



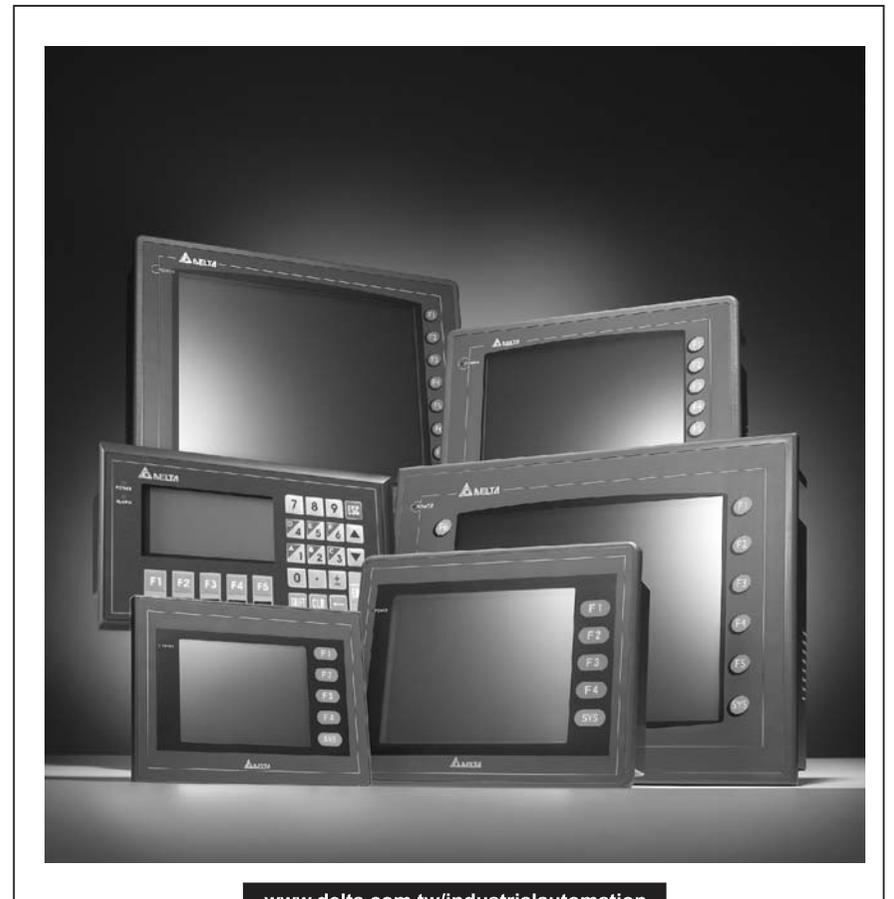
中达电通股份有限公司
上海市浦东新区民夏路238号, 201209
全国客服热线 ☎ 021-58639595
公司网址: www.delta-cimic.com

北京: 010-82253225
哈尔滨: 0451-53660643
南昌: 0791-6255010
武汉: 027-85448265
长春: 0431-88925060
杭州: 0571-88820610
南京: 025-83346585
西安: 029-88360640
长沙: 0731-2941118
合肥: 0551-2816777
上海: 021-63012827
厦门: 0592-5313601
成都: 028-84342072
济南: 0531-86907277
沈阳: 024-23341159
郑州: 0371-63842448
广州: 020-38792175

DELTA
DOP
系列 HMI-WPLSoft 操作说明



DOP 系列 HMI-WPLSoft 操作说明



www.delta.com.tw/industrialautomation

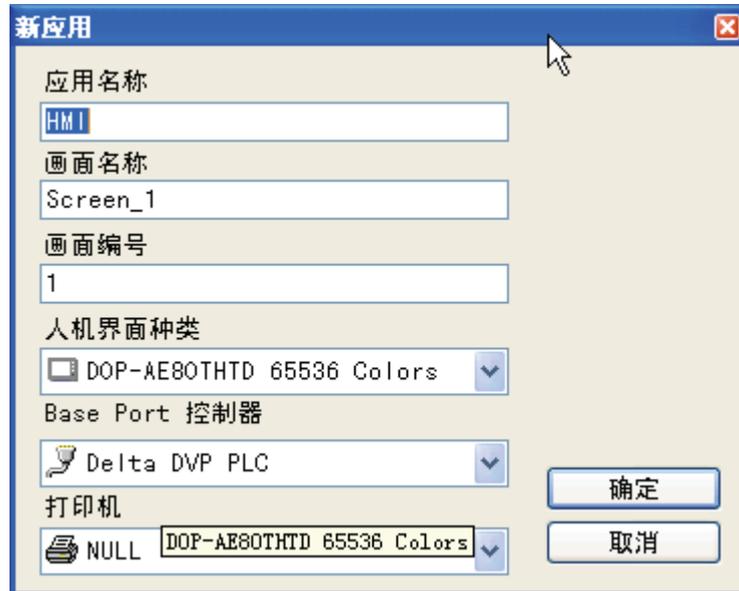
目录

第一章	启动 EXIO 功能.....	1-1
第二章	HMI-WPLSoft 编辑环境说明.....	2-1
第三章	梯形图编辑模式环境.....	3-1
第四章	IO 指示灯.....	4-1
第五章	内部存储器地址.....	5-1
附录 A	EXIO 装置一览表.....	A-1
附录 B	EXIO 指令一览表.....	B-1
附录 C	EXIO 基本指令说明.....	C-1
附录 D	EXIO 应用指令说明.....	D-1

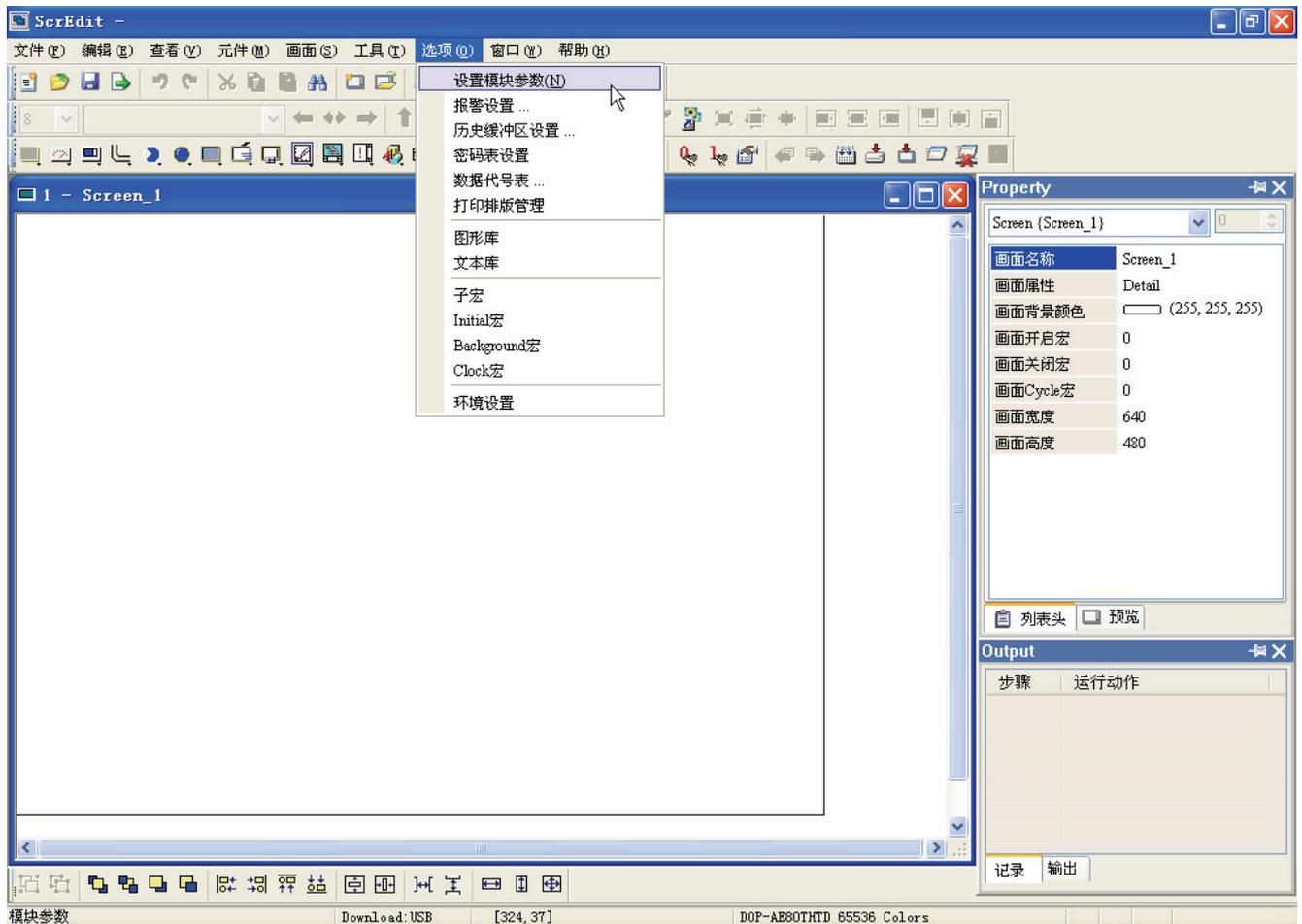
(此页有意留为空白)

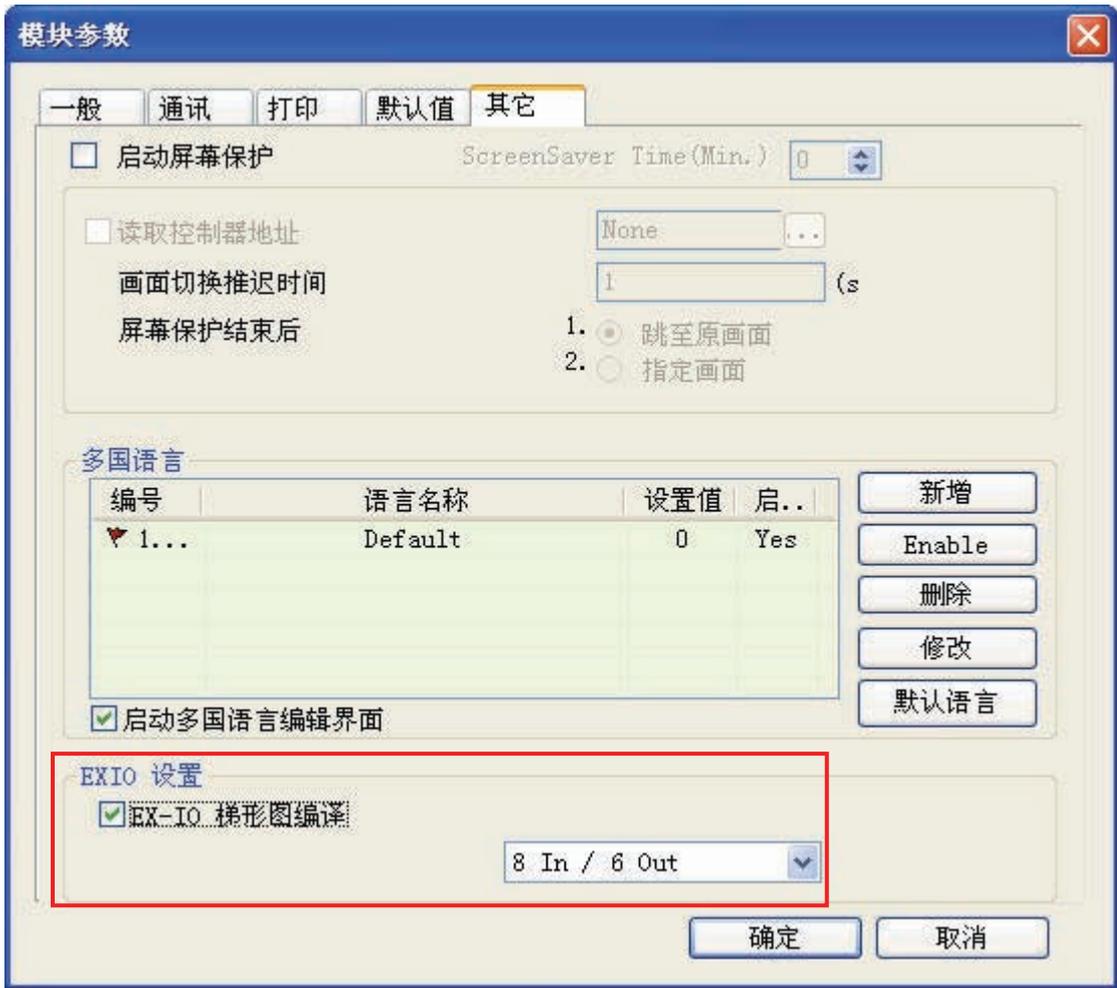
第一章 启动 EXIO 功能

目前全系列台达人机只有 AE 系列人机可以支持扩展模块，因此要使用 EXIO 模块的时候要先选择 AE 系列的机种后才能使用 EXIO 模块的功能。



选择人机机型后，选择『选项』→『设定模块参数』。在『设定模块参数』中的『其他』页面中，选择启动『启用 EX-IO（梯形图编译）』，并可选择 EXIO 模块的输出输入点数。





当启用 EX-IO（梯形图编译）后工具栏就会多一个梯形图编辑的图示，直接点击图案便可打开梯形图编辑。也可以直接在工具选单中打开梯形图编辑。



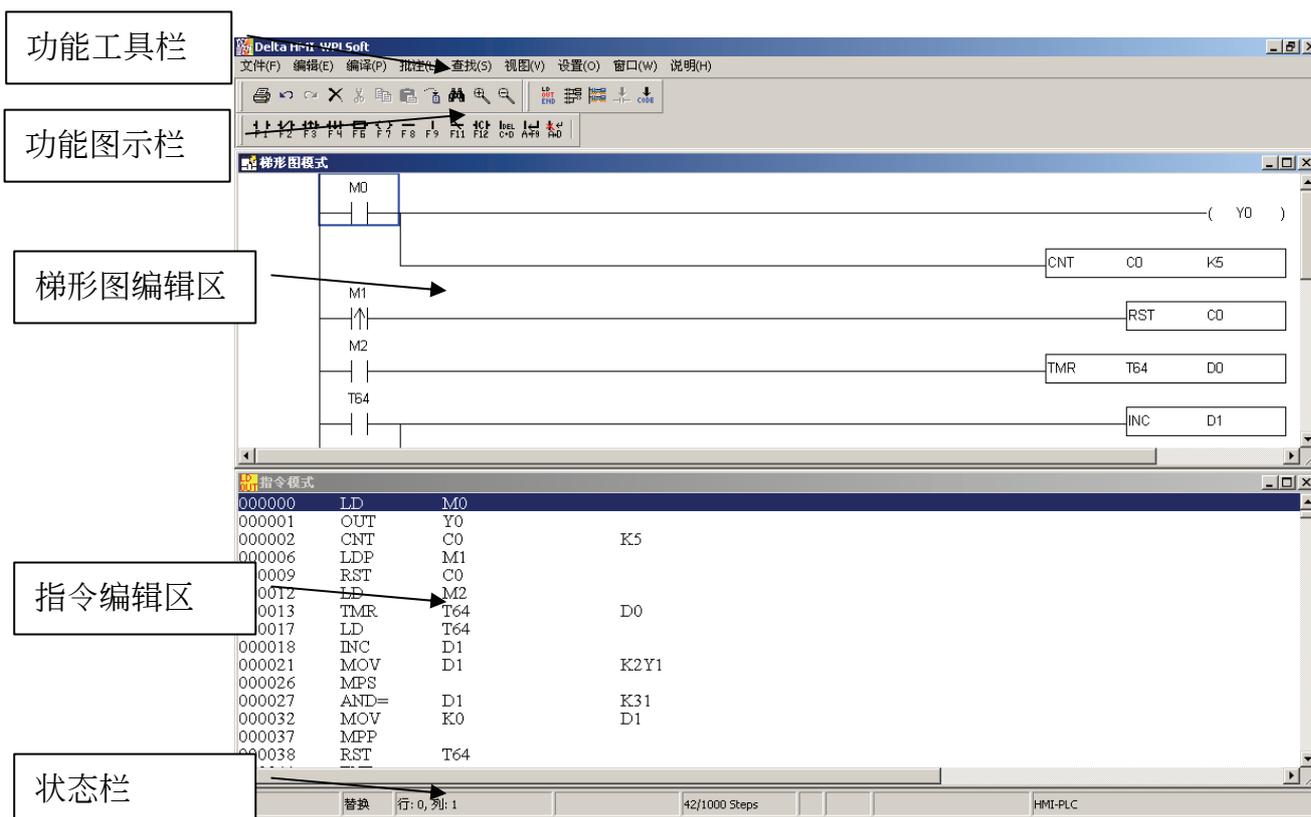
未启用 EXIO（梯形图编译）前工具栏的图案

启用 EXIO（梯形图编译）后工具栏的图案

	编译	Ctrl+F7
	下载画面数据与配方	Ctrl+F8
	下载画面数据	Ctrl+F9
	在线模拟	Ctrl+F4
	离线模拟	Ctrl+F5
配方		
32 bits 配方		
下载配方		
取得目前韧体序号		
	梯形图编辑	

第二章 HMI-WPLSoft 编辑环境说明

点选  图示即可打开梯形图编辑窗口，此时 Screen Editor 窗口会自动最小化，需要注意的是在编辑梯形图时是无法同时编辑人机画面的；必须等到梯形图编辑窗口关闭后，才会跳回 Screen Editor 编辑窗口。梯形图编辑窗口并无打开跟保存文件的选项，当此梯形图编辑窗口关闭时，就会自动保存文件。



功能工具栏: 在 HMI-WPLSoft 编辑软件的主功能工具栏中共有十种功能选项:『文件(F)』、『编辑(E)』、『编译(P)』、『批注(L)』、『查找(S)』、『视图(V)』、『设置(O)』、『窗口(W)』和『说明(H)』。

功能图示栏: 提供使用者可直观地由图示 (Icons) 利用鼠标直接点选所需功能的命令按钮栏，此栏主要有四种：

1. 一般工具栏:



2. 快速工具栏:



3. 梯形图工具栏：（于梯形图模式下显示）



编辑工作区：设计编辑程序的区域；可依使用者习惯选择指令编辑、梯形图编辑。

状态栏：可显示的信息种类包括替换/插入模式、编辑框所在位置等信息。



2.1 文件（File）选项

『文件（F）』功能的下拉菜单如下图，提供以下功能选项：



■ 打印 ⇒ 打印目前的文件。（会依目前工作窗口模式打印数据）

- ◆ 方法一：『文件（F）』功能表中『打印(P)』命令。
- ◆ 方法二：鼠标点选图示工具栏上的 .
- ◆ 方法三：利用快捷键，键盘输入复合键 [Ctrl] + [P]。

当程序设计完成时，为方便数据整理和查看，可在 HMI-WPLSoft 使用  或『文件（F）』功能内的『打印(P)』命令将相关程序或数据内容打印出来；依据编辑器的工作窗口不同，HMI-WPLSoft 提供梯形图打印、指令打印工作模式，以下将介绍各种打印的模式：

■ 梯形图打印

当 HMI-WPLSoft 编辑器的工作窗口为梯形图模式，执行『打印(P)』命令后会出现打印设置对话框窗口，打印选项（全部、部分）和部分打印范围（起始页、结束页），可选择是否打印标题、页码和封面，另可预览打印结果，打印设置。

在梯形图编辑区显示结果与打印出来的数据相同，即梯形图若有批注显示也会印出批注。

■ 指令打印

当 HMI-WPLSoft 编辑器的工作窗口为指令模式，执行『打印(P)』命令后会出现打印设置对话框窗口，打印选项（全部、部分）和部分打印范围（起始页、结束页），可选择是否打印标题、页码和封面，另可预览打印结果，打印设置。



■ 设置印表机 ⇨ 选择印表机和其相关属性设置。

- ◆ 方法一：『文件 (F)』功能表中『设置印表机(Q)』命令。
- ◆ 方法二：利用快捷键，键盘输入复合键 [Ctrl] + [Q]。

■ 退出 ⇨ 结束 HMI-WPLSoft 编辑软件。

- ◆ 方法一：『文件 (F)』功能表中『退出(X)』命令。
- ◆ 方法二：鼠标点选窗口右上角的  图示。
- ◆ 方法三：利用快捷键，键盘输入复合键 [Alt] + [X]。

文件说明：

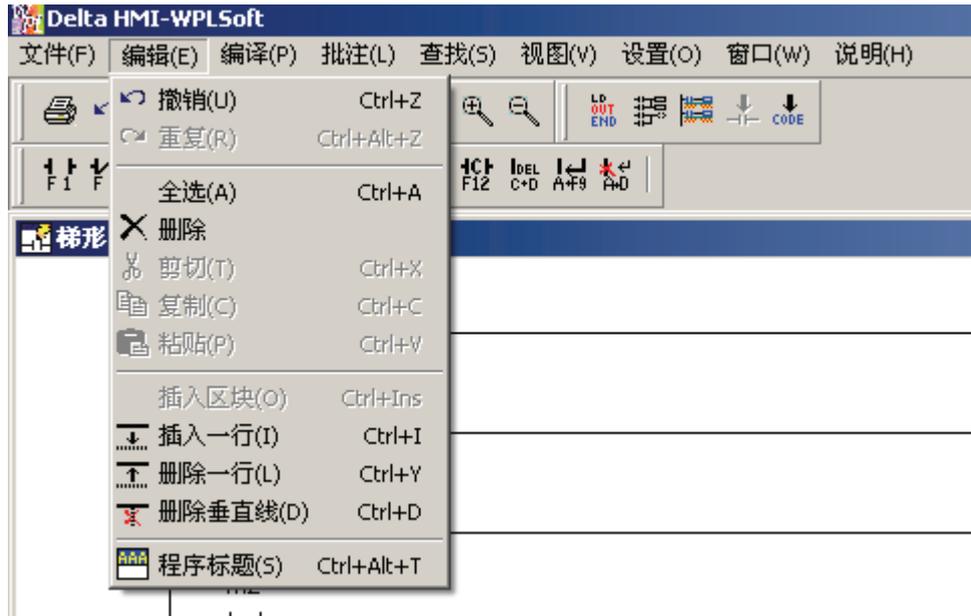
EXIO 程序完成编辑、编译和相关设置编辑后保存，Screen Editor 编辑软件会产生 6 种扩展名的文件形式。若欲将完整（含所有程序批注和设置）程序拷贝至其他磁碟或目录，建议将所有相同主档名的文件全部复制。一般程序要完整地作备份，须把下述的文件都复制才算完整。

扩展名	说明
1 * .DLP	⇨ EXIO 指令档。
2 * .LAD	⇨ 梯形图编辑档。
3 * .LMT	⇨ 梯形图区段批注记录档。
4 * .LAB	⇨ 标签 (Label) 记录档。

扩展名	说明
5 * .RCM	⇒ 特 D/特 M 批注预设档。
6 * .DOP*	⇒ 人机画面档。

2.2 编辑 (Edit) 选项

『编辑』功能表如下图，提供以下命令：



■ 撤销 ⇒ 还原上一动作。(最多 10 次)

- ◆ 方法一：『编辑 (E)』功能表中『撤销(U)』命令。
- ◆ 方法二：鼠标点选图示工具栏上的 。
- ◆ 方法三：利用快捷键，键盘输入复合键 [Ctrl] + [Z]。
- ◆ 方法四：鼠标右键功能表中『撤销』命令。

■ 重复 ⇒ 重做还原前的动作。

- ◆ 方法一：『编辑 (E)』功能表中『重复(R)』命令。
- ◆ 方法二：鼠标点选图示工具栏上的 。
- ◆ 方法三：利用快捷键，键盘输入复合键 [Ctrl] + [Alt] + [Z]。
- ◆ 方法四：鼠标右键功能表中『重复』命令。

■ 全选 ⇒ 选取程序文件所有内容并标示区块。

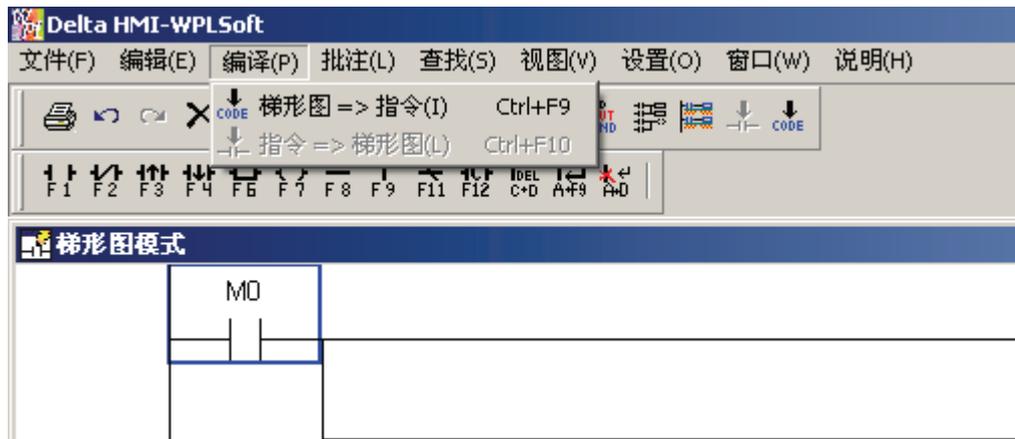
- ◆ 方法一：『编辑 (E)』功能表中『全选(A)』命令。
- ◆ 方法二：利用快捷键，键盘输入复合键 [Ctrl] + [A]。

- 删除 ⇨ 将标示区块或编辑方块所在位置的数据清除。
 - ◆ 方法一：『编辑 (E)』功能表中『删除』命令。
 - ◆ 方法二：鼠标点选图示工具栏上的 。
 - ◆ 方法三：利用快捷键，键盘按下 [Delete] 键。
 - ◆ 方法四：鼠标右键功能表中『删除』命令。
- 剪切 ⇨ 剪切文件中的区块数据。
 - ◆ 方法一：『编辑 (E)』功能表中『剪切(T)』命令。
 - ◆ 方法二：鼠标点选图示工具栏上的 。
 - ◆ 方法三：利用快捷键，键盘输入复合键 [Ctrl] + [X]。
 - ◆ 方法四：鼠标右键功能表中『剪切』命令。
- 复制 ⇨ 复制文件中的区块数据。
 - ◆ 方法一：『编辑 (E)』功能表中『复制(C)』命令。
 - ◆ 方法二：鼠标点选图示工具栏上的 。
 - ◆ 方法三：利用快捷键，键盘输入复合键 [Ctrl] + [C]。
 - ◆ 方法四：鼠标右键功能表中『复制』命令。
- 粘贴 ⇨ 粘贴区块数据到文件上。
 - ◆ 方法一：『编辑 (E)』功能表中『粘贴(P)』命令。
 - ◆ 方法二：鼠标点选图示工具栏上的 。
 - ◆ 方法三：利用快捷键，键盘输入复合键 [Ctrl] + [V]。
 - ◆ 方法四：鼠标右键功能表中『粘贴』命令。
- 插入区块 ⇨ 插入一区块数据至文件中。(只适用于梯形图编辑模式)
 - ◆ 方法一：『编辑 (E)』功能表中『插入区块(O)』命令。
 - ◆ 方法二：利用快捷键，键盘输入复合键 [Ctrl] + [Ins]。
 - ◆ 方法三：鼠标右键功能表中『插入区块』命令。
- 插入一行 ⇨ 插入一行空白行文件中。
 - ◆ 方法一：『编辑 (E)』功能表中『插入一行(I)』命令。
 - ◆ 方法二：利用快捷键，键盘输入复合键 [Ctrl] + [I]。
 - ◆ 方法三：鼠标右键功能表中『插入一行』命令。

- 删除一行 ⇨ 删除文件上的一行数据。
 - ◇ 方法一：『编辑 (E)』功能表中『删除一行(L)』命令。
 - ◇ 方法二：利用快捷键，键盘输入复合键 [Ctrl] + [Y]。
 - ◇ 方法三：鼠标右键功能表中『删除一行』命令。
- 删除垂直线 ⇨ 删除文件上的垂直线。（只适用于梯形图编辑模式）
 - ◇ 方法一：『编辑 (E)』功能表中『删除垂直线(D)』命令。
 - ◇ 方法二：鼠标点选图示工具栏上的 。
 - ◇ 方法三：利用快捷键，键盘输入复合键 [Ctrl] + [D]。
 - ◇ 方法四：鼠标右键功能表中『删除垂直线』命令。
- 程序标题 ⇨ 编辑建立程序标题、文件名称、公司名称和设计人，于打印时可以此标题页内容作为简易封面。
 - ◇ 方法一：『编辑 (E)』功能表中『程序标题(S)』命令。
 - ◇ 方法二：利用快捷键，键盘输入复合键 [Ctrl] + [Alt] + [T]。

2.3 编译 (Compile) 选项

『编译』功能表如下图，提供以下命令：

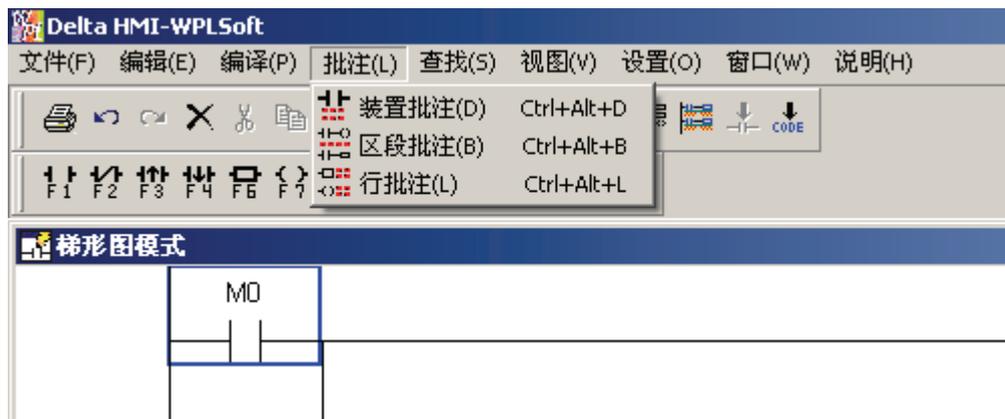


- 梯形图=>指令 ⇨ 梯形图程序转换为指令码。
 - ◇ 方法一：『编译 (P)』功能表中『梯形图=>指令(I)』命令。
 - ◇ 方法二：鼠标点选图示工具栏上的 。
 - ◇ 方法三：利用快捷键，键盘输入复合键 [Ctrl] + [F9]。

- 指令=>梯形图 ⇨ 将指令码转换成梯形图程序。
 - ◇ 方法一：『编译 (P)』功能表中『指令=>梯形图(L)』命令。
 - ◇ 方法二：鼠标点选图示工具栏上的 。
 - ◇ 方法三：利用快捷键，键盘输入复合键 [Ctrl] + [F10]。

2.4 批注 (Label) 选项

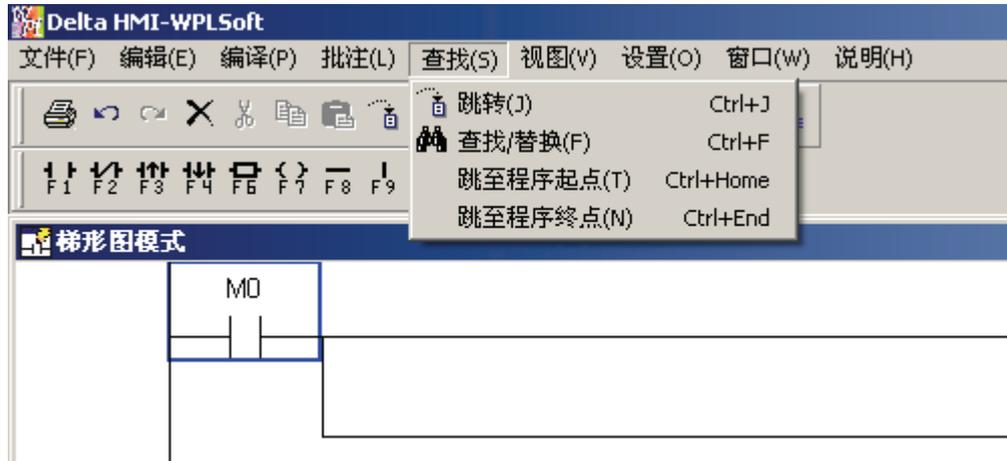
『批注』功能表如下图，提供以下命令：



- 装置批注 ⇨ 当编辑方块停于含有操作数的命令上时，可替该指令的每个操作数装置组件加上批注。
 - ◇ 方法一：『批注 (L)』功能表中『装置批注(D)』命令。
 - ◇ 方法二：利用快捷键，键盘输入复合键 [Ctrl] + [Alt] + [D]。
 - ◇ 方法三：鼠标右键功能表中『装置批注输入』命令。
- 区段批注 ⇨ 可在程序中空白行编辑区段批注，必须在一空白行才可以编辑。(只适用于梯形图编辑模式。)
 - ◇ 方法一：『批注 (L)』功能表中『区段批注(B)』命令。
 - ◇ 方法二：利用快捷键，键盘输入复合键 [Ctrl] + [Alt] + [B]
 - ◇ 方法三：鼠标右键功能表中『区段批注输入』命令。
- 行批注 ⇨ 可于每行的输出线圈或命命后加上行批注。(只适用于梯形图编辑模式。)
 - ◇ 方法一：『批注 (L)』功能表中『行批注(L)』命令。
 - ◇ 方法二：利用快捷键，键盘输入复合键 [Ctrl] + [Alt] + [L]
 - ◇ 方法三：鼠标右键功能表中『行批注输入』命令。

2.5 查找 (Search) 选项

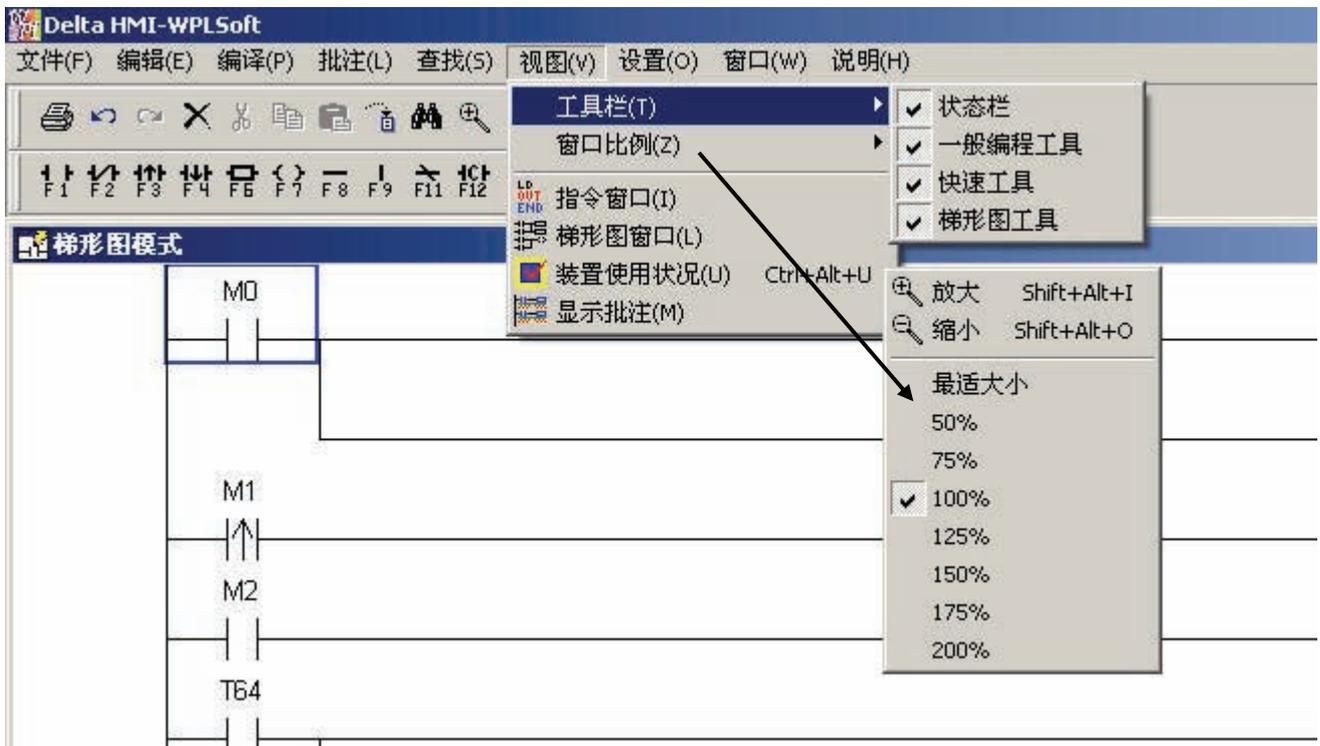
『查找』功能表如下图，提供以下命令：



- 跳转 ⇨ 可以跳转到指定的位置(以 Step 为单位)。
 - ◇ 方法一：『查找 (S)』功能表中『跳转(J)』命令。
 - ◇ 方法二：鼠标点选图示工具栏上的 。
 - ◇ 方法三：利用快捷键，键盘输入复合键 [Ctrl] + [J]。
- 查找/替换 ⇨ 可以查找或替换所指定的装置(组件)名称和命令。
 - ◇ 方法一：『查找 (S)』功能表中『查找/替换(F)』命令。
 - ◇ 方法二：鼠标点选图示工具栏上的 。
 - ◇ 方法三：利用快捷键，键盘输入复合键 [Ctrl] + [F]。
- 跳至程序起点 ⇨ 直接跳到程序的起点。
 - ◇ 方法一：『查找 (S)』功能表中『跳至程序起点(T)』命令。
 - ◇ 方法二：利用快捷键，键盘输入复合键 [Ctrl] + [Home]。
- 跳至程序终点 ⇨ 直接跳到程序最后一行的位置。
 - ◇ 方法一：『查找 (S)』功能表中『跳至程序终点(N)』命令。
 - ◇ 方法二：利用快捷键，键盘输入复合键 [Ctrl] + [End]。

2.6 视图 (View) 选项

『视图』功能表如下图，提供以下命令：



■ 工具栏 ⇨ 包含状态栏、一般编程工具、快速工具与梯形图工具：

○ 状态栏：显示或隐藏状态栏。



◇ 方法：『视图 (V)』功能表『工具栏(T)』命令的『状态栏』。

○ 一般编程工具：显示或隐藏一般编程工具栏。



◇ 方法：『视图 (V)』功能表『工具栏(T)』命令的『一般编程工具』。

○ 快速工具：显示或隐藏快速工具栏。



◇ 方法：『视图 (V)』功能表『工具栏(T)』命令的『快速工具』。

○ 梯形图工具：显示或隐藏梯形图编辑工具栏。(只适用于梯形图编辑模式。)

