第二部分

EasyProg 软件手册

第一章　欢迎使用 EasyProg

EasyProg 是 KDN-K3 系列小型一体化 PLC 的上位编程软件，编程环境完全符合 IEC61131-3 标准，是一套功能强大、使用方便、高效的开发系统。

IEC61131-3 是 IEC 为工业自动化编程制定的标准，是在吸收不同厂家编程语言风格、方言及适应未来软件技术发展要求制定的，独立于任何一家公司，适合不同领域、不同类编程人员习惯。自发布以来得到所有顶尖 PLC 厂家的认可，各厂家的编程软件也在尽量向 IEC 标准靠拢。

EasyProg 完全是自主研发，我们采用了符合 IEC61131-3 标准的方案，由于许多用户通过各种渠道已经掌握了大部分编程技巧，将使得 EasyProg 简便、易学，使用起来将会得心应手。

EasyProg 软件具有如下特点：

- 符合 IEC61131-3 标准
- 支持 IL（指令表）和 LD（梯形图）两种标准语言
- 丰富的指令集，内置 IEC61131-3 定义的标准功能、功能块以及一些特殊的应用指令
- 支持中断服务程序
- 支持用户自定义功能块（子程序）
- 允许在程序中选择显示绝对地址或者是其符号变量名称，便于工程的实施、维护
- 灵活的硬件配置方式，最大限度地允许用户自定义各种硬件参数
- 完善的联机功能，包括下载、上载、在线监测、强制、读写实时时钟等
- 定义了完善的快捷键、右键菜单，方便用户的使用
- 对用户的错误操作尽可能地予以屏蔽、提示，体贴用户的操作

第二章 EasyProg 安装及运行

本章对 EasyProg 的安装以及运行环境进行了详细的描述，可以帮助用户迈出使用 EasyProg 的第一步。若用户熟悉 Windows 下软件的使用，则可以跳过本章。

2.1 系统需求

EasyProg 软件对于计算机的要求并不高，此处做如下推荐：

2.1.1 硬件需求

- CPU：Pentium133MHz 或以上
- 硬盘：10M 以上空间
- 内存：64M 以上
- 键盘、鼠标、串行通信口（com 口）或者 USB 口（需另外使用 USB/RS232 转换器）
- 17"以上彩色显示器，分辨率 1024*768

2.1.2 软件需求

Windows 9x/Me/NT4.0（sp4 以上）/2000/XP 简体中文版操作系统

2.2 安装与卸载

2.2.1 安装步骤详解

注意：在安装过程的任何一步单击〔取消〕，都可以终止安装。

① 运行安装盘中的 EasyProgVxxxx_setup.exe（xxxx 代表版本号，比如 1004），弹出安装向

导的首页，如图 2-1：

图 2-1

② 单击〔下一步〕，将确认 EasyProg 的安装路径。用户可用使用默认路径，也可以对其进行修改，如图 2-2。

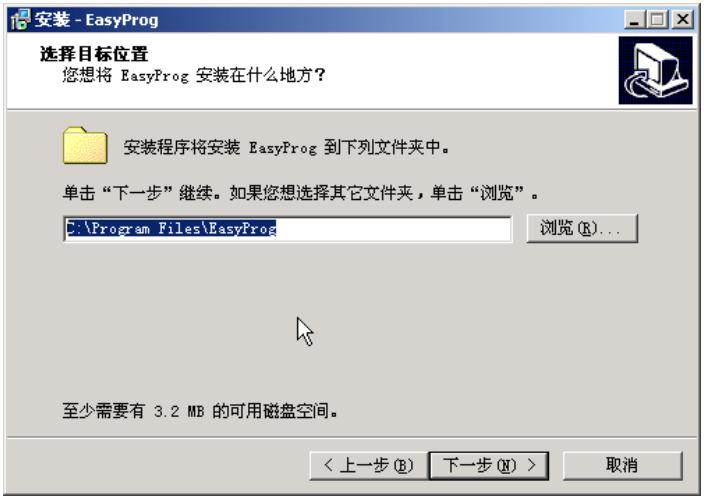


图 2-2

③ 单击〔下一步〕，将确认在【开始】菜单中生成的快捷方式文件夹名称，默认为“KDN”。如图 2-3：

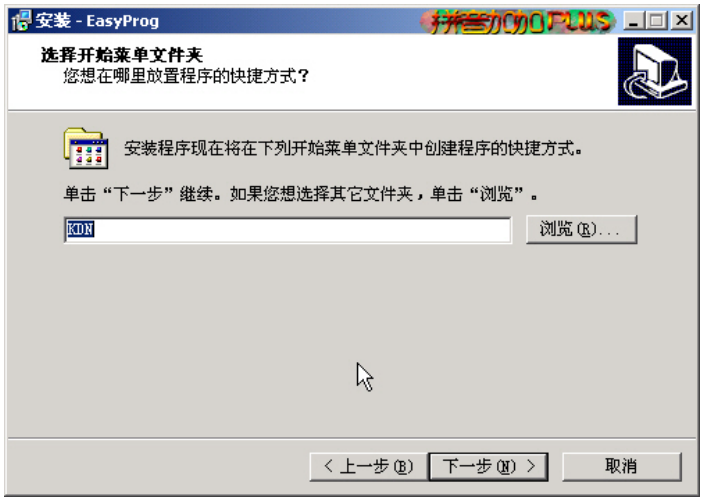


图 2-3

④ 单击〔下一步〕，将确认是否在桌面、运行栏中生成快捷方式。如图 2-4：

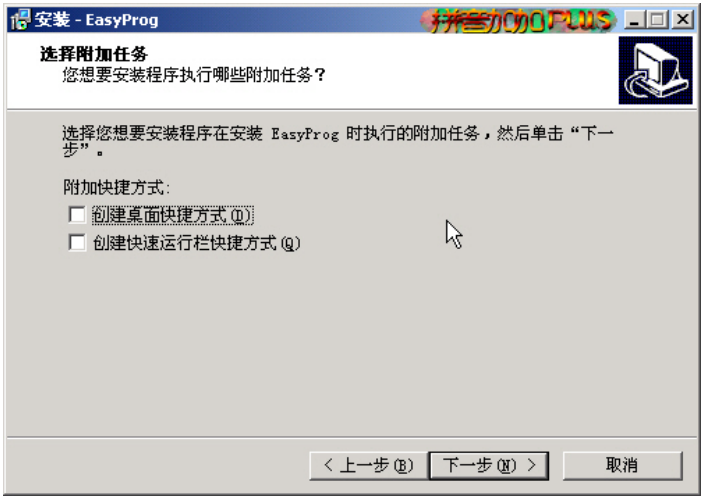


图 2-4

⑤ 用户选择完是否在桌面、运行栏中创建快捷方式后，单击〔下一步〕，将提示确认准备开始安装。如图 2-5：

图 2-5

⑥ 单击〔安装〕，将正式开始安装过程，安装结束后的提示如图 2-6：

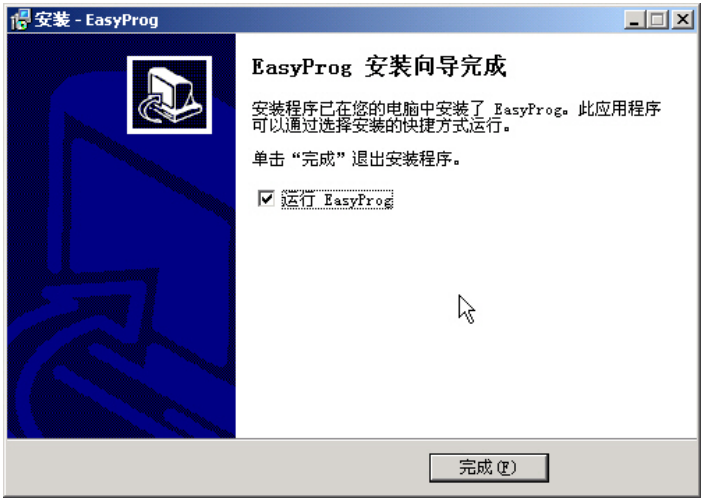


图 2-6

⑦ 单击〔完成〕，将结束安装过程。

用户若同时选中了〔运行 EasyProg〕，则 EasyProg 会立即启动。

2.2.2　卸载步骤详解

注意：执行卸载前，请先退出 EasyProg 程序。

卸载 EasyProg 软件共有两种方法：

· 方法一

① 执行【开始】|【程序】中 EasyProg 快捷方式组中的【卸载 EasyProg】命令，将开始卸载过程。如图 2-7 所示。

图 2-7

② 单击〔是〕，卸载程序将会干净地将 EasyProg 软件从计算机中卸载，最后提示如图 2-8。

图 2-8

· 方法二

单击【开始】|【设置】|【控制面板】，进入控制面板，执行其中的【添加或删除程序】命令，继而在弹出的“添加/删除程序”对话框中选择“EasyProg Vx.x.x.x”（x.x.x.x 代表版本号，如 1.0.0.4），然后单击〔更改/删除〕按钮，余下的操作过程如同方法一。

2.3 启动和退出 EasyProg

2.3.1 启动 EasyProg

您可以通过下面两种简单的方法启动 EasyProg：

- 执行【开始】|【程序】中 EasyProg 快捷方式组中的【EasyProg】命令。
- 若您在安装时选中了“在桌面上创建快捷方式”，则单击桌面上的  图标即可。

2.3.2 退出 EasyProg

启动 EasyProg 后，有三种方式可以退出软件：

- 执行【文件】|【退出】菜单命令
- 使用 Alt+F4 快捷键
- 单击 EasyProg 主窗口右上角的  图标。

第三章 EasyProg 编程基础

本章详细描述了使用 EasyProg 软件针对 KDN-K3 系列 PLC 编程的基础知识，同时也介绍了 IEC61131-3 标准中的一些基本概念，这些概念对于用户使用任何一种 IEC61131-3 软件都是非常有用的。本章的目的是帮助用户开始编程的学习和实践，达到“知其然并知其所以然”的程度。

在初次阅读时，并不需要用户对每个环节都理解得非常透彻，但建议用户采取“边阅读，边使用”的方式，这样会更有助于对本章内容的理解。这也是推荐用户阅读后续章节的方式。

3.1 程序组织单元（POU）

IEC61131-3 引入 POU 的概念，POU 就是构成项目的基本程序单元。传统的 PLC 制造商在编程方面为各自的 PLC 定义了各种类型的块，IEC61131-3 将这些块的种类减少为 3 种统一的基本类型。下面描述了标准的 POU 类型。

- 程序（Programme）

  关键字：PROGRAMME。

  “程序”用于执行一定的任务，无返回值。

  在所有的 POU 中，只有“程序”才能作为“任务”在 CPU 中得以运行。

- 功能（Function）

  关键字：FUNCTION 。

  “功能”可有输入参数，只有一个返回值，其返回值是通过功能名带回的。当输入参数相同时，返回值总是相同的。“功能”主要用于代码重用，可被其它 POU 调用。

- 功能块（Function Block）

关键字：FUNCTION_BLOCK。

“功能块”简称 FB。可有输入、输出参数，并具有静态变量（即能够记忆 FB 以前的状态）。FB 的输出值通过其输出参数传递。FB 的输出不仅取决于其输入值，而且也取决于在其静态变量中储存的状态值。FB 也主要用于代码重用，可被其它 POU 调用。

3.2 数据类型

数据类型定义了数据的位长度、取值范围及其初始化值。

在 IEC61131-3 中定义了一组最常用的基本数据类型，因此在 PLC 领域内这些数据类型的含义以及使用方式是开放、统一的。

这些基本数据类型按照其使用目的的不同又可以划分为两类：位串型和数值型。位串型的数据，可以形象地描述为由若干个二进制位构成的串，其值可以用于逻辑运算，比如移位、按位与、按位或等；数值型的数据可以用于数学计算，比如四则运算、三角函数、求对数等。

目前 K3 系列 PLC 支持的标准数据类型如表 3-1 所示。

分类	数据类型	描述	长度（位）	取值范围	缺省初始值
布尔/位串型	BOOL	布尔型	1	true, false	false
	BYTE	8 位位串	8	〔0, FF〕	0
	WORD	16 位位串	16	〔0, FFFF〕	0
	DWORD	32 位位串	32	〔0, FFFFFFFF〕	0
数值型	INT	整型，有符号	16	〔-2^{15}, 2^{15}-1〕	0
	DINT	双整型，有符号	32	〔-2^{31}, 2^{31}-1〕	0
	REAL	实型	32	采用 ANSI/IEEE754-1985 标准；约〔$1.18*10^{-38}$， $3.40*10^{38}$〕，〔$-3.40*10^{38}$，$-1.18*10^{-38}$〕	0.0

表 3-1 K3 系列 PLC 支持的数据类型

3.3 常量

在程序运行的过程中，其值不能改变的量称为常量。常量也要区分为不同的数据类型，并且根据数据类型的不同，常量的书写格式各不相同。

表 3-2 列出了 EasyProg 中各种类型的常量的定义。

分类	数据类型	常量格式(1)	描 述	示例
布尔/位串型	BOOL	**true、false**	true 代表真，false 代表假	false
	BYTE	**B#**x	x：十进制数字，范围〔0, 255〕	B#129
		B#2#x	x：二进制数字，范围[0, 11111111]	B#2#10010110
		B#8#x	x：八进制数字，范围〔0, 377〕	B#8#173
		B#16#x	x：十六进制数字，范围〔0, FF〕	B#16#3E
	WORD	**W#**x	x：十进制数字，范围〔0, 65535〕	W#39675
		2#x	x：二进制数字， 范围〔0, 1111111111111111〕	2#100110011
		W#2#x		W#2#110011
		8#x	x：八进制数字，范围〔0, 177777〕	8#7432
		W#8#x		8#174732
		16#x	x：十六进制数字，范围〔0, FFFF〕	16#6A7D
		W#16#x		W#16#9BFE
	DWORD	**DW#**x	x：十进制数字，范围〔0, 2^{32}-1〕	DW#547321
		DW#2#x	x：二进制数字，范围 〔0, 11111111111111111111111111111111〕	DW#2#10111
		DW#8#x	x：八进制数字，范围〔0, 37777777777〕	DW#8#76543
		DW#16#x	x：十六进制数字，范围〔0, FFFFFFFF〕	DW#16#FF7D
数值型	INT	x	x：十进制数字，范围〔 – 32768, 32767〕	12345
		I#x		I# – 2345
		I#2#x	x：二进制数字(2)	I#2#1111110
		I#8#x	x：八进制数字(2)	I#8#16732

		I#16#x	x：十六进制数字(2)	I#16#7FFF
	DINT	**DI#**x	x：十进制数字，范围〔-2^{31}, $2^{31}-1$〕	DI#8976540
		DI#2#x	x：二进制数字(2)	DI#2#101111
		DI#8#x	x：八进制数字(2)	DI#8#126732
		DI#16#x	x：十六进制数字(2)	DI#16#2A7FF
	REAL	带小数点的十进制数字		1.0，－243.456
		x**E**y	x：带小数点的十进制数字， 范围约〔1.18，3.40〕,〔－3.40，－1.18〕 y：整数，范围〔－38，38〕	-2.3E-23

表 3-2 常量的定义

注：

(1) 在 IEC61131-3 中，各种标识符的使用是大小写无关的。因此在程序中将格式字符写为大写或者小写均合法，比如 W#234, dw#12345 均为合法常量。

具体请参见下文关于标识符的定义部分。

(2) INT、DINT 型常量的二进制、八进制、十六进制表示方法均采用了通用计算机中标准的补码表示法，其最高有效位（MSB）是符号位：MSB 为 1 则代表负数，MSB 为 0 则代表正数。比如 I#16#FFFF = －1，I#7FFF =32767，I#8000 = －32768 等。

具体请参阅相关的计算机基础类书籍。

3.4 标识符

标识符是由数字、字母等字符组成的字符串。编程人员可以使用标识符来为变量、程序等指定名称。

3.4.1 标识符的定义

标识符的定义和使用必须依据如下原则：

- 必须以一个字母或者一个单一的下划线字符开始，随后是一定数量的数字、字母或者下划线。

- 标识符是大小写无关的。比如，abc、ABC、aBC 是同一个标识符。
- 标识符的长度仅受各编程系统的限制。在 EasyProg 中，标识符的最大长度是 16 个字符。
- 关键字不允许用于用户自定义的标识符。

  关键字是标准的标识符，其拼写形式和使用目的均由 IEC61131-3 明确规定。

  IEC61131-3 定义的保留关键字请参阅附录二。

3.4.2 标识符的使用

下面列出了在 EasyProg 中可使用标识符的语言元素：

- 程序、功能、功能块名称
- 变量
- 跳转标号等

3.5 变量

在程序的运行过程中，其值可以改变的量称为变量。变量用于初始化、存储和处理用户数据，每个变量都有其固定的数据类型。在 IEC61131-3 中，变量的存储位置可以由用户自行指定一个有效的 PLC 内存地址，也可以由编程系统自行分配。

变量的使用必须遵循“先定义，后使用”的原则。请参阅 3.4.1 标识符的定义 部分。

3.5.1 变量类型

在 IEC61131-3 中变量具有不同的形式。它们可以定义为一个 POU（Programme Orgnization Unit，程序组织单元）的形式参数；也可以在一个 POU 内定义，仅作为本 POU 的局部变量；也可以在 POU 外定义，而且在整个工程范围内使用。

表 3-3 中详细描述了 IEC61131-3 中定义的各种变量类型。

变量类型	存储权限		描述
	外部	内部	
VAR	---	读写	局部变量。仅可在定义它的 POU 内使用。
VAR_INPUT	写	读	输入变量。在定义它的 POU 内作为 POU 的输入参数仅可

			读，在调用此 POU 时此变量仅可写。
VAR_OUTPUT	读	读写	输出变量。在定义它的 POU 内作为 POU 的输出参数可读写，在调用此 POU 时此变量仅可读。
VAR_IN_OUT	读写	读写	输入/输出变量，是 VAR_INPUT 和 VAR_OUTPUT 的组合类型。在定义它的 POU 内以及在调用此 POU 时此变量可读写。
VAR_GLOBAL	读写	读写	全局变量。可被所有的 POU 直接读写。
VAR_EXTERNAL	读写	读写	外部变量。若全局变量是在某 POU 内定义的，则若要在该 POU 外访问该变量，就必须在访问之前再次将其声明为外部变量。
VAR_ACCESS	读写	读写	关于配置（configuration）的全局变量。作为各资源之间的联系通道。

