

Haiwell（海为）PLC

编 程 手 册

目录

- 第一章、产品简介10
 - 1.1、Haiwell（海为）PLC 的特点.....10
 - 1.2 编程软件特点10
- 第二章、快速入门11
 - 编程的一般流程11
 - 第一步：启动编程软件.....12
 - 第二步：新建程序项目.....13
 - 第三步：编写控制程序.....14
 - 第四步：PLC 硬件配置15
 - 第五步：脱机仿真调试.....15
 - 第六步：与 PLC 联机16
 - 第七步：下载程序17
 - 第八步：启动 PLC17
- 第三章、PLC元件与数据.....17
 - 3.1 概述17
 - 3.2 数据17
 - 3.3 元件一览表18
 - 3.4 开关量输入 [X].....19
 - 3.5 开关量输出 [Y].....19
 - 3.6 计时器 [T].....19
 - 3.7 计数器 [C].....19
 - 3.8 内部继电器 [M].....19
 - 3.9 步进继电器 [S].....19
 - 3.10 系统状态位 [SM].....20
 - 3.11 局部继电器 [LM].....20
 - 3.12 模拟量输入 [AI].....20
 - 3.13 模拟量输出 [AQ].....20
 - 3.14 计时器当前值 [TV].....20
 - 3.15 计数器当前值 [CV].....20
 - 3.16 内部寄存器 [V].....20
 - 3.17 系统寄存器 [SV].....20
 - 3.18 局部寄存器 [LV].....20
 - 3.19 变址指针 [P].....21
- 第四章、PLC 指令集.....21
 - 4.1 指令通用说明25
 - 4.2 比较开关26
 - 4.3 步进指令26
 - STL（步开始）27
 - SFROM（步合并）28
 - STO（步转移）30
 - 4.4 位指令31
 - AND（逻辑与）31
 - OR（逻辑或）32
 - XOR（逻辑异或）32
 - OUT（线圈输出）32
 - SET（置位保持）33
 - RST（复位）33
 - ALT（ON/OFF 交替输出）33

ZRST (批量复位)	34
ENO (取 ENO 输出)	34
4.5 计时器	35
TON (延时开)	35
TOF (延时关)	36
TP (脉冲计时器)	37
4.6 计数器	38
CTU、D.CTU (加计数器)	38
CTD、D.CTD (减计数器)	39
CTUD、D.CTUD (加减计数器)	40
4.7 高速控制指令	40
RESH (IO 更新)	46
HHSC (高速计数器)	47
HCWR (写高速计数器)	51
SPD (速度侦测)	52
PWM (脉宽调制)	53
PLSY、D.PLSY (脉冲输出)	54
PLSR、D.PLSR (加减速脉冲输出)	55
ZRN (原点回归)	57
SETZ (设置电气原点)	62
PPMR (直线插补)	63
CIMR (圆弧插补)	66
SPLS (简单脉冲输出)	69
MPTO (多段脉冲输出)	70
SYNP (随动脉冲输出)	73
PSTOP (停止脉冲输出)	74
4.8 比较指令	75
CMP、D.CMP (比较)	75
ZCP、D.ZCP (区域比较)	76
MATC、D.MATC (数值匹配)	77
ABSC、D.ABSC (绝对凸轮比较)	78
BON (ON 位判定)	79
BONC、D.BONC (ON 位数量)	80
MAX、D.MAX (最大值)	81
MIN、D.MIN (最小值)	82
SEL、D.SEL (条件选择)	82
MUX、D.MUX (多路选择)	83
4.9 移动指令	84
LBST (低字节赋值)	85
HBST (高字节赋值)	85
MOV、D.MOV (移动)	86
BMOV (块移动)	87
FILL (填充)	88
XCH (字节交换)、D.XCH (寄存器交换)	89
BXCH (块交换)	90
SHL (位左移)	91
SHR (位右移)	93
WSHL (字左移)	95
WSHR (字右移)	96
ROL (位循环左移)	97

ROR (位循环右移)	99
WROL (字循环左移)	101
WROR (字循环右移)	102
BSHL (字节左移)	103
BSHR (字节右移)	104
ATBL (添加到队列)	106
FIFO (先进先出)	108
LIFO (后进先出)	108
SORT (数据排序)	109
4.10 数据转换指令	111
ENCO (编码器)	111
DECO (译码器)	112
BTOW (位到字)	113
WTOB (字到位)	114
HEX、HEX.LB (ASCII 转换为 HEX)	114
ASCI、ASCI.LB (HEX 转换为 ASCII)	116
BUNB (离散位组合到连续位)	117
BUNW (离散位组合到连续字)	118
WUNW (离散字组合到连续字)	119
BDIB (连续位分散到离散位)	120
WDIB (连续字分散到离散位)	121
WDIW (连续字分散到离散字)	122
BCD、D.BCD (BIN 转换为 BCD)	123
BIN、D.BIN (BCD 转换为 BIN)	124
ITOL (整数转换为长整数)	125
GRAY (BIN 转换为 GRAY 码)	125
GBIN (GRAY 码转换为 BIN)	126
4.11 字符指令	128
GHLB (得到高低字节)	128
GETB (截取字节串)	129
BCMP、BCMP.LB (字节串比较)	130
ITOC、D.ITOC (整数转换为字符)	131
CTOI (字符转换为整数)	131
FTOC (浮点数转换为字符)	132
CTOF (字符转换为浮点数)	133
4.12 数学运算指令	134
WNOT、D.WNOT (取反)	135
WAND、D.WAND (与运算)	136
WOR、D.WOR (或运算)	136
WXOR、D.WXOR (异或运算)	137
ADD、D.ADD (加法)	137
SUB、D.SUB (减法)	138
INC、D.INC (加 1)	139
DEC、D.DEC (减 1)	139
MUL、D.MUL (乘法)	140
DIV、D.DIV (除法)	141
ACCU、D.ACCU (累加)	141
AVG、D.AVG (平均值)	142
ABS、D.ABS (绝对值)	143
NEG、D.NEG (求 2 个补码)	143

4.13 浮点数指令144

FCMP (浮点数比较)145

FZCP (浮点数区域比较)146

FMOV (浮点数移动)147

FADD (浮点数加法)147

FSUB (浮点数减法)148

FMUL (浮点数乘法)149

FDIV (浮点数除法)149

FACCU (浮点数累加)150

FAVG (浮点数平均值)151

FMAX (浮点数最大值)151

FMIN (浮点数最小值)152

FTOI (浮点数转换为整数)153

ITOF、D.ITOF (整数转换为浮点数)153

FABS (浮点数绝对值)154

FSQR (浮点数平方根)155

FSIN (正弦值)155

FCOS (余弦值)156

FTAN (正切值)157

FASIN (反正弦)157

FACOS (反余弦)158

FATAN (反正切)158

FLN (自然对数)159

FLOG (以 10 为底的对数)160

FEXP (自然指数)160

FRAD (角度转换为弧度)161

FDEG (弧度转换为角度)162

FXY (指数)162

4.14 时钟指令163

TCMP (实时时钟比较)163

TACCU (时间累计)164

SCLK (设置时钟)165

TIME (时间开关)166

DATE (日期开关)166

INVT (倒计时)167

4.15 通讯指令168

SUM、SUM.LB (SUM 累加和校验)169

BCC、BCC.LB (BCC 校验)170

CRC、CRC.LB (CRC 校验)170

LRC、LRC.LB (LRC 校验)171

COMM、COMM.LB (串行通讯)172

MODR (Modbus 读)174

MODW (Modbus 写)176

HWRD (Haiwellbus 读)177

HWWR (Haiwellbus 写)178

RCV (接收通讯数据)179

XMT、XMT.LB (发送通讯数据)181

FROM (读取扩展模块 CR 数据)182

TO (扩展模块 CR 数据写入)183

4.16 中断指令183

ATCH (中断绑定)	184
DTCH (中断释放)	185
ENI (允许中断)	186
DISI (禁止中断)	186
4.17 程序控制指令	186
MC (主控指令)	187
MCR (主控指令清除)	188
FOR (循环指令)	188
NEXT (循环结束)	189
WAIT (延时等待)	190
CALL (调用子程序)	190
EXIT (条件退出)	193
REWD (扫描时间复位)	193
JMPC (条件跳转)	193
LBL (跳转标号)	194
4.18 特殊功能指令	195
GPWM (通用脉宽调制)	195
FTC (模糊温度控制)	196
PID (PID 控制)	197
HAL、D.HAL (上限报警)	199
LAL、D.LAL (下限报警)	200
LIM、D.LIM (范围限制)	201
SC、D.SC (线性变换)	202
VC (阀门控制)	203
TTC (温度曲线控制)	204
第五章、编程操作手册.....	206
5.1 编程环境	206
概述	206
菜单	207
工具栏	209
右键菜单	209
快捷键	212
状态栏	214
工作区	214
工程管理器	214
PLC 资源	215
元件注释表	215
指令说明窗口.....	216
指令属性窗口.....	216
端子定义窗口.....	217
5.2 程序结构	217
主程序	217
子程序	217
中断程序	218
5.3 程序项目建立	218
新建程序项目.....	218
新建程序块	219
新建子程序	219
打开程序块	220
删除程序块	221

程序块执行顺序调整.....	221
5.4 指令使用表格	222
Haiwellbus 读通讯表	222
Haiwellbus 写通讯表	222
离散位元件表.....	223
离散寄存器元件表.....	223
初始寄存器值表.....	224
新建表格	224
打开表格	225
删除表格	226
5.5 PLC硬件配置	226
PLC 主机型号	227
扩展模块编辑.....	227
外部 I/O 元件的分配	228
X 开关量输入参数.....	228
Y 开关量输出参数.....	228
AI 模拟量输入参数	228
AQ 模拟量输出参数.....	229
HSC 高速计数器参数.....	229
PLS 高速脉冲输出参数.....	229
5.6 LD梯形图程序的编写.....	230
LD 工作区	230
开关编辑	230
开关状态改变.....	231
LD 指令编辑	231
指令输入端状态改变.....	232
分支编辑	232
网络编辑	234
5.7 FBD功能块程序编写.....	235
FBD 工作区.....	235
FBD 指令编辑.....	235
FBD 指令间连接.....	236
改变 FB 执行顺序.....	236
指令输入端状态改变.....	236
页面编辑	237
FBD 注释.....	237
5.8 IL指令表程序编写.....	237
IL 工作区.....	237
IL 指令编辑.....	238
指令输入状态改变.....	238
IL 注释.....	239
5.9 其他	239
查找、替换	239
编译程序	239
程序及表格的导入.....	240
程序及表格的导出.....	240
打印及预览	241
元件使用表	242
元件注释	242
第六章、仿真与在线调试.....	244

6.1 概述	244
6.2 仿真环境	244
仿真工具栏	244
右键菜单	244
硬件仿真窗口	245
AI/AQ 模拟窗口	246
数据锁定窗口	246
实时曲线窗口	246
消息窗口	246
6.3 仿真操作	246
脱机仿真的一般步骤	246
启动仿真器	247
元件监控	247
元件状态表	248
强制	249
锁定数据	250
实时曲线	251
仿真停电	251
6.4 通讯仿真器	251
启动通讯仿真器	251
手动输入从机应答信息	252
使用实际串口与从机通讯	252
6.5 高速计数仿真	252
6.6 脉冲输出仿真	254
6.7 插补仿真器	255
6.8 在线调试与脱机仿真的区别	256
第七章、在线控制 PLC	257
7.1 PLC站号设置	257
7.2 与 PLC联机	257
7.3 在线 PLC窗口	258
7.4 下载程序	259
7.5 上载程序	259
7.6 生成 PLC执行文件	260
7.7 下载 PLC执行文件	260
7.8 PLC固件升级	261
7.9 启动、停止 PLC	263
7.10 清除 PLC程序	263
7.11 程序比较	263
7.12 设置 PLC口令	264
7.13 设置 PLC时钟	264
7.14 设置 PLC通讯参数	265
7.15 设置 PLC参数	265
第八章、联网通讯功能	266
8.1 特点	266
8.2 联网示意图	267
8.3 Modbus通讯	268
8.4 Haiwell bus通讯	269
8.5 自由通讯	270
第九章、远程模块	274
9.1 概述	274

9.2	模块站号设置	275
9.3	与模块联机	276
9.4	参数修改	277
9.5	参数上载	278
9.6	参数下载	278
9.7	固件升级	278
9.8	在线监控	279
第十章、附录		280
10.1	SM系统状态位	280
10.2	SV系统寄存器	282
10.3	系统中断表	287
10.4	通讯地址代码表.....	288
10.5	错误代码表	289
10.6	编程电缆接线图.....	291

第一章、产品简介

Haiwell（海为）PLC 是一款通用高性能可编程控制器，产品广泛应用于塑料、包装、纺织、食品、医疗、制药、环保、市政、印刷、建材、电梯、中央空调、数控机床等领域的系统和控制设备。除自身带有各种外设接口（开关量输入、开关量输出、模拟量输入、模拟量输出、高速计数器、高速脉冲输出通道、电源、通信端口等）外，还可扩展各种类型的扩展模块，进行灵活的配置。

HaiwellHappy 编程软件是一款符合 IEC 61131-3 规范的 PLC 编程软件，用于 Haiwell（海为）PLC 的编程，它支持 LD（梯形图）、FBD（功能块图）和 IL（指令表）三种编程语言。它可运行于 Win98、Win200X、WinXP、Win7、Win8 及以后版本的 Windows 操作系统。

1.1、Haiwell（海为）PLC 的特点

- 1、固件升级功能：**率先在小型可编程控制器中实现固件升级功能，无论是 CPU 主机或扩展模块，都可以通过固件升级功能对固件进行免费升级，使先前购买的产品也能拥有海为公司不断推出的各种最新功能。
- 2、丰富的网络通讯功能：**CPU 主机带 2 个通讯口，可扩展至 5 个通讯口，每个通讯口都可以进行编程和联网，都可作为主站或从站。支持 1:N、N:1、N:N 联网方式，支持各种人机界面和组态软件，可与任何带通讯功能的第三方设备（如变频器、仪表、条码阅读器等）联网。
- 3、支持多种通讯协议：**内置 Modbus RTU/ASCII 协议、自由通讯协议以及海为公司的 Haiwellbus 高速通讯协议。极为便利的通讯指令系统，无论使用何种通讯协议都只需一条通讯指令便可完成复杂的通讯功能，无须再为通讯端口冲突、发送接收控制、通讯中断处理等问题烦恼，并且可以在程序中混合使用各种协议轻松完成所需的各种数据交换。
- 4、高速脉冲计数功能：**单机支持 8 路 200KHz 双相高速脉冲计数，支持 7 种计数模式（脉冲/方向 1 倍频、脉冲/方向 2 倍频、正/反转脉冲 1 倍频、正/反转脉冲 2 倍频、A/B 相脉冲 1 倍频、A/B 相脉冲 2 倍频、A/B 相脉冲 4 倍频）和 3 种比较方式（单段比较、绝对方式比较、相对方式比较），支持 8 段比较设定值，带自学习功能。
- 5、高速脉冲频率测量：**单机支持 16 路 200KHz 高速脉冲频率测量，支持以时间或脉冲数方式测量频率。
- 6、高速脉冲输出功能：**单机支持 8 路 200KHz 双相高速脉冲输出，支持加减速脉冲输出、多段包络脉冲输出功能，独有的同步脉冲输出功能可轻松实现精确的同步控制。单机可支持 16 路脉宽调制输出（PWM），可同时驱动 16 台伺服或者步进电机。
- 7、运动控制功能：**单机支持 8 轴 200KHz 运动控制，支持任意 2 轴的直线插补、圆弧插补，支持随动脉冲输出、绝对地址、相对地址、反向间隙补偿、原点回归、电气原点定义等功能。
- 8、边沿捕捉及中断：**CPU 主机支持 8 路的上下沿捕捉及中断功能，所有开关量输入支持信号滤波设定，所有开关量输出支持停电输出保持设定。提供多达 52 个实时中断。
- 9、强大的模拟量处理功能：**可用 AI 寄存器直接访问模拟量输入，模拟量输入支持工程量转换、采样次数设定及零点修正。可用 AQ 寄存器直接控制模拟量输出，模拟量输出支持工程量转换并且可配置停电输出保持功能。
- 10、强大的密码保护功能：**三级密码保护功能（程序文件口令、各程序块口令、PLC 硬件口令）以及禁止程序上载等保护。
- 11、自诊断功能、掉电保护功能、万年历(RTC)、浮点数运算等**

1.2 编程软件特点

- 1、内置 PLC 仿真器：**国内第一个带内置仿真器的 PLC 编程软件，全面实现了 PLC 程序的仿真运行。在编程过程中间或程序编写完成后，可用仿真器在完全脱离 PLC 的情况下仿真运行 PLC 程序，以检查程序执行是否正确，极大地减少现场调试时间，降低调试难度，提高调试效率。

- 2、**通讯仿真器**：是专用于调试通讯指令的仿真工具。它可以手动模拟输入从机返回给 PLC 的应答信息，也可以使用计算机的实际串口与从机真实通讯，真实仿真 PLC 执行通讯指令的过程以及对从机返回数据的处理。
- 3、**插补仿真器**：追踪并绘制出直线插补、圆弧插补等运动控制指令生成的运动轨迹，列出运动平面各轴对应的脉冲输出通道的参数，显示该通道的当前位置、机械原点位置、输出模式等，可以设定轴长、单位脉冲数。
- 4、**生成 PLC 执行文件**：可将 PLC 源程序生成为可以独立发布、独立执行的 PLC 执行文件，这样就可以非常方便、非常安全地把 PLC 执行文件交给用户自己下载，又不用担心用户看到源程序内容。
- 5、**创新的便利指令集**：在分析吸收现有各种 PLC 指令的基础上，Haiwell（海为）PLC 推出许多功能强大的创新便利指令。如通讯指令（COMM、MODR、MODW、HWRD、HWWR）、数据组合分散指令（BUNB、BUNW、WUNW、BDIB、WDIB、WDIW）、PID 控制（PID）、阀门控制（VC）、上下限报警（HAL、LAL）、范围变送（SC）、温度曲线（TTC）等，只需一条指令就能实现其他 PLC 需用多条指令来实现的功能，这些指令十分易于理解和使用，极大地提高了编程效率和程序运行速度。
- 6、**模块化程序项目结构**：可建立共 31 个程序块（主程序、子程序、中断程序），可任意选择语言进行编程，程序块的执行顺序可任意调整，每个块可单独导入导出并且具有与程序项目相同的口令保护。充分实现模块化编程和程序重用的梦想。
- 7、**指令使用表格**：提供多种指令使用表格，使用这些表格可减少许多程序量，节约程序空间，如初始化数据等。每个表格可单独导入导出并且具有与程序项目相同的口令保护。
- 8、**强大的在线联机功能**：可搜索出与 PC 上位机连接的所有 PLC，显示出所有在线 PLC 的运行状态、故障状态、RUN/STOP 开关位置、硬件配置信息、通讯端口参数等详尽信息，可选择对任意一台 PLC 进行在线监控、程序下载、硬件升级、控制 PLC 运行停止、调整 PLC 实时时钟、设置修改保护口令、修改通讯端口参数、修改看门狗时间和 PLC 站名称等。
- 9、**在线监控调试功能**：提供多达 10 页的元件监控表，可选择以十进制、十六进制、二进制、浮点数、字符方式显示数据，支持位元件与寄存器元件混合监控并且同时显示元件注释。各种指令使用表格可导入到监控表中。
- 10、**独有的实时曲线功能**：可对任意寄存器元件进行实时曲线监控，方便过程控制调试。
- 11、**人性化的输入方式**：提供快捷键、拖放、点选等多种指令输入方式，对每个输入输出端子都提示其有效的元件或数值范围，可直接输入，对一些组合数据（如通讯协议等）还可通过双击该指令以配置方式输入数据。
- 12、**便利的注释功能**：提供了元件注释、网络注释、指令注释、程序块注释、表格注释和项目注释功能，元件注释可通过在元件后跟“//”直接输入（如：X0//电机启动），注释可选择下载到 PLC 中，方便日后下载程序的阅读或修改。
- 13、**详尽的提示信息和在线帮助**：提供 PLC 资源窗口、指令说明窗口等信息窗口，所有指令、硬件模块的详细说明均可在编程界面中通过 F1 键打开功能强大的在线帮助系统找到答案，即使首次使用 HaiwellHappy 编程软件也可轻松完成一个控制程序的编写工作。
- 14、**方便的编辑功能**：支持所有常规的编辑操作以及查找替换、指令上下移、网络上下移、程序项目之间的拷贝粘贴等。
- 15、**硬件配置、子程序参数传递、局部元件、间接寻址、打印、预览、查错、CRC 计算、口令保护等**

第二章、快速入门

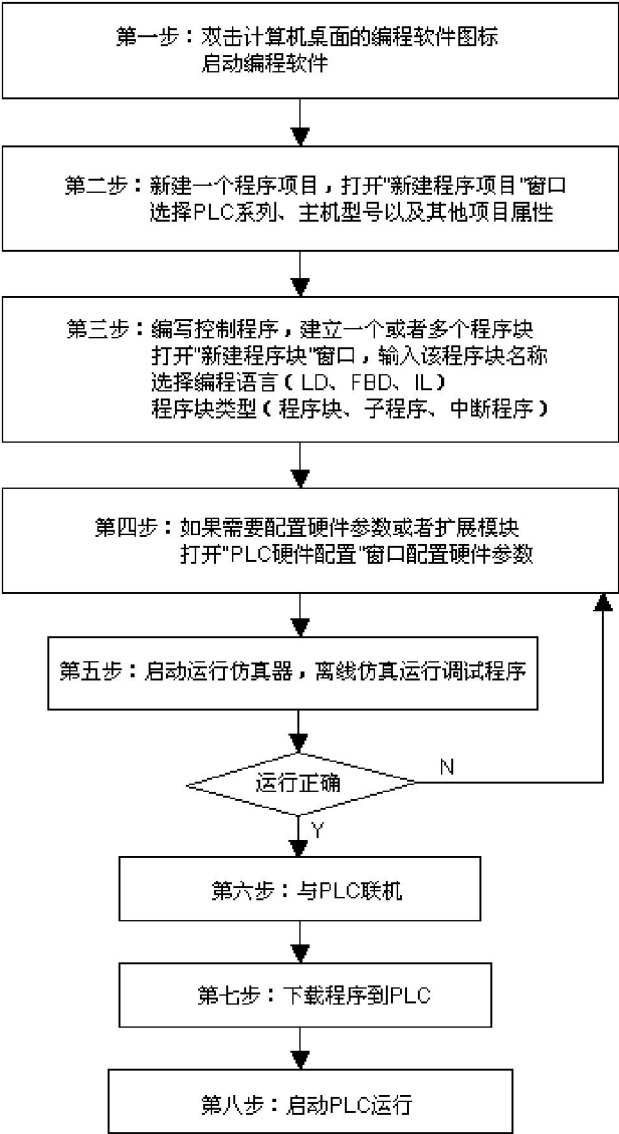
本章简要介绍编写 PLC 控制程序的一般步骤，以帮助初学者尽快熟悉、掌握 PLC 编程软件的编程操作。

编程的一般流程

第一步：双击计算机桌面的编程软件图标，启动编程软件。

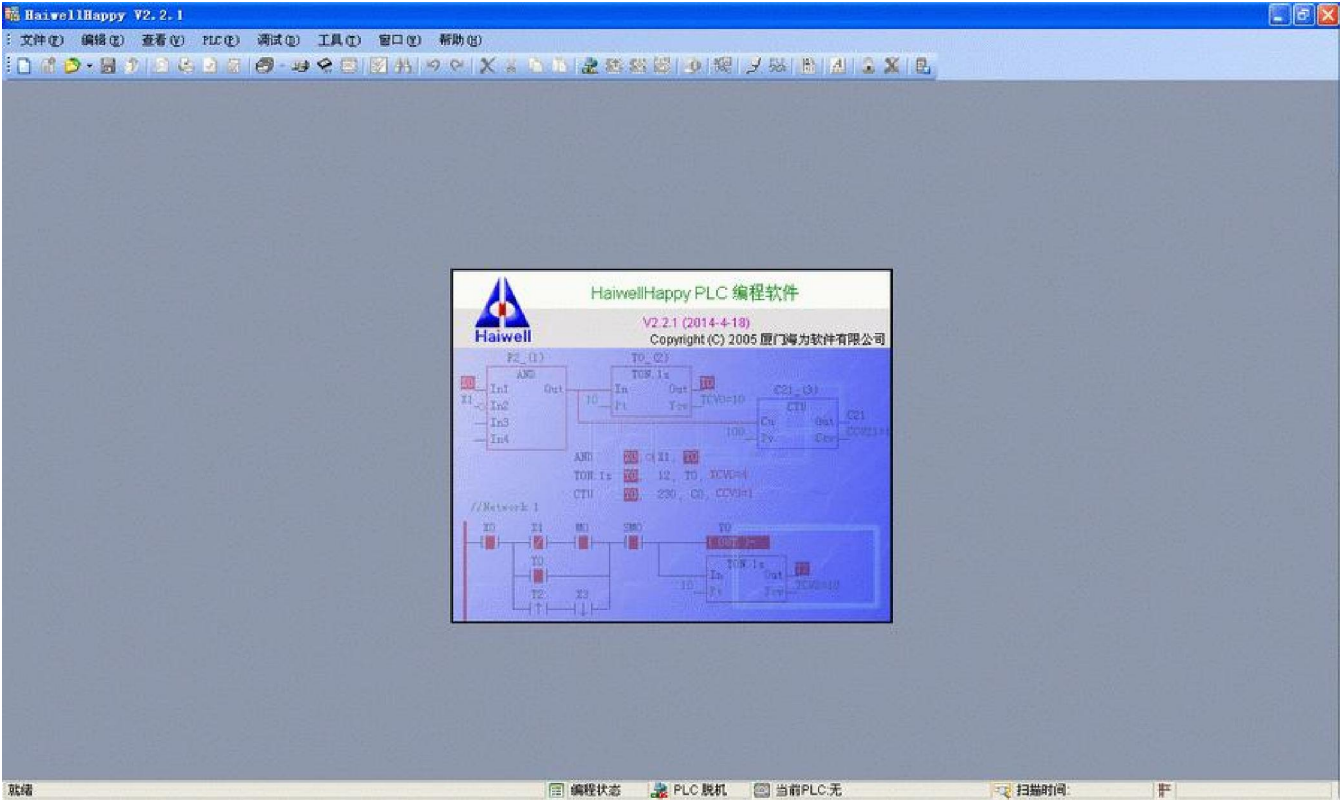
第二步：新建一个程序项目，用菜单【文件/新建程序项目】，打开“新建程序项目”窗口，选择 PLC 系列、PLC 主机型号以及其他项目

- 属性。
- 第三步：编写控制程序，建立一个或者多个程序块，用菜单【文件/新建...../主程序块】，打开“新建程序块”窗口，输入该程序块名称、选择编程语言（LD、FBD、IL）及程序块类型（程序块、子程序、中断程序）。
- 第四步：如果需要配置硬件参数或者扩展模块，用菜单【查看/PLC 硬件配置】打开“PLC 硬件配置”窗口。
- 第五步：启动仿真器，离线仿真运行调试程序，运行结果正确则到下一步，否则返回第 三步修改编写程序。
- 第六步：与 PLC 联机。
- 第七步：下载程序到 PLC。
- 第八步：启动 PLC 运行。



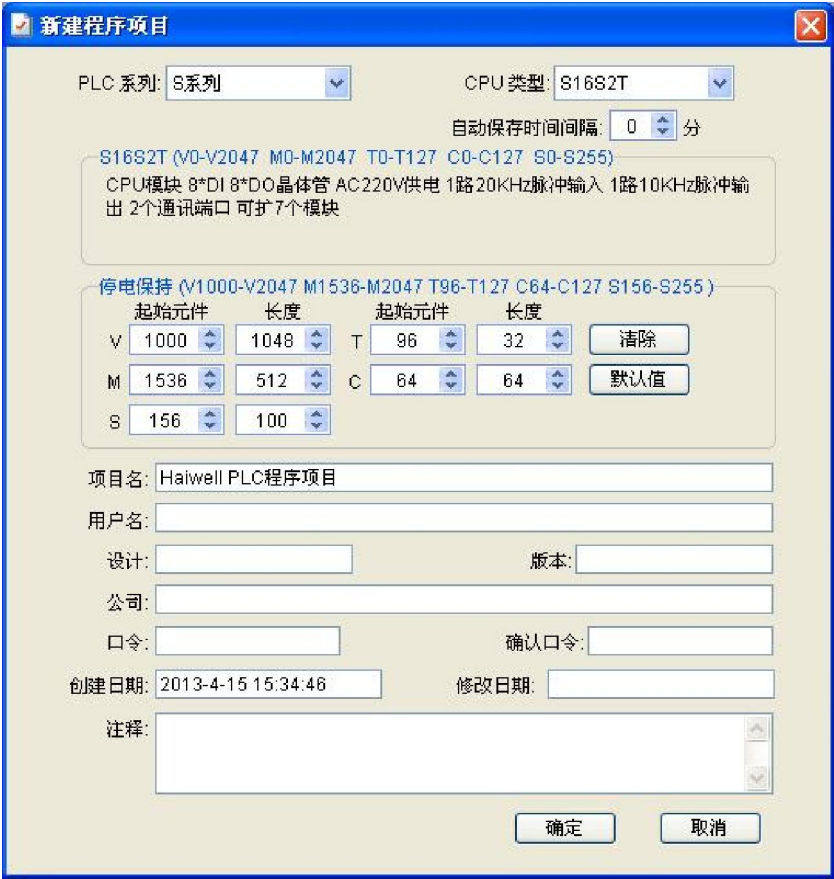
第一步：启动编程软件

双击计算机桌面的编程软件图标，即可启动编程软件，如下图：



第二步：新建程序项目

通过点击菜单【文件/新建程序项目】、点击工具栏的按钮或用快捷键 Ctrl+N，打开“新建程序项目”窗口。



- 1、在“PLC 系列”下拉列表中选择“PLC 系列”。
- 2、在“CPU 类型”下拉列表中选择“CPU 即主机的型号”。
- 3、停电保持区可以由用户任意定义，可设置 V、M、S、T、C 五种元件类型 的停电保持区的起始元件及长度。系统默认的停电保持区

域列表：

元件类型	停电保持区	元件个数
V	V1000~V2047	1048
M	M1536~M2047	512
S	S156~S255	100
T	T96~T127	32
C	C64~C127	64

- 4、在“项目名”中输入新建项目的名称，此项目名称将显示于主画面的工程管理器中。
- 5、在“用户名”、“设计”、“版本”、“公司”等栏目中输入该项目的 有关信息。
- 6、如需对新建的程序项目做口令保护，则在“口令”及“确认口令”中输入保护口令。
- 7、在“注释”中可以输入该项目的相关备注信息。

第三步：编写控制程序

控制程序是实现自动控制的核心，其本质是根据控制对象（如机台、自动化设备、生产流水线等）的具体要求运用编程软件提供的各种指令实现对控制对象的自动控制。

- 1、根据需要建立一个或多个程序块，选择自己熟悉的编程语言（LD、FBD、IL），每个块可以单独设置保护口令，实现局部加密功能。
- 2、在程序块中运用编程软件提供的各种指令编写程序，实现各种控制逻辑和算法。
- 3、在程序编写完成后，保存程序文件，启动仿真器运行，调试程序是否达到控制要求。
- 4、详细操作参见“[编程操作手册](#)”

【示例】

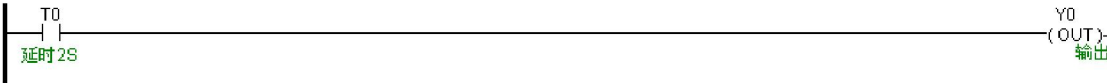
下面通过一个例子来讲解如何编写控制程序，示例控制要求：按 X0 按钮启动，延时 2 秒后输出 Y0。按 X1 按钮停止。

【示例程序】

//Network 1 启动、停止、自锁、延时控制

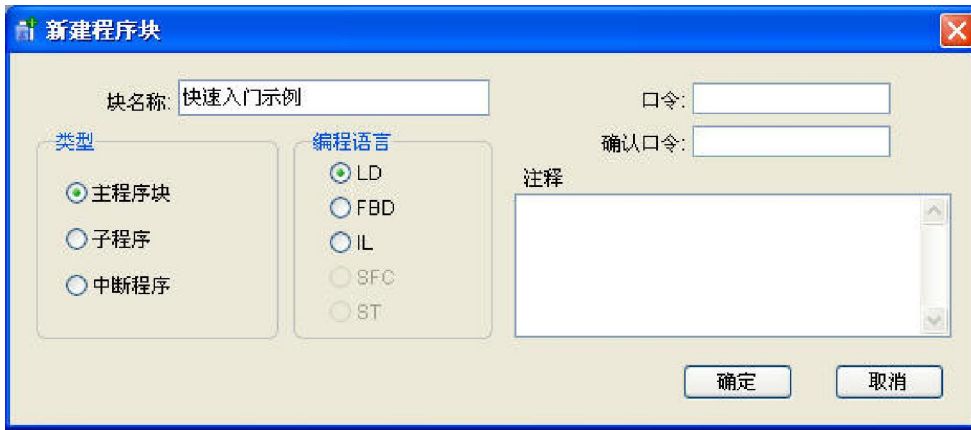


//Network 2 延时时间到，控制输出



【示例编程操作】

- 1、新建一个“快速入门示例”的主程序块，打开方式：A、点击菜单 【文件/新建...../主程序块】，B、点击工具栏的  按钮下的“主程序块”。“新建程序块”窗口如下图。



- 2、点击工具栏  按钮或者按 F9 快捷键添加 一个开关，输入开关元件“X0//启动”（“//”后面的“启动”为元件 X0 的注释，下同），按 Ctrl+3 快捷键将常开开关改变为上升沿。
 - 3、按 F10 快捷键并联一个开关，输入 开关元件“M0//自锁”
 - 4、移动光标到 X0 开关的右边，按 F9 快捷键串联一个开关，输入 开关元件“X1//停止”，按 Ctrl+2 快捷键将常开开关改变为常闭。
 - 5、将光标右移，输入“TON”回车，添加一条“TON”指令，连续按回车键，当输入框在“Pt”端子时输入计时器设定时间 2。
 - 6、按 F11 快捷键添加并联输出指令“OUT”，输入 元件“M0”。
 - 7、按工具栏  按钮或者 按 Ctrl+L 快捷键添加一个新的网络。
 - 8、按 F9 快捷键添加 一个开关，输入开关元件“T0//延时 2S”。
 - 9、按 F11 快捷键添加输出指令“OUT”，输入 元件“Y0//输出”。
- 至此程序输入完成。按 Ctrl+S 快捷键保存程序文件，可以启动仿真器运行，调试程序是否达到控制要求。

第四步：PLC 硬件配置

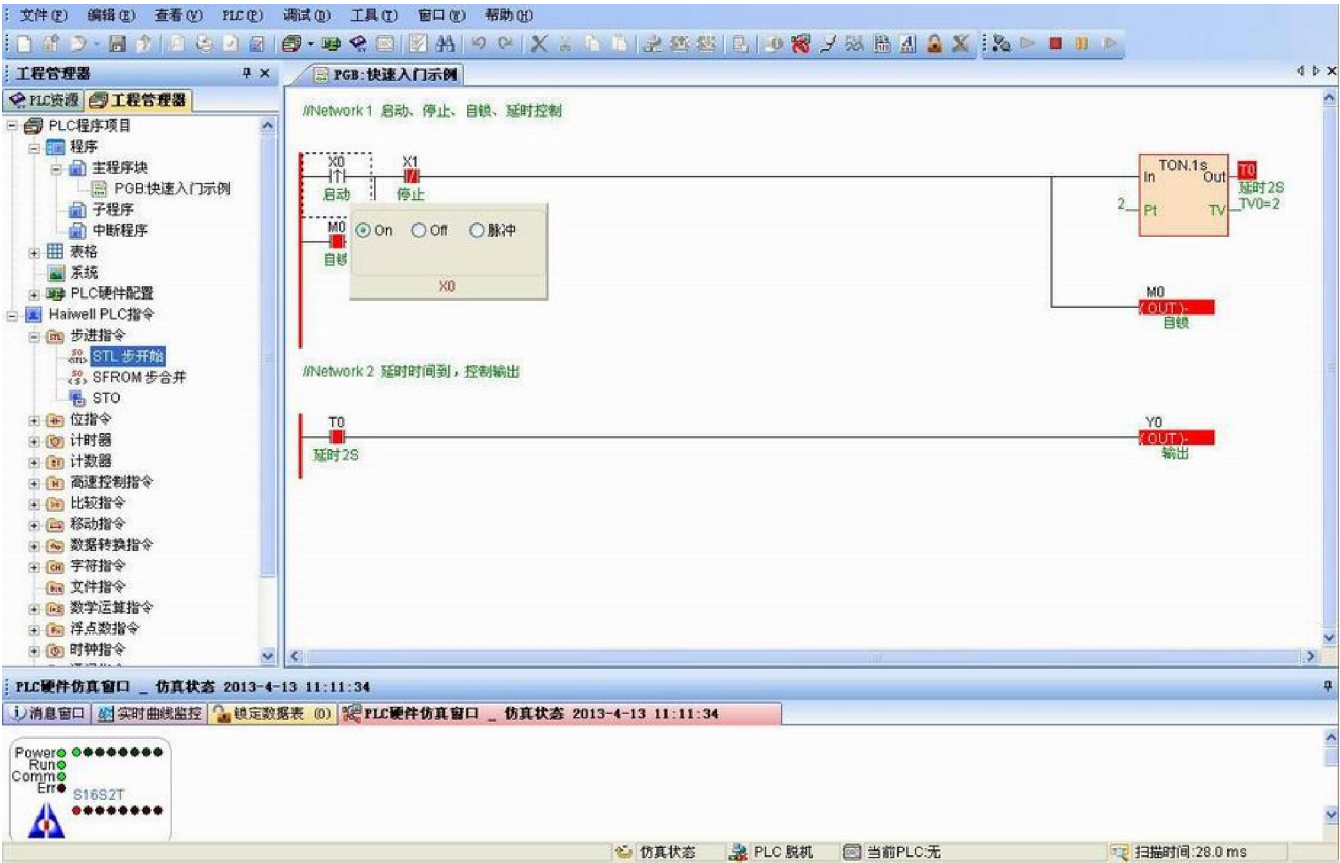
如果需要配置硬件参数（如配置 AI 输入通道的信号类型、工程量等）或者要添加扩展模块等与硬件相关的信息，那么就做本步，否则跳过本步。

- 1、双击“工程管理器”窗口中目录树的“PLC 硬件配置”节点或者用菜单【查看/PLC 硬件配置】即可打开“PLC 硬件配置”窗口。
- 2、点开“工程管理器”窗口中目录树的“PLC 模块”结点，选择要添加的模块，用鼠标拖拉该模块到右边的硬件配置列表中。
- 3、点击硬件配置列表中的模块，在列表的下方将显示该模块可进行的属性定义及配置。
- 4、进行对各种模块进行属性定义及配置操作。
- 5、详细操作参见程序编写部分的“[PLC 硬件配置](#)”。

第五步：脱机仿真调试

编程过程中间或程序编写完成后，可用仿真器在完全脱离 PLC 的情况下仿真运行 PLC 程序，以检查程序执行是否正确，极大地减少现场调试时间，降低调试难度，提高调试效率。

- 1、点击菜单【调试/启动仿真器】或点击工具栏的  按钮启动仿真器，系统将自动对当前程序项目进行编译。仿真器的详细操作参见“[仿真与在线调试](#)”。
- 2、若程序 编译有错误，仿真器不能运行，用户需要根据编译提示的错误信息对程序修改。
- 3、程序 编译没有错误或只有警告时，则启动仿真器。仿真器界面如下图所示：