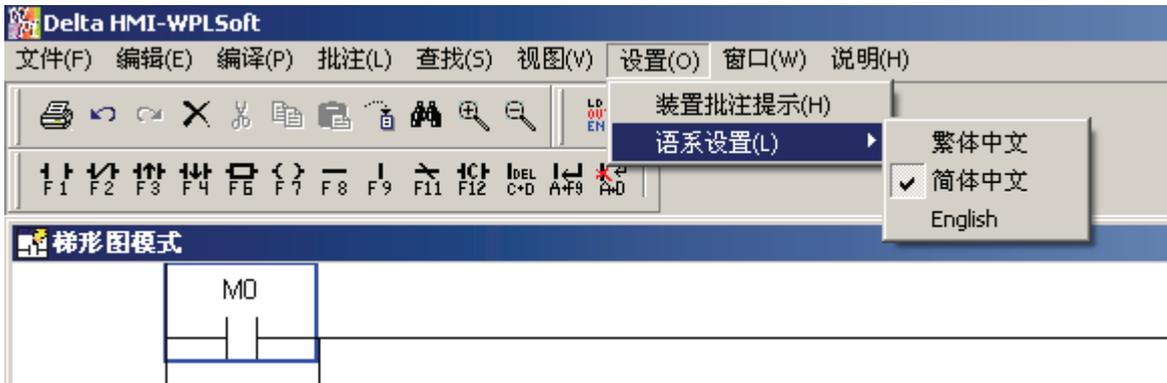


◇ 方法：『视图 (V)』功能表『工具栏(T)』命令的『梯形图工具』。

- 窗口比例 ⇨ 调整窗口的程序文字/图形内容显示比例，有 50%、75%、100%、125%、150%、175%、200% 和放大 、缩小 、最适大小的窗口显示比例供选择。
 - ◇ 方法一：『视图 (V)』功能表中『窗口比例(Z)』命令。
 - ◇ 方法二：窗口放大，利用快捷键，键盘输入复合键 [Shift] + [Alt] + [I]。鼠标点选图示工具栏上的 。
 - ◇ 方法三：窗口缩小，利用快捷键，键盘输入复合键 [Shift] + [Alt] + [O]。鼠标点选图示工具栏上的 。
- 指令窗口 ⇨ 切换工作窗口为指令模式。
 - ◇ 方法一：『视图 (V)』功能表中『指令窗口(I)』命令。
 - ◇ 方法二：鼠标点选图示工具栏上的 。
 - ◇ 方法三：由项目模式工作窗口区点选程序部分『指令表』。
- 梯形图窗口 ⇨ 切换工作窗口为梯形图模式。
 - ◇ 方法一：『视图 (V)』功能表中『梯形图窗口(L)』命令。
 - ◇ 方法二：鼠标点选图示工具栏上的 。
 - ◇ 方法三：由项目模式工作窗口区点选程序部分『梯形图』。
- 装置使用状况 ⇨ 显示所有装置组件的使用状况。
 - ◇ 方法一：『视图 (V)』功能表中『装置使用状况(U)』命令。
 - ◇ 方法二：利用快捷键，键盘输入复合键 [Ctrl] + [Alt] + [U]。
 - ◇ 方法三：由项目模式工作窗口区点选『装置使用状况』。
- 显示批注 ⇨ 切换装置和行批注显示或隐藏。
 - ◇ 方法一：『视图 (V)』功能表中『显示批注(M)』命令。
 - ◇ 方法二：鼠标点选图示工具栏上的 。

2.7 设置 (Option) 选项

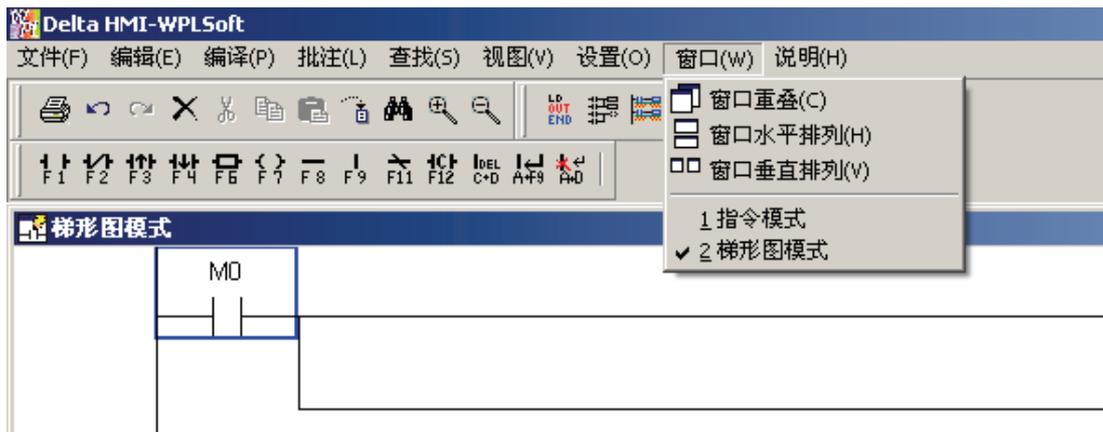
『设置』功能表如下图，提供以下命令：



- 装置批注提示 ⇒ 选取此功能后，在指令模式或梯形图模式下，使用指令码方式编辑 PLC 程序时亦会同时要求输入对应的装置批注。
 - ◇ 方法：『设置 (O)』功能表中『装置批注提示(H)』命令。
- 语系设置 ⇒ 使用者可依需求设置 HMI-WPLSoft 的操作界面语言，提供繁体中文，简体中文和英文三种语言。
 - ◇ 方法：『设置 (O)』功能表中『语系设置 (L)』命令。

2.8 窗口 (Window) 选项

『窗口』功能表如下图，提供以下命令：

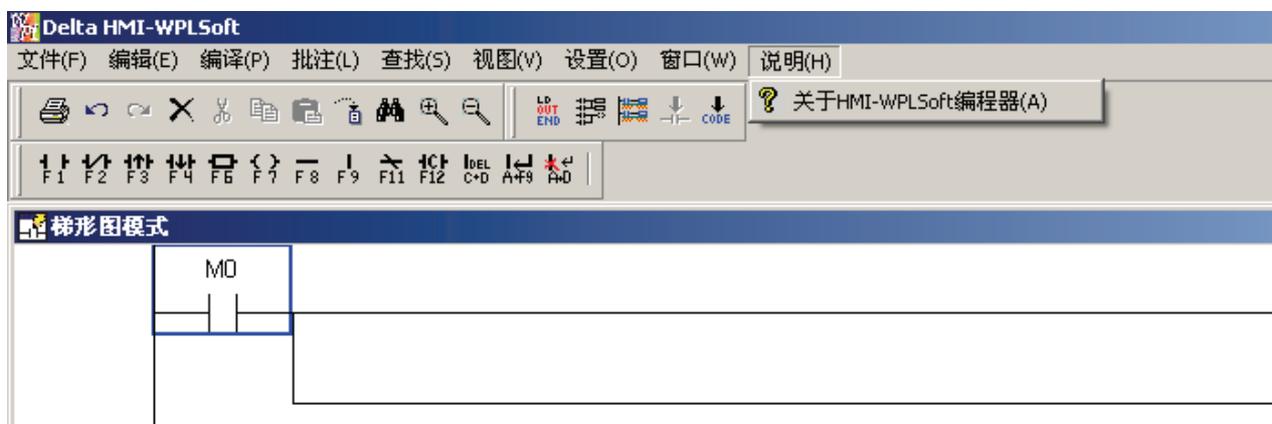


- 窗口重叠 ⇒ 以重叠方式安排窗口。
 - ◇ 方法：『窗口 (W)』功能表中『重叠窗口(C)』命令。
- 窗口水平并排 ⇒ 以水平方式并排窗口。
 - ◇ 方法：『窗口 (W)』功能表中『窗口水平并排(H)』命令。

- 窗口垂直并排⇒ 以垂直方式并排窗口。
 - ◇ 方法：『窗口 (W)』功能表中『窗口垂直并排(V)』命令。
- 目前编辑器打开的窗口 ⇒ 如指令模式、梯形图模式。
 - ◇ 方法：在 HMI-WPLSoft 编辑器打开指令模式、梯形图模式即可在目前编辑器打开的窗口栏显示。

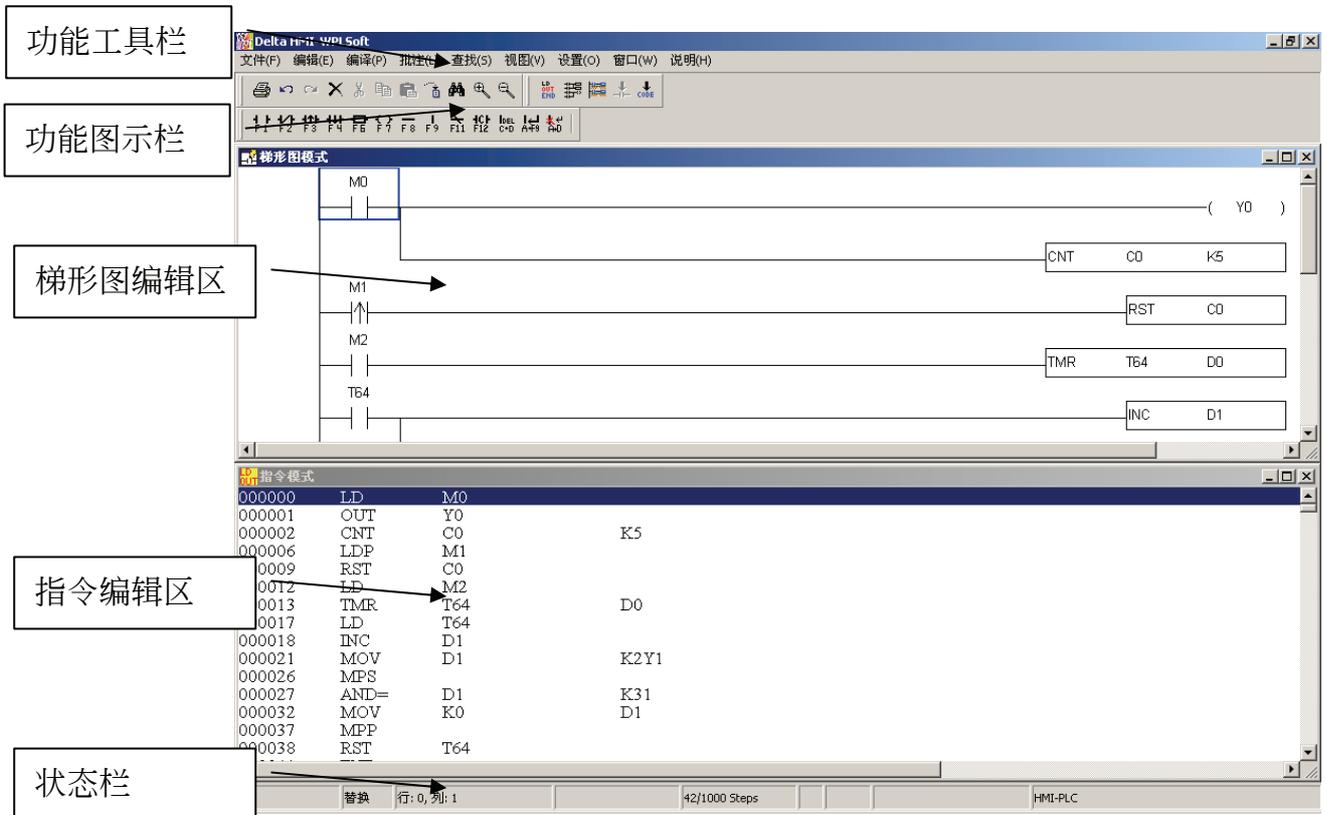
2.9 说明 (Help) 选项

『说明』功能表如下图，提供以下命令：



第三章 梯形图编辑模式环境

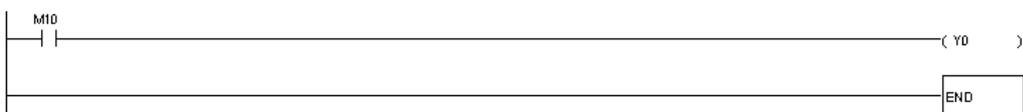
执行 HMI-WPLSoft 编辑器后就会进入梯形图模式的编辑环境，如下图所示。



在梯形图模式窗口上侧会显示出梯形图工具栏图示，使用者于编辑梯形图时，可以直接以鼠标移动到梯形图工具栏的组件图示上点选，或是将编辑方块移动到梯形图工作窗口的适当位置直接以指令输入编辑，另外也可利用键盘功能键（F1 ~ F12）作为输入方式。以下我们将说明各种操作方式步骤。

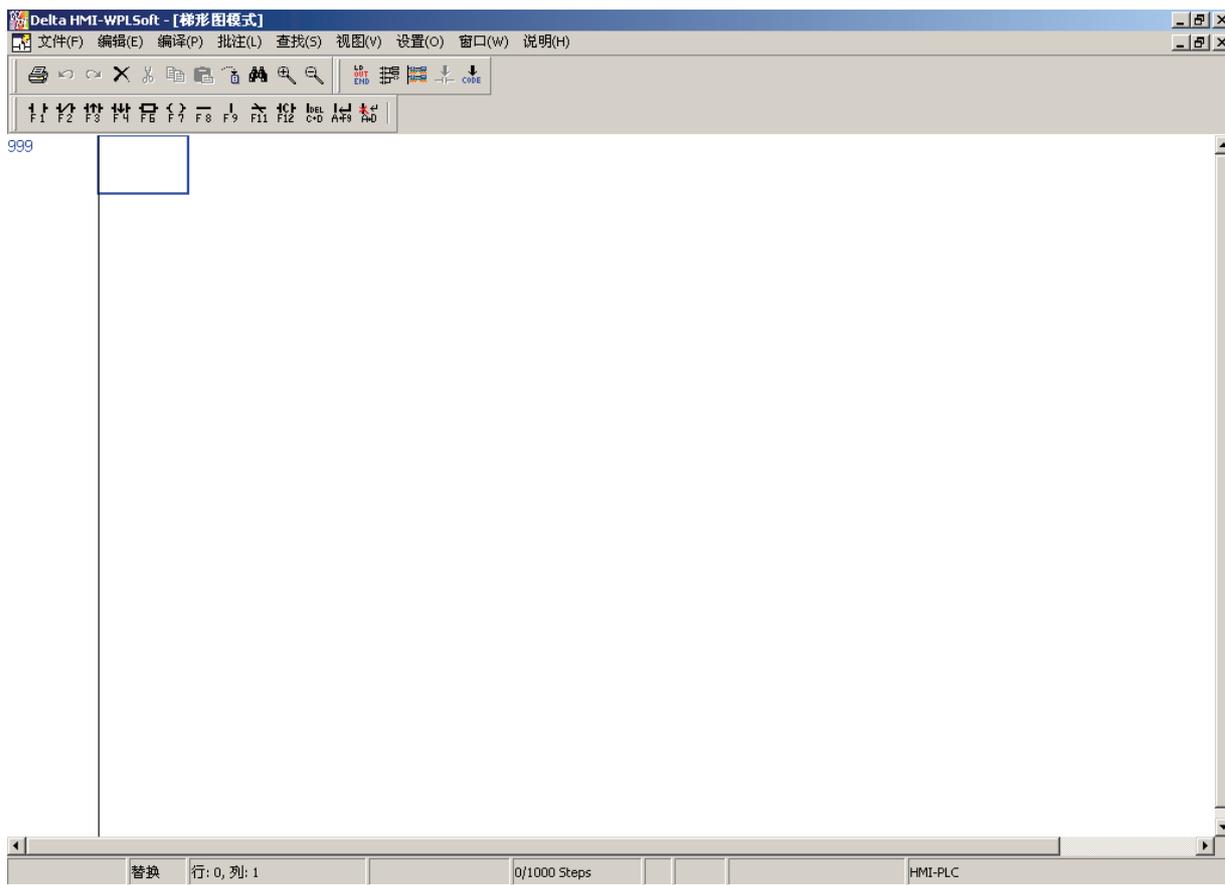
3.1 基本操作

范例：输入下图梯形图例。



■ 鼠标操作和键盘功能键（F1~F12）操作

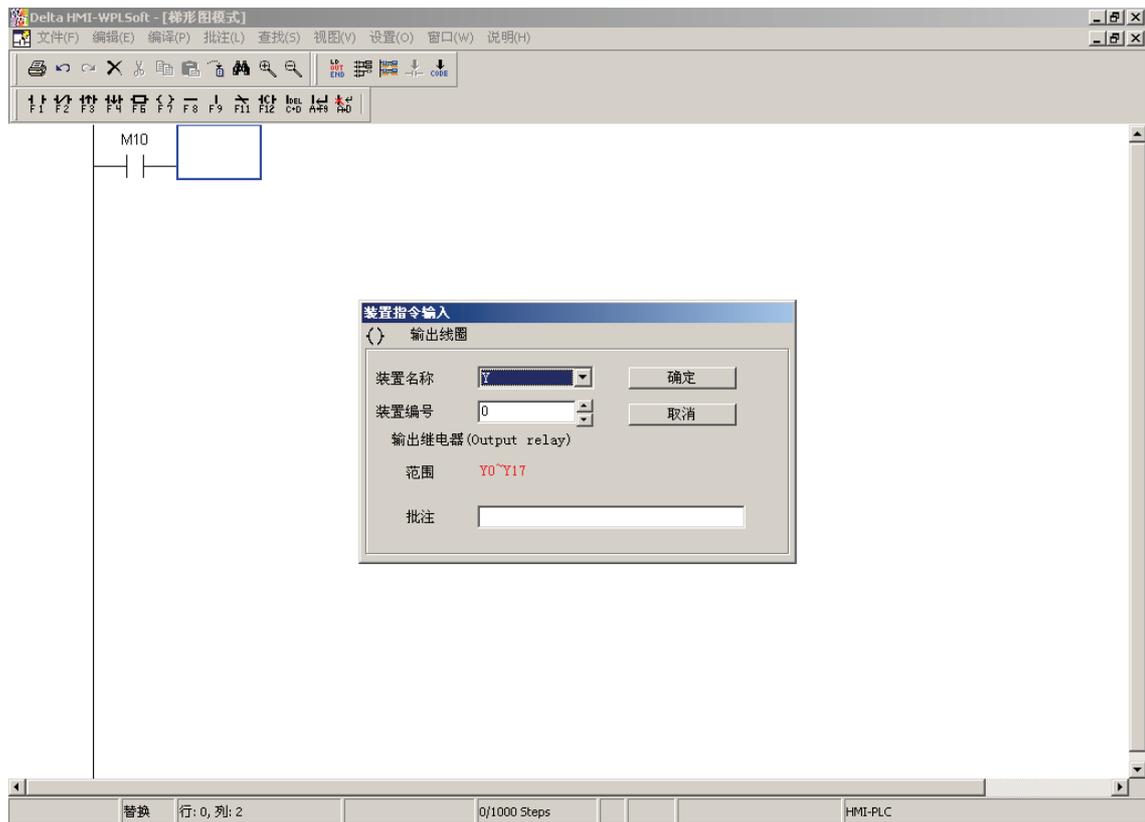
1. 鼠标点选常开接点图示  或按功能键 F1:



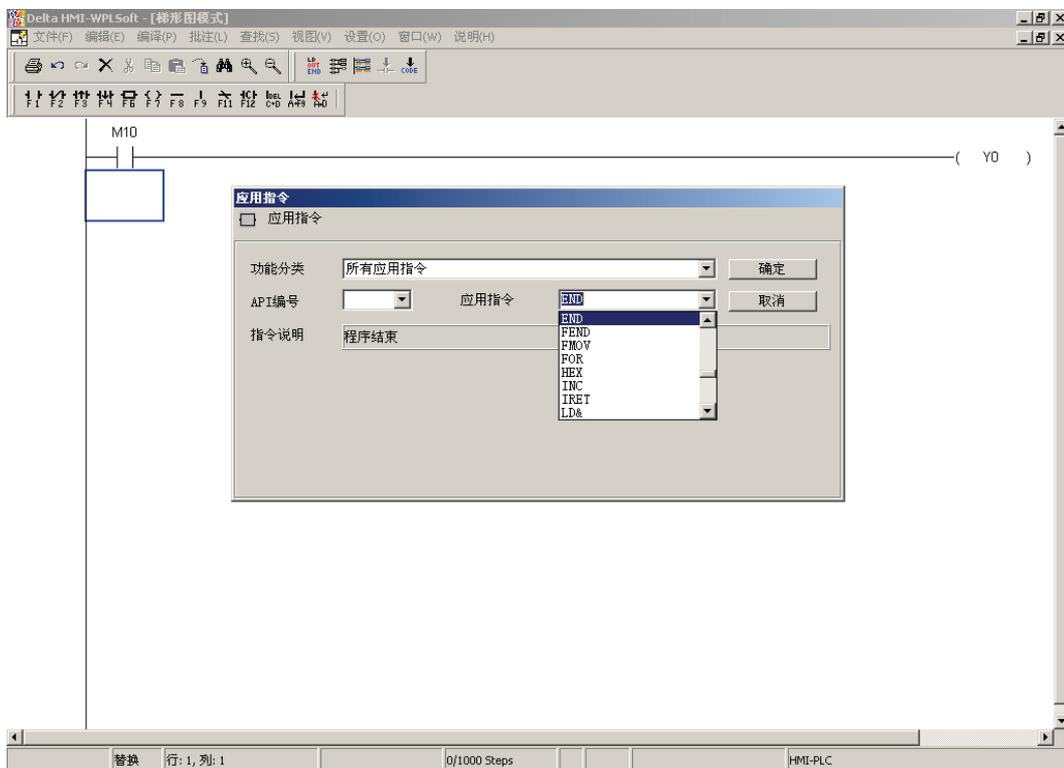
2. 出现输入装置名称与批注对话框后便可选取装置名称(例: M)、装置编号 (例: 10) 和输入批注, 完成后即可按下确定钮。



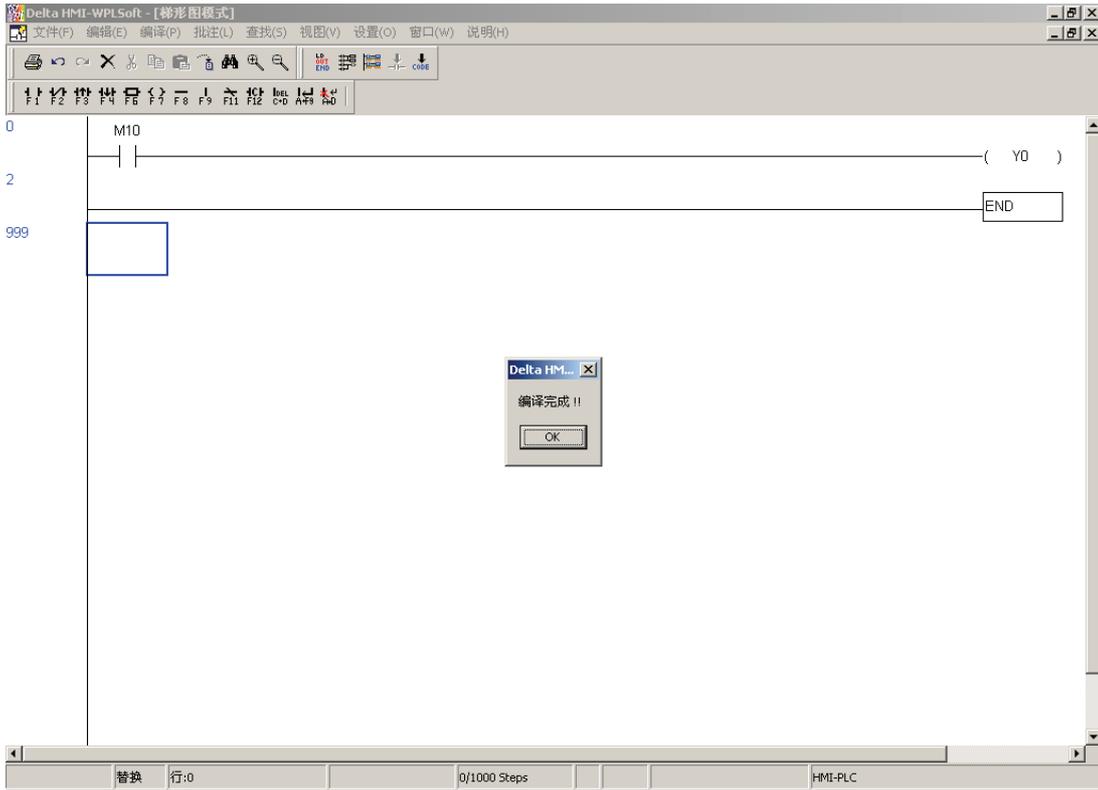
3. 点选输出线圈图示  或按功能键 F7, 出现输入装置名称与批注对话框后选取装置名称(例: Y)、装置编号 (例: 0), 完成后即可按下确定钮。



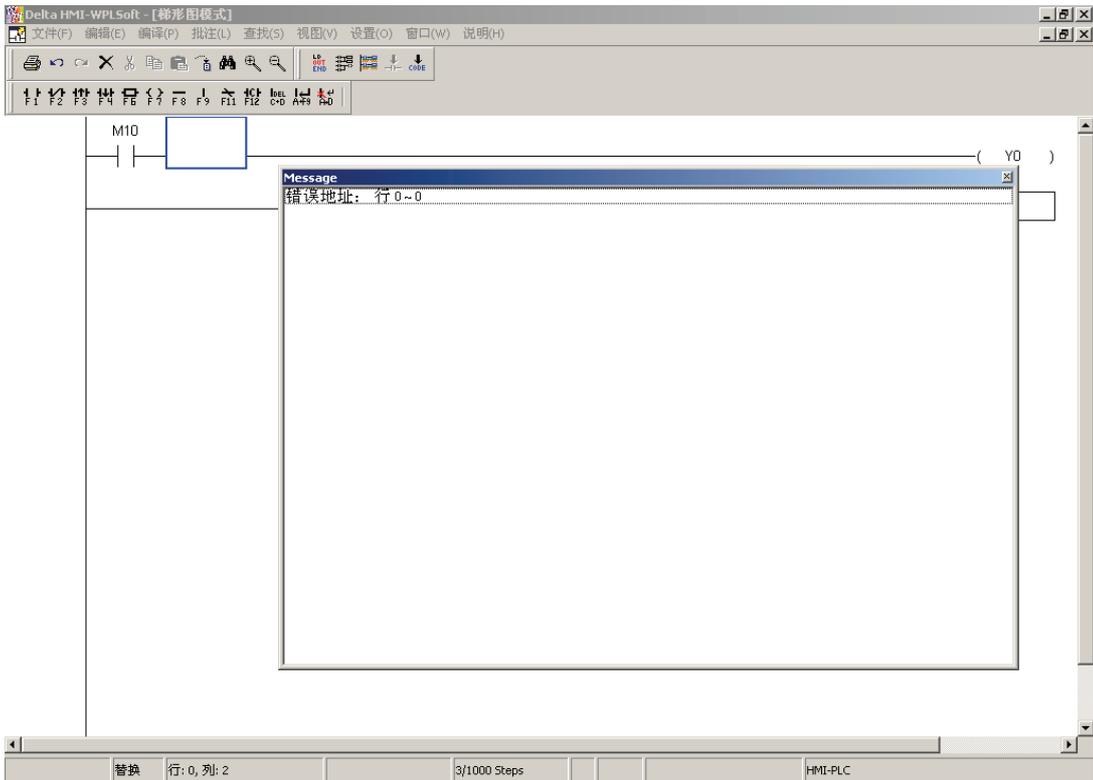
4. 点选应用命令图示  或按功能键 F6，在功能分类栏位中点选「所有应用命令」，在应用命令下拉选单中点选 END 指令或于该栏位直接键盘键入“END”后按下确定钮。



5. 点选  图示，将编辑完成的梯形图作编译转换成指令程序，编译完成后母线左边会出现步数(STEPS)。

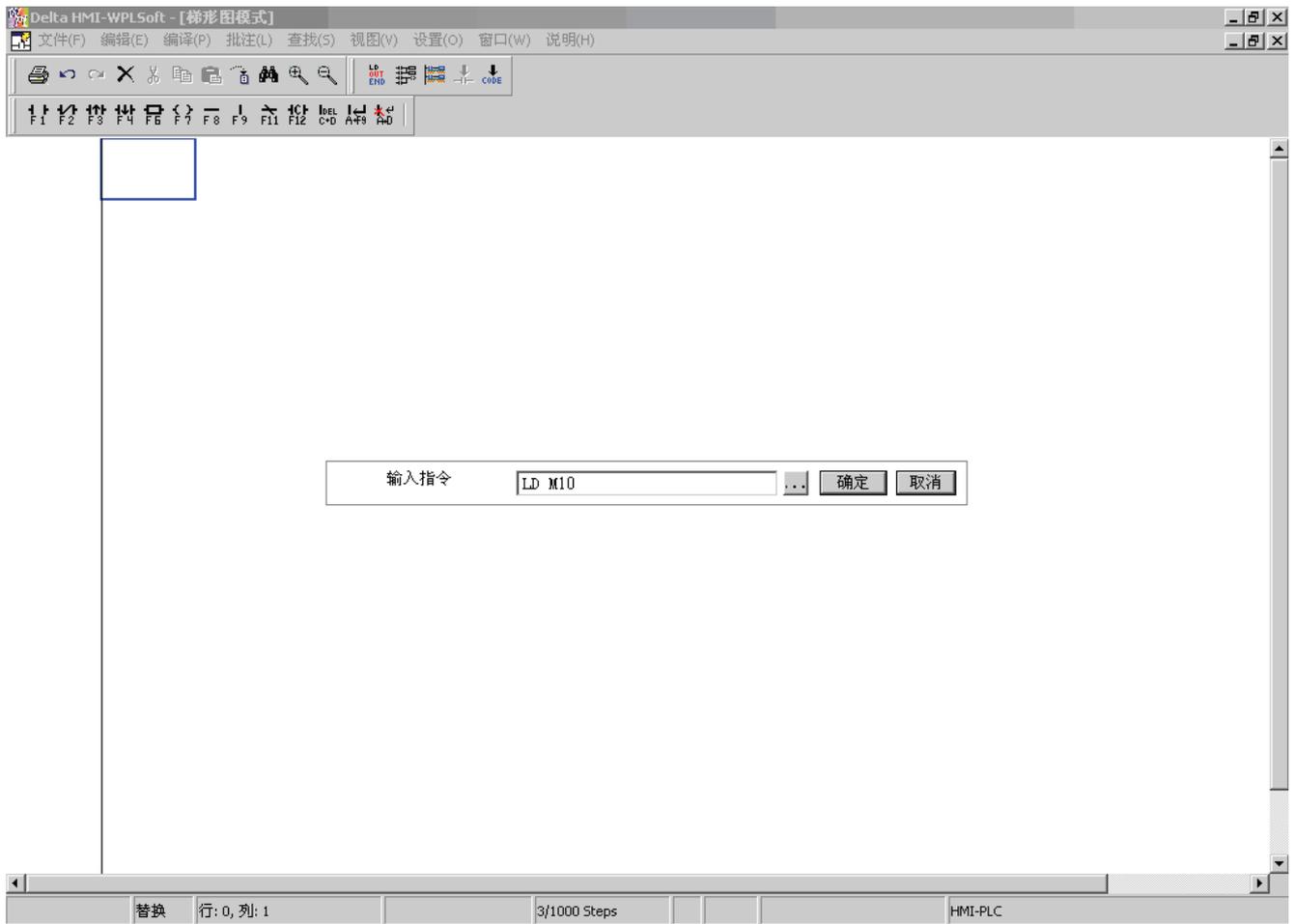


6. 若梯形图图形不正确，则编译后会产生信息对话框指出第几行有误。



■ 键盘指令码输入操作

1. 将编辑方块置于文件开头(行: 0, 列: 1), 由键盘输入 LD M10 按下 Enter 或用鼠标点选确定钮。



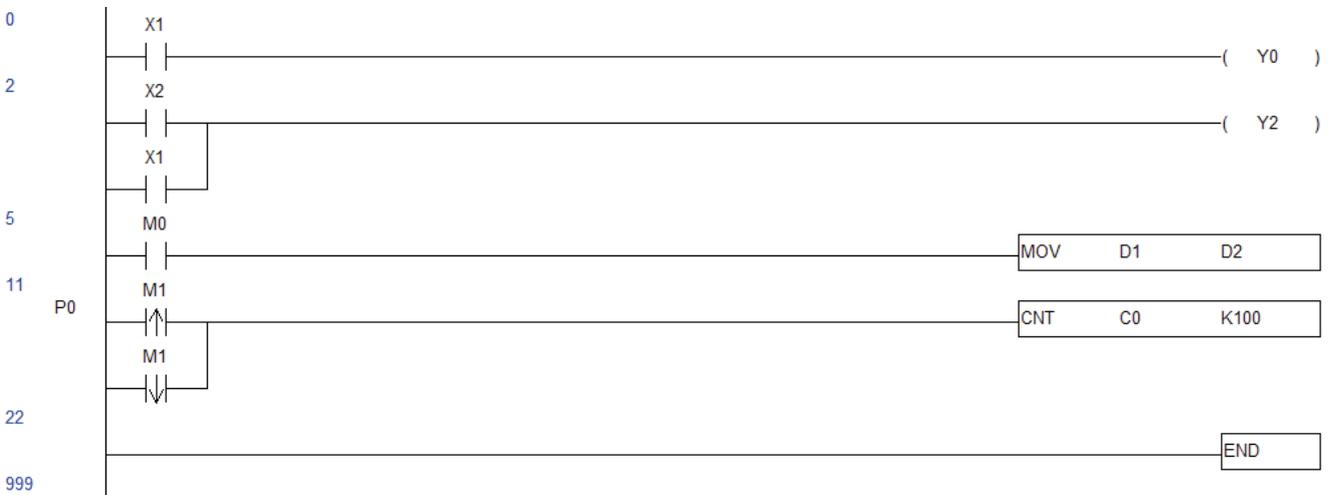
2. 键盘输入 OUT Y0→按下 Enter、键盘输入 END→按下 Enter，最后点选  图示将编辑完成的梯形图作编译。

以键盘指令码输入操作时欲同时输入装置的批注，可于『设置(O)』功能的下拉选单中选取装置批注提示，则指令正确输入后便会出现批注对话框窗口（如下图），此时便可继续输入对应的装置批注。



3.2 编辑范例

梯形图



■ 梯形图编辑操作步骤：

步骤	梯形符号	光标位置	鼠标点选功能键输入方式	键盘输入方式
1	┆┆	行：0，列：1	*注一	组件名称 X 组件编号 1 LD X1 ↓ 或 A X1 ↓
2	┆()	行：0，列：2	*注二	组件名称 Y 组件编号 0 OUT Y0 ↓ 或 O Y0
3	┆┆	行：1，列：1		组件名称 X 组件编号 2 LD X2 ↓ 或 A X2 ↓
4		行：1，列：2		F9
5	┆()	行：1，列：2		组件名称 Y 组件编号 2 OUT Y2 ↓ 或 O Y2 ↓
6	┆┆	行：2，列：1		组件名称 X 组件编号 1 LD X1 ↓ 或 A X1 ↓
7	┆┆	行：3，列：1		组件名称 M 组件编号 0 LD M0 ↓ 或 A M0 ↓
8	┆┆	行：3，列：2	*注三	应用命令 MOV 操作数 1: D 组件值: 1 操作数 2: D 组件值: 2 MOV D1 D2 ↓

※注一：基本指令输入

1. 鼠标点选  或按下键盘功能键 F1 键后进入装置命令输入对话框窗口，可输入装置名称、编号与批注等内容。



2. 分别点选装置名称下拉按钮选取 X 和装置编号下拉按钮选取 1 或以键盘输入装置名称 X 按下 Enter 键，输入装置编号 1，最后按下确定钮。



※注二：输出线圈输入

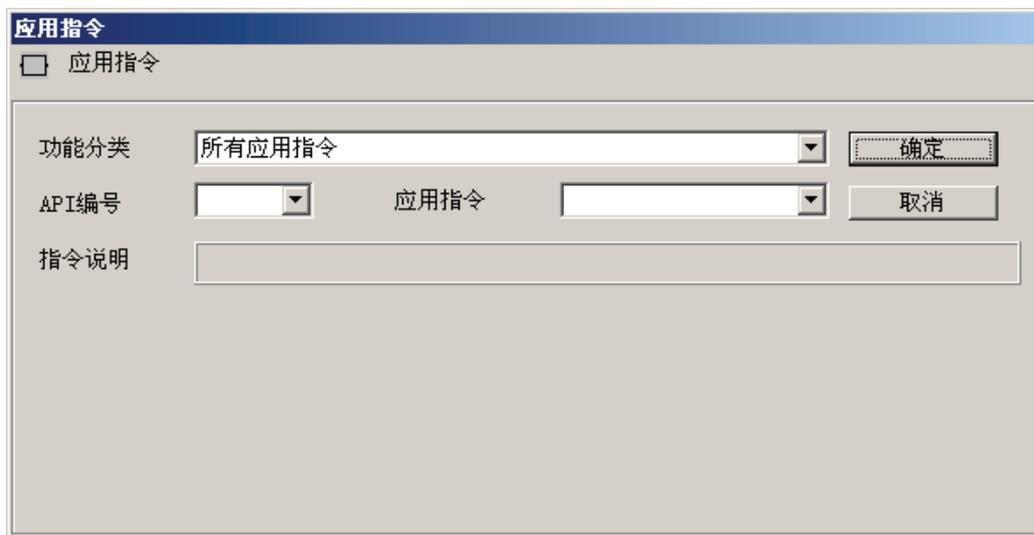
1. 鼠标点选  或按下键盘功能键 F7 键后进入装置命令输入对话框窗口，可输入装置名称、编号与批注等内容。



2. 分别点选装置名称下拉按钮选取 Y 和装置编号下拉按钮选取 1 或以键盘输入装置名称 Y 按下 Enter 键，输入装置编号 1，最后按下确定钮。

※注三：应用命令输入

1. 鼠标点选  或按下键盘功能键 F6 键后出现应用命令对话框窗口。



2. 先选择功能分类栏位（包含所有应用命令、输出命令等...选项）。鼠标点选应用命令栏位下拉按钮即可选择应用命令或直接在应用命令栏位以键盘输入命令名称（例：MOV）后按 Enter 键。
3. 在功能分类选择「传送比较命令」，应用命令栏位键入 MOV 指令后按下 Enter 键。（或用鼠标点选下拉按钮选取 MOV 指令）。即可显示出功能方块对话框窗口，如下图：



4. 依序输入操作数 1、操作数 2 和各组件值。若有索引可点选输入 E 或 F，若无则不用选。如上图所示，设置完成按下确定钮。
5. 另外，使用者亦可选择直接于功能方块下方可用操作数参考表格内以鼠标点选有 @ (表示该装置可 E、F 修饰)或 * (表示该装置不可 E、F 修饰) 符号的装置作为另一种输入方式。

3.3 梯形图编辑说明

☞ 指令简易代码输入

HMI-WPLSoft 提供部分基本指令以简易代码快速输入，可依据如下对照表编辑。

说明	指令图示	指令码	简易代码	范例
常开开关		LD	A	LD M0 或 A M0
常闭开关		LDI	B	LDI M0 或 B M0
上升沿开关		LDP	+	LDP M0 或 + M0
下降沿开关		LDF	-	LDF M0 或 - M0
输出线圈		OUT	O	OUT M0 或 O M0

☞ 插入/替换模式

使用〔Insert〕键可切换编辑时为插入模式或替换模式。

- ◆ 状态栏显示若为替换模式时，按下〔Insert〕键可切换编辑为插入模式。编辑方块所在位置插入梯形符号，其后的梯形符号往后位移一位。
- ◆ 状态栏显示若为插入模式，按下〔Insert〕键可切换编辑为替换模式。从编辑方块所在位置替换原来的梯形符号，其后方的梯形符号位置不变。

☞ 编辑 (E)

■ 撤销：还原上一动作。(最多 10 次)

- ◆ 方法一：从功能工具栏上选取「编辑 (E)」功能表中『撤销(U)』命令，可回复前一次的状态，最多可回复前 10 次的动作。
- ◆ 方法二：鼠标点选图示工具栏上的
- ◆ 方法三：利用快捷键，键盘输入复合键〔Ctrl〕+〔Z〕。
- ◆ 方法四：鼠标右键功能表中「撤销」命令。