表 3-3 IEC61131-3 中的变量类型

EasyProg 目前支持 VAR、VAR_INPUT、VAR_OUTPUT、VAR_IN_OUT、VAR_GLOBAL 五种变量类型。

3.5.2 EasyProg 中变量类型的使用

在 EasyProg 中，各种变量的定义均在相应的表格中进行，这样既避免了让用户进行繁琐的输入，同时软件还能够对用户的输入进行严格的语法检查。

全局变量的定义在全局变量表中完成。

POU 的局部变量、形式参数在该 POU 编辑界面的变量定义表格中完成。

若全局变量与局部变量的名称相同，则在程序中局部变量的使用优先。

具体的使用方法请参见本手册中关于界面的详细介绍部分。

3.5.3 变量的检验

在编辑、编译程序时，EasyProg 可以自动对变量的使用情况进行检验，即判别该变量是否按照其变量类型、数据类型进行处理。这也是 IEC61131-3 的重要优点之一。

比如，在程序中为一个 WORD 类型的变量赋一个 BOOL 类型的值，或者对一个 VAR_INPUT

型的变量进行赋值操作，EasyProg 就会向用户进行错误警告并提示修改。

由于变量的特性取决于其变量类型和数据类型，因此这种检验能够在很大程度上避免由于变量使用而引起的错误。

3.6 K3 系列 PLC 内存区域分配

3.6.1 基本内存区域类型及其特性

在 KDN-K3 系列 PLC 中，CPU 的内存被划分为不同类型的几个区域，各区域的使用目的不同，并且各有自己的特性。具体如表 3-4 所示。

I	
描述	DI（开关量输入）映像区
访问方式	可按位、字节、字、双字访问
存储权限	可读
其它	允许强制，不能掉电保持
Q	
描述	DO（开关量输出）映像区
访问方式	可按位、字节、字、双字访问
存储权限	可读、可写
其它	允许强制，不能掉电保持
AI	
描述	AI（模拟量输入）映像区
访问方式	可按字访问
存储权限	可读
其它	允许强制，不能掉电保持
AQ	
描述	AO（模拟量输出）映像区

访问方式	可按字访问
存储权限	可写
其它	允许强制，不能掉电保持
V	
描述	变量存储区。该区域比较大，可用于存储大量的数据。
访问方式	可按位、字节、字、双字访问
存储权限	可读可写，不能掉电保持
其它	允许强制，允许掉电保持
M	
描述	中间继电器区。用于存储控制继电器的中间状态或者其它数据。 与 V 区相比，M 区的访问速度更快，更有利于位操作。
访问方式	可按位、字节、字、双字访问
存储权限	可读可写
其它	允许强制，允许掉电保持
SM	
描述	系统存储区。存储着系统当前的各种状态信息，并且用户可以利用 SM 区的某些位来选择和控制 CPU 的某些特殊功能。
访问方式	可按位、字节、字、双字访问
存储权限	可读可写
其它	不允许强制，不能掉电保持
L	
描述	局部变量区。程序中定义的所有局部变量、POU 的输入/输出参数均在 L 区内自动分配地址。不建议用户直接操作 L 区。
访问方式	可按位、字节、字、双字访问
存储权限	可读可写
其它	不允许强制，不能掉电保持

表 3-4 K3 系列 PLC 的内存区域分配

3.6.2 基本内存区域的直接寻址方式

K3 系列 CPU 的每个内存单元都有其明确、唯一的地址。用户可以通过直接寻址方式，即直接使用某内存单元的地址，来访问各内存区域。

3.6.2.1 直接地址表示格式

IEC61131-3 规定，所有的直接地址都必须以 “%”开始。

各内存区域的直接寻址方式如下述列表。表中的 x、y 均代表十进制数字。

- **I 区**

位寻址	格式	**%I**x.y
	描述	x：字节地址，即在 I 区中该内存单元所在的字节的编号（地址）。 y：位地址，表示位于该字节的第几位。范围〔0,7〕。
	数据类型	BOOL
	示例	%I0.0 %I0.7 %I5.6
字节寻址	格式	**%IB**x
	描述	x：字节地址，即在 I 区中该内存单元所在的字节的编号（地址）。
	数据类型	BYTE
	示例	%IB0 %IB1 %IB10
字寻址	格式	**%IW**x
	描述	x：字节地址，即在 I 区中该内存单元首字节的地址。 由于 WORD 类型的数据长度为 2 字节，因此 x 必须为偶数。
	数据类型	WORD、INT
	示例	%IW0 %IW2 %IW12
双字寻址	格式	**%ID**x
	描述	x：字节地址，即在 I 区中该内存单元首字节的地址。 由于 DWORD 类型的数据长度为 4 字节，因此 x 必须为偶数。
	数据类型	DWORD、DINT

	示例	%ID0 %ID4 %ID12

表 3-5 I 区地址格式

- **Q 区**

位寻址	格式	**%Q**x.y
	描述	x：字节地址，即在 Q 区中该内存单元所在的字节的编号（地址）。 y：位地址，表示位于该字节的第几位。范围〔0,7〕。
	数据类型	BOOL
	示例	%Q0.0 %Q0.7 %Q5.6
字节寻址	格式	**%QB**x
	描述	x：字节地址，即在 Q 区中该内存单元所在的字节的编号（地址）。
	数据类型	BYTE
	示例	%QB0 %QB1 %QB10
字寻址	格式	**%QW**x
	描述	x：字节地址，即在 Q 区中该内存单元首字节的地址。 由于 WORD 类型的数据长度为 2 字节，因此 x 必须为偶数。
	数据类型	WORD、INT
	示例	%QW0 %QW2 %QW12
双字寻址	格式	**%QD**x
	描述	x：字节地址，即在 Q 区中该内存单元首字节的地址。 由于 DWORD 类型的数据长度为 4 字节，因此 x 必须为偶数。
	数据类型	DWORD、DINT
	示例	%QD0 %QD4 %QD12

表 3-6 Q 区地址格式

- **M 区**

位寻址	格式	**%M**x.y
	描述	x：字节地址，即在 M 区中该内存单元所在的字节的编号（地址）。 y：位地址，表示位于该字节的第几位。范围〔0,7〕。

	数据类型	BOOL
	示例	%M0.0　%M0.7　%M5.6
字节寻址	格式	**%MB**x
	描述	x：字节地址，即在 M 区中该内存单元所在的字节的编号（地址）。
	数据类型	BYTE
	示例	%MB0　%MB1　%MB10
字寻址	格式	**%MW**x
	描述	x：字节地址，即在 M 区中该内存单元首字节的地址。 由于字数据长度为 2 字节，因此 x 必须为偶数。
	数据类型	WORD、INT
	示例	%MW0　%MW2　%MW12
双字寻址	格式	**%MD**x
	描述	x：字节地址，即在 M 区中该内存单元首字节的地址。 由于双字数据长度为 4 字节，因此 x 必须为偶数。
	数据类型	DWORD、DINT
	示例	%MD0　%MD4　%MD12

表 3-7　M 区地址格式

- **V 区**

位寻址	格式	**%V**x.y
	描述	x：字节地址，即在 V 区中该内存单元所在的字节的编号（地址）。 y：位地址，表示位于该字节的第几位。范围〔0,7〕。
	数据类型	BOOL
	示例	%V0.0　%V0.7　%V5.6
字节寻址	格式	**%VB**x
	描述	x：字节地址，即在 V 区中该内存单元所在的字节的编号（地址）。
	数据类型	BYTE
	示例	%VB0　%VB1　%VB10

字寻址	格式	**%VW**x
	描述	x：字节地址，即在 V 区中该内存单元首字节的地址。 由于字数据长度为 2 字节，因此 x 必须为偶数。
	数据类型	WORD、INT
	示例	%VW0　%VW2　%VW12
双字寻址	格式	**%VD**x
	描述	x：字节地址，即在 V 区中该内存单元首字节的地址。 由于双字数据长度为 4 字节，因此 x 必须为偶数。
	数据类型	DWORD、DINT 另外，V 区的最后 256 字节（VD3840—VD4092）可作 REAL 型。
	示例	%VD0　%VD4　%VD12

表 3-8　V 区地址格式

- **SM 区**

位寻址	格式	**%SM**x.y
	描述	x：字节地址，即在 SM 区中该内存单元所在的字节的编号（地址）。 y：位地址，表示位于该字节的第几位。范围〔0,7〕。
	数据类型	BOOL
	示例	%SM0.0　%SM0.7　%SM5.6
字节寻址	格式	**%SMB**x
	描述	x：字节地址，即在 SM 区中该内存单元所在的字节的编号（地址）。
	数据类型	BYTE
	示例	%SMB0　%SMB1　%SMB10
字寻址	格式	**%SMW**x
	描述	x：字节地址，即在 SM 区中该内存单元首字节的地址。 由于字数据长度为 2 字节，因此 x 必须为偶数。
	数据类型	WORD、INT
	示例	%SMW0　%SMW2　%SMW12
双字寻址	格式	**%SMD**x

	描述	x：字节地址，即在 SM 区中该内存单元首字节的地址。 由于双字数据长度为 4 字节，因此 x 必须为偶数。
	数据类型	DWORD、DINT
	示例	%SMD0 %SMD4 %SMD12

表 3-9 SM 区地址格式

- **L 区（注意：不建议用户使用直接地址来访问 L 区）**

位寻址	格式	**%L**x.y
	描述	x：字节地址，即在 L 区中该内存单元所在的字节的编号（地址）。 y：位地址，表示位于该字节的第几位。范围〔0,7〕。
	数据类型	BOOL
	示例	%L0.0 %L0.7 %L5.6
字节寻址	格式	**%LB**x
	描述	x：字节地址，即在 L 区中该内存单元所在的字节的编号（地址）。
	数据类型	BYTE
	示例	%LB0 %LB1 %LB10
字寻址	格式	**%LW**x
	描述	x：字节地址，即在 L 区中该内存单元首字节的地址。 由于字数据长度为 2 字节，因此 x 必须为偶数。
	数据类型	WORD、INT
	示例	%LW0 %LW2 %LW12
双字寻址	格式	**%LD**x
	描述	x：字节地址，即在 L 区中该内存单元首字节的地址。 由于双字数据长度为 4 字节，因此 x 必须为偶数。
	数据类型	DWORD、DINT、REAL
	示例	%LD0 %LD4 %LD12

表 3-10 L 区地址格式

- **AI 区**

字寻址	格式	**%AIW**x
	描述	x：字节地址，即在 AI 区中该内存单元首字节的地址。 由于字数据长度为 2 字节，因此 x 必须为偶数。
	数据类型	INT
	示例	%AIW0 %AIW2 %AIW12

表 3-11 AI 区地址格式

- **AQ 区**

字寻址	格式	**%AQW**x
	描述	x：字节地址，即在 AQ 区中该内存单元首字节的地址。 由于字数据长度为 2 字节，因此 x 必须为偶数。
	数据类型	INT
	示例	%AQW0 %AQW2 %AQW12

表 3-12 AQ 区地址格式

3.6.2.2 直接地址与内存单元之间的映射

每一个合法的直接地址都对应于 CPU 中的一个内存单元。在程序中对直接地址的操作就是对其对应的内存单元进行操作。下面将以 V 区为例图解直接地址与内存单元之间的映射关系。

- **位地址**

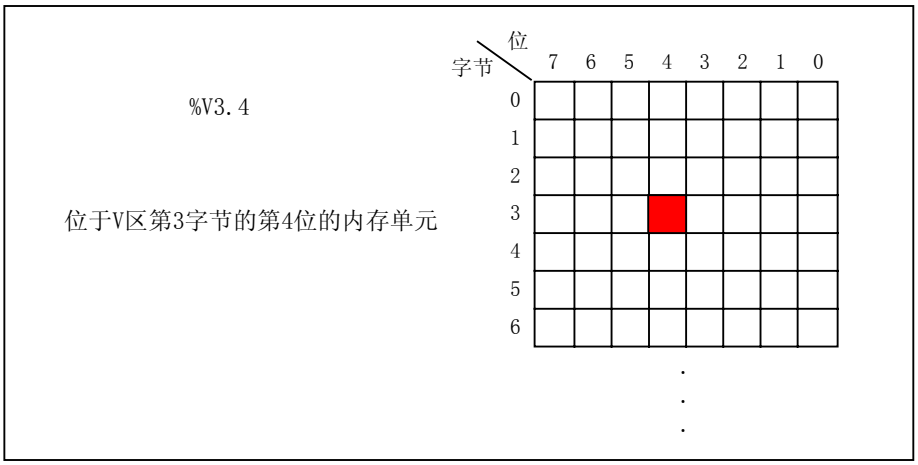


图 3-1

- **字节地址**

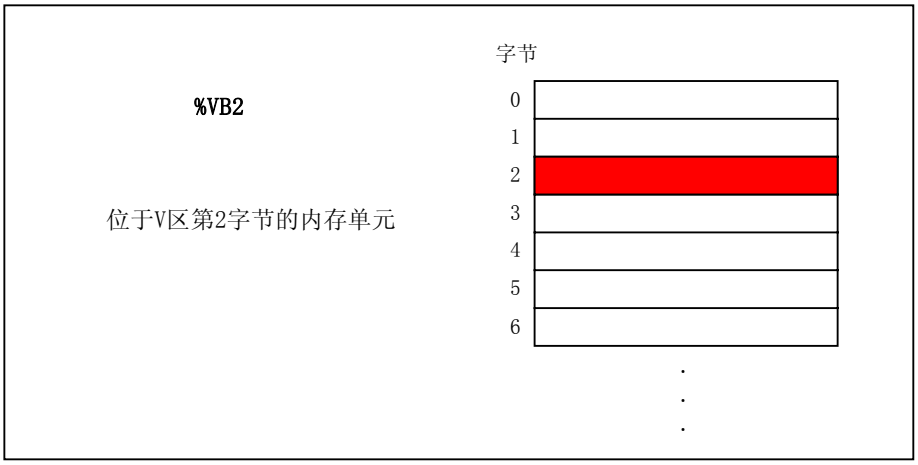


图 3-2

- **字地址**

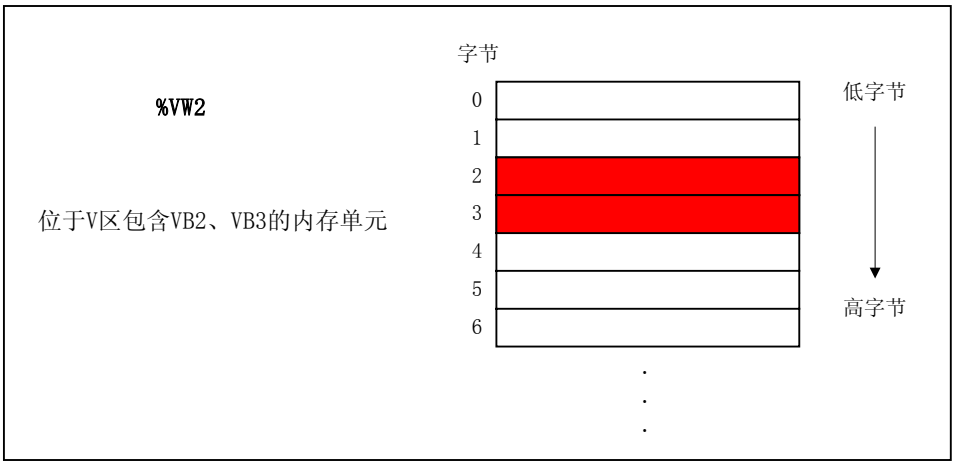


图 3-3

- **双字地址**

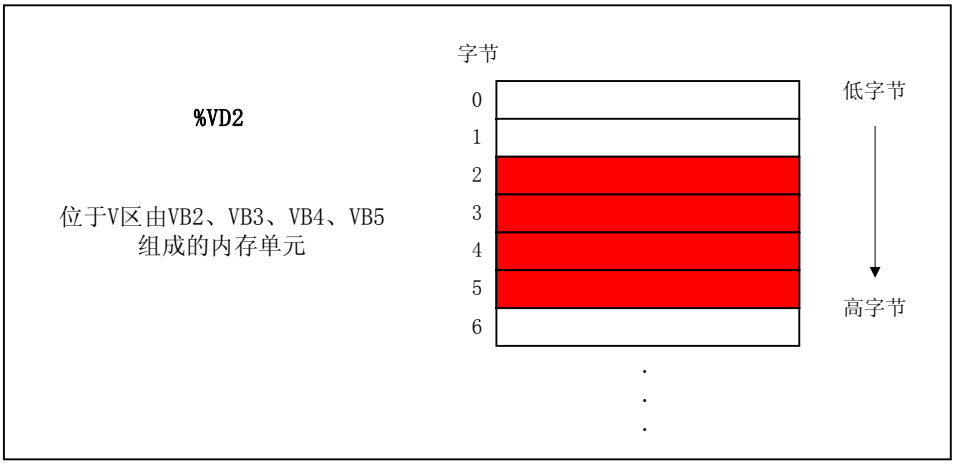


图 3-4

3.6.3 各内存区域的地址范围

KDN-K3 系列 PLC 有几种不同型号的 CPU，不同型号 CPU 中的各内存区域的地址范围可能

有所不同，超出允许范围的地址是非法的。表 3-13 对此进行了详细说明.