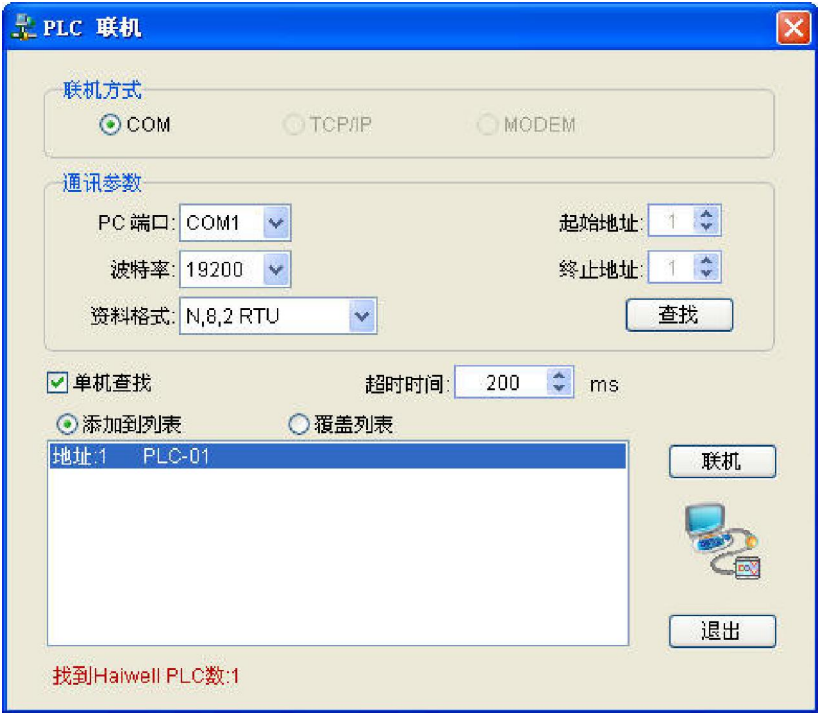


- 4、双击程序中“X0”开关强制 X0=ON，则 M0=ON 自锁，计时器 T0 开始计时，当 TV0=2 计时时间到则 T0=ON，Y0=ON。
- 5、双击程序中“X1”开关强制 X1=ON，X1 的常闭开关没有电，则 M0=OFF，T0=OFF，Y0=OFF。
- 6、经过仿真器运行验证，程序运行结果正确。

第六步：与 PLC 联机

与网络上的一个或多个 PLC 建立连接。只有与 PLC 联机后才可以对 PLC 进行控制操作，如：上载或下载程序等。

- 1、点击菜单【PLC/PLC 联机】或点击工具栏上按钮，打开“PLC 联机”窗口。



- 2、点击“联机”按钮（一般情况下，使用默认参数即可），已经联机的 PLC 会自动添加到列表框中，此时点击“退出”关闭窗口。

3、PLC 联机及参数设置参见“在线控制 PLC”章节的“[与 PLC 联机](#)”。

第七步：下载程序

将当前的程序项目（硬件配置及程序等内容）下载到联机的 PLC 中。下载前系统会自动对当前程序项目进行编译，若程序编译有错误，则列出所有错误，用户将程序修改至无任何错误方可下载。

点击菜单【PLC/PLC 程序下载】或点击工具栏上的按钮，打开“PLC 程序下载”窗口，点击“下载”按钮将程序下载到 PLC 中。



注：详细操作参见“在线控制 PLC”章节的“[下载程序](#)”。

第八步：启动 PLC

在下载完成后，若 PLC 已经处于运行状态（RUN 指示灯亮），可跳过此步。否则将 PLC 运行开关拨到“RUN”位置即可。

第三章、PLC元件与数据

3.1 概述

Haiwell（海为）PLC 在系统中配置了多种虚拟的元件（亦称：变量、地址），如：X、Y、M、AI、AQ、V 等。元件可分为位元件与寄存器元件 2 大类别，位元件只占 1 个位表示布尔变量，寄存器元件为 1 个字（16 位，2 个字节）表示数值变量，它们可以在程序中反复使用。PLC 内部采用软元件进行程序运算以实现所需的运算和控制功能。

3.2 数据

一、数据类型

数据类型	类型说明	数据长度	范围
BOOL	位	1 个位元件	1（ON）、0（OFF）
INT	有符号整数	16 位，1 个寄存器元件	- 32768~32767
DINT	有符号长整数	32 位，2 个寄存器元件	- 2147483648~2147483647
REAL	浮点数	32 位，2 个寄存器元件	- 3.402823e+38~3.402823e+38
CHAR	字符串	1 个字符占 1 个字节	

二、元件与数据类型的匹配关系

数据类型	元件类型															
BOOL		X	Y	T	C	M	SM	LM	S							
INT	常数									AI	AQ	TV	CV	V	LV	SV
DINT	常数											TV	CV	V	LV	SV

REAL	常数														V	LV		
CHAR	常数														V	LV		

三、常数

常数类型	示例	有效范围	说明
16 位有符号整数	1234、-7890	-32768~32767	
32 位有符号整数	12345678、-9876543	-2147483648~2147483647	
十六进制 16 位常数	0x2EF8、0x9A12	0x0~0xFFFF	
十六进制 32 位常数	0xA76DCFE9	0x0~0xFFFFFFFF	
单精度浮点常数	3.1415926、-0.02341	-3.402823e+38~3.402823e+38	采用 IEEE-754 标准

四、32 位数据的存储和使用

- 1、32 位数据在寄存器元件中的存储方式：DINT、REAL 型的数据都是 32 位长度的，而一个寄存器只有 16 位长度，因此需要 2 个地址连续的寄存器存储 32 位数据。存储 32 位数据时，按照低字在前、高字在后的顺序存储，例如：32 位长整数数据 0xA76DCFE9 存放在 V0V1 寄存器中，则 V0 中存放 0xCFE9，V1 中存放 0xA76D。
- 2、寄存器元件可以存储整数、长整数、浮点数，如果指令是以寄存器元件作为操作数，那么该元件是 16 位整数、32 位长整数或者是浮点数，将由该指令的操作数的数据类型决定。例如：在指令“MOV -23, V10”中，由于 MOV 是 16 位整数移动指令，因此元件 V10 为 16 位整数；在指令“D.MOV 7891223, V10”中，由于 D.MOV 是 32 位整数移动指令，因此元件 V10 为 32 位长整数（自动使用 V10V11）；在指令“FMOV -9.223, V10”中，由于 FMOV 是浮点数移动指令，因此元件 V10 为浮点数（自动使用 V10V11）。
- 3、浮点数指令使用浮点数作为操作数，其他指令使用整数作为操作数（16 位指令使用整数，32 位指令使用长整数），数据类型必须使用转换指令显式转换。

3.3 元件一览表

一、Haiwell（海为）PLC 位元件表

元件	名称	范围	读写属性	说 明
X	开关量输入	X0~X1023	只读	
Y	开关量输出	Y0~Y1023	可读/写	
M	内部继电器	M0~M12287	可读/写	系统默认 M1536~M2047 停电保持，512 点
T	计时器线圈	T0~T1023	可读/写	T0~T251、T256~T1023 的时基可 10ms、100ms、1s 任意设置 系统默认 T96~T127 停电保持，32 点 T252~T255 为 1ms 时基
C	计数器线圈	C0~C255	可读/写	系统默认 C64~C127 为停电保持，64 点
SM	系统状态位	SM0~SM215	全部可读部分可写	
S	步进继电器	S0~S2047	可读/写	系统默认 S156~S255 停电保持，100 点
LM	局部继电器	LM0~LM31	可读/写	局部(程序块、子程序、中断程序)区域内有效的内部继电器

二、Haiwell（海为）PLC 寄存器元件表

元件	名称	范围	读写属性	说 明
AI	模拟量输入寄存器	AI0~AI255	只读	
AQ	模拟量输出寄存器	AQ0~AQ255	可读/写	
V	内部寄存器	V0~V14847	可读/写	系统默认 V1000~V2047 停电保持，1048 点
TV	计时器当前值	TV0~TV1023	可读/写	T0~T251、T256~T1023 的时基可 10ms、100ms、1s 任意设置 系统默认 TV96~TV127 为停电保持，32 点 T252~T255 为 1ms 时基

元件	名称	范围	读写属性	说 明
CV	计数器当前值	CV0~CV255	可读/写	CV 为 16 位寄存器，其中 CV48~CV79 为 32 位寄存器，系统默认 CV64~CV127 为停电保持，64 点
SV	系统寄存器	SV0~SV154	全部可读部分可写	
LV	局部寄存器	LV0~LV31	可读/写	局部(程序块、子程序、中断程序)区域内有效的内部寄存器
P	变址指针	P0~P9	可读/写	特殊寄存器，用于变址寻址方式

注：表中 T（TV）、C（CV）、M、S、V 五种元件类型列出的停电保持区域是系统默认的，此五种元件类型的停电保持区域可由用户自行设定。

3.4 开关量输入 [X]

开关量输入 X：与外部开关量输入装置连接（如：开关、按钮等）， 获取外部开关量输入信号的状态（ON 或 OFF）进入 PLC。

软元件符号用 X 表示，如：X0、X1、... X8、X10、X11、...。编号固定从 X0 开始，在"PLC 硬件配置"窗口进行配置。系统会自动对其分配地址。

3.5 开关量输出 [Y]

开关量输出 Y：与外部负载连接，用于驱动连接输出接点 Y 的负载通断（ON 或 OFF）。对于同一个接点 Y 其输出线圈在程序中也多处输出，但建议仅使用一次，以提高程序的可靠性及可读性。

软元件符号用 Y 表示，如：Y0、Y1、... Y8、Y10、Y11、...。编号固定从 Y0 开始，在"PLC 硬件配置"窗口进行配置。系统会自动对其分配地址。

3.6 计时器 [T]

计时器 T：每个计时器由输出线圈 T 及当前值 TV 组成。

计时时间 = 计时单位（时基） * 设定值。其中 T0~T251、T256~T1023 的时基可 10ms、100ms、1s 任意设置，T252~T255 为 1ms 时基。

在一个程序项目中，一个计时器只能使用一次，但其输出线圈 T 及当前值 TV 在程序中的使用次数无限制。

计时器可分为停电保持型和非停电保持型两种：非停电保持型计时器在 PLC 停机时，其输出线圈 T 及当前值 TV 将被复位。停电保持型计时器在 PLC 停机时，其输出线圈 T 及当前值 TV 将全部被保持 。

3.7 计数器 [C]

计数器 C：每个计数器由输出线圈 C 及当前计数值 CV 组成。

在一个程序项目中，一个计数器只能使用一次，但其输出线圈 C 及当前计数值 CV 在程序中的使用次数无限制。

计数器分 16 位计数器和 32 位计数器，其中 CV48~CV79 为 32 位，其他都是 16 位。

计数器按计数方式的不同分为三种：加计数 CTU、减计数 CTD、加减计数 CTUD。

计数器可分为停电保持型和非停电保持型两种：非停电保持型计数器在 PLC 停机时，其输出线圈 C 及当前值 CV 将被复位。停电保持型计数器在 PLC 停机时，其输出线圈 C 及当前值 CV 将全部被保持。

3.8 内部继电器 [M]

内部继电器 M：又称为辅助继电器，用于内部逻辑运算，可利用辅助继电器 M 来组合控制回路，但无法直接驱动外部负载。

M 可分为停电保持型和非停电保持型两种：非停电保持型 M 在 PLC 停机时，其状态将被复位。停电保持型 M 在 PLC 停机时，其状态被保持 。

3.9 步进继电器 [S]

步进继电器 S：用于步进指令，每个步进继电器代表一个步，如果程序没有使用步进指令可以作为普通内部继电器使用 。

S 可分为停电保持型和非停电保持型两种：非停电保持型 S 在 PLC 停机时，其状态将被复位。停电保持型 S 在 PLC 停机时，其状态被保持。

3.10 系统状态位 [SM]

SM 系统状态位是系统定义的一组特殊的内部继电器，在程序中使用的次数无限制，每一个系统状态位均有其特定的作用。

参见“[SM 系统状态位](#)”

3.11 局部继电器 [LM]

局部继电器 LM：是一种作用范围仅限于一个程序区域（程序块、子程序、中断程序）内的特殊内部继电器，它与内部继电器 M 不同。

M 作用范围：当前程序项目的所有程序，所有的扫描周期

LM 作用范围：所在的程序区域（程序块、子程序、中断程序），状态（ON/OFF）只保留一个扫描周期，到下一扫描周期开始时，状态复位为 OFF，一般作为临时变量或作为被调用程序块的参数使用。

3.12 模拟量输入 [AI]

模拟量输入寄存器 AI：每个模拟量输入通道对应一个模拟量输入寄存器 AI，模拟量输入通道与外部设备相连，用于测量外部信号的连续变化如温度、压力、流量等。

当外部模拟量输入通道的信号变化时，会实时反映在对应的模拟量输入寄存器 AI 中。

每个模拟量输入通道的信号类型、使用工程量、量程上下限等可在“[PLC 硬件配置](#)”窗口进行配置。系统会自动对其分配地址。

3.13 模拟量输出 [AQ]

模拟量输出寄存器 AQ：每个模拟量输出通道对应一个模拟量输出寄存器 AQ，模拟量输出通道与外部设备相连。

改变模拟量输出寄存器 AQ 值的大小则对应的模拟量输出通道的信号发生相应变化。

每个模拟量输出通道的信号类型、使用工程量、量程上下限等可在“[PLC 硬件配置](#)”窗口进行配置，系统会自动对其分配地址。

3.14 计时器当前值 [TV]

计时器当前值寄存器 TV：用于存放计时器当前已计时的时间。

参见“[计时器 T](#)”

3.15 计数器当前值 [CV]

计数器当前值寄存器 CV：用于存放计数器的当前计数值。

参见“[计数器 C](#)”

3.16 内部寄存器 [V]

内部寄存器 V：用于储存数据，为 16 位寄存器（b0~b15），整数可表达的数值范围为-32768~+32767，2 个连续的 16 位寄存器为 32 位数据寄存器，长整数可表达的数值范围为-2147483648~+2147483647，浮点数可表达范围为-3.402823e+38~3.402823e+38。

V 可分为停电保持型和非停电保持型两种：非停电保持型 V 在 PLC 停机时，其数值将被复位为 0。停电保持型 V 在 PLC 停机时，其数值被保持。

3.17 系统寄存器 [SV]

SV 系统寄存器是系统定义的一组特殊的内部寄存器，在程序中使用的次数无限制，每一个系统寄存器均有其特定的作用。

参见“[SV 系统寄存器](#)”

3.18 局部寄存器 [LV]

局部寄存器 LV：是一种作用范围仅限于一个程序区域（程序块、子程序、中断程序）内的特殊内部寄存器，它与内部寄存器 V 不同。

V 作用范围：当前程序项目的所有程序，所有的扫描周期。

LV 作用范围：所在的程序区域（程序块、子程序、中断程序），数值只保留一个扫描周期，到下一扫描周期开始时，数值复位为 0，一般作为临时变量或作为被调用程序块的参数使用。

3.19 变址指针 [P]

变址指针 P：是一种用于变址寻址的特殊寄存器。

寄存器变址的输入及在程序中的表现形式：寄存器的基地址+变址指针 P。如：V100P6 表示以寄存器 V100 作为基地址，以变址指针 P6 的内容（值）作为偏移量进行变址寻址，若 P6 的内容为 10，则表明此时要访问的寄存器为 V110。

实际要访问的寄存器 =寄存器的基地址+变址指针 P 偏移量。

注意：

- 1、当用变址指针进行变址寻址时，若寄存器的基地址+变址指针 P 偏移量（实际要访问的寄存器）超出该类寄存器的上限时，应用于该变址寻址的指令将出错，该指令将不执行。
- 2、LM、LV、S、P 元件类型不支持变址
- 3、部分指令不支持变址：RESH、HHSC、HCWR、SPD、PWM、PLSY、PLSR、ZRN、SETZ、PPMR、CIMR、SPLS、MPTO、SYNP、COMM、MODR、MODW、HWRD、HWWR、RCV、XMT、SORT、ENO

第四章、PLC 指令集

Haiwell（海为）PLC 为使用者提供了一套丰富高效的指令系统，在充分吸收现有各种 PLC 指令的基础上，支持多达 200 多条应用指令，其中有许多功能强大的创新便利指令。如通讯指令（[COMM](#)、[MODR](#)、[MODW](#)、[HWRD](#)、[HWWR](#)）、字符转换指令（[ITOC](#)、[CTOI](#)、[FTOC](#)、[CTOF](#)）、数据组合分散指令（[BUNB](#)、[BUNW](#)、[WUNW](#)、[BDIB](#)、[WDIB](#)、[WDIW](#)）、上下限报警（[HAL](#)、[LAL](#)）、阀门控制（[VC](#)）、温度曲线（[TTC](#)）等

指令列表如下：

指令类型	指令名称	8 位模式	32 位模式	指令功能	支持语言		
					LD	FBD	IL
比较开关	=	LB.= HB.=	D.=	等于比较开关，分 16 位/32 位/低字节/高字节	√		
	<>	LB.<> HB.<>	D.<>	不等于比较开关，分 16 位/32 位/低字节/高字节	√		
	>	LB.> HB.>	D.>	大于比较开关，分 16 位/32 位/低字节/高字节	√		
	>=	LB.>= HB.>=	D.>=	大于等于比较开关，分 16 位/32 位/低字节/高字节	√		
	<	LB.< HB.<	D.<	小于比较开关，分 16 位/32 位/低字节/高字节	√		
	<=	LB.<= HB.<=	D.<=	小于等于比较开关，分 16 位/32 位/低字节/高字节	√		
	F.=			浮点数等于比较开关	√		
	F.<>			浮点数不等于比较开关	√		
	F.>			浮点数大于比较开关	√		
	F.>=			浮点数大于等于比较开关	√		
	F.<			浮点数小于比较开关	√		
	F.<=			浮点数小于等于比较开关	√		
步进指令	STL			步开始	√		
	SFROM			步合并	√		
	STO			步转移	√		
位指令	AND			逻辑与		√	√
	OR			逻辑或		√	√
	XOR			逻辑异或		√	√
	OUT			线圈输出	√	√	√
	SET			置位保持	√	√	√
	RST			复位	√	√	√
	ALT			ON/OFF 交替输出	√	√	√
	ZRST			批量复位	√	√	√

指令类型	指令名称	8 位模式	32 位模式	指令功能	支持语言		
					LD	FBD	IL
	ENO			取 ENO 输出			√
计时器	TON			延时开	√	√	√
	TOF			延时关	√	√	√
	TP			脉冲计时器	√	√	√
计数器	CTU		D.CTU	加计数器	√	√	√
	CTD		D.CTD	减计数器	√	√	√
	CTUD		D.CTUD	加减计数器	√	√	√
高速控制指令	RESH			IO 更新	√	√	√
	HHSC			高速计数器	√	√	√
	HCWR			写高速计数器	√	√	√
	SPD			速度侦测	√	√	√
	PWM			脉宽调制	√	√	√
	PLSY		D.PLSY	脉冲输出	√	√	√
	PLSR		D.PLSR	加减速脉冲输出	√	√	√
	ZRN			原点回归	√	√	√
	SETZ			设置电气原点	√	√	√
	PPMR			直线插补	√	√	√
	CIMR			圆弧插补	√	√	√
	SPLS			简单脉冲输出	√	√	√
	MPTO			多段脉冲输出	√	√	√
	SYNP			随动脉冲输出	√	√	√
	PSTOP			停止脉冲输出	√	√	√
比较指令	CMP		D.CMP	比较指令	√	√	√
	ZCP		D.ZCP	区域比较	√	√	√
	MATC		D.MATC	数值匹配	√	√	√
	ABSC		D.ABSC	绝对凸轮比较	√	√	√
	BON			ON 位判定	√	√	√
	BONC		D.BONC	ON 位数量	√	√	√
	MAX		D.MAX	最大值	√	√	√
	MIN		D.MIN	最小值	√	√	√
	SEL		D.SEL	条件选择	√	√	√
	MUX		D.MUX	多路选择	√	√	√
移动指令	LBST			低字节赋值	√	√	√
	HBST			高字节赋值	√	√	√
	MOV		D.MOV	移动	√	√	√
	BMOV			块移动	√	√	√
	FILL			填充	√	√	√
	XCH			字节交换	√	√	√
	BXCH			块交换	√	√	√
	SHL			位左移	√	√	√
	SHR			位右移	√	√	√
	WSHL			字左移	√	√	√

指令类型	指令名称	8 位模式	32 位模式	指令功能	支持语言		
					LD	FBD	IL
	WSHR			字右移	√	√	√
	ROL			位循环左移	√	√	√
	ROR			位循环右移	√	√	√
	WROL			字循环左移	√	√	√
	WROR			字循环右移	√	√	√
	BSHL			字节左移	√	√	√
	BSHR			字节右移	√	√	√
	ATBL			添加到队列	√	√	√
	FIFO			先进先出	√	√	√
	LIFO			后进先出	√	√	√
	SORT			数据排序	√	√	√
数据转换指令	ENCO			编码器	√	√	√
	DECO			译码器	√	√	√
	BTOW			位转换为字	√	√	√
	WTOB			字转换为位	√	√	√
	HEX	HEX.LB		ASCII 转换为 16 进制	√	√	√
	ASCI	ASCI.LB		16 进制转换为 ASCII	√	√	√
	BUNB			离散位组合到连续位	√	√	√
	BUNW			离散位组合到连续字	√	√	√
	WUNW			离散字组合到连续字	√	√	√
	BDIB			连续位分散到离散位	√	√	√
	WDIB			连续字分散到离散位	√	√	√
	WDIW			连续字分散到离散字	√	√	√
	BCD		D.BCD	BIN 转换为 BCD	√	√	√
	BIN		D.BIN	BCD 转换为 BIN	√	√	√
	ITOL			整数转换为长整数	√	√	√
	GRAY			BIN 转换为 GRAY 码	√	√	√
	GBIN			GRAY 码转换为 BIN	√	√	√
字符指令	GHLB			得出高低字节	√	√	√
	GETB			截取字节串	√	√	√
	BCMP	BCMP.LB		字节串比较	√	√	√
	ITOC		D.ITOC	整数转换为字符	√	√	√
	CTOI			字符转换为整数	√	√	√
	FTOC			浮点数转换为字符	√	√	√
	CTOF			字符转换为浮点数	√	√	√
数学运算指令	WNOT		D.WNOT	取反	√	√	√
	WAND		D.WAND	与运算	√	√	√
	WOR		D.WOR	或运算	√	√	√
	WXOR		D.WXOR	异或运算	√	√	√
	ADD		D.ADD	加法	√	√	√
	SUB		D.SUB	减法	√	√	√
	INC		D.INC	加 1	√	√	√

指令类型	指令名称	8 位模式	32 位模式	指令功能	支持语言		
					LD	FBD	IL
	DEC		D.DEC	减 1	√	√	√
	MUL		D.MUL	乘法	√	√	√
	DIV		D.DIV	除法	√	√	√
	ACCU		D.ACCU	累加	√	√	√
	AVG		D.AVG	平均值	√	√	√
	ABS		D.ABS	绝对值	√	√	√
	NEG		D.NEG	求 2 的补码	√	√	√
浮点数指令	FCMP			浮点数比较	√	√	√
	FZCP			浮点数区域比较	√	√	√
	FMOV			浮点数移动指令	√	√	√
	FADD			浮点数加法	√	√	√
	FSUB			浮点数减法	√	√	√
	FMUL			浮点数乘法	√	√	√
	FDIV			浮点数除法	√	√	√
	FACCU			浮点数累加	√	√	√
	FAVG			浮点数平均值	√	√	√
	FMAX			浮点数最大值	√	√	√
	FMIN			浮点数最小值	√	√	√
	FTOI			浮点数转换为整数	√	√	√
	ITOF		D.ITOF	整数转换为浮点数	√	√	√
	FABS			浮点数绝对值	√	√	√
	FSQR			浮点数平方根	√	√	√
	FSIN			正弦值	√	√	√
	FCOS			余弦值	√	√	√
	FTAN			正切值	√	√	√
	FASIN			反正弦	√	√	√
	FACOS			反余弦	√	√	√
	FATAN			反正切	√	√	√
	FLN			自然对数	√	√	√
	FLOG			以 10 为底的对数	√	√	√
	FEXP			自然指数	√	√	√
	FRAD			角度转换为弧度	√	√	√
	FDEG			弧度转换为角度	√	√	√
	FXI			指数指令	√	√	√
时钟指令	TCMP			实时时钟比较	√	√	√
	TACCU			时间累计	√	√	√
	SCLK			设置时钟	√	√	√
	TIME			时间开关	√	√	√
	DATE			日期开关	√	√	√
	INVT			倒计时	√	√	√
通讯指令	SUM	SUM.LB		SUM 累加和校验	√	√	√
	BCC	BCC.LB		BCC 校验	√	√	√

指令类型	指令名称	8 位模式	32 位模式	指令功能	支持语言		
					LD	FBD	IL
	CRC	CRC.LB		CRC 校验	√	√	√
	LRC	LRC.LB		LRC 校验	√	√	√
	COMM	COMM.LB		串行通讯	√	√	√
	MODR			Modbus 读	√	√	√
	MODW			Modbus 写	√	√	√
	HWRD			Haiwellbus 读	√	√	√
	HWWR			Haiwellbus 写	√	√	√
	RCV			接收通讯数据	√	√	√
	XMT	XMT.LB		发送通讯数据	√	√	√
	FROM			扩展模块 CR 数据读取	√	√	√
	TO			扩展模块 CR 数据写入	√	√	√
中断指令	ATCH			中断绑定	√	√	√
	DTCH			中断释放	√	√	√
	ENI			允许中断	√	√	√
	DISI			禁止中断	√	√	√
程序控制指令	MC			主控	√	√	√
	MCR			主控清除	√	√	√
	FOR			循环指令	√	√	√
	NEXT			循环结束	√	√	√
	WAIT			延时等待	√	√	√
	CALL			调用子程序	√	√	√
	EXIT			条件返回	√	√	√
	REWD			扫描时间复位	√	√	√
	JMPC			条件跳转	√	√	√
	LBL			跳转标号	√	√	√
特殊功能指令	GPWM			通用脉宽调制	√	√	√
	FTC			模糊温度控制	√	√	√
	PID			PID 控制	√	√	√
	HAL		D.HAL	上限报警	√	√	√
	LAL		D.LAL	下限报警	√	√	√
	LIM		D.LIM	范围限制	√	√	√
	SC		D.SC	线性变换	√	√	√
	VC			阀门控制	√	√	√
	TTC			温度曲线控制	√	√	√

4.1 指令通用说明

- 1、En 使能输入：En 为指令的使能输入端，只有当 En 有电（ON）时，指令被执行，否则指令不执行。
- 2、Eno 使能输出：Eno 为指令的使能输出端，指示该指令是否被正确执行。当 En 有电（ON）并且指令被正确执行时那么 Eno 输出有电（ON），当 En 没有电（OFF）或者指令在执行时出现错误（例如：指令参数不正确）那么 Eno 输出没有电（OFF）。在 LD、FBD 语言的应用指令中，大部分的指令均有 Eno 使能输出端，所有 IL 指令均无 Eno 输出端，由专用于 IL 语言的 ENO 指令代替。
- 3、LD 语言无 AND、OR、XOR 三个指令，由逻辑链路代替。

- 4、32 位指令在 16 位指令名称前加“D.”，表示使用 2 个连续的寄存器。如加法指令 ADD，16 位加法为 ADD，32 位加法为 D.ADD。
- 5、8 位指令在 16 位指令名称后面加“.LB”，表示只使用寄存器的低字节。如串行通讯指令 COMM，16 位指令为 COMM，8 位指令为 COMM.LB。
- 6、对于一些指令的参数端将自动占用几个连续元件的情况，在编写程序时要特别注意，应避免元件被重用而使程序执行不正确。

注：除了 CV48~CV79 为 32 位寄存器外（共 32 个），Haiwell（海为）PLC 的其他寄存器（AI、AQ、V、SV、LV、TV、CV、P）都是 16 位寄存器，一个 16 位寄存器由 2 个字节组成，一个 32 位寄存器由 2 个连续的 16 位寄存器组成。

4.2 比较开关

比较开关为 LD 编程语言专用，类型分为：16 位比较指令、32 位比较指令、浮点数比较指令、低字节比较指令、高字节比较指令。

比较方式分为：等于 (=)、不等于 (<>)、大于 (>)、大于等于 (≥)、小于 (<)、小于等于 (≤) 六种。

指令列表如下：

指令名称	8 位模式	32 位模式	指令功能	支持语言		
				LD	FBD	IL
=	LB.= HB.=	D.=	等于比较开关，分 16 位/32 位/低字节/高字节	√		
<>	LB.<> HB.<>	D.<>	不等于比较开关，分 16 位/32 位/低字节/高字节	√		
>	LB.> HB.>	D.>	大于比较开关，分 16 位/32 位/低字节/高字节	√		
>=	LB.>= HB.>=	D.>=	大于等于比较开关，分 16 位/32 位/低字节/高字节	√		
<	LB.< HB.<	D.<	小于比较开关，分 16 位/32 位/低字节/高字节	√		
<=	LB.<= HB.<=	D.<=	小于等于比较开关，分 16 位/32 位/低字节/高字节	√		
F.=			浮点数等于比较开关	√		
F.<>			浮点数不等于比较开关	√		
F.>			浮点数大于比较开关	√		
F.>=			浮点数大于等于比较开关	√		
F.<			浮点数小于比较开关	√		
F.<=			浮点数小于等于比较开关	√		

4.3 步进指令

步进指令列表

指令名称	8 位模式	32 位模式	指令功能	支持语言		
				LD	FBD	IL
<u>STL</u>			步开始	√		
<u>SFROM</u>			步合并	√		
<u>STO</u>			步转移	√		

【步进指令说明】

- 1、子程序、中断程序不支持步进指令，FBD，IL 语言不支持步进指令。
- 2、步内不支持跳转、循环指令。
- 3、如果某步有电，执行步内所有程序，否则该步进点内所有指令不会被执行。
- 4、支持步分支、步合并流程。
- 5、步之间跳转时，清除上一个步进点状态以及复位该步中 OUT 指令输出、定时器线圈 T 及当前值 TV、计数器线圈 C 及当前值 CV。

SET 指令不会被复位。

- 6、若步进转出后，Y 输出点仍需保持 ON 则需用 SET 指令推动该输出点，要清除该输出点为 OFF，则需用 RST 指令。
- 7、步进点 Sn 不能重复，如果没有步进指令则 S 步进继电器可以作为普通内部继电器使用。
- 8、如果要结束某步，使用 RST Sx 复位该步即可，批量复位使用 ZRST 指令。
- 9、任何 S 元件都可以作为起始步，起始步使用 STO 或者 SET 启动，步跳转使用 STO 指令。
- 10、程序可以同时启动最大 10 个不同的步流程。

注：E/S 系列主机在步转移或者步复位时不支持复位计数器线圈 C 及当前值 CV。

STL（步开始）

指令格式及参数说明

语言	LD	FBD	IL
指令格式		无	无

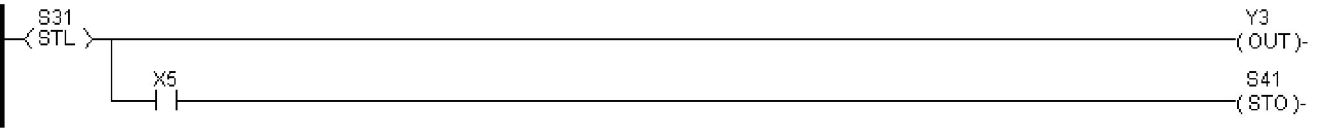
【指令功能及作用说明】

STL 指令代表一个步进点的开始，如果该步进点有电，执行步内所有程序，否则该步进点内所有指令不会被执行。

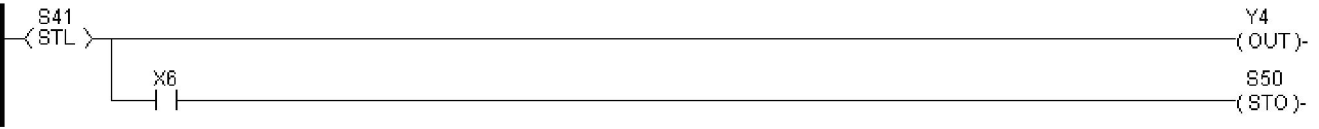
【指令示例】



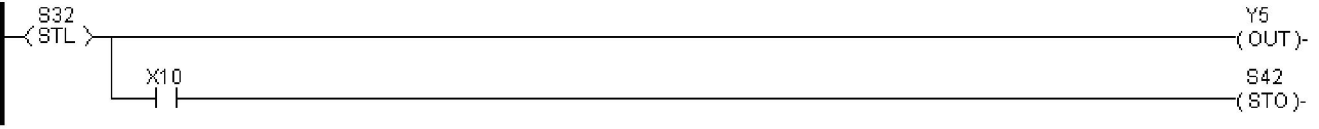
//Network 6



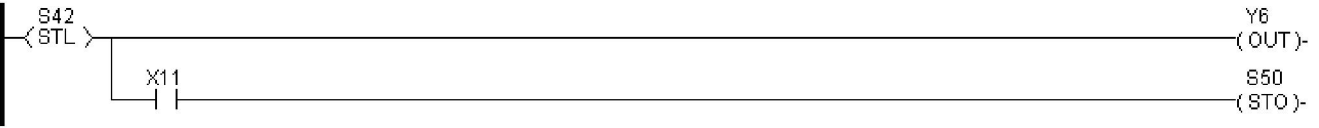
//Network 7



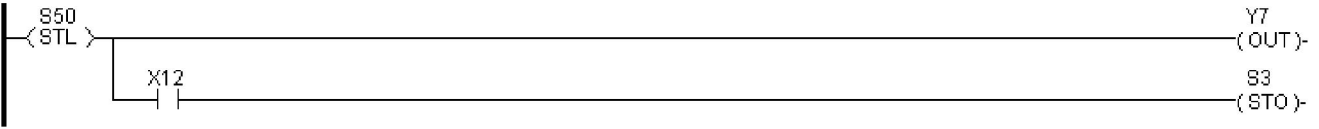
//Network 8



//Network 9



//Network 10



【程序说明】：

- 1、当 M2=ON 时 启动步进点 S3，这时 S3=ON
- 2、当 S3=ON，且转移条件 X0=ON 时，则转移到步进点 S20，这时 S20=ON、S3=OFF
- 3、当 S20=ON 则 Y0=ON，当转移条件 X1=ON 时，则转移到步进点 S30，这时 S30=ON、S20=OFF、Y0=OFF
，当转移条件 X4=ON 时，则转移到步进点 S31，这时 S31=ON、S20=OFF、Y0=OFF
，当转移条件 X7=ON 时，则转移到步进点 S32，这时 S32=ON、S20=OFF、Y0=OFF
(步选择性分支)
- 4、当 S30=ON 则 Y1=ON，当转移条件 X2=ON 时，则转移到步进点 S40，这时 S40=ON、S30=OFF、Y1=OFF
- 5、当 S40=ON 则 Y2=ON，当转移条件 X3=ON 时，则转移到步进点 S50，这时 S50=ON、S40=OFF、Y2=OFF
- 6、当 S31=ON 则 Y3=ON，当转移条件 X5=ON 时，则转移到步进点 S41，这时 S41=ON、S31=OFF、Y3=OFF
- 7、当 S41=ON 则 Y4=ON，当转移条件 X6=ON 时，则转移到步进点 S50，这时 S50=ON、S41=OFF、Y4=OFF
- 8、当 S32=ON 则 Y5=ON，当转移条件 X10=ON 时，则转移到步进点 S42，这时 S42=ON、S32=OFF、Y5=OFF
- 9、当 S42=ON 则 Y6=ON，当转移条件 X11=ON 时，则转移到步进点 S50，这时 S50=ON、S42=OFF、Y6=OFF
- 10、当 S50=ON 则 Y7=ON，当转移条件 X12=ON 时，则转移到步进点 S3，这时 S3=ON、S50=OFF、Y7=OFF，循环反复。

SFROM（步合并）

指令格式及参数说明

语言	LD	FBD	IL
----	----	-----	----

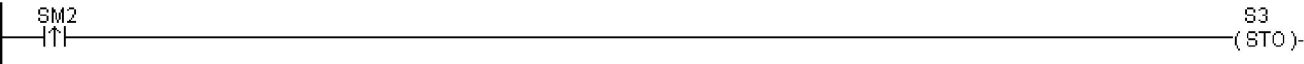
语言	LD	FBD	IL
指令格式	Sxxx S	无	无

【指令功能及作用说明】

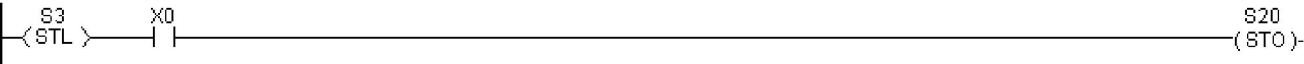
SFROM 指令用于步并行分支后的合并。

【指令示例】

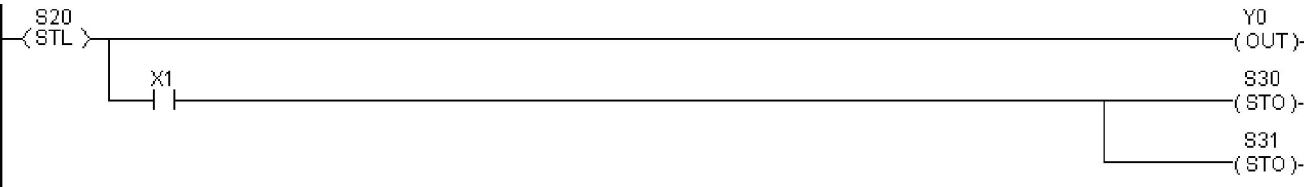
//Network 1



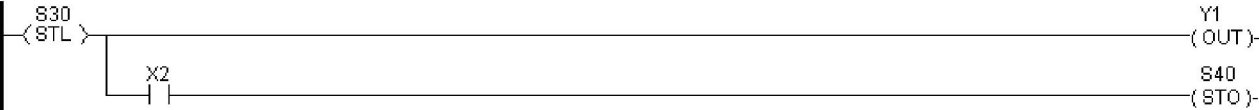
//Network 2



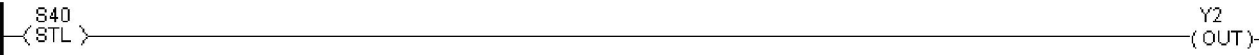
//Network 3



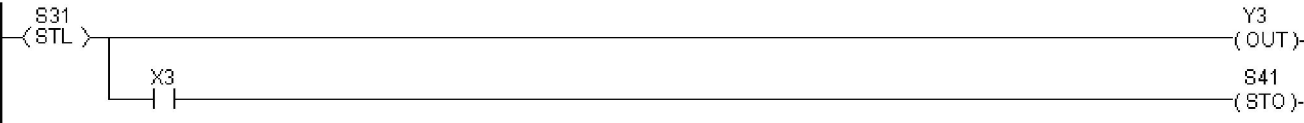
//Network 4



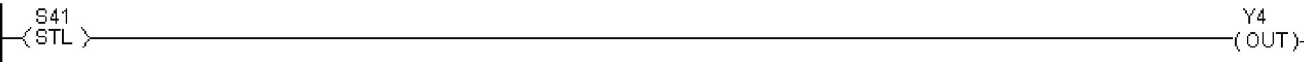
//Network 5



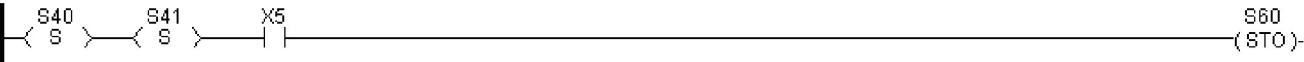
//Network 6



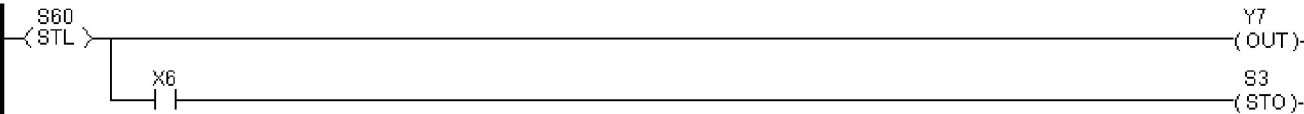
//Network 7



//Network 8



//Network 9



//Network 10



【程序说明】：

- 1、当程序启动首个扫描周期 SM2=ON，启动步进点 S3，这时 S3=ON

- 2、当 S3=ON，且转移条件 X0=ON 时，则转移到步进点 S20，这时 S20=ON、S3=OFF
- 3、当 S20=ON 则 Y0=ON，当转移条件 X1=ON 时，则同时转移到步进点 S30 和 S31（步并行分支），这时 S30=ON、S31=ON、S20=OFF、Y0=OFF
- 4、当 S30=ON 则 Y1=ON，当转移条件 X2=ON 时，则转移到步进点 S40，这时 S40=ON、S30=OFF、Y1=OFF
- 5、当 S40=ON 则 Y2=ON
- 6、当 S31=ON 则 Y3=ON，当转移条件 X3=ON 时，则转移到步进点 S41，这时 S41=ON、S31=OFF、Y3=OFF
- 7、当 S41=ON 则 Y4=ON
- 8、当 S40=ON 并且 S41=ON（步并行分支合并），且转移条件 X5=ON 时，则转移到步进点 S60，这时 S60=ON、S40=OFF、Y2=OFF、S41=OFF、Y4=OFF
- 9、当 S60=ON 则 Y7=ON，当转移条件 X6=ON 时，则转移到步进点 S3，这时 S3=ON、S60=OFF、Y7=OFF，循环反复。
- 10、如果 M2=ON，那么批量复位从 S0 开始的 100 个步，即 S0~S99。