重复⇒ 重做撤销前的动作。

- ◆ 方法一：从功能工具栏上选取「编辑 (E)」功能表中『重复(R)』命令，在撤销动作的后选择『重复(R)』动作则可重回前一次撤销前的状态。

- ◆ 方法二：鼠标点选图示工具栏上的 。
- ◆ 方法三：利用快捷键，键盘输入复合键〔Ctrl〕+〔Alt〕+〔Z〕。
- ◆ 方法四：鼠标右键功能表中「重复」命令。

■ 组件的删除

- ◆ 方法一：从功能工具栏上选取「编辑 (E)」功能表中『删除』命令，就会把目前编辑方块或标示区块内的梯形符号删除，且编辑方块不动。
- ◆ 方法二：鼠标点选图示工具栏上的 。
- ◆ 方法三：利用快捷键，键盘按下〔Delete〕键。
- ◆ 方法四：将编辑方块移到所要删除的梯形符号上，按下鼠标右键功能表(※注一)中「删除」命令。



整行的删除

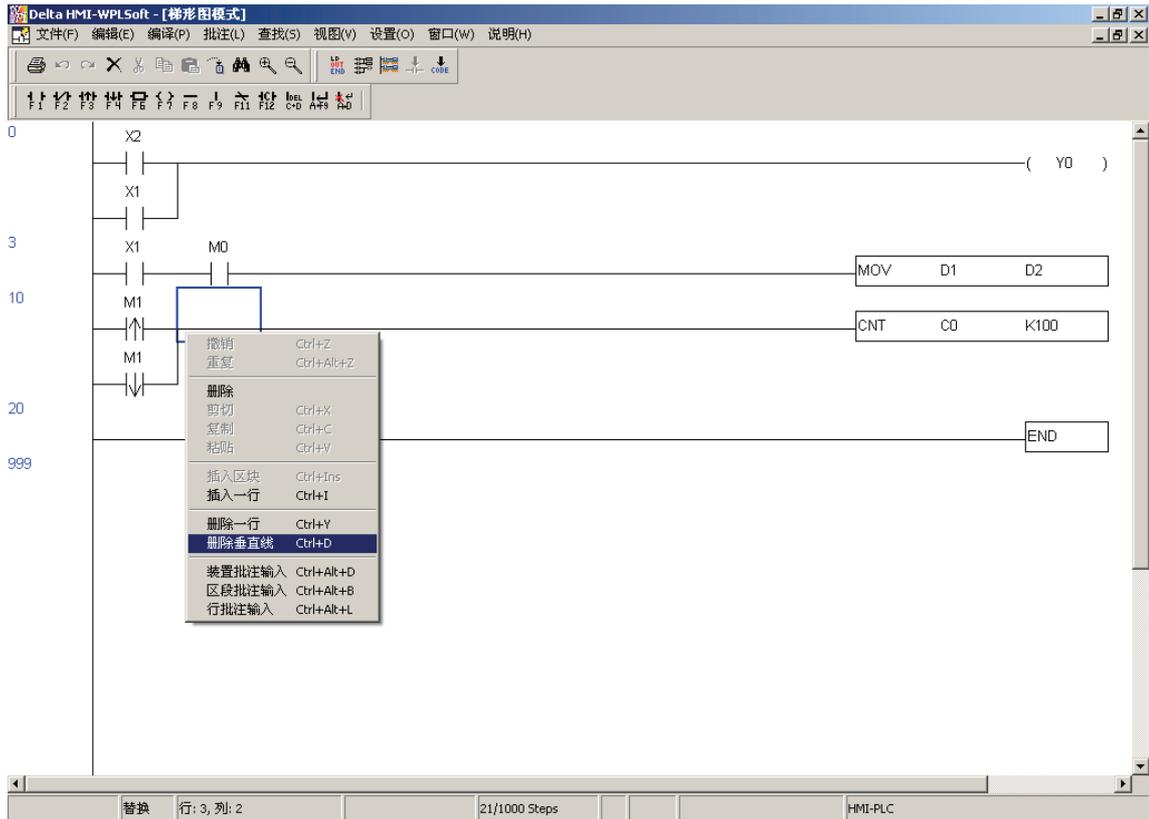
- ◆ 方法一：从功能工具栏上选取「编辑 (E)」功能表中『删除一行(L)』命令，就会删除目前编辑方块所在的这一行，该行以下梯形图会往上合并。
- ◆ 方法二：利用快捷键，键盘输入复合键〔Ctrl〕+〔Y〕。
- ◆ 方法三：将编辑方块移到所要删除的行上，按下鼠标右键功能表中「删除一行」命令。



- ◆ 方法四：将欲删除的一整行区块标示出来，按下鼠标右键功能表中「删除」命令或利用快捷键在键盘按下〔Delete〕键或鼠标点选图示 。

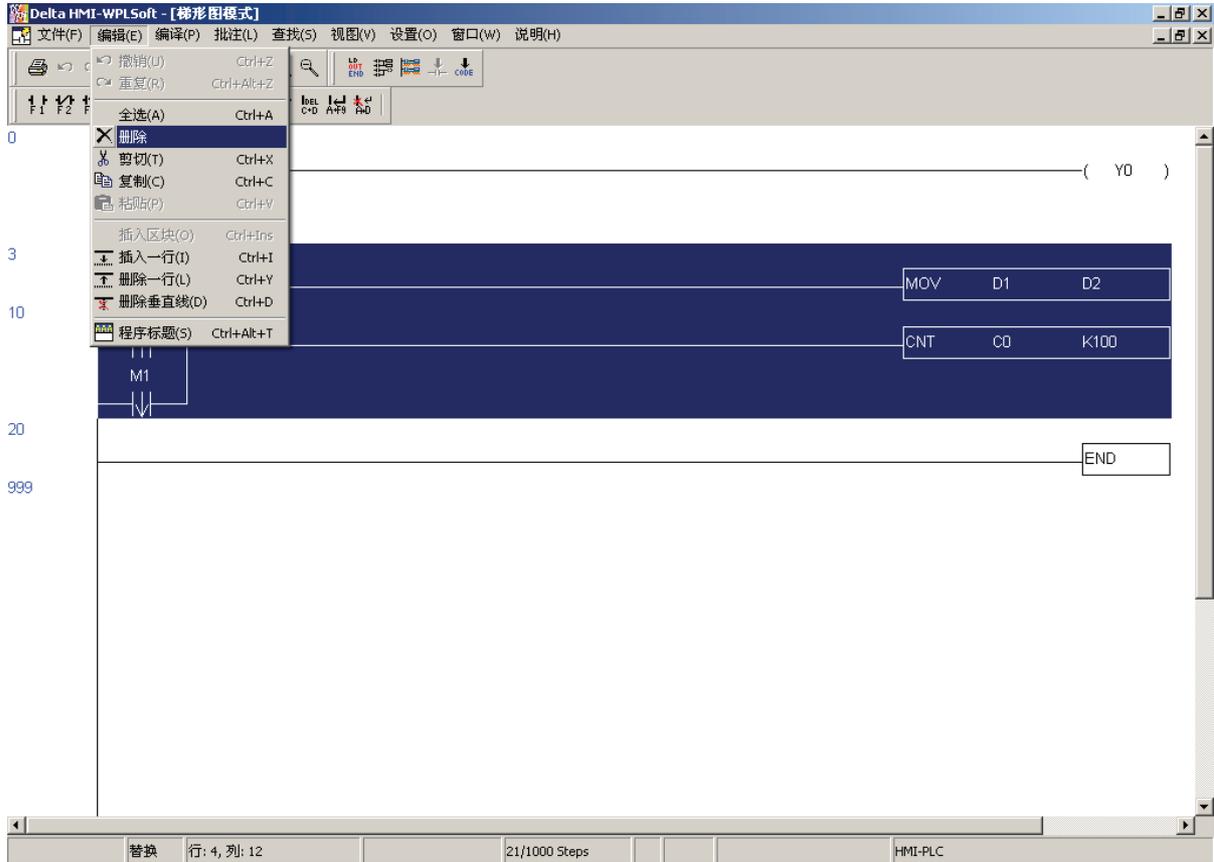
垂直线的删除

- ◆ 方法一：从功能工具栏上选取「编辑 (E)」功能表中『删除垂直线(D)』命令，可删除目前编辑方块所在的左侧的垂直线。
- ◆ 方法二：利用快捷键，键盘输入复合键 [Ctrl] + [D]。
- ◆ 方法三：将编辑方块置于欲删除的垂直线右侧，鼠标点选 ，可将垂直线删除。
- ◆ 方法四：将编辑方块置于欲删除的垂直线右侧，按下鼠标右键功能表中「删除垂直线」命令如下图，点选删除垂直线。



■ 区块的删除

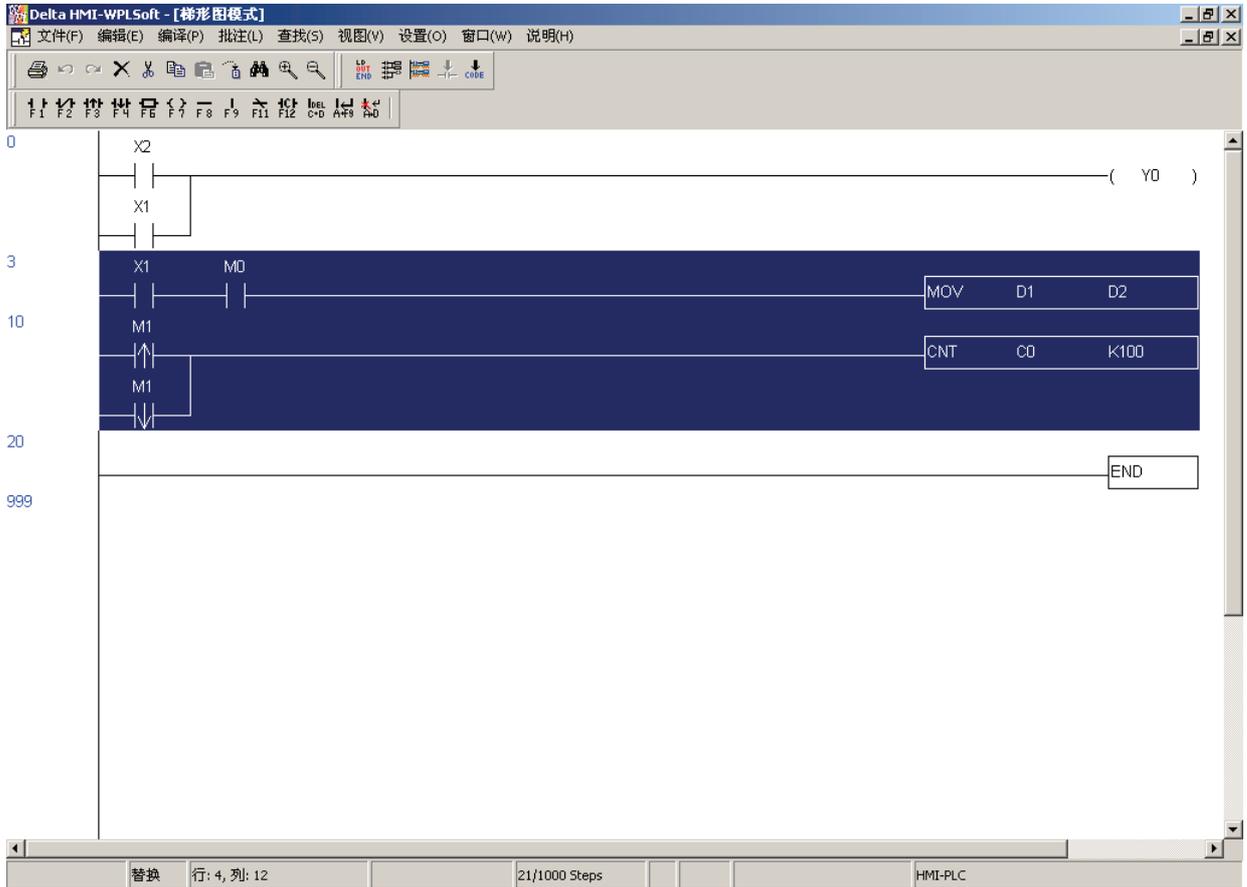
◆ 方法一：「编辑 (E)」功能表中『删除』命令，将文件中的标示区块删除。



- ◆ 方法二：将区块标示后，鼠标点选图示工具栏上的 。
- ◆ 方法三：当区块标示后，鼠标右键功能表中『删除』命令。
- ◆ 方法四：当区块标示后，利用快捷键，键盘按下〔Delete〕键。

■ 区块的复制

- ◆ 方法一：「编辑 (E)」功能表中『复制(C)』命令。
- ◆ 方法二：鼠标点选图示工具栏上的 。
- ◆ 方法三：利用快捷键，键盘输入复合键 [Ctrl] + [C]。
- ◆ 方法四：鼠标右键功能表中『复制』命令。



区块的剪切

- ◆ 方法一：「编辑 (E)」功能表中『剪切(T)』命令。
- ◆ 方法二：鼠标点选图示工具栏上的 。
- ◆ 方法三：利用快捷键，键盘输入复合键 [Ctrl] + [X]。
- ◆ 方法四：鼠标右键功能表中『剪切』命令。

区块粘贴

- ◆ 方法一：「编辑 (E)」功能表中『粘贴(P)』命令。
- ◆ 方法二：鼠标点选图示工具栏上的 。
- ◆ 方法三：利用快捷键，键盘输入复合键 [Ctrl] + [V]。
- ◆ 方法四：鼠标右键功能表中『粘贴』命令。

插入区块（需先执行区块复制才可插入区块）

- ◆ 方法一：「编辑（E）」功能表中『插入区块(O)』命令。
- ◆ 方法二：利用快捷键，键盘输入复合键〔Ctrl〕+〔Ins〕。
- ◆ 方法三：鼠标右键功能表中『插入区块』命令。

撤销	Ctrl+Z
重复	Ctrl+Alt+Z
删除	
剪切	Ctrl+X
复制	Ctrl+C
粘贴	Ctrl+V
插入区块	Ctrl+Ins
插入一行	Ctrl+I
删除一行	Ctrl+Y
删除垂直线	Ctrl+D
装置批注输入	Ctrl+Alt+D
区段批注输入	Ctrl+Alt+B
行批注输入	Ctrl+Alt+L

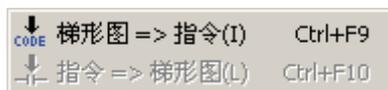
☞ 编译（P）

此功能是用来编译目前的 EXIO 程序。若使用者在梯形图模式时梯形图编辑完毕，执行此功能便会检查梯形图是否合法，若转换无误，则会将梯形图转换成指令码程序，同时梯形图编辑区母线左侧会出现梯形图形每个区块相对于程序存储器的地址（STEP）。若有错误，HMI-WPLSoft 会发出信息，将错误发生的所在行、列和错误码显示出来。

若使用者在指令模式时编辑完成执行此功能便会作检查，若转换无误，则会将指令程序转换成梯形图。若有错误，HMI-WPLSoft 会发出信息将错误发生的所在步数（STEP）和错误码显示出来。

■ 梯形图 → 指令（梯形图模式有效）

- ◆ 方法一：从功能工具栏上选取编译（P）中『梯形图=>指令(I)』命令。



- ◆ 方法二：鼠标点选  图示。
- ◆ 方法三：键盘输入复合键〔Ctrl〕+〔F9〕。

■ 指令 -> 梯形图 (指令模式有效)

◆ 方法一：从功能工具栏上选取编译 (P) 中『指令=>梯形图(L)』命令。



◆ 方法二：鼠标点选  图示。

◆ 方法三：键盘输入复合键 [Ctrl] + [F10]。

☞ 查找 (S)

跳转

使用此命令可指定程序要跳到哪一个 STEP, 如果输入的 STEP 存在, 则会跳到该 STEP 上, 并将找到的该 STEP 放在第一行。

◆ 方法一：从功能工具栏上选取『查找 (S)』中『跳转(J)』命令。

于对话框窗口内输入欲跳转的 STEP 地址, 则梯形图程序会将所指定的 STEP 地址程序置于第一行。

◆ 方法二：鼠标点选  图示。

◆ 方法三：键入复合键 [Ctrl] + [J]。

■ 查找/替换

可输入欲查找/替换的组件名称 (若仅作查找动作, 只须于查找组件对话框窗口内输入查找组件名称即可), 并可加入此组件指令型态作查找。

◆ 方法一：从功能工具栏上选取查找 (S) 中『查找/替换(F)』命令。

◆ 方法二：鼠标点选  图示。

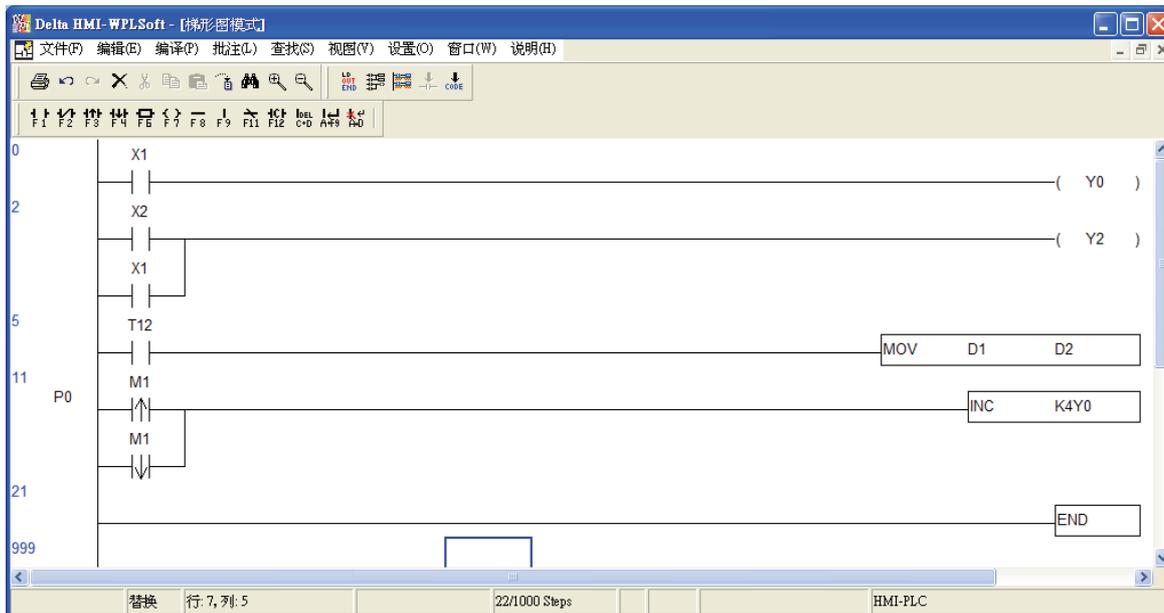
◆ 方法三：键入复合键 [Ctrl] + [F]。

使用此命令可打开一个查找/替换对话框窗口 (如下图), 在此对话框窗口主要有四种功能, 分别为查找组件、替换组件、查找命令和其他命令替换。

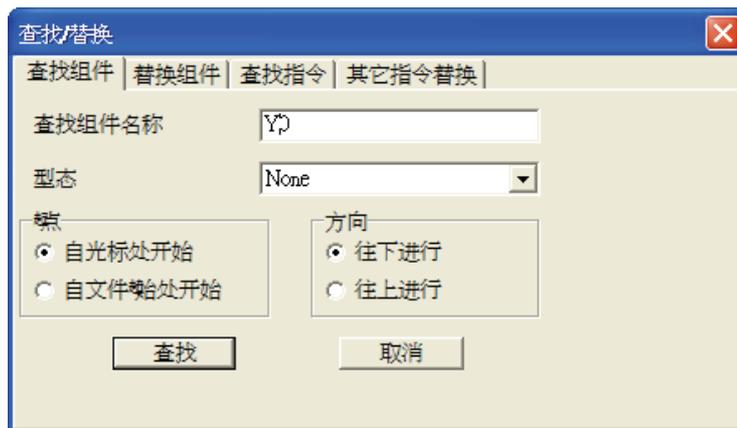


◆ 查找组件：

一范例程序如下图，程序中包含 Y0 组件的命令有 OUT Y0 与 INC K4Y0；

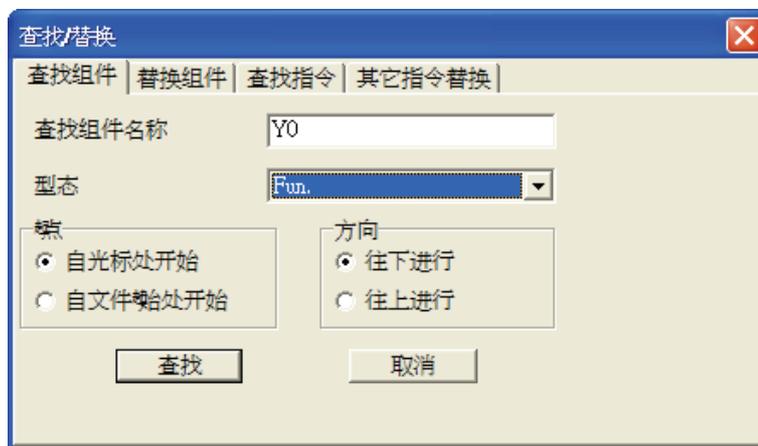


打开执行查找组件对话框窗口并输入查找组件名称 Y0，型态 None 如下图的条件；



则查找组件会找到 OUT Y0 与 INC K4Y0 这二个条件符合的命令；

若打开执行查找/替换对话框窗口并输入查找组件名称 Y0，型态 Fun. 如下图的条件；



则查找组件仅会找到 INC K4Y0 一个条件符合的命令。

◆ 替换组件：

打开执行替换组件对话框窗口，例：输入查找组件名称 X0，型态 LD；替换组件名称 M100，型态 LD 和欲替换的装置数 10，如下图执行替换命令，则程序中的符合上述条件的命令均会被替换为 LD M100~M109。

原始命令	替换条件	替换后命令
LD X0~X7	型态 LD+组件 X0→型态 LD+组件 M100	LD M100~M107
LD X10~X11	装置数 10	LD M108~M109



在替换组件功能中的指令型态项目若指定 None、Out 和 Fun.这三种型态将仅能针对相同组件名称进行替换工作，否则会出现如下图的警示信息。



另有选项勾选是否将批注复制到替换的装置内，即将 X0 批注复制到 M100，并询问是否将被替换批注删除，即将 X0 批注删除。

✎ 限制条件

替换组件功能内的装置组件型态必须相同方可替换，如将 D1 替换为 D11 则替换成功，D1 替换为 C100 替换失败。

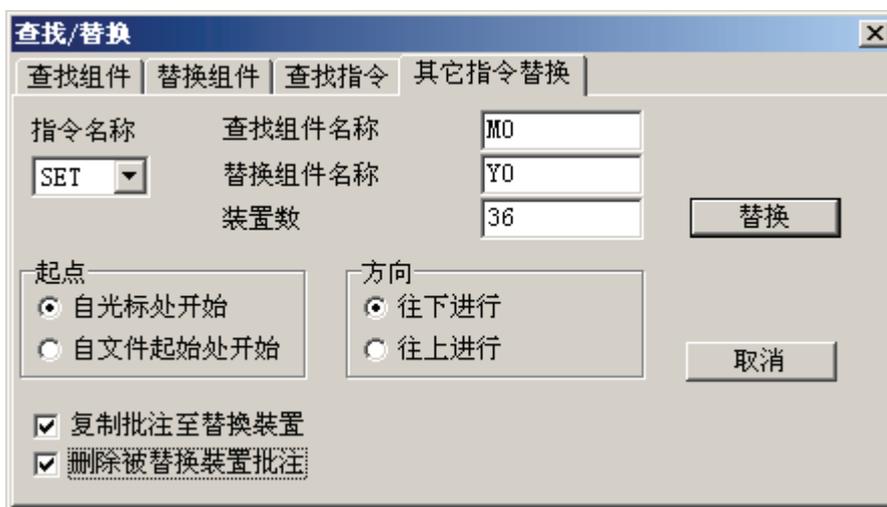
◆ 查找命令：

打开执行查找命令对话框窗口，于命令输入栏指定欲查找的指令名称，此功能亦会记录输入过的指令以便后续直接点选。



◆ 其他命令替换：

在查找/替换中的其他命令替换功能提供 SET、RST、PLS 与 PLF 四个命令的条件替换，可针对程序中符合这些命令码条件的组件作替换。例：欲将程序中 SET M0 ~ M35 全部替换为 SET Y0 ~ Y43 可如下图方式设置。



另有选项勾选是否将批注复制到替换的装置内，即将 M0 ~ M35 批注复制到 Y0 ~ Y43，并询问是否将被替换批注删除，即将 M0 ~ M35 批注删除。

■ 跳至程序起点

直接跳到程序的起点

◆ 方法一：从功能工具栏上选取『查找 (S)』中『跳至程序起点(T)』命令。

◆ 方法二：键入复合键 [Ctrl] + [Home]。

■ 跳至程序终点

直接跳到程序中最后一行

- ◆ 方法一：从功能工具栏上选取『查找 (S)』中『跳至程序终点(N)』命令。
- ◆ 方法二：键入复合键 [Ctrl] + [End]。

3.4 指令编辑操作

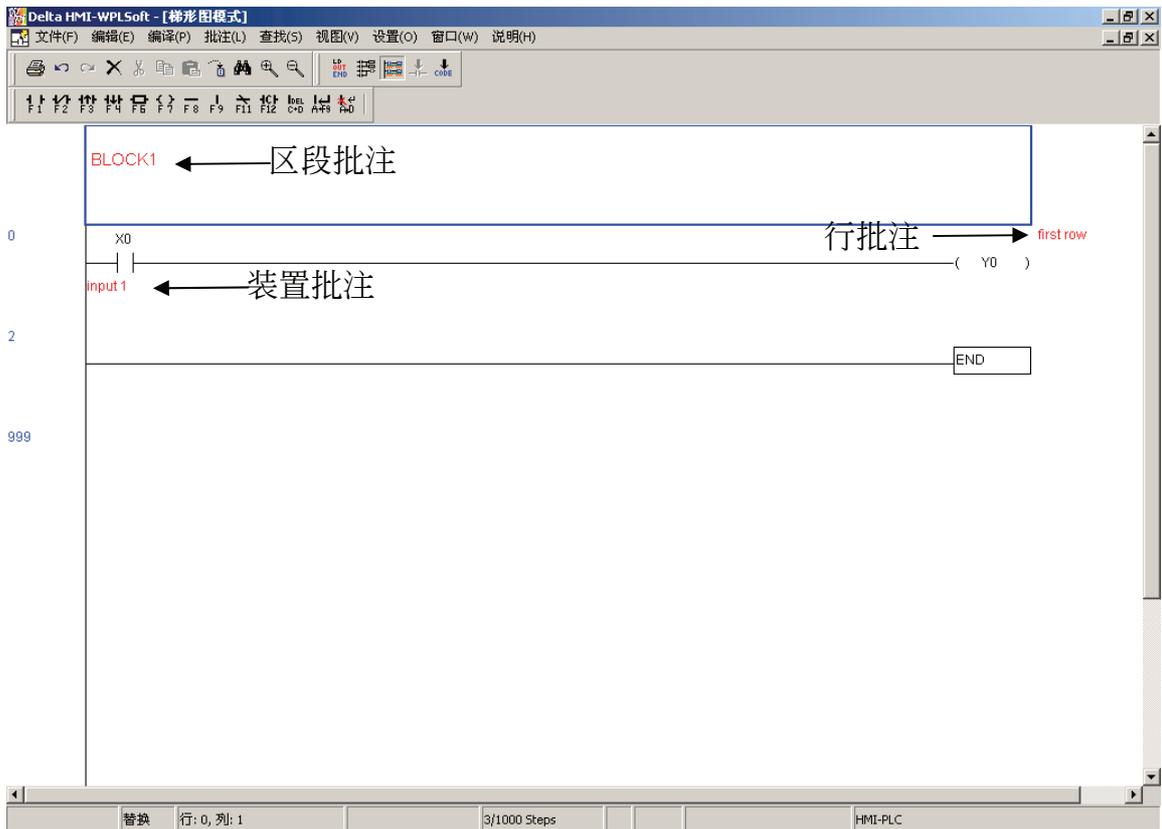
■ 输入 PLC 指令

在指令模式下，使用者可以直接输入指令，若指令的格式合法，按下 [Enter] 键就完成输入。输入完成后的指令在编辑区中，左边为该指令在 EXIO 的程序存储器地址，使用者可以清楚地得到指令在程序存储器的相对地址。各指令格式请参考本说明书附录 A 和附录 B。

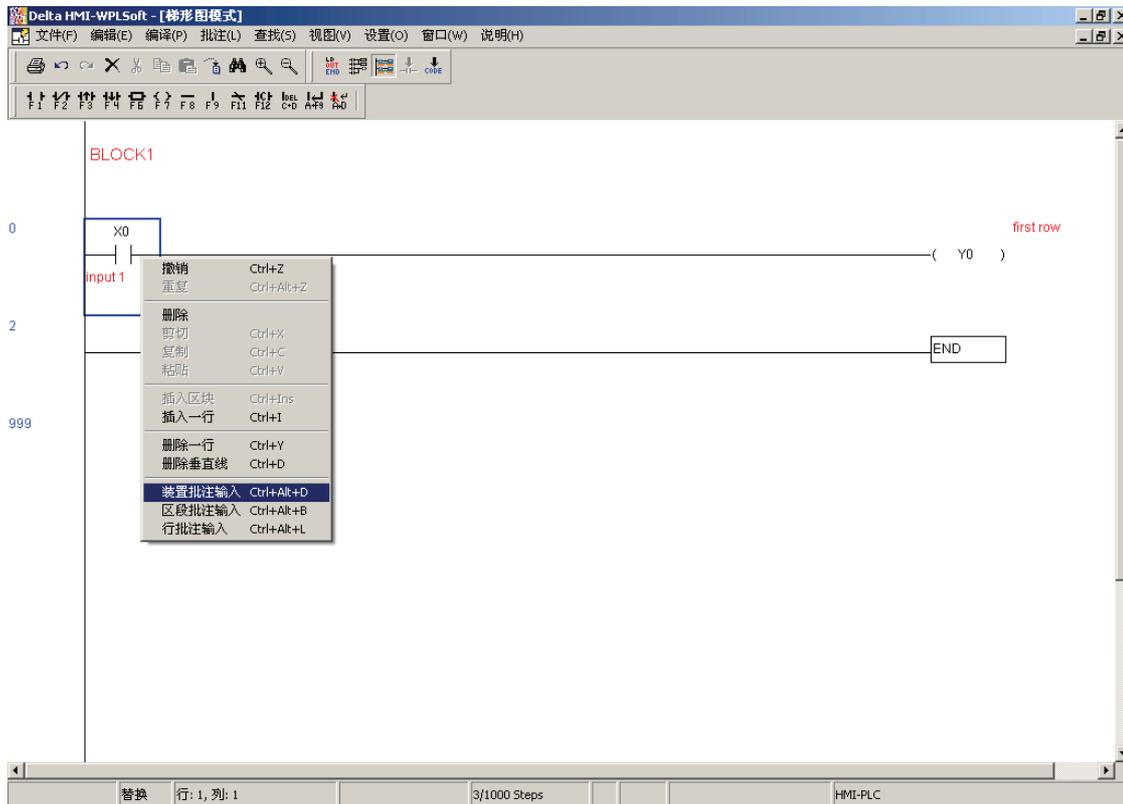
3.5 批注编辑

在梯形图编辑模式批注编辑包含装置批注、行批注和区段批注等三种，在指令编辑模式批注编辑仅有装置批注，底下我们将一一介绍：

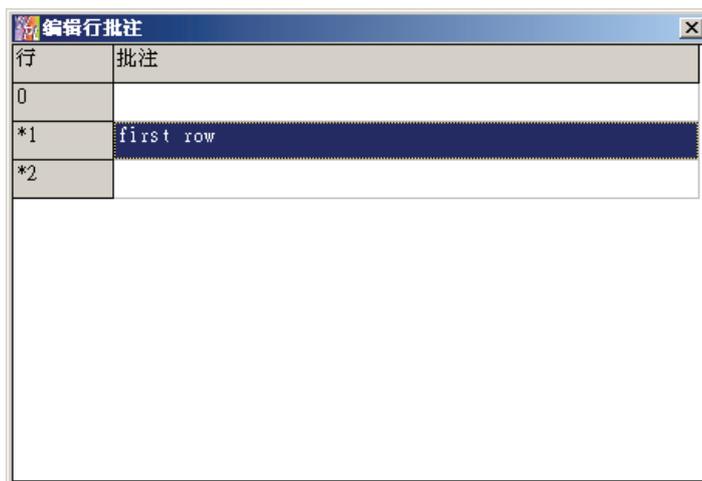
梯形图编辑模式：



- 编辑装置批注对话框窗口：将编辑方块置于有组件的命令上，可以打开装置批注输入的对话框，于此对话框窗口可编辑输入批注内容，完成后按下 **Enter** 键或鼠标按下关闭键即会记录保存。



- 编辑行批注对话框窗口：打开此可同时编辑所有行批注。



- 编辑区段批注对话框窗口：输入完成后按确定。

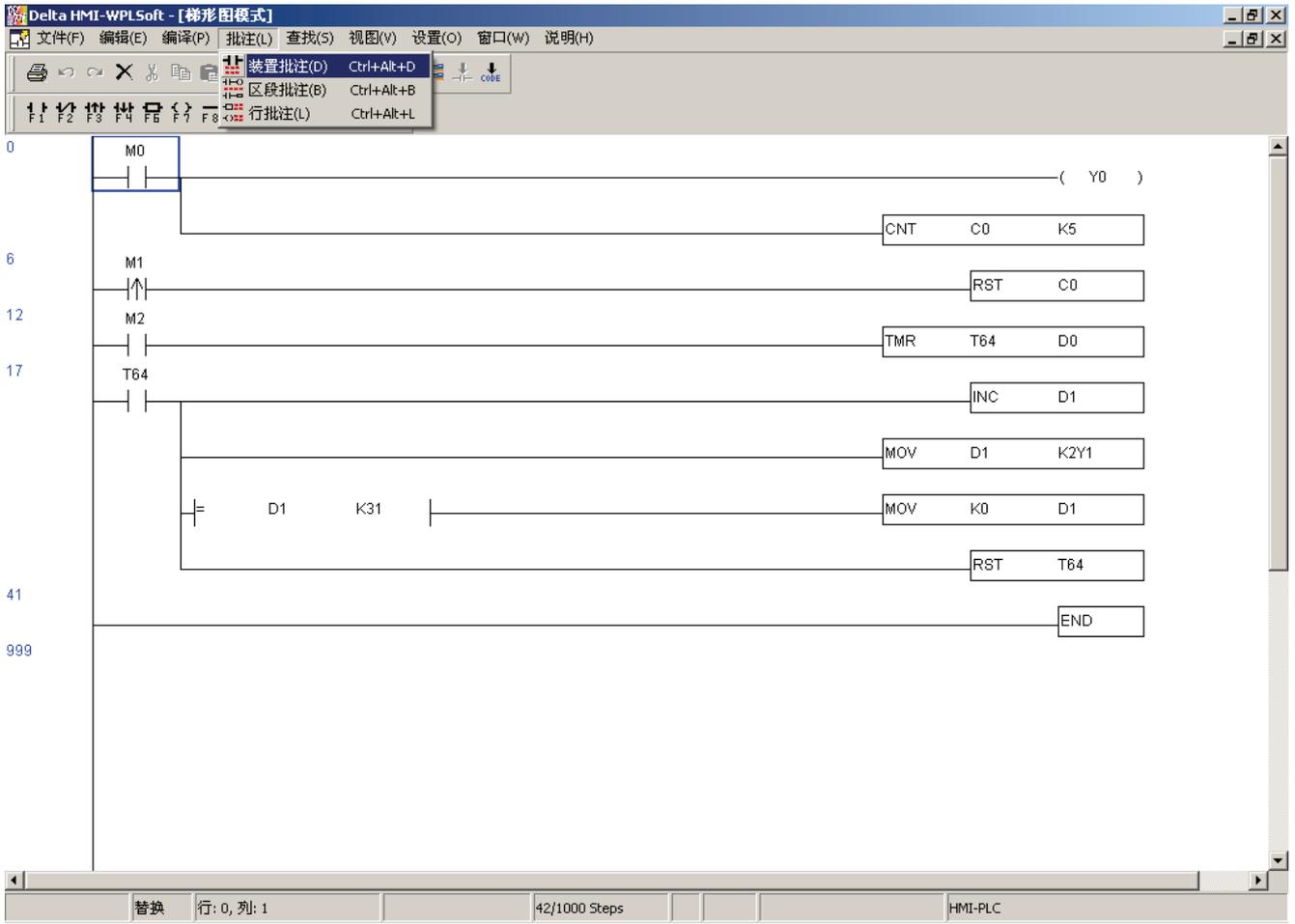


3.6 装置批注

在梯形图编辑模式或指令编辑模式，使用者可以对装置组件做批注编辑。

◆方法一：

1. 首先选取梯形图模式(或指令编辑模式)，再将编辑方块放在欲编辑的装置上，接下来选择功能工具栏上『批注 (L)』功能，点选装置批注（或键入复合键 [Ctrl] + [Alt] + [D]）。



2. 出现对话框窗口选取要编辑批注的装置（例：M0）→在输入批注栏内输入批注→按关闭钮或 ENTER 键。



3. 若欲在梯形图模式下显示批注可点选切换显示批注图示 ，或选择功能工具栏上『视图 (V)』功能，点选显示批注。

◆方法二：

1. 选取梯形图模式(或指令编辑模式)，将编辑方块移至欲批注的装置上（例：T64），按下鼠标右键出现一快捷操作框。

梯形图模式：

撤销	Ctrl+Z
重复	Ctrl+Alt+Z
删除	
剪切	Ctrl+X
复制	Ctrl+C
粘贴	Ctrl+V
插入区块	Ctrl+Ins
插入一行	Ctrl+I
删除一行	Ctrl+Y
删除垂直线	Ctrl+D
装置批注输入	Ctrl+Alt+D
区段批注输入	Ctrl+Alt+B
行批注输入	Ctrl+Alt+L

指令编辑模式：

全选	Ctrl+A
撤销	Ctrl+Z
重复	Ctrl+Alt+Z
删除	
剪切	Ctrl+X
复制	Ctrl+C
粘贴	Ctrl+V
插入一行	Ctrl+I
装置批注	Ctrl+Alt+D
导出至Excel	

