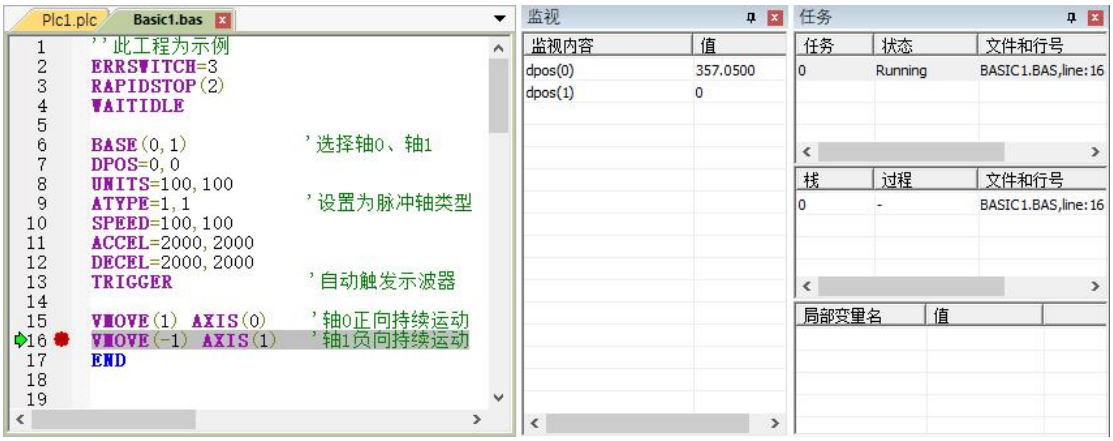
检查轴参数 AXISSTATUS 是否报警。

6.2.2 断点调试

断点调试可以查看程序运行的具体过程，主要用于判断程序逻辑错误。配合监视内容可以查看程序每执行一步对寄存器、变量、数组等的影响。调试程序和控制器程序必须一致。

断点快捷键 F9 添加，可以添加多个，调试完毕后将所有断点移除后再次下载程序到控制器运行。

对于使用 ZDevelop 软件开发的程序，连接好控制器，点击“调试”-“启动/停止调试”进入调试状态。详情参见“程序调试”。



此时可以查看个任务运行情况、监视内容、子函数调用过程、子函数局部变量值。

6.2.3 示波器抓取

示波器可以抓取各类数据，数据具体种类查看“视图”-“示波器”-“数据源”。

示波器抓取一般用来判断电机实际运行速度和实际位置。

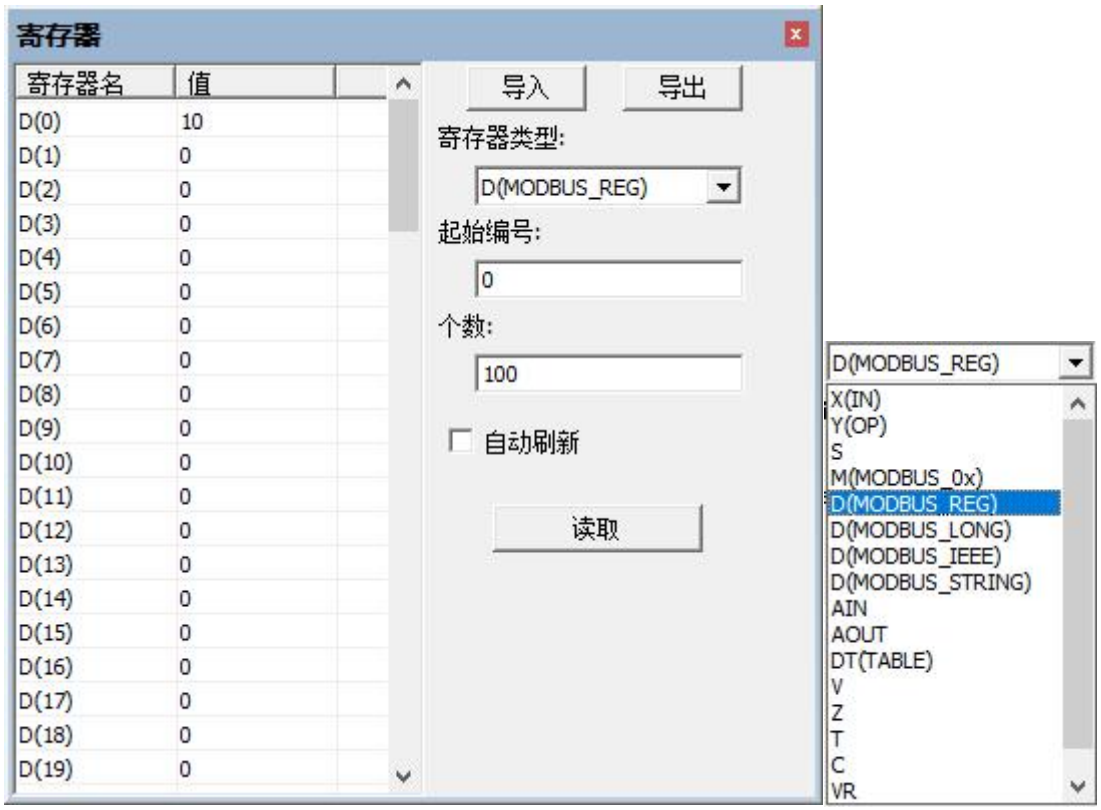
XY 模式可以查看二维的路径轨迹，需要将前两个通道的“数据源”设置为 DPOS 或 MPOS。具体使用方法查看“示波器的使用”章节。