		CPU304	CPU306	CPU308
I	长度（字节）	3	8	32
	位地址	I0.0 --- I2.7	I0.0 --- I7.7	I0.0 --- I31.7
	字节地址	IB0 --- IB2	IB0 --- IB7	IB0 – IB31
	字地址	IW0	IW0 --- IW6	IW0 --- IW30
	双字地址	———	ID0 --- ID4	ID0 --- ID28
Q	长度（字节）	3	8	32
	位地址	Q0.0 --- Q2.7	Q0.0 --- Q7.7	Q0.0 --- Q31.7
	字节地址	QB0 --- QB2	QB0 --- QB7	QB0 – QB31
	字地址	QW0	QW0 --- QW6	QW0 --- QW30
	双字地址	———	QD0 --- QD4	QD0 --- QD28
AI	长度（字节）	0	32	64
	字地址	———	AIW0 --- AIW30	AIW0 --- AIW62
AQ	长度（字节）	0	32	64
	字地址	———	AQW0 --- AQW30	AQW0 --- AQW62
V	长度（字节）	4096		
	位地址	V0.0 --- V4095.7		
	字节地址	VB0 --- VB4095		
	字地址	VW0 --- VW4094		
	双字地址	VD0 --- VD4092 其中，VD3840 --- VD4092 只允许存储 REAL 型数据		
M	长度（字节）	32		
	位地址	M0.0 --- M31.7		
	字节地址	MB0 --- MB31		
	字地址	MW0 --- MW30		

	双字地址	MD0 --- MD28
SM	长度（字节）	300
	位地址	SM0.0 --- SM299.7
	字节地址	SMB0 --- SMB299
	字地址	SMW0 --- SMW298
	双字地址	SMD0 --- SMD296
L	长度（字节）	16
	位地址	L0.0 --- L15.7
	字节地址	LB0 --- LB15
	字地址	LW0 --- LW14
	双字地址	LD0 --- LD12

表 3-13 K3 系列 PLC 内存区域的范围

3.6.4 FB 实例存储区的分配

3.6.4.1 IEC61131-3 中定义的标准功能块

- 定时器

  TP --- 脉冲定时器；

  TON --- 接通延时定时器；

  TOF --- 断开延时定时器。

- 计数器

  CTU --- 加计数；

  CTD --- 减计数；

  CTUD --- 加/减计数。

- 双稳态触发器

  SR --- SR 触发器;

  RS --- RS 触发器。

- 边沿检测

  R_TRIG --- 上升沿检测;

  F_TRIG --- 下降沿检测。

3.6.4.2　FB 的实例化

在 IEC61131-3 中，“FB 实例化”的概念特别重要。

所谓实例化，就是用户在变量定义部分通过指定变量名和数据类型来建立一个变量。

FB 也需要如同变量那样首先进行实例化。在程序中不允许直接调用 FB，而只能调用 FB 的实例。形象地讲，在程序中不能读写一个数据类型（比如 INT 型），而只能读写声明为该数据类型（比如 INT 型）的具体的变量。如下图，在程序中只能调用、访问 T1。

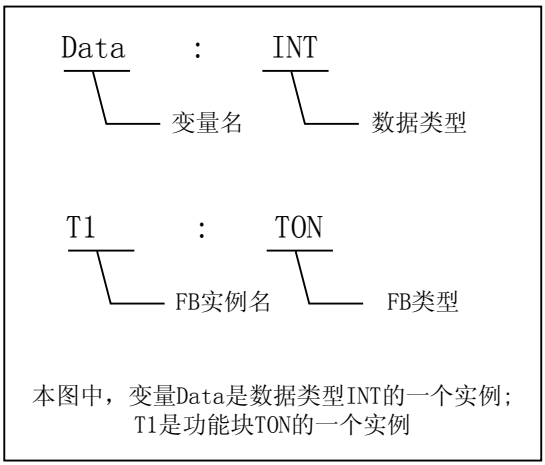


图 3-5　实例化举例

3.6.4.3 FB 实例存储区的分配

在 K3 系列 CPU 的内存内已经为每一种 FB 类型都分配了一个存储区域。当定义了某种 FB 的实例时，EasyProg 会自动在该 FB 类型对应的存储区域内为每一个实例分配一个独立的存储区。

表 3-14 描述了 K3 系列 CPU 为每种 FB 分配的实例存储区域。

T	
描述	定时器区，可在该区域内分配 TON、TOF、TP 的实例。 用于存储所有定时器实例的状态值和当前计时值。
访问方式	直接访问定时器的状态值、当前计时值
存储权限	可读
其它	允许掉电保持，不允许强制
C	
描述	计数器区，可在该区域内分配 CTU、CTD、CTUD 的实例。 用于存储所有计数器实例的状态值和当前计数值。
访问方式	直接访问定时器的状态值、计数值
存储权限	可读
其它	允许掉电保持，不允许强制
RS	
描述	RS 触发器区，可在该区域内分配 RS 的实例。 用于存储所有 RS 实例的状态值。
访问方式	直接访问 RS 状态值
存储权限	可读
其它	不允许强制，不允许掉电保持
SR	
描述	SR 触发器区，可在该区域内分配 SR 的实例。 用于存储所有 SR 实例的状态值。
访问方式	直接访问 SR 状态值
存储权限	可读

其它	不允许强制，不允许掉电保持

表 3-14 FB 示例的存储区

3.6.5 FB 实例的命名及使用

FB 的实例遵循“先定义，后使用”的原则。

为了方便用户，在 EasyProg 中特意作了如下处理：FB 实例的命名遵循传统 PLC 的常用方式，比如 T0、C3 等；用户不需要手工输入 FB 实例的定义语句，只需在程序中调用合法的实例即可，软件将会在全局变量表中为用户调用的实例自动生成定义语句。

- **T**

直接寻址	格式	**T**x
	描述	x：定时器编号，十进制数字。
	数据类型	BOOL --- 存储定时器的状态值 INT --- 存储定时器的当前计时值。 **T**x 兼具以上两种含义，但用户只需在程序中使用实例名即可，其含义将由软件自动识别。
	示例	T0、T5、T20

表 3-15 定时器实例

- **C**

直接寻址	格式	**C**x
	描述	x：计数器编号，十进制数字。
	数据类型	BOOL --- 存储计数器的状态值 INT --- 存储计数器的当前计时值。 **C**x 兼具以上两种含义，但用户只需在程序中使用实例名即可，其含义将由软件自动识别。
	示例	C0、C5、C20

表 3-16 计数器实例

- **RS**

直接寻址	格式	**RS**x
	描述	x：RS 触发器编号，十进制数字。
	数据类型	BOOL --- 存储 RS 触发器的状态值
	示例	RS0、RS5、RS10

表 3-17 RS 触发器实例

- **SR**

直接寻址	格式	**SR**x
	描述	x：SR 触发器编号，十进制数字。
	数据类型	BOOL --- 存储 SR 触发器的状态值
	示例	SR0、SR5、SR10

表 3-18 SR 触发器实例

3.6.6 FB 实例存储区的范围

系统能够为各种 FB 类型分配的存储区域的大小受到硬件本身资源的限制，因此，K3 系列各型号的 CPU 为各 FB 实例存储区分配的范围有所不同，如下表所示：

		CPU304	CPU306	CPU308
T	允许数量	32	128	256
	范围	T0 --- T31	T0 --- T127	T0 --- T255
	分辨率	待定	T0 --- T3：1ms T4 --- T19：10ms T20 --- T127：100ms	待定
	最大定时时间	32767*分辨率	32767*分辨率	32767*分辨率

C	允许数量	32	128	256
	范围	T0 --- T31	T0 --- T127	T0 --- T255
	最大计数值	32767	32767	32767
RS	最大允许数量	16		
	范围	RS0 --- RS15		
SR	最大允许数量	16		
	范围	SR0 --- SR15		

表 3-19 FB 实例存储区的分配

3.7 EasyProg 中应用程序的组织

3.7.1 工程的组织结构

在 EasyProg 中应用程序被组织成“工程（Project)”，工程中包含了用户程序、硬件配置等应用程序的所有信息。

表 3-20 详细描述了工程的组织结构。表中注明“可选”的项表示此项并非工程的必要元素，用户在工程在工程中可以忽略它们。

程序	初始化数据表（可选）	用户可以在此表中为 V 区中的变量指定初始值。 允许类型：BYTE、WORD、INT、DWORD、DINT、REAL。
	主程序	CPU 的主循环任务。在一个工程中有且仅有一个主程序。 主程序实际是 PROGRAMME 类型的 POU。
	中断服务程序（可选）	中断任务，当指定的中断事件发生时才会执行。 在一个工程中最多允许有 32 个中断服务程序 中断服务程序实际是 PROGRAMME 类型的 POU。
	子程序（可选）	可重用的程序模块。只有在主程序或者中断服务程序中被调用时才能得以执行。在一个工程中最多允许有 32 个子程序。 子程序实际是 PROGRAMME 类型的 POU。

配置	资源	硬件配置	用户在此对工程中用到的 PLC 模块及其参数进行配置。 CPU 将在冷启动时读取一次硬件配置。
		全局变量表 （可选）	用户在此定义工程中需要的全局变量。

表 3-20 工程的组织结构

工程的存储目录

在建立工程时 EasyProg 软件将会要求用户输入工程的存放路径，工程主文件（扩展名为.kpr）就存放于此路径下。另外，在此路径下将会自动建立一个与工程名称相同的子目录，用于存放该工程所有的程序文件、变量文件以及其它的一些临时文件等。

例如，用户若选择在 c:\temp 目录下建立一个名为 project 的工程，则工程主文件的路径是 c:\temp\project.kpr，其它文件存放在 c:\temp\project 目录中。

3.8 CPU 中程序的执行

在 CPU 中只有主程序、中断服务程序才能得以执行，其中主程序是被循环连续执行的，是主循环，中断服务程序在系统定义的中断事件发生时方能执行。

CPU 连续执行用户主程序的过程称为扫描。如图 3-6，在扫描周期内 CPU 将执行如下任务：

- 执行自诊断
- 读取物理输入通道的状态并将其写至输入映像寄存器
- 执行用户程序
- 处理通讯请求
- 将输出寄存器中的状态写至物理输出通道

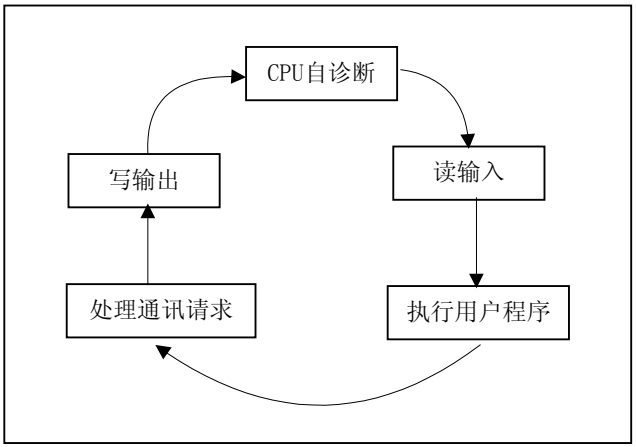


图 3-6 扫描周期

中断事件可能发生在扫描过程的任一时刻，这时候 CPU 将会先暂时中断主循环，转去执行中断服务程序，中断服务程序执行完成后，再在主循环的断点重新进入主循环。如图 3-7。

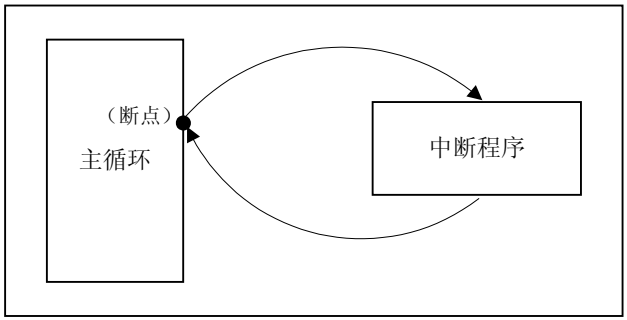


图 3-7 中断服务程序的执行

第四章 EasyProg 软件的使用

本章对 EasyProg 软件界面的组成以及各部分的功能、使用进行了详细的描述，用户在掌握上一章介绍的基本概念的基础上，通过阅读本章可以快速认识并理解 EasyProg 的具体功能以及操作。

LD 编辑器和 IL 编辑器的使用将涉及到 IEC61131-3 标准中的许多语法，因此在本章中未作介绍，相关内容以及语法将在后续章节中进行详细描述。

4.1 界面总体介绍

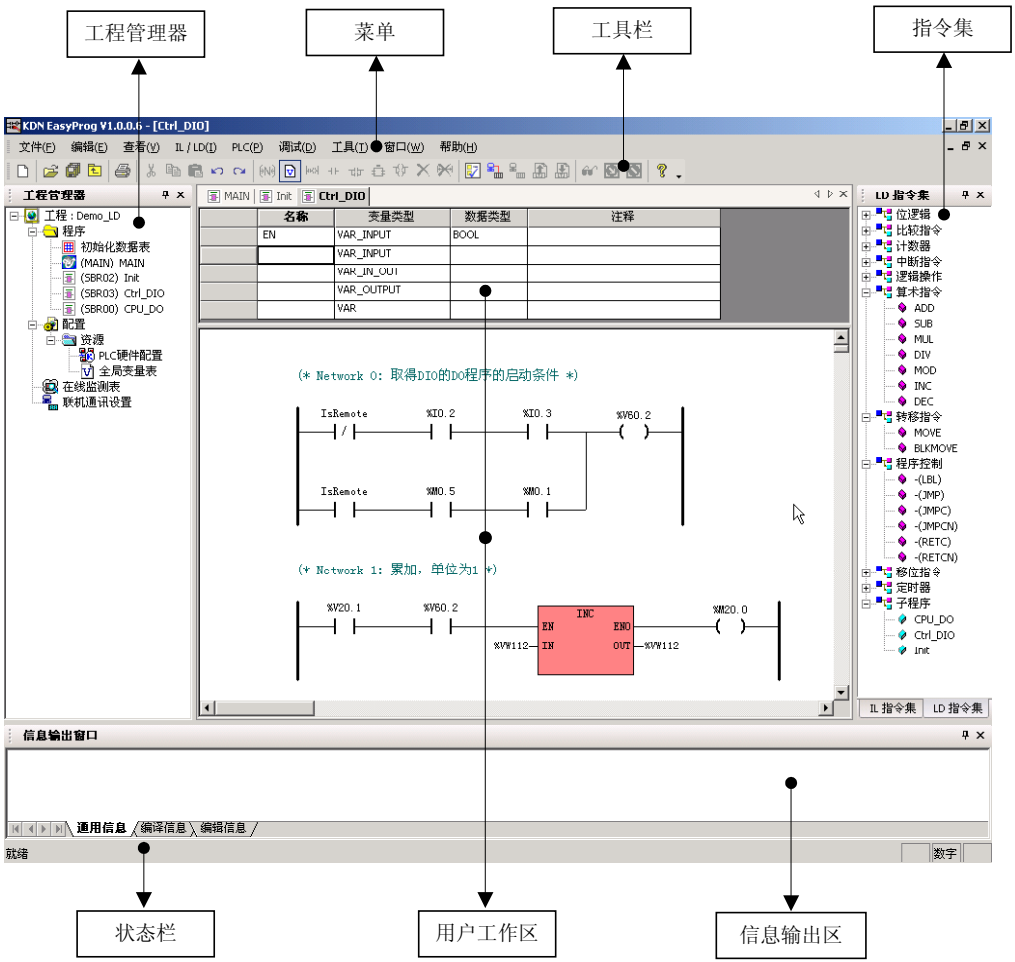


图 4-1 界面的总体组成

- 菜　　单：菜单中包含了 EasyProg 软件所有的操作命令。
- 工 具 栏：工具栏中包含了用户使用频度较高的一些操作命令。
- 状 态 栏：状态条提供了软件当前的状态信息和操作命令的提示信息。

- 工程管理器：工程管理器中采用树型结构显示了整个工程的组织结构，包括程序、配置等。用户可以在此对当前工程进行操作、管理。工程管理器中支持右键菜单。
- 用户工作区：这是用户使用的主要区域，用户可以在此打开硬件配置窗口、全局变量表、编辑器等窗口，完成配置硬件、声明全局变量、编辑程序等任务。

  图 4-1 显示的是编辑器，包括区域上部的变量定义表格和区域下部的程序编辑器。编辑器又分为 LD 编辑器、IL 编辑器。
- 指　令　集：以树状列出了 K3 系列 PLC 支持的所有指令、功能、功能块和用户自己编写的子程序。指令集又分为 LD 指令集、IL 指令集。
- 指　令　集：以树状列出了 K3 系列 PLC 支持的所有指令、功能、功能块和用户自己编写的子
- 信息输出区：用于显示软件输出的提示信息，包括编译信息、查找结果等。