2. 点选装置批注输入便会出现一对话框窗口，如下图，先点选欲编辑批注的装置组件（例 T64）再于输入批注栏内输入批注名称，输入完成按下关闭钮。



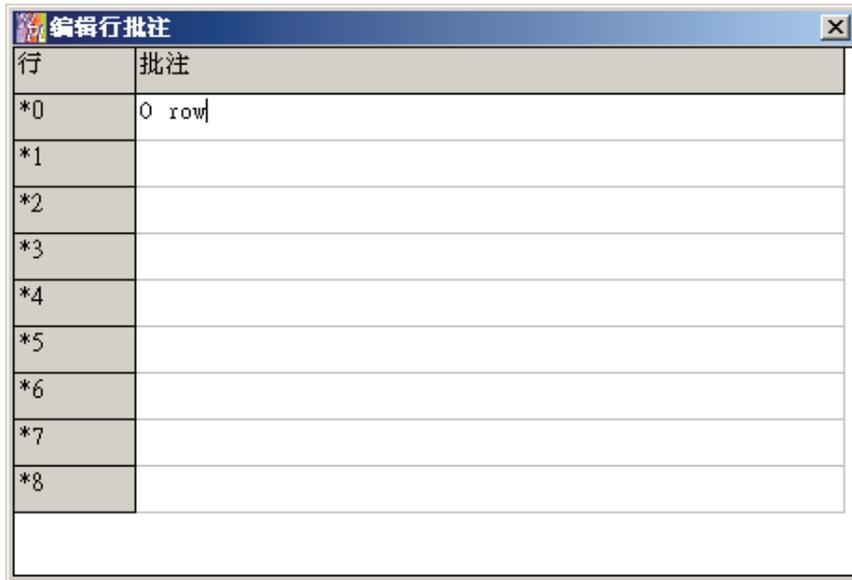
3.7 行批注

◆方法一：

1. 编辑方块移至欲输入行批注的行上，按下鼠标右键出现一快捷操作框，点选行批注输入。

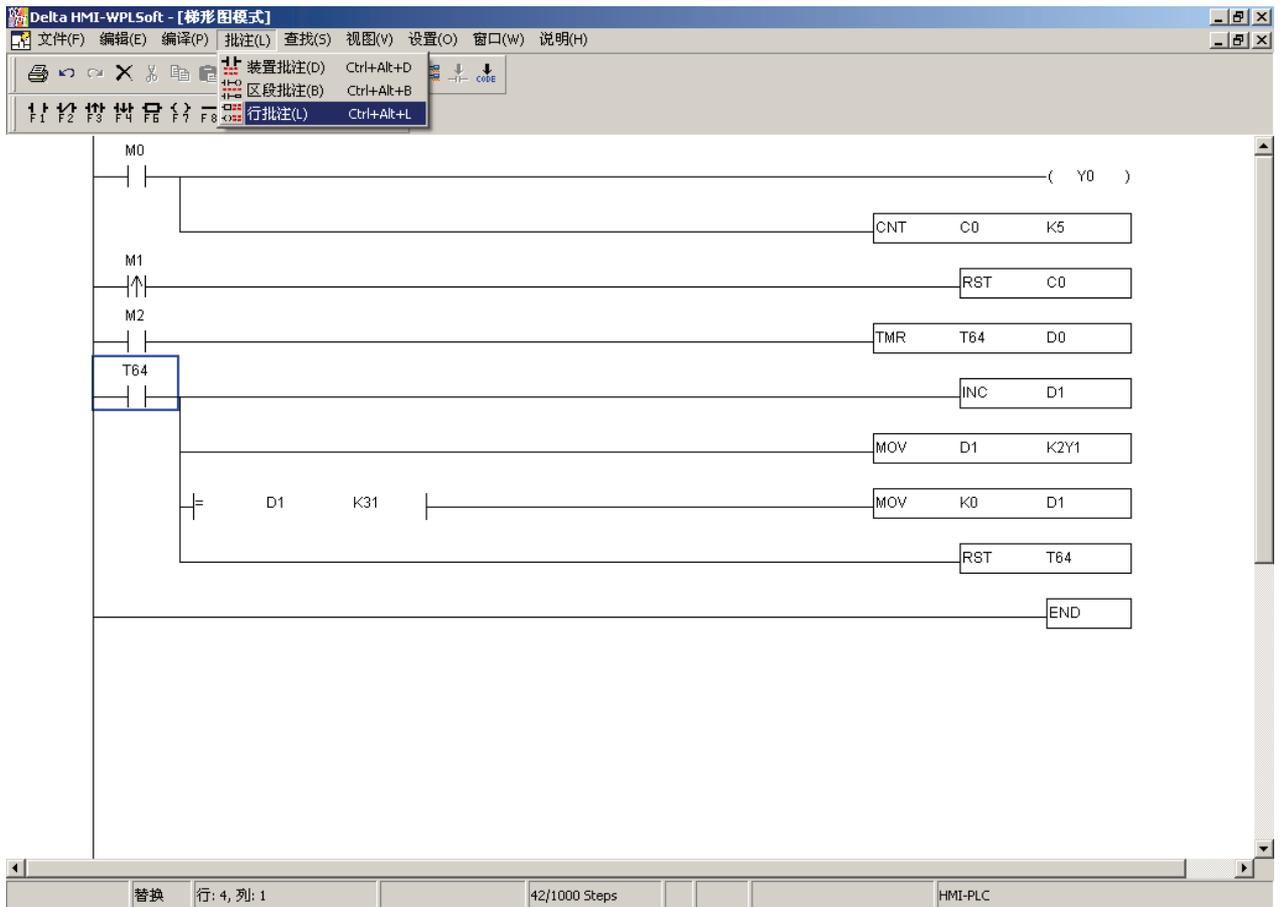
撤销	Ctrl+Z
重复	Ctrl+Alt+Z
删除	
剪切	Ctrl+X
复制	Ctrl+C
粘贴	Ctrl+V
插入区块	Ctrl+Ins
插入一行	Ctrl+I
删除一行	Ctrl+Y
删除垂直线	Ctrl+D
装置批注输入	Ctrl+Alt+D
区段批注输入	Ctrl+Alt+B
行批注输入	Ctrl+Alt+L

2. 出现编辑行批注对话框窗口，如下图，于行批注栏输入批注内容，可同时编辑多行输出批注，完成直接关闭窗口即可。



◆方法二：

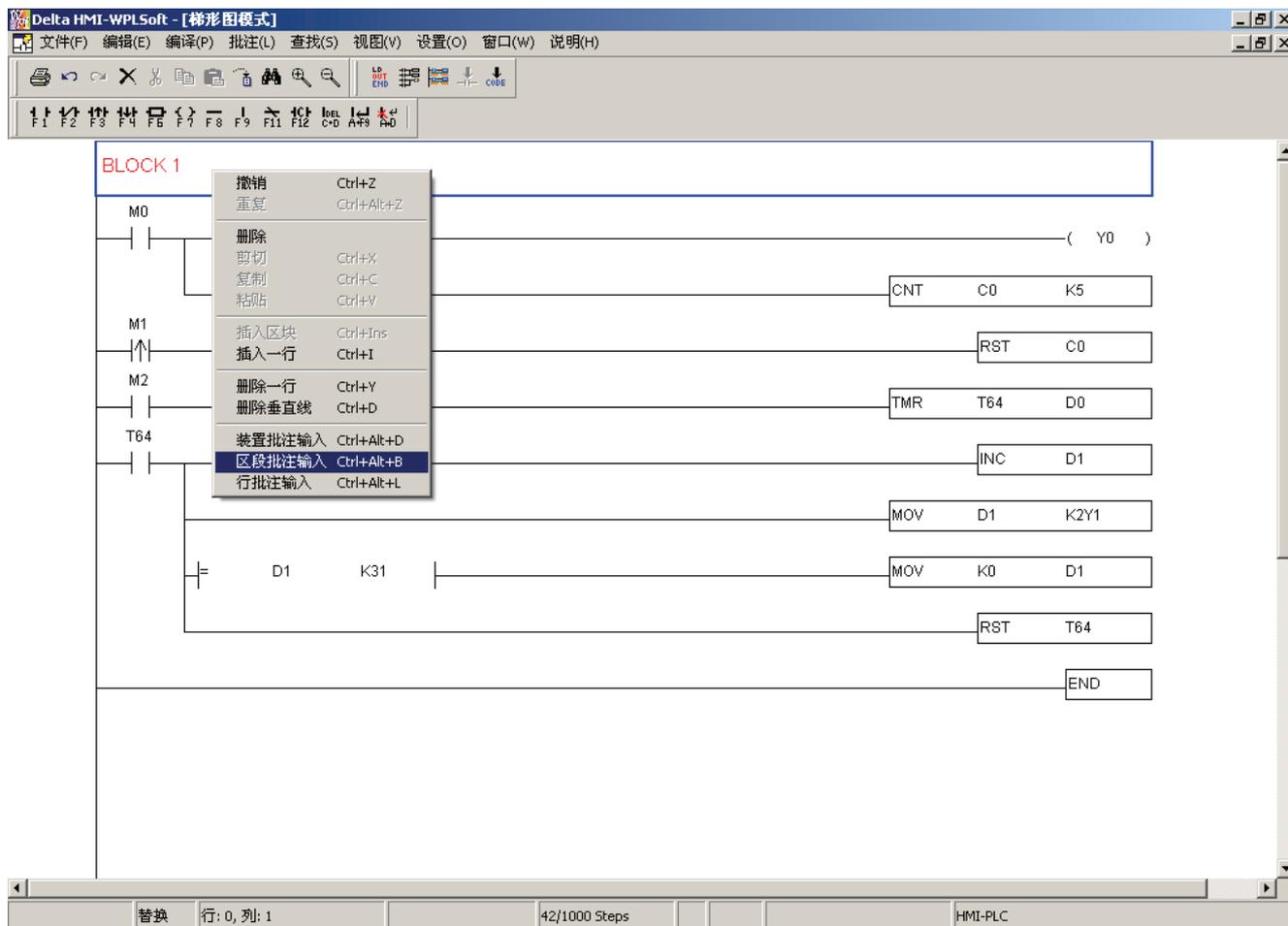
编辑方块移至欲输入输出批注的行上，从功能工具栏上选取批注（L）点选行批注（或键入复合键〔Ctrl〕+〔Alt〕+〔L〕），→出现编辑行批注对话框窗口→于对应的批注栏位键入批注内容，当输入完成后直接关窗口即可。



3.8 区段批注

◆方法一：

编辑方块移至欲输入区段批注的空白行上（可利用〔Ctrl〕+〔I〕插入新的一行），按下鼠标右键出现一快捷操作框，点选区段批注输入便会出现一对话框窗口，如下图，于输入区段批注栏键入批注（最多可输入 60 个字元），最后按下确定钮，编辑完成。



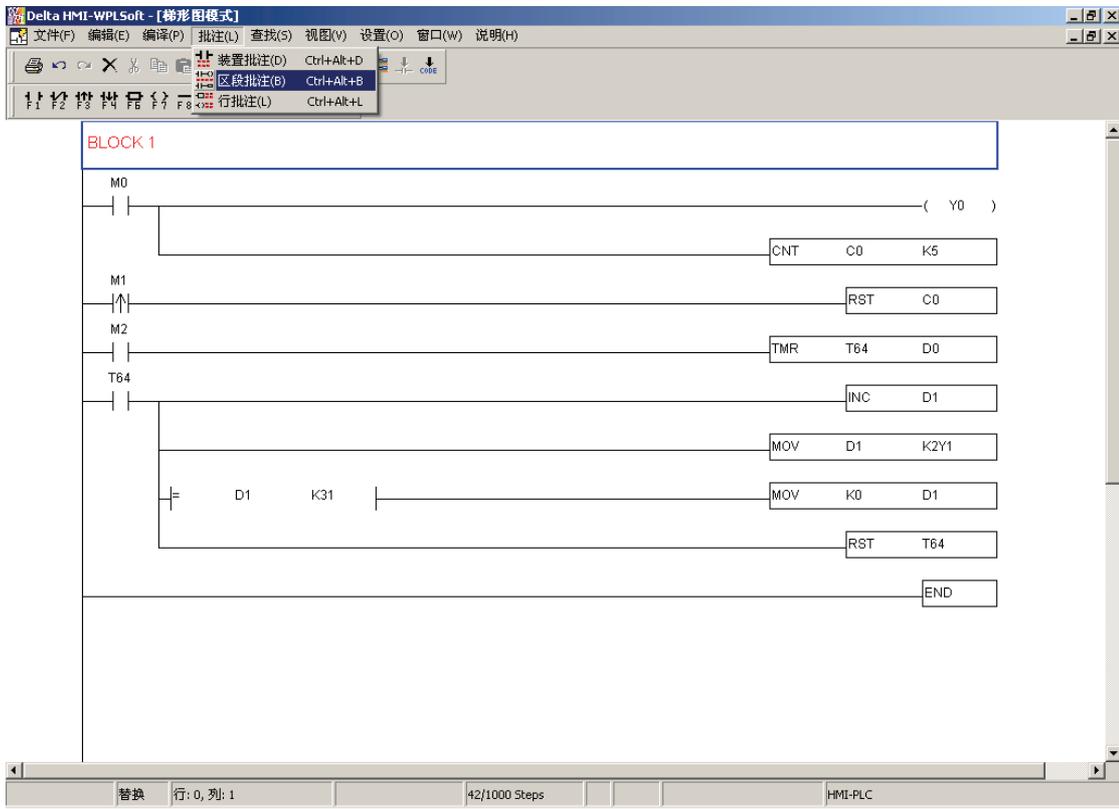
输入批注

BLOCK 1|

确定 取消

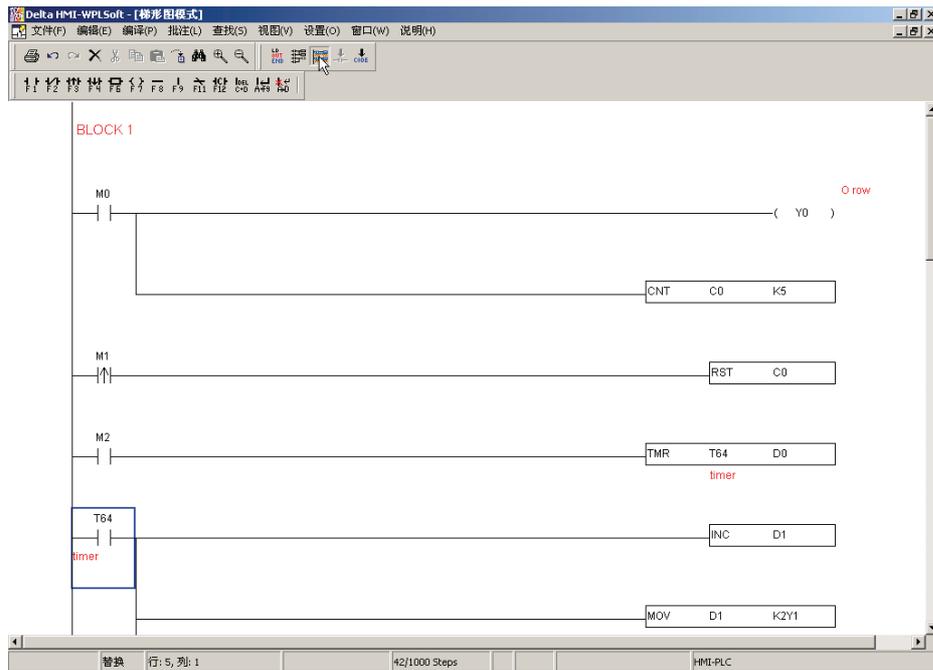
◆方法二：

从功能工具栏上选取『批注 (L)』后点选编辑区块批注或键入复合键 [Ctrl] + [Alt] + [B] 便会会出现区段批注输入行，便可编辑区段批注。



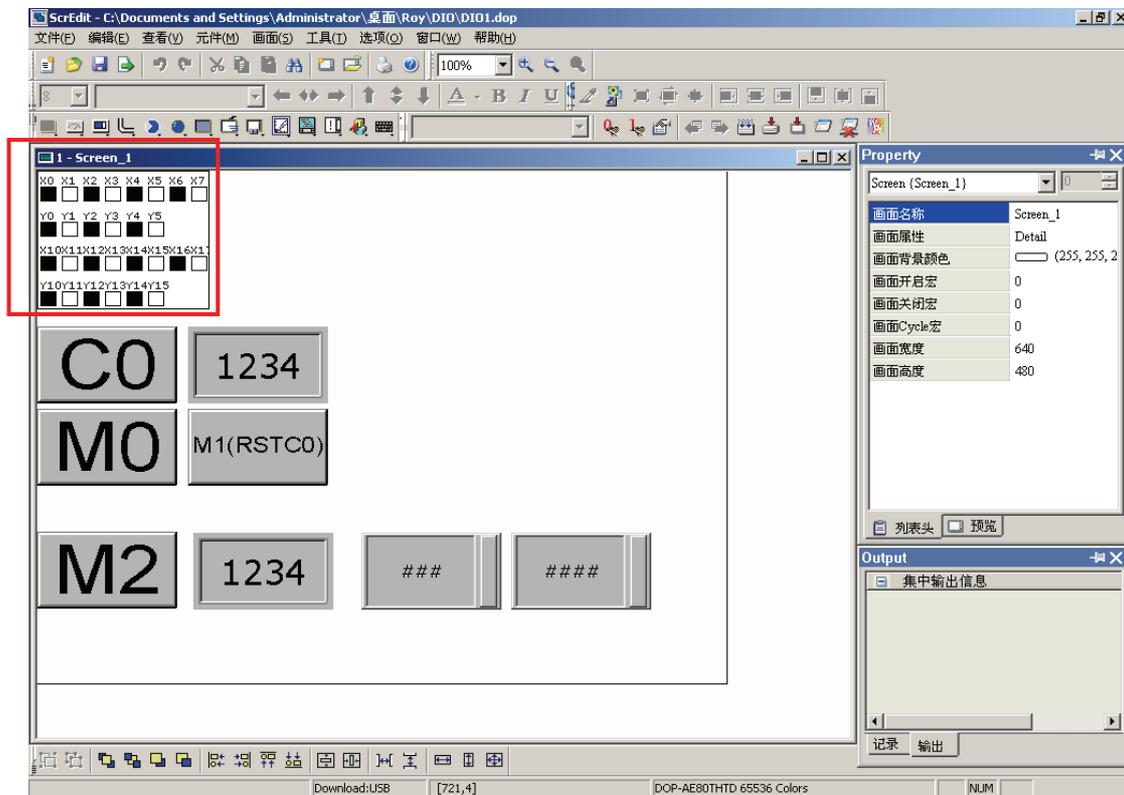
■ 显示批注关闭或打开

从功能工具栏选取『视图(V)』后点选显示批注或点选  图示。在显示批注被打开后梯形图会拉长高度以显示批注内容。

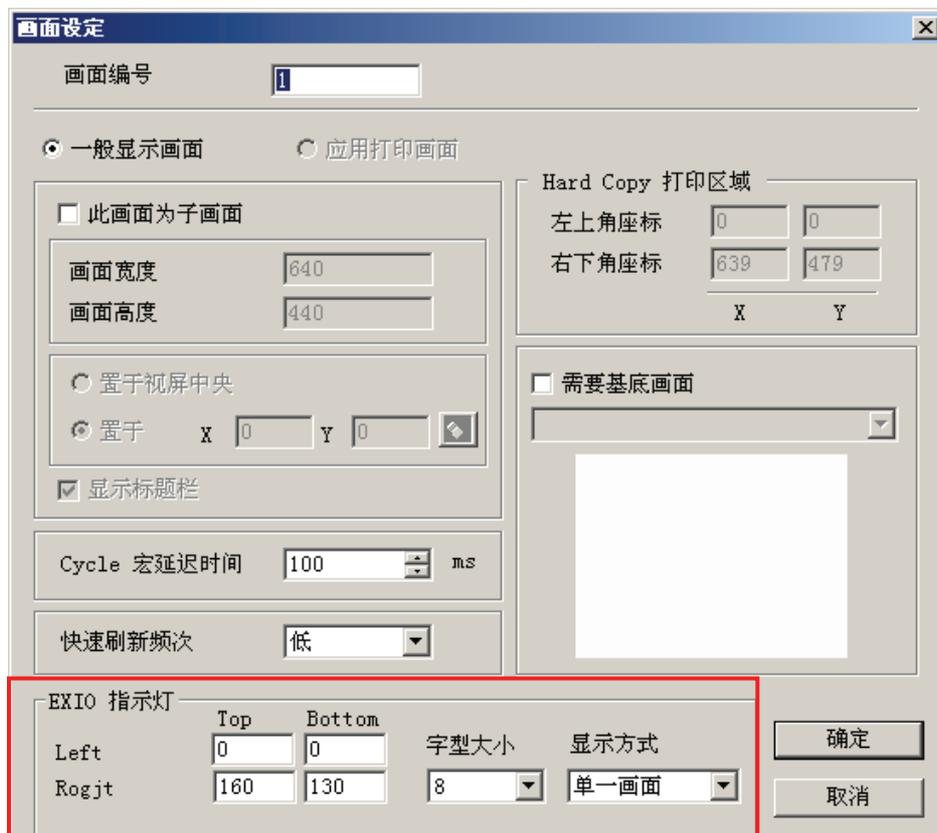


第四章 IO 指示灯

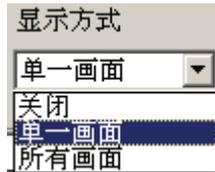
在人机画面上，可以透过 IO 指示灯来监控显示目前 EXIO 模块的输入动作状态，如下图：



进入画面属性的设置页面就可以设置 IO 指示灯各种属性。



1. 显示位置：设置左上角和右下角的座标来决定 I/O 指示灯的显示位置。
2. 字体大小：决定指示灯里面字的大小（可选择 8、10、12、14、16、18、20、24、28、32、40、48、64）。
3. 显示方式：有三种显示方式，「关闭」、「单一画面」、「所有画面」。指示灯的图形显示，会随著画面属性所设置显示灯的数据而变动。
选择「关闭」时，就不会显示指示灯。
选择「单一画面」时，就会在单一设置的画面上显示指示灯。
选择「所有画面」时，则所有的画面都会显示指示灯。



第五章 内部存储器地址

当打开梯形图编译后，人机所有的组件就可以选择使用 EXIO 的存储器地址，使用方法跟内部存储器\$0~\$65535 一样。其中有些地址是断电保持，详细地址和范围请参考附录 A。

当启动 EXIO 编译功能后，内部存储器(Internal Memory)选项中便会出现 EXIO 的地址，有一些地址含有特别的定义，会在特定指令内被设置或是被参考，详细的使用方法请参考附录说明。



(此页有意留为空白)

附录 A EXIO 装置一览表

类别	装置	项 目	范 围	功 能		
继电器 位 型 态	X	输入继电器	X0 ~ X7, 8 点, 8 进制编码	DOP-EXIO14RAE	对应至外部的输入点	
			X0 ~ X17, 16 点, 8 进制编码	DOP-EXIO28RAE	对应至外部的输入点	
	Y	输出继电器	Y0 ~ Y7, 6 点, 8 进制编码	DOP-EXIO14RAE	内部输出点	
			Y0 ~ Y5, Y10 ~ Y15, 12 点, 8 进制编码	DOP-EXIO28RAE	内部输出点	
	M	辅助继电器	一般用	M0 ~ M511, M768 ~ M999, 744 点; M1000 ~ M1279, 280 点 ^{*2}	合计 1,280 点	接点可于程序内做 On/Off 切换
			停电保持用 ^{*1}	M512 ~ M767, 256 点		
	T	定时器	100ms 定时器	T0 ~ T63, 64 点	合计 128 点	TMR 指令所指定的定时器, 若计时到达则此同编号 T 的接点将会 On
			10ms 定时器	T64 ~ T126, 63 点		
			1ms 定时器	T127, 1 点		
	C	计数器	16 位上数一般用	C0 ~ C111, 112 点	合计 128 点	CNT(DCNT) 指令所指定的计数器, 若计数到达则此同编号 C 的接点将会 On
16 位上数停电保持用 ^{*1}			C112 ~ C127, 16 点			
32 位上下数计数器停电保持用 ^{*1}			C235, C236, C237, C238, C241, C242, C244, C246, C247, C249, C251, C252, C254, 13 点	合计 13 点		
S	步进点	停电保持用 ^{*1}	S0 ~ S127, 128 点	合计 128 点	步进梯形图使用装置	
寄存器 字 数 据	T	定时器当前值	T0 ~ T127, 128 点		计时到达时, 接点导通	
	C	计数器当前值	C0 ~ C127, 16 位计数器 128 点 C235, C236, C237, C238, C241, C242, C244, C246, C247, C249, C251, C252, C254, 32 位计数器 13 点		计数到达时, 该计数器接点导通	
	D	数据寄存器	一般用	D0 ~ D407, 408 点	合计 600 点	作为数据保存的存储器区域, E、F 可做为间接指定的特殊用途
			停电保持用 ^{*1}	D408 ~ D599, 192 点		
		间接指定用	E、F, 2 点	合计 2 点		
N	主控回路用		N0 ~ N7, 8 点		主控回路控制点	
P	CJ, CALL 指令用		P0 ~ P63, 64 点		CJ, CALL 的位置指针	
常量	K	10 进制	K-32,768 ~ K32,767 (16 位运算) K-2,147,483,648 ~ K2,147,483,647 (32 位运算)			
	H	16 进制	H0000 ~ HFFFF (16 位运算) H00000000 ~ HFFFFFFFF (32 位运算)			

*1 停电保持用区域为固定区域, 不可变更。

*2 M1000、M1001、M1002、M1003、M1020、M1021、M1022、M1067、M10068 和 M1161 为特 M。

特殊辅助继电器

特 M	功能说明	Power Off ↓ Power On	STOP ↓ RUN	RUN ↓ STOP	属性	停电保持	出厂值	支持型号
M1000	M1000 为 RUN 中常时 On 接点, 即运行监视常开接点 (A 接点), PLC 于 RUN 的状态下, M1000 保持为 On	Off	On	Off	R	No	Off	DOP-EXIO14RAE DOP-EXIO28RAE
M1001	M1001 为 RUN 中常时 Off 接点, 即运行监视常闭接点 (B 接点), PLC 于 RUN 的状态下 M1001 保持为 Off	On	Off	On	R	No	On	
M1002	M1002 开始 RUN 的第一次扫描 On, 之后保持为 Off。该脉冲的宽度为一次扫描时间, 当要作各种初始设置工作时使用本接点	Off	On	Off	R	No	Off	
M1003	M1003 开始 RUN 的第一次扫描 Off, 之后一直 On。即起始负向 (RUN 的瞬间 Off) 脉冲	On	Off	On	R	No	On	
M1020	零标志 (Zero flag)	Off	-	-	R	否	Off	
M1021	借位标志 (Borrow flag)	Off	-	-	R	否	Off	
M1022	进位标志 (Carry flag)	Off	-	-	R	否	Off	
M1067	演算错误	Off	Off	-	R	否	Off	
M1068	演算错误锁定	Off	-	-	R	否	Off	
M1161	8 位处理模式 (On 时 8 位模式)	Off	-	-	R/W	否	Off	

附录 B EXIO 指令一览表

指令总表

所支持的指令			所支持的指令		
16bit 指令	32bit 指令	指令说明	16bit 指令	32bit 指令	指令说明
LD	×	常开接点	MPP	×	读出堆栈（指针移动）
LDI	×	常关接点	OUT	×	驱动线圈
AND	×	串联常开接点	SET	×	动作保持（ON）
ANI	×	串联常关接点	RST	×	接点或寄存器清除
OR	×	并联常开接点	TMR	×	16 位定时器
ORI	×	并联常关接点	CNT	DCNT	16 / 32 位计数器
ANB	×	串联回路方块	MC	×	公共串联接点的连结
ORB	×	并联回路方块	MCR	×	公共串联接点的解除
MPS	×	存入堆栈	LDP	×	上升沿检出动作开始
MRD	×	读出堆栈（指针不动）	LDF	×	下降沿检出动作开始
ANDP	×	上升沿检出串联连接	STL	×	程序跳至副母线(步进梯形开始)
ANDF	×	下降沿检出串联连接	RET	×	程序返回主母线(步进梯形结束)
ORP	×	上升沿检出并联连接	CJ	×	条件转移
ORF	×	下降沿检出并联连接	CALL	×	调用子程序
PLS	×	上升沿检出	SRET	×	子程序结束
PLF	×	下降沿检出	FEND	×	主程序结束
END	×	程序结束	FOR	×	循环范围开始
NOP	×	无动作	NEXT	×	循环范围结束
INV	×	运算结果反相	CMP	DCMP	比较设置输出
P	×	指针	ZCP	DZCP	区间比较
MOV	DMOV	数据传送	DIV	DDIV	BIN 除法
CML	DCML	反转传送	INC	DINC	BIN 加一
BMOV	×	全部传送	DEC	DDEC	BIN 减一
FMOV	DFMOV	多点传送	WAND	DAND	逻辑和(AND)运算
XCH	DXCH	数据的交换	WOR	DOR	逻辑或(OR)运算
BCD	DBCD	BIN→BCD 变换	WXOR	DXOR	逻辑异或(XOR)运算
BIN	DBIN	BCD→BIN 变换	NEG	DNEG	求补码
ADD	DADD	BIN 加法	ROR	DROR	右循环移位
SUB	DSUB	BIN 减法	ROL	DROL	左循环移位
MUL	DMUL	BIN 乘法	RCR	DRCR	附进位标志右循环
RCL	DRCL	附进位标志左循环	HEX	×	ASCII 转为 HEX
SFTR	×	位右移	ABS	DABS	绝对值

所支持的指令			所支持的指令		
16bit 指令	32bit 指令	指令说明	16bit 指令	32bit 指令	指令说明
SFTL	×	位左移	SWAP	DSWAP	上下字节互换
ZRST	×	批次复位	LD=	DLD=	接点型态比较等于
SUM	DSUM	ON 位数量	LD>	DLD>	接点型态比较大于
BON	DBON	ON 位判定	LD<	DLD<	接点型态比较小于
MEAN	DMEAN	平均值	LD<>	DLD<>	接点型态比较不等于
REF	×	I/O 状态即时刷新	LD<=	DLD<=	接点型态比较小于等于
ALT	×	ON/OFF 交替输出	LD>=	DLD>=	接点型态比较大等于
ASCI	×	HEX 转为 ASCII	AND=	DAND=	串联接点型态比较等于
AND>	DAND>	串联接点型态比较大	OR>	DOR>	并联接点型态比较大
AND<	DAND<	串联接点型态比较小	OR<	DOR<	并联接点型态比较小
AND<>	DAND<>	串联接点型态比较不等于	OR<>	DOR<>	并联接点型态比较不等于
AND<=	DAND<=	串联接点型态比较小于等于	OR<=	DOR<=	并联接点型态比较小于等于
AND>=	DAND>=	串联接点型态比较大等于	OR>=	DOR>=	并联接点型态比较大等于
OR=	DOR=	并联接点型态比较等于			

附录 C EXIO 基本指令说明

指令	功 能						
LD	载入 A 接点 (ON 表示通路)						
操作数	X0~X17	Y0~Y17	M0~M1279	S0~S127	T0~T127	C0~C254	D0~D599
	✓	✓	✓	✓	✓	✓	—

指令说明

- ◆ LD 指令用于左母线开始的 A 接点或一个接点回路块开始的 A 接点，它的作用是把当前内容保存，同时把取来的接点状态存入累加器内。

程序范例

梯形图：



指令码：

说明：

LD	X0	载入 X0 的 A 接点
AND	X1	串联 X1 的 A 接点
OUT	Y1	驱动 Y1 线圈

指令	功 能						
LDI	载入 B 接点 (OFF 表示通路)						
操作数	X0~X17	Y0~Y17	M0~M1279	S0~S127	T0~T127	C0~C254	D0~D599
	✓	✓	✓	✓	✓	✓	—

指令说明

- ◆ LDI 指令用于左母线开始的 B 接点或一个接点回路块开始的 B 接点，它的作用是把当前内容保存，同时把取来的接点状态存入累加器内。

程序范例

梯形图：



指令码：

说明：

LDI	X0	载入 X0 的 B 接点
AND	X1	串联 X1 的 A 接点
OUT	Y1	驱动 Y1 线圈

指令	功能						
AND	串联 A 接点						
操作数	X0~X17	Y0~Y17	M0~M1279	S0~S127	T0~T127	C0~C254	D0~D599
	✓	✓	✓	✓	✓	✓	—

指令说明

- ◆ AND 指令用于 A 接点的串联连接，它的作用是先读取目前所指定串联接点的状态再与接点的前逻辑运算结果作“和”（AND）的运算，并将结果存入累加器内。

程序范例

梯形图：



指令码：

说明：

LDI	X1	载入 X1 的 B 接点
AND	X0	串联 X0 的 A 接点
OUT	Y1	驱动 Y1 线圈

指令	功能						
ANI	串联 B 接点						
操作数	X0~X17	Y0~Y17	M0~M1279	S0~S127	T0~T127	C0~C254	D0~D599
	✓	✓	✓	✓	✓	✓	—

指令说明

- ◆ ANI 指令用于 B 接点的串联连接，它的作用是先读取目前所指定串联接点的状态再与接点的前逻辑运算结果作“和”（AND）的运算，并将结果存入累加器内。

程序范例

梯形图：



指令码：

说明：

LD	X1	载入 X1 的 A 接点
ANI	X0	串联 X0 的 B 接点
OUT	Y1	驱动 Y1 线圈

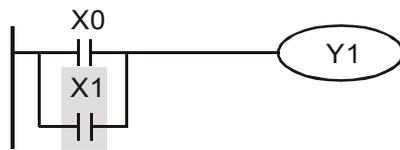
指令	功 能						
OR	并联 A 接点						
操作数	X0~X17	Y0~Y17	M0~M1279	S0~S127	T0~T127	C0~C254	D0~D599
	✓	✓	✓	✓	✓	✓	—

指令说明

- ◆ **OR** 指令用于 A 接点的并联连接，它的作用是先读取目前所指定串联接点的状态再与接点的前逻辑运算结果作“或”（OR）的运算，并将结果存入累加器内。

程序范例

梯形图：



指令码：

说明：

LD	X0	载入 X0 的 A 接点
OR	X1	并联 X1 的 A 接点
OUT	Y1	驱动 Y1 线圈

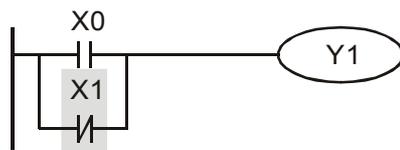
指令	功 能						
ORI	并联 B 接点						
操作数	X0~X17	Y0~Y17	M0~M1279	S0~S127	T0~T127	C0~C254	D0~D599
	✓	✓	✓	✓	✓	✓	—

指令说明

- ◆ **ORI** 指令用于 B 接点的并联连接，它的作用是先读取目前所指定串联接点的状态再与接点的前逻辑运算结果作“或”（OR）的运算，并将结果存入累加器内。

程序范例

梯形图：



指令码：

说明：

LD	X0	载入 X0 的 A 接点
ORI	X1	并联 X1 的 B 接点
OUT	Y1	驱动 Y1 线圈

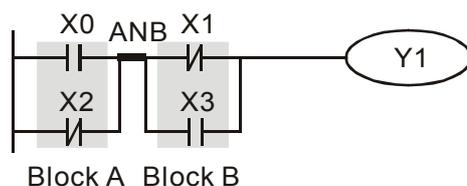
指令	功	能
ANB	串联回路方块	
操作数	无	

指令说明

◆ ANB 是将前一保存的逻辑结果与目前累加器的内容作“和”（AND）的运算。

程序范例

梯形图：



指令码：

LD	X0	载入 X0 的 A 接点
ORI	X2	并联 X2 的 B 接点
LDI	X1	载入 X1 的 B 接点
OR	X3	并联 X3 的 A 接点
ANB		串联回路方块
OUT	Y1	驱动 Y1 线圈

说明：

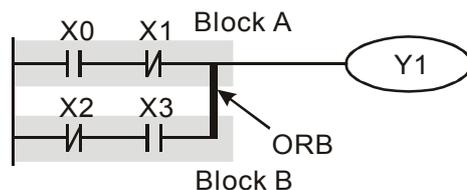
指令	功	能
ORB	并联回路方块	
操作数	无	

指令说明

◆ ORB 是将前一保存的逻辑结果与目前累加器的内容作“或”（OR）的运算。

程序范例

梯形图：



指令码：

LD	X0	载入 X0 的 A 接点
ANI	X1	串联 X1 的 B 接点
LDI	X2	载入 X2 的 B 接点
AND	X3	串联 X3 的 A 接点
ORB		并联回路方块
OUT	Y1	驱动 Y1 线圈

说明：

指令	功	能
MPS	存入堆栈	
操作数	无	

指令说明

- ◆ 将目前累加器的内容存入堆栈。（堆栈指针加一）

指令	功	能
MRD	读出堆栈（指针不动）	
操作数	无	

指令说明

- ◆ 读取堆栈内容存入累加器。（堆栈指针不动）

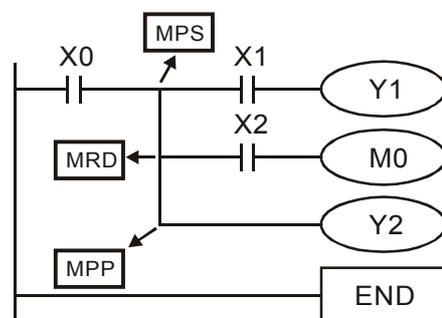
指令	功	能
MPP	读出堆栈	
操作数	无	

指令说明

- ◆ 自堆栈取回前一保存的逻辑运算结果，存入累加器。（堆栈指针减一）

程序范例

梯形图：



指令码：

说明：

LD	X0	载入 X0 的 A 接点
MPS		存入堆栈
AND	X1	串联 X1 的 A 接点
OUT	Y1	驱动 Y1 线圈
MRD		读出堆栈（指针不动）
AND	X2	串联 X2 的 A 接点
OUT	M0	驱动 M0 线圈
MPP		读出堆栈
OUT	Y2	驱动 Y2 线圈
END		程序结束

指令	功 能						
OUT	驱动线圈						
操作数	X0~X17	Y0~Y17	M0~M1279	S0~S127	T0~T127	C0~C254	D0~D599
	—	✓	✓	✓	—	—	—

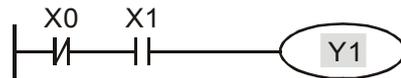
指令说明

- ◆ 将 OUT 指令的前的逻辑运算结果输出至指定的组件。
- ◆ 线圈接点动作：

运算结果	OUT 指令		
	线圈	接点	
		A 接点 (常开)	B 接点 (常闭)
FALSE	Off	不导通	导通
TRUE	On	导通	不导通

程序范例

梯形图：



指令码：

说明：

LDI	X0	载入 X0 的 B 接点
AND	X1	串联 X1 的 A 接点
OUT	Y1	驱动 Y1 线圈

指令	功 能						
SET	动作保持 (ON)						
操作数	X0~X17	Y0~Y17	M0~M1279	S0~S127	T0~T127	C0~C254	D0~D599
	—	✓	✓	✓	—	—	—

指令说明

- ◆ 当 SET 指令被驱动，其指定的组件被设置为 On，且被设置的组件会维持 On，不管 SET 指令是否仍被驱动。可利用 RST 指令将该组件设为 Off。

程序范例

梯形图：



指令码：

说明：

LD	X0	载入 X0 的 A 接点
ANI	Y0	串联 Y0 的 B 接点
SET	Y1	Y1 动作保持 (ON)

指令	功							能
RST	接点或寄存器清除							
操作数	X0~X17	Y0~Y17	M0~M1279	S0~S127	T0~T127	C0~C254	D0~D599	E、F
	—	✓	✓	✓	✓	✓	✓	✓

指令说明

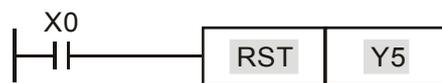
- ◆ 当 RST 指令被驱动，其指定的组件的动作如下：

元件	状态
S, Y, M	线圈和接点都会被设置为 Off。
T, C	目前计时或计数值会被设为 0，且线圈和接点都会被设置为 Off。
D, E, F	内容值会被设为 0。