STO（步转移）

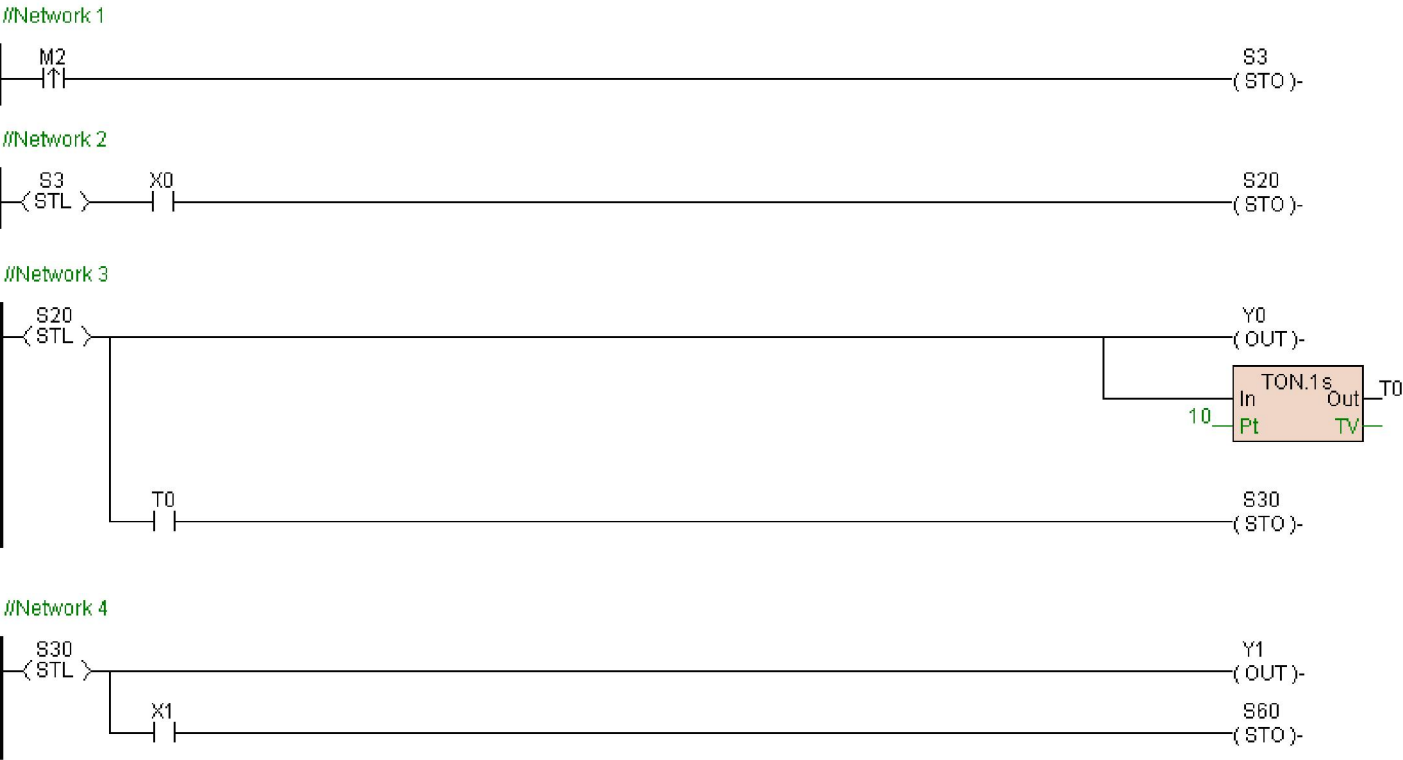
指令格式及参数说明

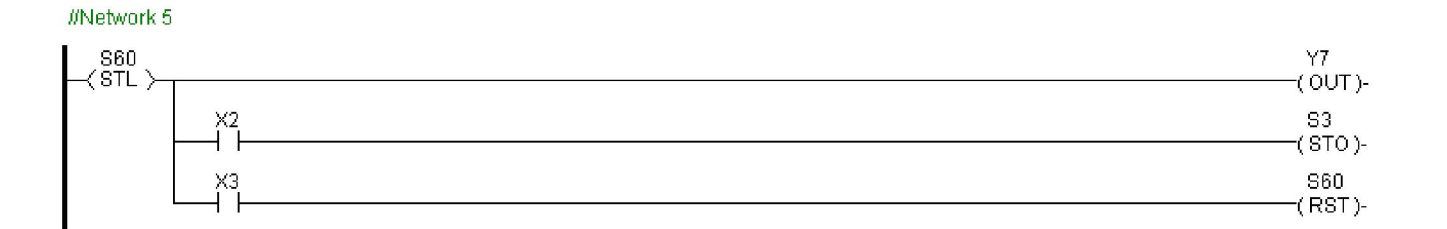
语言	LD	FBD	IL
指令格式	Sxxx —(STO)-	无	无

【指令功能及作用说明】

STO 用于启动一个步流程，或者将程序流程转移到指定的步进点。

【指令示例】





【程序说明】：

- 1、当 M2=ON 时 启动步进点 S3，这时 S3=ON
- 2、当 S3=ON，且转移条件 X0=ON 时，则转移到步进点 S20，这时 S20=ON、S3=OFF
- 3、当 S20=ON 则 Y0=ON，同时启动计时器 T0，当 T0 计时时间到，则转移到步进点 S30，这时 S30=ON、S20=OFF、Y0=OFF、T0=OFF
- 4、当 S30=ON 则 Y1=ON，当转移条件 X1=ON 时，则转移到步进点 S60，这时 S60=ON、S30=OFF、Y1=OFF
- 5、当 S60=ON 则 Y7=ON，当转移条件 X2=ON 时，则转移到步进点 S3，这时 S3=ON、S60=OFF、Y7=OFF，循环反复。
- ，当退出条件 X3=ON 时，则复位步进点 S60，这时 S60=ON、Y7=OFF，结束步进。

4.4 位指令

位指令列表

指令名称	8 位模式	32 位模式	指令功能	支持语言		
				LD	FBD	IL
AND			逻辑与		√	√
OR			逻辑或		√	√
XOR			逻辑异或		√	√
OUT			线圈输出	√	√	√
SET			置位保持	√	√	√
RST			复位	√	√	√
ALT			ON/OFF 交替输出	√	√	√
ZRST			批量复位	√	√	√
ENO			取 ENO 输出			√

AND（逻辑与）

指令格式及参数说明

语言	LD	FBD	IL
指令格式	无		AND In1, In2 [, ... , In15], Out

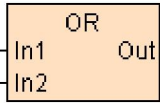
参数	参数定义	输入	输出	说明
In1	操作位 1	√		
In2	操作位 2	√		
...	...	...		
In15	操作位 15	√		
Out	状态输出		√	

【指令功能及作用说明】

AND 指令对一组输入的位元件进行逻辑与运算后输出, 只有 In1~In15 输入状态全部为 ON 时 Out=ON, 否则 Out=OFF, 它支持 2~15 个可变输入。

OR（逻辑或）

指令格式及参数说明

语言	LD	FBD	IL
指令格式	无		OR In1, In2 [, ... , In15], Out

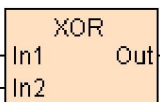
参数	参数定义	输入	输出	说明
In1	操作位 1	√		
In2	操作位 2	√		
...	...	...		
In15	操作位 15	√		
Out	状态输出		√	

【指令功能及作用说明】

OR 指令对一组输入的位元件进行逻辑或运算后输出, 只要 In1~In15 其中一个输入状态为 ON 则 Out=ON, 否则 Out=OFF, 它支持 2~15 个可变输入。

XOR（逻辑异或）

指令格式及参数说明

语言	LD	FBD	IL
指令格式	无		XOR In1, In2 [, ... , In15], Out

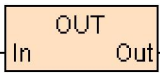
参数	参数定义	输入	输出	说明
In1	操作位 1	√		
In2	操作位 2	√		
...	...	...		
In15	操作位 15	√		
Out	状态输出		√	

【指令功能及作用说明】

XOR 指令对一组输入的位元件进行逻辑异或运算后输出, 当 In1~In15 输入中状态为 ON 的个数与状态为 OFF 的个数不同则 Out=ON, 否则 Out=OFF, 它支持 2~15 个可变输入。

OUT（线圈输出）

指令格式及参数说明

语言	LD	FBD	IL
指令格式	—(OUT)—		OUT In, Out

参数	参数定义	输入	输出	说明
In	输入	√		
Out	输出		√	

【指令功能及作用说明】

OUT 指令把输入状态直接赋给输出，In=ON 则 Out=ON，In=OFF 则 Out=OFF。

SET（置位保持）

指令格式及参数说明

语言	LD	FBD	IL
指令格式	—(SET)-		SET In, Out

参数	参数定义	输入	输出	说明
In	输入	√		
Out	输出		√	

【指令功能及作用说明】

SET 指令根据输入状态对输出置位，In=ON 则 Out=ON，In=OFF 则 Out 保持原来状态不变。SET 指令一般用边沿输入执行。

RST（复位）

指令格式及参数说明

语言	LD	FBD	IL
指令格式	—(RST)-		RST In, Out

参数	参数定义	输入	输出	说明
In	输入	√		
Out	输出		√	

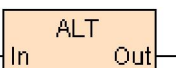
【指令功能及作用说明】

- 1、RST 指令根据输入状态对输出复位，In=ON 则 Out=OFF，In=OFF 则 Out 保持原来状态不变。RST 指令一般用边沿输入执行。
- 2、如果 Out 是计时器 Tx，同时复位计时器线圈 T 及当前值 TV，如果输出是计时器 Cx，同时复位计数器线圈 C 及当前值 CV。
- 3、如果 Out 是步进状态位 S，除了复位该步进点状态外，如果该步进点在执行中会同时复位该步进点内的 OUT 指令输出、计时器线圈 T 及当前值 TV、计数器线圈 C 及当前值 CV。

注：E/S 系列主机的 RST 指令不支持复位 S。

ALT（ON/OFF 交替输出）

指令格式及参数说明

语言	LD	FBD	IL
指令格式	—(ALT)-		ALT In, Out

参数	参数定义	输入	输出	说明
In	输入	√		

参数	参数定义	输入	输出	说明
Out	输出		√	

【指令功能及作用说明】

ALT 指令根据输入状态对输出状态取反，In=ON 则 Out 对自身取反，In=OFF 则 Out 保持原来状态不变。ALT 指令一般用边沿输入执行。

ZRST（批量复位）

指令格式及参数说明

语言	LD	FBD	IL
指令格式			ZRST En, N, Des

参数	参数定义	输入	输出	说明
En	使能输入	√		
N	复位元件个数	√		
Eno	使能输出		√	
Des	复位起始元件		√	

【指令功能及作用说明】

- 1、ZRST 指令批量复位以 Des 为起始的 N 个元件。ZRST 指令一般用边沿输入执行。
- 2、如果 Des 是计时器 Tx，会复位计时器线圈 T 及当前值 TV，如果输出是计时器 Cx，会复位计数器线圈 C 及当前值 CV。
- 3、如果 Des 是步进状态位 S，除了复位该步进点状态外，如果该步进点在执行中会同时复位该步进点内的 OUT 指令输出、计时器线圈 T 及当前值 TV、计数器线圈 C 及当前值 CV。

注：E/S 系列主机不支持 ZRST 指令。

ENO（取 ENO 输出）

指令格式及参数说明

语言	LD	FBD	IL
指令格式	无	无	ENO Out

参数	参数定义	输入	输出	说明
Out	输出		√	

【指令功能及作用说明】

- 1、IL 语言的所有指令均无 Eno 使能输出端，为使用 IL 语言编写的程序与 FBD、LD 的功能相同，在 IL 语言的应用指令中特别增加 ENO 指令，其功能相当于 FBD、LD 语言中应用指令的 Eno 使能输出端的作用。
- 2、ENO 指令只有一个输出端，其输出端的状态只与排在本 ENO 指令前面最靠近 ENO 指令的在 FBD 或 LD 语言中有 Eno 输出端的指令有关。

【指令示例】

```
AND      X0, QX2, M100
MAX      M100, V1000, 3, V10 //V10等于V1000、V1001、V1002这3个寄存器的最大值
OUT      X2, Y0
ENO      M0 //取MAX指令的Eno使能输出，如果MAX指令执行正确，那么M0=ON
5 ADD    M0, V10, 200, AQ0
```

【程序说明】

- 1、MAX 指令为在 FBD 或 LD 语言中有 Eno 输出端的指令，且最靠近 ENO 指令（因为 OUT 指令没有 Eno 输出端），所以在程序中，ENO 指令的输出端 M0 的状态只与 MAX 指令的执行情况有关。
- 2、当 X0=ON 时（X2 为常闭点），M100=ON
- 3、M100=ON，MAX 指令执行，V10 等于 V1000、V1001、V1002 这 3 个寄存器的最大值，MAX 指令的 Eno 输出=ON
- 4、X2=OFF，Y0=OFF
- 5、ENO 指令取前面指令的 Eno 输出，因为 MAX 指令的 Eno 输出=ON，所以 M0 状态为 ON
- 6、M0=ON，ADD 指令执行，AQ0=V10+200

4.5 计时器

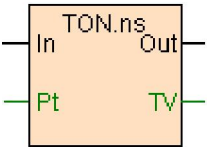
计时器列表

指令名称	8 位模式	32 位模式	指令功能	支持语言		
				LD	FBD	IL
TON			延时开	√	√	√
TOF			延时关	√	√	√
TP			脉冲计时器	√	√	√

注：T252~T255 的时基固定为 1ms。其他计时器时基可 10ms、100ms、1s 任意设置。

TON（延时开）

指令格式及参数说明

语言	LD	FBD	IL
指令格式			TON.ns In, Pt, Tx

参数	参数定义	输入	输出	说明
ns	时基值			T252~T255 的时基值为 1ms。其他可 10ms、100ms、1s 任意设置
In	输入	√		
Pt	设定时间	√		
Out	计时器线圈 Tx		√	
TV	当前时间		√	

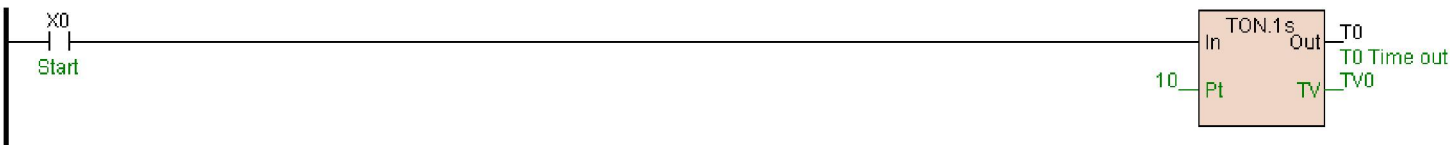
【指令功能及作用说明】

- 1、TON 为延时开指令，当 In=ON，启动计时器计时，TV 为计时器当前值，当 TV 等于设定时间（计时时间到），Out（计时器线圈 Tx）=ON，并停止计时。当 In 变为 OFF，Out（计时器线圈 Tx）=OFF，并且 TV 值清为零。若在计时过程中（计时时间未到），In 变为 OFF，则计时器停止计时，Out（计时器线圈 Tx）=OFF，并且 TV 值清零。

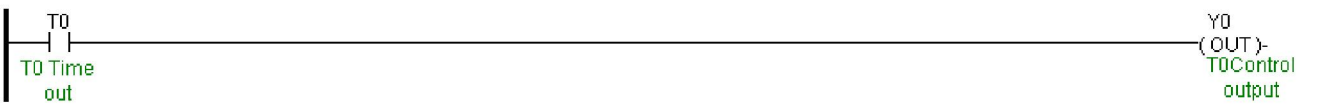
2、计时时间= 时基（ns）× 设定时间（Pt）。如：时基为 1s，设定时间 Pt=10 时，则其延时开时间为 1s × 10 = 10s。

【指令示例】

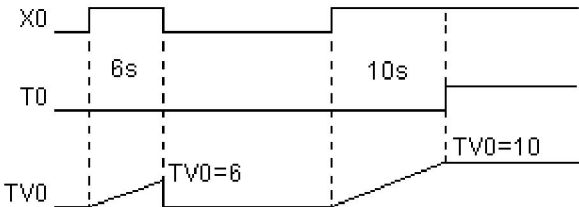
//Network 1



//Network 2



【程序示意图】



【程序说明】

计时时间= 时基（1s）× 设定时间（Pt=10）=10s。当 X0=ON，计时器 T0 开始计时，当 TV0=10，T0=ON（Y0=ON）并且停止计时。如果 X0=OFF，那么 T0=OFF（Y0=OFF），TV=0。

TOF（延时关）

指令格式及参数说明

语言	LD	FBD	IL
指令格式			TOF.ns In, Pt, Tx

参数	参数定义	输入	输出	说明
ns	时基值			T252~T255 的时基值为 1ms。其他可 10ms、100ms、1s 任意设置
In	输入	√		
Pt	设定时间	√		
Out	计时器线圈 Tx		√	
TV	当前时间		√	

【指令功能及作用说明】

1、TOF 为延时关指令，当 In=ON，Out（计时器线圈 Tx）=ON，当 n 的状态由 ON 变为 OFF 时，启动计时器计时，TV 为计时器当前值，当 TV 等于设定时间（计时时间到），Out（计时器线圈 Tx）=OFF，并停止计时，TV=0。若在计时过程中（计时时间未到），In 的状态变为 ON，则 Out（计时器线圈 Tx）=ON，计时器停止计时，TV=0。

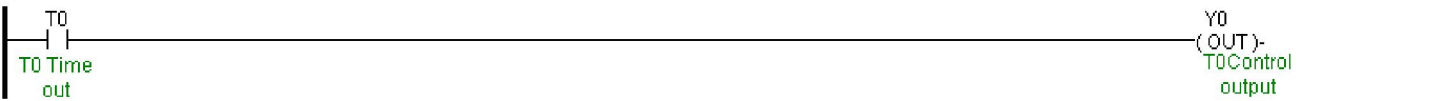
2、计时时间= 时基（ns）× 设定时间（Pt）。如：时基为 10ms，设定时间 Pt=1000 时，则其延时开时间为 10ms × 1000 = 10s。

【指令示例】

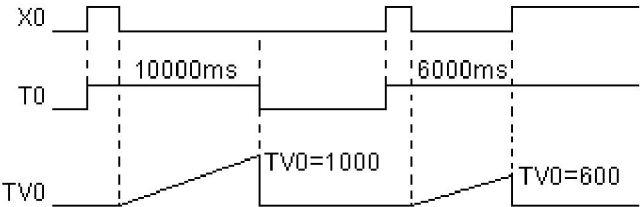
//Network 1



//Network 2



【程序示意图】



【程序说明】

计时时间= 时基（10ms）× 设定时间（Pt=1000）=10s。当 X0=ON，T0=ON（Y0=ON），当 X0=OFF，计时器 T0 开始计时，当 TV0=1000，T0=OFF（Y0=OFF）并且停止计时 TV=0。如果在计时时间未到 X0=ON，那么 T0=ON（Y0=ON），停止计时 TV=0。

TP（脉冲计时器）

指令格式及参数说明

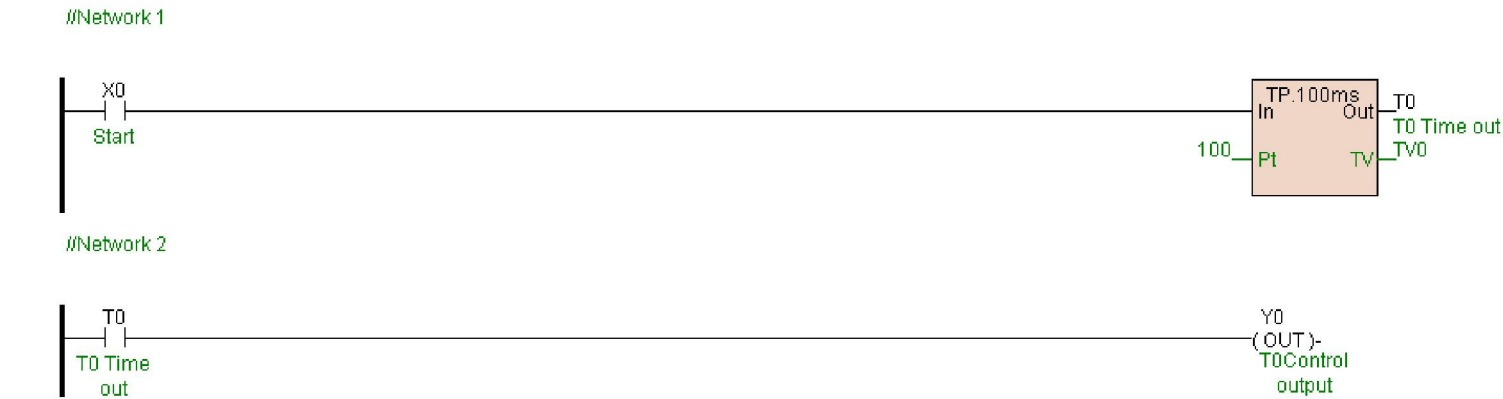
语言	LD	FBD	IL
指令格式			TP.ns In, Pt, Tx

参数	参数定义	输入	输出	说明
ns	时基值			T252~T255 的时基值为 1ms。其他可 10ms、100ms、1s 任意设置
In	输入	√		
Pt	设定时间	√		
Out	计时器线圈 Tx		√	
TV	当前时间		√	

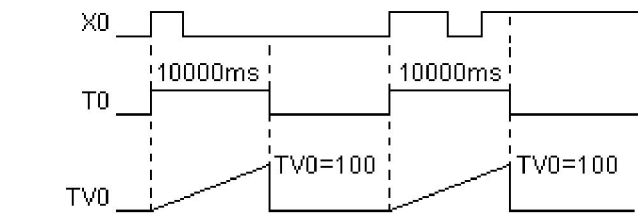
【指令功能及作用说明】

- 1、TP 为脉冲计时器，当 In=ON，Out（计时器线圈 Tx）=ON，启动计时器计时，TV 为计时器当前值，当 TV 等于设定时间（计时时间到），Out（计时器线圈 Tx）=OFF，停止计时并且 TV 值清为零。在计时过程中（计时时间未到），无论 In 的状态如何变化，计时器保持计时，Out（计时器线圈 Tx）保持为 ON。
- 2、计时时间= 时基（ns）× 设定时间（Pt）。如：若时基为 100ms，设定时间 Pt=100 时，则其延时开时间为 100ms × 100 = 10s。

【指令示例】



【程序示意图】



【程序说明】

计时时间= 时基 (100ms) × 设定时间 (Pt=100) =10s。当 X0=ON, T0=ON (Y0=ON)，计时器 T0 开始计时，当 TV0=100, T0=OFF (Y0=OFF) 并且停止计时 TV0=0。在计时状态下，无论 X0 状态如何变化，计时器保持计时，T0 保持为 ON 直到计时结束。

4.6 计数器

计数器列表

指令名称	8 位模式	32 位模式	指令功能	支持语言		
				LD	FBD	IL
CTU		D.CTU	加计数器	√	√	√
CTD		D.CTD	减计数器	√	√	√
CTUD		D.CTUD	加减计数器	√	√	√

注：C48~C79 为 32 位计数器，其他计数器都是 16 位计数器。

CTU、D.CTU（加计数器）

指令格式及参数说明

语言	LD	FBD	IL
16、32 位 指令格式			CTU Cu, PV, Cx D.CTU Cu, PV, Cx

参数	参数定义	输入	输出	说明
Cu	加计数输入	√		
PV	计数器设定值	√		
Out	计数器线圈 Cx		√	其中 C48~C79 为 32 位计数器，共 32 点
CV	计数器当前值		√	

【指令功能及作用说明】

CTU 为 16 位加计数器指令（D.CTU 为 32 位），当加计数输入 Cu 由 OFF 变为 ON 时，计数器加 1，当 CV 大于等于 PV，Out（计

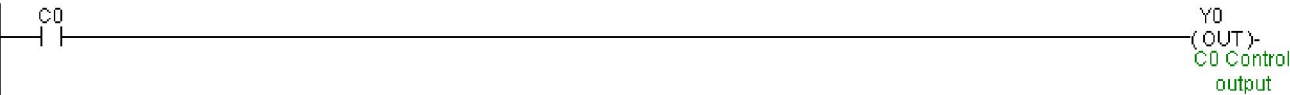
数器线圈 Cx) =ON。计数到后，若再有计数脉冲输入，计数将继续，CV 值继续加 1，直到最大值（16 位计数器最大值为 32767，32 位计数器最大值为 2147483647），到最大值后不再计数。

【指令示例】

//Network 1



//Network 2



【程序说明】

当 X0 由 OFF 到 ON 变化一次，CV0 加 1，当 CV0 ≥ 10 时计数到，C0=ON（Y0=ON）。

CTD、D.CTD（减计数器）

指令格式及参数说明

语言	LD	FBD	IL
16、32 位 指令格式			CTD Cd, PV, Cx D.CTD Cd, PV, Cx

参数	参数定义	输入	输出	说明
Cd	减计数输入	√		
PV	计数器设定值	√		
Out	计数器线圈 Cx		√	其中 C48~C79 为 32 位计数器，共 32 点
CV	计数器当前值		√	

【指令功能及作用说明】

- 1、CTD 为 16 位减计数器指令（D.CTD 为 32 位），当减计数输入 Cd 由 OFF 变为 ON 时，计数器减 1，当 CV=0 则计数到，Out（计数器线圈 Cx）=ON。计数到后，若再有计数脉冲输入，不再计数。
- 2、CTD 指令被加载或复位时，CV = PV，即 CTD 是从设定值开始一直减到 0。

【指令示例】

//Network 1



//Network 2



【程序说明】

程序运行时 CV=10，当 X0 由 OFF 到 ON 变化一次，CV0 减 1，当 CV0 = 0 时计数到，C0=ON（Y0=ON）。

CTUD、D.CTUD（加减计数器）

指令格式及参数说明

语言	LD	FBD	IL
16、32 位 指令格式			CTUD Cu, Cd, PV, Cx D.CTUD Cu, Cd, PV, Cx

参数	参数定义	输入	输出	说明
Cu	加计数输入	√		
Cd	减计数输入	√		
PV	计数器设定值	√		
Out	计数器线圈 Cx		√	其中 C48~C79 为 32 位计数器，共 32 点
CV	计数器当前值		√	

【指令功能及作用说明】

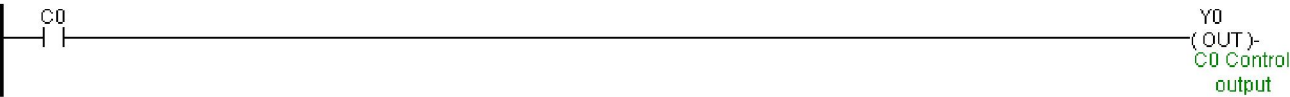
- 1、CTUD 为 16 位加减计数器指令（D.CTUD 为 32 位），当加计数输入 Cu 由 OFF 变为 ON 时，计数器加 1，当减计数输入 Cd 由 OFF 变为 ON 时，计数器减 1，当 CV 大于等于 PV，Out（计数器线圈 Cx）=ON，当 CV 小于 PV，Out（计数器线圈 Cx）=OFF。
- 2、16 位计数器 CV 的最大值为 32767，最小值为-32768，32 位计数器 CV 的最大值为 2147483647，最小值为-2147483648。

【指令示例】

//Network 1



//Network 2



【程序说明】

当 X0 由 OFF 到 ON 变化一次，CV0 加 1，当 X1 由 OFF 到 ON 变化一次，CV0 减 1，当 CV0 ≥ 10 时，C0=ON（Y0=ON），当 CV0 <10 时，C0=OFF（Y0=OFF）。

4.7 高速控制指令

高速控制指令列表

指令名称	8 位模式	32 位模式	指令功能	支持语言		
				LD	FBD	IL

指令名称	8 位模式	32 位模式	指令功能	支持语言		
				LD	FBD	IL
RESH			IO 更新	√	√	√
HHSC			高速计数器	√	√	√
HCWR			写高速计数器	√	√	√
SPD			速度侦测	√	√	√
PWM			脉宽调制	√	√	√
PLSY		D.PLSY	脉冲输出	√	√	√
PLSR		D.PLSR	加减速脉冲输出	√	√	√
ZRN			原点回归	√	√	√
SETZ			设置电气原点	√	√	√
PPMR			直线插补	√	√	√
CIMR			圆弧插补	√	√	√
SPLS			简单脉冲输出	√	√	√
MPTO			多段脉冲输出	√	√	√
SYNP			随动脉冲输出	√	√	√
PSTOP			停止脉冲输出	√	√	√

【高速控制指令说明】

- 1、Haiwell（海为）PLC 的高速计数器和高速脉冲输出以通道进行定义，它们与 PLC 硬件相关，需要通过“[PLC 硬件配置](#)”窗口进行配置。高速计数器通道用 HSCx 表示，每个通道使用 2 个高速脉冲输入点。高速脉冲输出通道用 PLSx 表示，每个通道使用 2 个高速脉冲输出点。
- 2、高速计数器支持：脉冲/方向、正/反脉冲、A/B 相脉冲输入模式，支持 1、2、4 倍频计数模式，参考“[HSC 高速计数器参数](#)”。
- 3、高速脉冲输出支持：单脉冲、脉冲/方向、正/反脉冲、A/B 相脉冲、同步脉冲输出，参考“[PLS 高速脉冲输出参数](#)”。
- 4、运动控制功能支持直线插补、圆弧插补、随动脉冲输出等；支持绝对地址、相对地址；支持反向间隙补偿；支持电气原点重新定义等。
- 5、高速计数器有关 SM 系统状态位如下：

系统状态位	功能说明	属性	停电保持	出厂值
SM25	HSC0 学习使能控制，0 为正常状态，1 为学习状态	R/W	否	0
SM26	HSC0 学习确认控制	R/W	否	0
SM30	HSC0 方向指示，0 为加，1 为减	R	否	0
SM31	HSC0 错误指示	R	否	0
SM33	HSC1 学习使能控制，0 为正常状态，1 为学习状态	R/W	否	0
SM34	HSC1 学习确认控制	R/W	否	0
SM38	HSC1 方向指示，0 为加，1 为减	R	否	0
SM39	HSC1 错误指示	R	否	0
SM41	HSC2 学习使能控制，0 为正常状态，1 为学习状态	R/W	否	0
SM42	HSC2 学习确认控制	R/W	否	0
SM46	HSC2 方向指示，0 为加，1 为减	R	否	0
SM47	HSC2 错误指示	R	否	0
SM49	HSC3 学习使能控制，0 为正常状态，1 为学习状态	R/W	否	0
SM50	HSC3 学习确认控制	R/W	否	0
SM54	HSC3 方向指示，0 为加，1 为减	R	否	0
SM55	HSC3 错误指示	R	否	0

SM57	HSC4 学习使能控制, 0 为正常状态, 1 为学习状态	R/W	否	0
SM58	HSC4 学习确认控制	R/W	否	0
SM62	HSC4 方向指示, 0 为加, 1 为减	R	否	0
SM63	HSC4 错误指示	R	否	0
SM65	HSC5 学习使能控制, 0 为正常状态, 1 为学习状态	R/W	否	0
SM66	HSC5 学习确认控制	R/W	否	0
SM70	HSC5 方向指示, 0 为加, 1 为减	R	否	0
SM71	HSC5 错误指示	R	否	0
SM73	HSC6 学习使能控制, 0 为正常状态, 1 为学习状态	R/W	否	0
SM74	HSC6 学习确认控制	R/W	否	0
SM78	HSC6 方向指示, 0 为加, 1 为减	R	否	0
SM79	HSC6 错误指示	R	否	0
SM81	HSC7 学习使能控制, 0 为正常状态, 1 为学习状态	R/W	否	0
SM82	HSC7 学习确认控制	R/W	否	0
SM86	HSC7 方向指示, 0 为加, 1 为减	R	否	0
SM87	HSC7 错误指示	R	否	0

6、高速计数器有关 SV 系统寄存器如下：

系统寄存器	功能说明	属性	停电保持	出厂值
SV60	HSC0 的当前段号	R	是	0
SV61	HSC0 的当前值低字	R	是	0
SV62	HSC0 的当前值高字	R	是	0
SV63	HSC0 的错误代码	R	是	0
SV801	HSC0 的频率值低字	R	是	0
SV802	HSC0 的频率值高字	R	是	0
SV64	HSC1 的当前段号	R	是	0
SV65	HSC1 的当前值低字	R	是	0
SV66	HSC1 的当前值高字	R	是	0
SV67	HSC1 的错误代码	R	是	0
SV803	HSC1 的频率值低字	R	是	0
SV804	HSC1 的频率值高字	R	是	0
SV68	HSC2 的当前段号	R	是	0
SV69	HSC2 的当前值低字	R	是	0
SV70	HSC2 的当前值高字	R	是	0
SV71	HSC2 的错误代码	R	是	0
SV805	HSC2 的频率值低字	R	是	0
SV806	HSC2 的频率值高字	R	是	0
SV72	HSC3 的当前段号	R	是	0
SV73	HSC3 的当前值低字	R	是	0
SV74	HSC3 的当前值高字	R	是	0
SV75	HSC3 的错误代码	R	是	0
SV807	HSC3 的频率值低字	R	是	0
SV808	HSC3 的频率值高字	R	是	0
SV76	HSC4 的当前段号	R	是	0
SV77	HSC4 的当前值低字	R	是	0
SV78	HSC4 的当前值高字	R	是	0
SV79	HSC4 的错误代码	R	是	0
SV809	HSC4 的频率值低字	R	是	0
SV810	HSC4 的频率值高字	R	是	0

SV80	HSC5 的当前段号	R	是	0
SV81	HSC5 的当前值低字	R	是	0
SV82	HSC5 的当前值高字	R	是	0
SV83	HSC5 的错误代码	R	是	0
SV811	HSC5 的频率值低字	R	是	0
SV812	HSC5 的频率值高字	R	是	0
SV84	HSC6 的当前段号	R	是	0
SV85	HSC6 的当前值低字	R	是	0
SV86	HSC6 的当前值高字	R	是	0
SV87	HSC6 的错误代码	R	是	0
SV813	HSC6 的频率值低字	R	是	0
SV814	HSC6 的频率值高字	R	是	0
SV88	HSC7 的当前段号	R	是	0
SV89	HSC7 的当前值低字	R	是	0
SV90	HSC7 的当前值高字	R	是	0
SV91	HSC7 的错误代码	R	是	0
SV815	HSC7 的频率值低字	R	是	0
SV816	HSC7 的频率值高字	R	是	0

7、高速脉冲输出有关 SM 系统状态位如下：

系统状态位	功能说明	属性	停电保持	出厂值
SM93	PLS0 正转脉冲禁止	R/W	是	0
SM94	PLS0 反转脉冲禁止	R/W	是	0
SM95	PLS0 刹车功能禁止	R/W	是	0
SM96	PLS0 脉冲输出指示	R	是	0
SM97	PLS0 脉冲输出方向指示，0 为正向，1 为反向	R	是	0
SM98	PLS0 错误指示	R	是	0
SM99	PLS0 定位模式 0 为相对地址 1 为绝对地址	R/W	是	0
SM100	PLS0 脉冲输出完成	R	是	0
SM109	PLS1 正转脉冲禁止	R/W	是	0
SM110	PLS1 反转脉冲禁止	R/W	是	0
SM111	PLS1 刹车功能禁止	R/W	是	0
SM112	PLS1 脉冲输出指示	R	是	0
SM113	PLS1 脉冲输出方向指示，0 为正向，1 为反向	R	是	0
SM114	PLS1 错误指示	R	是	0
SM115	PLS1 定位模式 0 为相对地址 1 为绝对地址	R/W	是	0
SM116	PLS1 脉冲输出完成	R	是	0
SM125	PLS2 正转脉冲禁止	R/W	是	0
SM126	PLS2 反转脉冲禁止	R/W	是	0
SM127	PLS2 刹车功能禁止	R/W	是	0
SM128	PLS2 脉冲输出指示	R	是	0
SM129	PLS2 脉冲输出方向指示，0 为正向，1 为反向	R	是	0
SM130	PLS2 错误指示	R	是	0
SM131	PLS2 定位模式 0 为相对地址 1 为绝对地址	R/W	是	0
SM132	PLS2 脉冲输出完成	R	是	0
SM141	PLS3 正转脉冲禁止	R/W	是	0
SM142	PLS3 反转脉冲禁止	R/W	是	0
SM143	PLS3 刹车功能禁止	R/W	是	0
SM144	PLS3 脉冲输出指示	R	是	0

SM145	PLS3 脉冲输出方向指示, 0 为正向, 1 为反向	R	是	0
SM146	PLS3 错误指示	R	是	0
SM147	PLS3 定位模式 0 为相对地址 1 为绝对地址	R/W	是	0
SM148	PLS3 脉冲输出完成	R	是	
SM157	PLS4 正转脉冲禁止	R/W	是	0
SM158	PLS4 反转脉冲禁止	R/W	是	0
SM159	PLS4 刹车功能禁止	R/W	是	0
SM160	PLS4 脉冲输出指示	R	是	0
SM161	PLS4 脉冲输出方向指示, 0 为正向, 1 为反向	R	是	0
SM162	PLS4 错误指示	R	是	0
SM163	PLS4 定位模式 0 为相对地址 1 为绝对地址	R/W	是	0
SM164	PLS4 脉冲输出完成	R	是	0
SM173	PLS5 正转脉冲禁止	R/W	是	0
SM174	PLS5 反转脉冲禁止	R/W	是	0
SM175	PLS5 刹车功能禁止	R/W	是	0
SM176	PLS5 脉冲输出指示	R	是	0
SM177	PLS5 脉冲输出方向指示, 0 为正向, 1 为反向	R	是	0
SM178	PLS5 错误指示	R	是	0
SM179	PLS5 定位模式 0 为相对地址 1 为绝对地址	R/W	是	0
SM180	PLS5 脉冲输出完成	R	是	0
SM189	PLS6 正转脉冲禁止	R/W	是	0
SM190	PLS6 反转脉冲禁止	R/W	是	0
SM191	PLS6 刹车功能禁止	R/W	是	0
SM192	PLS6 脉冲输出指示	R	是	0
SM193	PLS6 脉冲输出方向指示, 0 为正向, 1 为反向	R	是	0
SM194	PLS6 错误指示	R	是	0
SM195	PLS6 定位模式 0 为相对地址 1 为绝对地址	R/W	是	0
SM196	PLS6 脉冲输出完成	R	是	0
SM205	PLS7 正转脉冲禁止	R/W	是	0
SM206	PLS7 反转脉冲禁止	R/W	是	0
SM207	PLS7 刹车功能禁止	R/W	是	0
SM208	PLS7 脉冲输出指示	R	是	0
SM209	PLS7 脉冲输出方向指示, 0 为正向, 1 为反向	R	是	0
SM210	PLS7 错误指示	R	是	0
SM211	PLS7 定位模式 0 为相对地址 1 为绝对地址	R/W	是	0
SM212	PLS7 脉冲输出完成	R	是	0

8、高速脉冲输出有关 SV 系统寄存器如下：

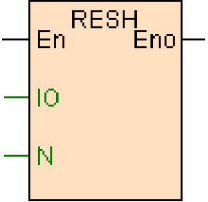
系统寄存器	功能说明	属性	停电保持	出厂值
SV92	PLS0 的当前段号	R	是	0
SV93	PLS0 的已输出脉冲数低字	R	是	0
SV94	PLS0 的已输出脉冲数高字	R	是	0
SV95	PLS0 的当前位置低字	R	是	0
SV96	PLS0 的当前位置高字	R	是	0
SV97	PLS0 的错误代码	R	是	0
SV156	PLS0 的机械原点低字	R	是	0
SV157	PLS0 的机械原点高字	R	是	0
SV158	PLS0 的反向间隙补偿脉冲数	R/W	是	0
SV98	PLS1 的当前段号	R	是	0