机台实际运行抖动大时，可以用示波器抓取编码器 MSPEED，查看波形是否光滑；如果光滑说明脉冲发送平稳，此时查看速度曲线加减速过程是否过陡，匀速时速度值是否过大。

6.2.4 寄存器查看

使用 Zdevelop 软件可以方便的查看寄存器的数据，点击“视图”-“寄存器”查看。寄存器种类有 IN、OP、MODBUS_4X、MODBUS_0X、TABLE、VR、AIN、AOUT。

必须支持 PLC 功能的控制器才能使用。



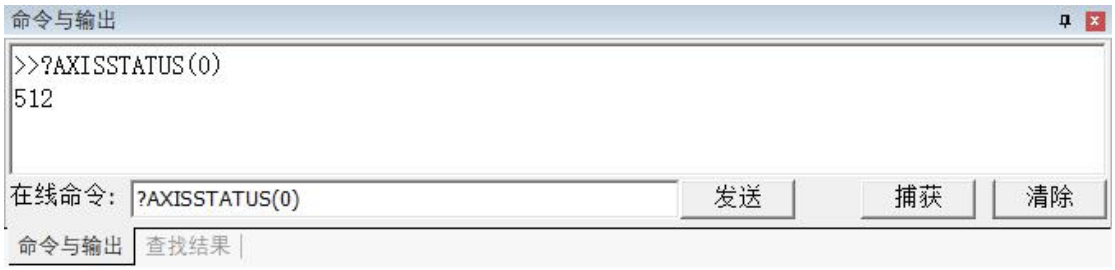
比如出现触摸屏某个寄存器参数没有按照预期变化时，此时可直接查看触摸屏对应的寄存器类型及编号，确认控制器中此寄存器数值有无变化。

如果变化，说明控制器与触摸屏通讯出现问题，检查接线和通讯参数设置；如果无变化，说明程序执行有问题，断点调试检查程序执行是否正确。

6.2.5 在线发送命令

在线命令可以实时执行发送的指令，一般用于直接判断控制器指令功能是否正常。

比如程序已经执行，但实际电机没有按照程序预设功能运行，无法确定是否是程序中其他任务或流程影响，还是控制器功能错误。此时就可以下载空程序到控制器，然后直接在线命令发送功能指令，观察现象即可判断问题原因。



6.2.6 打印程序信息

在各段程序间打印信息，可以方便的查看程序段是否执行，执行了几次，执行时对应的参数是什么。

打印指令为 PRINT，也可以使用简写“?”（注意是英文符号）。

```
dim ccv(20),aa,bb
ccv="ad"
aa=100
bb=300

base(0,1,2)
atype=1,1,1
units=200,200,200
speed=100,100,100
accel=1000,1000,1000

stoptask 1
runtask 1,aaa
```