4.2　菜单命令介绍

4.2.1 【文件】菜单

- 〔新建工程…〕

建立一个新工程。

用户可通过鼠标单击此菜单或者快捷键 Ctrl+N 来执行。执行该命令后，将弹出对话框如图 4-2。

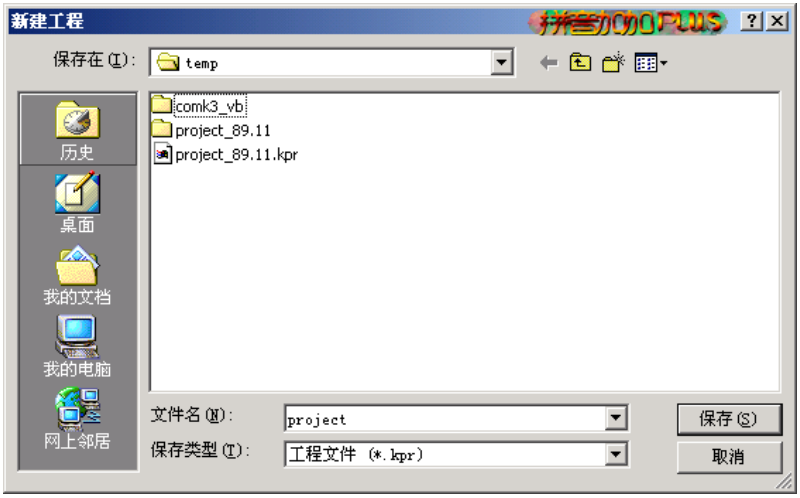


图 4-2　新建工程

选择好路径并输入工程文件名后，单击〔保存〕按钮即可。

- 〔打开工程…〕

打开一个已经存在的工程。

用户可通过鼠标单击此菜单或者快捷键 Ctrl+O 来执行。执行该命令后，将弹出对话框如图 4-3。

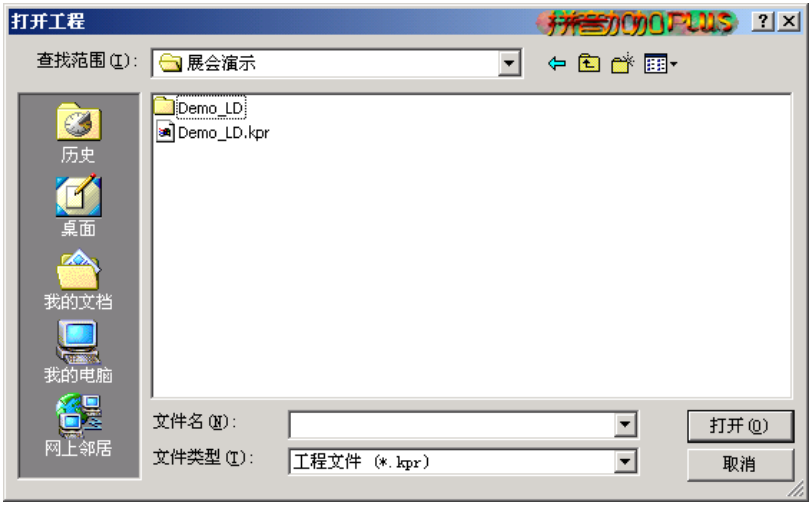


图 4-3 打开工程

选择好要打开的工程后，单击〔打开〕按钮即可。

- 〔保存工程〕

  保存当前正打开的工程，工程名称不变。

  用户可通过鼠标单击此菜单或者快捷键 Ctrl+S 来执行。

- 〔工程另存为…〕

  将当前已经打开的工程重新命名并以新名字保存。

  用户可通过鼠标单击此菜单来执行。执行该命令后，将弹出对话框如图 4-4。

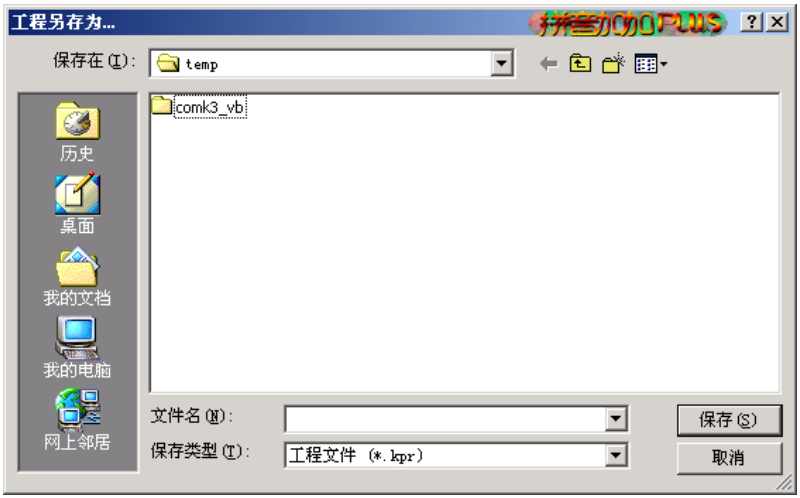


图 4-4 工程另存为...

选择好路径并输入工程文件名后，单击〔保存〕按钮即可。

- 〔导入工程…〕

  导入一个已存在的工程备份文件（扩展名为.zip）并将其打开。

  用户可通过鼠标单击此菜单来执行。

  ① 执行该命令后，首先将弹出“导入工程…”对话框，如图 4-5。

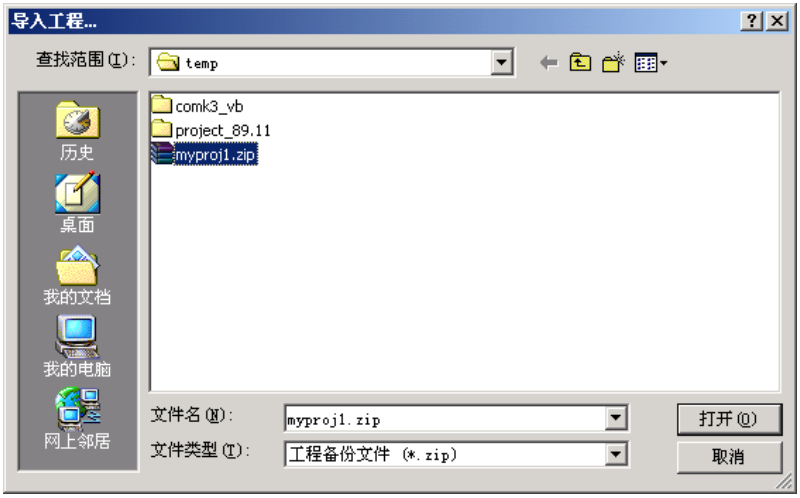


图 4-5

② 选择好备份文件后并单击〔打开〕按钮，将弹出对话框，以选择备份文件解压之后得到的工程文件的存放目录，如图 4-6。

图 4-6

选择好路径后，单击〔确定〕按钮即可工程文件解压到选定的目录下并将其打开。

若在选定的目录下存在与压缩文件中所包含的工程同名的工程，则会被直接覆盖。

- 〔导出工程…〕

  将当前打开的工程所涉及到的所有文件压缩至一个文件（扩展名.zip）中以便于用户备份。

  用户可通过鼠标单击此菜单来执行。执行后弹出“导出工程…”对话框，如图 4-7 所示。

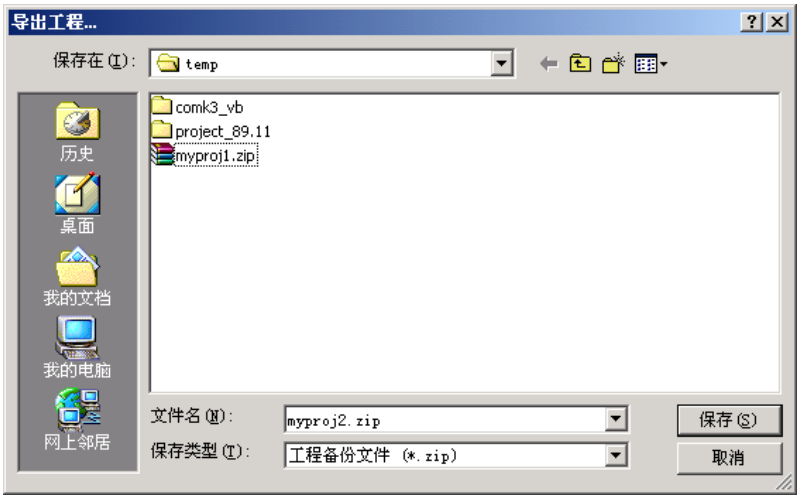


图 4-7 导出工程

选择好路径并输入备份文件名后，单击〔保存〕按钮即可。

在压缩文件中，所有的文件名、相对路径以及工程的相关设置均保持不变。

例如，工程名称为 project，工程文件为 project.kpr，则压缩文件中的工程文件仍旧为 project.kpr，工程名称仍旧为 project。

- 〔关闭工程〕

  关闭当前打开的工程；

- 〔最近文件列表〕

  在此处列出了用户最近使用过的 4 个工程文件。

用户可以直接单击列表中的文件名称来打开相应的工程。

- 〔退出〕

  退出本软件。

  用户可通过鼠标单击此菜单或者快捷键 Alt+F4 来执行。

4.2.2 【编辑】菜单

【编辑】菜单中的各命令主要在编写 IL、LD 程序时使用。

- 〔撤销〕

  撤销最近一次的操作。

  用户可通过鼠标单击此菜单或者快捷键 Ctrl+Z 来执行。

- 〔重复〕

  重复最近一次撤销的操作。

  用户可通过鼠标单击此菜单或者快捷键 Ctrl+Y 来执行。

- 〔全选〕

  用于选定编辑区中所有可编辑的内容。

用户可通过鼠标单击此菜单或者快捷键 Ctrl+A 来执行。

- 〔剪切〕

  删除选中的内容，并将选中的内容复制至剪贴板中。

  用户可通过鼠标单击此菜单或者快捷键 Ctrl+X 来执行。

- 〔复制〕

  将选中的内容复制至剪贴板中。

  用户可通过鼠标单击此菜单或者快捷键 Ctrl+C 来执行。

- 〔粘贴〕

  将剪贴板中剪切或者复制的内容放至编辑器中。

  用户可通过鼠标单击此菜单或者快捷键 Ctrl+V 来执行。

- 〔查找〕

  用于在当前活动的 IL 文档中查找指定的字符串。

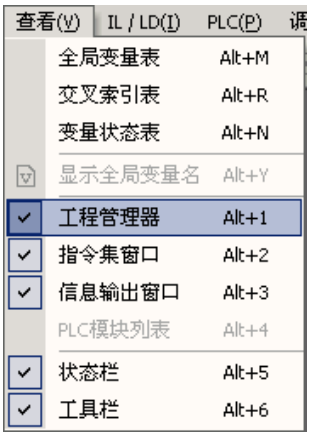
  用户可通过鼠标单击此菜单或者快捷键 Ctrl+F 来执行。执行后将弹出“查找…”对话框。

- 〔查找〕

  用于在当前活动的 IL 文档中查找指定的字符串，并用指定的字符串来代替查找到的结果。

  用户可通过鼠标单击此菜单或者快捷键 Ctrl+R 来执行。执行后将弹出“替换…”对话框。

4.2.3 【查看】菜单



- ［全局变量表］

  打开全局变量表窗口。关于全局变量表的详细说明请参见 4.9 全局变量表 部分。

  用户可通过鼠标单击此菜单或者快捷键 Alt+M 来执行。

- ［交叉索引表］

  打开交叉索引表窗口。关于交叉索引表的详细说明请参见 4.10 交叉索引表 部分。

  用户可通过鼠标单击此菜单或者快捷键 Alt+R 来执行。

- 〔变量状态表〕

  打开变量状态表视图。关于变量状态表的详细说明请参见 4.11 变量状态表 部分。

  用户可通过鼠标单击此菜单或者快捷键 Alt+N 来执行。

- 〔显示全局变量名〕

  复选。在 IL、LD 编辑器中切换显示全局变量名或者直接 I/O 地址。

  用户可通过鼠标单击此菜单或者快捷键 Alt+Y 来执行。

- 〔工程管理器〕

  复选。切换是否显示工程管理器窗口。

  用户可通过鼠标单击此菜单或者快捷键 Alt+1 来执行。

- 〔指令集窗口〕

  复选。切换是否显示指令集窗口。

  用户可通过鼠标单击此菜单或者快捷键 Alt+2 来执行。

- 〔信息输出窗口〕

  复选。切换是否显示信息输出窗口。

  用户可通过鼠标单击此菜单或者快捷键 Alt+3 来执行。

- 〔PLC 模块列表〕

  复选。切换是否显示 PLC 模块列表。在进行硬件配置时该命令方能生效;

  用户可通过鼠标单击此菜单或者快捷键 Alt+4 来执行。

- 〔状态栏〕

  复选。切换是否显示状态栏。

  用户可通过鼠标单击此菜单或者快捷键 Alt+5 来执行。

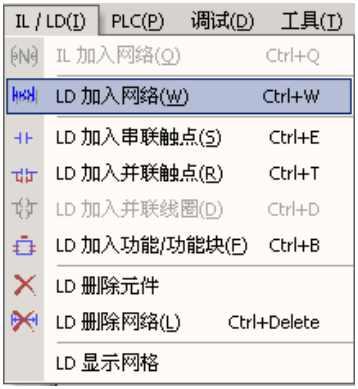
- 〔工具栏〕

  复选。切换是否显示工具栏。

  用户可通过鼠标单击此菜单或者快捷键 Alt+6 来执行。

4.2.4 【IL/LD】菜单

【IL/LD】菜单中的所有菜单命令仅当编辑 IL、LD 程序时方能生效。



- 〔IL 加入网络〕

  在 IL 程序中加入一个网络 (Network)。

  用户可通过鼠标单击此菜单或者快捷键 Ctrl+Q 来执行。

- 〔LD 加入网络〕

  在 LD 程序中在选中的元件所在的网络之后加入一个网络 (Network)。

  用户可通过鼠标单击此菜单或者快捷键 Ctrl+W 来执行。

- 〔LD 加入串联触点〕

  在 LD 程序中以选中的元件为基准加入一个串联触点。

  用户可通过鼠标单击此菜单或者快捷键 Ctrl+E 来执行。

- 〔LD 加入并联触点〕

  在 LD 程序中以选中的元件为基准加入一个并联触点。

  用户可通过鼠标单击此菜单或者快捷键 Ctrl+T 来执行。

- 〔LD 加入并联线圈〕

  在 LD 程序中以选中的线圈为基准加入一个并联线圈。

用户可通过鼠标单击此菜单或者快捷键 Ctrl+D 来执行。

- 〔LD 加入功能/功能块〕

  在 LD 程序中以选中的元件为基准加入一个串联的功能或者功能块（包含用户编写的子程序）。

  用户可通过鼠标单击此菜单或者快捷键 Ctrl+B 来执行。

- 〔LD 删除元件〕

  在 LD 程序中删除一个选中的触点、线圈、功能或者功能块。

  用户可通过鼠标单击此菜单或者快捷键 Delete 来执行。

- 〔LD 删除网络〕

  在 LD 程序中删除一个选中元件所在的梯级。

  用户可通过鼠标单击此菜单或者快捷键 Ctrl+Delete 来执行。

- 〔LD 显示网格〕

  复选。切换是否在 LD 编辑器中显示网格。

4.2.5 【PLC】菜单

- 〔编译工程〕

  编译当前工程，生成 PLC 可执行的代码。

  用户可通过鼠标单击此菜单或者快捷键 F7 来执行。

- 〔联机〕

  将当前编程设备与 PLC 进行连接。

  用户可通过鼠标单击此菜单或者快捷键 F2 来执行。

- 〔取消联机〕

  取消编程设备与 PLC 当前的联机状态。

  该命令在联机成功后方能生效。用户可通过鼠标单击此菜单或者快捷键 Ctrl+F2 来执行。

- 〔上载…〕

  将 PLC 中应用工程数据，包括程序、配置等，上载当前工程中并打开。

  该命令在联机成功后方能生效。用户可通过鼠标单击此菜单或者快捷键 Ctrl+F8 来执行。

  执行此命令后，将首先弹出对话框如图 4-8 所示。

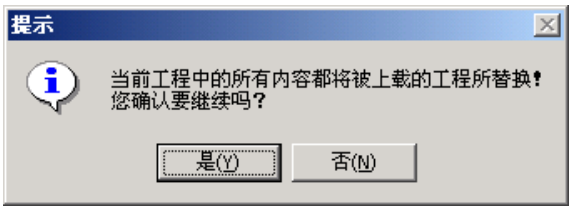


图 4-8

►单击〔是(Y)〕，将继续上载工程并以上载的内容完全替换当前打开的工程中的内容。

►单击〔否(N)〕，则将停止上载。

- 〔下载…〕

  将当前工程下载至 PLC 中。

该命令在联机成功后方能生效。用户可通过鼠标单击此菜单或者快捷键 F8 来执行。

为保证下载到 CPU 中的程序是最新的，下载之前会自动进行编译。

在下载时 CPU 必须处于停止状态，因此下载之前会自动查询 CPU 的当前状态并进行提示。

- 〔比较…〕

比较当前打开的工程与 CPU 内正在运行的工程是否一致。

该命令在联机成功后方能生效。用户可通过鼠标单击此菜单来执行。

- 〔清除…〕

将 CPU 中的工程信息（包括程序、配置等）完全删除，并将所有的内存区域复位。

该命令在联机成功后方能生效。用户可通过鼠标单击此菜单来执行。

- 〔修改密码…〕

修改当前 CPU 的保护密码以及保护等级。执行后的窗口如图 4-9 所示。

该命令在联机成功后方能生效。用户可通过鼠标单击此菜单来执行。

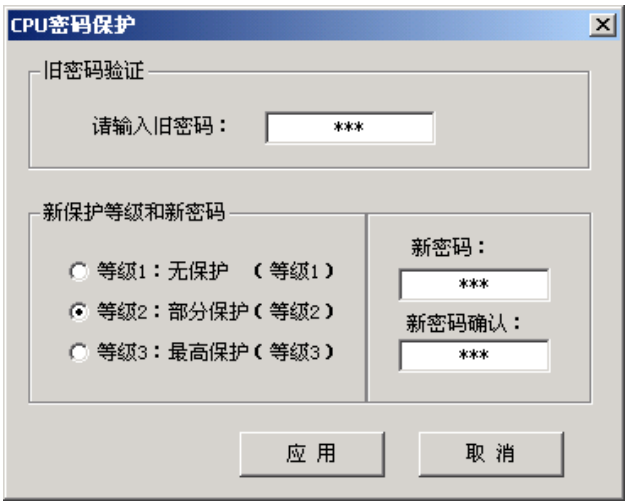


图 4-9　修改保护密码

►〔旧密码验证〕：输入当前 CPU 的旧密码。在修改密码之前，CPU 将首先验证输入的旧密码

与现有的保护密码是否一致，若一致才允许修改密码。

► 保护等级：选择 CPU 的密码保护等级。共分三级，详细的等级说明请参见下面的表 4-1。

►〔新密码〕：输入将要设置的新密码。密码的长度最大允许 8 个字符。

►〔新密码确认〕：再次输入将要设置的新密码进行确认。

►〔应用〕：单击此按钮将把新密码和新保护等级写入 CPU 中。

►〔取消〕：单击此按钮将取消当前的输入并关闭本窗口。

- 〔模块信息…〕

读取并显示所连 CPU 的相关信息，包括版本、硬件错误信息等。执行后的窗口如图 4-10 所示。该命令在联机成功后方能生效。用户可通过鼠标单击此菜单来执行。

图 4-10　模块信息

►〔CPU 固件版本号〕：显示当前 CPU 固件（firmware）的版本号。

►〔模块信息〕：显示当前 CPU 及其所连所有模块的型号和诊断到的硬件错误信息。

►〔刷新〕：单击此按钮，将立即从 CPU 中读取一次模块信息并对窗口的显示进行刷。

►〔退出〕：单击此按钮，将退出此窗口。

- 〔调整 CPU 时钟…〕

读取并可修改 CPU 内的实时时钟值。执行后的窗口如下图所示。

该命令在联机成功后方能生效。用户可通过鼠标单击此菜单来执行。

图 4-11 调整 CPU 时钟

►〔系统当前时间〕：显示编程所用 PC 机当前的日期、时间。

►〔PLC 当前时间〕：显示所连 CPU 内实时时钟的当前时间。

若背景色是绿色，代表成功地从 CPU 内读到了实时时钟。

若背景色是黄色，代表从 CPU 内读取实时时钟失败。

►〔将 PLC 时间调整为〕：用户可以在此输入将要设置的 CPU 实时时钟的新时间值。

►〔修改时钟〕：单击此按钮，用户输入的新的时间值将被写入 CPU 内的实时时钟中。

►〔读取时钟〕：单击此按钮，将从 CPU 内读取实时时钟值并刷新〔PLC 当前时间〕显示。

►〔关闭〕：单击此按钮，将取消所有的输入并退出本窗口。

4.2.6 【调试】菜单

- 〔启动…〕

  远程启动 CPU。

  该命令在联机成功后方能生效。用户可通过鼠标单击此菜单来执行。

- 〔停止…〕

  远程停止 CPU。

  该命令在联机成功后方能生效。用户可通过鼠标单击此菜单来执行。

- 〔在线监测〕

  复选，切换进入或者退出在线监测状态。

  在线监测时，EasyProg 软件将连续从 CPU 内读取当前工程中用到的所有变量并显示。

  若 EasyProg 中当前打开的工程与 CPU 内运行的工程不一致，则不允许进入在线监测状态。

  该命令在联机成功后且当前视图是 IL、LD 编辑器或者变量状态表时方可生效。

  用户可通过鼠标单击此菜单或者快捷键 F4 来执行。

- 〔变量强制〕

  将变量状态表中所输入的“地址”置为强制状态，并将其值强行设置为输入的“强制值”。

  若“地址”对应的“强制值”没有输入，则不对该“地址”进行强制。

  该命令在联机成功后且当前视图为变量状态表时方可生效。用户可通过鼠标单击此菜单来执行。

注意：KDN-K3 系列 CPU 采用强制优先模式。

比如，若 I1.0 处于强制状态，则其强制值将生效，而来自现场的输入将无效!