SV99	PLS1 的已输出脉冲数低字	R	是	0
SV100	PLS1 的已输出脉冲数高字	R	是	0
SV101	PLS1 的当前位置低字	R	是	0
SV102	PLS1 的当前位置高字	R	是	0
SV103	PLS1 的错误代码	R	是	0
SV160	PLS1 的机械原点低字	R	是	0
SV161	PLS1 的机械原点高字	R	是	0
SV162	PLS1 的反向间隙补偿脉冲数	R/W	是	0
SV104	PLS2 的当前段号	R	是	0
SV105	PLS2 的已输出脉冲数低字	R	是	0
SV106	PLS2 的已输出脉冲数高字	R	是	0
SV107	PLS2 的当前位置低字	R	是	0
SV108	PLS2 的当前位置高字	R	是	0
SV109	PLS2 的错误代码	R	是	0
SV164	PLS2 的机械原点低字	R	是	0
SV165	PLS2 的机械原点高字	R	是	0
SV166	PLS2 的反向间隙补偿脉冲数	R/W	是	0
SV110	PLS3 的当前段号	R	是	0
SV111	PLS3 的已输出脉冲数低字	R	是	0
SV112	PLS3 的已输出脉冲数高字	R	是	0
SV113	PLS3 的当前位置低字	R	是	0
SV114	PLS3 的当前位置高字	R	是	0
SV115	PLS3 的错误代码	R	是	0
SV168	PLS3 的机械原点低字	R	是	0
SV169	PLS3 的机械原点高字	R	是	0
SV170	PLS3 的反向间隙补偿脉冲数	R/W	是	0
SV116	PLS4 的当前段号	R	是	0
SV117	PLS4 的已输出脉冲数低字	R	是	0
SV118	PLS4 的已输出脉冲数高字	R	是	0
SV119	PLS4 的当前位置低字	R	是	0
SV120	PLS4 的当前位置高字	R	是	0
SV121	PLS4 的错误代码	R	是	0
SV172	PLS4 的机械原点低字	R	是	0
SV173	PLS4 的机械原点高字	R	是	0
SV174	PLS4 的反向间隙补偿脉冲数	R/W	是	0
SV122	PLS5 的当前段号	R	是	0
SV123	PLS5 的已输出脉冲数低字	R	是	0
SV124	PLS5 的已输出脉冲数高字	R	是	0
SV125	PLS5 的当前位置低字	R	是	0
SV126	PLS5 的当前位置高字	R	是	0
SV127	PLS5 的错误代码	R	是	0
SV176	PLS5 的机械原点低字	R	是	0
SV177	PLS5 的机械原点高字	R	是	0
SV178	PLS5 的反向间隙补偿脉冲数	R/W	是	0
SV128	PLS6 的当前段号	R	是	0
SV129	PLS6 的已输出脉冲数低字	R	是	0
SV130	PLS6 的已输出脉冲数高字	R	是	0
SV131	PLS6 的当前位置低字	R	是	0
SV132	PLS6 的当前位置高字	R	是	0

SV133	PLS6 的错误代码	R	是	0
SV180	PLS6 的机械原点低字	R	是	0
SV181	PLS6 的机械原点高字	R	是	0
SV182	PLS6 的反向间隙补偿脉冲数	R/W	是	0
SV134	PLS7 的当前段号	R	是	0
SV135	PLS7 的已输出脉冲数低字	R	是	0
SV136	PLS7 的已输出脉冲数高字	R	是	0
SV137	PLS7 的当前位置低字	R	是	0
SV138	PLS7 的当前位置高字	R	是	0
SV139	PLS7 的错误代码	R	是	0
SV184	PLS7 的机械原点低字	R	是	0
SV185	PLS7 的机械原点高字	R	是	0
SV186	PLS7 的反向间隙补偿脉冲数	R/W	是	0

RESH (IO 更新)

指令格式及参数说明

语言	LD	FBD	IL
指令格式			RESH En, IO, N

参数	参数定义	输入	输出	说明
En	使能	√		
IO	欲更新 IO 起始元件	√		占用 N 个连续元件
N	欲更新元件个数	√		
Eno	使能输出		√	

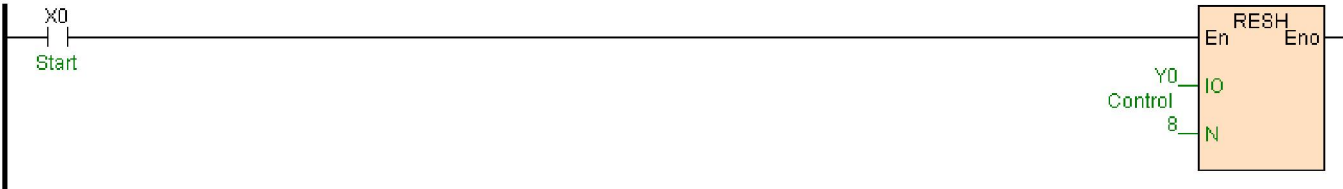
【指令功能及作用说明】

- 1、RESH 指令用于立即更新外部开关量输入、输出（X、Y）的状态，以提高对外部信号的反应速度。
- 2、En 为指令的使能端，当 En 的状态为 ON 时，对 IO 指定的连续 N 个元件（X、Y）的状态被立即更新，而无须等到程序扫描结束，即 IO 状态的更新与程序扫描周期无关。

注：本指令一般用于对实时性或精度控制要求比较高的情况，如高速控制、中断处理等。在未使用 RESH 指令的情况下，PLC 的外部输入、输出的状态要等到整个程序扫描结束后更新。

【指令示例】

//Network 1



【程序说明】

当 X0=ON，对 Y0~Y7 的输出状态作立即更新，无须等到扫描结束。

HHSC（高速计数器）

指令格式及参数说明

语言	LD	FBD	IL
指令格式	HHSC EnOut PVHVal NHFre Mod HSCx		HHSC En, PV, N, Mod, HSCx, Out

参数	参数定义	输入	输出	说明
En	使能	√		
PV	设定值起始元件	√		每段占 2 个寄存器
N	比较段数	√		1-48
Mod	比较模式	√		0-2：0 为单段比较，1 为绝对方式比较，2 为相对方式比较
HSCx	高速计数器号	√		
Out	比较结果起始元件		√	每段占 1 个位元件
HVal	高速计数器当前值		√	占用 2 个系统寄存器
HFre	当前频率值		√	占用 2 个系统寄存器

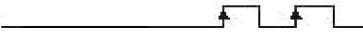
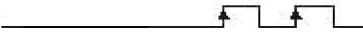
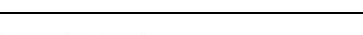
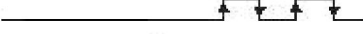
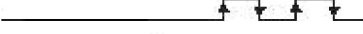
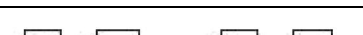
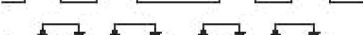
【指令功能及作用说明】

- 1、HHSC 指令用于对高速脉冲输入信号的处理，它可以同时对输入脉冲计数和测量脉冲的频率值。
- 2、高速计数器支持：脉冲/方向、正/反脉冲、A/B 相脉冲输入模式，支持 1、2、4 倍频计数模式，参考“[HSC 高速计数器参数](#)”
- 3、HHSC 指令与 [SM 系统状态位](#)、[SV 系统寄存器](#) 有关，当计数值=设定值时产生“HSCx 当前值=设定值(每段设定值都会产生)”中断，当输入的脉冲改变方向时产生“HSCx 输入方向改变”中断。
- 4、支持多段比较，支持 3 种比较方式：单段比较、绝对方式比较、相对方式比较。
- 5、HHSC 指令具有自学习功能，在学习状态下可以把当前值记录到设定值，可以连续多段学习。在进入和退出学习状态时会自动复位高速计数器 HSCx。
- 6、复位高速计数器、实时修改设定值、修改当前值、修改当前段号，使用 [HCWR](#) 指令。
- 7、当 En=ON 时，HHSC 指令执行，当 En=OFF 时，停止计数。

注：E/S 系列主机的 HHSC 指令请参见“[HHSC](#)”。

【高速计数模式与脉冲波形图】

计数模式		脉冲波形图	
模式	倍频数	加计数	减计数
0 --脉冲/方向	1	脉冲 方向	脉冲 方向
1 --脉冲/方向	2	脉冲 方向	脉冲 方向

计数模式		脉冲波形图	
模式	倍频数	加计数	减计数
2 --正转/反转	1	正转脉冲 	反转脉冲 
		反转脉冲 	正转脉冲 
3 --正转/反转	2	正转脉冲 	反转脉冲 
		反转脉冲 	正转脉冲 
4 -- A 相/B 相	1	A相脉冲 	B相脉冲 
		B相脉冲 	A相脉冲 
5 -- A 相/B 相	2	A相脉冲 	B相脉冲 
		B相脉冲 	A相脉冲 
6 -- A 相/B 相	4	A相脉冲 	B相脉冲 
		B相脉冲 	A相脉冲 

【指令示例一】

//Network 1 单段比较模式，比较段数设定为4，初始段号为1



//Network 2 把当前段号改为2，这样高速计数器以第2段的设定值进行比较



//Network 3 修改当前段的设定值



//Network 4 复位高速计数器HSC0



【程序一说明】

PV 元件	值	说明
V1000V1001	200	第 1 段设定值
V1002V1003	500	第 2 段设定值
V1004V1005	1200	第 3 段设定值
V1006V1007	1500	第 4 段设定值

- 1、当 M0=ON，HHSC 指令执行，高速计数器 HSC0 设定为单段比较模式，比较段数为 4。初始段号为 1，所以 HSC0 以第 1 段的设定值=200（V1000V1001）进行比较。
- 2、当 M1=ON 时，把 HSC0 的当前段号改为 2，这样 HSC0 以第 2 段的设定值=500（V1002V1003）进行比较。
- 3、当 M2=ON 时，把 HSC0 当前段的设定值修改为 1000，即第 2 段的设定值=1000（V1002V1003）。
- 4、当 M3=ON 时，把 HSC0 复位，HSC0 的当前计数器=0，HSC0 的当前段号=1

【指令示例二】

//Network 1 绝对方式比较，比较段数设定为4，初始段号为1

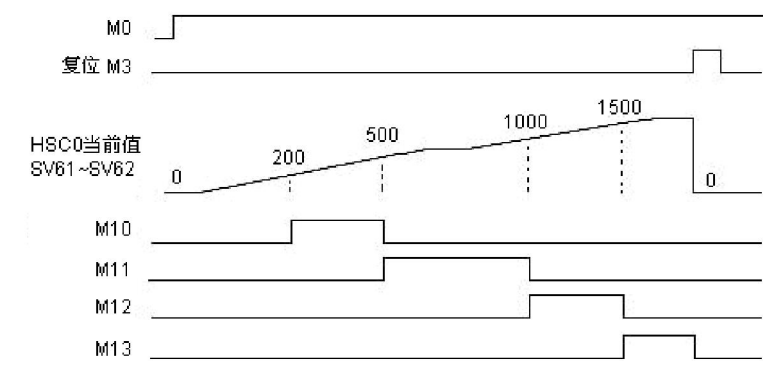


//Network 2 复位高速计数器HSC0



【程序二示意图】

PV 元件	值	说明
V1000V1001	200	第 1 段设定值
V1002V1003	500	第 2 段设定值
V1004V1005	1000	第 3 段设定值
V1006V1007	1500	第 4 段设定值



【程序二说明】

- 1、当 M0=ON，HHSC 指令执行，高速计数器 HSC0 设定为绝对方式比较，比较段数为 4。
- 2、当 M3=ON 时，把 HSC0 复位，HSC0 的当前计数器=0，HSC0 的当前段号=1

【指令示例三】

//Network 1 相对方式比较，比较段数设定为4，初始段号为1

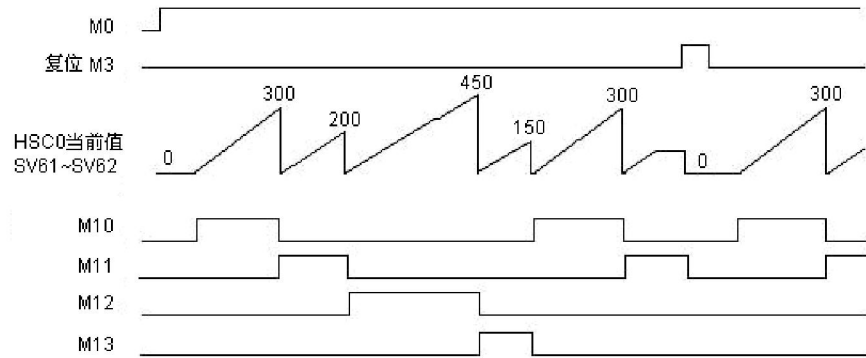


//Network 2 复位高速计数器HSC0



【程序三示意图】

PV 元件	值	说明
V1000V1001	300	第 1 段设定值
V1002V1003	200	第 2 段设定值
V1004V1005	450	第 3 段设定值
V1006V1007	150	第 4 段设定值



【程序三说明】

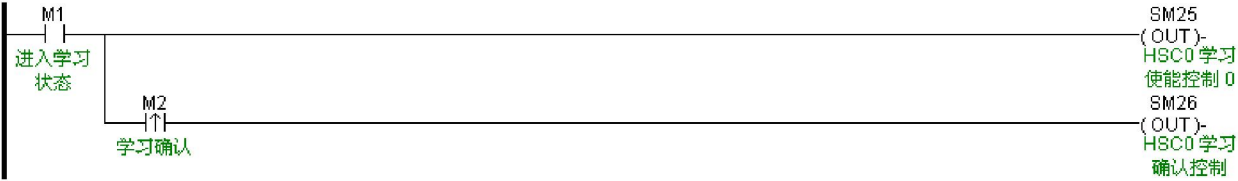
- 1、当 M0=ON，HHSC 指令执行，高速计数器 HSC0 设定为相对方式比较，比较段数为 4。
- 2、当 M3=ON 时，把 HSC0 复位，HSC0 的当前计数器=0，HSC0 的当前段号=1

【指令示例四】

//Network 1 单段比较模式，比较段数设定为4，初始段号为1



//Network 2 M1=ON进入学习状态，M1=OFF返回正常计数状态



【程序四说明】

- 1、当 M0=ON，HHSC 指令执行，高速计数器 HSC0 设定为单段比较模式，比较段数为 4。初始段号为 1。
- 2、当 M1=ON 时，SM25=ON，HSC0 进入学习状态，进入时自动复位 HSC0。
- 3、当 M2=ON，SM26=ON，HSC0 学习确认即把 HSC0 当前值记录到当前段的设定值，段号自动加 1 到下一段，如果段号大于段数（本示例 N=4）则段号=1。每 SM26=ON 一次就记录一段设定值。
- 4、在学习结束后，复位 M1=OFF，SM25=OFF，HSC0 退出学习状态，退出时自动复位 HSC0。

HCWR（写高速计数器）

指令格式及参数说明

语言	LD	FBD	IL
指令格式			HCWR En, Val, Kind, HSCx

参数	参数定义	输入	输出	说明
En	使能	√		
Val	欲写入的值	√		
Kind	写入值类型	√		0-3：0-写入当前段号，1-写入当前段设定值，2-写入高速计数器当前值，3- 复位高速计数器
HSCx	高速计数器号	√		
Eno	使能输出		√	

【指令功能及作用说明】

- 1、HCWR 指令用于辅助控制高速计数器，它配合 HHSC 指令使用，完成复位高速计数器、实时修改设定值、修改高速计数器当前值、修改当前段号。
- 2、如果写入的段号超过 HSCx 设定的段数（HHSC 指令端子 N 定义）范围，HSCx 报 1 号参数错误。
- 3、HCWR 要用边沿执行。

【指令示例】

//Network 4 复位高速计数器HSC0



【程序说明】

当 M3=ON，复位高速计数器 HSC0，HSC0 当前值=0，当前段号=1。

SPD（速度侦测）

指令格式及参数说明

语言	LD	FBD	IL
指令格式			SPD En, TnP, X, HFre

参数	参数定义	输入	输出	说明
En	使能	√		
TnP	侦测时间或脉冲数	√		TnP>0 为侦测时间（单位 0.1ms），TnP<0 为侦测脉冲数
X	脉冲输入	√		
Eno	使能输出		√	
HFre	频率值		√	占用 2 个连续元件

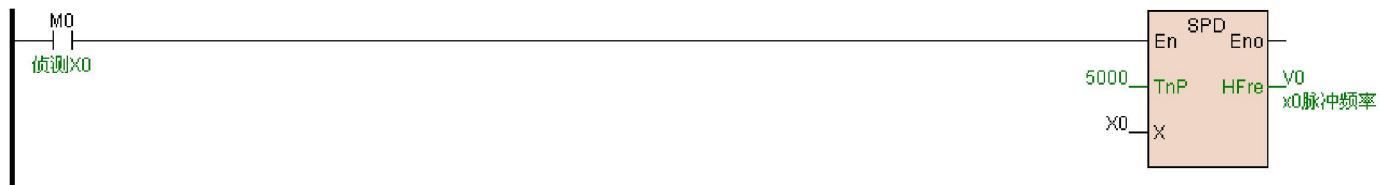
【指令功能及作用说明】

- 1、SPD 指令侦测主机高速脉冲输入点 Xn 输入脉冲的频率值。它不使用 HSCx 高速脉冲输入通道，1 路高速脉冲输入通道有 2 个高速脉冲输入点。这样有 8 个脉冲输入通道的主机可以实现 16 路高速输入脉冲频率的侦测。
- 2、SPD 指令支持以时间或者脉冲数方式测量频率，当 TnP>0 则以时间侦测方式测量频率（时间单位 0.1ms），如果 TnP<0 则以脉冲个数方式测量频率。TnP=0 则 HFre=0。
- 3、为确保频率测量的精度，当输入脉冲的频率高于 19KHz 时使用时间侦测方式（建议侦测时间大于 500ms，TnP>5000），当输入脉冲频率低于 19KHz 时使用侦测脉冲个数方式。

注：E/S 系列主机的 SPD 指令参见“[SPD](#)”。

【指令示例】

//Network 1 侦测X0输入脉冲的频率，侦测时间500ms



//Network 2 侦测X7输入脉冲的频率，侦测600个脉冲



【程序说明】

- 1、当 M0=ON，侦测 X0 脉冲输入的频率值，侦测时间 5000*0.1ms=500ms，即 500ms 更新一次。
- 2、当 M1=ON 时，侦测 X7 脉冲输入的频率值，侦测脉冲个数为 600，即每侦测到 600 个脉冲更新一次。

PWM（脉宽调制）

指令格式及参数说明

语言	LD	FBD	IL
指令格式			PWM En, PulR, PulF, Out

参数	参数定义	输入	输出	说明
En	使能	√		
PulR	脉冲占空比	√		单位 0.1%，范围 0~1000
PulF	脉冲输出频率	√		占用 2 个连续元件
Out	脉冲输出		√	

【指令功能及作用说明】

- 1、PWM 指令通过主机高速脉冲输出点 Yn 输出指定频率、占空比的高速脉冲。它不使用 PLSx 高速脉冲输出通道，1 路高速脉冲输出通道有 2 个高速脉冲输出点。这样有 8 个脉冲输出通道的主机可以实现 16 路高速脉冲输出。
- 2、当 PulF≤0 不输出脉冲，否则当 PulF<最小频率（10Hz）则使用最小频率，当 PulF>最大频率则使用最大频率。
- 3、当 PulF>0，若 PulR>0 并且 PulR<1000 时，Out 输出占空比为 PulR、频率为 PulT 的脉冲，若 PulR=0 则 Out 输出低电平，若 PulR≥1000 则 Out 输出高电平信号。
- 4、PulR、PulF 的值可实时修改。

注：E/S 系列主机的 PWM 指令参见“[PWM](#)”。

【指令示例】



【程序说明】

- 1、当 M0=ON，从 Y3 输出占空比为 30%、频率为 50KHz 的脉冲信号。
- 2、当 M0=OFF，脉冲输出停止。

PLSY、D.PLSY（脉冲输出）

指令格式及参数说明

语言	LD	FBD	IL
16、32 位 指令格式			PLSY En, PulF, PulN, PLSx D.PLSY En, PulF, PulN, PLSx

参数	参数定义	输入	输出	说明
En	使能	√		
PulF	脉冲输出频率	√		
PulN	脉冲输出个数	√		
PLSx	脉冲输出通道号	√		
Eno	使能输出		√	
Pn	已输出脉冲数		√	占用 2 个系统寄存器
Pos	当前位置		√	占用 2 个系统寄存器

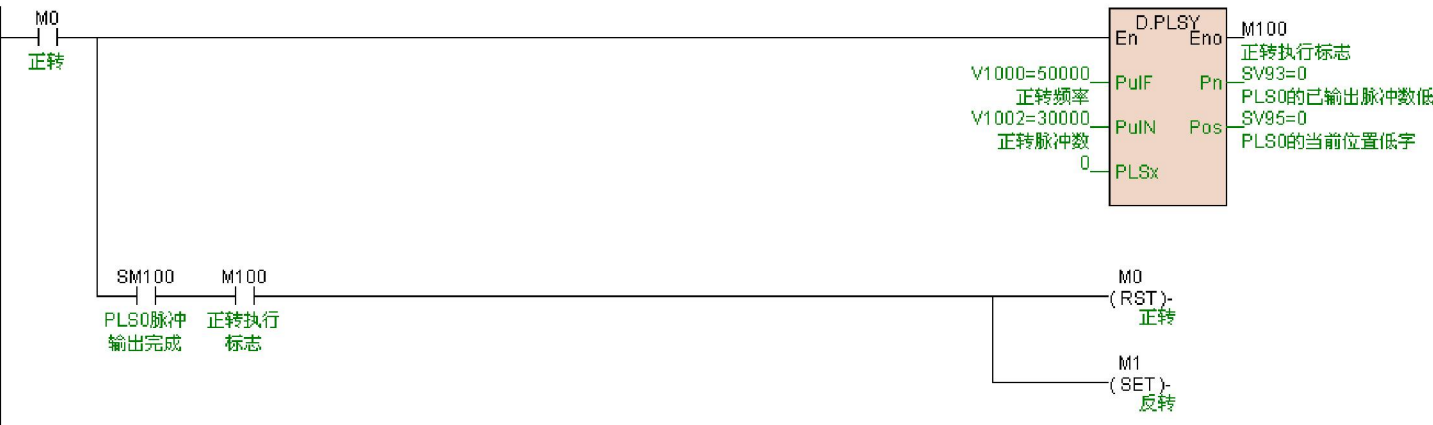
【指令功能及作用说明】

- 1、PLSY 为单段脉冲输出指令。
- 2、PulN 为输出脉冲数，PulN>0 表示输出正向脉冲，PulN<0 表示输出反向脉冲，PulN=0 表示不计脉冲个数连续输出脉冲。当 PulN=0 时不影响当前位置。
- 3、PulF 为脉冲输出频率。在 PulN=0 时（不计脉冲个数连续输出脉冲），如果 PulF=0 则不输出，PulF>0 表示输出正向脉冲，PulF<0 表示输出反向脉冲。在 PulN≠0 时，如果 PulF≤0 则不输出，PulF 大于 0 却小于最小频率（10Hz）则按照最小频率输出，PulF 大于最大频率则按照最大频率输出
- 4、PLSY 指令与 [SM 系统状态位](#)、[SV 系统寄存器](#) 有关，并且会产生脉冲输出中断，指令执行开始输出脉冲时产生“PLSx 开始输出脉冲”中断，指令执行完成停止输出脉冲时产生“PLSx 输出脉冲结束”中断。
- 5、En=ON 时指令执行，Eno=ON；当脉冲输出过程中 En 变为 OFF，则停止脉冲输出，Eno=OFF。
- 6、PLSY 指令没有条数限制，可以与其他脉冲输出指令并存，但每个脉冲输出通道同时只能执行一条指令。

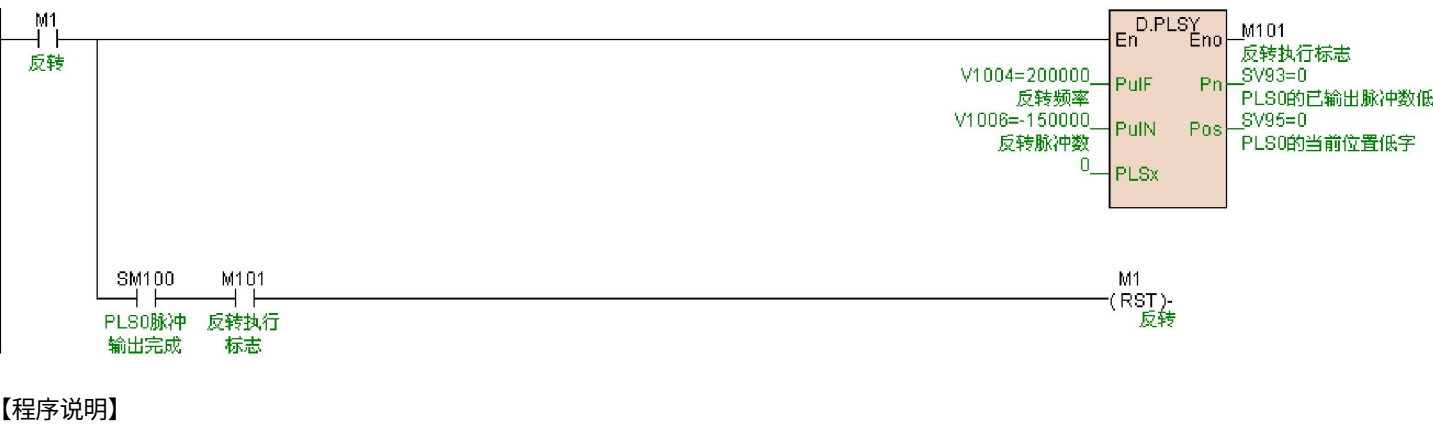
7、在指令执行脉冲输出过程中，PulF 脉冲输出频率可以被实时修改，PulN 不能被实时修改。

【指令示例】

//Network 1 正转控制



//Network 2 反转控制



【程序说明】

- 1、当 M0=ON，脉冲输出通道 PLS0 以 50KHz 的频率输出 30000 个正向脉冲，M100=ON，输出完成 SM100=ON，复位 M0，置位 M1。
- 2、当 M1=ON 时，脉冲输出通道 PLS0 以 200KHz 的频率输出 150000 个反向脉冲，M101=ON，输出完成 SM100=ON，复位 M1。

PLSR、D.PLSR（加减速脉冲输出）

指令格式及参数说明

语言	LD	FBD	IL
16、32 位 指令格式	PLSR EnEno MaxF Pn PuIN Pos Tms PLSx	D.PLSR EnEno MaxF Pn PuIN Pos Tms PLSx	PLSR En, MaxF, PuIN, Tms, PLSx D.PLSR En, MaxF, PuIN, Tms, PLSx

参数	参数定义	输入	输出	说明
En	使能	√		
MaxF	脉冲输出最大频率	√		

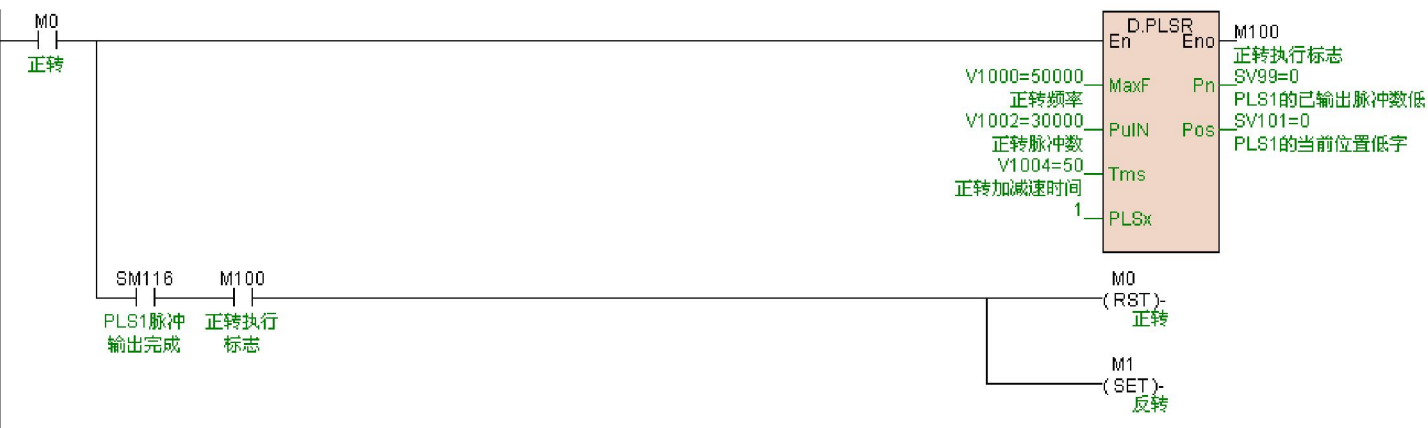
参数	参数定义	输入	输出	说明
PuIN	脉冲输出总数	√		
Tms	加减速时间	√		(5~5000ms)
PLSx	脉冲输出通道号	√		
Eno	使能输出		√	
Pn	已输出脉冲数		√	占用 2 个系统寄存器
Pos	当前位置		√	占用 2 个系统寄存器

【指令功能及作用说明】

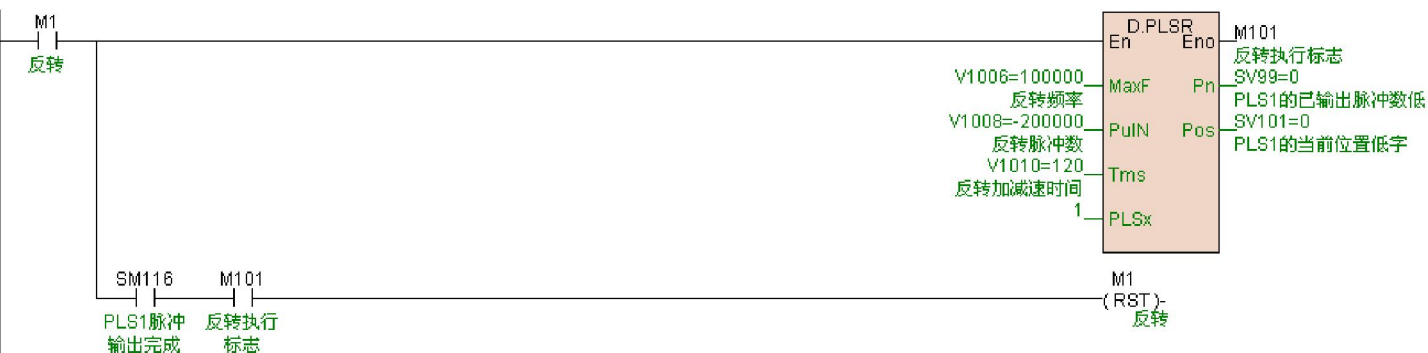
- 1、PLSR 为带加减速单段脉冲输出指令。Tms 为加减速时间，当 Tms=0 表示不进行加减速，否则当 Tms<最小加减速时间则使用最小加减速时间，当 Tms>最大加减速时间则使用最大加减速时间。
- 2、MaxF 为脉冲输出最大频率，MaxF 必须大于 0，MaxF 大于 0 却小于最小频率（10Hz）则按照最小频率输出，MaxF 大于最大频率则按照最大频率输出，MaxF≤0 则不输出，报 3 号参数错误。
- 3、PuIN 为输出脉冲数，大于 0 表示输出正向脉冲，小于 0 表示输出反向脉冲，等于 0 则不输出，报 3 号参数错误。
- 4、PLSR 指令与 [SM 系统状态位](#)、[SV 系统寄存器](#) 有关，并且会产生脉冲输出中断，指令执行开始输出脉冲时产生“PLSx 开始输出脉冲”中断，指令执行完成停止输出脉冲时产生“PLSx 输出脉冲结束”中断。
- 5、En=ON 时指令执行,Eno=ON ;当脉冲输出过程中 En 变为 OFF,指令根据 Tms 减速刹车停止 (如果没有设置禁止刹车),Eno=OFF。
- 6、PLSR 指令没有条数限制，可以与其他脉冲输出指令并存，但每个脉冲输出通道同时只能执行一条指令。
- 7、在指令执行脉冲输出过程中，MaxF、PuIN 不能被修改（修改不会实时生效，需要重新执行指令）。

【指令示例】

//Network 1 正转控制



//Network 2 反转控制



//Network 3 停止输出脉冲



【程序说明】

- 1、当 M0=ON，脉冲输出通道 PLS1 以 50KHz 的频率输出 30000 个正向脉冲，加减速时间 50ms，M100=ON，输出完成 SM116=ON，复位 M0，置位 M1。
- 2、当 M1=ON 时，脉冲输出通道 PLS1 以 100KHz 的频率输出 200000 个反向脉冲，加减速时间 120ms，M101=ON，输出完成 SM116=ON，复位 M1。
- 3、当 M55=ON 时，脉冲输出通道 PLS1 停止输出脉冲。

ZRN（原点回归）

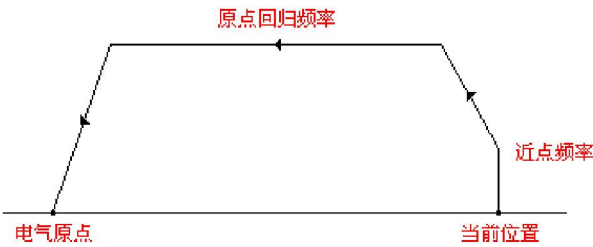
指令格式及参数说明

语言	LD	FBD	IL
指令格式			ZRN En, DOG, PulF, DPulF, Tms, PLSx, End

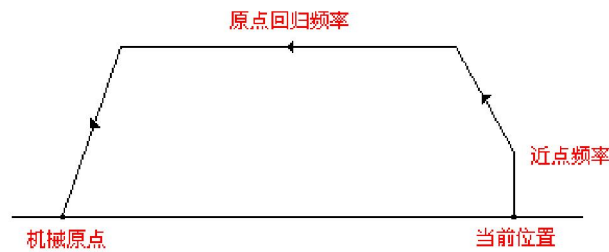
参数	参数定义	输入	输出	说明
En	使能	√		
DOG	近点信号	√		
PulF	原点回归频率	√		占用 2 个连续元件
DPulF	近点频率	√		占用 2 个连续元件
Tms	加减速时间	√		(5~5000ms)
PLSx	脉冲输出通道号	√		
Eno	使能输出		√	
End	原点回归完成		√	
Pos	当前位置		√	占用 2 个系统寄存器

【指令功能及作用说明】

- 1、当 ZRN 执行时，会以近点频率做为初始频率，加速至原点回归频率开始移动，接近原点时减速至近点频率。
- 2、DOG=0，表示回电气原点。ZRN 根据 PLSx 通道的当前位置和电气原点位置决定回归动作方向，当前位置>电气原点，以减脉冲方向运行，当前位置<电气原点，以加脉冲方向运行，自动回电气原点。回电气原点后，将当前位置 设为 0。如果当前位置=电气原点，则指令不动作。

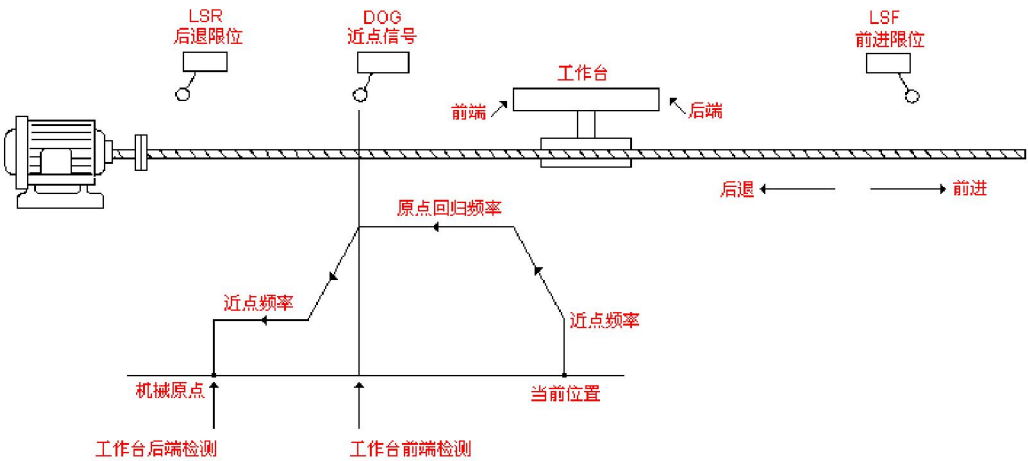


3、DOG=1，表示回机械原点，无近点信号。ZRN 根据 PLSx 通道的当前位置和机械原点位置决定回归动作方向，当前位置>机械原点，以减脉冲方向运行，当前位置<机械原点，以加脉冲方向运行，自动回机械原点。回机械原点后，设置当前位置=机械原点。如果当前位置=机械原点，则指令不动作。

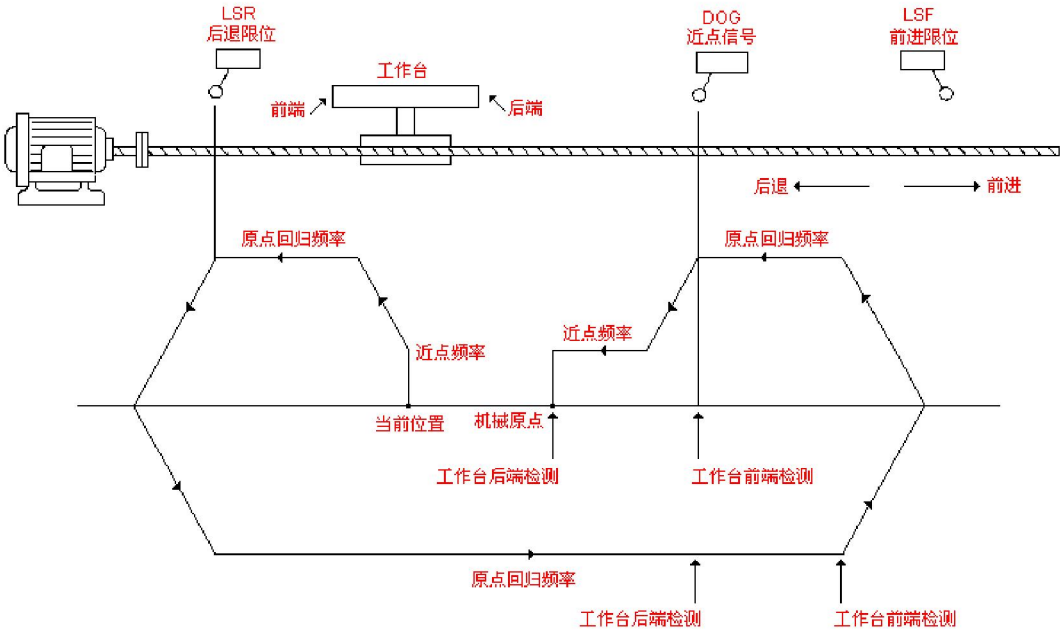


4、DOG 端子指定 X 外部输入点，表示使用近点信号回机械原点。当使用 DOG 近点信号时，ZRN 指令带 DOG 搜索功能，ZRN 固定以减脉冲方向运行。

a)、当前位置在 DOG 的外部时，ZRN 指令从近点频率加速到原点回归频率，当近点信号(DOG)由 OFF→ON 时，开始减速至近点频率（爬行速度），当近点信号(DOG)由 ON→OFF 时，脉冲输出停止，原点回归完成。回机械原点后，设置当前位置=机械原点。



b)、当前位置在 DOG 的内部或者在 DOG 之上时，ZRN 指令从近点频率加速到原点回归频率，直到碰到 LSR 后退限位开关时减速到 0，然后反向前进；当近点信号(DOG)由 ON→OFF 时，减速到 0 然后再次反向；在当近点信号(DOG)由 OFF→ON 时，开始减速至近点频率（爬行速度），当近点信号(DOG)由 ON→OFF 时，脉冲输出停止，原点回归完成。回机械原点后，设置当前位置=机械原点。



- 5、当设计近点信号(DQG)时，请考虑 DQG 前后端有足够的长，这样在测到 DQG 监测信号由 OFF 变为 ON 后有充分的 ON 时间减速到 近点频率，否则会导致停止位置偏移。
- 6、近点信号(DQG)应该接到 PLC 主机本体的 X 输入点，否则会受扫描周期影响导致位置偏差。
- 7、ZRN 指令与 [SM 系统状态位](#)、[SV 系统寄存器](#)有关。
- 8、当 PulF=0，报 3 号参数错误不输出脉冲，否则当 PulF<最小频率（10Hz）则使用最小频率，当 PulF>最大频率则使用最大频率。
- 9、当 DPulF<最小频率（10Hz）或者 DPulF>最大频率，即近点频率超范围时使用最小频率。
- 10、当 Tms<最小加减速时间则使用最小加减速时间，当 Tms>最大加减速时间则使用最大加减速时间。
- 11、在此指令启动后，其所有参数将不接受修改，直到指令关闭。
- 12、当 En=OFF 时，所以输出动作都会立即停止。