- ◆ 若 RST 指令没有被执行，其指定组件的状态保持不变。

程序范例

梯形图：



指令码：

LD X0 载入 X0 的 A 接点
RST Y5 Y5 接点清除

说明：

指令	功		能
TMR	16 位定时器		
操作数	T-K	T0~T127, K0~K32,767	
	T-D	T0~T127, D0~D599	

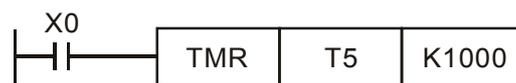
指令说明

- ◆ 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，当到达所指定的定时值（计时值 \geq 设置值），其接点动作如下：

NO (Normally Open) 接点	开路
NC (Normally Closed) 接点	闭合

程序范例

梯形图：



指令码：

LD X0 载入 X0 的 A 接点
TMR T5 K1000 T5 定时器
 设置值为 K1000

说明：

指令	功		能
CNT	16 位计数器		
操作数	C-K	C0~C127, K0~K32,767	
	C-D	C0~C127, D0~D599	

指令说明

- ◆ 当 CNT 指令由 Off→On 执行，表示所指定的计数器线圈由失电→受电，则该计数器计数值加 1，当计数到达所指定的定数值（计数值 = 设置值），其接点动作如下：

NO (Normally Open) 接点	开路
NC (Normally Closed) 接点	闭合

- ◆ 当计数到达的后，若再有计数脉冲输入，其接点和计数值均保持不变，若要重新计数或作清除的动作，请利用 RST 指令。

程序范例

梯形图：



指令码：

说明：

LD	X0	载入 X0 的 A 接点
CNT	C20 K100	C20 计数器设置值为 K100

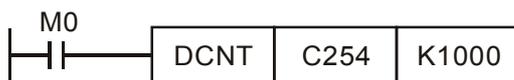
指令	功		能
DCNT	32 位计数器		
操作数	C-K	C235~C254	
	C-D	C235~C254, D0~D598	

指令说明

- ◆ DCNT 为计数器 C235 至 C254 为 32 位计数器启动指令，使用方法与 16 位计数器 C0~C127 指令相同。
- ◆ 当 DCNT 指令 OFF 时，该计数器停止计数，但原有计数值不会被清除，可使用指令 RST C2XX 清除计数值和其接点。

程序范例

梯形图：



指令码：

说明：

LD	M0	载入 M0 的 A 接点
DCNT	C254 K1000	C254 计数器设置值为 K1000

指令	功能
MC / MCR	公共串联接点的连结 / 解除
操作数	N0~N7

指令说明

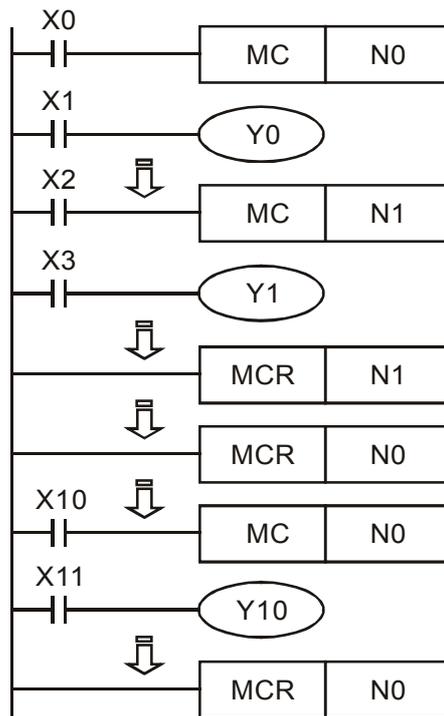
- ◆ MC 为主控起始指令，当 MC 指令执行时，位于 MC 与 MCR 指令的间的指令照常执行。当 MC 指令 OFF 时，位于 MC 与 MCR 指令的间的指令动作如下所示：

定时器	计时值归零，线圈失电，接点不动作
计数器	线圈失电，计数值和接点保持目前状态
OUT 指令驱动的线圈	全部不受电
SET, RST 指令驱动的组件	保持目前状态
应用指令	全部不动作

- ◆ MCR 为主控结束指令，置于主控程序最后，在 MCR 指令的前不可有接点指令。
- ◆ MC-MCR 主控程序指令支持嵌套结构，最多可 8 层，使用时依 N0~N7 的顺序，请参考如下程序所示：

程序范例

梯形图：



指令码：

说明：

LD	X0	载入 X0 的 A 接点
MC	N0	N0 公共串联接点的连结
LD	X1	载入 X1 的 A 接点
OUT	Y0	驱动 Y0 线圈
:		
LD	X2	载入 X2 的 A 接点
MC	N1	N1 公共串联接点的连结
LD	X3	载入 X3 的 A 接点
OUT	Y1	驱动 Y1 线圈
:		
MCR	N1	N1 公共串联接点的解除
:		
MCR	N0	N0 公共串联接点的解除
:		
LD	X10	载入 X10 的 A 接点
MC	N0	N0 公共串联接点的连结
LD	X11	载入 X11 的 A 接点
OUT	Y10	驱动 Y10 线圈
:		
MCR	N0	N0 公共串联接点的解除

指令	功 能						
LDP	上升沿检出动作开始 (OFF → ON)						
操作数	X0~X17	Y0~Y17	M0~M1279	S0~S127	T0~T127	C0~C254	D0~D599
	✓	✓	✓	✓	✓	✓	—

指令说明

- ◆ LDP 指令用法上与 LD 相同，但动作不同，它的作用是指当前内容保存，同时把取来的接点上升沿检出状态存入累加器内。

程序范例

梯形图：



指令码：

说明：

LDP	X0	X0 上升沿检出动作开始
AND	X1	串联 X1 的 A 接点
OUT	Y1	驱动 Y1 线圈

指令	功 能						
LDF	下降沿检出动作开始 (ON → OFF)						
操作数	X0~X17	Y0~Y17	M0~M1279	S0~S127	T0~T127	C0~C254	D0~D599
	✓	✓	✓	✓	✓	✓	—

指令说明

- ◆ LDF 指令用法上与 LD 相同，但动作不同，它的作用是指当前内容保存，同时把取来的接点下降沿检出状态存入累加器内。

程序范例

梯形图：



指令码：

说明：

LDF	X0	X0 下降沿检出动作开始
AND	X1	串联 X1 的 A 接点
OUT	Y1	驱动 Y1 线圈

指令	功 能						
ANDP	上升沿检出串联连接						
操作数	X0~X17	Y0~Y17	M0~M1279	S0~S127	T0~T127	C0~C254	D0~D599
	✓	✓	✓	✓	✓	✓	—

指令说明

- ◆ **ANDP** 指令用于接点上升沿检出的串联连接。

程序范例

梯形图：



指令码：

说明：

LD	X0	载入 X0 的 A 接点
ANDP	X1	X1 上升沿检出串联连接
OUT	Y1	驱动 Y1 线圈

指令	功 能						
ANDF	下降沿检出串联连接						
操作数	X0~X17	Y0~Y17	M0~M1279	S0~S127	T0~T127	C0~C254	D0~D599
	✓	✓	✓	✓	✓	✓	—

指令说明

- ◆ **ANDF** 指令用于接点下降沿检出的串联连接。

程序范例

梯形图：



指令码：

说明：

LD	X0	载入 X0 的 A 接点
ANDF	X1	X1 下降沿检出串联连接
OUT	Y1	驱动 Y1 线圈

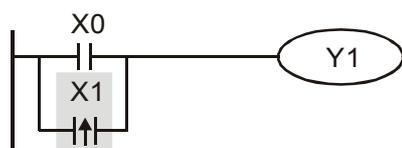
指令	功 能						
ORP	上升沿检出并联连接						
操作数	X0~X17	Y0~Y17	M0~M1279	S0~S127	T0~T127	C0~C254	D0~D599
	✓	✓	✓	✓	✓	✓	—

指令说明

◆ ORP 指令用于接点上升沿检出的并联连接。

程序范例

梯形图：



指令码：

说明：

LD	X0	载入 X0 的 A 接点
ORP	X1	X1 上升沿检出并联连接
OUT	Y1	驱动 Y1 线圈

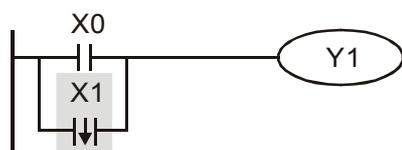
指令	功 能						
ORF	下降沿检出并联连接						
操作数	X0~X17	Y0~Y17	M0~M1279	S0~S127	T0~T127	C0~C254	D0~D599
	✓	✓	✓	✓	✓	✓	—

指令说明

◆ ORF 指令用于接点下降沿检出的并联连接。

程序范例

梯形图：



指令码：

说明：

LD	X0	载入 X0 的 A 接点
ORF	X1	X1 下降沿检出并联连接
OUT	Y1	驱动 Y1 线圈

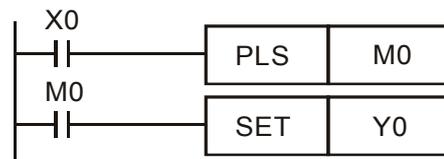
指令	功 能						
PLS	上升沿检出						
操作数	X0~X17	Y0~Y17	M0~M1279	S0~S127	T0~T127	C0~C254	D0~D599
	—	✓	✓	—	—	—	—

指令说明

- ◆ 上升沿检出指令。当 X0=OFF→ON（上升沿触发）时 PLS 指令被执行，M0 送出一脉冲，脉冲长度为一次扫描时间。

程序范例

梯形图：



指令码：

说明：

LD	X0	载入 X0 的 A 接点
PLS	M0	M0 上升沿检出
LD	M0	载入 M0 的 A 接点
SET	Y0	Y0 动作保持(ON)

时序图：



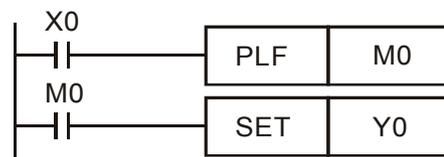
指令	功 能						
PLF	下降沿检出						
操作数	X0~X17	Y0~Y17	M0~M1279	S0~S127	T0~T127	C0~C254	D0~D599
	—	✓	✓	—	—	—	—

指令说明

- ◆ 下降沿检出指令。当 X0=ON→OFF（下降沿触发）时 PLF 指令被执行，M0 送出一脉冲，脉冲长度为一次扫描时间。

程序范例

梯形图：



指令码：

说明：

LD	X0	载入 X0 的 A 接点
PLF	M0	M0 下降沿检出
LD	M0	载入 M0 的 A 接点
SET	Y0	Y0 动作保持(ON)

时序图：



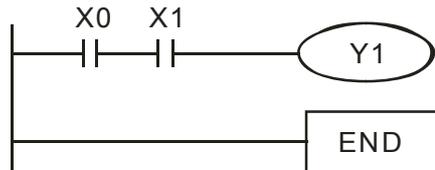
指令	功	能
END	程序结束	
操作数	无	

指令说明

- ◆ 在梯形图程序或指令程序最后必须加入 END 指令。EXIO 由地址 0 扫描到 END 指令，执行的后，返回到地址 0 重新作扫描执行。

程序范例

时序图：



指令码：

说明：

LD	X0	载入 X0 的 B 接点
AND	X1	串联 X1 的 A 接点
OUT	Y1	驱动 Y1 线圈
END		程序结束

指令	功	能
NOP	无动作	
操作数	无	

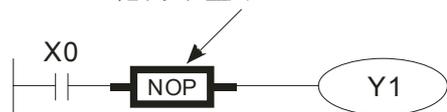
指令说明

- ◆ 指令 NOP 在程序不做任何运算，因此执行后仍会保持原逻辑运算结果，使用时机如下：
 1. 预先保留部分程序记忆空间，作为 EXIO 程序除错时，可写入侦错程序。
 2. 想要删除某一指令，而又不想改变程序长度，则可以 NOP 指令替换。
- ◆ 想暂时性的删除某一指令，先以 NOP 指令替代。

程序范例

梯形图：

梯形图显示时,会将指令NOP
化简不显示



指令码：

说明：

LD	X0	载入 X0 的 B 接点
NOP		无动作
OUT	Y1	驱动 Y1 线圈

指令	功	能
INV	运算结果反相	
操作数	无	

指令说明

- ◆ 将 INV 指令的前的逻辑运算结果反相存入累加器内。

程序范例

梯形图:



指令码:

说明:

LD	X0	载入 X0 的 A 接点
INV		运算结果反相
OUT	Y1	驱动 Y1 线圈

指令	功	能
P	指针	
操作数	P0~P63	

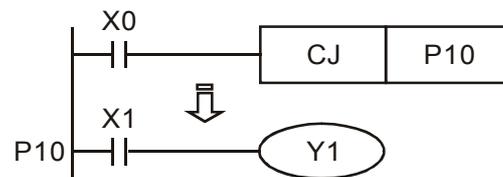
指令说明

- ◆ 指针 P 用于转移指令 CJ 和子程序呼叫指令 CALL 使用，不须从编号 0 开始，但是编号不能重复使用，否则会发生不可预期的错误。使用时机如下所示：

1. 使用于指令 CJ，指示程序执行转移的目的地址，并在目的程序的开头输入同编号的指针 P。如下所示：
2. 使用于指令 CALL，指示子程序的目的地址，并在子程序的开头输入同编号的指针 P。如下所示：

程序范例

梯形图:



指令码:

说明:

LD	X0	载入 X0 的 A 接点
CJ	P10	转移指令 CJ 到 P10
:		
P10		指针 P10
LD	X1	载入 X1 的 A 接点
OUT	Y1	驱动 Y1 线圈

指令	功能						
STL	程序跳至副母线(步进梯形开始)						
操作数	X0~X17	Y0~Y17	M0~M1279	S0~S127	T0~T127	C0~C254	D0~D599
	—	—	—	✓	—	—	—

指令说明

- ◆ 步进梯形指令 **STL Sn** 构成一个步进点，当 **STL** 指令出现在程序中，代表程序进入以步进流程控制的步进梯形图状态。初始状态必须由 **S0~S9** 开始，步进梯形指令 **RET** 则代表以 **S0~S9** 为起始的步进梯形图结束，母线回归到一般梯形图的命令。步进点 **S** 编号不能重复。

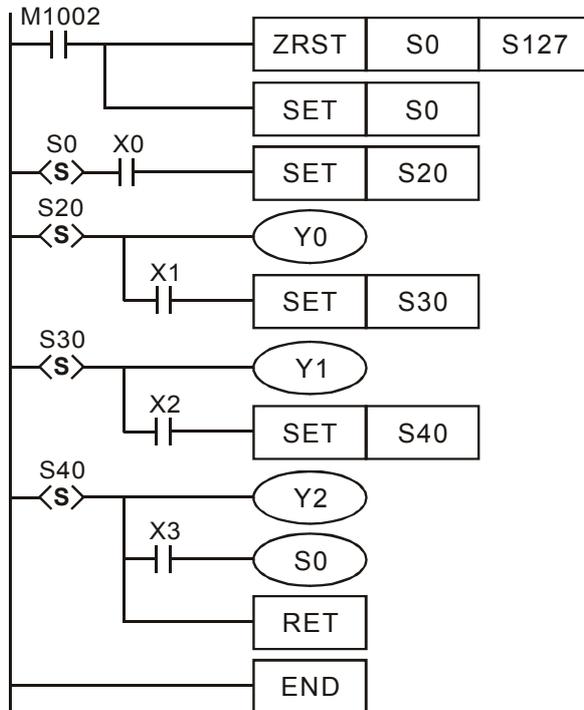
指令	功能						
RET	程序返回主母线(步进梯形结束)						
操作数	无						

指令说明

- ◆ **RET** 指令代表一个步进流程的结束，所以一连串步进点的最后一定要有 **RET** 指令。一个 PLC 程序最多可写入 **S0~S9** 共 10 个步进流程，而每一个步进流程结束就要有 **RET** 指令。

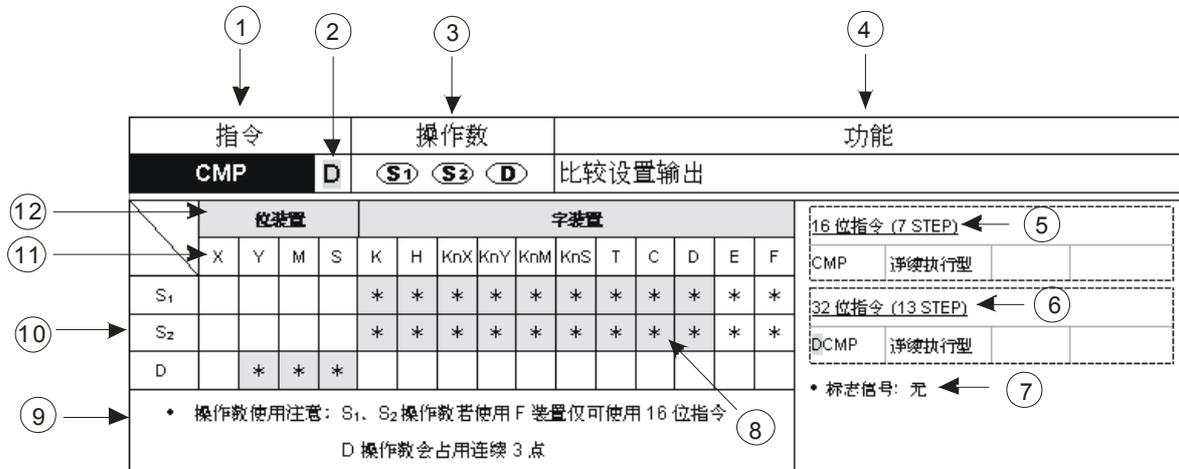
程序范例

梯形图：



附录 D EXIO 应用指令说明

◆ 应用指令的格式说明



- ① 应用指令名称
- ② 若为虚线表示此应用指令无 32 位指令，只具有 16 位指令。若有 32 位指令方框内以 **D** 表示（例：DCMP）
- ③ 应用指令的操作数格式
- ④ 应用指令功能描述
- ⑤ 16 位指令所占的地址数，连续执行型指令名称
- ⑥ 32 位指令所占的地址数，连续执行型指令名称
- ⑦ 与该应用指令有相关的标志信号
- ⑧ 符号 ‘*’ 标示者又含灰底色者，表示该装置可使用间接指定寄存器 E、F 修饰
- ⑨ 操作数使用注意事项
- ⑩ 有符号 ‘*’ 标示者，表示该操作数可使用的装置
- ⑪ 装置名称
- ⑫ 装置型式

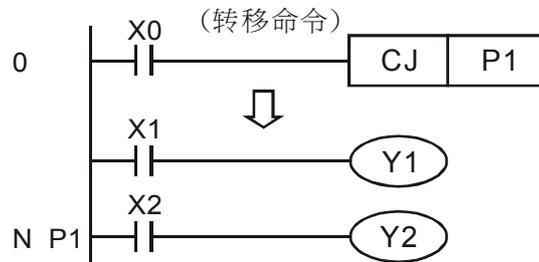
指令		操作数		功能												
CJ		S		条件转移												
	位装置				字装置								16 位指令 (3 STEP)			
	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	CJ
• 操作数使用注意：S 操作数可指定 P P 编号可使用 E、F 修饰																
32 位指令 — — — — • 标志信号：无																

指令说明

- ◆ **S**：条件转移的目的指针。
- ◆ 使用者希望 PLC 程序中的某一部分在不需要时，不去执行以缩短扫描时间，和用于双重输出时，可使用 CJ 指令。
- ◆ 标 P 所指的程序若在 CJ 指令的前，需注意会发生 WDT 超时的错误，PLC 停止运转，请勿如此使用。

程序范例 (一)

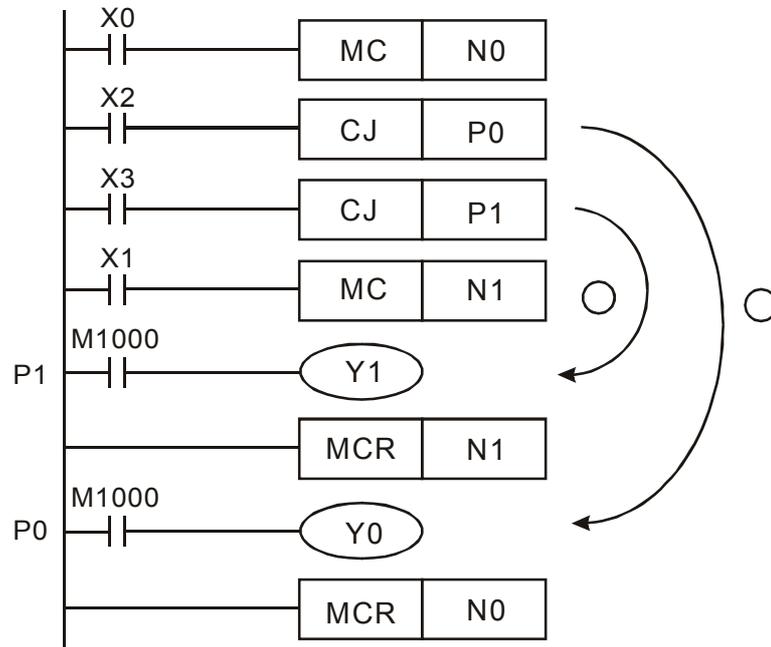
- ◆ J 指令可重复指定同一指针 P，但 CJ 与 CALL 不可指定同一指针 P 会产生错误。
- ◆ 当 X0=On 时，程序自动从地址 0 转移至地址 N (即指定的标签 P1) 继续执行，中间地址跳过不执行。
- ◆ 当 X0=Off 时，程序如同一般程序由地址 0 继续往下执行，此时 CJ 指令不被执行。



程序范例 (二)

- ◆ CJ 指令在 MC、MCR 指令间可使用在下列五种状况：
 1. 在 MC~MCR 外。
 2. 在 MC 外至 MC 内，如下图 P1 以下回路有效。
 3. 同一 N 层 MC 内至 MC 内。
 4. 在 MC 内至 MCR 外。
 5. 从此 MC~MCR 内 跳至另一 MC~MCR 内。

- ◆ CJ 指令在 MC、MCR 指令间使用仅可使用在 MC~MCR 外或 MC~MCR 同一 N 层内，不可从此 MC~MCR 跳至另一 MC~MCR 会产生错误。即上列的状况 1、3 可正确动作，其余会产生错误。

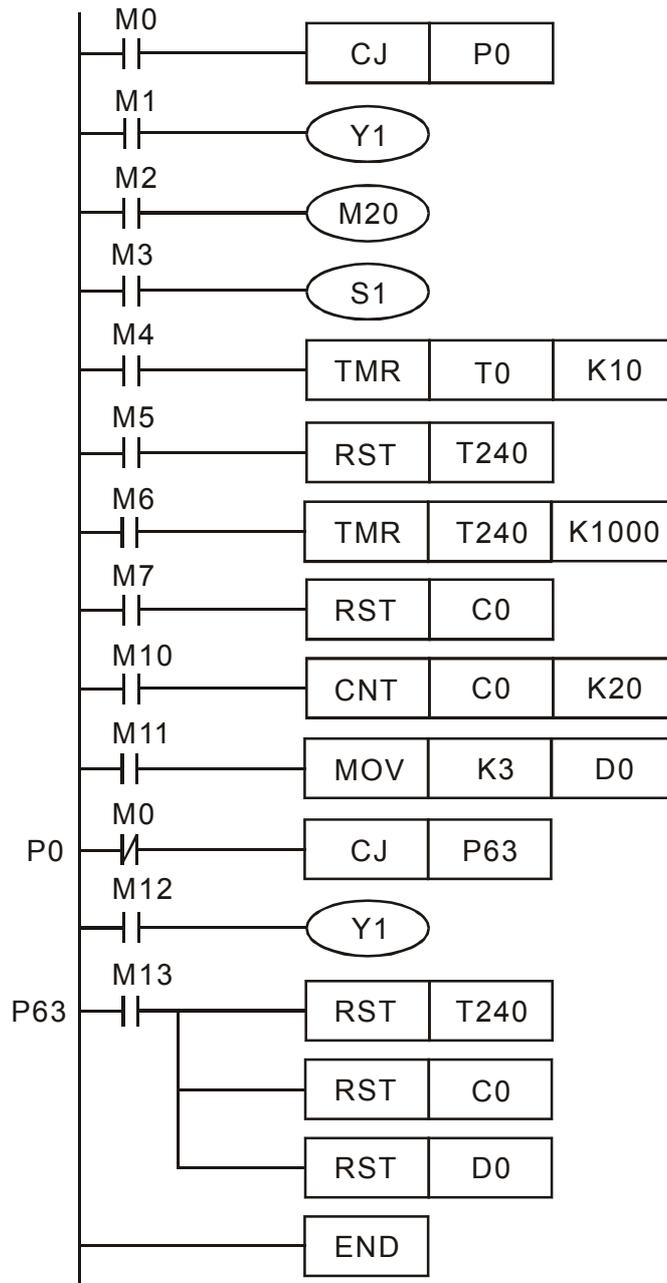


程序范例
(三)

- ◆ 转移执行中各种装置动作情形说明：

- ◆ 此指令的使用方法，类似 C 语言中的 goto 指令，转移时所有的状态都不会被改变。
- ◆ 执行中的定时器会继续计时，但以和计数值都要等到执行计时指令时才会正常动作。
- ◆ 计数器会停止计数（这是因为软件计数的关系）。
- ◆ 被转移过的指令都不执行。

- ◆ Y1 为双重输出，M0=Off 时，由 M1 来控制，M0=On 时，由 M12 来控制。



指令		操作数		功能														
CALL		Ⓢ		调用子程序														
	位装置				字装置											16 位指令 (3 STEP)		
	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	CALL	连续执行型	
<ul style="list-style-type: none"> 操作数使用注意：S 操作数可指定 P P 编号可使用 E、F 修饰 																	32 位指令	
																	— — — —	
																	• 标志信号：无	

指令说明

- ◆ Ⓢ：调用子程序的指针。
- ◆ 指针所指定的子程序请于 FEND 指令后编写。
- ◆ 指针 P 的号码在被 CALL 使用时，不可与 CJ 指令指定相同的号码。
- ◆ 若仅使用 CALL 指令则可不限次数呼叫同一指针号码的子程序。
- ◆ 子程序中再使用 CALL 指令呼叫其他子程序，包括本身最多可五层。(若进入第六层则该子程序不执行)

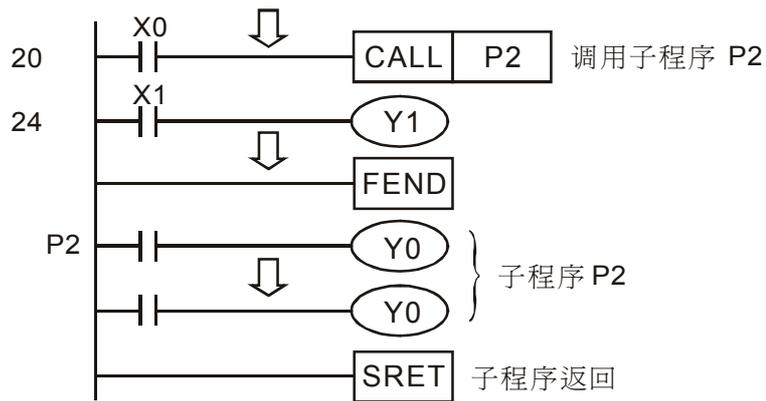
指令		操作数		功能																
SRET		无		子程序结束																
	位装置				字装置												16 位指令 (1 STEP)			
	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	SRET	连续执行型	-	-	
<ul style="list-style-type: none"> 操作数使用注意：无操作数 不须接点驱动的指令 																32 位指令				
																- - - -				
																<ul style="list-style-type: none"> 标志信号：无 				

指令说明

- ◆ 表示子程序结束。
- ◆ 子程序执行结束由 SRET 返回主程序，执行原呼叫该子程序 CALL 指令的下一个指令。

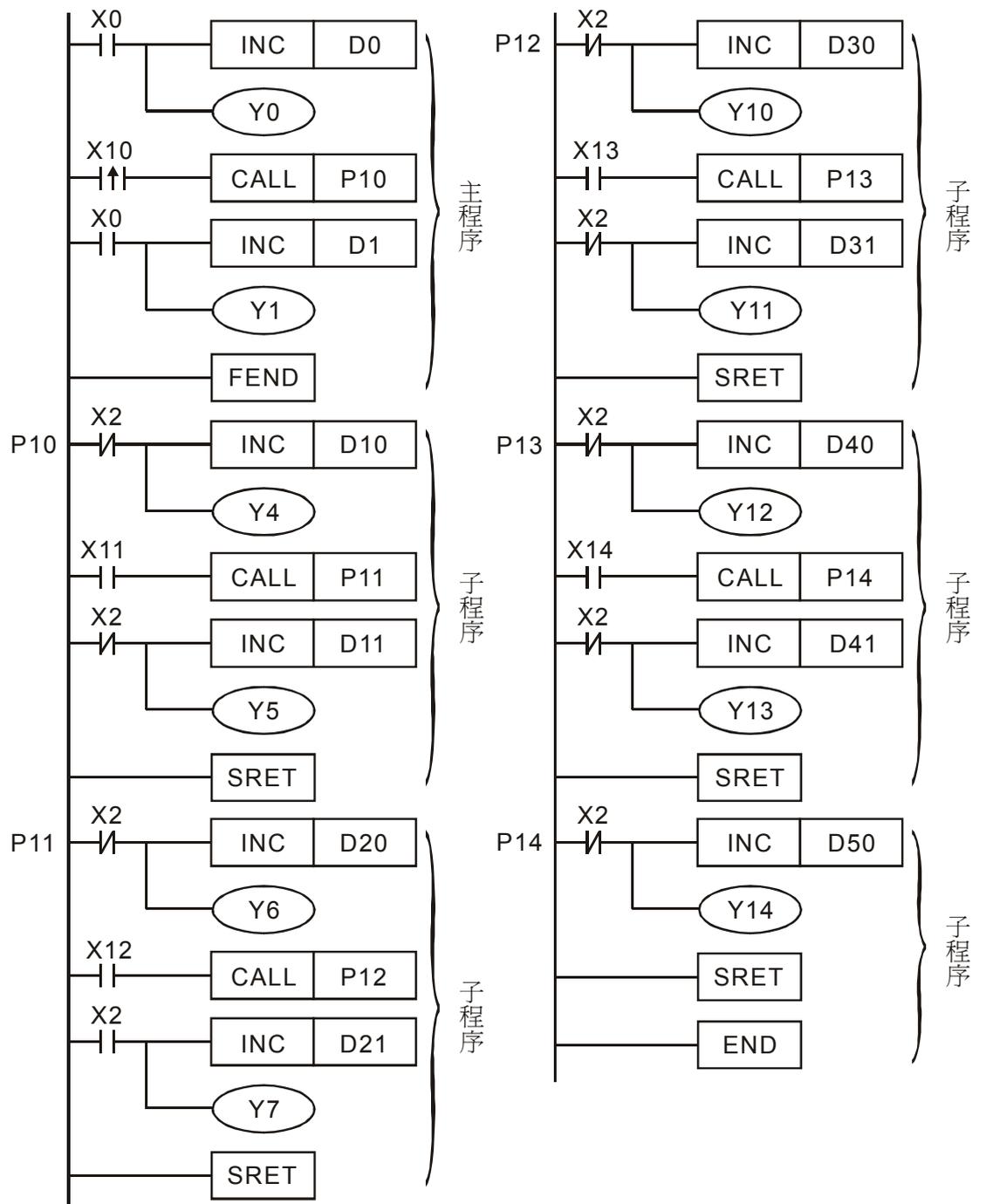
程序范例 (一)

- ◆ 当 X0 为 On 时，则执行 CALL 指令，转移到 P2 执行所指定的子程序，当执行 SRET 指令时，则回到地址 24，继续往下执行。



程序范例 (二)

- ◆ 当 X10 为由 Off 到 On 的 PLUSE 执行 CALL P10 指令，转移到 P10 执行所指定的子程序。
- ◆ 当 X11 为 On 时，则执行 CALL P11，转移到 P11 执行所指定的子程序。
- ◆ 当 X12 为 On 时，则执行 CALL P12，转移到 P12 执行所指定的子程序。
- ◆ 当 X13 为 On 时，则执行 CALL P13，转移到 P13 执行所指定的子程序。
- ◆ 当 X14 为 On 时，则执行 CALL P14，转移到 P14 执行所指定的子程序，当执行到 SRET 指令时，则回到前一个 P** 子程序继续往下执行。
- ◆ 在 P10 子程序中执行到 SRET 指令后回到主程序。

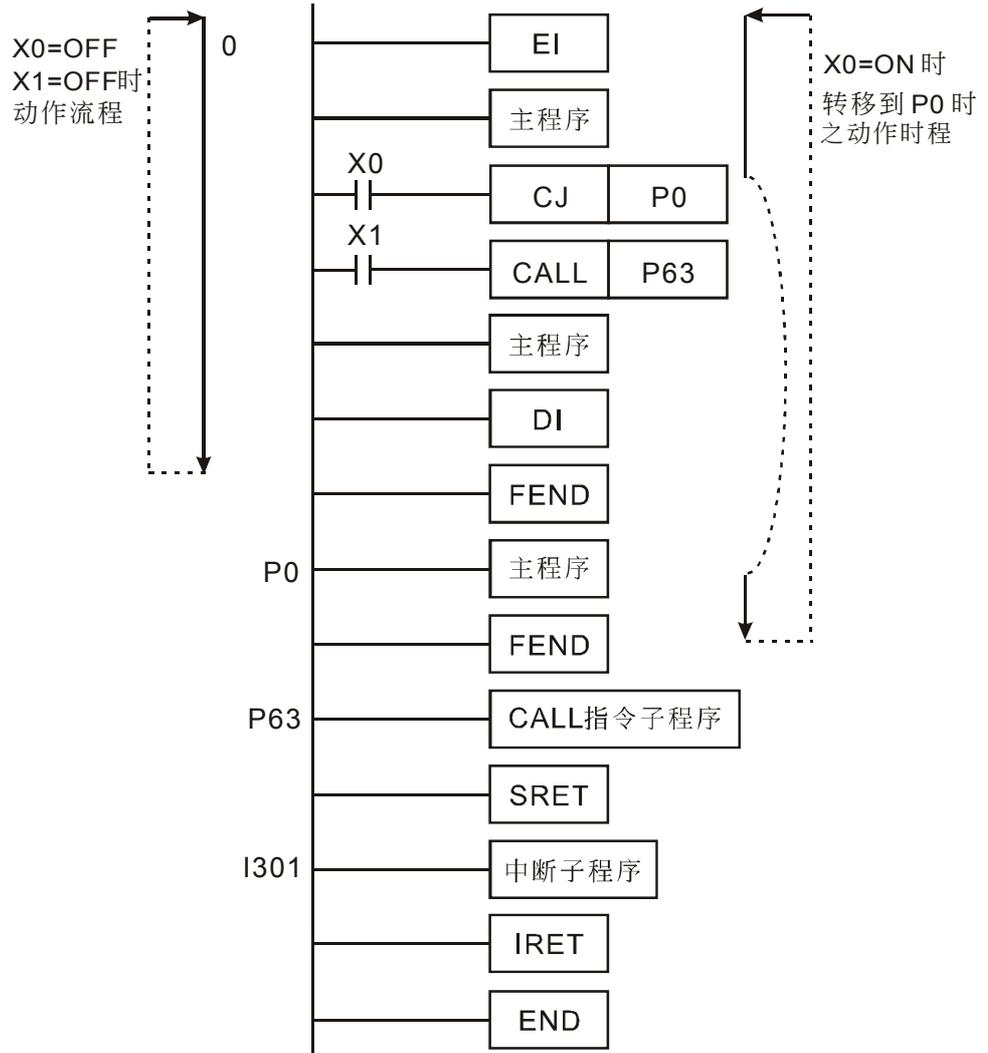


指令		操作数		功能														
FEND		无		主程序结束														
	位装置				字装置								16 位指令 (1 STEP)					
	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	FEND	连续执行型	-
<ul style="list-style-type: none"> 操作数使用注意：无操作数 不须接点驱动的指令 														32 位指令				
														- - - -				
														<ul style="list-style-type: none"> 标志信号：无 				

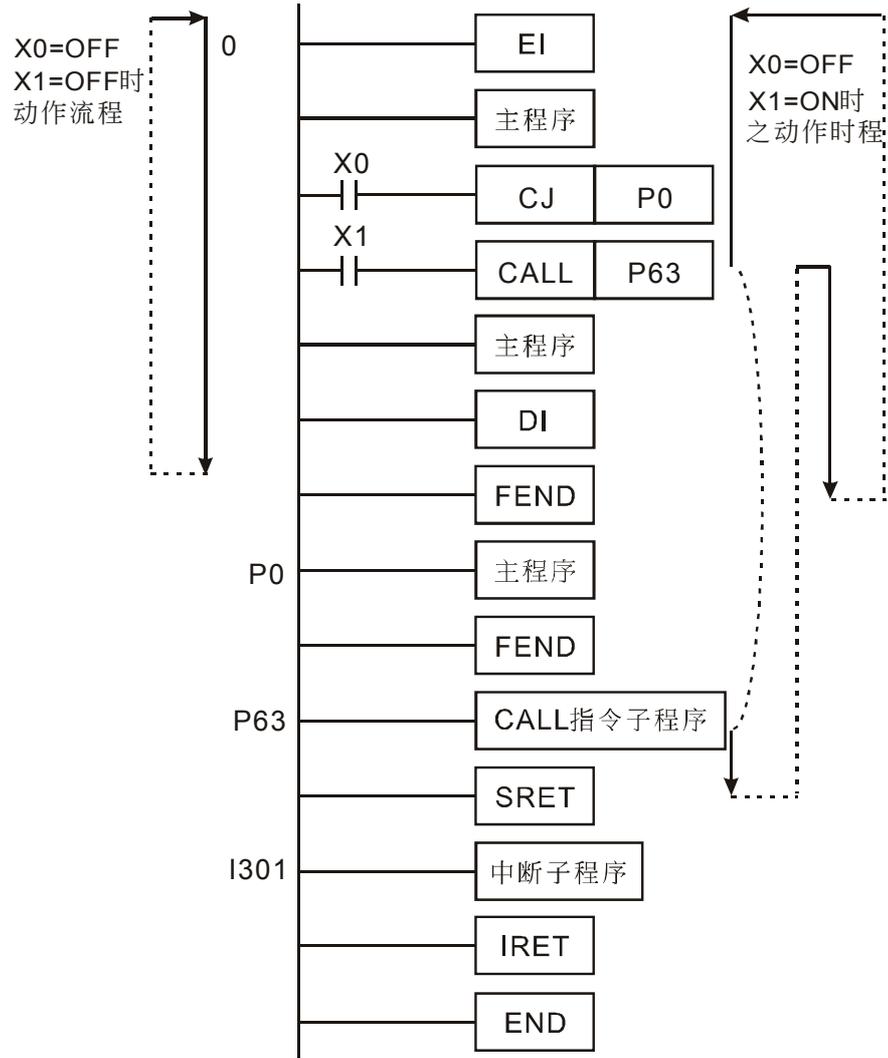
指令说明

- ◆ 此指令代表主程序结束，当 PLC 执行至此指令时，与 END 指令相同。
- ◆ CALL 指令的程序必须写在 FEND 指令后，并且在该子程序结束加上 SRET 指令，而在中断程序亦必须写在 FEND 的后，并在该服务程序结束加上 IRET 指令。
- ◆ 若使用多数个 FEND 指令时，请将子程序和中断服务程序设计于最后的 FEND 和 END 指令的间。
- ◆ CALL 指令执行后，在 SRET 指令执行前执行 FEND 指令会发生程序错误。
- ◆ FOR 指令执行后，在 NEXT 指令执行前执行 FEND 指令会发生程序错误。