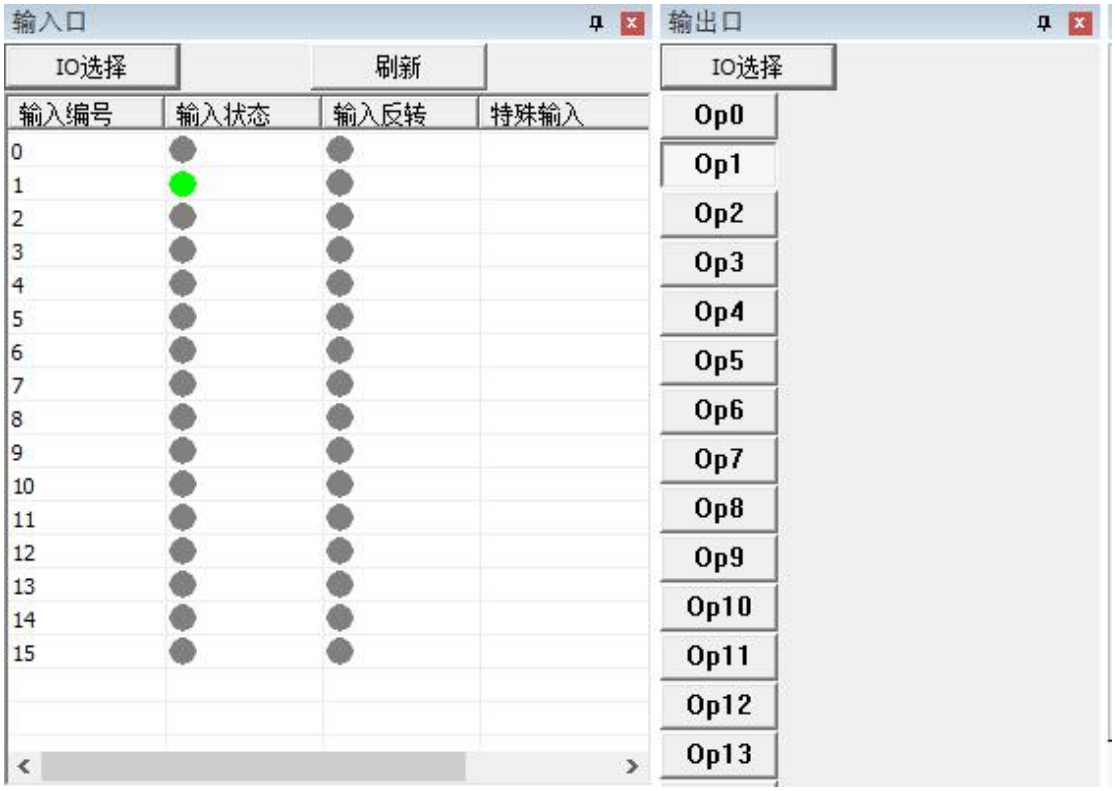
```
while 1
    ?bb
    move(bb)
wend
end


sub aaa(num)
  while 1
    ?aa
    move(aa)
  wend
end sub
```

6.2.7 IO 口快速检测

ZDevelop 连接控制器，点击“视图”-“输入口”、“输出口”，同时，将输出口和输入口一对一连接起来（将输入口和输出口的 EGND 接在一起），然后操作“输出口”，可以查看到“输入口”中对应状态变化。

某些控制器 IO 口需要另接 24V 电源给 IO 单独口供电。

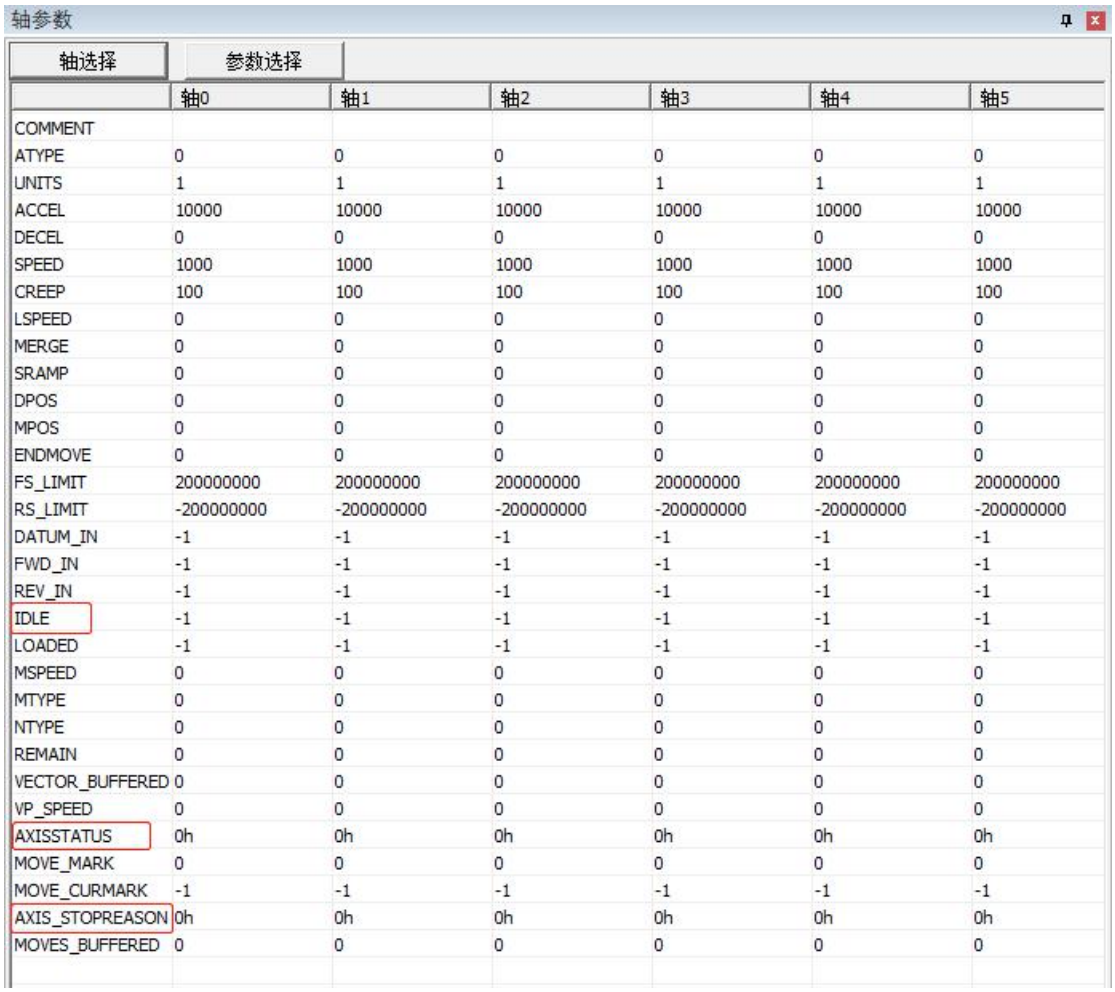
如果检测扩展板 IO，先确认 CANL 和 CANH 接线时是否接入 120 欧电阻；然后确认拨码开关设置是否正确（不能覆盖控制器原本 IO 编号）；再确认主电源与 IO 电源是否接好；最后按照上面所述进行检测。