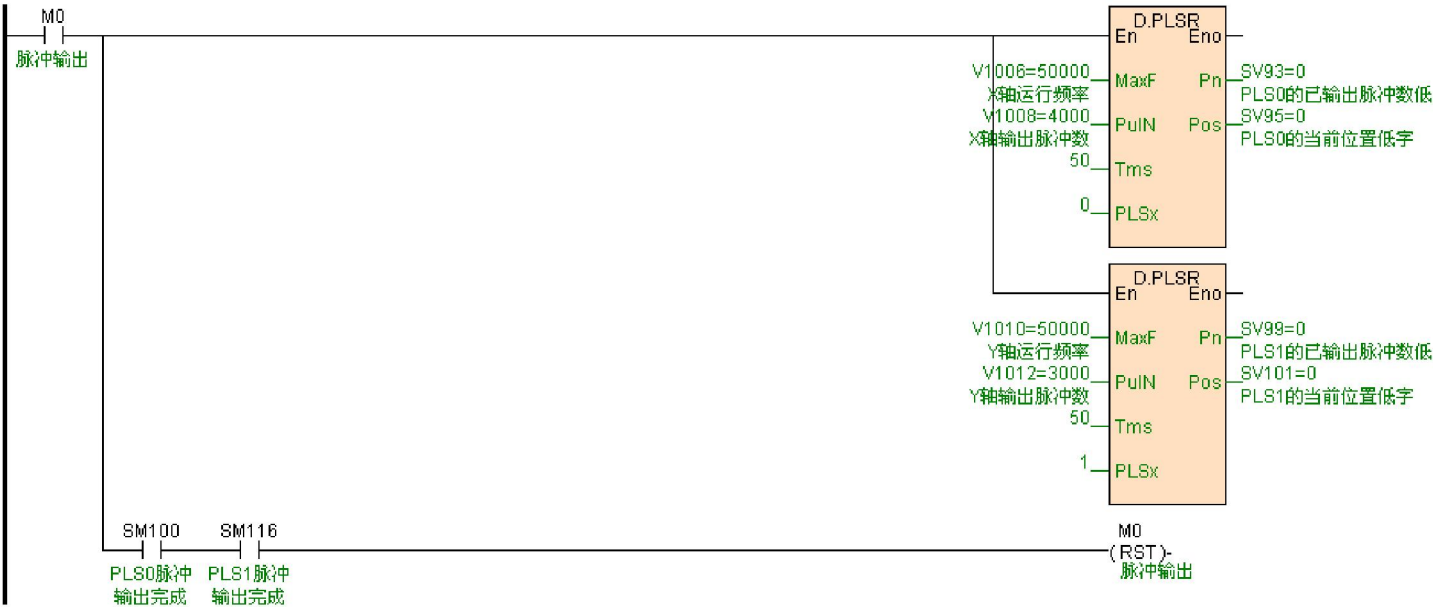
注：E/S 系列主机不支持 ZRN 指令。

【指令示例】

//Network 1 假如X轴前进限位X0、后退限位X1、近点X2；Y轴前进限位X3、后退限位X4、近点X5。限位信号赋值给系统状态位



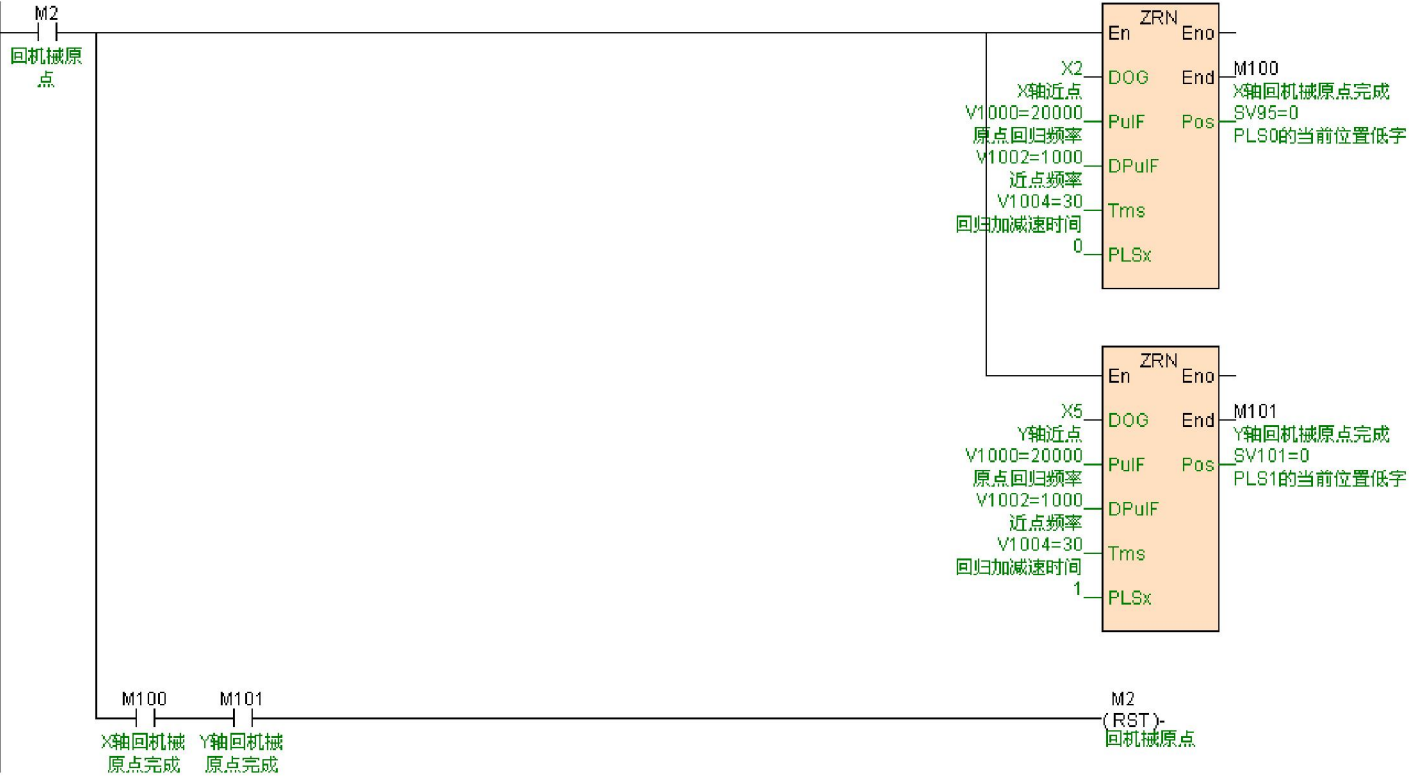
//Network 2 PLS0正转4000，PLS1正转3000，当前位置为(4000,3000)



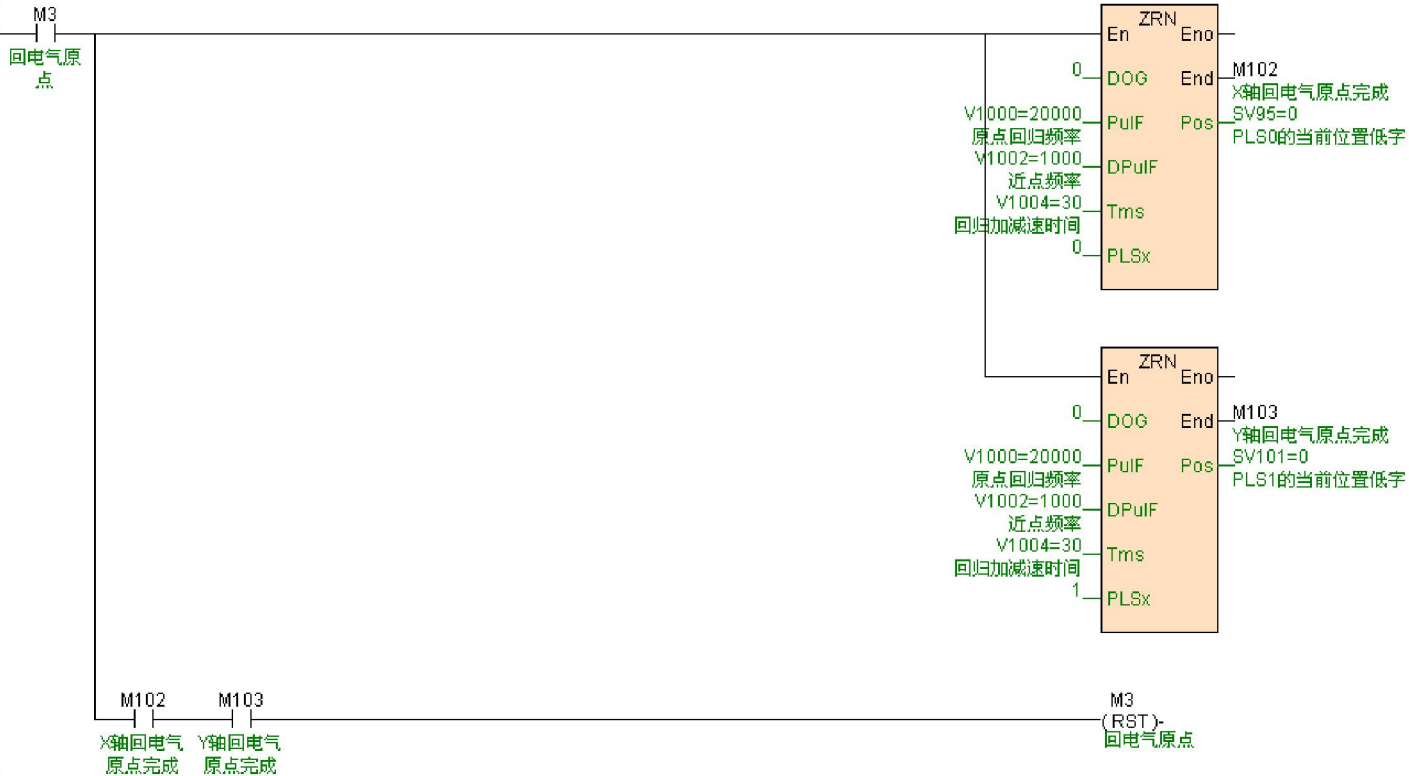
//Network 3 把当前位置(4000,3000)设为电气原点，则当前位置为(0,0)



//Network 4 回机械原点，当前位置为(-4000,-3000)



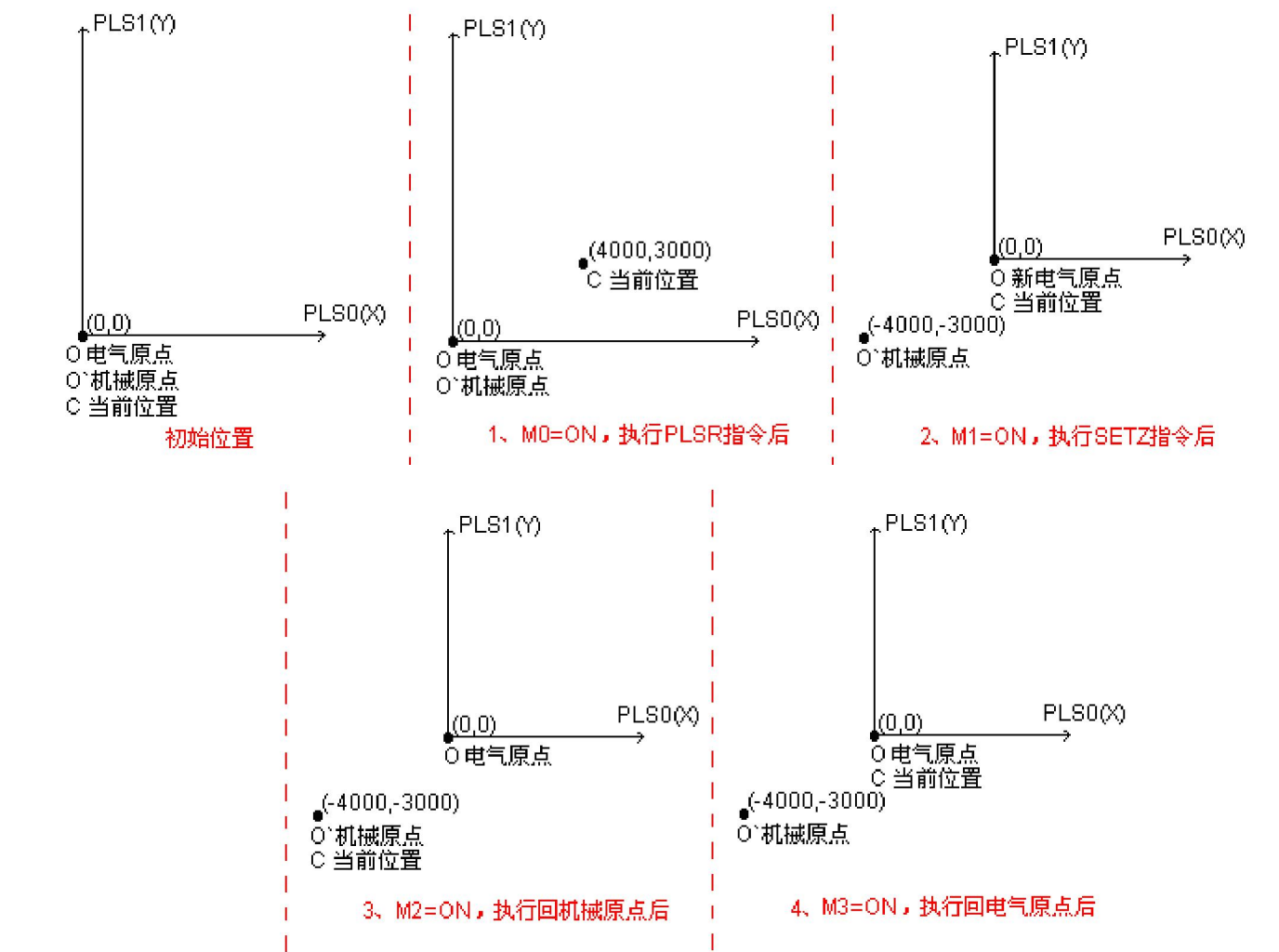
//Network 5 回电气原点，当前位置为(0,0)



【程序说明】

- 1、当 M0=ON，PLS0 以 50KHz 的频率输出 4000 个正向脉冲，输出完成 SM100=ON，PLS1 以 50KHz 的频率输出 3000 个正向脉冲，输出完成 SM116=ON。复位 M0
- 2、当 M1=ON 时，把（4000,3000）设为电气原点，机械原点变为（-4000,-3000）。
- 3、当 M2=ON 时，PLS0 和 PLS1 以 20KHz 的 回原点频率、1KHz 的近点频率回机械原点，当前位置变为（-4000,-3000）。

4、当 M3=ON 时，PLS0 和 PLS1 以 20KHz 的回原点频率、1KHz 的近点频率回电气原点，当前位置变为 (0,0)。



SETZ（设置电气原点）

指令格式及参数说明

语言	LD	FBD	IL
指令格式			SETZ En, PLSx

参数	参数定义	输入	输出	说明
En	使能	√		
PLSx	脉冲输出通道号	√		
Eno	使能输出		√	

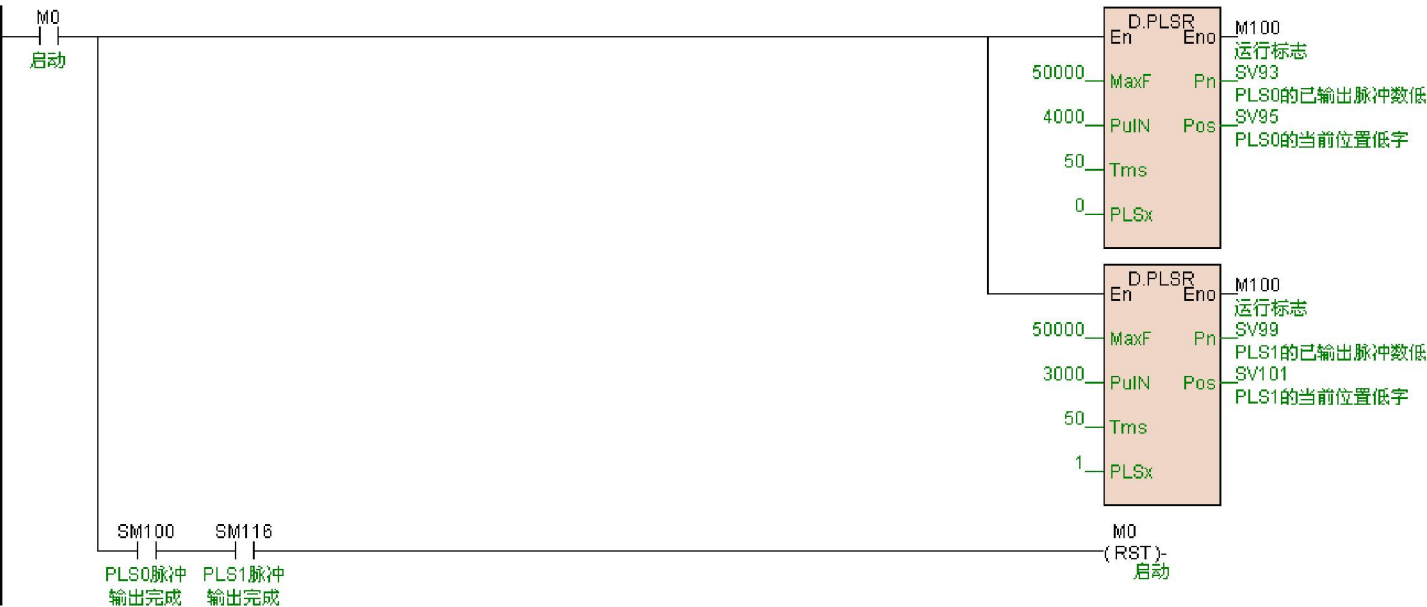
【指令功能及作用说明】

系统初始时，电气原点就是机械原点。在执行 SETZ 指令后，将当前位置设置为电气原点，当前位置清零。

注：E/S 系列主机不支持 SETZ 指令。

【指令示例】

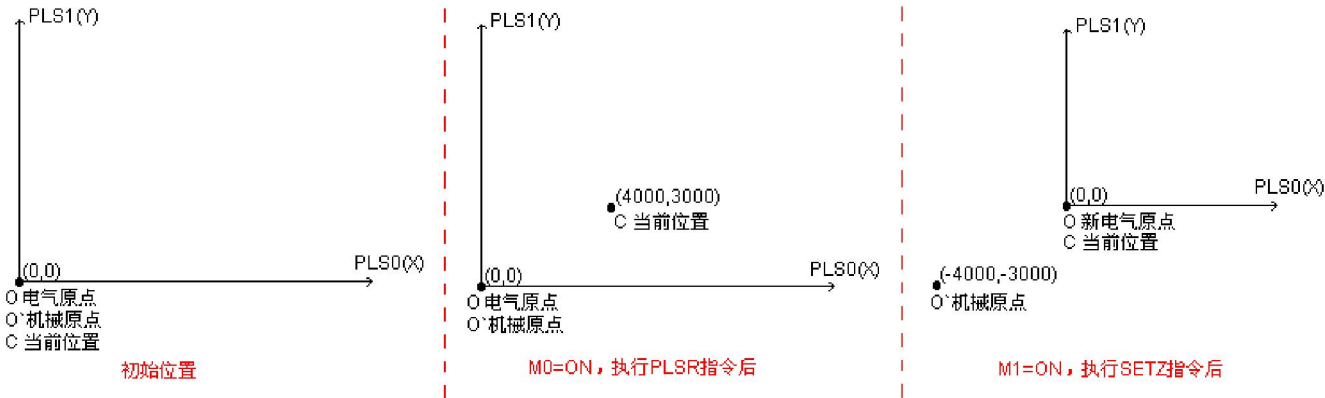
//Network 1 PLS0正转4000，PLS1正转3000



//Network 2 把（4000,3000）设为电气原点



【程序示意图】



【程序说明】

- 当 M0=ON，PLS0 以 50KHz 的频率输出 4000 个 正向脉冲，输出完成 SM100=ON，PLS1 以 50KHz 的频率输出 3000 个正向脉冲，输出完成 SM116=ON。复位 M0
- 当 M1=ON 时，把（4000,3000）设为电气原点，机械原点变为（-4000,-3000）。

PPMR（直线插补）

指令格式及参数说明

语言	LD	FBD	IL
指令格式	PPMR EnEno XTposEnd YTpos PulF Tms XPLSx YPLSx		PPMR En, XTpos, YTpos, PulF, Tms, XPLSx, YPLSx, End

参数	参数定义	输入	输出	说明
En	使能	√		
XTpos	X 目标位置	√		占用 2 个连续元件
YTpos	Y 目标位置	√		占用 2 个连续元件
PulF	脉冲输出频率	√		占用 2 个连续元件
Tms	加减速时间	√		(5~5000ms)
XPLSx	X 脉冲输出通道	√		
YPLSx	Y 脉冲输出通道	√		
Eno	使能输出		√	
End	直线插补完成		√	

【指令功能及作用说明】

- 1、PPMR 指令以当前位置为起点，以（XTpos,YTpos）为终点，做直线插补输出。
- 2、起点和终点不能是同一个点，否则报 3 号参数错误。
- 3、PPMR 指令与 [SM 系统状态位](#)、[SV 系统寄存器](#) 有关。
- 4、当 PulF=0，报 3 号参数错误不输出脉冲，否则当 PulF<最小频率（10Hz）则使用最小频率，当 PulF>最大频率则使用最大频率。
- 5、当 Tms<最小加减速时间则使用最小加减速时间，当 Tms>最大加减速时间则使用最大加减速时间。
- 6、在此指令启动后，其所有参数将不接受修改，直到指令关闭。
- 7、当 En=OFF 时，所以输出动作都会立即停止。

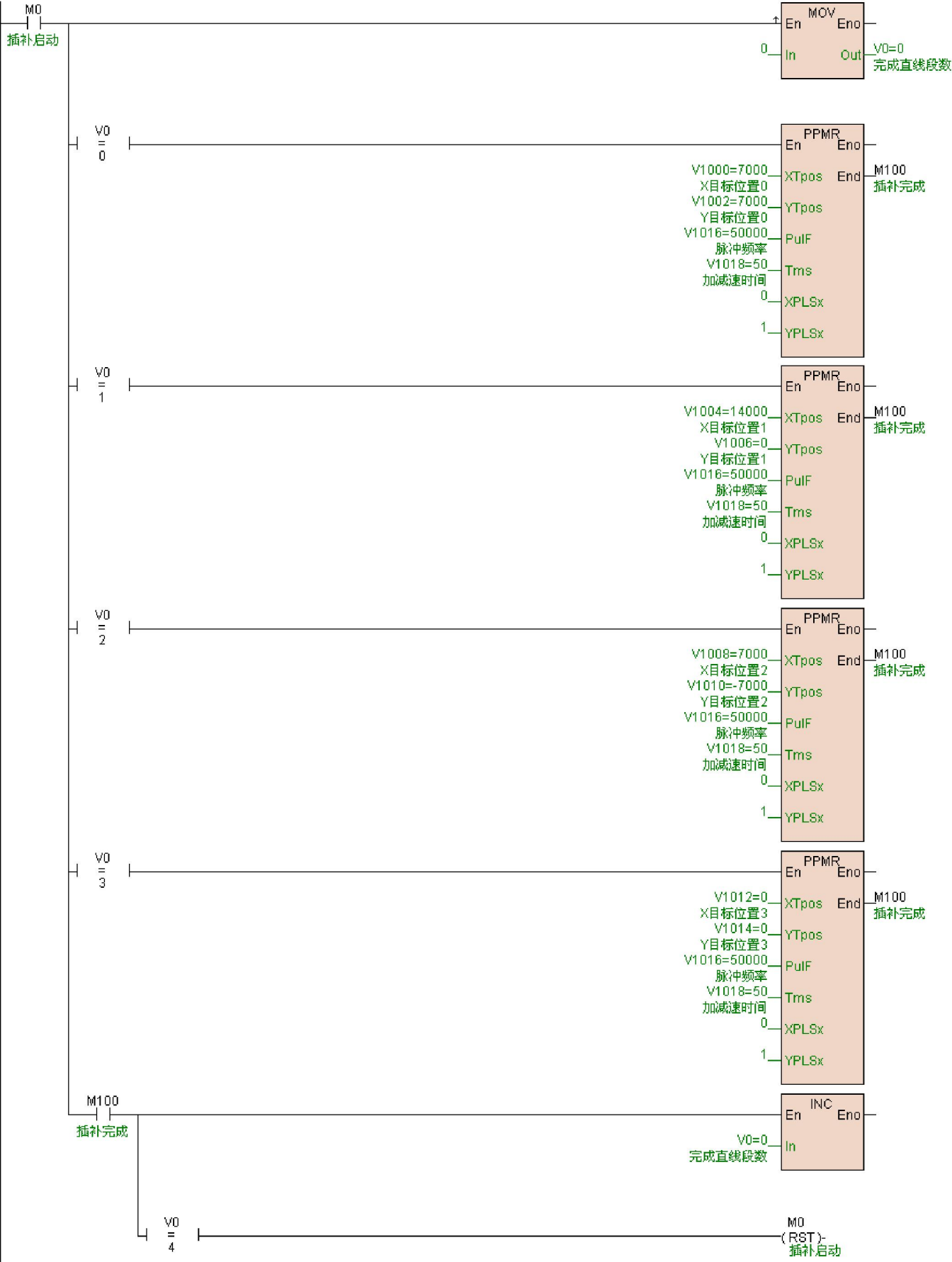
注：E/S 系列主机不支持 PPMR 指令。

【指令示例】

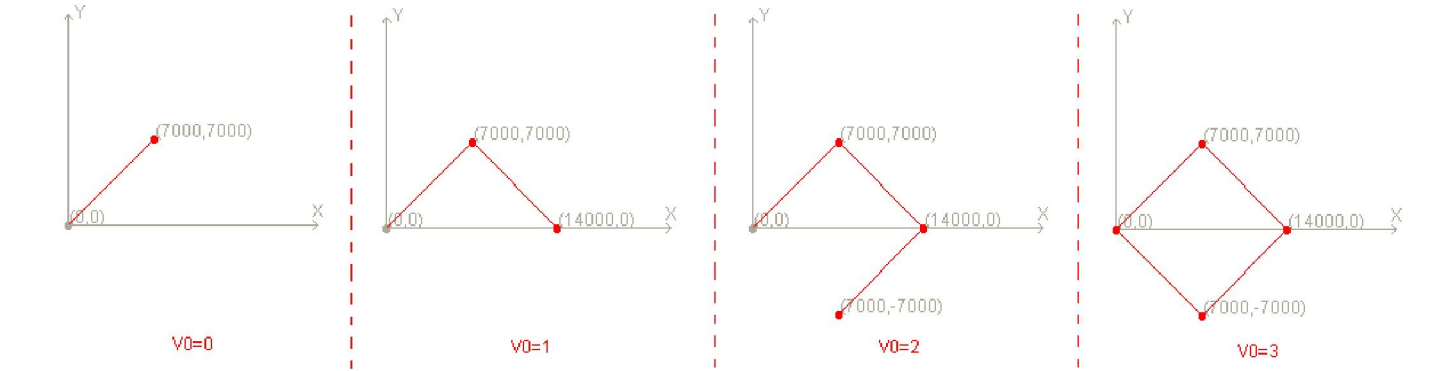
//Network 1 设置PLS0、PLS1定位模式为绝对地址模式



//Network 2 X轴为PLS0，Y轴为PLS1，做直线插补控制



【程序示意图】



【程序说明】

1、通过初始寄存器值表“直线插补参数”设置 4 段直线的坐标值。

寄存器元件	值	说明
V1000V1001	7000	X 目标位置 0
V1002V1003	7000	Y 目标位置 0
V1004V1005	14000	X 目标位置 1
V1006V1007	0	Y 目标位置 1
V1008V1009	7000	X 目标位置 2
V1010V1011	-7000	Y 目标位置 2
V1012V1013	0	X 目标位置 3
V1014V1015	0	Y 目标位置 3
V1016V1017	50000	脉冲频率
V1018	50	加减速时间

- 2、程序首次扫描时 SM2=ON，设置 PLS0、PLS1 定位模式为绝对地址模式（SM99=ON，SM115=ON）。
- 3、X 轴为 PLS0，Y 轴为 PLS1，当 M0=ON，V0=0，执行从当前位置（0,0）到（7000,7000）直线插补。
- 4、1 段插补完成 M100=ON，V0=1，执行从（7000,7000）到（14000,0）直线插补。
- 5、2 段插补完成 M100=ON，V0=2，执行从（14000,0）到（7000,-7000）直线插补。
- 6、3 段插补完成 M100=ON，V0=3，执行从（7000,-7000）到（0,0）直线插补。
- 7、4 段插补完成 M100=ON，V0=4，复位 M0，M0=OFF。

CIMR（圆弧插补）

指令格式及参数说明

语言	LD	FBD	IL
----	----	-----	----

语言	LD	FBD	IL
指令格式	CIMR EnEno XTposEnd YTpos CR PulF Tms Dir XPLSx YPLSx		CIMR En, XTpos, YTpos, CR, PulF, Tms, Dir, XPLSx, YPLSx, End

参数	参数定义	输入	输出	说明
En	使能	√		
XTpos	X 目标位置	√		占用 2 个连续元件
YTpos	Y 目标位置	√		占用 2 个连续元件
CR	圆弧半径	√		占用 2 个连续元件
PulF	脉冲输出频率	√		占用 2 个连续元件
Tms	加减速时间	√		(5~5000ms)
Dir	运动方向	√		
XPLSx	X 脉冲输出通道	√		
YPLSx	Y 脉冲输出通道	√		
Eno	使能输出		√	
End	圆弧插补完成		√	

【指令功能及作用说明】

- 1、CIMR 指令以当前位置为起点，以（XTpos,YTpos）为终点，以 CR 为半径，做圆弧插补输出。
- 2、起点和终点不能是同一个点，如果半径小于起点到终点距离的一半（ $CR < L / 2$ ，L=起点到终点的距离），那么报 3 号参数错误。
- 3、 $CR > 0$ 表示劣弧（小于半圆的弧）， $CR < 0$ 表示优弧（大于半圆的弧），Dir 为运动方向（0-顺时针，1-逆时针）
- 4、CIMR 指令与 [SM 系统状态位](#)、[SV 系统寄存器](#) 有关。
- 5、当 PulF=0，报 3 号参数错误不输出脉冲，否则当 $PulF < \text{最小频率}(10\text{Hz})$ 则使用最小频率，当 $PulF > \text{最大频率}$ 则使用最大频率。
- 6、当 $Tms < \text{最小加减速时间}$ 则使用最小加减速时间，当 $Tms > \text{最大加减速时间}$ 则使用最大加减速时间。
- 7、在此指令启动后，其所有参数将不接受修改，直到指令关闭。
- 8、当 En=OFF 时，所以输出动作都会立即停止。

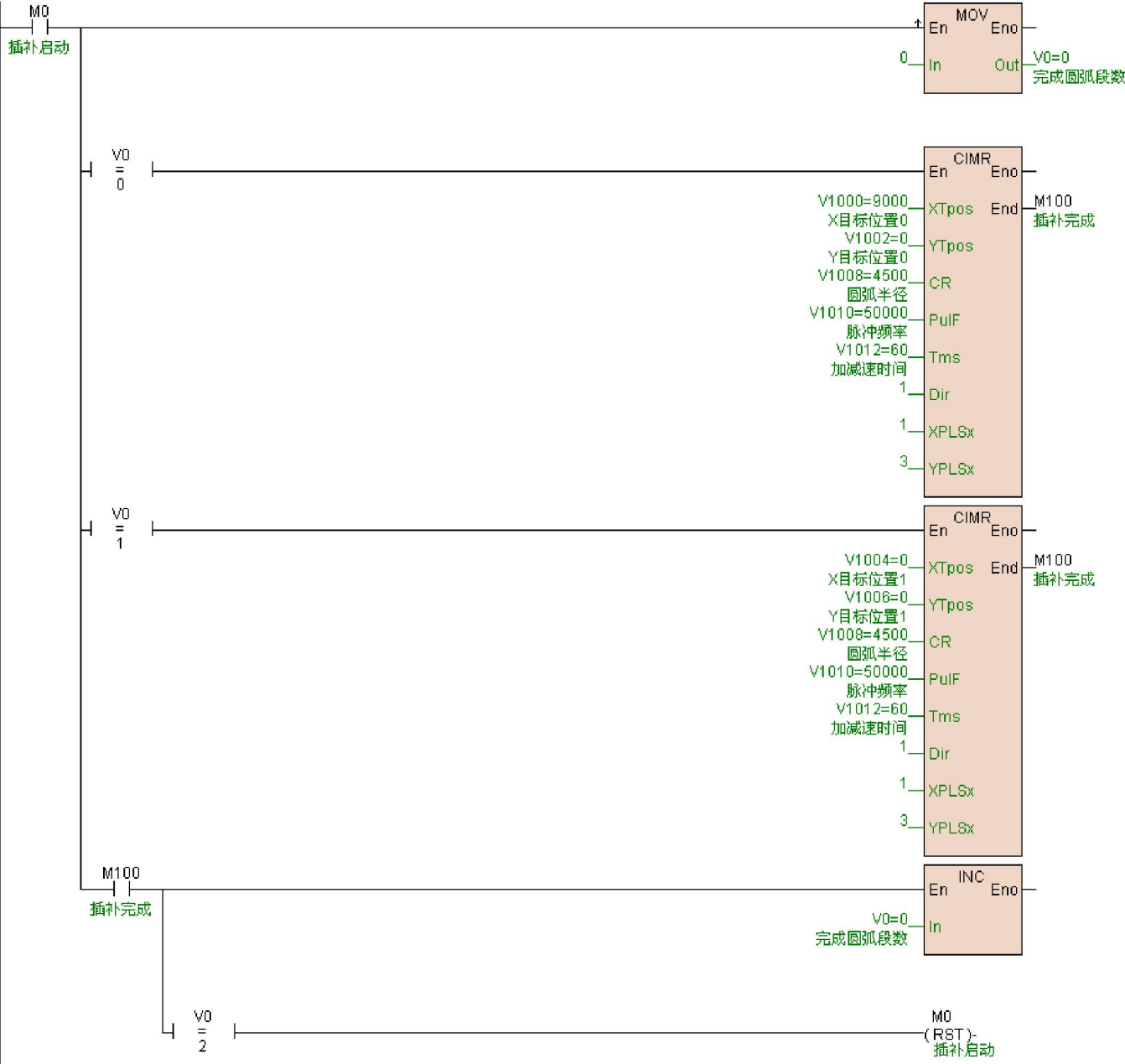
注：E/S 系列主机不支持 CIMR 指令。

【指令示例】

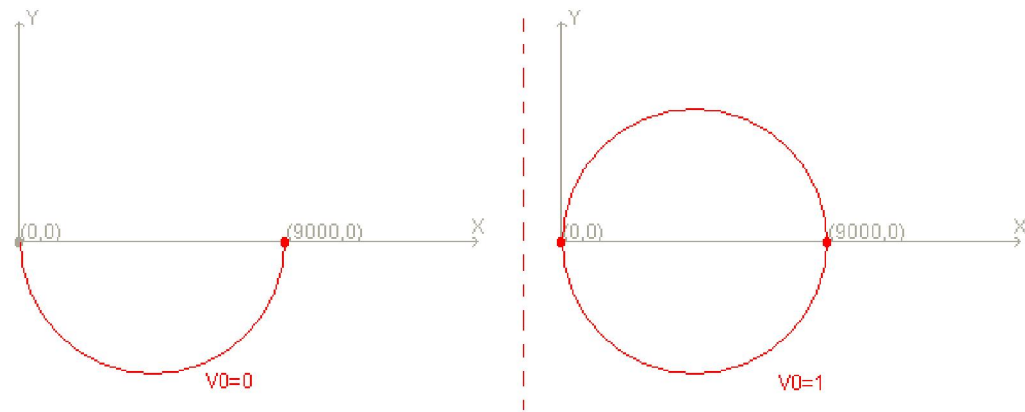
//Network 1 设置PLS1、PLS3定位模式为绝对地址模式



//Network 2 X轴为PLS1，Y轴为PLS3，做圆弧插补控制



【程序示意图】



【程序说明】

1、通过初始寄存器值表“圆弧插补参数”设置 2 段圆弧的坐标值。

寄存器元件	值	说明
V1000V1001	9000	X 目标位置 0
V1002V1003	0	Y 目标位置 0
V1004V1005	0	X 目标位置 1
V1006V1007	0	Y 目标位置 1
V1008V1009	4500	圆弧半径
V1010V1011	50000	脉冲频率
V1012	60	加减速时间

2、程序首次扫描时 SM2=ON，设置 PLS1、PLS3 定位模式为绝对地址模式（SM115=ON，SM147=ON）。

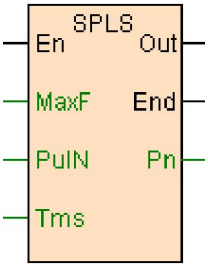
3、X 轴为 PLS1，Y 轴为 PLS3，当 M0=ON，V0=0，执行以当前位置（0,0）为起点到（9000,0）半径为 4500 的逆时针圆弧插补。

4、1 段圆弧插补完成 M100=ON，V0=1，执行从（9000,0）到（0,0）半径为 4500 的逆时针圆弧插补。

5、2 段圆弧插补完成 M100=ON，V0=2，复位 M0，M0=OFF。

SPLS（简单脉冲输出）

指令格式及参数说明

语言	LD	FBD	IL
指令格式			SPLS En, MaxF, PulN, Tms, Out, End, Pn

参数	参数定义	输入	输出	说明
En	使能	√		
MaxF	脉冲输出最大频率	√		占用 2 个连续元件
PulN	脉冲输出总数	√		占用 2 个连续元件
Tms	加减速时间	√		(5~5000ms)
Out	脉冲输出		√	
End	脉冲输出完成		√	
Pn	已输出脉冲数		√	占用 2 个连续元件

【指令功能及作用说明】

1、SPLS 指令通过主机高速脉冲输出点 Yn 输出高速脉冲，它不使用 PLSx 高速脉冲输出通道，1 路高速脉冲输出通道有 2 个高速脉冲输出点。这样有 8 个脉冲输出通道的主机可以实现 16 路高速脉冲输出。

2、SPLS 指令只输出脉冲，方向指示需通过程序控制其他 Y 点输出，它不产生脉冲输出中断，不使用 SM、SV。

3、当 Tms=0 表示不进行加减速，否则当 Tms<最小加减速时间则使用最小加减速时间，当 Tms>最大加减速时间则使用最大加减速时间。

4、当 MaxF≤0 不输出脉冲，否则当 MaxF<最小频率（10Hz）则使用最小频率，当 MaxF>最大频率则使用最大频率。

- 5、当 PuIN=0 表示不计脉冲数输出；当 PuIN>0 则按照 PuIN 个脉冲输出，脉冲输出完成 End 置位；当 PuIN<0 不输出脉冲。
- 6、MaxF 可实时改变，PuIN 不能被实时修改。

注：E/S 系列主机不支持 SPLS 指令。

【指令示例】

//Network 1 正转控制



//Network 2 反转控制



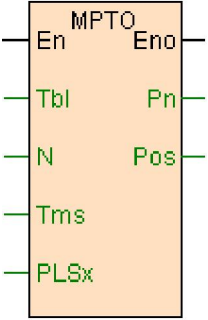
【程序说明】

- 1、当 M0=ON 时，Y3 以 50KHz 的频率输出 30000 个脉冲，Y15=OFF（OFF 表示正方向），M100=OFF
- 2、当脉冲输出完成后，M100=ON，复位 M0，置位 M1。
- 3、当 M1=ON 时，Y3 以 200KHz 的频率输出 15000 个脉冲，Y15=ON（ON 表示反方向），M101=OFF
- 4、当脉冲输出完成后，M101=ON，复位 M1。

MPTO（多段脉冲输出）

指令格式及参数说明

语言	LD	FBD	IL
----	----	-----	----

语言	LD	FBD	IL
指令格式			MPTO En, Tbl, N, Tms, PLSx

参数	参数定义	输入	输出	说明
En	使能	√		
Tbl	段起始元件	√		每段占用 4 个寄存器，占用 4N 个连续元件
N	段数	√		1~255
Tms	加减速时间	√		(5~5000ms)
PLSx	脉冲输出通道	√		
Eno	使能输出		√	
Pn	当前段号		√	占用 1 个系统寄存器
Pos	当前位置		√	占用 2 个系统寄存器