CJ指令
动作流程



CALL指令
动作流程



指令		操作数		功能														
FOR		S		循环范围开始														
	位装置				字装置											16 位指令 (3 STEP) FOR 连续执行型 — —		
	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F			
S					*	*	*	*	*	*	*	*	*	*	*	*	32 位指令 — — — —	
<ul style="list-style-type: none"> 操作数使用注意：不须接点驱动的指令 																	<ul style="list-style-type: none"> 标志信号：无 	

指令说明

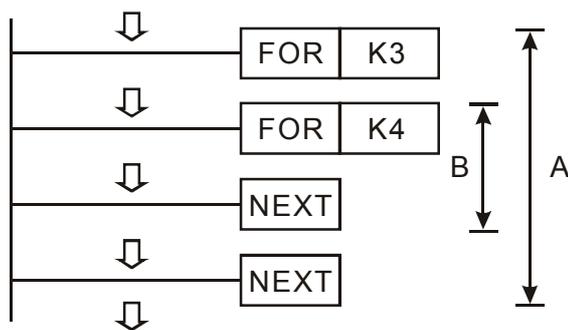
◆ **S**：回路重复执行的次数。

指令		操作数		功能														
NEXT		无		循环范围结束														
	位装置				字装置								16 位指令 (1 STEP)					
	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	NEXT	连续执行型	-
<ul style="list-style-type: none"> 操作数使用注意：无操作数 不须接点驱动的指令 												32 位指令						
												- - - -						
												<ul style="list-style-type: none"> 标志信号：无 						

指令说明

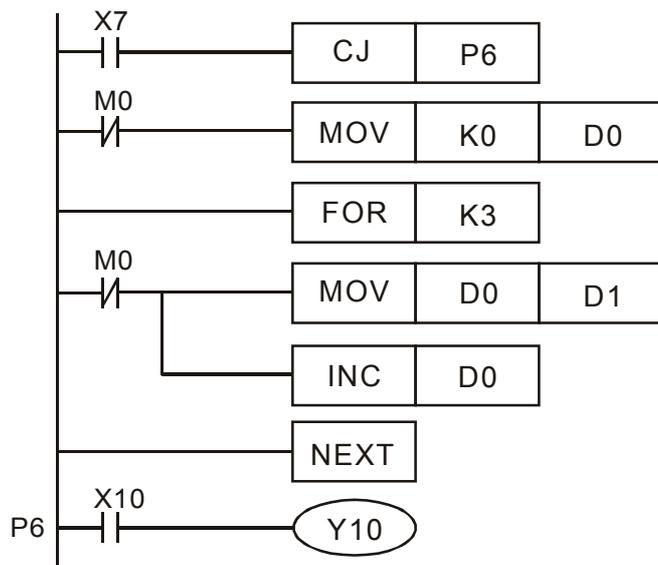
- ◆ 由 FOR 指令指定 FOR~NEXT 回圈来回执行 N 次后跳出 FOR~NEXT 回圈往下继续执行。
- ◆ 指定次数范围 N=K1~K32,767，当指定次数范围 $N \leq K1$ 时，都视为是 K1。
- ◆ 当不执行 FOR~NEXT 回路时，可使用 CJ 指令来跳出回路。
- ◆ 下列情形会产生错误：1. NEXT 指令在 FOR 指令的前。2. 有 FOR 指令没有 NEXT 指令。3. FEND 或 END 指令的后有 NEXT 指令时。4. FOR~NEXT 指令个数不同时。
- ◆ 嵌套式 FOR~NEXT 回路最多可使用 5 层，但要注意回路次数过多时，会使 PLC 扫描时间增加有可能造成超时监视定时器动作，而导致错误产生。
- ◆ A 程序执行 3 次后在到 NEXT 指令以后的程序继续执行。而 A 程序每执行一次 B 程序会执行四次，所以 B 程序合计共执行 $3 \times 4 = 12$ 次。

程序范例 (一)



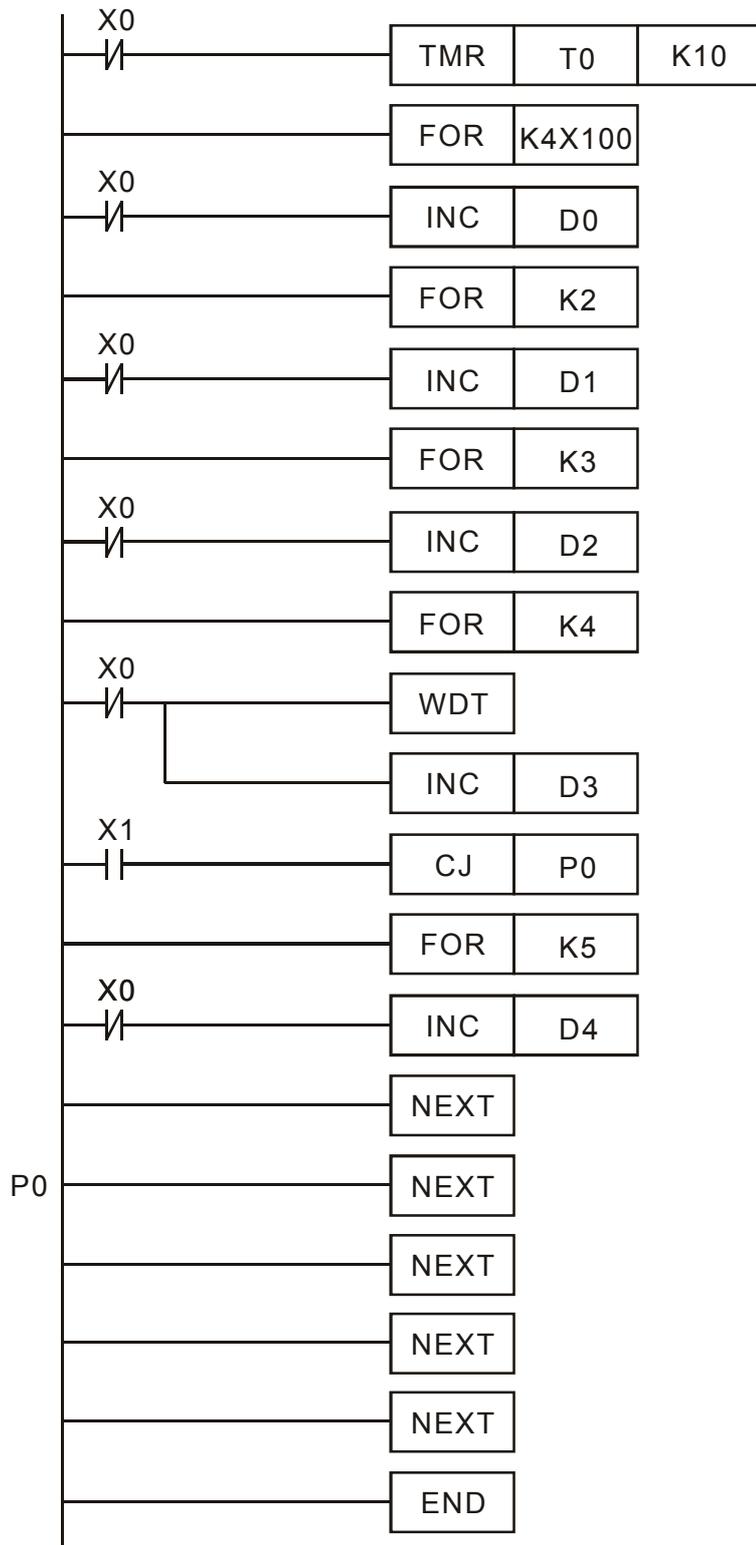
程序范例
(二)

- ◆ 当 X7=Off 时, PLC 会执行 FOR-NEXT 的间的程序, 当 X7=On 时, CJ 指令执行转移至 P6 处, FOR-NEXT 的间的程序跳过不执行。



程序范例
(三)

- ◆ 当不执行 FOR~NEXT 时，可使用 CJ 指令来转移。最内层 FOR ~ NEXT 回圈在 X1=On 时，CJ 指令执行转移至 P0 处而跳过不执行。



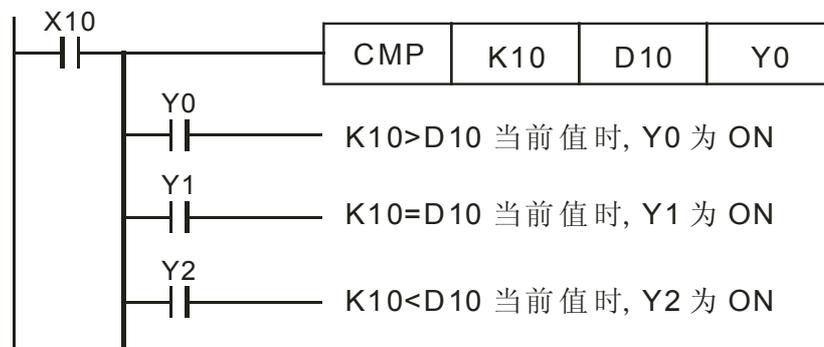
指令		操作数										功能								
CMP		D	(S1) (S2) (D)										比较设置输出							
	位装置				字装置											16 位指令 (7 STEP)				
	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	CMP 连续执行型				
S ₁					*	*	*	*	*	*	*	*	*	*	*	32 位指令 (13 STEP)				
S ₂					*	*	*	*	*	*	*	*	*	*	*	DCMP 连续执行型				
D		*	*	*												标志信号: 无				
<ul style="list-style-type: none"> 操作数使用注意: S₁、S₂操作数若使用 F 装置仅可使用 16 位指令 D 操作数会占用连续 3 点 																				

指令说明

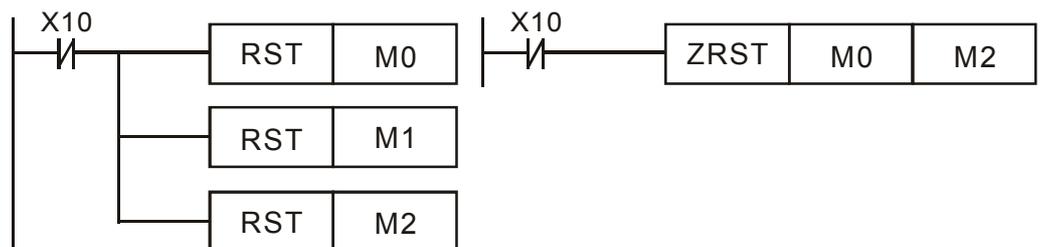
- ◆ **(S1)**: 比较值 1。 **(S2)**: 比较值 2。 **(D)**: 比较结果, 占用连续 3 点。
- ◆ 将操作数 **(S1)** 和 **(S2)** 的内容作大小比较, 其比较结果在 **(D)** 作表示。
- ◆ 大小比较是以代数来进行, 全部的数据是以有号数二进制数值来作比较。因此 16 位指令, b15 为 1 时, 表示为负数, 32 位指令, 则 b31 为 1 时, 表示为负数。

程序范例

- ◆ 定装置为 Y0, 则自动占有 Y0, Y1, Y2。
- ◆ X10=On 时, CMP 指令执行, Y0, Y1, Y2 其中的一会 On, 当 X10=Off 时, CMP 指令不执行, Y0, Y1, Y2 状态保持在 X10=Off 的前的状态。
- ◆ 需要得到 ≥、≤、≠的结果时, 可将 Y0~Y2 串并联即可取得。



- ◆ 要清除其比较结果请使用 RST 或 ZRST 指令。



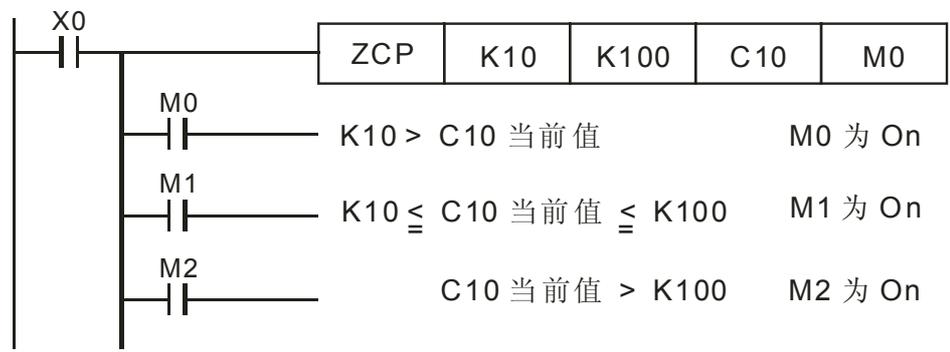
指令		操作数										功能								
ZCP		D	S1	S2	S	D	区间比较													
	位装置				字装置											16 位指令 (9 STEP) ZCP 连续执行型 <hr/> 32 位指令 (17 STEP) DZCP 连续执行型 • 标志信号: 无				
	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F					
S ₁					*	*	*	*	*	*	*	*	*	*	*					
S ₂					*	*	*	*	*	*	*	*	*	*	*					
S					*	*	*	*	*	*	*	*	*	*	*					
D		*	*	*																
• 操作数使用注意: S ₁ 、S ₂ 、S 操作数若使用 F 装置仅可使用 16 位指令 S ₁ 操作数内容值请小于 S ₂ 操作数内容值 D 操作数会占用连续 3 点																				

指令说明

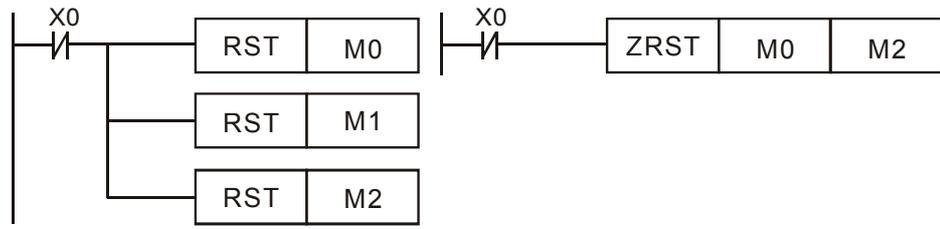
- ◆ **S1**: 区间比较的下限值。 **S2**: 区间比较的上限值。 **S**: 比较值。
- ◆ **D**: 比较结果, 占用连续 3 点。
- ◆ 比较值 **S** 与下限值 **S1** 和上限值 **S2** 作比较, 其比较结果在 **D** 作表示。
- ◆ 下限值 **S1** > 上限值 **S2** 时, 则指令以下限值 **S1** 作为上下限值进行比较。
- ◆ 大小比较是以代数来进行, 全部的数据是以有号数二进制数值来作比较。因此 16 位指令, b15 为 1 时, 表示为负数, 32 位指令, 则 b31 为 1 时, 表示为负数。

程序范例

- ◆ 定装置为 M0, 则自动占有 M0, M1, M2。
- ◆ X0=On 时, ZCP 指令执行, M0,M1,M2 其中的一会 On, 当 X0=Off 时, ZCP 指令不执行, M0,M1,M2 状态保持在 X0=Off 的前的状态。



- ◆ 要清除其结果请使用 RST 或 ZRST 指令。



指令		操作数		功能													
MOV		D	S D	数据传送													
	位装置				字装置										16 位指令 (5 STEP)		
	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	MOV	连续执行型
S					*	*	*	*	*	*	*	*	*	*	*		
D							*	*	*	*	*	*	*	*	*		
<ul style="list-style-type: none"> 操作数使用注意：S、D 操作数若使用 F 装置仅可使用 16 位指令 																<ul style="list-style-type: none"> 标志信号：无 	

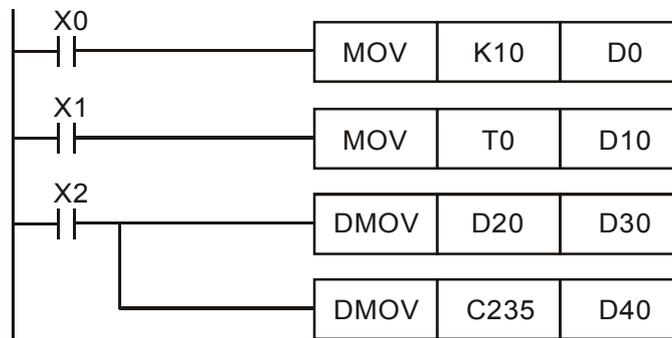
指令说明

- ◆ **S**：数据的来源。 **D**：数据的搬移目的地。
- ◆ 当该指令执行时，将 **S** 的内容直接搬移至 **D** 内。当指令不执行时，**D** 内容不会变化。
- ◆ 若演算结果为 32 位输出时，（如应用指令 MUL 等）和 32 位装置高速计数器的当前值数据搬动则必须要用 DMOV 指令。

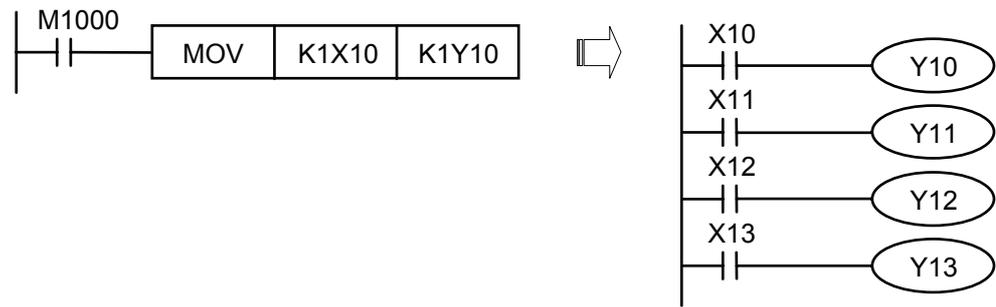
程序范例

- ◆ 16 位数据搬移，须使用 MOV 指令。
 - ◆ 当 X0=Off 时，D10 内容没有变化，若 X0=On 时，将数值 K10 传送至 D10 数据寄存器内。
 - ◆ 当 X1=Off 时，D10 内容没有变化，若 X1=On 时，将 T0 当前值传送至 D10 数据寄存器内。
- ◆ 32 位数据搬移，须使用 DMOV 指令。

当 X2=Off 时，(D31、D30)、(D41、D40)内容没有变化，若 X2=On 时，将(D21、D20)当前值传送至(D31、D30)数据寄存器内。同时，将 C235 当前值传送至(D41、D40)数据寄存器内。



- ◆ 位数据搬移：当程序执行时，将 X10~X13 四个 bit 内容搬移至 Y10~Y13 四个 bit 内。功能与右列程序相同。



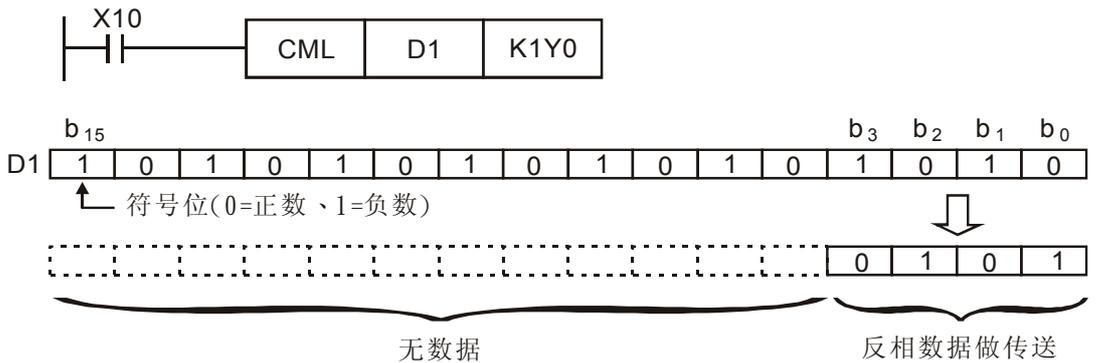
指令		操作数		功能													
CML		D	S D	反转传送													
	位装置				字装置										16 位指令 (5 STEP)		
	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	CML	连续执行型
S					*	*	*	*	*	*	*	*	*	*	*		
D							*	*	*	*	*	*	*	*	*		
• 操作数使用注意：S、D 操作数若使用 F 装置仅可使用 16 位指令												• 标志信号：无					

指令说明

- ◆ **S**：传送的数据来源。 **D**：传送的目的地装置。
- ◆ 将 **S** 的内容全部反相 (0→1、1→0) 传送至 **D** 当中。如果的内容为 K 常量时，此 K 常量自动被转换成 BIN 值。

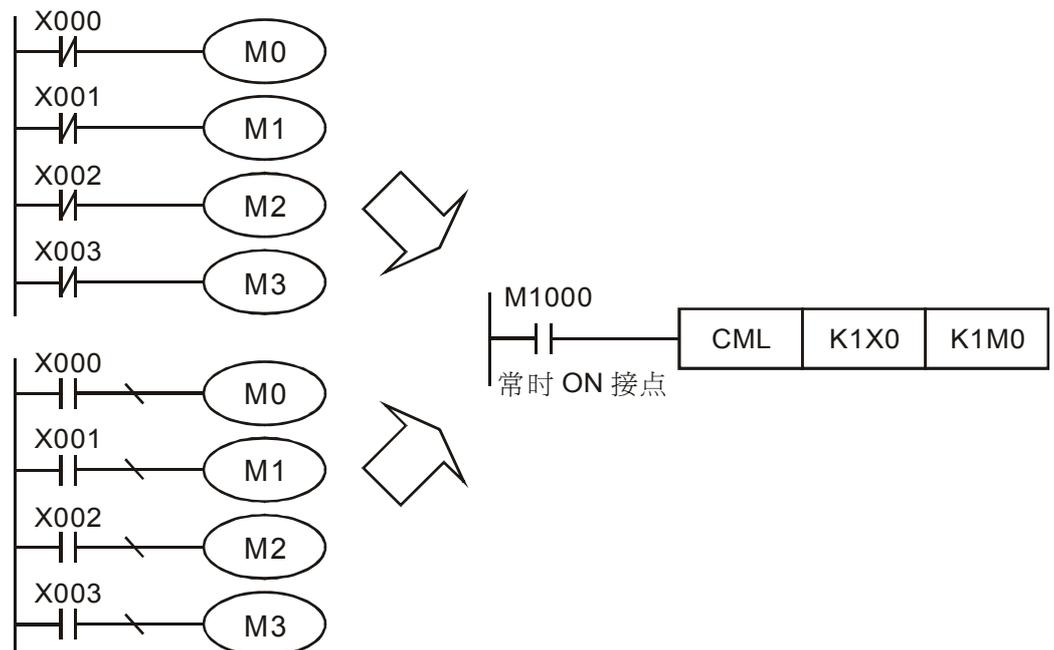
程序范例 (一)

- ◆ 希望作反相输出时，使用本指令。
- ◆ 当 X10=On 时，将 D10 的 b0~b3 内容反相后传送到 Y0~Y3。



程序范例 (二)

- ◆ 下图左边的回路也可以使用 CML 指令来表现，如下图右所示

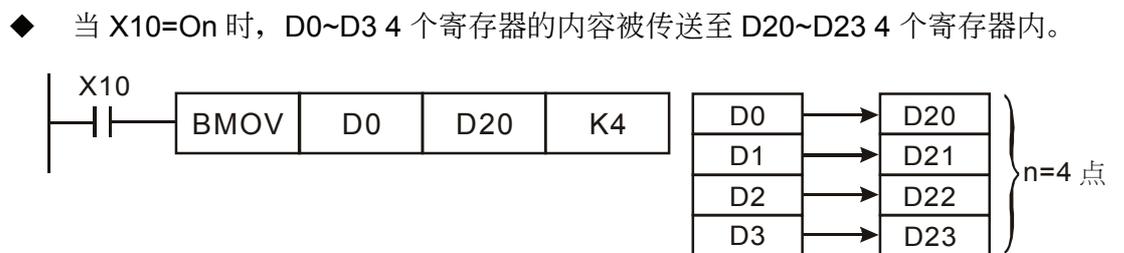


指令		操作数		功能												
BMOV		D	(S) (D) (n)	全部传送												
	位装置				字装置										16 位指令 (7 STEP) BMOV 连续执行型	
	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E		
S							*	*	*	*	*	*	*			
D								*	*	*	*	*	*			
n					*	*										
• 操作数使用注意: n 操作数范围 n = 1~512																
• 标志信号: 无																

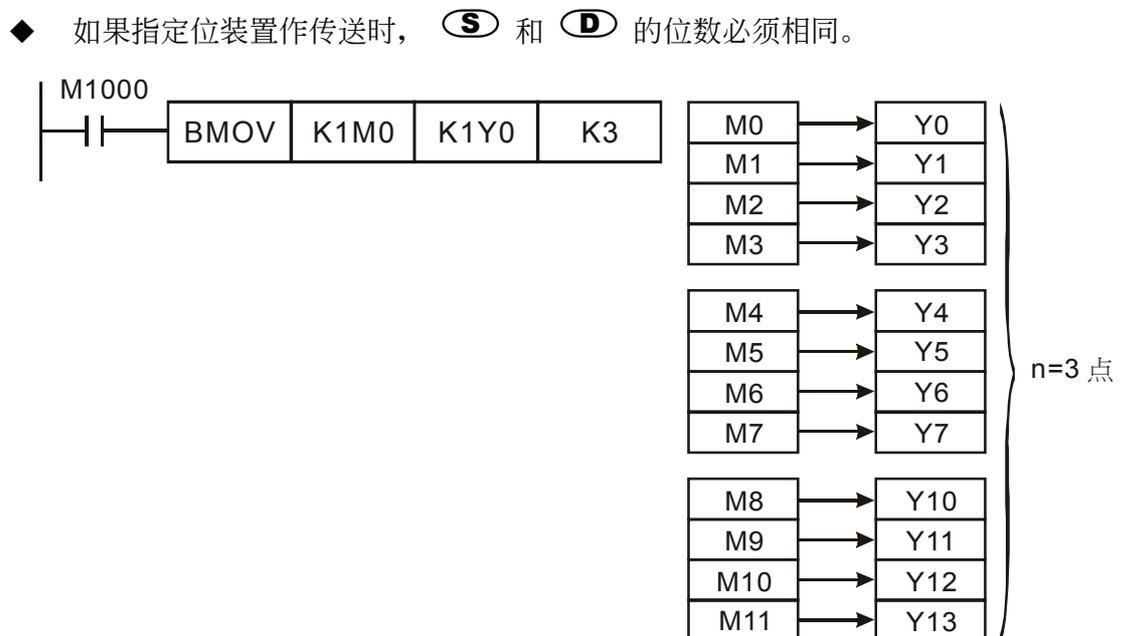
指令说明

- ◆ **(S)**: 来源装置起始。 **(D)**: 目的地装置起始。 **(n)**: 传送区块长度。
- ◆ **(S)** 所指定的装置起始号码开始算 n 个寄存器的内容被传送至 **(D)** 所指定的装置起始号码开始算 n 个寄存器当中, 如果 n 所指定点数超过该装置的使用范围时, 只有有效范围被传送。

程序范例 (一)



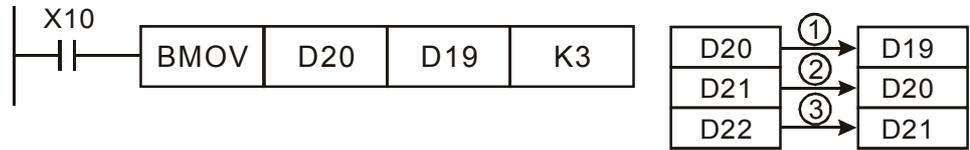
程序范例 (二)



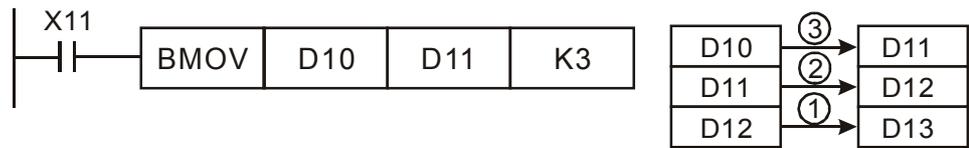
程序范例
(三)

◆ 为了防止两个操作数所指定传送的号码重叠时，所造成的混乱，请注意两个操作数所指定号码大小的安排，如下所示：

1. 当 **S** > **D** 的时，以①→②→③的顺序传送



2. 当 **S** < **D** 的时，是以③→②→①的顺序传送。



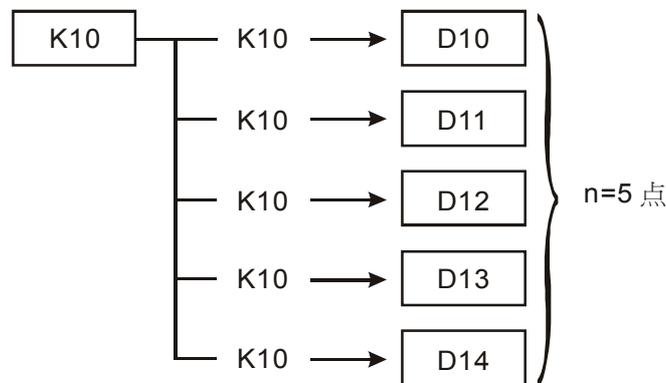
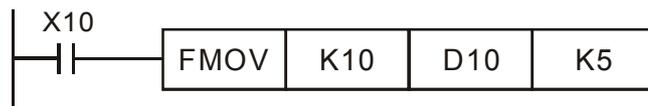
指令		操作数		功能												
FMOV		D	(S) (D) (n)	多点传送												
	位装置				字装置										16 位指令 (7 STEP)	
	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	FMOV 连续执行型
S					*	*	*	*	*	*	*	*	*	*	*	32 位指令 (13 STEP)
D							*	*	*	*	*	*			DFMOV 连续执行型	
n					*	*										<ul style="list-style-type: none"> 标志信号：无
<ul style="list-style-type: none"> 操作数使用注意：S 操作数若使用 F 装置仅可使用 16 位指令 n 操作数范围 n = 1~512 																

指令说明

- ◆ **(S)**：数据的来源。 **(D)**：目的地装置的起始。 **(n)**：传送区块长度。
- ◆ **(S)** 的内容被传送至 **(D)** 所指定的装置起始号码开始算 n 个寄存器当中，如果 n 所指定点数超过该装置的使用范围时，只有有效范围被传送。

程序范例

- ◆ 当 X10=On 时，K10 被传送到由 D10 开始的连续 5 个寄存器中。

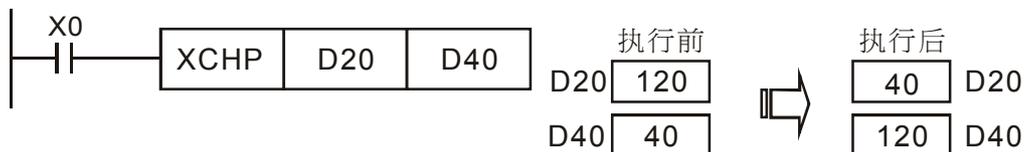


指令		操作数		功能													
XCH		D	(D1) (D2)	数据的交换													
	位装置				字装置										16 位指令 (5 STEP)		
	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	XCH	连续执行型
D ₁							*	*	*	*	*	*	*	*	*	32 位指令 (9 STEP)	
D ₂							*	*	*	*	*	*	*	*	*	DXCH	连续执行型
<ul style="list-style-type: none"> 操作数使用注意：D₁、D₂操作数若使用 F 装置仅可使用 16 位指令 														<ul style="list-style-type: none"> 标志信号：无 			

指令说明

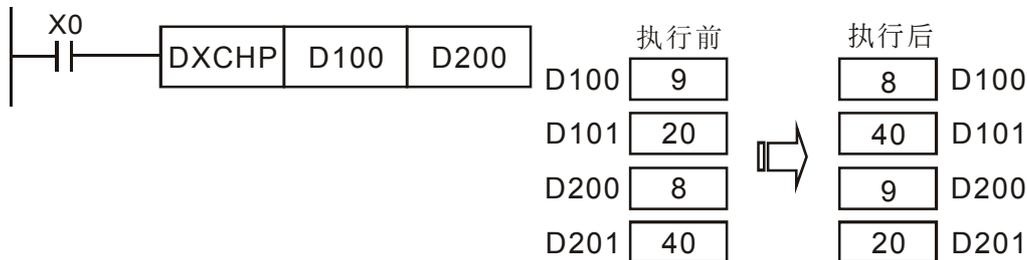
- ◆ **(D1)**：欲互相交换的数据 1。 **(D2)**：欲互相交换的数据 2。
- ◆ 将 **(D1)** 和 **(D2)** 所指定的装置内容值互相交换。
- ◆ X0=On 时，D20 与 D40 的内容互相交换。

程序范例 (一)



程序范例 (二)

- ◆ X0=On 时，D20 与 D40 的内容互相交换。



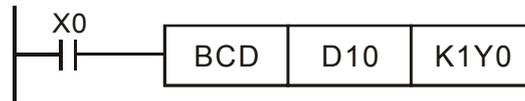
指令		操作数		功能													
BCD		D	S D	BIN→BCD 变换													
	位装置				字装置												16 位指令 (5 STEP) BCD 连续执行型
	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F		
S							*	*	*	*	*	*	*	*	*		
D							*	*	*	*	*	*	*	*	*		
<ul style="list-style-type: none"> 操作数使用注意：S、D 操作数若使用 F 装置仅可使用 16 位指令 												32 位指令 (9 STEP) DBCD 连续执行型 <ul style="list-style-type: none"> 标志信号：M1067 (演算错误) M1068 (演算错误锁定) 					

指令说明

- ◆ 数据来源 **S** 的内容 (BIN 值) 作 BCD 的转换, 存于 **D**。
- ◆ 在 BCD 变换结果若超过 0~9,999, 会设置演算错误标志 (M1067 和 M1068)。
- ◆ 在 DBCD 转换结果若超过 0~99,999,999, 会设置演算错误标志 (M1067 和 M1068)。
- ◆ PLC 内的四则运算、应用和 INC、DEC 指令都是以 BIN 方式来执行。所以在应用方面, 当要看到 BCD 数值的显示器时, 用 BCD 转换即可将 BIN 值变为 BCD 值输出。

程序范例

- ◆ 当 X0=On 时, D10 的 BIN 值被转换成 BCD 值后, 将结果存于 K1Y0 (Y0~Y3) 四个 BIT 中。



指令		操作数		功能													
BIN		D	S D	BCD→BIN 变换													
	位装置				字装置											16 位指令 (5 STEP)	
	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	BIN	连续执行型
S							*	*	*	*	*	*	*	*	*	32 位指令 (9 STEP)	
D							*	*	*	*	*	*	*	*	*	DBIN	连续执行型
<ul style="list-style-type: none"> 操作数使用注意：S、D 操作数若使用 F 装置仅可使用 16 位指令 												<ul style="list-style-type: none"> 标志信号：M1067 (演算错误) M1068 (演算错误锁定) 					

指令说明

- ◆ **S**：数据来源。 **D**：变换的结果。
- ◆ 数据来源 **S** 的内容（BCD：0~9,999）作 BIN 的转换，存于 **D**。
- ◆ 数据来源 **S** 的内容有效数值范围：BCD（0~9,999），DBCD（0~99,999,999）。当 **S** 的数据内容并非为 BCD 值（有任一位数不在 0~9 的范围内），则将会产生运算错误并设置错误标志（M1067 和 M1068）。
- ◆ 常量 K、H 会自动转换成 BIN 故不需运用此指令。

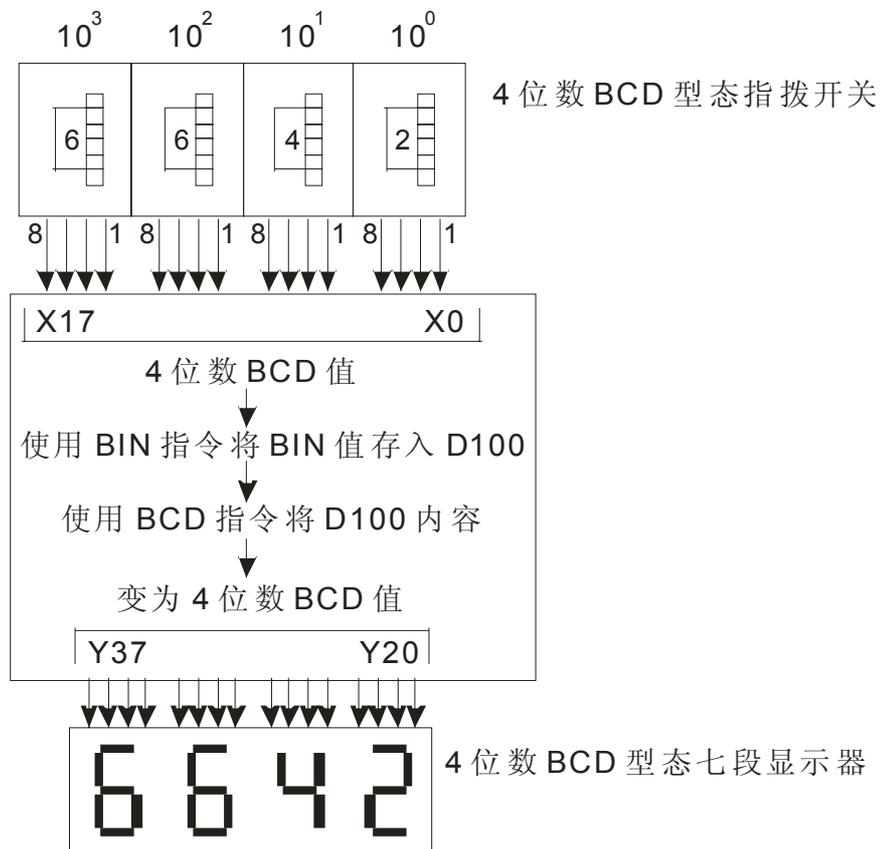
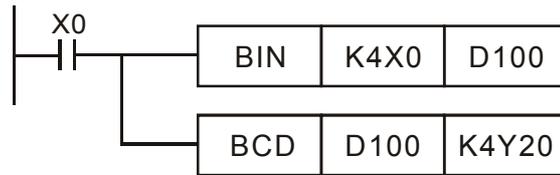
程序范例



补充说明

◆ BCD 与 BIN 指令应用说明:

- ◆ 当 PLC 要从外界读取一个 BCD 形态指拨开关时, 就必须使用 BIN 指令先将读取到的数据转换成 BIN 值再保存在 PLC 内。
- ◆ 当 PLC 要将内部保存的数据经由外界一个 BCD 形态的 7 段显示器显示出来时, 就必须使用 BCD 指令先将要显示的内部数据转换成 BCD 值再送到 7 段显示器。
- ◆ 当 X0=On 时, 将 K4X0 BCD 值转换成 BIN 值传送到 D100, 再将 D100 的 BIN 值转换成 BCD 值传送到 K4Y20。



指令		操作数										功能							
ADD		D	(S1) (S2) (D)										BIN 加法						
	位装置				字装置											16 位指令 (7 STEP)			
	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	ADD 连续执行型			
S ₁					*	*	*	*	*	*	*	*	*	*	*	32 位指令 (13 STEP)			
S ₂					*	*	*	*	*	*	*	*	*	*	*	DADD 连续执行型			
D								*	*	*	*	*	*	*	*	<ul style="list-style-type: none"> 标志信号: M1020 零标志 Zero flag M1021 借位标志 Barrow flag M1022 进位标志 Carry flag 			
<ul style="list-style-type: none"> 操作数使用注意: S₁、S₂、D 操作数若使用 F 装置仅可使用 16 位指令 																			

指令说明

- ◆ **(S1)**: 被加数。 **(S2)**: 加数。 **(D)**: 和。
- ◆ 将两个数据源: **(S1)** 和 **(S2)** 以 BIN 方式相加的结果存于 **(D)**。
- ◆ 各数据的最高位为符号位 0 表 (正) 1 表 (负), 因此可做代数加法运算。(例如: 3+(-9)=-6)
- ◆ 加法相关标志变化。