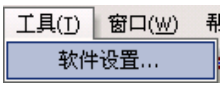
- 〔取消强制〕

取消 CPU 内所有变量的强制状态。

该命令在联机成功后方可生效。用户可通过鼠标单击此菜单来执行。

由于 KDN-K3 系列 CPU 采用强制优先模式，一定要记得在调试结束后取消强制状态!

4.2.7 【工具】菜单



- 〔软件设置…〕

对 EasyProg 软件进行配置。

用户可通过鼠标单击此菜单来执行。执行此命令后，将弹出“软件设置”对话框。对话框内分为如下几个页面:

① 〔界面风格〕页面

此处允许对界面的外观风格进行设置。

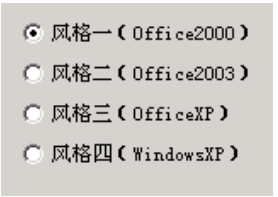


图 4-12　〔界面风格〕页面

界面风格有如下四种选择:

► 风格一 (Office2000): 仿 Microsoft Office 2000 的界面样式。

► 风格二 (Office2003): 仿 Microsoft Office 2003 的界面样式。

► 风格三 (OfficeXP) : 仿 Microsoft Office XP 的界面样式。

► 风格四 (Windows XP): 在 Windows XP 操作系统下建议用户选择此风格。

② 〔编辑器〕页面

此处允许对编辑器的某些环境变量，包括字体、颜色、数据显示格式等，进行配置。

图 4-13 〔编辑器〕页面

► 〔在线监测时数据显示格式〕

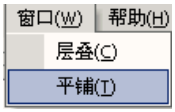
选择当在线监测时整数的显示格式。有如下三种选项:

〔混合〕: INT、DINT 型数据以十进制格式，BYTE、WORD、DWORD 型数据以十六进制格式显示。

〔十 进 制〕: 所有整数都以十进制格式显示。

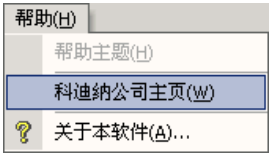
〔十六进制〕: 所有整型数据都以十六进制格式显示。

4.2.7 【窗口】菜单



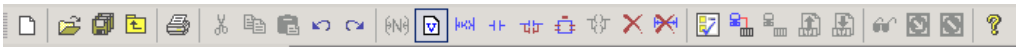
这些全是标准的 Windows 命令，不再赘述。

4.2.8 【帮助】菜单



这些命令非常简单，此处不再赘述。

4.3 工具栏介绍



在工具栏中集成了经常使用的功能选项，每一个工具按钮都对应着一个菜单命令，为用户的开发提供了非常大的便利。

当把光标移到某个工具按钮图标上并稍微停留一会，就会弹出信息框来提示该按钮的功能，同时在状态栏上也会有提示。

- 

新建工程或者程序。

若在软件中当前没有打开的工程，单击此按钮将新建工程；若当前有打开的工程，单击此按钮将新建一个程序。

4.4 工程管理器介绍

工程管理器是界面中的主要窗口之一，以树状列表的形式直观地显示出了当前工程的所有组成部分，包括程序、硬件配置、变量状态表、全局变量表等。用户可以在此窗口中对当前打开的工程进行管理、操作、维护。

工程管理器的各个树节点均支持右键，用右键单击某个节点将会弹出相应的右键菜单来。

工程管理器的外观如图 4-13 所示。

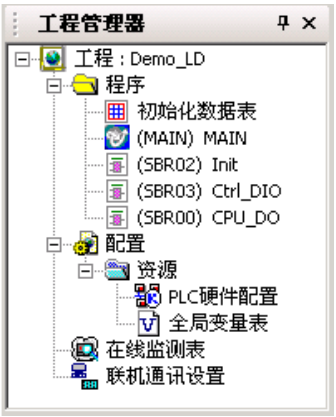


图 4-13　工程管理器

4.4.1 浮动的工程管理器窗口

工程管理器采用了浮动窗口技术，用户可以根据需要方便地进行切换使其浮动或者在固定位置显示，方法如下：

① 方法一

执行【查看】→〔工程管理器〕菜单命令，即可进入浮动状态并切换使其隐藏或者显示。

② 方法二

- 单击窗口右上角的  图标，即可让窗口进入浮动状态，并缩成一个图标停靠在屏幕左边，图标的标题是〔工程管理器〕。
- 在浮动状态下，将鼠标置于屏幕左边的〔工程管理器〕图标上并停留片刻，工程管理器窗口就会出现在屏幕上并允许用户进行正常的操作，然后若将鼠标置于工程管理器窗口之外，它又将收缩于屏幕左侧。

• 在浮动状态下，单击工程管理器窗口右上角的  图标，就会重新回复至固定位置显示。

4.4.2 【程序】组

4.4.2.1 描述

在【程序】组下列出了当前工程中包含的所有程序块，包括：

① 初始化数据表（可选）

用户可在初始化数据表中为 V 区中 BYTE、WORD、DWORD、INT、DINT、REAL 类型的数据指定初始值。初始化数据表是工程中的可选部分，用户也可以不在表中输入任何内容。

关于初始化数据表的更详细的描述请参见下面　4.6 初始化数据表　部分。

② 主程序

主程序是在 CPU 内运行的主循环任务，在 CPU 的每个扫描周期内都会被执行一次。

请参阅前文　3.8　CPU 中程序的执行　部分。

在工程中有且仅有 1 个主程序。主程序必须在其它程序之前创建，且创建之后不能删除。

③ 中断服务程序（可选）

中断任务，当指定的中断事件发生时方会被执行。

请参阅前文　3.8　CPU 中程序的执行　部分。

在一个工程中最多可有 32 个中断服务程序。中断服务程序可以从工程中删除，其名称也可以修改。

④ 子程序（可选）

子程序是可重用的代码段，是为了方便用户进行结构化程序设计而提供的，用户可以将

常用的控制功能编写成一个个子程序。

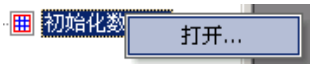
子程序仅供主程序和中断服务程序调用，只有被调用的子程序才能在 CPU 中得以运行。

在一个工程中子程序最多可有 32 个。子程序可以从工程中删除，其名称也可以修改。

4.4.2.2 操作方法

① 初始化数据表

在〔初始化数据表〕节点上单击鼠标右键，将会弹出如下右键菜单：



执行此〔打开...〕菜单命令，或者双击〔初始化数据表〕节点，将会打开初始化数据表。

②【程序】组

在【程序】组节点上单击鼠标右键，将会弹出如下右键菜单：

执行此〔新建...〕菜单命令，将开始新建一个程序。

1）首先弹出〔新建程序...〕对话框如图 4-14。

图 4-14　“新建程序...”对话框

- 〔名称〕：输入该程序的名称。取名的原则请参见 3.4.1 标识符的定义。
- 〔编号〕：为该程序任选一个编号。
- 〔作者〕：输入作者的名字。可选。
- 〔类型〕：选择该程序的类型：主程序、子程序、中断服务程序。

  若工程中没有主程序，将只允许选择“主程序”。

  若工程中已存在主程序，则不允许选择“主程序”。
- 〔语言〕：选择该程序所用的语言：指令表（IL）、梯形图（LD）。

2) 对上述各项作好选择后，单击〔确定〕按钮即可进入 IL 或者 LD 编辑器开始编程。

③ 各个程序节点

在各个程序的名称上单击鼠标右键，将会弹出如下右键菜单：

► 〔打开〕：打开该程序。双击该程序名称，也可以打开该程序。

► 〔删除〕：从工程中删除该程序。

► 〔重命名〕：为该程序重新命名。

► 〔属性〕：打开“程序属性“对话框，如图 4-15，可以对该程序的属性进行修改。

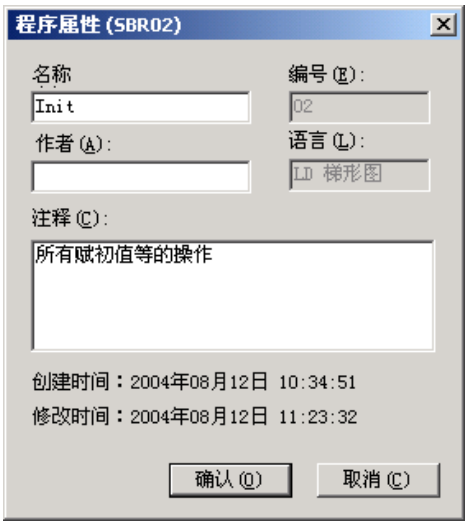


图 4-15　“程序属性”对话框

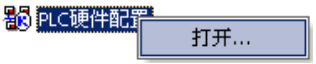
4.4.3 【配置】→【资源】组

所谓资源，主要是指应用系统的硬件构成。

【资源】组中包含了〔PLC 硬件配置〕和〔全局变量表〕两个子项。

4.4.3.1 〔PLC 硬件配置〕

在〔PLC 硬件配置〕节点上单击鼠标右键，将会弹出如下右键菜单：



执行此〔打开…〕命令，或者双击〔PLC 硬件配置〕节点，均可以进入硬件配置窗口。

4.4.3.2〔全局变量表〕

在〔全局变量表〕节点上单击鼠标右键，将会弹出如下右键菜单：

执行此〔打开…〕命令，或者双击〔全局变量表〕节点，均可以进入全局变量定义窗口。

4.4.4 【变量状态表】

在〔变量状态表〕节点上单击鼠标右键，将会弹出如下右键菜单：

执行此〔打开…〕命令，或者双击〔变量状态表〕节点，均可以进入变量状态表窗口。

4.4.5 【联机通讯设置】

4.4.5.1 描述

在〔联机通讯设置〕节点上单击鼠标右键，将会弹出如下右键菜单：

执行此〔打开…〕命令，或者〔联机通讯设置〕节点，均可进入“编程设备联机通讯设置”窗口，对当前编程设备的串行通讯口的通讯参数进行设置，以便与 PLC 联机。

4.4.5.2 “编程设备联机通讯设置”窗口描述

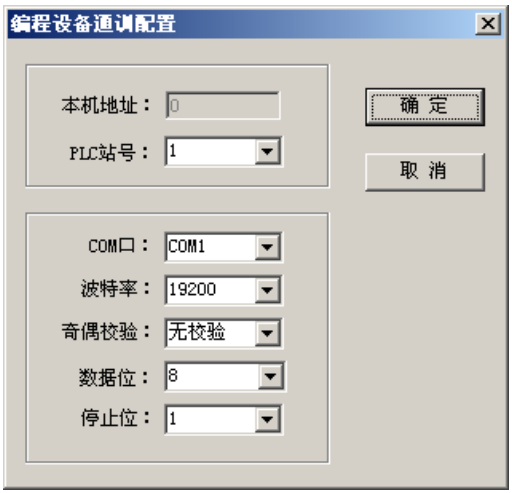


图 4-16 编程设备联机通讯设置

- 〔本机站号〕：当与 K3 系列 PLC 联机时，本机作为主站，其站号固定为 0。
- 〔PLC 站号〕：选择将要连接的 CPU 的站号，该站号即 CPU 作为 Modbus 从站时的站号。
- 〔COM 口〕 ：选择本机将要使用的 COM 口。
- 〔波特率〕 ：设置串行通讯的波特率，其值有：2400、4800、9600、19200、38400。
- 〔奇偶校验〕：设置串行通讯的奇偶校验方式，其值有：无校验、奇校验、偶校验。
- 〔数据位〕 ：设置串行通讯的数据位，其值有：7、8、9。
- 〔停止位〕 ：设置串行通讯的停止位，其值有：1、2。

若没有进入该窗口进行设置，则 EasyProg 软件默认使用的联机通讯参数为：将要连接的 CPU 的〔PLC 站号〕为 1；本机使用 COM1 口，波特率 19200，无校验，8 位数据位，1 位停止位。

4.5 指令集窗口

在指令集窗口中以树状列表的形式列出了 KDN-K3 系列 PLC 支持的所有指令，目的是为用

户的编程提供方便。指令集又分为 IL 指令集和 LD 指令集两大类。目前“IL 指令集”仅供用户浏览，没有实际操作。在进行 LD 编程时可以操作 LD 指令集。指令集窗口的外观如图 4-17。

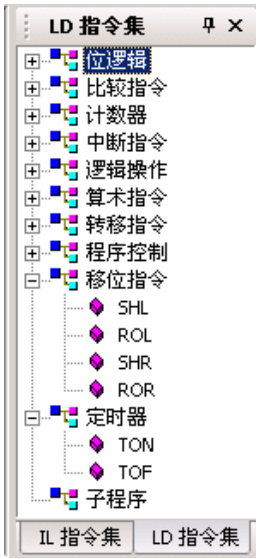


图 4-17　指令集窗口

4.5.1 浮动的指令集窗口

指令集窗口支持浮动技术，具体的操作方法请参见 4.4.1 浮动的工程管理器窗口 部分。

4.5.2　在编程时使用指令集窗口

当编辑 LD 程序时，可按如下步骤来使用指令树：

- 单击当前程序中的某个元件，该元件将被选中作为基准；
- 展开 LD 指令树中相应的文件夹，双击要加入的指令即可将该指令加入到程序中。

4.6　信息输出窗口

信息输出窗口用于显示 EasyProg 软件提示的各种信息，其中“编译信息”窗口显示了用户

最近一次的编译信息，而“通用信息”窗口显示了其它一些提示信息，比如查找结果等。

信息输出窗口的外观如图 4-18。

图 4-18　信息输出窗口

4.6.1 浮动的信息输出窗口

信息输出窗口支持浮动技术，具体的操作方法请参见 4.4.1 浮动的工程管理器窗口 部分。

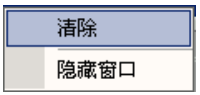
4.6.2　在信息输出窗口中进行操作

①　错误定位

若当前工程编译有错误，则双击“编译信息”窗口中的错误信息就会自动打开相应的程序并定位到错误所在的位置，其中：IL 程序将定位到行，LD 程序将定位到网络（Network）。

②　右键菜单

在输出区内任意位置单击鼠标右键，将弹出如下右键菜单：



- 〔清除〕　：清除当前输出区内的所有内容。
- 〔隐藏窗口〕：隐藏（或者称浮动）信息输出窗口

4.7 PLC 硬件配置

EasyProg 软件为用户提供了功能完善、使用灵活方便的硬件配置环境，用户可以根据需要在

此对工程中所用的 PLC 进行配置，包括定义用到的 PLC 模块，配置各个模块的具体参数等。硬件配置窗口的样式如图 4-19。从图中可以看出，窗口可分为两部分：

图 4-19　硬件配置窗口

- PLC 模块列表

以表格形式列出了本工程中用到的 PLC 模块，并允许用户进行配置。PLC 模块在表格中的排列次序应当与实际模块在背板总线上的连接次序一致。

- PLC 模块参数配置窗口

在该窗口可以对列表中各 PLC 模块的实际参数进行配置。

硬件配置信息必须下载到 CPU 中方可生效。在每次冷启动时，CPU 会首先检测实际所连模

块的信息，并将检测到的实际信息与储存的配置信息进行比较，若有错误就会转入 STOP 状态，否则就进入正常运行状态。

硬件配置是工程中必需的信息，另外，根据所配 CPU 的不同，系统允许访问的 I/O 区域也有所不同，因此建议用户在工程中首先完成硬件配置。当用户新建一个工程时，EasyProg 将缺省地加入一个 K306-24DT 型号的 CPU。

4.7.1 如何进入硬件配置窗口?