【指令功能及作用说明】

- 1、MPTO 为多段脉冲输出指令，Tbl 指定段起始元件，每段占用 4 个寄存器，前 2 个寄存器为该段输出脉冲的频率，后 2 个寄存器为该段输出脉冲的个数。
- 2、如果段频率小于最小频率（10Hz）则按照最小频率输出，如果段频率大于最大频率则按照最大频率输出。
- 3、如果段脉冲数大于 0 表示输出正向脉冲，小于 0 表示输出反向脉冲，等于 0 则不输出，报 3 号参数错误。
- 4、当 Tms=0 表示不进行加减速；否则当 Tms<最小加减速时间则使用最小加减速时间，当 Tms>最大加减速时间则使用最大加减速时间。
- 5、段之间如果频率不同，MPTO 会根据加减速时间 Tms 进行平滑变速。
- 6、MPTO 指令与 [SM 系统状态位](#)、[SV 系统寄存器](#) 有关，并且会产生脉冲输出中断，指令执行开始输出脉冲时产生“PLSx 开始输出脉冲”中断，指令执行完成停止输出脉冲时产生“PLSx 输出脉冲结束”中断。
- 7、En=ON 时指令执行，Eno=ON ；当脉冲输出过程中 En 变为 OFF，指令根据 Tms 减速刹车停止（如果没有设置禁止刹车），Eno=OFF。
- 8、MPTO 指令没有条数限制，可以与其他脉冲输出指令并存，但每个脉冲输出通道同时只能执行一条指令。
- 9、在指令执行脉冲输出过程中，当前段及下一段的脉冲频率、脉冲数不能被修改（修改不会生效），其他段数据可以实时修改。

注：E/S 系列主机不支持 MPTO 指令。

【指令示例】

本示例以多段脉冲输出指令 MPTO 控制伺服做前进、后退的动作

“前进段数据”定义如下表：

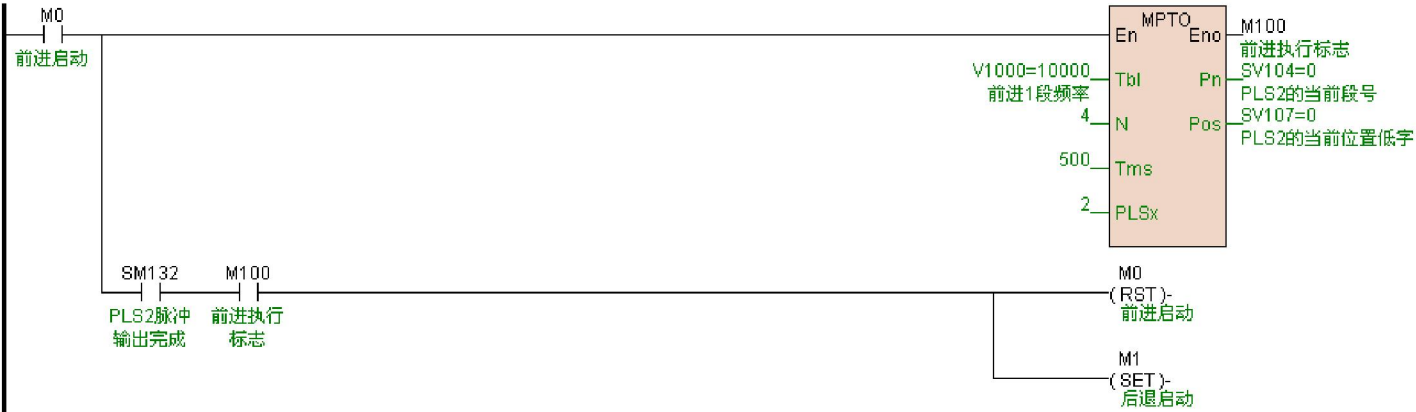
寄存器元件	32 位寄存器值	注释
V1000V1001	10000	前进 1 段频率
V1002V1003	25000	前进 1 段脉冲数

V1004V1005	50000	前进 2 段频率
V1006V1007	100000	前进 2 段脉冲数
V1008V1009	50000	前进 3 段频率
V1010V1011	120000	前进 3 段脉冲数
V1012V1013	15000	前进 4 段频率
V1014V1015	23000	前进 4 段脉冲数

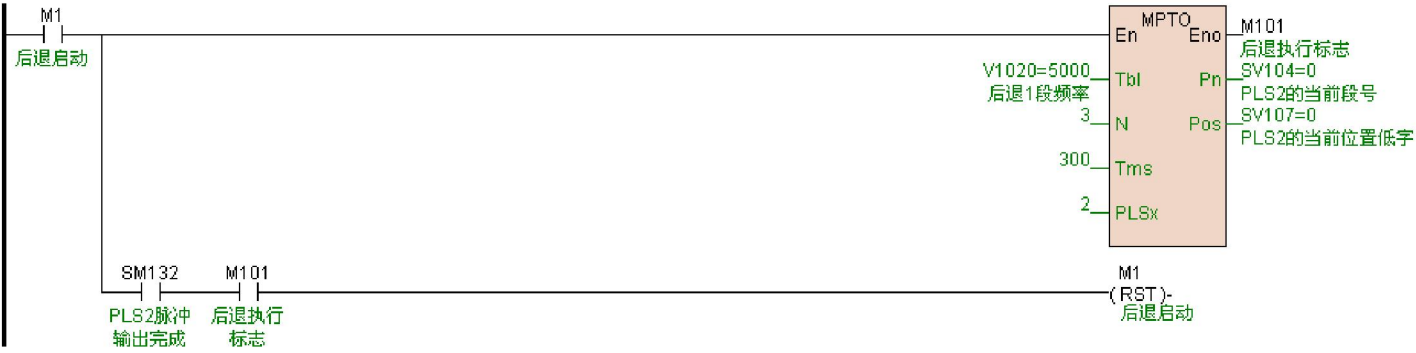
“后退段数据”定义如下表：

寄存器元件	32 位寄存器值	注释
V1020V1021	5000	后退 1 段频率
V1022V1023	-9000	后退 1 段脉冲数
V1024V1025	30000	后退 2 段频率
V1026V1027	-45000	后退 2 段脉冲数
V1028V1029	10000	后退 3 段频率
V1030V1031	-20000	后退 3 段脉冲数

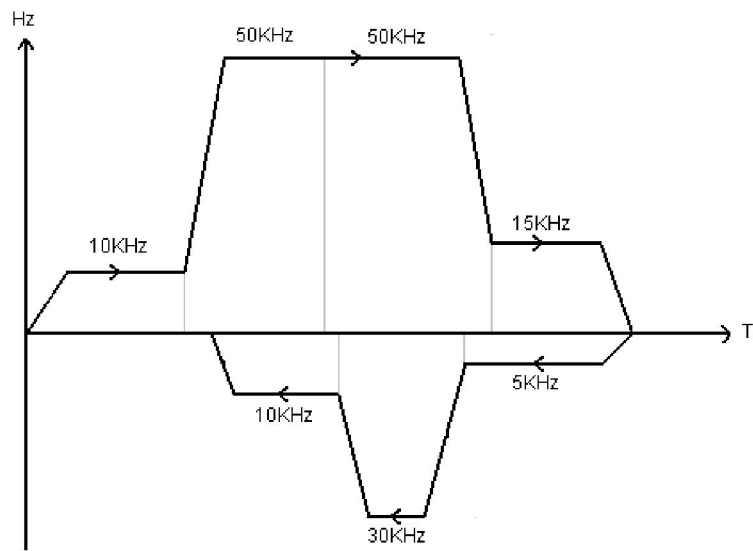
//Network 1 前进



//Network 2 后退



【程序示意图】



【程序说明】

- 1、当 M0=ON 时，Network 1 的 MPTO 指令加载执行，如果参数没有错误，PLS2 开始输出脉冲（Y4 脉冲，Y5 方向），SM132 被复位同时产生中断 5（PLS2 开始输出脉冲），M100=ON。
- 2、当前进 4 段脉冲都输出完成后，SM132 被置位同时产生中断 6（PLS2 输出脉冲结束），因为 M0=ON SM132=ON M100=ON，执行 RST M0 和 SET M1，M0=OFF M1=ON。
- 3、当 M1=ON 时，Network 2 的 MPTO 指令加载执行，如果参数没有错误，PLS2 开始输出脉冲（Y4 脉冲，Y5 方向），SM132 被复位同时产生中断 5（PLS2 开始输出脉冲），M101=ON。
- 4、当后退 3 段脉冲都输出完成后，SM132 被置位同时产生中断 6（PLS2 输出脉冲结束），因为 M1=ON SM132=ON M101=ON，执行 RST M1，M1=OFF。

SYNP（随动脉冲输出）

指令格式及参数说明

语言	LD	FBD	IL
指令格式	SYNP EnEno RMul RDiv PuIN HSCx PLSx		SYNP En, RMul, RDiv, PuIN, HSCx, PLSx

参数	参数定义	输入	输出	说明
En	使能	√		
RMul	乘系数	√		占用 2 个连续元件
RDiv	除系数	√		占用 2 个连续元件
PuIN	延迟脉冲数	√		
HSCx	高速计数器号	√		

参数	参数定义	输入	输出	说明
PLSx	脉冲输出通道	√		
Eno	使能输出		√	

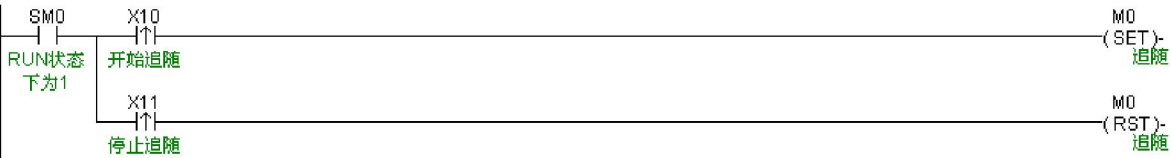
【指令功能及作用说明】

- 1、SYNP 指令指定 PLSx 的脉冲输出是跟随 HSCx 的脉冲输入而变化，它们之间的比例系数 $K=RMul / RDiv$ ，PLSx 的方向=HSCx 的方向。RMul 和 RDiv 都必须大于 0。
- 2、PulN 指定延迟脉冲数，在 HSCx 输入了 PulN 个脉冲后，PLSx 开始跟随 HSCx，PulN≤0 则不延迟。
- 3、SYNP 指令与 [SM 系统状态位](#)、[SV 系统寄存器](#) 有关，并且会产生高速计数器方向改变中断。
- 4、SYNP 指令执行时，HSCx、PLSx 不能被其他指令占用。
- 5、在此指令启动后，其所有参数将不接受修改，直到指令关闭。

注：E/S 系列主机不支持 SYNP 指令。

【指令示例】

//Network 1



//Network 2



【程序说明】

- 1、X10=ON，则 M0=ON，SYNP 指令执行，PLS1 在延迟 15 个脉冲后，按照 1:1 比例跟随 HSC0 的脉冲输入而输出。
- 2、X11=ON，则 M0=OFF，SYNP 指令停止执行。

PSTOP（停止脉冲输出）

指令格式及参数说明

语言	LD	FBD	IL
指令格式			PSTOP En, PLSx

参数	参数定义	输入	输出	说明
En	使能	√		
PLSx	脉冲输出通道	√		

参数	参数定义	输入	输出	说明
Eno	使能输出		√	

【指令功能及作用说明】

PSTOP 指令停止脉冲输出通道 PLSx 的脉冲输出。

注：E/S 系列主机不支持 PSTOP 指令。

【指令示例】

参见 [PLSR](#) 指令示例。

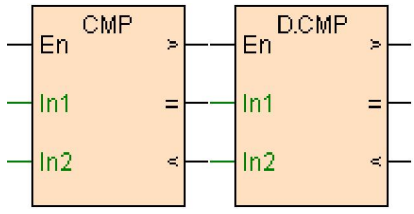
4.8 比较指令

比较指令列表

指令名称	8 位模式	32 位模式	指令功能	支持语言		
				LD	FBD	IL
CMP		D.CMP	比较指令	√	√	√
ZCP		D.ZCP	区域比较	√	√	√
MATC		D.MATC	数值匹配	√	√	√
ABSC		D.ABSC	绝对凸轮比较	√	√	√
BON			ON 位判定	√	√	√
BONC		D.BONC	ON 位数量	√	√	√
MAX		D.MAX	最大值	√	√	√
MIN		D.MIN	最小值	√	√	√
SEL		D.SEL	条件选择	√	√	√
MUX		D.MUX	多路选择	√	√	√

CMP、D.CMP（比较）

指令格式及参数说明

语言	LD	FBD	IL
16、32 位 指令格式			CMP En, In1, In2, Out D.CMP En, In1, In2, Out

参数	参数定义	输入	输出	说明
En	使能	√		
In1	输入 1	√		
In2	输入 2	√		
Out	状态输出		√	占用 3 个连续元件

【指令功能及作用说明】

CMP 是 16 位整数比较指令（D.CMP 为 32 位整数比较），它同时输出 >、=、< 这 3 个结果。

【指令示例】

//Network 1



【程序说明】

- 1、16 位比较 CMP 指令，当 AI1>500 则 M10=ON，当 AI1=500 则 M11=ON，当 AI1<500 则 M12=ON。
- 2、32 位比较 D.CMP 指令，当 V10V11>V0V1 则 M20=ON，当 V10V11=V0V1 则 M21=ON，当 V10V11<V0V1 则 M22=ON。

ZCP、D.ZCP（区域比较）

指令格式及参数说明

语言	LD	FBD	IL
16、32 位指令格式	En ZCP > In () Up < Down	En D.ZCP > In () Up < Down	ZCP En, In, Up, Down, Out D.ZCP En, In, Up, Down, Out

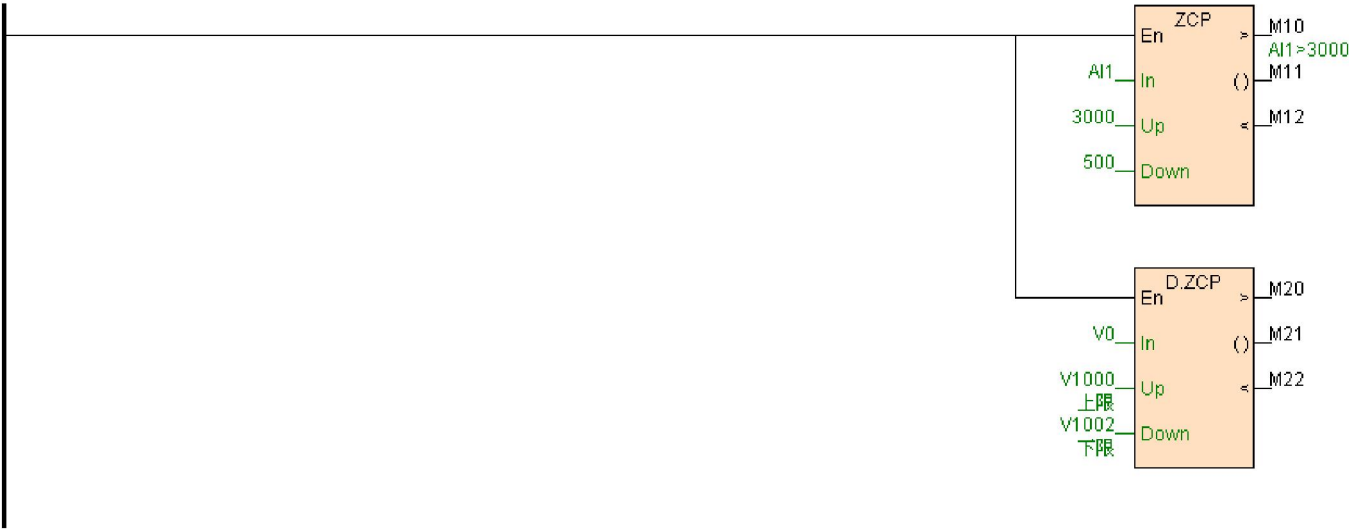
参数	参数定义	输入	输出	说明
En	使能	√		
In	输入	√		
Up	区域上限	√		
Down	区域下限	√		
Out	状态输出		√	占用 3 个连续元件

【指令功能及作用说明】

- 1、ZCP 是 16 位整数区域比较指令（D.CMP 为 32 位整数区域比较），它同时输出>、=、<这 3 个结果。
- 2、如果区域上限 Up<区域下限 Down，那么指令会自动将它们交换。

【指令示例】

//Network 1



【程序说明】

- 1、16 位比较 ZCP 指令 ， 当 AI1>3000 则 M10=ON， 当 AI1≤3000 并且 AI1≥500 则 M11=ON， 当 AI1<500 则 M12=ON。
- 2、32 位比较 D.ZCP 指令， 当 V0V1>V1000V1001 则 M20=ON， 当 V0V1≤V1000V1001 并且 V0V1≥V1002V1003 则 M21=ON， 当 V0V1<V1002V1003 则 M22=ON。

MATC、D.MATC（数值匹配）

指令格式及参数说明

语言	LD	FBD	IL
16、32 位 指令格式			MATC En, In, Par, N, Out D.MATC En, In, Par, N, Out

参数	参数定义	输入	输出	说明
En	使能	√		
In	输入	√		
Par	数值匹配起始元件	√		MATC：占用 N 个连续元件，D.MATC：占用 2N 个连续元件
N	比较个数	√		
Out	状态输出		√	

【指令功能及作用说明】

MATC 指令将 In 输入与以 Par 起始的 N 个数字进行比较，如果 In 等于其中一个则表示匹配正确 Out=ON，否则 Out=OFF。

【指令示例】

//Network 1



【程序说明】

- 1、MATC 指令从母排取电一直执行。
- 2、只要 V0 等于 V1000、V1001、V1002、V1003、V1004、V1005 其中一个则匹配正确 M0=ON，否则 M0=OFF。

ABSC、D.ABSC（绝对凸轮比较）

指令格式及参数说明

语言	LD	FBD	IL
16、32 位 指令格式			ABSC En, In, Par, N, Out D.ABSC En, In, Par, N, Out

参数	参数定义	输入	输出	说明
En	使能	√		
In	输入	√		
Par	多段比较起始元件	√		ABSC：占用 2N 个连续 元件，D.ABSC：占用 4N 个连续元件
N	比较段数	√		1-64
Out	比较结果起始元件		√	占用 N 个连续元件

【指令功能及作用说明】

- 1、ABSC 指令将 In 输入与以 Par 起始的 N 段区域进行比较 ，比较结果输出到 Out 起始的 N 个连续元件。
- 2、对于下限值≤上限值的段，若下限值≤In≤上限值，则该段比较结果输出=ON，若 In<下限值或 In>上限值，则该段比较结果输出=OFF。
- 3、对于下限值>上限值的段，若上限值≤In≤下限值，则该段比较结果输出=OFF，若 In>下限值或 In<上限值,则该段比较结果输出=ON。
- 4、参数 Par 与 N 两者的关系说明：

编号	比较段数 N 值	Par 元件含义	Out 元件含义
1	1	第 1 段下限	第 1 段比较结果输出
2		第 1 段上限	
3	2	第 2 段下限	第 2 段比较结果输出
4		第 2 段上限	
...	...	...	...

编号	比较段数 N 值	Par 元件含义	Out 元件含义
15	8	第 8 段下限	第 8 段比较结果输出
16		第 8 段上限	

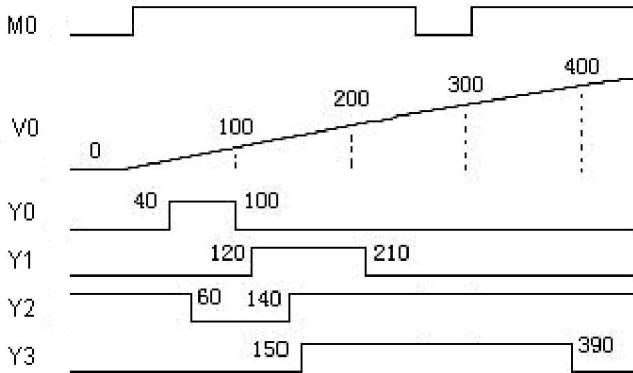
【指令示例】

//Network 1



【程序示意图】

Par 元件	值	说明
V1000	40	第 1 段 下限
V1001	100	第 1 段 上限
V1002	120	第 2 段 下限
V1003	210	第 2 段 上限
V1004	140	第 3 段 下限
V1005	60	第 3 段 上限
V1006	150	第 4 段 下限
V1007	390	第 4 段 上限



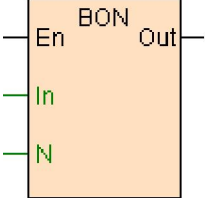
【程序说明】

- 1、M0=ON，将 V0 与以 V1000 起始的 4 段区域进行比较。
- 2、当 V0 为 40~100 时，Y0=ON，否则 Y0=OFF；当 V0 为 120~210 时，Y1=ON，否则 Y1=OFF；当 V0 为 60~140 时，Y2=OFF，否则 Y2=ON；当 V0 为 150~390 时，Y3=ON，否则 Y3=OFF。
- 3、当 M0=OFF，指令停止执行，Y0~Y3 保持不变。

BON（ON 位判定）

指令格式及参数说明

语言	LD	FBD	IL
----	----	-----	----

语言	LD	FBD	IL
指令格式			BON En, In, N, Out

参数	参数定义	输入	输出	说明
En	使能			
In	输入	√		
N	第几位	√		
Out	状态输出		√	

【指令功能及作用说明】

BON 指令用于判定寄存器某个数位是否为 1，结果输出到 Out。

【指令示例】

//Network 1

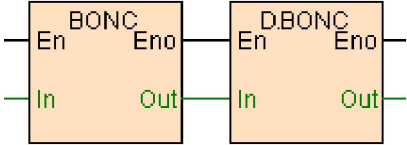


【程序说明】

- 1、BON 指令从母排取电一直执行
- 2、假如 V0=8（二进制 00000000 00001000，第 4 位为 1），那么 M0=ON。

BONC、D.BONC（ON 位数量）

指令格式及参数说明

语言	LD	FBD	IL
16、32 位 指令格式			BONC En, In, Out D.BONC En, In, Out

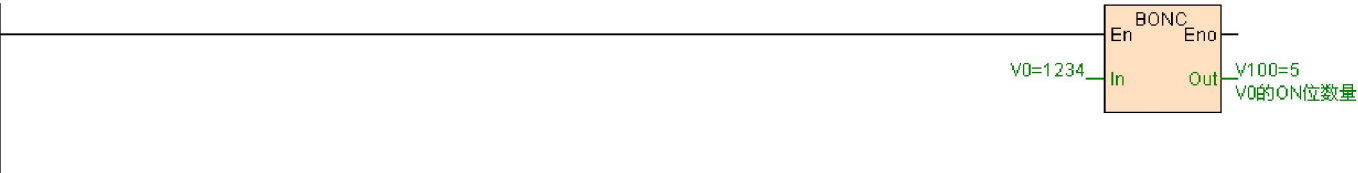
参数	参数定义	输入	输出	说明
En	使能	√		
In	输入	√		
Eno	使能输出		√	
Out	输出		√	

【指令功能及作用说明】

BONC 指令用于得到寄存器中数位为 1 的数量，结果输出到 Out。

【指令示例】

//Network 1



【程序说明】

- 1、B0NC 指令从母排取电一直执行
- 2、假如 V0=1234（二进制 00000100 11010010，共 5 位为 1），那么 V100=5。

MAX、D.MAX（最大值）

指令格式及参数说明

语言	LD	FBD	IL
16、32 位 指令格式			MAX En, Par, N, Out D.MAX En, Par, N, Out

参数	参数定义	输入	输出	说明
En	使能	√		
Par	比较数值起始元件	√		MAX：占用 N 个连续元件，D.MAX：占用 2N 个连续元件
N	比较数值个数	√		
Eno	使能输出		√	
Out	比较结果输出		√	

【指令功能及作用说明】

MAX 指令将以 Par 起始的 N 个数字进行比较，将最大值输出到 Out。

【指令示例】

//Network 1



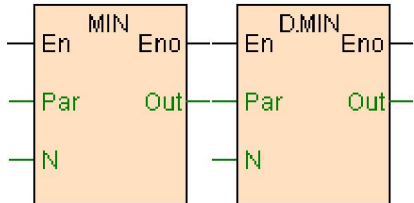
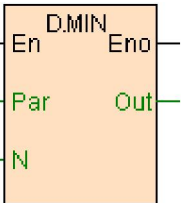
【程序说明】

- 1、M0=ON，假如 V1000=30、V1001=-150、V1002=25、V1003=8、V1004=95、V1005=-20，那么 V0=95。
- 2、M0=ON，假如 V1100V1101=30000、V1102V1103=-50000、V1104V1105=23000、V1106V1107=600、V1108V1109=1500，那么 V10V11=30000。

3、当 M0=OFF，指令停止执行，Out 保持不变。

MIN、D.MIN（最小值）

指令格式及参数说明

语言	LD	FBD	IL
16、32 位 指令格式			MIN En, Par, N, Out D.MIN En, Par, N, Out

参数	参数定义	输入	输出	说明
En	使能	√		
Par	比较数值起始元件	√		MIN：占用 N 个连续元件，D.MIN：占用 2N 个连续元件
N	比较数值个数	√		
Eno	使能输出		√	
Out	比较结果输出		√	

【指令功能及作用说明】

MIN 指令将以 Par 起始的 N 个数字进行比较，将最小值输出到 Out。

【指令示例】

//Network 1



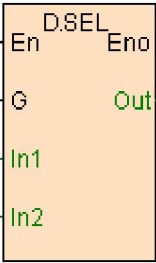
【程序说明】

- 1、M0=ON，假如 V1000=30、V1001=-150、V1002=25、V1003=8、V1004=95、V1005=-20，那么 V0=-150。
- 2、M0=ON，假如 V1100V1101=30000、V1102V1103=-50000、V1104V1105=23000、V1106V1107=600、V1108V1109=1500，那么 V10V11=-50000。
- 3、当 M0=OFF，指令停止执行，Out 保持不变。

SEL、D.SEL（条件选择）

指令格式及参数说明

语言	LD	FBD	IL
----	----	-----	----

语言	LD	FBD	IL
16、32 位 指令格式			SEL En, G, In1, In2, Out D.SEL En, G, In1, In2, Out

参数	参数定义	输入	输出	说明
En	使能	√		
G	选择条件	√		
In1	选择数 1	√		
In2	选择数 2	√		
Eno	使能输出		√	
Out	选择结果输出		√	

【指令功能及作用说明】

SEL 指令为二选一指令，G=OFF 则 Out=In1，G=ON 则 Out=In2。

【指令示例】

//Network 1

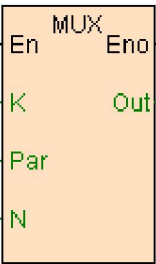
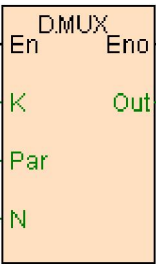


【程序说明】

- 1、SEL 指令从母排取电一直执行。
- 2、假如 AI0=230、AI1=512，那么 M0=OFF 时 V0=230，M0=ON 时 V0=512。

MUX、D.MUX（多路选择）

指令格式及参数说明

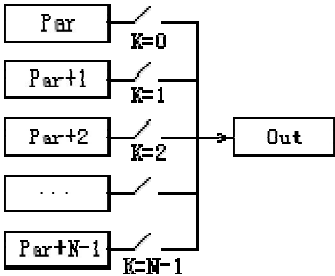
语言	LD	FBD	IL
16、32 位 指令格式			MUX En, K, Par, N, Out D.MUX En, K, Par, N, Out

参数	参数定义	输入	输出	说明
En	使能	√		

参数	参数定义	输入	输出	说明
K	选择通道号	√		
Par	选择数据起始元件	√		MUX：占用 N 个连续元件，D.MUX：占用 2N 个连续元件
N	选择数据个数	√		
Eno	使能输出		√	
Out	选择结果输出		√	

【指令功能及作用说明】

MUX 指令据选择通道号 K 的值（K=0~N-1）从 N 个地址连续的寄存器中选择其中的一个输出到 Out（多选一）。示意图如下图所示：



【指令示例】

//Network 1



【程序说明】

- 1、MUX 指令从母排取电一直执行。
- 2、假如 V1000=30、V1001=-150、V1002=25、V1003=8、V1004=95、V1005=-20，当 V00=3 表示选择第 4 个数值输出，V10=8。

4.9 移动指令

移动指令列表

指令名称	8 位模式	32 位模式	指令功能	支持语言		
				LD	FBD	IL
LBST			低字节赋值	√	√	√
HBST			高字节赋值	√	√	√
MOV		D.MOV	移动	√	√	√
BMOV			块移动	√	√	√
FILL			填充	√	√	√
XCH			字节交换	√	√	√
BXCH			块交换	√	√	√
SHL			位左移	√	√	√
SHR			位右移	√	√	√
WSHL			字左移	√	√	√
WSHR			字右移	√	√	√

指令名称	8 位模式	32 位模式	指令功能	支持语言		
				LD	FBD	IL
ROL			位循环左移	√	√	√
ROR			位循环右移	√	√	√
WROL			字循环左移	√	√	√
WROR			字循环右移	√	√	√
BSHL			字节左移	√	√	√
BSHR			字节右移	√	√	√
ATBL			添加到队列	√	√	√
FIFO			先进先出	√	√	√
LIFO			后进先出	√	√	√
SORT			数据排序	√	√	√

LBST（低字节赋值）

指令格式及参数说明

语言	LD	FBD	IL
指令格式			LBST En, In, Out

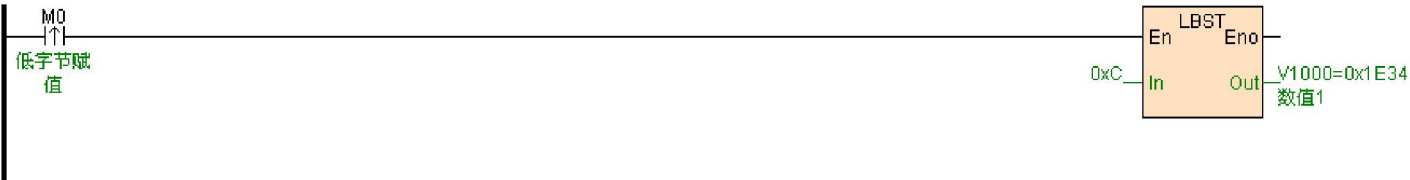
参数	参数定义	输入	输出	说明
En	使能	√		
In	输入	√		
Eno	使能输出		√	
Out	数据输出		√	

【指令功能及作用说明】

LBST 用于将指定的数据赋值到输出寄存器 Out 的低字节中，高字节保持不变。

【指令示例】

//Network 1 低字节赋值



【程序说明】

假如初始 V1000=0x1E34，那么 M0=ON 时 V1000=0x1E0C。

HBST（高字节赋值）

指令格式及参数说明

语言	LD	FBD	IL
----	----	-----	----

语言	LD	FBD	IL
指令格式			HBST En, In, Out

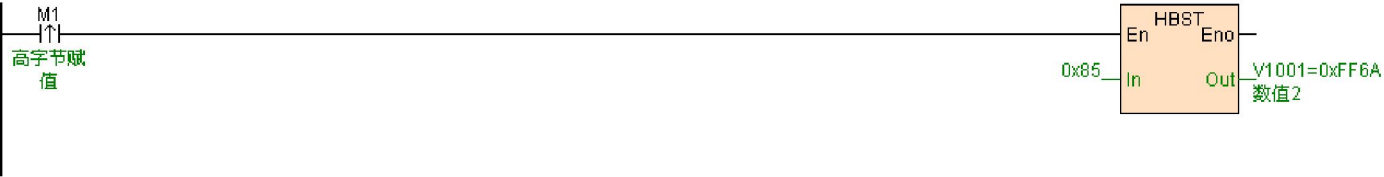
参数	参数定义	输入	输出	说明
En	使能	√		
In	输入	√		
Eno	使能输出		√	
Out	数据输出		√	

【指令功能及作用说明】

HBST 用于将指定的数据赋值到输出寄存器 Out 的高字节中，低字节保持不变。

【指令示例】

//Network 2 高字节赋值

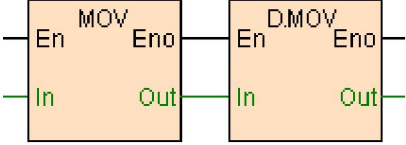


【程序说明】

假如初始 V1001=0xFF6A，那么 M1=ON 时 V1001=0x856A

MOV、D.MOV（移动）

指令格式及参数说明

语言	LD	FBD	IL
16、32 位 指令格式			MOV En, In, Out D.MOV En, In, Out

参数	参数定义	输入	输出	说明
En	使能	√		
In	输入	√		
Eno	使能输出		√	
Out	数据输出		√	

【指令功能及作用说明】

移动指令 MOV 也叫赋值指令，用于将指定的数据赋值到输出寄存器 Out。

【指令示例】

//Network 1 程序开始赋初始值



【程序说明】

程序首次扫描时赋初始值，V0=80，V10V11=-50。

BMOV（块移动）

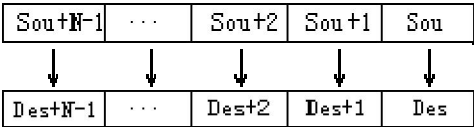
指令格式及参数说明

语言	LD	FBD	IL
指令格式			BMOV En, Sou, N, Des

参数	参数定义	输入	输出	说明
En	使能	√		
Sou	欲移块起始元件	√		占用 N 个连续元件
N	欲移数据的个数	√		
Eno	使能输出		√	
Des	移动目的块起始元件		√	占用 N 个连续元件

【指令功能及作用说明】

BMOV 块移动指令将 Sou 起始的 N 个元件移到 Des 起始的 N 个元件中。如下图所示：



【指令示例】

//Network 1



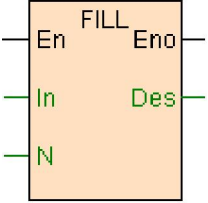
【程序说明】

当 M0=ON，把 V1000~V1005 移动到 V0~V5，把 X0~X4 移动到 Y2~Y6。

Sou 元件	初始值	移动结果
V1000	30	V0=30
V1001	-150	V1=-150
V1002	25	V2=25
V1003	8	V3=8
V1004	95	V4=95
V1005	-20	V5=-20
X0	ON	Y2=ON
X1	OFF	Y3=OFF
X2	ON	Y4=ON
X3	ON	Y5=ON
X4	OFF	Y6=OFF

FILL（填充）

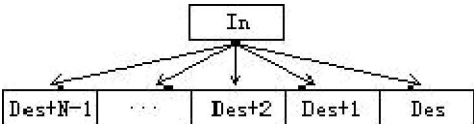
指令格式及参数说明

语言	LD	FBD	IL
指令格式			FILL En, In, N, Des

参数	参数定义	输入	输出	说明
En	使能	√		
In	欲填充的数据	√		
N	欲填充数据的个数	√		
Eno	使能输出		√	
Des	移动目的块起始元件		√	占用 N 个连续元件

【指令功能及作用说明】

填充指令 FILL 用于将 In 的值填充 Des 起始的 N 个元件中。可以用来批量复位、置位寄存器元件和位元件。如下图所示：



【指令示例】

//Network 1 把V100~104这5个寄存器复位为0，把Y0~Y11复位为OFF，把M100~M105置位为ON



【程序说明】

当 M0=ON，把 V100~104 这 5 个寄存器复位为 0，把 Y0~Y11 复位为 OFF，把 M100~M105 置位为 ON。

XCH（字节交换）、D.XCH（寄存器交换）

指令格式及参数说明

语言	LD	FBD	IL
16、32 位 指令格式			XCH En, Sou, N D.XCH En, Sou, N

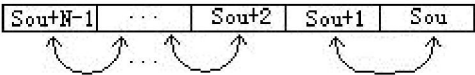
参数	参数定义	输入	输出	说明
En	使能	√		
Sou	欲交换的寄存器起始元件	√		占用 N 个连续元件
N	欲交换寄存器的个数	√		
Eno	使能输出		√	

【指令功能及作用说明】

1、16 位指令 XCH 为字节交换，用于将 Sou 起始的 N 个寄存器的高低字节互相交换。如下图所示：



2、32 位指令 D.XCH 为寄存器交换，用于将 Sou 起始的 N 个寄存器元件的每相邻 2 个寄存器互相交换，如果 N 为奇数则最后一个寄存器不变。如下图所示：

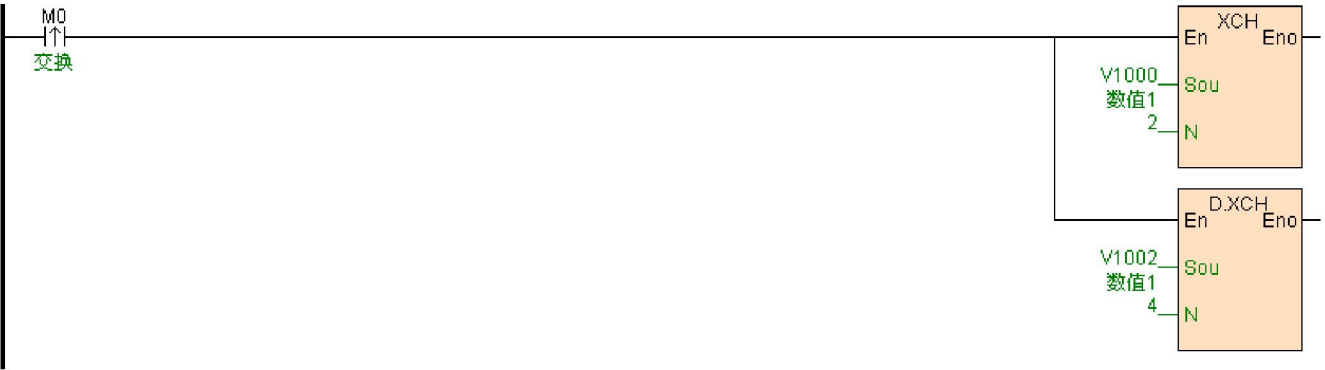


3、XCH 指令一般用边沿执行。

注：E/S 系列主机的 XCH 指令参见“XCH”。

【指令示例】

//Network 1



【程序说明】

当 M0=ON，把 V1000、V1001 这 2 个寄存器的高低字节交换，V1002、V1003 这 2 个寄存器交换，V1004、V1005 这 2 个寄存器交换。

Sou 元件	初始值	交换结果
V1000	30 (0x001E)	V1000=7680 (0x1E00)
V1001	-150 (0xFF6A)	V1001=27391 (0x6AFF)
V1002	25	V1002=8
V1003	8	V1003=25
V1004	95	V1004=-20
V1005	-20	V1005=95

BXCH（块交换）

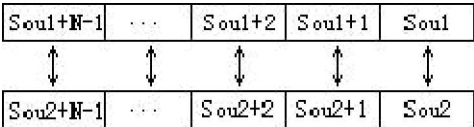
指令格式及参数说明

语言	LD	FBD	IL
指令格式			BXCH En, Sou1, Sou2, N

参数	参数定义	输入	输出	说明
En	使能	√		
Sou1	源 1 起始元件	√		占用 N 个连续元件
Sou2	源 2 起始元件	√		占用 N 个连续元件
N	元件个数	√		
Eno	使能输出		√	

【指令功能及作用说明】

1、块交换指令 BXCH 用于将 Sou1 起始的 N 个元件与以 Sou2 起始的 N 个元件互相交换。如下图所示：



2、BXCH 指令一般用边沿执行。

【指令示例】

//Network 1



【程序说明】

当 M0=ON，把 V1000~V1002 这 3 个寄存器与 V1003~V1005 这 3 个寄存器交换。

元件	初始值	交换结果
V1000	30	V1000=8
V1001	-150	V1001=95
V1002	25	V1002=-20
V1003	8	V1003=30
V1004	95	V1004=-150
V1005	-20	V1005=25