16 位 BIN 加法:

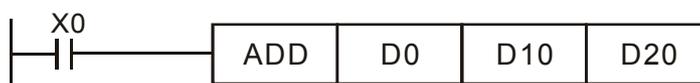
- ◆ 演算结果为 0 时, 零标志 (Zero flag) M1020 为 On。
- ◆ 演算结果小于 -32,768 时, 借位标志 (Barrow flag) M1021 为 On。
- ◆ 演算结果大于 32,767 时, 进位标志 (Carry flag) M1022 为 On。

32 位 BIN 加法:

1. 演算结果为 0 时, 零标志 (Zero flag) M1020 为 On。
2. 演算结果小于 -2,147,483,648 时, 借位标志 (Barrow flag) M1021 为 On。
3. 演算结果大于 2,147,483,647 时, 进位标志 (Carry flag) M1022 为 On。

程序范例 (一)

- ◆ 16 位 BIN 加法: 当 X0=On 时, 被加数 D0 内容加上加数 D10 的内容将结果存在 D20 的内容当中。



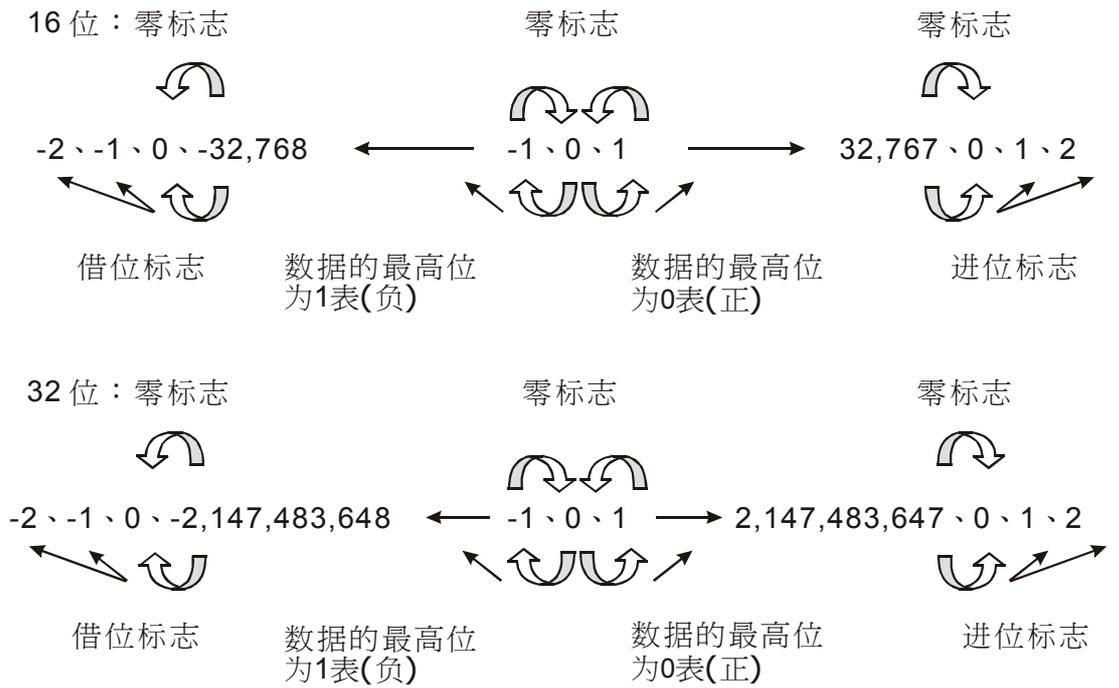
程序范例 (二)

- ◆ 32 位 BIN 加法: 当 X1=On 时, 被加数(D31、D30)内容加上加数(D41、D40)的内容将结果存在(D51、D50)的中。(其中 D30、D40、D50 为低 16 位数据, D31、D41、D51 为高 16 位数据)



补充说明

◆ 标志动作与数值的正负关系：



指令		操作数										功能								
SUB		D	(S1) (S2) (D)										BIN 减法							
	位装置				字装置											16 位指令 (7 STEP)				
	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	SUB 连续执行型				
S ₁					*	*	*	*	*	*	*	*	*	*	*	32 位指令 (13 STEP)				
S ₂					*	*	*	*	*	*	*	*	*	*	*	DSUB 连续执行型				
D								*	*	*	*	*	*	*	*	<ul style="list-style-type: none"> 标志信号: M1020 零标志 Zero flag M1021 借位标志 Barrow flag M1022 进位标志 Carry flag 				
<ul style="list-style-type: none"> 操作数使用注意: S₁、S₂、D 操作数若使用 F 装置仅可使用 16 位指令 																				

指令说明

- ◆ **(S1)**: 被减数。 **(S2)**: 减数。 **(D)**: 差。
- ◆ 将两个数据源: **(S1)** 和 **(S2)** 以 BIN 方式相减的结果存于 **(D)**。
- ◆ 各数据的最高位为符号位 0 表 (正) 1 表 (负), 因此可做代数减法运算。
- ◆ 减法相关标志变化。

16 位 BIN 减法:

- ◆ 演算结果为 0 时, 零标志 (Zero flag) M1020 为 On。
- ◆ 演算结果小于 -32,768 时, 借位标志 (Barrow flag) M1021 为 On。
- ◆ 演算结果大于 32,767 时, 进位标志 (Carry flag) M1022 为 On。

32 位 BIN 减法:

1. 演算结果为 0 时, 零标志 (Zero flag) M1020 为 On。
2. 演算结果小于 -2,147,483,648 时, 借位标志 (Barrow flag) M1021 为 On。
3. 演算结果大于 2,147,483,647 时, 进位标志 (Carry flag) M1022 为 On。

- ◆ 标志动作与数值的正负关系参考标志动作与数值的正负关系请参考上页指令 ADD 的补充说明。

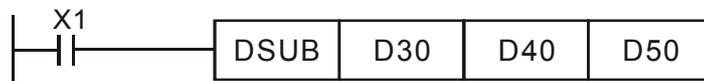
程序范例 (一)

- ◆ 16 位 BIN 减法: 当 X0=On 时, 将 D0 内容减掉 D10 内容将差存在 D20 的内容中。



程序范例 (二)

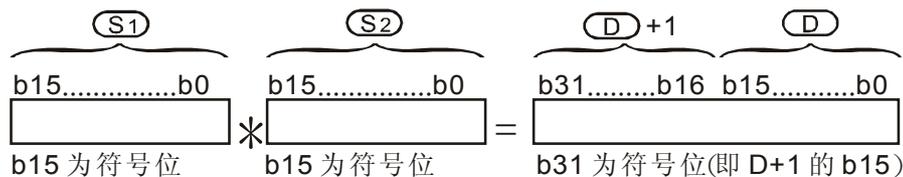
- ◆ 32 位 BIN 减法: 当 X1=On 时, (D31、D30)内容减掉(D41、D40)的内容将差存在(D51、D50)的中。(其中 D30、D40、D50 为低 16 位数据, D31、D41、D51 为高 16 位数据)



指令		操作数										功能							
MUL		D	(S1) (S2) (D)										BIN 乘法						
	位装置				字装置										16 位指令 (7 STEP)				
	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	MUL 连续执行型			
S ₁					*	*	*	*	*	*	*	*	*	*	*	32 位指令 (13 STEP)			
S ₂					*	*	*	*	*	*	*	*	*	*	*	DMUL 连续执行型			
D								*	*	*	*	*	*	*	*	标志信号: 无			
操作数使用注意: S ₁ 、S ₂ 操作数若使用 F 装置仅可使用 16 位指令 D 操作数若使用 E 装置仅可使用 16 位指令																			

指令说明

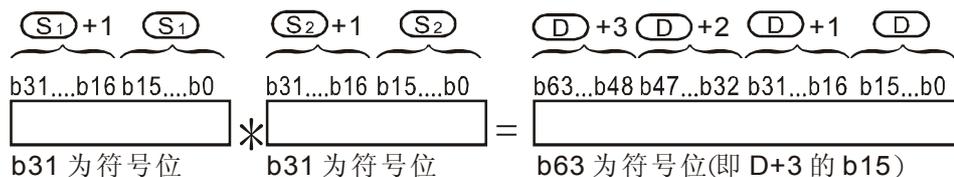
- ◆ **(S1)**: 被乘数。 **(S2)**: 乘数。 **(D)**: 积。
- ◆ 将两个数据源: **(S1)** 和 **(S2)** 以有号数二进制方式相乘后的积存于 **(D)**。必须注意 16 位和 32 位运算时, **(S1)**、**(S2)** 和 **(D)** 的正负号位。
- ◆ 16 位 BIN 乘法运算:



符号位=0 为正数, 符号位=1 为负数。

(D) 为位装置时, 可指定 K1~K8 构成 32 位。

- ◆ 32 位 BIN 乘法运算: :

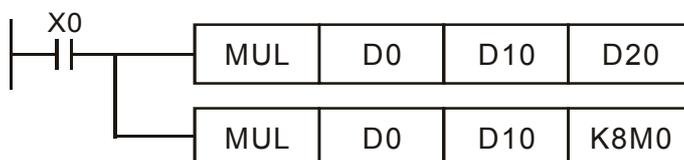


符号位=0 为正数, 符号位=1 为负数。

(D) 为位装置时, 仅可指定 K1~K8 构成 32 位, 只保存低 32 位数据。

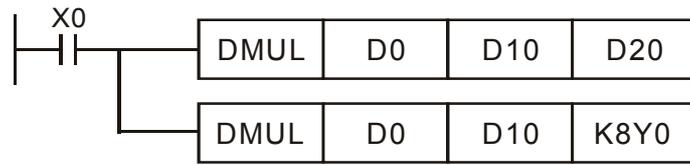
程序范例 (一)

- ◆ 16 位 DO 乘上 16 位 D10 其结果是 32 位的积, 上 16 位存于 D21, 下 16 位存于 D20 内, 结果的正负由最左边位的 Off / On 来代表正 0 负 1 值。



程序范例
(二)

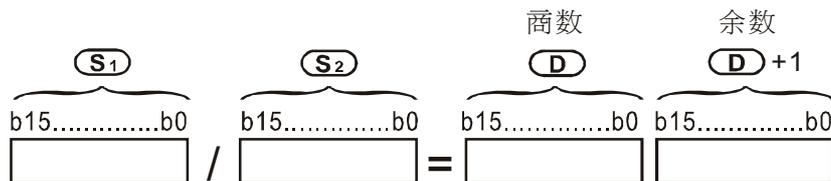
- ◆ 32 位 (D1, D0) 乘上 32 位 (D11, D10) 乘数其结果存于 64 位 (D23, D22, D21, D20) 结果的正负由最左边位的 Off / On 来代表正 0 负 1 值。



指令	D	操作数	功能													
DIV	P	(S1) (S2) (D)	BIN 除法													
	位装置				字装置											16 位指令 (7 STEP) DIV 连续执行型
	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	
S ₁					*	*	*	*	*	*	*	*	*	*	*	
S ₂					*	*	*	*	*	*	*	*	*	*	*	
D							*	*	*	*	*	*	*			
• 标志信号：无 • 操作数使用注意：S ₁ 、S ₂ 操作数若使用 F 装置仅可使用 16 位指令 D 操作数若使用 E 装置仅可使用 16 位指令																

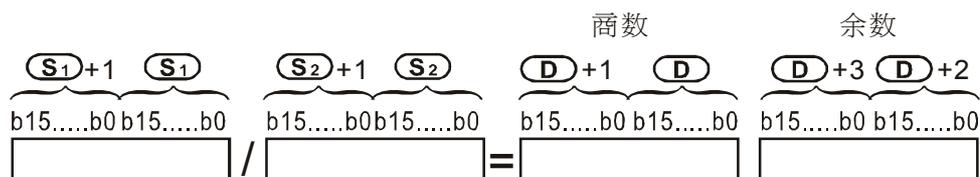
指令说明

- ◆ **(S1)**：被除数。**(S2)**：除数。**(D)**：商和余数。
- ◆ 将两个数据源：**(S1)** 和 **(S2)** 以有号数二进制方式相除后的商和余数存于 **(D)**。必须注意 16 位和 32 位运算时，**(S1)**、**(S2)** 和 **(D)** 的正负号位。
- ◆ 除数为 0 时，指令不执行。
- ◆ 16 位 BIN 除法运算：



(D) 为位装置时，可指定 K1~K8 构成 32 位得到商和余数。

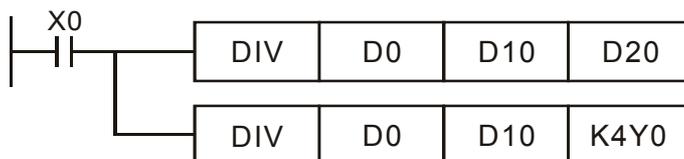
- ◆ 32 位 BIN 除法运算：



(D) 为位装置时，仅可指定 K1~K8 构成 32 位，只得到商数无余数。

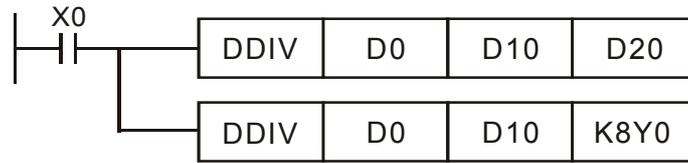
程序范例 (一)

- ◆ 当 X0=On 时，被除数 D0 除以除数 D10 而结果商被指定放于 D20，余数指定放于 D21 内。所得结果的正负由最高位的 Off / On 来代表正 (0) 负 (1) 值。



程序范例
(二)

- ◆ 当 X0=On 时，被除数 (D1, D0) 除以除数 (D11, D10) 而结果商被指定放于 (D21, D20)，余数指定放于 (D23, D22) 内。所得结果的正负由最高位的 Off / On 来代表正 (0) 负 (1) 值。

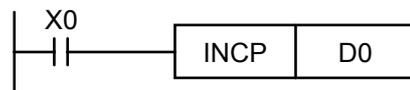


指令		操作数		功能													
INC		D	(D)	BIN 加一													
	位装置				字装置											16 位指令 (3 STEP)	
	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	INC	连续执行型
D							*	*	*	*	*	*	*	*	*	32 位指令 (5 STEP)	
<ul style="list-style-type: none"> 操作数使用注意: D 操作数若使用 F 装置仅可使用 16 位指令 																DINC 连续执行型	
																<ul style="list-style-type: none"> 标志信号: 无 	

指令说明

- ◆ **(D)**: 目的地装置。
- ◆ 当指令执行时, 程序每次扫描周期被指定的装置 **(D)** 内容都会加 1。
- ◆ 16 位运算时, 32,767 再加 1 则变为-32,768。32 位运算时, 2,147,483,647 再加 1 则变为-2,147,483,648,
- ◆ 本指令运算结果不会影响任何标志信号。
- ◆ 当 X0=On 时, D0 内容自动加 1。

程序范例



指令		操作数		功能													
DEC		D	D	BIN 减一													
	位装置				字装置											16 位指令 (3 STEP)	
	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	DEC	连续执行型
D							*	*	*	*	*	*	*	*	*	32 位指令 (5 STEP)	
<ul style="list-style-type: none"> 操作数使用注意：D 操作数若使用 F 装置仅可使用 16 位指令 																<ul style="list-style-type: none"> 标志信号：无 	

指令说明

- ◆ **D**：目的地装置。
- ◆ 当指令执行时，程序每扫描一次被指定的装置 **D** 内容减 1。
- ◆ 16 位运算时，-32,768 再减 1 则变为 32,767。32 位运算时，-2,147,483,648 再减 1 则变为 2,147,483,647。
- ◆ 本指令运算结果不会影响任何标志信号。
- ◆ 当 X0=On 时，D0 内容自动减 1。

程序范例



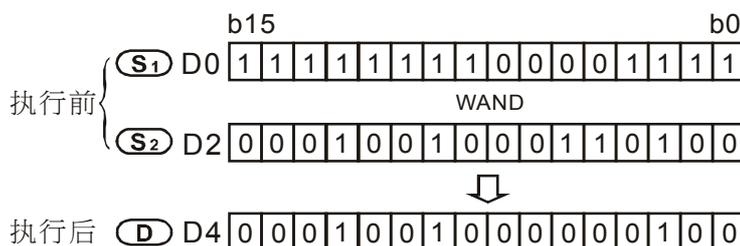
指令		操作数		功能														
AND		D	(S1) (S2) (D)	逻辑和(AND)运算														
	位装置				字装置										16 位指令 (7 STEP) WAND 连续执行型			
	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E			F	
S ₁					*	*	*	*	*	*	*	*	*	*	*	*	32 位指令 (13 STEP) DAND 连续执行型	
S ₂					*	*	*	*	*	*	*	*	*	*	*	*		
D								*	*	*	*	*	*	*	*	*	标志信号: 无	
• 操作数使用注意: S ₁ 、S ₂ 、D 操作数若使用 F 装置仅可使用 16 位指令																		

指令说明

- ◆ **(S1)**: 数据来源装置 1。 **(S2)**: 数据来源装置 2。 **(D)**: 运算结果。
- ◆ 两个数据源: **(S1)** 和 **(S2)** 作逻辑的'和' (AND) 运算并将结果存于 **(D)**。
- ◆ 逻辑的'和' (AND) 运算的规则为任一为 0 结果为 0。

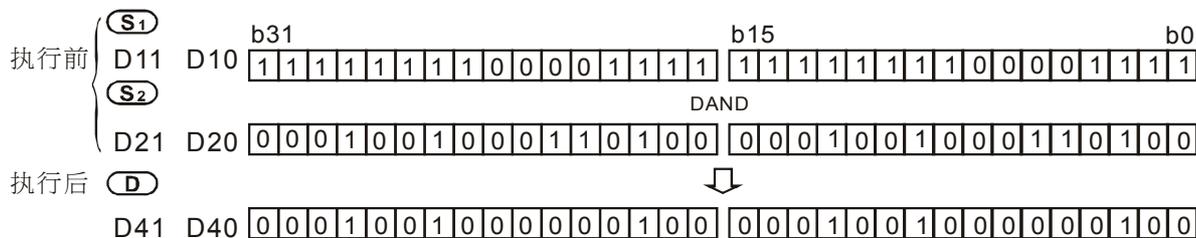
程序范例 (一)

- ◆ 当 X0=On 时, 16 位 D0 与 D2 作 WAND, 逻辑和(AND)运算, 将结果存于 D4 中。



程序范例 (二)

- ◆ 当 X1=On 时, 32 位(D11、D10)与(D21、D20)作 DAND, 逻辑和(AND)运算, 将结果存于(D41、D40)中。

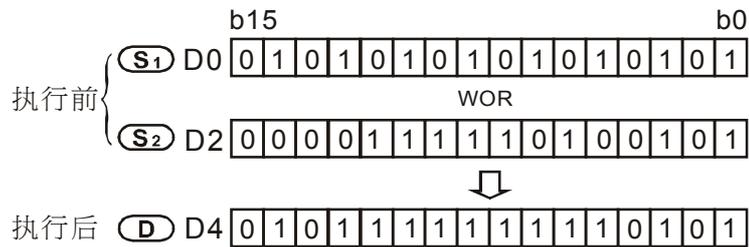


指令		操作数		功能													
OR		D	(S1) (S2) (D)	逻辑或(OR)运算													
	位装置				字装置										16 位指令 (7 STEP)		
	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	WOR	连续执行型
S ₁					*	*	*	*	*	*	*	*	*	*	*		32 位指令 (13 STEP) DOR 连续执行型
S ₂					*	*	*	*	*	*	*	*	*	*	*		
D								*	*	*	*	*	*	*	*		
• 操作数使用注意: S ₁ 、S ₂ 、D 操作数若使用 F 装置仅可使用 16 位指令 • 标志信号: 无																	

指令说明

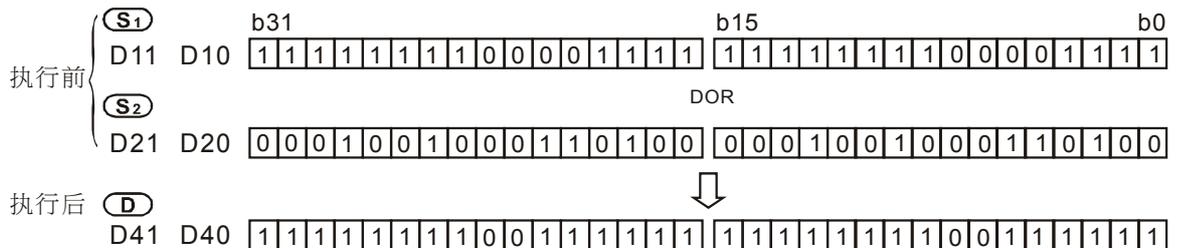
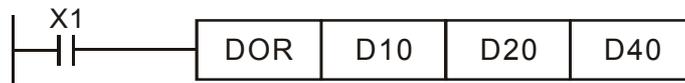
- ◆ **(S1)**: 数据来源装置 1。 **(S2)**: 数据来源装置 2。 **(D)**: 运算结果。
- ◆ 两个数据源: **(S1)** 和 **(S2)** 作逻辑的'或' (OR) 运算结果存于 **(D)**。
- ◆ 逻辑的'或' (OR) 运算的规则为任一为 1 结果为 1。
- ◆ 当 X1=On 时, 16 位 D0 与 D2 作 WOR, 逻辑或(OR)运算, 将结果存于 D4 中。

程序范例 (一)



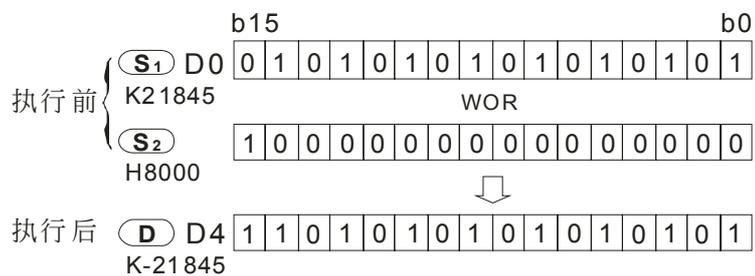
程序范例 (二)

- ◆ 当 X1=On 时, 32 位(D11、D10)与(D21、D20)作 DOR, 逻辑或(OR)运算, 将结果存于(D41、D40)中。



程序范例
(三)

◆ 正数变负数。



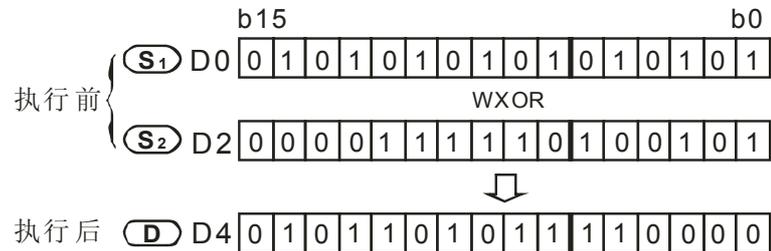
指令		操作数		功能											
XOR		D	(S1) (S2) (D)	逻辑异或(XOR)运算											
	位装置				字装置										16 位指令 (7 STEP) WXOR 连续执行型
	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	
S ₁					*	*	*	*	*	*	*	*	*	*	*
S ₂					*	*	*	*	*	*	*	*	*	*	*
D								*	*	*	*	*	*	*	*
• 标志信号：无 • 操作数使用注意：S ₁ 、S ₂ 、D 操作数若使用 F 装置仅可使用 16 位指令															

指令说明

- ◆ **(S1)**：数据来源装置 1。 **(S2)**：数据来源装置 2。 **(D)**：运算结果。
- ◆ 两个数据源：**(S1)** 和 **(S2)** 作逻辑的'异或' (XOR) 运算结果存于 **(D)**。
- ◆ 逻辑的'异或' (OR) 运算的规则为两者相同结果为 0，两者不同结果为 1。

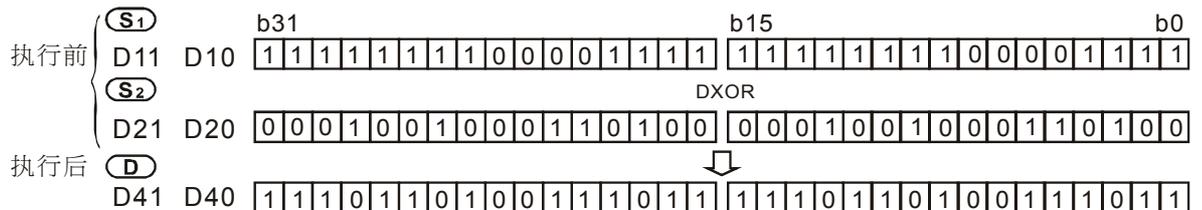
程序范例 (一)

- ◆ 当 X0=On 时，16 位 D0 与 D2 作 WXOR，逻辑异或(XOR)运算，将结果存于 D4 中。



程序范例 (二)

- ◆ 当 X1=On 时，32 位(D11、D10)与(D21、D20)作 DXOR，逻辑异或(XOR)运算，将结果存于(D41、D40)中。



指令		操作数		功能											
NEG		D	D	求补码											
	位装置				字装置										16 位指令 (3 STEP) NEG 连续执行型
	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	
D							*	*	*	*	*	*	*	*	*
• 操作数使用注意：D 操作数若使用 F 装置仅可使用 16 位指令											32 位指令 (5 STEP) DNEG 连续执行型 • 标志信号：无				

指令说明

- ◆ **D**：求补码的装置。
- ◆ 本指令可将负数的 BIN 值转换成绝对值。

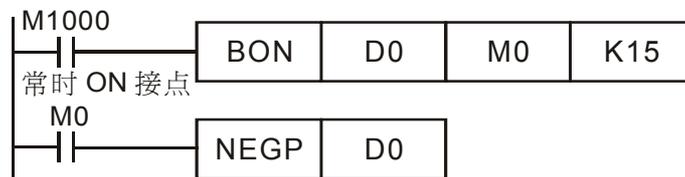
程序范例 (一)

- ◆ 当 X=Off → On 时，D10 内容的各位全部反相 (0→1、1→0) 后再加 1 存放于原寄存器 D10 当中。



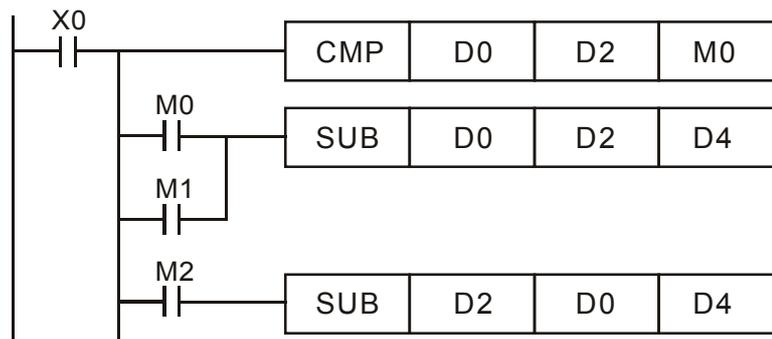
程序范例 (二)

- ◆ 求负数的绝对值
 1. 当 D0 的第 15 个位为“1”时，M0=On。(D0 为负数)
 2. M0=On 时，用 NEG 指令将 D0 取 2 的补码可得到其绝对值。



程序范例 (三)

- ◆ 减法运算的差取绝对值。
 1. 当 D0>D2 时，M0=On。
 2. 当 D0=D2 时，M1=On。
 3. 当 D0<D2 时，M2=On。
 4. 此可得 D4 一直为正值。



补充说明

◆ 负数的表现和绝对值

- ◆ 正负数是以寄存器最上位（最左边）的位内容来表现，为“0”时，为正数、为“1”时，为负数。
- ◆ 碰到负数时，可使用 **NEG** 指令将它转成绝对值。

(D0)=2
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0

(D0)=1
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1

(D0)=0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

(D0)=-1 → $(\overline{D0})+1=1$
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 → 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1

(D0)=-2 → $(\overline{D0})+1=2$
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 → 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0

(D0)=-3 → $(\overline{D0})+1=3$
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 → 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1

(D0)=-4 → $(\overline{D0})+1=4$
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 → 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0

(D0)=-5 → $(\overline{D0})+1=5$
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 → 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1

⋮

⋮

(D0)=-32,765 → $(\overline{D0})+1=32,765$
 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 → 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1

(D0)=-32,766 → $(\overline{D0})+1=32,766$
 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 → 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0

(D0)=-32,767 → $(\overline{D0})+1=32,767$
 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 → 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

(D0)=-32,768 → $(\overline{D0})+1=-32,768$
 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 → 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0



绝对值最大只可到32,767

指令		操作数		功能													
ROR		D	(D) (n)	右循环移位													
	位装置				字装置												16 位指令 (5 STEP) ROR 连续执行型
	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F		
D								*	*	*	*	*	*	*	*		
n					*	*											
<ul style="list-style-type: none"> 操作数使用注意：D 操作数若使用 F 装置仅可使用 16 位指令 D 操作数若指定为 KnY、KnM、KnS 时，只有 K4（16 位）和 K8（32 位）有效 n 操作数中 n=1~16（16 位），n=1~32（32 位） 														32 位指令 (9 STEP) DROR 连续执行型 <ul style="list-style-type: none"> 标志信号：M1022 进位标志 Carry flag 			

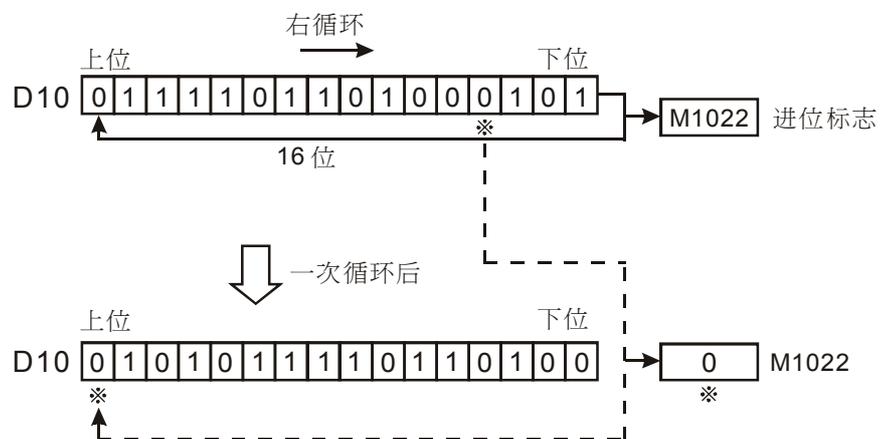
指令说明

◆ **(D)**：欲循环的装置。**(n)**：一次循环的位数。

◆ 将 **(D)** 所指定的装置内容一次向右循环 **(n)** 个位。

程序范例

◆ 当 X0 从 Off → On 变化时，D10 的 16 个位以 4 个位为一组往右循环，如下图所示标明※的位内容被传送至进位标志信号 M1022 里。

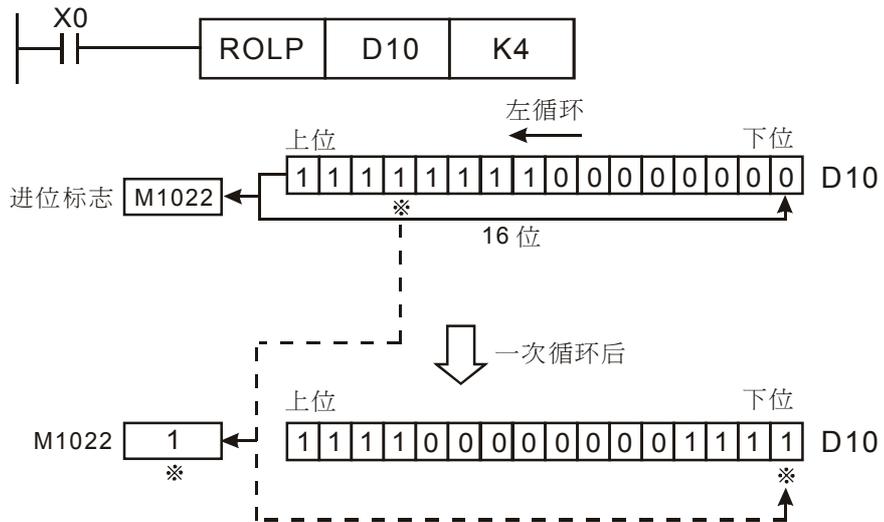


指令		操作数		功能													
ROL		D	(D) (n)	左循环移位													
	位装置				字装置												16 位指令 (5 STEP) ROL 连续执行型
	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F		
D							*	*	*	*	*	*	*	*	*		
n				*	*												
<ul style="list-style-type: none"> 操作数使用注意：D 操作数若使用 F 装置仅可使用 16 位指令 D 操作数若指定为 KnY、KnM、KnS 时，只有 K4（16 位）和 K8（32 位）有效 n 操作数中 n=1~16（16 位），n=1~32（32 位） 												32 位指令 (9 STEP) DROL 连续执行型 <ul style="list-style-type: none"> 标志信号：M1022 进位标志 Carry flag 					

指令说明

程序范例

- ◆ **(D)**：欲循环的装置。**(n)**：一次循环的位数。
- ◆ 将 **(D)** 所指定的装置内容一次向左循环 **(n)** 个位。
- ◆ 当 X0 从 Off → On 变化时，D0 的 16 个位以 4 个位一组往左循环，如下图所示标明※的位内容被传送至进位标志信号 M1022 里。

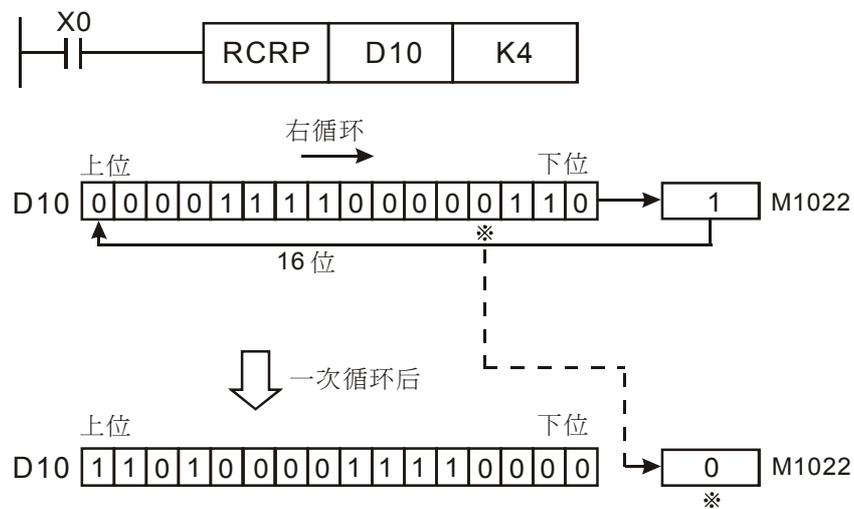


指令		操作数		功能													
RCR		D	(D) (n)	附进位标志右循环													
	位装置				字装置										16 位指令 (5 STEP)		
	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	RCR	连续执行型
D								*	*	*	*	*	*	*	*	32 位指令 (9 STEP)	
n					*	*										DRCR	连续执行型
<ul style="list-style-type: none"> 操作数使用注意：D 操作数若使用 F 装置仅可使用 16 位指令 D 操作数若指定为 KnY、KnM、KnS 时，只有 K4（16 位）和 K8（32 位）有效 n 操作数中 n=1~16（16 位），n=1~32（32 位） 														<ul style="list-style-type: none"> 标志信号：M1022 进位标志 Carry flag 			

指令说明

程序范例

- ◆ **(D)**：欲循环的装置。**(n)**：一次循环的位数。
- ◆ 将 **(D)** 所指定的装置内容连同进位标志 M1022，一次向右循环 **(n)** 个位。
- ◆ 当 X0 从 Off → On 变化时，D0 的 16 个位连同进位标志 M1022 共 17 个位以 4 个位为一组往右旋转，如下图所示标明※的位内容被传送至进位标志信号 M1022 里。



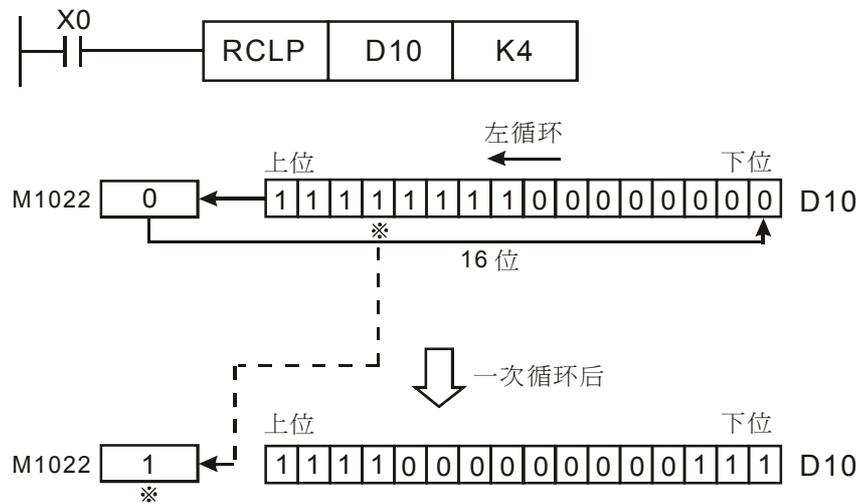
指令		操作数		功能													
RCL		D	(D) (n)	附进位标志左循环													
	位装置				字装置										16 位指令 (5 STEP)		
	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	RCL	连续执行型
D							*	*	*	*	*	*	*	*	*	32 位指令 (9 STEP)	
n					*	*										DRCL	连续执行型
<ul style="list-style-type: none"> 操作数使用注意：D 操作数若使用 F 装置仅可使用 16 位指令 D 操作数若指定为 KnY、KnM、KnS 时，只有 K4（16 位）和 K8（32 位）有效 n 操作数中 n=1~16（16 位），n=1~32（32 位） 														<ul style="list-style-type: none"> 标志信号：M1022 进位标志 Carry flag 			

指令说明

- ◆ **(D)**：欲循环的装置。**(n)**：一次循环的位数。
- ◆ 将 **(D)** 所指定的装置内容连同进位标志 M1022，一次向左循环 **(n)** 个位。

程序范例

- ◆ 当 X0 从 Off → On 变化时，D0 的 16 个位连同进位标志 M1022 共 17 个位以 4 个位一组往左循环，如下图所示标明※的位内容被传送至进位标志信号 M1022 里。