有如下两种方式可以进入硬件配置窗口:

- 双击工程管理器中【资源】组下的〔PLC 硬件配置〕节点;
- 在〔PLC 硬件配置〕节点上单击鼠标右键，然后执行弹出的〔打开…〕菜单命令。

4.7.2 添加、删除模块

- 添加模块

鼠标单击 PLC 模块列表的表格中将要加入模块的位置，将焦点置于该行，然后在右侧的〔PLC 模块列表〕中双击要加入的模块即可。若该行已有模块存在，则必须首先删除已存在的模块，然后才能够加入新的模块。

第 1 行（“位置”为 1 的行）只能加入 CPU 模块，其余各行只能加入扩展 I/O、功能模块。

各个模块之间不允许有空行存在。若有空行，则软件不允许在其后添加模块，并且在保存、编译时会提示错误。

CPU304、CPU306、CPU308 所允许的最大 I/O 点数均有明确的规定，若已添加的模块的点数超出了 CPU 所允许的限制，则软件将不允许继续添加模块，并且在保存、编译时会提示错误。

- 删除模块

鼠标单击模块配置表格中将要删除的模块，然后敲 Delete 键删除即可，或者单击鼠标右键，执行〔删除模块〕命令即可。

4.7.3 模块参数的配置

K3 系列 PLC 的每种模块都提供了多种参数和选项设置以适应不同的具体应用，包括模块的 I/O 地址、模拟量模块各通道的信号类型等等，在 EasyProg 软件允许用户自定义所有的这些参数和选项。

鼠标单击模块列表中的某项，或者使用键盘上的向上、向下键移动到某项，均会在下边的 PLC 模块参数配置窗口中自动切换到相应模块的参数配置页面。

在 PLC 模块参数配置区的右侧有两个公用的按钮：〔缺省值〕、〔取消〕。

〔缺省值〕：单击此按钮，将会取消当前页面用户的输入，软件为本模块自动分配参数。

〔取　消〕：单击此按钮，将会取消当前页面用户的输入，恢复本模块原有的配置。

在配置时请注意：各模块在同一区域（I、Q、AI 或者 AQ）内的地址不允许重叠！

4.7.3.1 CPU 参数配置

① 〔I/O 设置〕页面

在此页面中可以完成 CPU 本体 I/O 的相关配置。如图 4-20。

图 4-20　CPU 本体 I/O 参数配置页面

• 输入区

用于配置 CPU 本体集成的 DI 点的参数。

►〔起始地址〕：指定 DI 部分在 I 区中所占用地址的起始字节。

►〔输入滤波〕：为部分或者全部的 DI 点选择输入滤波延时，延时范围为〔0.2，12.8〕ms。

CPU 上所有的 DI 点以 8 个为单位被分组，系统允许为前两组选择此项配置。

来自于现场的 DI 信号至少要保持所配置的延时时间方可被 CPU 认为有效，这样有助于滤除输入噪声，增强系统的抗干扰能力。

• 输出区

用于配置 CPU 本体集成的 DO 点的参数。

►〔起始地址〕：指定 DO 部分在 Q 区中所占用地址的起始字节。

►〔输出保持〕：设置当 CPU 处于 STOP 状态时各 DO 点的输出状态。

将某 DO 点对应的复选框设置为  ，则当 CPU 处于 STOP 时，该点输出为 1。

将某 DO 点对应的复选框设置为  ，则当 CPU 处于 STOP 时，该点输出为 0。

此项功能对于 CPU 停机后所需要的安全连锁非常有意义。

② 〔通讯设置〕页面

用于配置 CPU 本体所带串行通讯口（Port0、Port1）的参数。如图 4-21。

图 4-21　CPU 本体串口参数配置页面

- Port0

► 〔站号〕　：为 Port0 指定一个唯一的站号。该站号作为 Modbus 从站时的站号。

► 〔波特率〕：设置串行通讯的波特率，其值有：2400、4800、9600、19200、38400。

► 〔奇偶校验〕：设置串行通讯的奇偶校验方式，其值有：无校验、奇校验、偶校验。

► 〔数据位〕：设置串行通讯的数据位，其值有：7、8、9。

► 〔停止位〕：设置串行通讯的停止位，其值有：1、2。

- Port1

请参见上面关于 Port0 的参数的描述。

在 CPU304 和 CPU306 中只有一个串口，不提供 Port1。

- CPU 通讯处理时间

在 3.8 CPU 中程序的执行 部分提到过，在每个扫描周期中 CPU 都要处理用户的通讯请求。〔CPU 通讯处理时间〕就是用来设置通讯处理允许占用 CPU 扫描周期的最大百分比，其范围是〔5%，50%〕。

③ 〔保持区域〕页面

当 CPU 掉电时，在此页面中定义的 4 个内存区域中的数据将由超级电容供电来得到保护，以供再次上电时使用。该页面如图 4-22。

图 4-22　保持区域参数配置页面

- ► 〔数据区〕 ：指定被保护的数据区所在的内存区，有 V、M、T、C 四个选项。
- ► 〔起始地址〕：指定该报持区域的起始字节地址。
- ► 〔长度〕　 ：指定该保持区域的长度，单位：字节。

④ 〔脉冲捕捉〕页面

用于指定 CPU 本体的前 14 个 DI 点是否启用脉冲捕捉功能。

K3 系列 CPU 提供了短脉冲的捕捉功能。由于 CPU 总在每个扫描周期的开始来读取物理 DI 信号的状态并更新输入映像区（I 区），因此，若物理 DI 信号在扫描周期中到达并且只持续非常短的时间时，CPU 有可能丢失这个信号。当一个 DI 点的脉冲捕捉被启用时，物理输入的状态变化会被锁存并一直保持到下一个扫描周期的的 I 区刷新。脉冲捕捉功能提高了 CPU 检测短脉冲信号的可靠性，其作用方式如图 4-23。

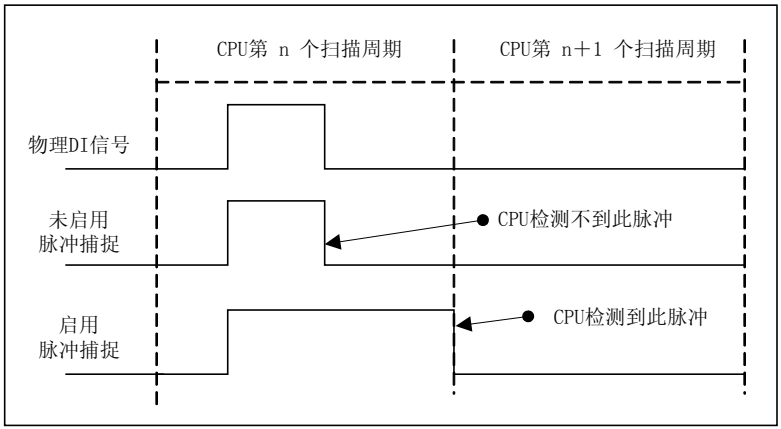


图 4-23　脉冲捕捉功能描述

另外，需要注意的是，物理 DI 信号首先通过输入滤波，然后再脉冲捕捉功能才能生效。因此，当启用脉冲捕捉功能时，必须要保证〔输入滤波〕的时间要小于脉冲的持续时间以免脉冲被滤掉。

〔脉冲捕捉〕页面如图 4-24。

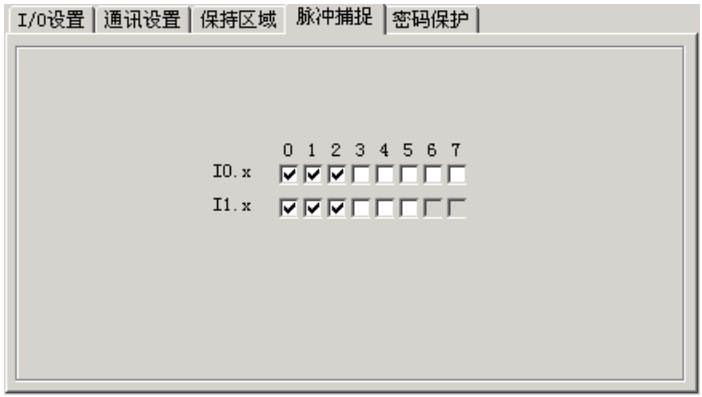


图 4-24　脉冲捕捉参数配置页面

在页面上，若某个 DI 点对应的复选框被选中，则该 DI 点的脉冲捕捉功能被启用，否则就被禁用。如上图，I0.0、I0.1、I0.2、I1.0、I1.1、I1.2 对应通道的脉冲捕捉功能被启用

4.7.3.2　DI 模块的参数配置

DI 模块的参数配置非常简单，仅需配置其地址即可，如图 4-25 所示。

图 4-25　DI 模块参数配置页面

►〔起始地址〕：指定该 DI 模块所有通道在 I 区中所占用地址的起始字节。

4.7.3.3　DO 模块的参数配置

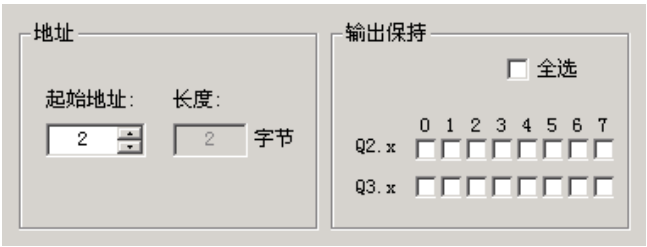


图 4-26　DO 模块参数配置页面

►〔起始地址〕：指定 DO 模块所有通道在 Q 区中所占用地址的起始字节。

►〔输出保持〕：设置当 CPU 处于 STOP 状态时该模块各 DO 点的输出状态。

将某 DO 点对应的复选框设置为☑，则当 CPU 处于 STOP 时，该点输出为 1。

将某 DO 点对应的复选框设置为☐，则当 CPU 处于 STOP 时，该点输出为 0。

此项功能对于 CPU 停机后所需要的安全连锁非常有意义。

4.7.3.4　DIO、DI/O 模块的参数配置

DIO 模块是指该模块上的所有通道均可以作为 DI 或者 DO 点使用。模块上的每个点都具有 DI 点、DO 点两种特性，用户不需要作任何配置，直接将其作为 DI 或者 DO 点使用即可。

DI/O 模块是指在该模块上集成了若干个 DI 点和若干个 DO 点，DI、DO 点不能混用。

DIO、DI/O 模块的参数配置界面是一样的，如图 4-27 所示。

图 4-27　DIO、DI/O 模块参数配置页面

界面上所有参数的含义请参见上面关于 DI、DO 模块配置的说明。

4.7.3.5　AI 模块的参数配置

图 4-28　AI 模块参数配置页面

① 参数说明

►〔起始地址〕：指定该 AI 模块所有通道在 AI 区中所占用地址的起始字节。

每个 AI 点在 AI 区占用 2 个字节。因此，地址必须为偶数。

►〔信号形式〕：为各个通道选择所接入的信号形式。

可选的信号形式取决于所选的模块类型，有电流/电压、热电阻、热电偶等。

关于各种信号数据转换格式的说明请参见　　。

►〔滤波方式〕：为各个通道选择软件滤波器。

对于变化较快的模拟量信号使用滤波器可以让其 AI 值变得比较稳定。

注意：若系统需要对某 AI 信号快速地响应则不应该启用该点的软件滤波器。

② 关于软件滤波器的说明

软件滤波器的输出就是对一定数量的信号采样值取平均值。软件滤波器有如下选项：

► *无*　　--- 不启用软件滤波器。

► *算术平均* --- 对一定数量的信号采样值取算术平均值。

► *中值平均* --- 将一定数量的信号采样值去掉最大、最小值后，对剩下的数取平均值。

4.7.3.6　AO 模块的参数配置

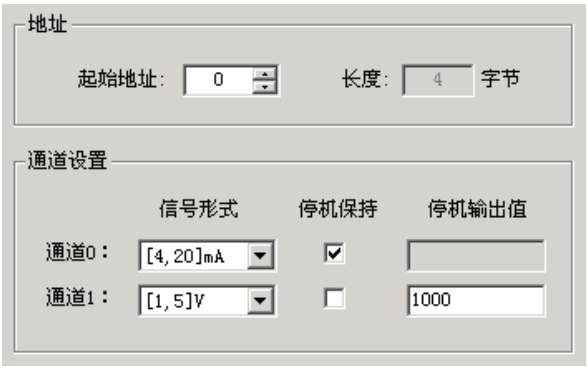


图 4-29　AO 模块参数配置页面

►〔起始地址〕：指定该 AO 模块所有通道在 AO 区中所占用地址的起始字节。

每个 AO 点在 AO 区占用 2 个字节。因此，地址必须为偶数。

►〔信号形式〕：为各个通道选择所接入的信号形式。

可选的信号形式取决于所选的模块类型，有电流/电压等。

关于各种信号数据转换格式的说明请参见　　。

►〔停机保持〕：指定当 CPU 处于 STOP 状态时，该点是否保持原来的输出。

若将该电对应的复选框设置为☑，则该点采用停机保持方式。

若将该电对应的复选框设置为☐，则该点不采用停机保持方式。

►〔停机输出值〕：若该点不设置为停机保持方式，则在此设置停机时该点的输出值。

此处应当输入经过线性转换之后的数值而非实际的信号值。例如，若用户选择信号形式为〔1,5〕V，停机时希望输出 1V，则由于 PLC 将〔1,5〕V 线性转换为〔1000,5000〕，因此用户应当在此处输入 1000。

4.8 初始化数据表

在初始化数据表中用户可以为 V 区中的 BYTE、WORD、DWORD、INT、DINT、REAL 类型的数据指定初始值。其样式如图 4-31。

初始化数据表

起始地址	数据类型	初始值	初始值	初始值	初始值
%VW0	INT	5	6	7	8
%VB10	BYTE	B#21	B#22		
%VD100	DWORD	DW#2	DW#12		DW#32

图 4-31　初始化数据表

表中共分六列：〔起始地址〕、〔数据类型〕以及连续 4 列〔初始值〕。

一个典型的行的含义是：为从〔起始地址〕开始、指定〔数据类型〕的一个或连续多个（最多为 4 个）V 区直接变量赋予指定的〔初始值〕。在连续赋值时，〔初始值〕列中不允许为空，否

则其后续的〔初始值〕均被认为无效。

如图 4-31 中，第一行的含义是 VW0、VW2、VW4、VW6 均被设置为 INT 型，并分别赋予初值 5、6、7 和 8；第二行的含义是 VB10、VB11 均被设置为 BYTE 型，并分别赋予初值 B#21、B#22；第三行的含义是 VD100、VD104 被设置为 DWORD 型，并分别赋予初值 DW#2、DW#12。

4.8.1 如何进入“初始化数据表”窗口?

有如下两种方式可以进入“初始化数据表”窗口：

- 双击工程管理器中【程序】组下的〔初始化数据表〕节点；
- 在〔初始化数据表〕节点上单击鼠标右键，然后执行弹出的〔打开…〕菜单命令。

4.8.2 在初始化数据表中进行编辑

- 输入数据

鼠标单击某表格单元即可让该单元进入编辑状态。

敲任意键也可以让具有焦点的表格单元进入编辑状态。

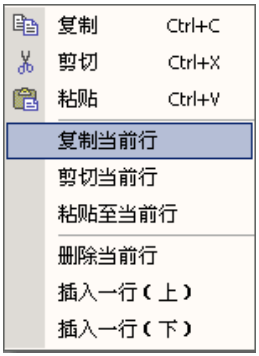
输入结束后，须敲 Enter 键，或者鼠标单击输入单元之外的表格任意地方来进行确认。

若输入的数据有错误，将会自动变成红色进行提示。错误的数据在保存时会被略掉。

用户在输入〔初始值〕时，可以输入任何合法的常量，在需要时软件会根据所选的〔数据类型〕自动对用户输入的数据进行类型转换。

- 右键菜单

在表中任意一个单元单击鼠标右键，将会弹出如下菜单：



- 〔复制〕：将具有焦点的表格单元的内容复制至剪贴板中，快捷键 Ctrl+C。
- 〔剪切〕：删除具有焦点的表格单元的内容，并将其复制至剪贴板中，快捷键 Ctrl+X。
- 〔粘贴〕：将剪贴板中的内容复制至具有焦点的表格单元中，快捷键 Ctrl+V。
- 〔复制当前行〕：复制焦点所在行中各表格单元的内容。
- 〔剪切当前行〕：剪切焦点所在行中各表格单元的内容。
- 〔粘贴当前行〕：将剪贴板中内容复制至焦点所在行的各表格单元中。
- 〔删除当前行〕：删除焦点所在的行。
- 〔插入一行（上)〕：在焦点所在行的上一行的位置插入一个新的空行。
- 〔插入一行（下)〕：在焦点所在行的下一行的位置插入一个新的空行。

4.9　全局变量表

在全局变量表中用户可以方便地完成全局变量的定义，在 IEC61131-3 中，全局变量的关键字是 VAR_GLOBAL。全局变量表窗口分为〔变量〕和〔功能块实例〕两部分页面。

- 〔变量〕页面

用于定义直接存取 PLC 内直接地址的符号变量。样式如下图所示。


VAR_GLOBAL

序 号	名 称	地 址	数据类型	注 释
1	Motor1_Err	%I0.0	BOOL	1#电机故障信号
2	Motor1_Run	%I0.1	BOOL	1#电机运行信号
3	Tank1_Level	%AIW0	INT	1#罐液位
4				
5				
6				
7				
8				

变量　功能块实例

图 4-32　全局变量表中的〔变量〕页面

符号变量可以等效地代替对应的 PLC 直接地址在程序中使用，从而能够保证用户程序具有良好的可读性。每一个直接地址只允许赋予一个符号变量名，同样地，每一个符号变量名只允许有一个对应的直接地址。

表中分四列：〔名称〕、〔地址〕、〔数据类型〕、〔注释〕。

一个典型的行的含义是：为输入的〔地址〕定义一个符号变量〔名称〕，并指定其〔数据类型〕，也可以为这个符号变量写一些〔注释〕，当然，〔注释〕是可有可无的。比如，图 4-32 的第 1 行的含义是：为地址“%V20.0”定义一个符号变量，变量名称为“PreviousT20”，其数据类型为“BOOL”。

符号变量的命名规则请参见 3.4.1 标识符的定义 部分。

- 〔功能块实例〕页面

用于定义功能块的实例。样式如下图所示。

图 4-33　全局变量表中的〔功能块实例〕页面

在 3.6.5 FB 实例的命名及使用 提到过，为了方便用户的使用，功能块实例的定义是由 EasyProg 自动完成的，因此在〔功能块实例〕页面中，〔FB 实例名称〕以及〔FB 类型〕是不可编辑的，用户仅可以在〔注释〕部分输入对于该实例的有关说明。由于功能块实例定义区的操作非常简单，因此下面将略过不予介绍。

4.9.1 如何进入全局变量表窗口?