SHL（位左移）

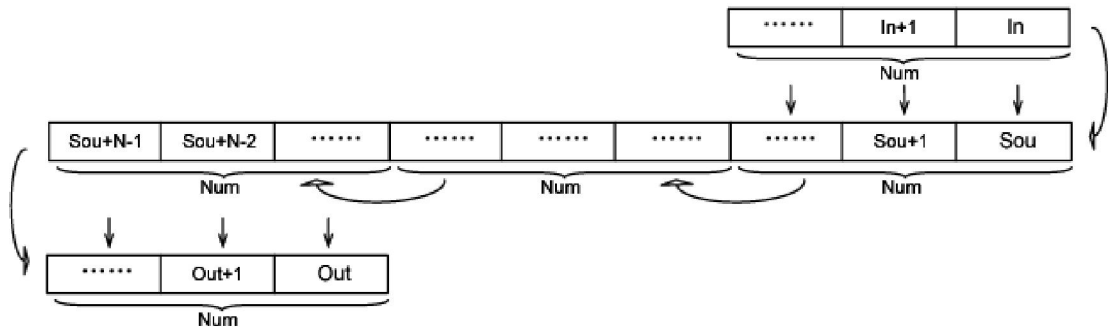
指令格式及参数说明

语言	LD	FBD	IL
指令格式			SHL En, In, Sou, N, Num, Out

参数	参数定义	输入	输出	说明
En	使能	√		
In	移入位起始元件	√		占用 Num 个连续元件
Sou	源起始元件	√		占用 N 个连续元件
N	元件个数	√		
Num	移位次数	√		
Eno	使能输出		√	
Out	移出元件		√	占用 Num 个连续元件

【指令功能及作用说明】

- 1、以 Num 位一组，SHL 指令将以 Sou 起始的 N 个元件左移 Num 位，移入以 In 起始的 Num 个元件，移出放在以 Out 起始的 Num 个元件。如下图所示：



- 2、如果 Sou 是寄存器元件，那么就将以 Sou 起始的 N 个寄存器按位左移 Num 位。
- 3、 $1 \leq \text{Num} \leq N$ ，否则指令不执行。
- 4、SHL 指令一般用边沿执行。

注：E/S 系列主机的 SHL 指令参见“[SHL](#)”。

【指令示例】

//Network 1 位地址位左移



//Network 2 寄存器位左移



【程序说明】

- 1、当 M0=ON，把 M100~M105 左移 3 位，移入 X0~X2，移出放在 Y0~Y2。

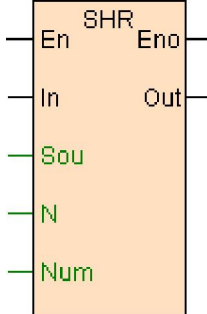
Sou 元件	初始值	左移结果
M100	ON	M100=OFF
M101	OFF	M101=ON
M102	ON	M102=ON
M103	ON	M103=ON
M104	ON	M104=OFF
M105	OFF	M105=ON
X0	OFF	
X1	ON	
X2	ON	
Y0		Y0=ON
Y1		Y1=ON
Y2		Y2=OFF

2、当 M1=ON，把 V1000~V1005 按位左移 3 位，移入 X0~X2，移出放在 M200~M202。

Sou 元件	初始值	左移结果
V1000	00000000 00011110	V1000=00000000 11110110
V1001	11111111 01101010	V1001=11111011 01010000
V1002	00000000 00011001	V1002=00000000 11001111
V1003	00000000 00001000	V1003=00000000 01000000
V1004	00000000 01011111	V1004=00000010 11111000
V1005	11111111 11101100	V1005=11111111 01100000
X0	OFF	
X1	ON	
X2	ON	
M200		M200=ON
M201		M201=ON
M202		M202=ON

SHR（位右移）

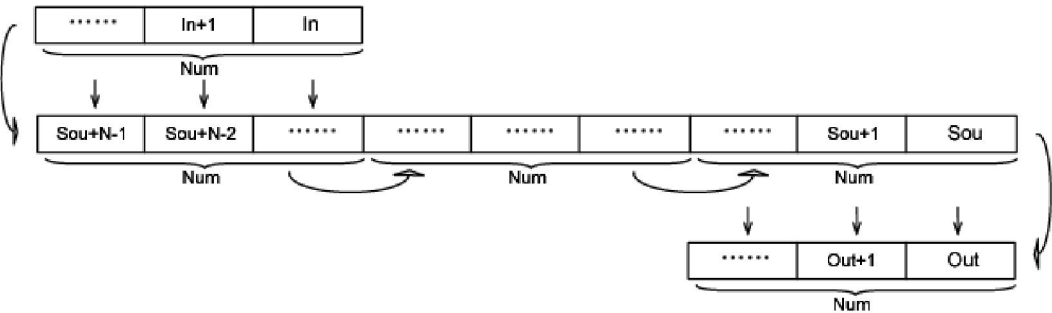
指令格式及参数说明

语言	LD	FBD	IL
指令格式			SHR En, In, Sou, N, Num, Out

参数	参数定义	输入	输出	说明
En	使能	√		
In	移入位起始元件	√		占用 Num 个连续元件
Sou	源起始元件	√		占用 N 个连续元件
N	元件个数	√		
Num	移位次数	√		
Eno	使能输出		√	
Out	移出元件		√	占用 Num 个连续元件

【指令功能及作用说明】

1、以 Num 位一组，SHR 指令将 Sou 起始的 N 个元件右移 Num 位，移入以 In 起始的 Num 个元件，移出放在以 Out 起始的 Num 个元件。如下图所示：



- 2、如果 Sou 是寄存器元件，那么就将以 Sou 起始的 N 个寄存器按位右移 Num 位。
- 3、 $1 \leq \text{Num} \leq N$ ，否则指令不执行。
- 4、SHR 指令一般用边沿执行。

注：E/S 系列主机的 SHR 指令参见“[SHR](#)”。

【指令示例】

//Network 1 位地址位右移



//Network 2 寄存器位右移



【程序说明】

- 1、当 M0=ON，把 M100~M105 右移 3 位，移入 X0~X2，移出放在 Y0~Y2。

Sou 元件	初始值	右移结果
M100	ON	M100=ON
M101	OFF	M101=ON
M102	OFF	M102=OFF
M103	ON	M103=OFF
M104	ON	M104=ON
M105	OFF	M105=ON
X0	OFF	
X1	ON	
X2	ON	
Y0		Y0=ON
Y1		Y1=OFF
Y2		Y2=OFF

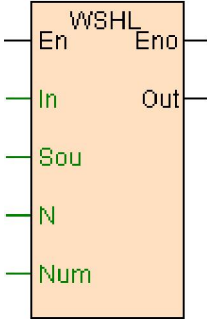
- 2、当 M1=ON，把 V1000~V1005 按位右移 3 位，移入 X0~X2，移出放在 M200~M202。

Sou 元件	初始值	右移结果
V1000	00000000 00011110	V1000=01000000 00000011
V1001	11111111 01101010	V1001=00111111 11101101
V1002	00000000 00011001	V1002=00000000 00000011
V1003	00000000 00001000	V1003=11100000 00000001
V1004	00000000 01011111	V1004=10000000 00001011
V1005	11111111 11101100	V1005=11011111 11111101
X0	OFF	
X1	ON	

X2	ON	
M200		M200=OFF
M201		M201=ON
M202		M202=ON

WSHL（字左移）

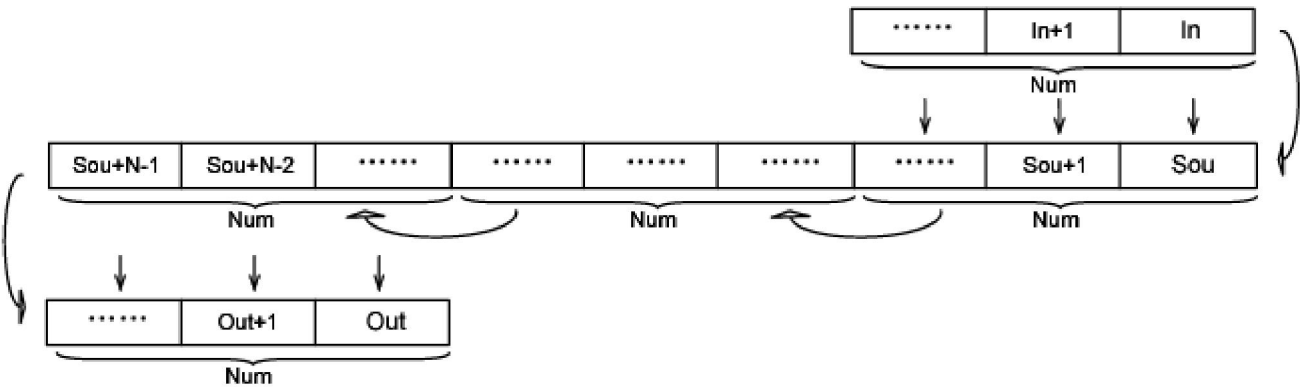
指令格式及参数说明

语言	LD	FBD	IL
指令格式			WSHL En, In, Sou, N, Num, Out

参数	参数定义	输入	输出	说明
En	使能	√		
In	移入字起始元件	√		占用 Num 个连续元件
Sou	源起始元件	√		占用 N 个连续元件
N	元件个数	√		
Num	移位次数	√		
Eno	使能输出		√	
Out	移出元件		√	占用 Num 个连续元件

【指令功能及作用说明】

1、WSHL 指令将以 Sou 起始的 N 个元件左移 Num 个字，移入以 In 起始的 Num 个元件，移出放在以 Out 起始的 Num 个元件。如下图所示：



- 2、1≤Num≤N，否则指令不执行。
- 3、WSHL 指令一般用边沿执行。

注：E/S 系列主机的 WSHL 指令参见“[WSHL](#)”。

【指令示例】

//Network 1 字左移



【程序说明】

当 M0=ON，把 V1000~V1005 左移 3 个字，移入 V0~V2，移出放在 V100~V102。

Sou 元件	初始值	左移结果
V1000	30	V1000=100
V1001	-150	V1001=200
V1002	25	V1002=300
V1003	8	V1003=30
V1004	95	V1004=-150
V1005	-20	V1005=25
V0	100	
V1	200	
V2	300	
V100		V100=8
V101		V101=95
V102		V102=-20

WSHR（字右移）

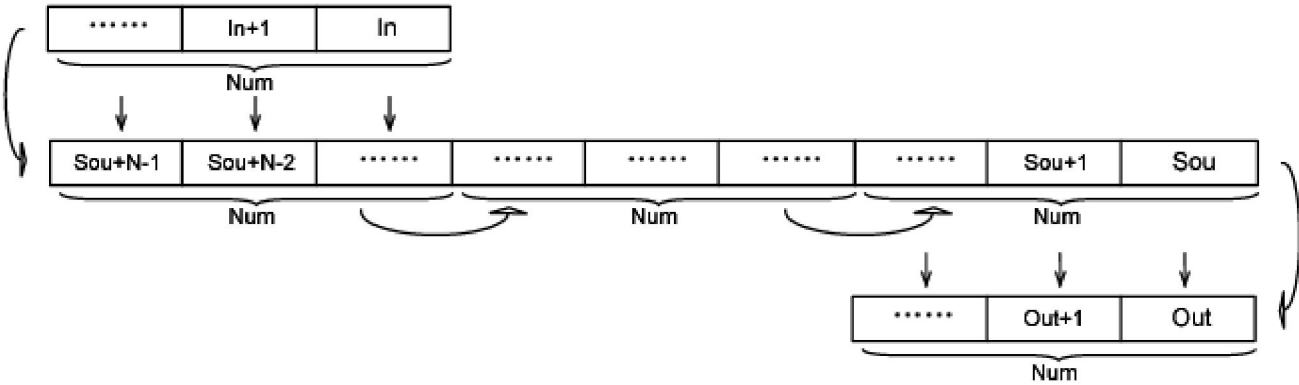
指令格式及参数说明

语言	LD	FBD	IL
指令格式			WSHR En, In, Sou, N, Num, Out

参数	参数定义	输入	输出	说明
En	使能	√		
In	移入字起始元件	√		占用 Num 个连续元件
Sou	源起始元件	√		占用 N 个连续元件
N	元件个数	√		
Num	移位次数	√		
Eno	使能输出		√	
Out	移出元件		√	占用 Num 个连续元件

【指令功能及作用说明】

1、WSHR 指令将 Sou 起始的 N 个元件右移 Num 个字，移入以 In 起始的 Num 个元件，移出放在以 Out 起始的 Num 个元件。如下图所示：



- 2、 $1 \leq \text{Num} \leq N$ ，否则指令不执行。
- 3、WSHR 指令一般用边沿执行。

注：E/S 系列主机的 WSHR 指令参见“[WSHR](#)”。

【指令示例】

//Network 1 字右移



【程序说明】

当 M0=ON，把 V1000~V1005 右移 3 个字，移入 V0~V2，移出放在 V100~V102。

Sou 元件	初始值	右移结果
V1000	30	V1000=8
V1001	-150	V1001=95
V1002	25	V1002=-20
V1003	8	V1003=100
V1004	95	V1004=200
V1005	-20	V1005=300
V0	100	
V1	200	
V2	300	
V100		V100=30
V101		V101=-150
V102		V102=25

ROL（位循环左移）

指令格式及参数说明

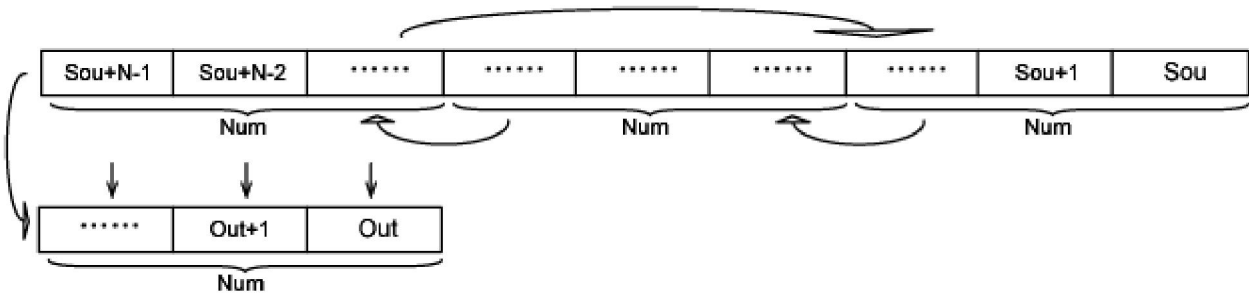
语言	LD	FBD	IL
----	----	-----	----

语言	LD	FBD	IL
指令格式	En Rou Eno Sou Out N Num		Rou En, Sou, N, Num, Out

参数	参数定义	输入	输出	说明
En	使能	√		
Sou	源起始元件	√		占用 N 个连续元件
N	元件个数	√		
Num	移位次数	√		
Eno	使能输出		√	
Out	移出元件		√	占用 Num 个连续元件

【指令功能及作用说明】

1、以 Num 位一组，ROL 指令将以 Sou 起始的 N 个元件循环左移 Num 位，移出放在以 Out 起始的 Num 个元件。如下图所示：



- 2、如果 Sou 是寄存器元件，那么就将以 Sou 起始的 N 个寄存器按位循环左移 Num 位。
- 3、1≤Num≤N，否则指令不执行。
- 4、ROL 指令一般用边沿执行。

注：E/S 系列主机的 ROL 指令参见“[ROL](#)”。

【指令示例】

//Network 1 位地址位循环左移



//Network 2 寄存器位循环左移



【程序说明】

1、当 M0=ON，把 M100~M105 循环左移 3 位，移出放在 Y0~Y2。

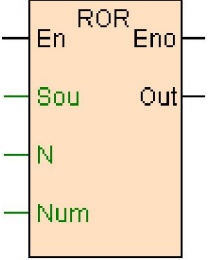
Sou 元件	初始值	左移结果
M100	ON	M100=ON
M101	OFF	M101=ON
M102	ON	M102=OFF
M103	ON	M103=ON
M104	ON	M104=OFF
M105	OFF	M105=ON
Y0		Y0=ON
Y1		Y1=ON
Y2		Y2=OFF

2、当 M1=ON，把 V1000~V1005 按位循环左移 3 位，移出放在 M200~M202。

Sou 元件	初始值	左移结果
V1000	00000000 00011110	V1000=00000000 11110111
V1001	11111111 01101010	V1001=11111011 01010000
V1002	00000000 00011001	V1002=00000000 11001111
V1003	00000000 00001000	V1003=00000000 01000000
V1004	00000000 01011111	V1004=00000010 11111000
V1005	11111111 11101100	V1005=11111111 01100000
M200		M200=ON
M201		M201=ON
M202		M202=ON

ROR（位循环右移）

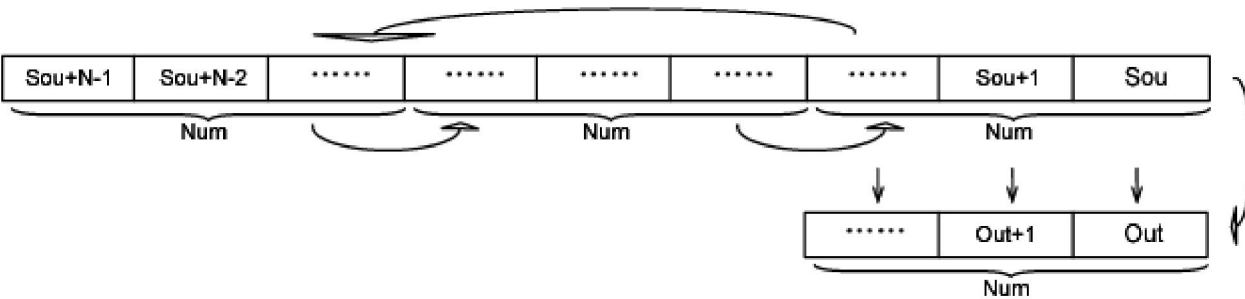
指令格式及参数说明

语言	LD	FBD	IL
指令格式			ROR En, Sou, N, Num, Out

参数	参数定义	输入	输出	说明
En	使能	√		
Sou	源起始元件	√		占用 N 个连续元件
N	元件个数	√		
Num	移位次数	√		
Eno	使能输出		√	
Out	移出元件		√	占用 Num 个连续元件

【指令功能及作用说明】

1、以 Num 位一组，ROR 指令将 Sou 起始的 N 个元件循环右移 Num 位，移出放在以 Out 起始的 Num 个元件。如下图所示：



- 2、如果 Sou 是寄存器元件，那么就将以 Sou 起始的 N 个寄存器按位循环右移 Num 位。
- 3、 $1 \leq \text{Num} \leq N$ ，否则指令不执行。
- 4、ROR 指令一般用边沿执行。

注：E/S 系列主机的 ROR 指令参见“[ROR](#)”。

【指令示例】

//Network 1 位地址位循环右移



//Network 2 寄存器位循环右移



【程序说明】

- 1、当 M0=ON，把 M100~M105 循环右移 3 位，移出放在 Y0~Y2。

Sou 元件	初始值	右移结果
M100	ON	M100=ON
M101	OFF	M101=ON
M102	OFF	M102=OFF
M103	ON	M103=ON
M104	ON	M104=OFF
M105	OFF	M105=OFF
Y0		Y0=ON
Y1		Y1=OFF
Y2		Y2=OFF

- 2、当 M1=ON，把 V1000~V1005 按位循环右移 3 位，移出放在 M200~M202。

Sou 元件	初始值	右移结果
V1000	00000000 00011110	V1000=01000000 00000011
V1001	11111111 01101010	V1001=00111111 11101101
V1002	00000000 00011001	V1002=00000000 00000011
V1003	00000000 00001000	V1003=11100000 00000001
V1004	00000000 01011111	V1004=10000000 00001011

V1005	11111111 11101100	V1005=11011111 11111101
M200		M200=OFF
M201		M201=ON
M202		M202=ON

WROL（字循环左移）

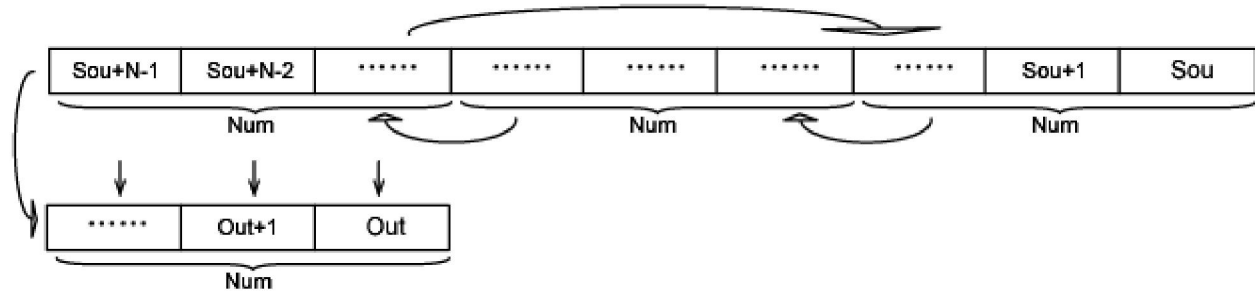
指令格式及参数说明

语言	LD	FBD	IL
指令格式			WROL En, Sou, N, Num, Out

参数	参数定义	输入	输出	说明
En	使能	√		
Sou	源起始元件	√		占用 N 个连续元件
N	元件个数	√		
Num	移位次数	√		
Eno	使能输出		√	
Out	移出元件		√	占用 Num 个连续元件

【指令功能及作用说明】

1、WROL 指令将以 Sou 起始的 N 个元件循环左移 Num 个字，移出放在以 Out 起始的 Num 个元件。如下图所示：



- 2、 $1 \leq \text{Num} \leq N$ ，否则指令不执行。
- 3、WROL 指令一般用边沿执行。

注：E/S 系列主机的 WROL 指令参见“[WROL](#)”。

【指令示例】

//Network 1 字循环左移



【程序说明】

当 M0=ON，把 V1000~V1005 循环左移 3 个字，移出放在 V100~V102。

Sou 元件	初始值	左移结果
V1000	30	V1000=8
V1001	-150	V1001=95
V1002	25	V1002=-20
V1003	8	V1003=30
V1004	95	V1004=-150
V1005	-20	V1005=25
V100		V100=8
V101		V101=95
V102		V102=-20

WROR（字循环右移）

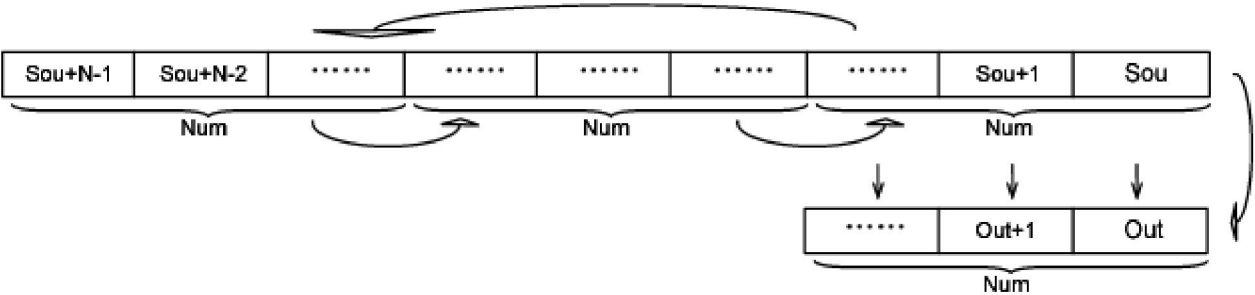
指令格式及参数说明

语言	LD	FBD	IL
指令格式			WROR En, Sou, N, Num, Out

参数	参数定义	输入	输出	说明
En	使能	√		
Sou	源起始元件	√		占用 N 个连续元件
N	元件个数	√		
Num	移位次数	√		
Eno	使能输出		√	
Out	移出元件		√	占用 Num 个连续元件

【指令功能及作用说明】

1、WROR 指令将 Sou 起始的 N 个元件循环右移 Num 个字，移出放在以 Out 起始的 Num 个元件。如下图所示：



- 2、1≤Num≤N，否则指令不执行。
- 3、WROR 指令一般用边沿执行。

注：E/S 系列主机的 WROR 指令参见“[WROR](#)”。

【指令示例】

//Network 1 字循环右移



【程序说明】

当 M0=ON，把 V1000~V1005 循环右移 3 个字，移出放在 V100~V102。

Sou 元件	初始值	右移结果
V1000	30	V1000=8
V1001	-150	V1001=95
V1002	25	V1002=-20
V1003	8	V1003=30
V1004	95	V1004=-150
V1005	-20	V1005=25
V100		V100=30
V101		V101=-150
V102		V102=25

BSHL（字节左移）

指令格式及参数说明

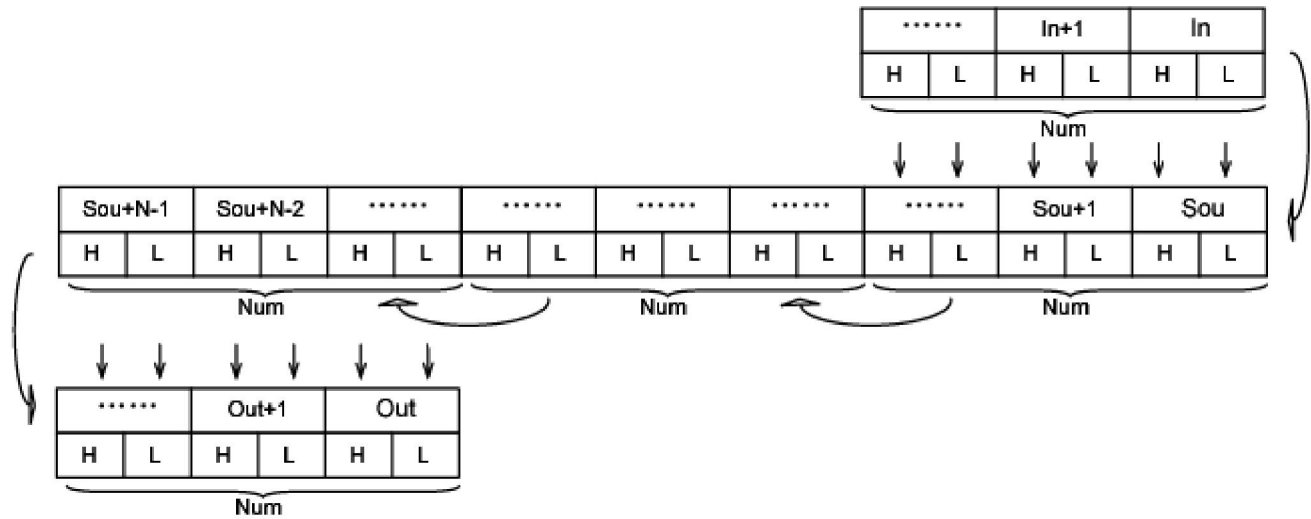
语言	LD	FBD	IL
指令格式			BSHL En, In, Sou, N, Num, Out

参数	参数定义	输入	输出	说明
En	使能	√		
In	移入字节起始元件	√		占用(Num-1)\2+1 个连续元件
Sou	源起始元件	√		占用 N 个连续元件
N	元件个数	√		
Num	移位次数	√		
Eno	使能输出		√	
Out	移出元件		√	占用(Num-1)\2+1 个连续元件

【指令功能及作用说明】

1、BSHL 指令将以 Sou 起始的 N 个元件左移 Num 个字节，移入以 In 起始的 Num 个字节，移出放在以 Out 起始的 Num 个字节。

如下图所示：



- 2、 $1 \leq \text{Num} \leq 2 \times N$ ，否则指令不执行。
- 3、BSHL 指令一般用边沿执行。

注：E/S 系列主机的 BSHL 指令参见“[BSHL](#)”。

【指令示例】

//Network 1 字节左移



【程序说明】

当 M0=ON，把 V1000~V1005 左移 3 个字节，移入 V0 及 V1 的低字节，移出放在 V100 及 V101 的低字节。

Sou 元件	初始值	左移结果
V1000	30 (0x001E)	V1000=100 (0x1234)
V1001	-150 (0xFF6A)	V1001=200 (0x1E78)
V1002	25 (0x0019)	V1002=300 (0x6A00)
V1003	8 (0x0008)	V1003=30 (0x19FF)
V1004	95 (0x005F)	V1004=-150 (0x0800)
V1005	-20 (0xFFEC)	V1005=25 (0x5F00)
V0	4660 (0x1234)	
V1	22136 (0x5678)	
V100		V100=8 (0xEC00)
V101		V101=95 (0x00FF)

BSHR（字节右移）

指令格式及参数说明

语言	LD	FBD	IL
----	----	-----	----

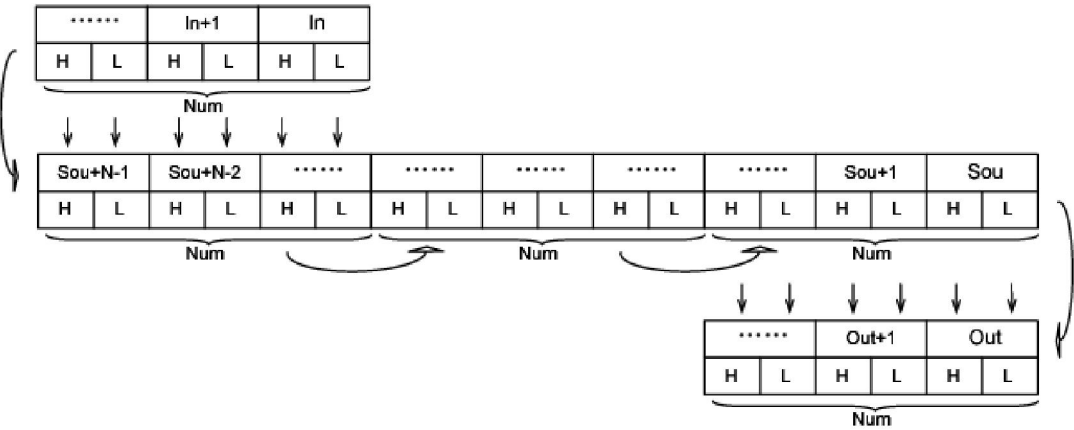
语言	LD	FBD	IL
指令格式	BSHR EnEno InOut Sou N Num		BSHR En, In, Sou, N, Num, Out

参数	参数定义	输入	输出	说明
En	使能	√		
In	移入字节起始元件	√		占用(Num-1)\2+1 个连续元件
Sou	源起始元件	√		占用 N 个连续元件
N	元件个数	√		
Num	移位次数	√		
Eno	使能输出		√	
Out	移出元件		√	占用(Num-1)\2+1 个连续元件

【指令功能及作用说明】

1、BSHR 指令将 Sou 起始的 N 个元件右移 Num 个字节，移入以 In 起始的 Num 个字节，移出放在以 Out 起始的 Num 个字节。如

下图所示：



2、 $1 \leq \text{Num} \leq 2 \times N$ ，否则指令不执行。

3、BSHR 指令一般用边沿执行。

注：E/S 系列主机的 BSHR 指令参见“[BSHR](#)”。

【指令示例】

//Network 1 字节右移



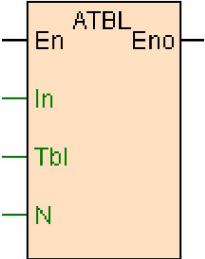
【程序说明】

当 M0=ON，把 V1000~V1005 右移 3 个字节，移入 V0 及 V1 的低字节，移出放在 V100 及 V101 的低字节。

Sou 元件	初始值	右移结果
V1000	30 (0x001E)	V1000=100 (0x19FF)
V1001	-150 (0xFF6A)	V1001=200 (0x0800)
V1002	25 (0x0019)	V1002=300 (0x5F00)
V1003	8 (0x0008)	V1003=30 (0xEC00)
V1004	95 (0x005F)	V1004=-150 (0x34FF)
V1005	-20 (0xFFEC)	V1005=25 (0x7812)
V0	4660 (0x1234)	
V1	22136 (0x5678)	
V100		V100=8 (0x001E)
V101		V101=95 (0x006A)

ATBL（添加到队列）

指令格式及参数说明

语言	LD	FBD	IL
指令格式			ATBL En, In, Tbl, N

参数	参数定义	输入	输出	说明
En	使能	√		
In	输入数据	√		
Tbl	队列起始元件	√		占用 N+1 个连续元件
N	队列长度	√		
Eno	使能输出		√	

【指令功能及作用说明】

- 1、添加到队列指令 ATBL 是用于将位状态或寄存器值按顺序存放放到 Tbl 指定 的数据队列。
- 2、En 为指令的使能端，一般使用边沿型（上升沿或下降沿）信号。当 En 状态每跳变 ON 一次，将 In 添加到 Tbl 指定的数据队列中，队列元素个数加 1（Tbl 寄存器的内容加 1）。
- 3、Tbl 定义存放数据队列的起始 元件，N 为队列长度，其中队列的首个寄存器（Tbl）为队列元素的个数，Tbl+1 至 Tbl+N 共 N 个寄存器用于存放队列数据。所以，若 In 为位元件时，可存放的队列元素最大为 N*16；若 In 为寄存器时，可存放的队列元素最大为 N。当队列元素个数超过队列元素最大值时，数据不再进入队列。
- 4、队列的存放顺序如下：
- A、位 元件数据：若输入到队列的数据是位，存放顺序如下表所示：

队列元件	队列内容
Tbl	队列元素个数

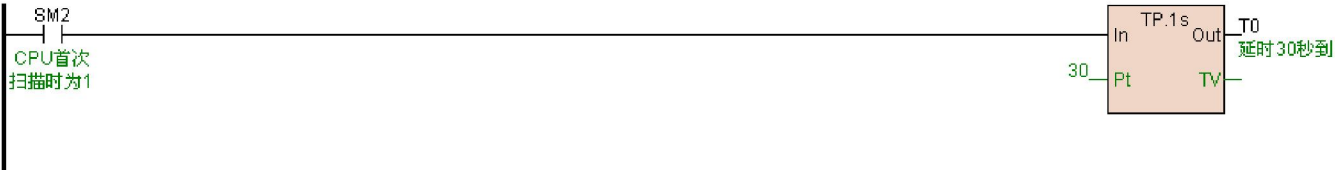
Tbl+1	b0	第 1 个队列元素
	b1	第 2 个队列元素
		
	b15	第 16 个队列元素
Tbl+2	b0	第 17 个队列元素
	b1	第 18 个队列元素
		
.....	b0	
		

B、寄存器数据：若输入到队列的数据是 16 位寄存器，存放顺序如下表所示：

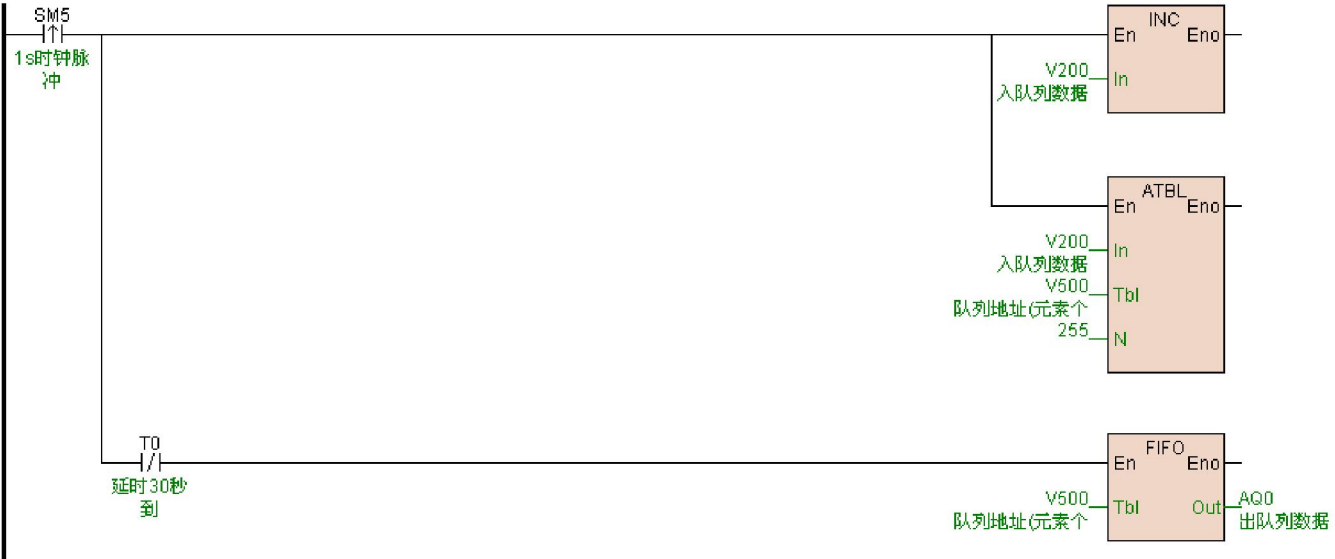
队列元件	队列内容
Tbl	队列元素个数
Tbl+1	第 1 个队列元素
Tbl+2	第 2 个队列元素
Tbl+3	第 3 个队列元素
.....	