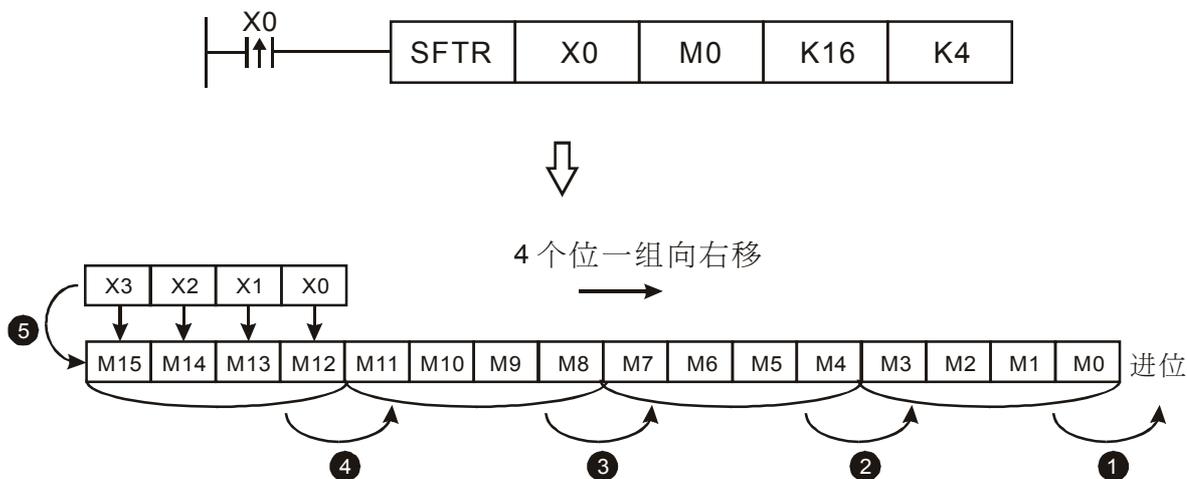
指令	操作数		功能														
SFTR	S	D n1 n2	位右移														
	位装置				字装置												16 位指令 (9 STEP) SFTR 连续执行型 32 位指令 - - - - • 标志信号: 无
	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F		
S	*	*	*	*													
D		*	*	*													
n ₁					*	*											
n ₂					*	*											
• 操作数使用注意: n ₁ 操作数中 n ₁ =1~1024 n ₂ 操作数中 n ₂ =1~n ₁																	

指令说明

- ◆ **S**: 移位装置的起始编号。 **D**: 欲移位装置的起始编号。 **n1**: 欲移位的数据长度。 **n2**: 一次移位的位数。
- ◆ 将 **D** 开始的起始编号, 具有 **n1** 个数位 (位移寄存器长度) 的位装置, 以 **n2** 位个数来右移。而 **S** 开始起始编号以 **n2** 位个数移入 **D** 中来填补位空位。

程序范例

- ◆ 在 X0 上升沿时, 由 M0~M15 组成 16 位, 以 4 位作右移。
- ◆ 扫描一次的位右移动作依照下列编号 1~5 动作。
 - ① M3~M0 → 进位
 - ② M7~M4 → M3~M0
 - ③ M11~M8 → M7~M4
 - ④ M15~M12 → M11~M8
 - ⑤ X3~X0 → M15~M12 完成



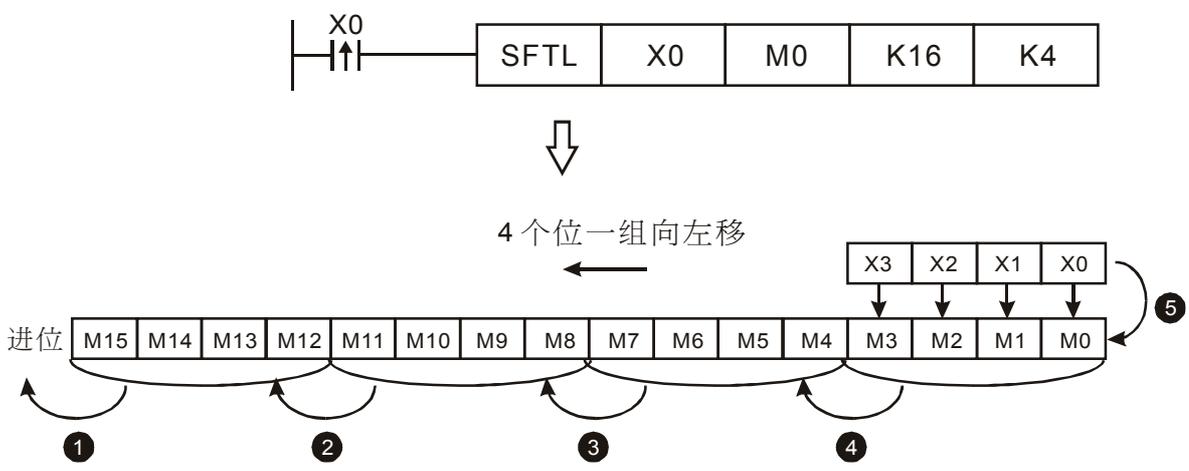
指令	操作数														功能	
SFTL	(S)	(D)	(n1)	(n2)	位左移											
	位装置				字装置										16 位指令 (9 STEP) SFTL 连续执行型	
	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E		F
S	*	*	*	*												
D		*	*	*												
n1					*	*										
n2					*	*										
• 操作数使用注意: n1 操作数中 n1=1~1024 n2 操作数中 n2=1~n1																
32 位指令 - - - - • 标志信号: 无																

指令说明

- ◆ **(S)**: 移位装置的起始编号。 **(D)**: 欲移位装置的起始编号。 **(n1)**: 欲移位的数据长度。 **(n2)**: 一次移位的位数。
- ◆ 将 **(D)** 开始的起始编号, 具有 **(n1)** 个数位 (位移寄存器长度) 的位装置, 以 **(n2)** 位个数来左移。而 **(S)** 开始起始编号以 **(n2)** 位个数移入 **(D)** 中来填补位空位。

程序范例

- ◆ 在 X0 上升沿时, 由 M0~M15 组成 16 位, 以 4 位作左移。
- ◆ 扫描一次的位左移动作依照下列编号 1~5 动作。
 - ① M15~M12 → 进位
 - ② M11~M8 → M15~M12
 - ③ M7~M4 → M11~M8
 - ④ M3~M0 → M7~M4
 - ⑤ X3~X0 → M3~M0 完成



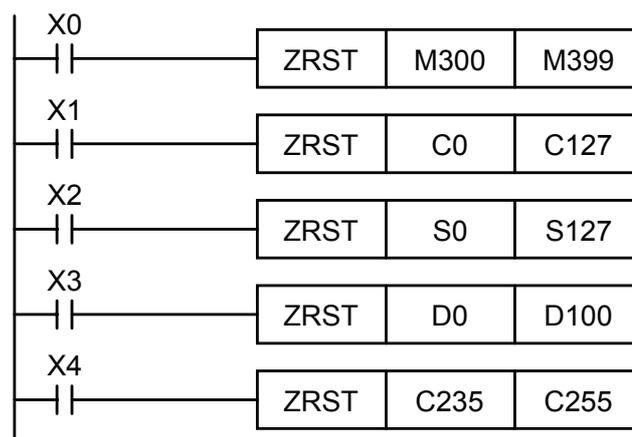
指令		操作数		功能													
ZRST		(D1)	(D2)	批次复位													
	位装置				字装置										16 位指令 (5 STEP)		
	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	ZRST	连续执行型
D ₁		*	*	*							*	*	*	*	*	32 位指令 - - - -	
D ₂		*	*	*							*	*	*	*	*		
<ul style="list-style-type: none"> 操作数使用注意：D₁操作数编号 ≤ D₂操作数编号 D₁、D₂操作数必须指定相同类型装置 																<ul style="list-style-type: none"> 标志信号：无 	

指令说明

- ◆ **(D1)**：批次复位起始装置。 **(D2)**：批次复位结束装置。
- ◆ 16 位计数器与 32 位计数器不可混在一起使用 ZRST 指令。
- ◆ 当 D1 操作数编号 > D2 操作数编号时，只有 D2 指定的操作数被清除。

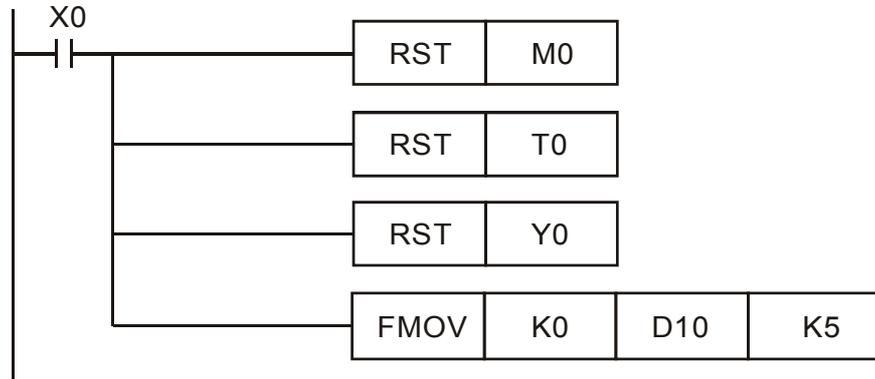
程序范例

- ◆ 当 X0 为 On 时，位装置辅助继电器 M300~M399 被清除成 Off。
- ◆ 当 X1 为 On 时，字装置 16 位计数器 C0~C127 全部清除。(写入 0，并将接点和线圈清除成 Off)。(字装置定时器 T 同理)
- ◆ 当 X2 为 On 时，状态 S0~S127 被清除成 Off。
- ◆ 当 X3 为 On 时，数据寄存器 D0~ D100 数据被清除为 0。
- ◆ 当 X4 为 On 时，字装置 32 位计数器 C235~C254 全部清除。(写入 0，并将接点和线圈清除成 Off)。



补充说明

- ◆ 装置可以单独使用清除指令(RST)，如位装置 Y、M、S 和字装置 T、C、D。
- ◆ 也可使用 FMOV 指令，将 K0 多点传送到字装置 T、C、D 或位寄存器 KnY、KnM、KnS 来达到清除的功能。



指令		操作数		功能												
SUM		D	S D	ON 位数量												
	位装置				字装置											16 位指令 (5 STEP) SUM 连续执行型
	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	
S					*	*	*	*	*	*	*	*	*	*	*	
D							*	*	*	*	*	*	*	*	*	
<ul style="list-style-type: none"> 操作数使用注意：S、D 操作数若使用 F 装置仅可使用 16 位指令 												32 位指令 (9 STEP) DSUM 连续执行型 <ul style="list-style-type: none"> 标志信号：M1020 零标志 Zero flag 				

指令说明

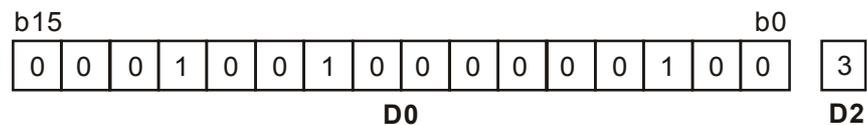
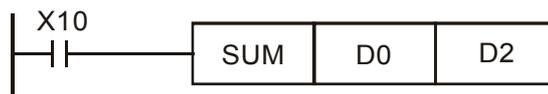
◆ **S**：来源装置。 **D**：存放计数值的目的地装置。

◆ 如果 16 个位全部为“0”时，零标志信号 M1020=On。

◆ 使用 32 位指令时，**D** 仍会占用 2 个寄存器。

◆ 当 X10 为 On 时，D0 的 16 个位中，内容为 “1” 的位总数被存于 D2 当中。

程序范例



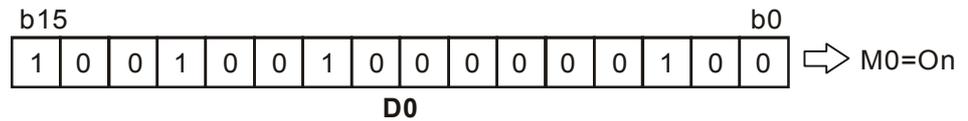
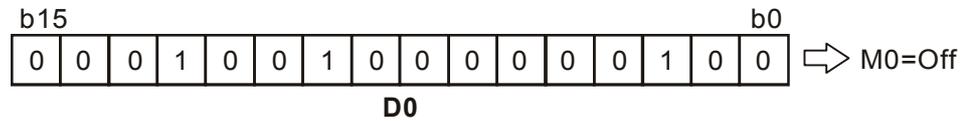
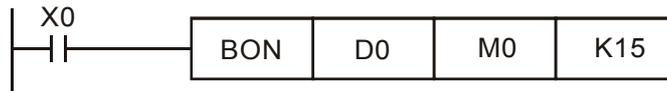
指令		操作数										功能								
BON		D	(S) (D) (n)										ON 位判定							
	位装置				字装置											16 位指令 (7 STEP) BON 连续执行型 <hr/> 32 位指令 (13 STEP) DBON 连续执行型 <hr/> • 标志信号: 无				
	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F					
S					*	*	*	*	*	*	*	*	*	*	*					
D		*	*	*																
n					*	*														
• 操作数使用注意: S 操作数若使用 F 装置仅可使用 16 位指令 n=0~15(16 位指令)。n=0~31(32 位指令)																				

指令说明

◆ **(S)**: 来源装置。 **(D)**: 存放判定结果的装置。 **(n)**: 指定判定的位。

程序范例

- ◆ 当 X0=On 时, 若是 D0 的第 15 个位为 "1" 时, M0=On, 为 "0" 时, M0=Off。
- ◆ X0 变成 Off 时, M0 仍保持的前的 On / Off 状态。

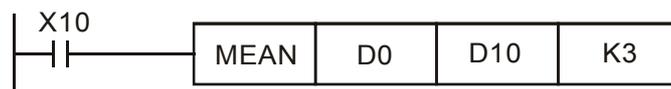


指令		操作数										功能								
MEAN		D	S	D	n											平均值				
	位装置					字装置										16 位指令 (7 STEP) MEAN 连续执行型 <hr/> 32 位指令 (13 STEP) DMEAN 连续执行型 <hr/> • 标志信号：无				
	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F					
S							*	*	*	*	*	*	*							
D								*	*	*	*	*	*	*	*					
n					*	*														
• 操作数使用注意：D 操作数若使用 F 装置仅可使用 16 位指令 n=1~64																				

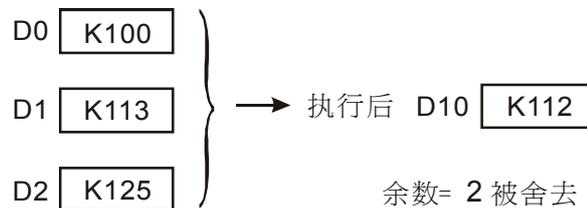
指令说明

- ◆ **S**：欲取平均值的起始装置。 **D**：存放平均值的装置。 **n**：取平均值的装置个数。
- ◆ 将 **S** 起始的 **n** 个装置内容值相加后取平均值存入 **D** 中。
- ◆ 如果计算中出现余数时，余数会被舍去。
- ◆ 如果指定的装置号码超过该装置可使用的正常范围时，只有正常范围内的装置编号被处理。
- ◆ n 如果是 1~64 以外的数值时，PLC 认定为“指令运算错误”。
- ◆ 当 X10=On 时， D0 开始算的 3 个(n=3)寄存器的内容全部相加，相加的后再除以 3 以求得平均值并存储于指定的 D10 当中，余数被舍去。。

程序范例



$(D0+D1+D2)/3 \rightarrow D10$



指令		操作数		功能														
REF		(D) (n)		I/O 状态即时刷新														
	位装置				字装置											16 位指令 (5 STEP)		
	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	REF	连续执行型	
D	*	*																
n					*	*												
<ul style="list-style-type: none"> 操作数使用注意：D 操作数必须指定 X0、X10、Y0、Y10...等最右边为 0 的编号，请参考下列补充说明 n 操作数范围 n=8~256，且为 8 的倍数 																	<ul style="list-style-type: none"> 标志信号：无 	

指令说明

- ◆ **(D)**：欲 I/O 状态即时刷新的起始装置。 **(n)**：I/O 状态即时刷新的数目。
- ◆ PLC 的输入/出端子的状态全部为程序扫描至 END 后，才作状态的更新，其中输入点的状态是在程序开始扫描时，自外部输入点的状态读入存在输入点存储器中，而输出端子在 END 指令后，才将输出点存储器内容送至输出装置。因此在演算过程中需要最新的输入/出数据，则可利用本指令。
- ◆ 一般 REF 指令可运用在 FOR~NEXT 指令的间、CJ 指令的间、输出入动作有中断处理时，或中断子程序。
- ◆ D 操作数必须指定 X0、X10、Y0、Y10...等最右边为 0 的编号。n 操作数范围 n=8~256，且为 8 的倍数，除此的外的数字多被视为错误。

程序范例 (一)

- ◆ 当 X0 为 On 时，X0~X7 的 8 点输入信号优先被读入，输入信号更新。
- ◆ 执行本指令时，PLC 会立即读取 X0~X7 的输入点状态，但在输入点上约 10ms 的输入延迟仍然存在。



程序范例 (二)

- ◆ 当 X0 为 On 时，Y0~Y7 的 8 点输出信号即时被送至输出端，输出信号更新。
- ◆ 执行本指令时，PLC 会立即送出 Y0~Y7 的输出点状态，但在输出点上约 10ms 的继电器输出延迟仍然存在。



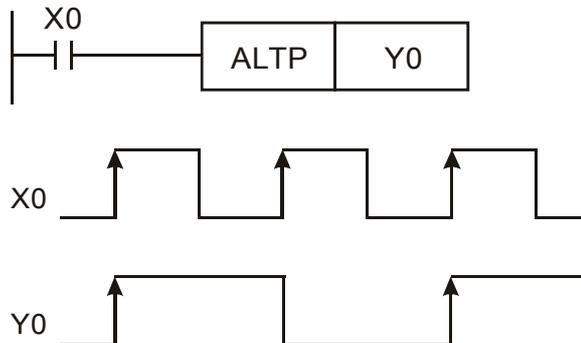
指令		操作数		功能												
ALT		D		ON/OFF 交替输出												
	位装置				字装置										16 位指令 (3 STEP)	
	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	ALT 连续执行型
D		*	*	*												32 位指令
																• 标志信号: 无

指令说明

◆ **D**: 目的地装置。

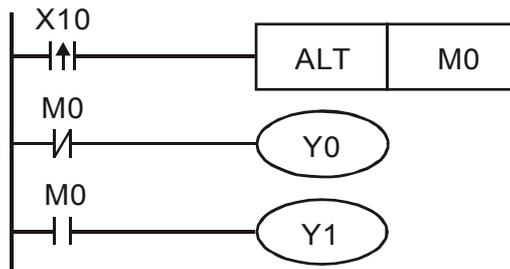
程序范例 (一)

◆ 当第一次 X0 从 Off→On 时, Y0= On。第二次 X0 从 Off→On 时, Y0=Off。



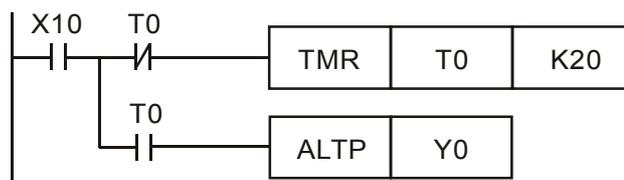
程序范例 (二)

◆ 使用单一开关控制启动与停止。一开始时, M0=Off 故 Y0= On、Y1=Off, 当 X10 作第一次 On /Off 时, M0= On 故 Y1= On、Y1=Off, 第二次 On /Off 时, M0=Off 故 Y0= On 而 Y1=Off。



程序范例 (三)

◆ 产生闪烁的动作。当 X10= On 时, T0 每隔 2 秒产生一个脉冲, Y0 输出会依 T0 脉冲产生 On /Off 交替。



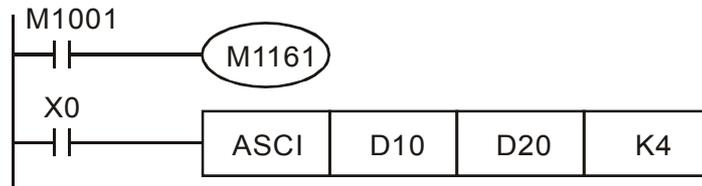
指令		操作数		功能															
ASCII		(S) (D) (n)		HEX 转为 ASCII															
	位装置				字装置										16 位指令 (7 STEP)				
	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	ASCII 连续执行型			
S					*	*	*	*	*	*	*	*	*	*	*	32 位指令			
D								*	*	*	*	*	*			- - - -			
n					*	*										标志信号: M1161 8/16 位模式切换			
<ul style="list-style-type: none"> 操作数使用注意: n 操作数指定范围 n=1~256 																			

指令说明

- ◆ **(S)**: 数据来源起始装置。 **(D)**: 存放变换结果的起始装置。 **(n)**: 变换的位数。
- ◆ 16 位转换模式: 当 M1161=Off 时, 将 **(S)** 的 16 进位数据, 将各个位数转换为 ASCII 码后, 传送到 **(D)** 的上 8 位和下 8 位中, 转换的位数以 **(n)** 来设置。
- ◆ 8 位转换模式: 当 M1161=On 时, 将 **(S)** 的 16 进位数据, 将各个位数转换为 ASCII 码后, 传送到 **(D)** 的下 8 位中, 转换的位数以 **(n)** 来设置。(**(D)** 的上 8 位全部为 0)

程序范例 (一)

- ◆ M1161=Off 时, 指定为 16 位转换模式。
- ◆ 当 X0=On 时, 将 D10 内的 4 个 16 进位数值转换成 ASCII 码传送到由 D20 起始的寄存器。



- ◆ 假设条件:

(D10) = 0ABC H	'0' = 30H	'1' = 31H	'5' = 35H
(D11) = 1234 H	'A' = 41H	'2' = 32H	'6' = 36H
(D12) = 5678 H	'B' = 42H	'3' = 33H	'7' = 37H
	'C' = 43H	'4' = 34H	'8' = 38H

D10=0ABC H															
0	0	0	0	1	0	1	0	1	0	1	1	1	1	0	0
0				A				B				C			

D20=B 的ASCII码 = 42H															
0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0
										4			2		

D21=C 的ASCII码 = 43H															
0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1
										4			3		

◆ 当 n = 4, 位的组成:

b15	D10= H 0123														b0	
0	0	0	0	0	0	0	0	1	0	0	1	0	0	0	1	1
0				1				2				3				

转换

b15	D20														b0	
0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0
										"0" → 30H						

b15	D21														b0	
0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	1
										"1" → 31H						

b15	D22														b0	
0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	1	0
										"2" → 32H						

b15	D23														b0	
0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	1	1
										"3" → 33H						

◆ 当 n = 1~16 时:

D \ n	K1	K2	K3	K4	K5	K6	K7	K8
D20	"3"	"2"	"1"	"0"	"7"	"6"	"5"	"4"
D21		"3"	"2"	"1"	"0"	"7"	"6"	"5"
D22			"3"	"2"	"1"	"0"	"7"	"6"
D23				"3"	"2"	"1"	"0"	"7"
D24					"3"	"2"	"1"	"0"
D25						"3"	"2"	"1"
D26							"3"	"2"
D27								"3"
D28								
D29								
D30								
D31								
D32								
D33								
D34								
D35								
D \ n	K9	K10	K11	K12	K13	K14	K15	K16
D20	"B"	"A"	"9"	"8"	"F"	"E"	"D"	"C"
D21	"4"	"B"	"A"	"9"	"8"	"F"	"E"	"D"
D22	"5"	"4"	"B"	"A"	"9"	"8"	"F"	"E"
D23	"6"	"5"	"4"	"B"	"A"	"9"	"8"	"F"
D24	"7"	"6"	"5"	"4"	"B"	"A"	"9"	"8"
D25	"0"	"7"	"6"	"5"	"4"	"B"	"A"	"9"
D26	"1"	"0"	"7"	"6"	"5"	"4"	"B"	"A"
D27	"2"	"1"	"0"	"7"	"6"	"5"	"4"	"B"
D28	"3"	"2"	"1"	"0"	"7"	"6"	"5"	"4"
D29		"3"	"2"	"1"	"0"	"7"	"6"	"5"
D30			"3"	"2"	"1"	"0"	"7"	"6"
D31				"3"	"2"	"1"	"0"	"7"
D32					"3"	"2"	"1"	"0"
D33						"3"	"2"	"1"
D34							"3"	"2"
D35								"3"

无变化

无变化

指令		操作数	功能												
HEX	P	(S) (D) (n)	ASCII 转为 HEX												
	位装置				字装置										16 位指令 (7 STEP) HEX 连续执行型 <hr/> 32 位指令 - - - - • 标志信号: M1161 8/16 位模式切换
	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	
S					*	*	*	*	*	*	*	*	*		
D								*	*	*	*	*	*	*	*
n					*	*									
• 操作数使用注意: n 操作数指定范围 n=1~256															

指令说明

- ◆ **(S)**: 数据来源起始装置。 **(D)**: 存放变换结果的起始装置。 **(n)**: 变换的 ASCII 码位数。
- ◆ 16 位转换模式: 当 M1161=Off 时, 指定为 16 位转换模式。**(S)** 的 16 进位数据上、下各 8 位的 ASCII 码转换为转换为 16 进位数值, 每 4 位数传送到 **(D)**, 转换的 ASCII 码位数以 **(n)** 来设置。
- ◆ 8 位转换模式: 当 M1161=On 时, 指定为 8 位转换模式。将 **(S)** 的 16 进位数据 将各个位数转换为 ASCII 码后, 传送到 **(D)** 的下 8 位中, 转换的位数以 **(n)** 来设置。(**(D)** 的上 8 位全部为 0)

程序范例 (一)

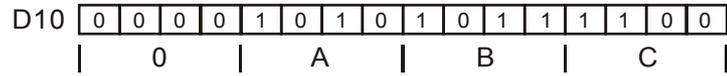
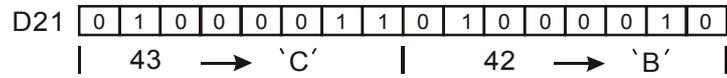
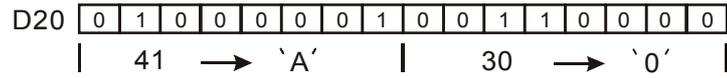
- ◆ 当 M1161=Off 时, 指定为 16 位转换模式。
- ◆ 当 X0=On 时, 将 D20 起始的寄存器中的 ASCII 码转换为 16 进位数值, 每 4 位数传送到 D10 起始的寄存器中, 转换的 ASCII 码位数 n=4。



◆ 假设条件:

(S)	ASCII 码	HEX 转换	(S)	HEX 转换	HEX 转换
D20 下	H 43	"C"	D24 下	H 34	"4"
D20 上	H 44	"D"	D24 上	H 35	"5"
D21 下	H 45	"E"	D25 下	H 36	"6"
D21 上	H 46	"F"	D25 上	H 37	"7"
D22 下	H 38	"8"	D26 下	H 30	"0"
D22 上	H 39	"9"	D26 上	H 31	"1"
D23 下	H 41	"A"	D27 下	H 32	"2"
D23 上	H 42	"B"	D27 上	H 33	"3"

◆ 当 n = 4 时，位的组成：



◆ 当 n = 1~16 时

n \ D	D13	D12	D11	D10	
1	此区不变化 全部为 0			. . . C H	
2				. . CD H	
3				. CDE H	
4				CDEF H	
5	此区不变化 全部为 0		. . . C H	DEF8 H	
6			. . CD H	EF89 H	
7			. CDE H	F89A H	
8			CDEF H	89AB H	
9	此区不变化 全部为 0		. . . C H	DEF8 H	9AB4 H
10			. . CD H	EF89 H	AB45 H
11			. CDE H	F89A H	B456 H
12			CDEF H	89AB H	4567 H
13	. . . C H	DEF8 H	9AB4 H	5670 H	
14	. . CD H	EF89 H	AB45 H	6701 H	
15	. CDE H	F89A H	B456 H	7012 H	
16	CDEF H	89AB H	4567 H	0123 H	

程序范例
(二)

◆ 当 M 1161 On 时，指定为 8 位转换模式。



◆ 当 n = 1~16 时

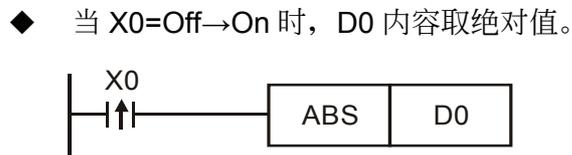
$\begin{matrix} \text{D} \\ n \end{matrix}$	D13	D12	D11	D10		
1	此区不变化 全部为 0			. . . C H		
2				. . CD H		
3				. CDE H		
4				CDEF H		
5			. . . C H	DEF8 H		
6			. . CD H	EF89 H		
7			. CDE H	F89A H		
8			CDEF H	89AB H		
9			. . . C H	DEF8 H	9AB4 H	
10			. . CD H	EF89 H	AB45 H	
11			. CDE H	F89A H	B456 H	
12			CDEF H	89AB H	4567 H	
13			. . . C H	DEF8 H	9AB4 H	5670 H
14			. . CD H	EF89 H	AB45 H	6701 H
15			. CDE H	F89A H	B456 H	7012 H
16			CDEF H	89AB H	4567 H	0123 H

指令		操作数		功能													
ABS		D	D	绝对值													
	位装置				字装置											16 位指令 (3 STEP)	
	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	ABS 连续执行型	
D					*	*	*	*	*	*	*	*	*	*	*	32 位指令 (5 STEP)	
																DABS 连续执行型	
																• 标志信号: 无	

指令说明

- ◆ **D**: 欲取绝对值的装置。
- ◆ 当 ABS 指令执行时, 被指定的组件 **D** 取绝对值。

程序范例



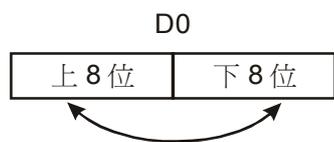
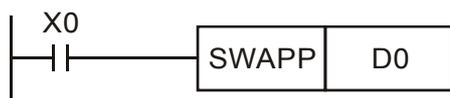
指令		操作数		功能												
SWAP		D	(S)	上下字节互换												
	位装置				字装置										16 位指令 (5 STEP)	
	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	SWAP 连续执行型
S							*	*	*	*	*	*	*	*	*	32 位指令 (9 STEP)
• 操作数使用注意：D 操作数若使用 F 装置仅可使用 16 位指令 • 标志信号：无																

指令说明

- ◆ **(S)**：欲执行上下字节互相交换的装置。
- ◆ 16 进位指令时，上位 8 位与下位 8 位的内容互相交换。
- ◆ 32 进位指令时，两个寄存器的上位 8 位与下位 8 位的内容各别互相交换。

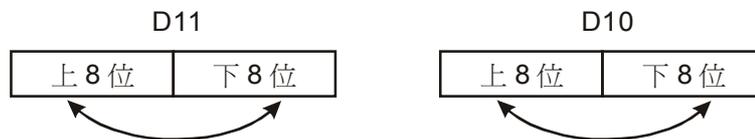
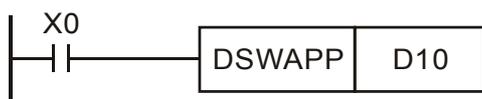
程序范例 (一)

- ◆ 当 X0=On 时，将 D0 的上位 8 位与下位 8 位的内容互相交换。



程序范例 (二)

- ◆ 当 X0=On 时，将 D11 的上位 8 位与下位 8 位的内容互相交换，D10 的上位 8 位与下位 8 位的内容互相交换。



指令		操作数		功能																
LD※		D	(S1) (S2)	接点型态比较 LD※																
	位装置				字装置											16 位指令 (5 STEP)				
	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	LD※	连续执行型	-	-	
S ₁					*	*	*	*	*	*	*	*	*	*	*		32 位指令 (9 STEP)			
S ₂					*	*	*	*	*	*	*	*	*	*	*		DLD※	连续执行型	-	-
<ul style="list-style-type: none"> 操作数使用注意：※：=、>、<、<>、≤、≥ 																<ul style="list-style-type: none"> 标志信号：无 				

指令说明

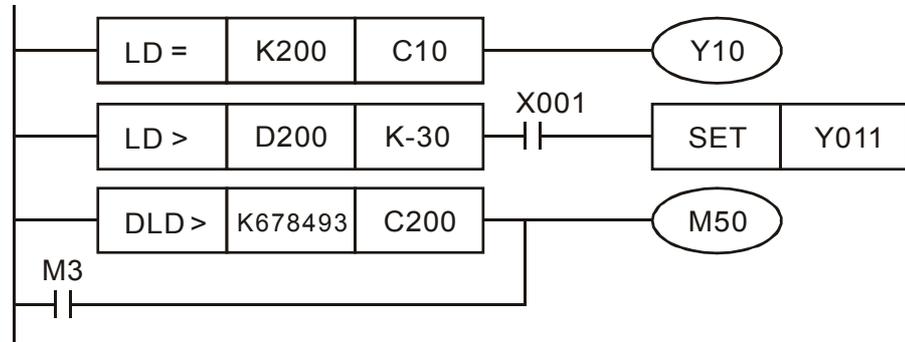
- ◆ **(S1)**：比较值 1。**(S2)**：比较值 2。
- ◆ **(S1)** 与 **(S2)** 的内容作比较的指令，比较结果为“等于”时，该指令导通，“不等于”时，该指令不导通。
- ◆ LD※的指令可直接与母线连接使用

16 位指令	32 位指令	导通条件	非导通条件
LD=	DLD=	(S1) = (S2)	(S1) ≠ (S2)
LD>	DLD>	(S1) > (S2)	(S1) ≤ (S2)
LD<	DLD<	(S1) < (S2)	(S1) ≥ (S2)
LD<>	DLD<>	(S1) ≠ (S2)	(S1) = (S2)
LD≤	DLD≤	(S1) ≤ (S2)	(S1) > (S2)
LD≥	DLD≥	(S1) ≥ (S2)	(S1) < (S2)

- ◆ **(S1)** 与 **(S2)** 的最左边位(16 位指令：b15、32 位指令：b31)为 1 时，该比较值被视为负值来比较。
- ◆ 32 位长度计数器(C200~)代入本指令作比较时，一定要使用 32 位指令(DLD※)，若是使用 16 位指令(LD※)时，CPU 判定为“程序错误”、主机面板上 ERROR 红色指示灯闪烁，CPU 不能 RUN。

程序范例

- ◆ C10 的内容等于 K200 时，Y10= On。
- ◆ 当 D200 的内容大于 -29，而且 X1= On 的时候 Y11= On 并保持住。
- ◆ C200 的内容小于 678,493 或者是 M3= On 的时候 M50= On。



指令		操作数		功能												
AND※		D	(S1) (S2)	接点型态比较 AND※												
	位装置				字装置											16 位指令 (5 STEP) AND※ 连续执行型 — —
	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	
S ₁					*	*	*	*	*	*	*	*	*	*	*	
S ₂					*	*	*	*	*	*	*	*	*	*	*	
<ul style="list-style-type: none"> 操作数使用注意：※：=、>、<、<>、≤、≥ 																32 位指令 (9 STEP) DAND※ 连续执行型 — —
<ul style="list-style-type: none"> 标志信号：无 																

指令说明

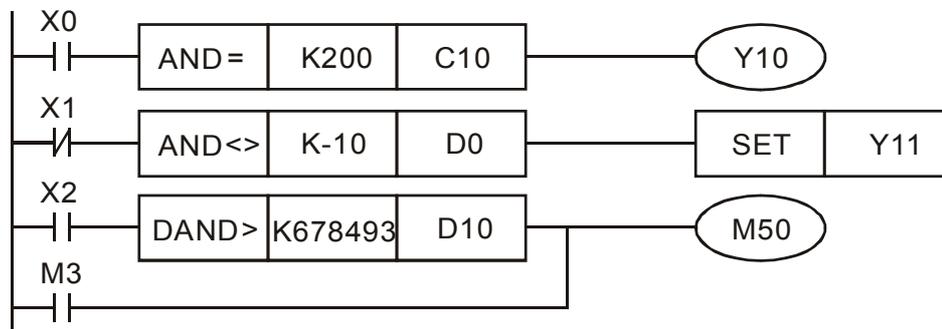
- ◆ **(S1)**：比较值 1。 **(S2)**：比较值 2。
- ◆ **(S1)** 与 **(S2)** 的内容作比较的指令，比较结果为等于时，该指令导通、不等于时，该指令不导通。而 AND※的指令是与接点串接的比较指令。

16 位指令	32 位指令	导通条件	非导通条件
AND=	DAND=	(S1) = (S2)	(S1) ≠ (S2)
AND>	DAND>	(S1) > (S2)	(S1) ≤ (S2)
AND<	DAND<	(S1) < (S2)	(S1) ≥ (S2)
AND<>	DAND<>	(S1) ≠ (S2)	(S1) = (S2)
AND≤	DAND≤	(S1) ≤ (S2)	(S1) > (S2)
AND≥	DAND≥	(S1) ≥ (S2)	(S1) < (S2)

- ◆ **(S1)** 与 **(S2)** 的最左边位(16 位指令：b15、32 位指令：b31)为 1 时，该比较值被视为负值来比较。
- ◆ 32 位长度计数器(C200~)代入本指令作比较时，一定要使用 32 位指令(DAND※)，若是使用 16 位指令(AND※)时，CPU 判定为“程序错误”、主机面板上 ERROR 红色指示灯闪烁，CPU 不能 RUN。

程序范例

- ◆ 当 X0= On 时, C10 的当前值又等于 K200 时, Y10= On。
- ◆ 当 X0=Off 而寄存器 D0 的内容又不等于 -10 的时候, Y11= On 并保持住。
- ◆ 当 X2= On 而且寄存器 D11、D0 的内容又小于 678,493 的时候, M50= On。



指令	D	操作数	功能													
OR※		(S1) (S2)	接点型态比较 OR※													
S ₁	位装置				字装置										16 位指令 (5 STEP) OR※ 连续执行型 - -	
	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E		F
S ₂					*	*	*	*	*	*	*	*	*	*	*	32 位指令 (9 STEP) DOR※ 连续执行型 - -
• 操作数使用注意：※：=、>、<、<>、≤、≥															• 标志信号：无	

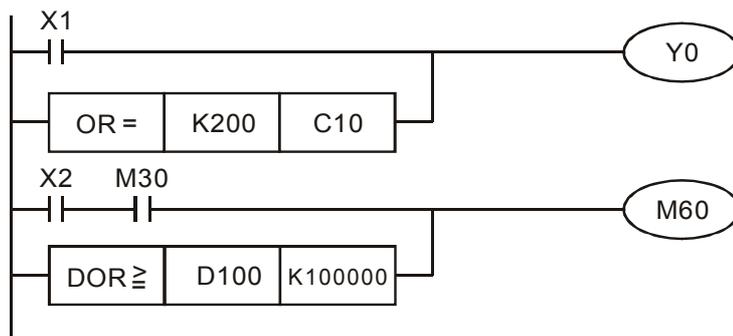
指令说明

- ◆ **(S1)**：比较值 1。 **(S2)**：比较值 2。
- ◆ **(S1)** 与 **(S2)** 的内容作比较的指令，比较结果为等于时，该指令导通、不等于时，该指令不导通。而 AND※的指令是与接点并接的比较指令。

16 位指令	32 位指令	导通条件	非导通条件
OR=	DOR=	(S1) = (S2)	(S1) ≠ (S2)
OR>	DOR>	(S1) > (S2)	(S1) ≤ (S2)
OR<	DOR<	(S1) < (S2)	(S1) ≥ (S2)
OR<>	DOR<>	(S1) ≠ (S2)	(S1) = (S2)
OR≤	DOR≤	(S1) ≤ (S2)	(S1) > (S2)
OR≥	DOR≥	(S1) ≥ (S2)	(S1) < (S2)

- ◆ **(S1)** 与 **(S2)** 的最左边位(16 位指令：b15、32 位指令：b31)为 1 时，该比较值被视为负值来比较。
- ◆ 32 位长度计数器(C200~)代入本指令作比较时，一定要使用 32 位指令(DOR※)，若是使用 16 位指令(OR※)时，CPU 判定为“程序错误”、主机面板上 ERROR 红色指示灯闪烁，CPU 不能 RUN。
- ◆ 当 X1= On 时，或者是 C10 的当前值等于 K200 时，Y0= On。
- ◆ 当 X2 和 M30 都等于 On 的时候，或者是 32 位寄存器 D101、D100 的内容大于或等于 K100,000 时，M60= On。

程序范例



(此页有意留为空白)