有如下三种方式可以进入全局变量表窗口:

- 双击工程管理器中【资源】组下的〔全局变量表〕节点;
- 在〔全局变量表〕节点上单击鼠标右键，然后执行弹出的〔打开…〕菜单命令。
- 执行【查看】→〔全局变量表〕菜单命令。

4.9.2 在全局变量表窗口中进行编辑

- 输入数据

鼠标单击某表格单元，即可让该单元进入编辑状态。

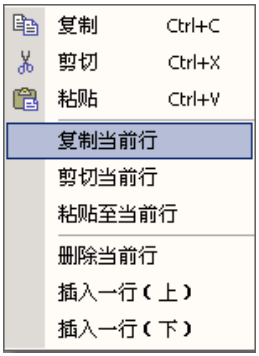
敲任意键也可以让具有焦点的表格单元进入编辑状态。

输入结束后，须敲 Enter 键，或者鼠标单击输入单元之外的表格任意地方来进行确认。

若输入的数据有错误，将会自动变成红色进行提示。错误的数据在保存时会被略掉。

• 右键菜单

在表中任意一个单元单击鼠标右键，将会弹出如下菜单：



► 〔复制〕：将具有焦点的表格单元的内容复制至剪贴板中，快捷键 Ctrl+C。

► 〔剪切〕：删除具有焦点的表格单元的内容，并将其复制至剪贴板中，快捷键 Ctrl+X。

► 〔粘贴〕：将剪贴板中的内容复制至具有焦点的表格单元中，快捷键 Ctrl+V。

► 〔复制当前行〕：复制焦点所在行中各表格单元的内容。

► 〔剪切当前行〕：剪切焦点所在行中各表格单元的内容。

► 〔粘贴当前行〕：将剪贴板中内容复制至焦点所在行的各表格单元中。

► 〔删除当前行〕：删除焦点所在的行。

► 〔插入一行（上）〕：在焦点所在行的上一行的位置插入一个新的空行。

► 〔插入一行（下）〕：在焦点所在行的下一行的位置插入一个新的空行。

4.10 交叉索引表

交叉索引表包含如下部分：

• 〔交叉索引〕页面

用于分析当前工程中用到的所有地址变量并列表显示详细信息。

〔交叉索引〕页面如图 4-34 所示。

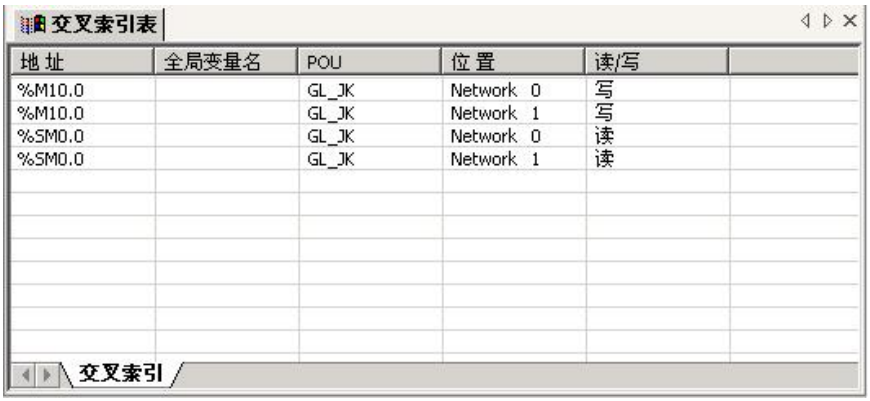


图 4-34　交叉索引表中的〔交叉索引〕页面

► 〔地址〕　　：显示了当前工程中用到的所有直接地址。

► 〔全局变量名〕：显示本行〔地址〕所对应的全局变量名。

► 〔POU〕　　：显示本行〔地址〕所在的 POU 名称。

► 〔位置〕　　：指明本行〔地址〕在〔POU〕中的具体位置。

若 POU 用 IL 编写，则显示的是行号；若用 LD 编写，则显示的是网络号。

► 〔读/写〕　　：指明本行〔地址〕在所处的〔位置〕上被进行了读或者是写操作。

如图 4-34，表格中第一行的意思是：在本工程 *GL_JK* 程序的 *Network 0* 中用到了 一次 *%M10.0*，此处对*%M10.0* 进行的是 *写* 操作，*%M10.0* 没有被定义全局变量名称。

交叉索引表中的信息在在第一次编译之后才能生成，并在以后的编译过程中自动刷新。

4.10.1　如何进入交叉索引表?

有如下三种方式可以进入全局变量表窗口：

- 执行【查看】→〔交叉索引表〕菜单命令；
- 鼠标单击工具栏上的 图标。

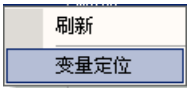
4.10.2　在交叉索引表中进行操作

- 对表格的操作

鼠标双击某一行，即可打开本行的〔POU〕并定位到〔地址〕所在的具体〔位置〕上。

- 右键菜单

在表格任何一行上单击鼠标右键，将弹出如下右键菜单：



► 〔地　　址〕：刷新显示交叉索引数据。

► 〔变量定位〕：打开本行的〔POU〕并定位到〔地址〕所在的具体〔位置〕上。

4.11 变量状态表

用户可以利用变量状态表来对当前工程中用到的任意地址变量进行在线监测和强制。变量状态表的样式如下图所示。

变量状态表

序 号	地 址	变量名	格 式	当前值	强制值
1	%I1.2	test	BOOL	0	0
2	%I2.3		BOOL	0	0
3	%Q1.2		BOOL	0	0
4	%VW100		有符号数	0	
5					

图 4-35 变量状态表

表中共分如下五列：

► 〔地　址〕：输入将要被监测和强制的直接地址。

► 〔变量名〕：显示本行〔地址〕所对应的全局变量名。

► 〔格　式〕: 选择当前值和强制值的数据显示格式，

可选的格式有 BOO、REAL、有符号数、无符号数、二进制、16 进制等。

► 〔当前值〕: 在线监测时将显示监测到的本行〔地址〕的值。

► 〔强制值〕: 在线监测时可以在此输入本行〔地址〕将要被强制为的值。

提示:

为了提高效率，EasyProg 只允许对当前工程中用到的变量进行监测和强制。若用户输入了未被用到的变量，EasyProg 虽然不提示错误，但当前值、强制值均不会起作用。

4.11.1 如何进入变量状态表窗口?

有如下三种方式可以进入变量状态表窗口:

- 双击工程管理器中【资源】组下的〔变量状态表〕节点;
- 在〔变量状态表〕节点上单击鼠标右键，然后执行弹出的〔打开…〕菜单命令。
- 执行【查看】→〔变量状态表〕菜单命令。

4.11.2 在变量状态表中进行编辑

- 输入数据

鼠标单击某允许编辑的表格单元即可让该单元进入编辑状态。

敲任意键也可以让具有焦点的表格单元进入编辑状态。

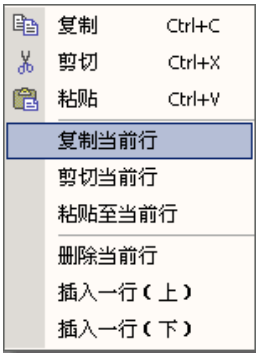
输入结束后，须敲 Enter 键，或者鼠标单击输入单元之外的表格任意地方来进行确认。

若输入的数据有错误，将会自动变成红色进行提示。错误的数据在保存时会被略掉。

用户在输入〔强制值〕时，可以输入任何合法的常量，在需要时软件会根据所选的〔显示格式〕自动对用户输入的数据进行格式的转换。

- 右键菜单

在表中任意一个单元单击鼠标右键，将会弹出如下菜单：



► 〔复制〕：将具有焦点的表格单元的内容复制至剪贴板中，快捷键 Ctrl+C。

► 〔剪切〕：删除具有焦点的表格单元的内容，并将其复制至剪贴板中，快捷键 Ctrl+X。

► 〔粘贴〕：将剪贴板中的内容复制至具有焦点的表格单元中，快捷键 Ctrl+V。

► 〔复制当前行〕：复制焦点所在行中各表格单元的内容。

► 〔剪切当前行〕：剪切焦点所在行中各表格单元的内容。

► 〔粘贴当前行〕：将剪贴板中内容复制至焦点所在行的各表格单元中。

► 〔删除当前行〕：删除焦点所在的行。

► 〔插入一行（上）〕：在焦点所在行的上一行的位置插入一个新的空行。

► 〔插入一行（下）〕：在焦点所在行的下一行的位置插入一个新的空行。

4.11.3 如何利用变量状态表强制变量?

① 进入变量状态表，输入有效的地址、强制值；

② 与 PLC 联机；

③ 执行【调试】→〔变量强制〕菜单命令，或者单击工具栏上的  图标，即可将变量状态表中的所有强制值写入相应的地址。

④ 执行【调试】→〔取消全部强制〕菜单命令，或者单击工具栏上的  图标，即可取消全部变量的强制状态。

注意：

KDN-K3 系列 CPU 采用强制优先模式！一定要记得在调试结束后取消强制状态!

第五章 LD 编程和 IL 编程

本章对 EasyProg 中 LD 和 IL 编辑器的功能、使用进行了详细的描述，同时也阐述了 IEC61131-3 标准中关于 LD、IL 语言的相关语法、规定。

通过阅读本章，用户可以轻松地使用 EasyProg 编写出符合 IEC 标准的应用程序。

针对 PLC 应用程序的编写，IEC61131-3 中规定了 3 种文本化语言和 3 种图形化语言。文本化语言包括：指令表（IL）、结构化文本（ST）、顺序功能图（SFC，文本化版本）；图形化语言包括：梯形图（LD）、功能块图（FBD）、顺序功能图（SFC，图形化版本）。

EasyProg 目前支持 IL 语言和 LD 语言编程。在一个工程中，IL 和 LD 语言可以混用，但是在一个 POU 中就只允许使用一种语言。

5.1 EasyProg 中的 POU 类型

在 3.7.1 工程的组织结构 部分中已经介绍了在 EasyProg 共有三类 POU：主程序、子程序、中断服务程序。其中主程序、中断服务程序即为 IEC61131-3 中规定的 PROGRAMME 类型，可以作为任务在 CPU 中运行；而子程序相当于用户自定义的 FB。

5.1.1 主程序

主程序是 CPU 的主循环任务，在 CPU 的每次循环中均扫描、执行主程序一次。

主程序没有参数，但允许使用局部变量。

5.1.2 子程序

子程序是可重用的代码段，使用子程序可以更好地组织应用程序的结构，方便调试和维护，

有效地缩短程序代码的长度。

在子程序中可以使用局部变量，也可以有形式参数。在每次调用带参数的子程序时可以将不同的实际变量赋给形式参数，从而能够对不同的变量、数据进行相同的处理运算。

参数、局部变量的命名规则请参见 3.4.1 标识符的定义 。

参数、局部变量允许使用如下几种变量类型：

- VAR：局部变量。
- VAR_INPUT：输入参数。
- VAR_OUTPUT：输出参数。
- VAR_IN_OUT：输入、输出参数。

5.1.3 中断服务程序

在 KDN-K3 系列 PLC 中提供了中断处理功能用于快速响应内部或者外部特定的中断事件。响应某个中断事件而被执行的程序称为中断服务程序。每个中断事件都有一个中断事件号。若在程序中将中断事件号与中断服务程序联系起来，则当该事件号对应的中断事件发生时，CPU 将自动调用中断服务程序进行处理。

K3 系列 PLC 支持三类中断事件：通讯中断、I/O 中断、时基中断。中断事件各有不同的中断优先级。当中断事件发生时会按照优先级和发生的时序进行排队，队列中优先级高的中断事件首先得到处理，优先级相同的中断按照发生的时序进行处理。

一个中断事件只能对应一个中断服务程序，但一个中断服务程序可以对应多个中断事件。

中断服务程序没有参数，但允许使用局部变量。

5.2 IL 编程

5.2.1 IL 的背景

IL 语言是一种低级语言，与汇编语言非常相似，是在借鉴、吸收世界上各著名 PLC 厂商的

指令表语言的基础上而形成的一种标准语言。

在其它语言进行编译时，IL 是作为其中的公用中间语言。因此通常情况下，与其它语言相比，使用 IL 编写的程序效率更高、更加紧凑。

5.2.2　IL 的语法规定

5.2.2.1　IL 语句格式

IL 程序面向行，每行语句只能有一条指令。IL 程序中允许有空白行存在。

IL 程序中一个语句的基本格式如下图：

```
标号：
操作符/功能/功能块    操作数（表）    (* 注释 *)
```

图 5-1　IL 语句格式

- 〔标号〕

  可选。使用标号的目的就在于实现程序的跳转。标号的命名格式如同变量。

- 〔操作符/功能/功能块〕

  请参见下一章中对于各操作符、功能、功能块的详细说明。

- 〔操作数表〕

  在操作符、功能、功能块的说明中包含了对应的操作数的描述。

  在操作符、操作数之间至少有一个空格。

- 〔注释〕

  可选。在所有的语言中，注释的格式都是相同的，由 (*　*) 进行界定。

  每行只允许有一个注释，注释不允许嵌套，嵌套的注释将被编译器认为是错误。

举例如下：

```
(* NETWORK 0 *)
Run:                    (* 语句标号，供跳转时使用 *)
LD      %I1.0
TP      T2, 168         (* 若 I1.0 = true，启动定时器 T2，T2 被声明为 TP 型 *)
```

5.2.2.2　CR 值

IL 中提供了一个“当前结果”（CR，Current Result）累加器，在 CR 中存储了用户程序的当前执行结果。用户程序中的每一行语句执行后，CR 值都会被“刷新”。CR 值可以是编程软件支持的任意数据类型，这取决于刚刚执行完的语句。依据后续语句的不同，CR 值可能作为下一条语句的执行条件，也可能作为下一条语句的操作数之一。

操作符不同，执行之后对 CR 值的影响也不同。下表根据对 CR 值不同的影响将 EasyProg 中的操作符进行了初步的分组。更详细的描述请参见下一章对于各指令的介绍。

操作符	分组缩写	对 CR 值的影响
LD，LDN	C	CR 值重新建立
逻辑指令，比较指令等	P	CR 值被更新为操作结果
ST，R，S，JMP，JMPCN，JMPC 等	U	CR 值保持不变

表 5-1　各操作符执行之后对于 CR 值的影响

在 IEC61131－3 中并没有完善地定义各种操作符对于 CR 的影响，因此在不同的编程系统中这些定义可能有所不同。

5.2.2.3　网络

与其它语言相同，IL 编写的 POU 包括如下部分：

- POU 的起始和结束指令。编程软件可以要求用户输入这一部分，也可以将其自动生成。
- 参数、变量声明部分。

- 代码部分。

在 IL 程序中也存在着网络（Network）的概念，以网络作为基本的段落，用户程序的代码部分就是由若干个网络组成。一个典型的网络包括：

- 网络标号。
- 网络中的语句。

5.2.3　在 EasyProg 中使用 IL 编程

当建立一个新程序时，若选择其语言为“指令表（IL）”，就将进入 IL 编辑器进行编程；若打开一个用 IL 编写的程序，也将进入 IL 编辑器。IL 编辑器的外观如下图。

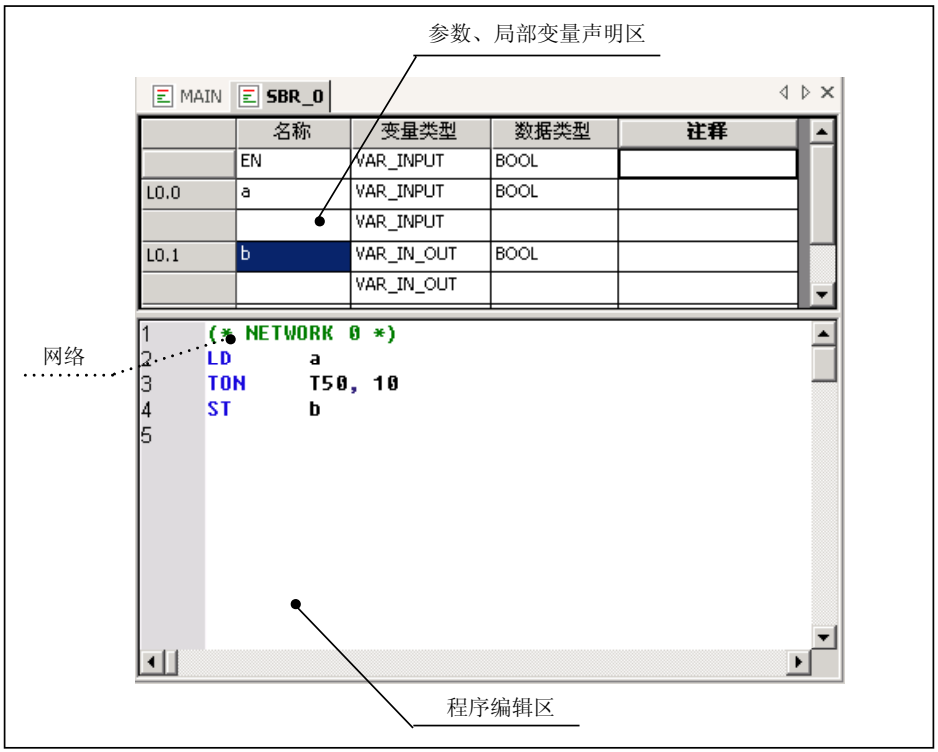


图 5-2　IL 编辑器

编辑器分为两部分区域：

- 参数、局部变量声明区：用于声明 POU 的参数和 POU 内用到的局部变量。支持右键菜单。
- 程序编辑区：用户在此编写自己的应用程序。

5.2.3.1　添加一个网络

在 IL 程序中添加一个网络的方法如下：

- 执行 【IL/LD】→〔IL 加入网络〕 菜单命令；
- 单击工具栏上的  图标；
- 单击鼠标右键，执行弹出的 〔IL 加入网络〕 菜单命令。

5.2.3.2　网络中允许的格式

① 在一个网络中可以只有一个语句标号。例如：

```
(* NETWORK 0 *)
MRun:    (* 可以只有一个标号 *)
```

② 在一个网络中可以只有程序语句。

在 表 5-1 各操作符对于 CR 值的影响 中，我们将操作符分了“C”、“P“U”几类。

网络中的程序必须以“C”组指令开始。

网络中的程序必须以“U“组指令或者功能、功能块的调用结束。

举例如下：

```
(* NETWORK 0 *)
LD %M3.5   (* 以 LD 指令开始 *)
… …        (* 用户可以输入其它指令 *)
ST %Q2.3   (* 以允许作为结尾的指令结束 *)
```