6.2.8 轴参数状态判断

轴的基本参数在轴参数窗口里随时查看，例如 ATYPE、UNITS、SPEED 等重要参数的设置情况。在此窗口可直接修改轴参数，修改完成立即生效，只读的参数不支持修改。

轴运行状态判断主要依据 IDLE、AXISSTATUS 和 AXIS_STOPREASON 这三个指令，在轴参数窗口对这三个指令参数实时监控。



	轴0	轴1	轴2	轴3	轴4	轴5
COMMENT						
ATYPE	0	0	0	0	0	0
UNITS	1	1	1	1	1	1
ACCEL	10000	10000	10000	10000	10000	10000
DECEL	0	0	0	0	0	0
SPEED	1000	1000	1000	1000	1000	1000
CREEP	100	100	100	100	100	100
LSPEED	0	0	0	0	0	0
MERGE	0	0	0	0	0	0
SRAMP	0	0	0	0	0	0
DPOS	0	0	0	0	0	0
MPOS	0	0	0	0	0	0
ENDMOVE	0	0	0	0	0	0
FS_LIMIT	200000000	200000000	200000000	200000000	200000000	200000000
RS_LIMIT	-200000000	-200000000	-200000000	-200000000	-200000000	-200000000
DATUM_IN	-1	-1	-1	-1	-1	-1
FWD_IN	-1	-1	-1	-1	-1	-1
REV_IN	-1	-1	-1	-1	-1	-1
IDLE	-1	-1	-1	-1	-1	-1
LOADED	-1	-1	-1	-1	-1	-1
MSPEED	0	0	0	0	0	0
MTYPE	0	0	0	0	0	0
NTYPE	0	0	0	0	0	0
REMAIN	0	0	0	0	0	0
VECTOR_BUFFERED	0	0	0	0	0	0
VP_SPEED	0	0	0	0	0	0
AXISSTATUS	0h	0h	0h	0h	0h	0h
MOVE_MARK	0	0	0	0	0	0
MOVE_CURMARK	-1	-1	-1	-1	-1	-1
AXIS_STOPREASON	0h	0h	0h	0h	0h	0h
MOVES_BUFFERED	0	0	0	0	0	0

1、IDLE 指令用于判断加在轴上的运动指令是否完成，运动中返回 0，运动结束返回-1，程序中一般使用 WAIT IDLE(轴号)语句判断轴状态。

2、AXISSTATUS 查看轴的各种状态。按十进制显示数值，按二进制对应位判断状态，可同时发生多个错误。

轴参数窗口显示的是八进制，使用 PRINT 指令打印的值为十进制。

位	说明	打印值	
1	随动误差超限告警	2	2h
2	与远程轴通讯出错	4	4h
3	远程驱动器报错	8	8h
4	正向硬限位	16	10h
5	反向硬限位	32	20h

6	找原点中	64	40h
7	HOLD 速度保持信号输入	128	80h
8	随动误差超限出错	256	100h
9	超过正向软限位	512	200h
10	超过负向软限位	1024	400h
11	CANCEL 执行中	2048	800h
12	脉冲频率超过 MAX_SPEED 限制，需要修改降速或修改 MAX_SPEED	4096	1000h
14	机械手指令坐标错误	16384	4000h
18	电源异常	262144	40000h
21	运动中触发特殊运动指令失败	2097152	200000h
22	告警信号输入	4194304	400000h
23	轴进入了暂停状态	8388608	800000h

3、AXIS_STOPREASON 轴历史停止原因锁存，写 0 清除，按位锁存，锁存的是 AXISSTATUS 的信息。