③　在一个网络中可以有语句标号和程序语句。

程序语句的起始、结束必须遵循在②中的原则。

举例如下：

```
(* NETWORK 0 *)
MRun:
LD %M3.5   (* 以 LD 指令开始 *)
… …        (* 用户可以输入其它指令 *)
ST %Q2.3   (* 以允许作为结尾的指令结束 *)
```

5.2.3.3　IL 编辑器中的操作

IL 编辑器能自动地对用户输入的语句进行格式化。若输入的语句有错误，则会在该行的开头显示一个红色的问号（? ）。

IL 编辑器类似于一个文本编辑器，支持通常的键盘操作，比如删除（Del）、退格（BackSpace）、光标移动、移动光标时同时按下 Shift 键可以选中文本等。

在 IL 编辑器中可以使用【编辑】菜单中的所有编辑命令。

在程序编辑区中单击鼠标右键，将会弹出如下右键菜单：

这些菜单命令在前面已经介绍国，在此不再详述。

5.2.3.4　在线调试

若当前窗口是 IL 编辑器，则执行【调试】→〔在线监测〕菜单命令可让其进入在线监测状态。

若当前已经处于线监测状态，将当前窗口切换至 IL 编辑器窗口，则该 IL 窗口也将进入在线监测状态。

在线监测状态下是不允许进行编辑的。

在线监测状态下，在原程序编辑区中将分为两栏，右边显示程序，左边显示相应的变量值，中间以一条竖线分割开。将光标置于竖线上，待其形状变为 ↔ 后，即可拖动该线改变左、右区域的大小。

5.2.3.5　IL 程序示例

下面这段程序让 T0、T1 相互循环启动，目的是在 T0 的输出端输出一周期为 2 秒的方波。

```
(* NETWORK 0 *)
LDN     %M0.0
TON     T0, 1000        (*  依靠 T1 的输出来启动 T0，定时为 1000×1ms *)
ST      %M0.1

LD      %M0.1
TON     T1, 1000        (*  依靠 T0 的输出来启动 T1，定时为 1000×1ms *)
ST      %M0.0

LD      %M0.1
ST      %Q0.0           (*  在 Q0.0 输出方波  *)
```

5.3 LD 编程

5.3.1 LD 的背景

LD（梯形图）语言是 IEC61131-3 标准中规定的五种编程语言之一，是 PLC 编程中被最广泛使用的一种图形化语言。

LD 语言源于机电一体化领域中图形表示的继电器逻辑，用它编写的程序能够与实际的电气操作原理图相对应，非常直观，易于学习和掌握。LD 语言的长处在于布尔逻辑的处理。下图是用 LD 编写的一段简单程序。

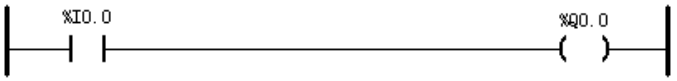


图 5-3 LD 程序示例

5.3.2 LD 的网络结构

如同 IL 一样，在 LD 中也使用“网络（Network）”的概念，以网络作为基本的段落。

一个 LD 网络的边界是位于左、右两侧的电源线（Power Rails）。左侧的电源线相当于“火线”，右侧的电源线相当于“零线”。

我们可以对 LD 程序作如下形象的描述：电能自左侧的“火线”沿着连接线流经线上所有的元件（包括触点、线圈、功能、功能块等），这些元件依据自身的逻辑状态，或是中断能量流，或是将电能传输到后继的元件并最终到达右侧的“零线”。

5.3.3 LD 的图形对象

LD 网络中可以包括如下图形对象：

① 连接

LD 中使用水平连接线和垂直连接线，分别对应着串联和并联的关系。

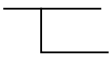
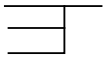
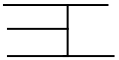
图形对象	名 称	描 述
	水平连接	可以将其理解为一根理想的导线。 在水平连接线上任一点处的值（true 或 false）都是相同的。
	垂直连接 （含有水平连接）	垂直连接线左侧的值被传递到右侧所有的水平连接上。
		垂直连接线左侧是并联的水平连接线。 左侧所有水平连接的值首先进行“或”运算，然后再将运算结果传递到垂直连接线的右侧。
		相当于上述两种垂直连接的组合。 左侧所有水平连接的值首先进行“或”运算，然后再将运算结果传递到垂直连接线右侧所有的水平连接上。

表 5-2　LD 中的连接

② 接点

接点有两种类型：常开接点和常闭接点。

每个接点都必须对应一个有效的布尔型变量，变量名称被写在图形元素的上面，该变量的值决定了接点的状态（闭合或者断开），并进而决定了接点所在线路的通或者断。

图形对象	名 称	描 述
	常开接点	若变量的值为 TRUE，则接点闭合，线路导通； 若变量的值为 FALSE，则接点断开，线路断开。
	常闭接点	若变量的值为 TRUE，则接点断开，线路断开； 若变量的值为 FALSE，则接点闭合，线路导通。

表 5-3　LD 中的接点

③ 线圈

线圈是用于给布尔型变量赋值的图形元素。

图形对象	名 称	描 述
变量名 —( )—	线圈	将左连接的值传递至变量
变量名 —(/)—	取反线圈	将左连接的值取反，然后传递至变量
变量名 —(S)—	置位线圈	若左连接的值为 true，则将变量置为 true； 若左连接的值为 false，则变量值保持不变。
变量名 —(R)—	复位线圈	若左连接的值为 true，则将变量置为 false； 若左连接的值为 false，则变量值保持不变。

表 5-4　LD 中的线圈

④　程序执行顺序控制

为了定义程序的执行顺序，在 LD 中提供了如下两种类型的控制执行顺序的元素：

图形对象	名 称	描 述
├—<RETURN>	无条件返回	无条件脱离当前 POU 并返回到调用 POU 中
├— (1) —<RETURN>	条件返回	若 RETURN 元素左边连接的值为 true，则脱离当前 POU 并返回到调用 POU 中；否则将其略过。
├—>> 语句标号	无条件跳转	无条件跳转到指定的“语句标号”处。
├— (1) —>> 语句标号	条件跳转	若元素左连接的值为 TRUE，则跳转到指定的“语句标号”处；否则将其略过。

表 5-5　LD 中的执行顺序控制元素

注：

(1)　表示此处存在结果为布尔型的图形代码。

⑤　调用功能和功能块

LD 支持对功能和功能块的调用。被调用的功能和功能块以矩形框来表示，其形参显示在矩形框内，实参显示在矩形框外部的连接线上。

功能块的参数中至少有一个 BOOL 型的输入参数和一个 BOOL 型的输出参数，用于将功能块接至连接线上。

功能必须有一个名为 **EN** 的 BOOL 型输入和一个名为 **ENO** 的 BOOL 型输出，用于控制该功能的执行。若 **EN** 的值为 TRUE，则该功能被执行且 **ENO** 也将被置为 TRUE；若 **EN** 的值为 FALSE，则不执行该功且 **ENO** 也将被置为 FALSE。

图 5-4　功能块和功能调用的示例

5.3.4　在 EasyProg 中使用 LD 编程

当建立一个新程序时，若选择其语言为“**梯形图（LD）**”，就将进入 LD 编辑器进行编程；若打开一个用 LD 编写的程序，也将进入 LD 编辑器。LD 编辑器的外观如下图。

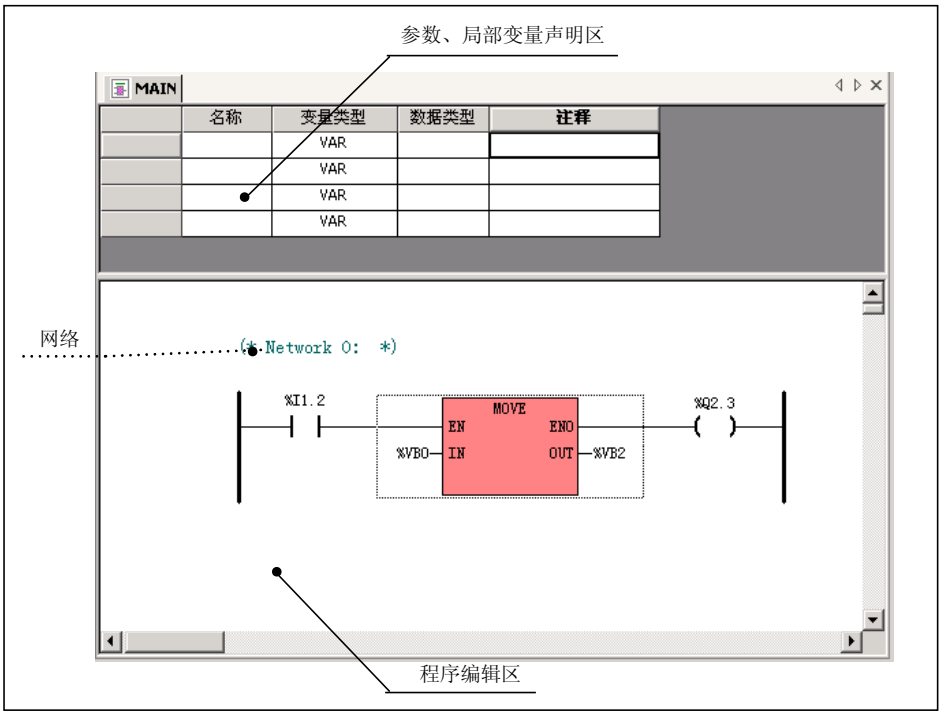


图 5-5　LD 编辑器

5.3.4.1　LD 程序的限制

在每一个 LD 程序中，最多不超过 100 个网络。

在 LD 程序中，每条连接线路径上的元件数量限制：若只有线圈、触点，则建议不超过有 31 个触点加 1 个线圈；若只有功能/功能快，则建议不超过 12 个块加 1 个线圈加 1 个触点。

在 LD 程序中，不允许出现单独两个功能/功能块并联的情况。

5.3.4.2　LD 编程步骤

① 首先执行相应的命令加入一个网络（network），如图 5-6。

双击网络标号部分，即可弹出对话框，用户可以输入该网络的注释。

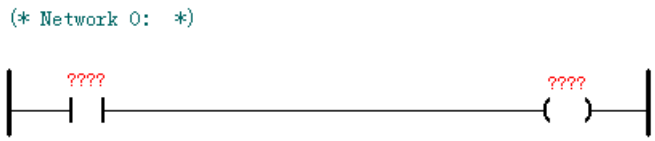


图 5-6　LD 编程第一步：加入网络

② 双击接点或者线圈将弹出如下对话框，可修改该元件的类型、连接的变量：

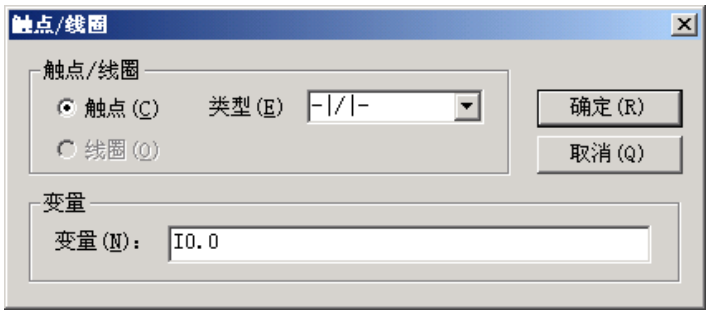


图 5-7　LD 编程：修改接点或者线圈的属性

③ 单击某元件将其选中作为基准，可以继续执行菜单或工具栏命令或快捷键加入其它元件；选好基准后，也可以双击右侧“**LD 指令集**”中的相应指令将其加入；或者右键单击某元件，执行相应的弹出菜单命令也可。 下面是加入一个 MOVE 功能之后的图形。

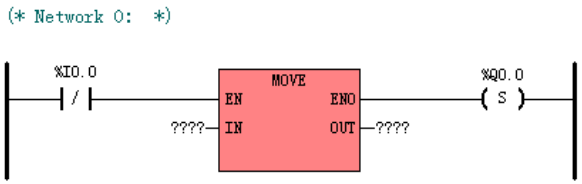
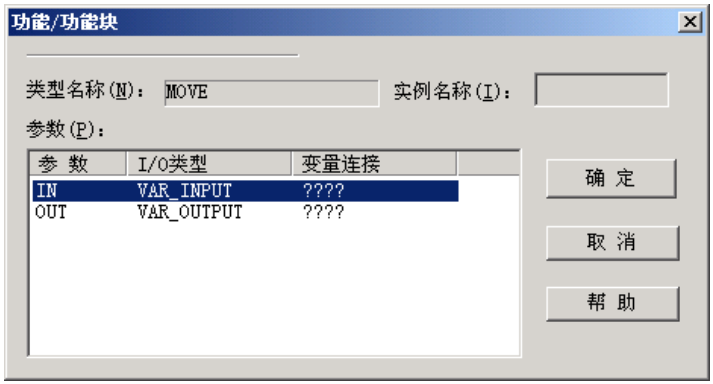


图 5-8　LD 编程：继续加入其它的元件

④　双击程序中的功能/功能块，将弹出如下对话框可以修改功能/功能块的属性：



双击〔**参数（P）**〕中的任何一项参数，即可修改该参数连接的变量：或者使用上、下方向键选择某项参数，敲 Enter 键，也可修该参数连接的变量。使用 Tab 键可以在各个按钮、列表之间切换焦点。

软件将对用户的输入进行严格的语法检查，并将对错误进行提示。

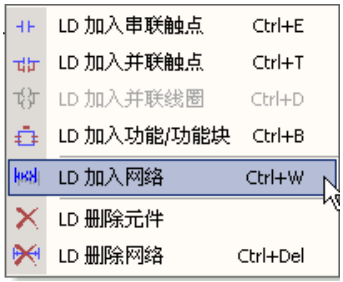
图 5-9　LD 编程：继续修改其它元件的属性

⑤　继续进行操作，包括使用菜单命令、工具栏命令、快捷键、右键菜单命令或者利用右侧的“LD 指令集”，直到完成 LD 程序的编写。

5.3.4.3　使用鼠标和键盘进行操作

鼠标左键单击某元件可将其选中并将焦点置于其上（为该元件“套”上一个矩形框），双击可以打开其属性对话框修改属性。

鼠标右键单击某元件会弹出如下右键菜单：



所有的菜单命令均已在前面描述过，此处不再赘述。

在 LD 编辑器内，使用键盘上的上、下、左、右方向键可以移动焦点；敲 Enter 键，则可以打开具有焦点的元件的属性对话框以修改其属性；敲 Delete 键可以删除具有焦点的元件；另外，各菜单命令都有对应的快捷键，使用快捷键与使用菜单命令等效。

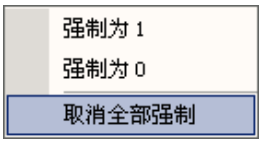
5.2.3.5　在线调试

若当前窗口是 LD 窗口，则执行【调试】→〔在线监测〕菜单命令可让其进入在线监测状态。

若当前已经处于线监测状态，将当前窗口切换至 LD 编辑器窗口，则该 LD 窗口也将进入在线监测状态。

在线监测状态下是不允许进行编辑的。

在线监测状态下，在接点或者线圈上单击鼠标右键，将弹出如下右键菜单：



- 〔强制为 1〕：将当前接点或者线圈连接的变量强制为 TRUE。
- 〔强制为 0〕：将当前接点或者线圈连接的变量强制为 FALSE。
- 〔取消全部强制〕：取消全部变量的强制状态。

5.2.3.6　LD 程序示例

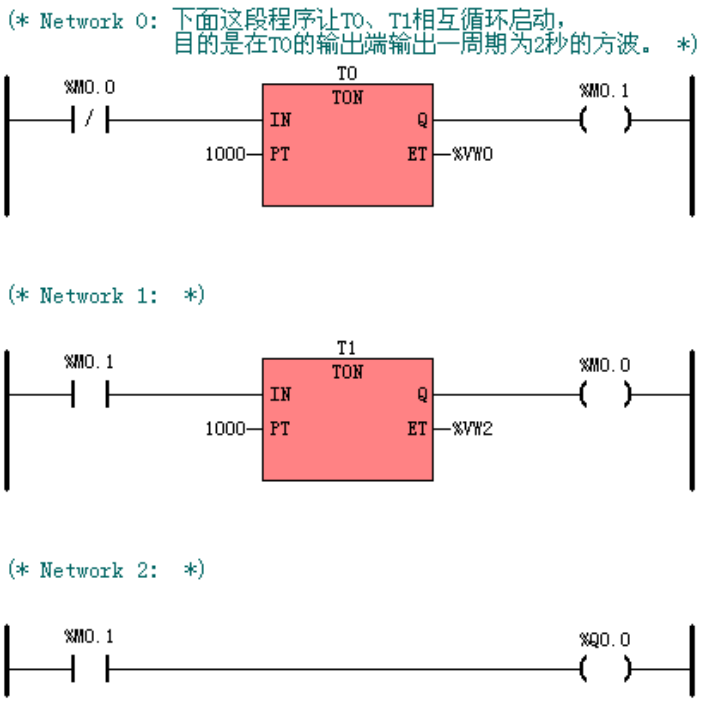


图 5-10　LD 程序示例