【指令示例】

//Network 1 启动延时计时



//Network 2 每秒计算并且进入队列，T0延时到后，以先进先出方式出队列



【程序说明】

- 1、首次扫描 SM2=ON，启动 T0 计时器（30 秒）。

- 2、SM5 为每秒时钟脉冲，以每秒周期，INC 指令将 V200 加 1（用于模拟数据），ATBL 指令将 V200 进入到以 V500 起始的队列中，队列长度 255。
- 3、在 30 秒过后，T0=OFF，每秒 FIFO 指令以先进先出从队列中取数据，输出到 AQ0。

FIFO（先进先出）

指令格式及参数说明

语言	LD	FBD	IL
指令格式			FIFO En, Tbl, Out

参数	参数定义	输入	输出	说明
En	使能	√		
Tbl	队列起始元件	√		
Eno	使能输出		√	
Out	数据输出		√	

【指令功能及作用说明】

- 1、FIFO 指令以先进先出方式从队列 ATBL 取出数据，每出一个数据队列元素减 1。
- 2、如果队列的元素个数≤0，则指令不执行。
- 3、FIFO 指令配合 ATBL 指令现成先进先出队列，FIFO 指令一般边沿执行。

【指令示例】

参见 [ATBL](#) 指令。

LIFO（后进先出）

指令格式及参数说明

语言	LD	FBD	IL
指令格式			LIFO En, Tbl, Out

参数	参数定义	输入	输出	说明
En	使能	√		
Tbl	队列起始元件	√		
Eno	使能输出		√	
Out	数据输出		√	

【指令功能及作用说明】

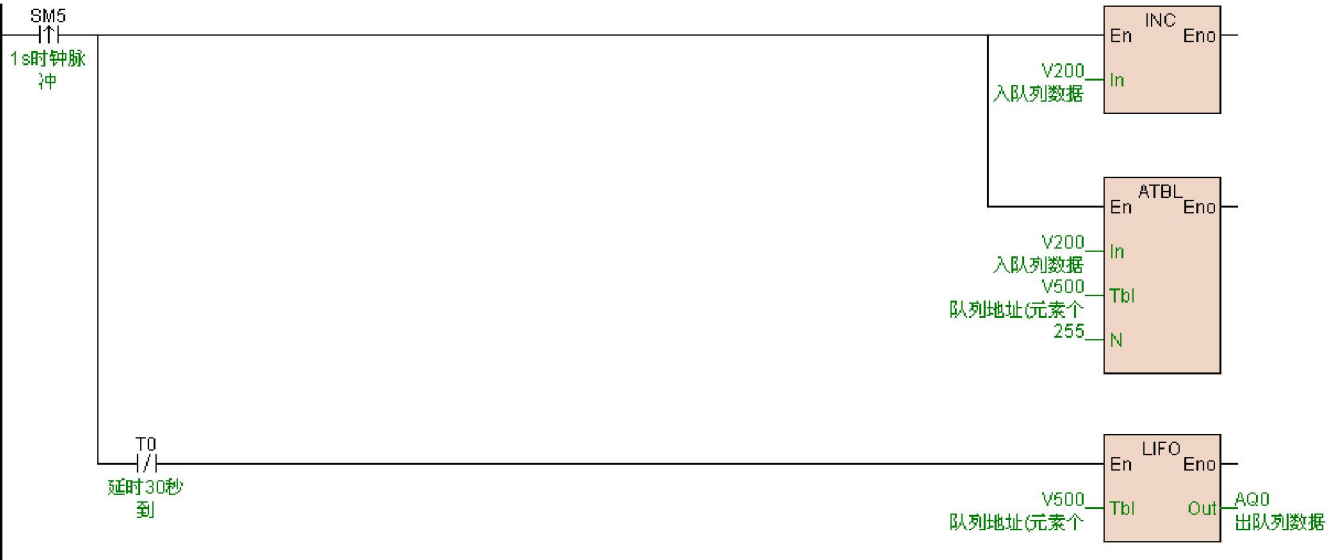
- 1、LIFO 指令以后进先出方式从队列 [ATBL](#) 取出数据，每出一个数据队列元素减 1。
- 2、如果队列的元素个数≤0，则指令不执行。
- 3、LIFO 指令配合 ATBL 指令现成后进先出队列（堆栈），LIFO 指令一般边沿执行。

【指令示例】

//Network 1 启动延时计时



//Network 2 每秒计算并且进入队列，T0延时到后，以后进先出方式出队列



【程序说明】

- 1、首次扫描 SM2=ON，启动 T0 计时器（30 秒）。
- 2、SM5 为每秒时钟脉冲，以每秒周期，INC 指令将 V200 加 1（用于模拟数据），ATBL 指令将 V200 进入到以 V500 起始的队列中，队列长度 255。
- 3、在 30 秒过后，T0=OFF，每秒 LIFO 指令以后进先出从队列中取数据，输出到 AQ0。

SORT（数据排序）

指令格式及参数说明

语言	LD	FBD	IL
指令格式			<code>SORT En, UpDown, Sou, Row, Col, Index, Out</code>

参数	参数定义	输入	输出	说明
En	使能	√		
UpDown	升降控制	√		

参数	参数定义	输入	输出	说明
Sou	源起始元件	√		占用 Row * Col 个连续元件
Row	行	√		
Col	列	√		
Index	排序列	√		
Eno	使能输出		√	
Out	输出		√	占用 Row * Col 个连续元件

【指令功能及作用说明】

- 1、将以 Sou 起始的 Row 行 Col 列共 Row×Col 个元件中的数据进行排序，排序以 Index 指定的列作为参考，排序方向由 UpDown 控制，UpDown 为 OFF 则升序排列，UpDown 为 ON 则降序排列，排序后的数据存入以 Out 为首的共 Row×Col 个元件中。
- 2、SORT 指令是边沿执行，如果排序后修改了 Sou 数据，那么需要重新触发。
- 3、必须满足 Row≥1、Col≥1、Index≤Col，否则指令不执行。

注：E/S 系列主机不支持 SORT 指令。

【指令示例】

//Network 1



【程序说明】

假如初始数据（成绩表）如下：

姓名	语文	数学	英语
吴一	V1000=98	V1001=65	V1002=81
陈皮	V1003=78	V1004=89	V1005=65
王二	V1006=87	V1007=99	V1008=68
李四	V1009=60	V1010=92	V1011=83
张三	V1012=72	V1013=90	V1014=56

如果 M100=OFF，当 M0=ON，把 V1000 起始的 5 行 3 列数组，以第 2 列（数学成绩）按照升序排序，结果放在 V0 起始的 5x3 个元件中。

姓名	语文	数学	英语
吴一	V0=98	V1=65	V2=81
陈皮	V3=78	V4=89	V5=65
张三	V12=72	V13=90	V14=56
李四	V9=60	V10=92	V11=83
王二	V6=87	V7=99	V8=68

如果 M100=ON，当 M0=ON，把 V1000 起始的 5 行 3 列数组，以第 2 列（数学成绩）按照降序排序，结果放在 V0 起始的 5x3 个元

件中。

姓名	语文	数学	英语
王二	V6=87	V7=99	V8=68
李四	V9=60	V10=92	V11=83
张三	V12=72	V13=90	V14=56
陈皮	V3=78	V4=89	V5=65
吴一	V0=98	V1=65	V2=81

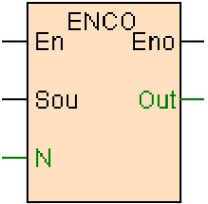
4.1Q 数据转换指令

数据转换指令列表

指令名称	8 位模式	32 位模式	指令功能	支持语言		
				LD	FBD	IL
ENCO			编码器	√	√	√
DECO			译码器	√	√	√
BTOW			位转换为字	√	√	√
WTOB			字转换为位	√	√	√
HEX	HEX.LB		ASCII 转换为 16 进制	√	√	√
ASCI	ASCI.LB		16 进制转换为 ASCII	√	√	√
BUNB			离散位组合到连续位	√	√	√
BUNW			离散位组合到连续字	√	√	√
WUNW			离散字组合到连续字	√	√	√
BDIB			连续位分散到离散位	√	√	√
WDIB			连续字分散到离散位	√	√	√
WDIW			连续字分散到离散字	√	√	√
BCD		D.BCD	BIN 转换为 BCD	√	√	√
BIN		D.BIN	BCD 转换为 BIN	√	√	√
ITOL			整数转换为长整数	√	√	√
GRAY			BIN 转换为 GRAY 码	√	√	√
GBIN			GRAY 码转换为 BIN	√	√	√

ENCO（编码器）

指令格式及参数说明

语言	LD	FBD	IL
指令格式			ENCO En, Sou, N, Out

参数	参数定义	输入	输出	说明
En	使能	√		
Sou	源起始元件	√		
N	编码位数	√		
Eno	使能输出		√	

参数	参数定义	输入	输出	说明
Out	编码输出		√	

【指令功能及作用说明】

- 1、ENCO 指令用于求出在 Sou 数据中 最大 ON 位的位置。
- 2、 $0 \leq N \leq 8$ ，最大可以编码 $2^8=256$ 位。
- 3、若数据来源有多位为 1（ON）时，则只处理最高位。

注：E/S 系列主机不支持 ENCO 指令。

【指令示例】

//Network 1



【程序说明】

- 1、M0=ON，对 X0~X7（ $2^3=8$ ）进行编码，假如 X6=ON，X7=OFF 其他无所谓，那么 V0=7。
- 2、M0=ON，对 V100 的低 8 位（ $2^3=8$ ，高 8 位不使用）进行编码，假如 V100=2345（00001001 00101001），那么 V10=6。

DECO（译码器）

指令格式及参数说明

语言	LD	FBD	IL
指令格式			DECO En, In, N, Out

参数	参数定义	输入	输出	说明
En	使能	√		
In	译码输入	√		
N	译码位数	√		
Eno	使能输出		√	
Out	译码输出		√	

【指令功能及作用说明】

- 1、DECO 指令对 In 数据的 N 个位数进行译码，结果输出到 Out。
- 2、 $0 \leq N \leq 8$ ，最大可以译码输出 $2^8=256$ 位。

注：E/S 系列主机不支持 DECO 指令。

【指令示例】

//Network 1



【程序说明】

M0=ON，将 V0 译码输出到 Y0~Y7 ($2^3=8$) 及 V100 的低 8 位 ($2^3=8$ ，高 8 位全部为 0)。假如 V0=7，那么 Y0~Y7 中仅 Y6=ON 其他 OFF，V100=64 (00000000 01000000)。

BTOW（位到字）

指令格式及参数说明

语言	LD	FBD	IL
指令格式			BTOW En, Sou, N, Out

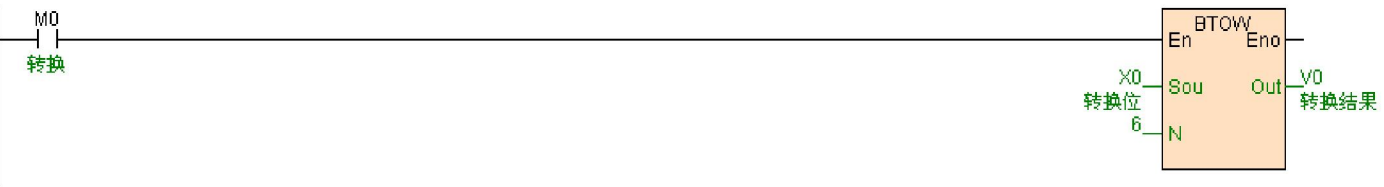
参数	参数定义	输入	输出	说明
En	使能	√		
Sou	源起始元件	√		占用 N 个连续元件
N	转换位数	√		1-32
Eno	使能输出		√	
Out	输出		√	N>16 时占用 2 个寄存器

【指令功能及作用说明】

BTOW 指令将 Sou 起始的 N 个位元件，转换为整数结果输出到 Out， $1\leq N\leq 32$ 。

【指令示例】

//Network 1

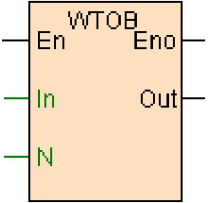


【程序说明】

当 M0=ON，将 X0~X5 转换为数值，如果 X1=ON、X2=ON、X5=ON 其他为 OFF，那么 V0=38 (00000000 00100110)。

WTOB（字到位）

指令格式及参数说明

语言	LD	FBD	IL
指令格式			WTOB En, In, N, Out

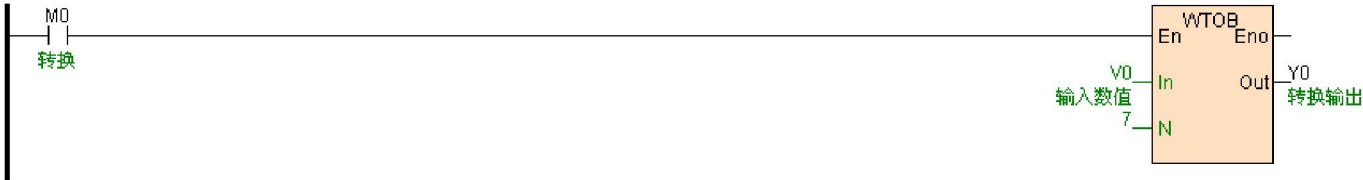
参数	参数定义	输入	输出	说明
En	使能	√		
In	输入	√		N>16 时占用 2 个寄存器
N	转换位数	√		1-32
Eno	使能输出		√	
Out	转换结果起始元件		√	占用 N 个连续元件

【指令功能及作用说明】

WTOB 指令将 In 的低 N 个 位转换输出到 Out，1≤N≤32。

【指令示例】

//Network 1

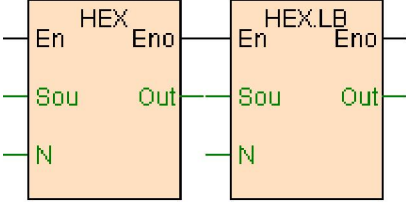


【程序说明】

当 M0=ON，将 V0 的低 7 位转换输出到 Y0~Y6，如果 V0=38（00000000 00100110），那么 Y1=ON、Y2=ON、Y5=ON 其他为 OFF。

HEX、HEX.LB（ASCII 转换为 HEX）

指令格式及参数说明

语言	LD	FBD	IL
16、8 位 指令格式			HEX En, Sou, N, Out HEX.LB En, Sou, N, Out

参数	参数定义	输入	输出	说明
En	使能	√		
Sou	源起始元件	√		HEX 占用(N-1)\2+1 个连续元件，HEX.LB 占用 N 个连续元件
N	转换字符数	√		
Eno	使能输出		√	
Out	输出		√	占用(N-1)\4+1 个连续元件

【指令功能及作用说明】

- 1、HEX 指令将以 Sou 起始的 ASCII 码转换为数值，转换字符数为 N。
- 2、8 位模式指令 HEX.LB 只转换 Sou 的低字节，高字节不使用。
- 3、ASCII 码字符只能是 0~9 和 A、B、C、D、E、F 这 6 个字符，如果 Sou 含有非法字符则指令不执行。

注：E/S 系列主机的 HEX 指令参见“[HEX](#)”。

【指令示例】

//Network 1 ASCII转换为数值



【程序说明】

假如初始数据（ASCII 码数据）如下：

元件	寄存器值（ASCII 码）	高低字节值（ASCII 码）	ASCII 字符
V1000	0x3938	低字节 0x38	"8"
		高字节 0x39	"9"
V1001	0x4241	低字节 0x41	"A"
		高字节 0x42	"B"
V1002	0x3534	低字节 0x34	"4"
		高字节 0x35	"5"
V1003	0x3332	低字节 0x32	"2"
		高字节 0x33	"3"
V1004	0x4645	低字节 0x45	"E"
		高字节 0x46	"F"
V1005	0x3039	低字节 0x39	"9"
		高字节 0x30	"0"
V1006	0x3831	低字节 0x31	"1"
		高字节 0x38	"8"
V1007	0x4443	低字节 0x43	"C"
		高字节 0x44	"D"

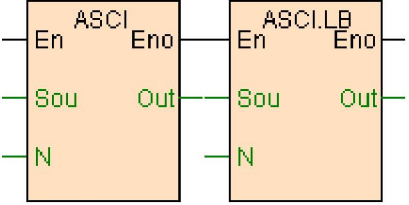
当 M0=ON，把 V1000 起始的 ASCII 码转换为数值，HEX 转换结果放在 V0 起始的元件中，HEX.LB 只转换 V1000 起始的 ASCII 码的低字节，转换结果放在 V10 起始的元件中，对 N=1~8 转换结果如下表。

N	HEX		HEX.LB	
	V1	V0	V11	V10
1		0x8		0x8
2		0x89		0x8A
3		0x89A		0x8A4
4		0x89AB		0x8A42

5	0x8	0x9AB4	0x8	0xA42E
6	0x89	0xAB45	0x8A	0x42E9
7	0x89A	0xB452	0x8A4	0x2E91
8	0x89AB	0x4523	0x8A42	0xE91C

ASCI、ASCI.LB（HEX 转换为 ASCII）

指令格式及参数说明

语言	LD	FBD	IL
16、8 位 指令格式			ASCI En, Sou, N, Out ASCI.LB En, Sou, N, Out

参数	参数定义	输入	输出	说明
En	使能	√		
Sou	源起始元件	√		占用(N-1)\4+1 个连续元件
N	转换字符数	√		
Eno	使能输出		√	
Out	输出		√	ASCI 占用(N-1)\2+1 个连续元件，ASCI.LB 占用 N 个连续元件

【指令功能及作用说明】

- 1、ASCI 指令将以 Sou 起始的数值转换为 ASCII 码字符，转换字符数为 N，转换结果存储到以 Out 起始的元件。
- 2、8 位模式指令 ASCII.LB 将转换结果只存储到 Out 的低字节，高字节为 0。

注：E/S 系列主机的 ASCII 指令参见“[ASCII](#)”。

【指令示例】

//Network 2 数值转换为ASCII



【程序说明】

假如初始数据（转换为 ASCII 码）如下：

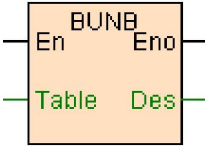
元件	寄存器值
V1100	0x4523
V1101	0x89AB

当 M1=ON，ASCI 指令把 V1100 起始的数值转换为 ASCII 码，结果放在 V100 起始的元件中，ASCI.LB 指令把结果放在 V200 起始的元件中（只存储低字节，高字节为 0），在 N=1~8 时转换结果如下表。

指令格式	Out 元件	N	1	2	3	4	5	6	7	8
ASCII	V100	低字节	"3"	"2"	"5"	"4"	"B"	"A"	"9"	"8"
		高字节		"3"	"2"	"5"	"4"	"B"	"A"	"9"
	V101	低字节			"3"	"2"	"5"	"4"	"B"	"A"
		高字节				"3"	"2"	"5"	"4"	"B"
	V102	低字节					"3"	"2"	"5"	"4"
		高字节						"3"	"2"	"5"
	V103	低字节							"3"	"2"
		高字节								"3"
ASCII.LB	V200	低字节	"3"	"2"	"5"	"4"	"B"	"A"	"9"	"8"
	V201	低字节		"3"	"2"	"5"	"4"	"B"	"A"	"9"
	V202	低字节			"3"	"2"	"5"	"4"	"B"	"A"
	V203	低字节				"3"	"2"	"5"	"4"	"B"
	V204	低字节					"3"	"2"	"5"	"4"
	V205	低字节						"3"	"2"	"5"
	V206	低字节							"3"	"2"
	V207	低字节								"3"

BUNB（离散位组合到连续位）

指令格式及参数说明

语言	LD	FBD	IL
指令格式			BUNB En, Table, Des

参数	参数定义	输入	输出	说明
En	使能	√		
Table	离散位元件表	√		
Eno	使能输出		√	
Des	目标起始元件		√	

【指令功能及作用说明】

BUNB 指令用于将 Table 定义的离散位 元件组合到连续的位元件中。

离散位 元件表：可作为 BUNB、BUNW、BDIB、WDIB 指令的 Table 端调用。离散位元件表的定义请参考编程 编写的“[指令使用表格](#)”部分。

【指令示例】

//Network 1 离散位组合到连续位



【程序说明】

假如“读离散位表”定义如下：

序号	位元件
1	X3
2	M10
3	M301
4	S21
5	M77
6	M100
7	X1
8	Y6

当 M0=ON，BUNB 指令把“读离散位表”定义的位组合移动到以 M500 起始的连续元件。

序号	执行结果
1	M500 = X3
2	M501 = M10
3	M502 = M301
4	M503 = S21
5	M504 = M77
6	M505 = M100
7	M506 = X1
8	M507 = Y6

BUNW（离散位组合到连续字）

指令格式及参数说明

语言	LD	FBD	IL
指令格式			BUNW En, Table, Des

参数	参数定义	输入	输出	说明
En	使能	√		
Table	离散位元件表	√		
Eno	使能输出		√	
Des	目标起始元件		√	

【指令功能及作用说明】

BUNW 指令用于将 Table 定义的离散位 元件按位组合到连续的寄存器中。

离散位 元件表：可作为 BUNB、BUNW、BDIB、WDIB 指令的 Table 端调用。离散位元件表的定义请参考编程 编写的[“指令使用表格”](#)部分。

【指令示例】

//Network 2 离散位组合到连续字



【程序说明】

假如“读离散位表”定义如下：

序号	位元件
1	X3
2	M10
3	M301
4	S21
5	M77
6	M100
7	X1
8	Y6

当 M1=ON，BUNW 指令把“读离散位表”定义的位按位组合到以 V0 起始的寄存器中。

序号	执行结果
1	V0 的 b0 = X3
2	V0 的 b1 = M10
3	V0 的 b2 = M301
4	V0 的 b3 = S21
5	V0 的 b4 = M77
6	V0 的 b5 = M100
7	V0 的 b6 = X1
8	V0 的 b7 = Y6

WUNW（离散字组合到连续字）

指令格式及参数说明

语言	LD	FBD	IL
指令格式			WUNW En, Table, Des

参数	参数定义	输入	输出	说明
En	使能	√		
Table	离散寄存器元件表	√		
Eno	使能输出		√	
Des	目标起始元件		√	

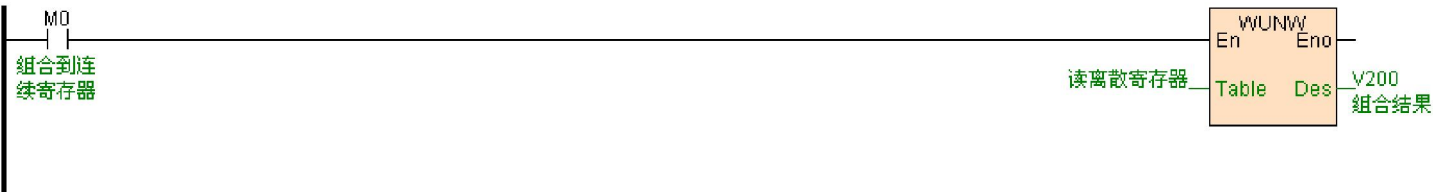
【指令功能及作用说明】

WUNW 指令用于将 Table 定义的离散寄存器组合到连续的寄存器中。

离散寄存器元件表：可作为 WUNW、WDIW 指令的 Table 端调用。离散寄存器元件表的定义请参考编程编写的“[指令使用表格](#)”部分。

【指令示例】

//Network 1 离散寄存器组合到连续寄存器



【程序说明】

假如“读离散寄存器表”定义如下：

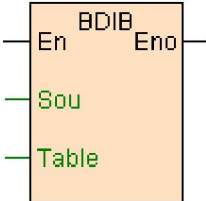
序号	位元件
1	AI1
2	V10
3	V106
4	AQ0

当 M0=ON，WUNW 指令把“读离散寄存器表”定义的寄存器组合移动到以 V200 起始的连续元件。

序号	执行结果
1	V200 = AI1
2	V201 = V10
3	V202 = V106
4	V203 = AQ0

BDIB（连续位分散到离散位）

指令格式及参数说明

语言	LD	FBD	IL
指令格式			BDIB En, Sou, Table

参数	参数定义	输入	输出	说明
En	使能	√		
Sou	源起始元件	√		
Table	离散位元件表	√		
Eno	使能输出		√	

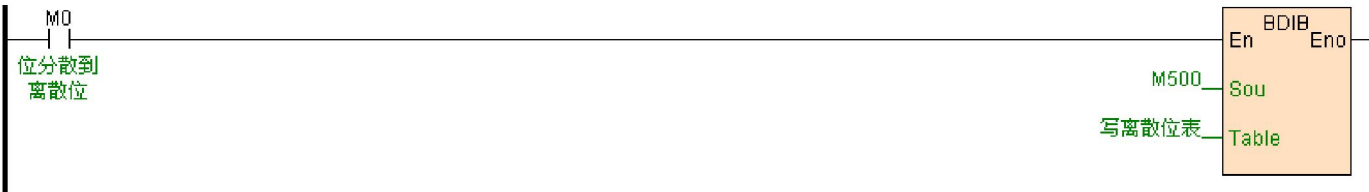
【指令功能及作用说明】

BDIB 指令用于将以 Sou 起始的位 元件分散到 Table 定义的离散位元件中。

离散位 元件表：可作为 BUNB、BUNW、BDIB、WDIB 指令的 Table 端调用。离散位元件表的定义请参考编程 编写的“[指令使用表格](#)”部分。

【指令示例】

//Network 1 连续位分散到离散位



【程序说明】

假如“写离散位表”定义如下：

序号	位元件
1	Y4
2	M10

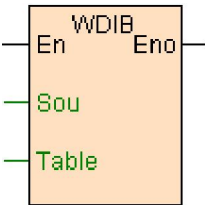
3	M301
4	S21
5	M77
6	M100
7	Y0
8	Y6

当 M0=ON，BDIB 指令把以 M500 起始的连续位“写离散位表”定义的位组合移动到。

序号	执行结果
1	Y4 = M500
2	M10 = M501
3	M301 = M502
4	S21= M503
5	M77= M504
6	M100 = M505
7	Y0 = M506
8	Y6 = M507

WDIB（连续字分散到离散位）

指令格式及参数说明

语言	LD	FBD	IL
指令格式			WDIB En, Sou, Table

参数	参数定义	输入	输出	说明
En	使能	√		
Sou	源起始元件	√		
Table	离散位元件表	√		
Eno	使能输出		√	

【指令功能及作用说明】

WDIB 指令用于将以 Sou 起始的寄存器按位分散到 Table 定义的离散位元件中。

离散位 元件表：可作为 BUNB、BUNW、BDIB、WDIB 指令的 Table 端调用。离散位元件表的定义请参考编程 编写的“[指令使用表格](#)”部分。

【指令示例】

//Network 1 字分散到离散位



【程序说明】

假如“写离散位表”定义如下：

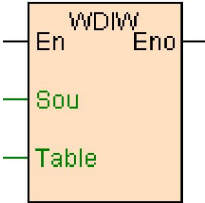
序号	位元件
1	Y4
2	M10
3	M301
4	S21
5	M77
6	M100
7	Y0
8	Y6

当 M1=ON，WDIB 指令把以 V0 起始的寄存器按位分散到“读离散位表”定义的位元件中。

序号	执行结果
1	Y4 = V0 的 b0
2	M10 = V0 的 b1
3	M301 = V0 的 b2
4	S21= V0 的 b3
5	M77= V0 的 b4
6	M100 = V0 的 b5
7	Y0 = V0 的 b6
8	Y6 = V0 的 b7

WDIW（连续字分散到离散字）

指令格式及参数说明

语言	LD	FBD	IL
指令格式			WDIW En, Sou, Table

参数	参数定义	输入	输出	说明
En	使能	√		
Sou	源起始元件	√		
Table	离散寄存器元件表	√		
Eno	使能输出		√	

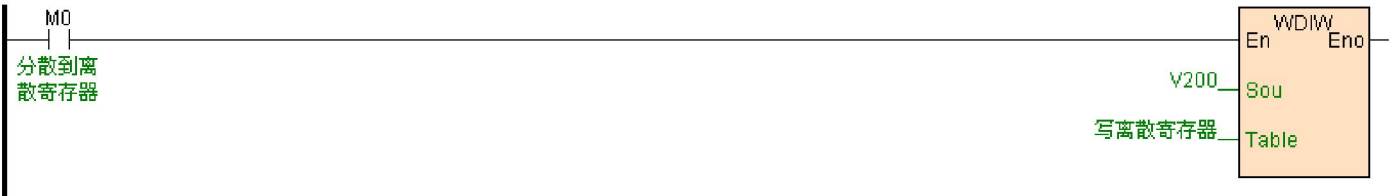
【指令功能及作用说明】

WDIW 指令用于将以 Sou 起始的寄存器分散到 Table 定义的离散寄存器中。

离散寄存器元件表：可作为 WUNW、WDIW 指令的 Table 端调用。离散寄存器元件表的定义请参考编程编写的[“指令使用表格”](#)部分。

【指令示例】

//Network 1 连续寄存器分散到离散寄存器



【程序说明】

假如“写离散寄存器表”定义如下：

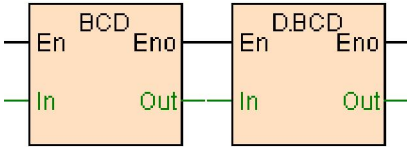
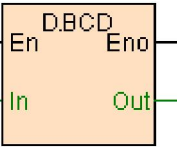
序号	位元件
1	V90
2	V10
3	V106
4	AQ0

当 M0=ON，WDIW 指令把以 V200 起始的寄存器分散到“读离散寄存器表”定义的离散寄存器。

序号	执行结果
1	V90 = V200
2	V10 = V201
3	V106 = V202
4	AQ0 = V203

BCD、D.BCD（BIN 转换为 BCD）

指令格式及参数说明

语言	LD	FBD	IL
16,32 位 指令格式			BCD En, In, Out D.BCD En, In, Out

参数	参数定义	输入	输出	说明
En	使能	√		
In	输入	√		
Eno	使能输出		√	
Out	输出		√	

【指令功能及作用说明】

- 1、BCD 指令（D.BCD 为 32 位指令）用于将数值转换为 BCD 码。
- 2、16 位指令数值输入范围为 0~9999，32 位指令数值输入范围为 0~99999999，In 超过范围则指令不执行。

【指令示例】

//Network 1 数值转换为BCD



【程序说明】

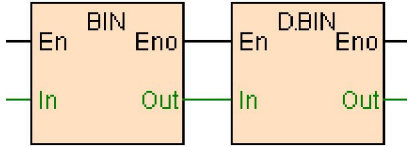
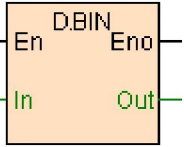
当 M0=ON,BCD 指令把 V1000 的值转换为 BCD 码结果放在 V0,D.BCD 指令把 V1001V1002 的值转换为 BCD 码结果放在 V10V11,

如下表。

In 元件	初始值	BCD 转换结果	D.BCD 转换结果
V1000	2345	V0 = 0x2345	
V1001V1002	48702861		V10V11 = 0x48702861

BIN、D.BIN（BCD 转换为 BIN）

指令格式及参数说明

语言	LD	FBD	IL
16、32 位 指令格式			BIN En, In, Out D.BIN En, In, Out

参数	参数定义	输入	输出	说明
En	使能	√		
In	输入	√		
Eno	使能输出		√	
Out	输出		√	

【指令功能及作用说明】

- 1、BIN 指令（D.BIN 为 32 位指令）用于把 BCD 码转换为数值。
- 2、BCD 码（Binary-Coded Decimal）亦称二进制十进数、二-十进制代码或 8421 码，就是将十进制的数以 8421 的形式展开成二进制。BCD 码是四位二进制码，也就是将十进制的数字转化为二进制，但是和普通的转化有一点不同，每一个十进制的数字 0-9 都对应着一个四位的二进制码，对应关系如下：十进制 0 对应 0000；十进制 1 对应 0001 十进制 9 对应 1001，接下来的 10 就有两个上述的码来表示，十进制 10 表示为 00010000，也就是 BCD 码是遇见 1001 就产生进位，不象普通的二进制码到 1111 才产生进位 10000。
- 3、如果 In 输入含非 BCD 码（含 0xA、0xB、0xC、0xD、0xE、0xF）则指令不执行。

【指令示例】

//Network 2 BCD转换为数值



【程序说明】

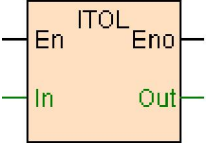
当 M1=ON，BIN 指令把 V1100 的 BCD 码转换 为数值结果放在 V20，D.BIN 指令把 V1101V1102 的 BCD 码转换为数值结果放在 V30V31，如下表。

In 元件	初始值	BIN 转换结果	D.BIN 转换结果
-------	-----	----------	------------

V1100	0x3938	V20 = 3938	
V1101V1102	0x35344241		V30V31 = 35344241

ITOL（整数转换为长整数）

指令格式及参数说明

语言	LD	FBD	IL
指令格式			ITOL En, In, Out

参数	参数定义	输入	输出	说明
En	使能	√		
In	输入	√		
Eno	使能输出		√	
Out	输出		√	占用 2 个连续元件

【指令功能及作用说明】

ITOL 指令用于把 16 位整数转换为 32 位长整数。

【指令示例】

//Network 1 整数转换为长整数



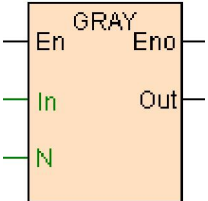
【程序说明】

当 M0=ON，ITOL 指令把 V1000 转换为 长整数结果放在 V0V1，把 V1001 转换为长整数结果放在 V2V3，如下表。

In 元件	初始值	转换结果
V1000	4648	V0V1 = 4648
V1001	-16961	V2V3 = -16961

GRAY（BIN 转换为 GRAY 码）

指令格式及参数说明

语言	LD	FBD	IL
指令格式			GRAY En, In, N, Out

参数	参数定义	输入	输出	说明
En	使能	√		
In	输入	√		In 为位元件时占用 N 个连续元件，为寄存器时占用(N-1)\16+1 个连续元件
N	GRAY 码长度	√		1-32
Eno	使能输出		√	
Out	输出		√	Out 为位 元件时占用 N 个连续元件，为寄存器时占用(N-1)\16+1 个连续元件

【指令功能及作用说明】

- 1、GRAY 指令用于将数值转换为格雷码。
- 2、In 必须>0 则指令不执行。

【指令示例】

//Network 1 转换为GRAY码



【程序说明】

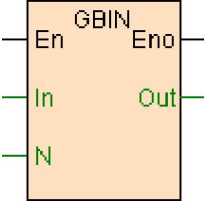
当 M0=ON，将 V0 的低 7 位转换为格雷码输出到 Y0~Y6，将 M10~M16 转换为格雷码输出到 V10。

In 元件	值	转换为格雷码结果
V0	25	Y0=ON
		Y1=OFF
		Y2=ON
		Y3=OFF
		Y4=ON
		Y5=OFF
		Y6=OFF
M10	ON	V10=21
M11	OFF	
M12	OFF	
M13	ON	
M14	ON	
M15	OFF	
M16	OFF	

GBIN（GRAY 码转换为 BIN）

指令格式及参数说明

语言	LD	FBD	IL
----	----	-----	----

语言	LD	FBD	IL
指令格式			GBIN En, In, N, Out

参 数	参数定义	输入	输出	说明
En	使能	√		
In	输入	√		In 为位 元件时占用 N 个连续元件，为寄存器时占用(N-1)\16+1 个连续元件
N	GRAY 码长度	√		1-32
Eno	使能输出		√	
Out	输出		√	Out 为位 元件时占用 N 个连续元件，为寄存器时占用(N-1)\16+1 个连续元件

【指令功能及作用说明】

- 1、GBIN 指令用于将格雷码转换为数值。
- 2、In 必须>0 则指令不执行。

【指令示例】

//Network 2 GRAY码转换为数值



【程序说明】

当 M1=ON，将 Y0~Y6 格雷码转换为数值输出到 V20，将 V10 的低 7 位格雷码转换为数值输出到 M100~M106。

In 元件	值	格雷码转换为数值
Y0	ON	V20=25
Y1	OFF	
Y2	ON	
Y3	OFF	
Y4	ON	
Y5	OFF	
Y6	OFF	
V10	21	M100=ON
		M101=OFF
		M102=OFF
		M103=ON
		M104=ON

		M105=OFF
		M106=OFF

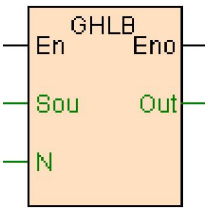
4.11 字符指令

字符指令列表

指令名称	8 位模式	32 位模式	指令功能	支持语言		
				LD	FBD	IL
GHLB			得出高低字节	√	√	√
GETB			截取字节串	√	√	√
BCMP	BCMP.LB		字节串比较	√	√	√
ITOC		D.ITOC	整数转换为字符	√	√	√
CTOI			字符转换为整数	√	√	√
FTOC			浮点数转换为字符	√	√	√
CTOF			字符转换为浮点数	√	√	√

GHLB（得到高低字节）

指令格式及参数说明

语言	LD	FBD	IL
指令格式			GHLB En, Sou, N, Out

参数	参数定义	输入	输出	说明
En	使能	√		
Sou	源起始元件	√		占用 N 个连续元件
N	元件个数	√		
Eno	使能输出		√	
Out	输出		√	占用 2N 个连续元件

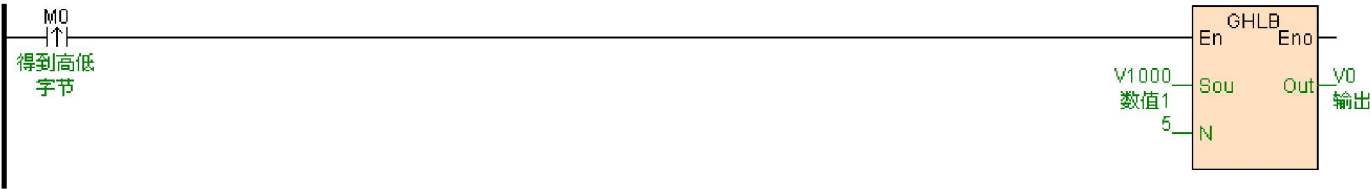
【指令功能及作用说明】

GHLB 指令将以 Sou 起始的 N 个寄存器的高低字节分开输出到 Out。

注：E/S 系列主机的 GHLB 指令参见“[GHLB](#)”。

【指令示例】

//Network 1 得到高低字节



【程序说明】

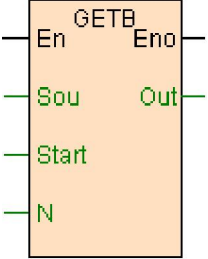
当 M0=ON，将 V1000~V1004 的高低字节分开输出到 V0~V9。

Sou 元件	值	输出元件	输出结果
--------	---	------	------

V1000	0x001E	V0	0x1E
		V1	0x00
V1001	0xFF6A	V2	0x6A
		V3	0xFF
V1002	0x0E19	V4	0x19
		V5	0x0E
V1003	0x1208	V6	0x08
		V7	0x12
V1004	0x0D5F	V8	0x5F
		V9	0x0D

GETB（截取字节串）

指令格式及参数说明

语言	LD	FBD	IL
指令格式			GETB En, Sou, Start, N, Out

参数	参数定义	输入	输出	说明
En	使能	√		
Sou	源起始元件	√		占用(Start+N)\2 个连续元件
Start	起始字节序号	√		
N	字节数	√		
Eno	使能输出		√	
Out	输出		√	占用(N+1)\2 个连续元件

【指令功能及作用说明】

GETB 指令在以 Sou 起始的字节串中从第 Start 个字节开始截取 N 个字节。

注：E/S 系列主机不支持 GETB 指令。

【指令示例】

//Network 1 截取字节串



【程序说明】

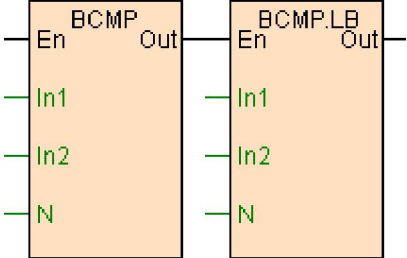
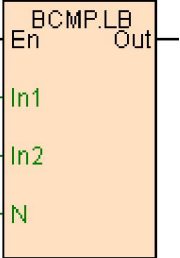
当 M0=ON，在 V1000 起始的字节串中从第 2 字节开始截取 7 个字节，输出到 V0~V3。

Sou 元件	值	输出元件	输出结果
V1000	0x011E	V0	0x6A01

V1001	0xFF6A	V1	0x19FF
V1002	0x0E19	V2	0x080E
V1003	0x1208	V3	0x0012
V1004	0x0D5F		

BCMP、BCMP.LB（字节串比较）

指令格式及参数说明

语言	LD	FBD	IL
16、8 位 指令格式			BCMP En, In1, In2, N, Out BCMP.LB En, In1, In2, N, Out

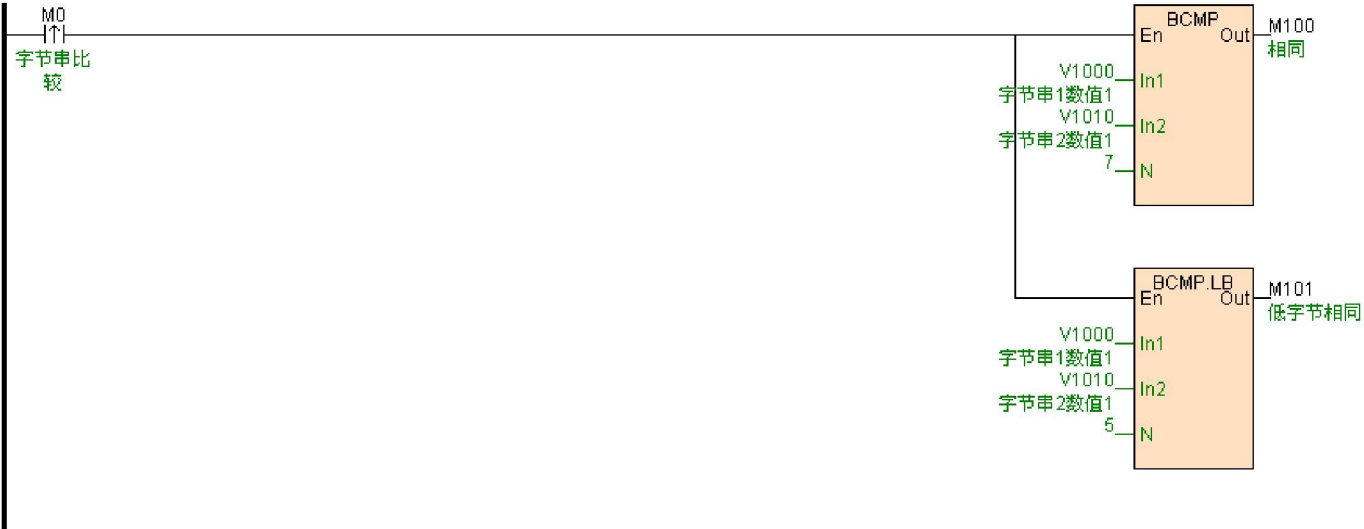
参数	参数定义	输入	输出	说明
En	使能	√		
In1	比较字符串 1	√		BCMP 占用(N+1)/2 个连续元件，BCMP.LB 占用 N 个连续元件
In2	比较字符串 2	√		BCMP 占用(N+1)/2 个连续元件，BCMP.LB 占用 N 个连续元件
N	比较字节数	√		
Out	比较结果输出		√	

【指令功能及作用说明】

BCMP 指令将 In1 和 In2 这 2 个字符串进行比较，比较 N 字节，如果相等那么 Out=ON，否则 Out=OFF。BCMP.LB 为低字节模式，只比较低字节部分。

【指令示例】

//Network 1 字符串比较



【程序说明】

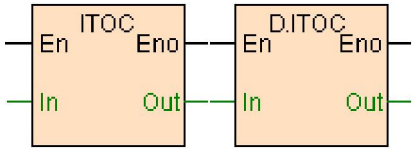
当 M0=ON，将字符串 V1000~V1004 和字符串 V1010~V1014 比较，BCMP 指令比较 7 个字节，BCMP.LB 指令比较 5 个低字节。

In1 元件	初始值	In2 元件	初始值	比较结果
V1000	0x011E	V1010	0x011E	M100=OFF
V1001	0xFF6A	V1011	0x026A	

V1002	0x0E19	V1012	0x0019	M101=ON
V1003	0x1208	V1013	0x1208	
V1004	0x0D5F	V1014	0x5D5F	

ITOC、D.ITOC（整数转换为字符）

指令格式及参数说明

语言	LD	FBD	IL
16、32 位 指令格式			ITOC En, In, Out D.ITOC En, In, Out

参数	参数定义	输入	输出	说明
En	使能	√		
In	输入	√		
Eno	使能输出		√	
Out	输出		√	占用 6 个连续元件

【指令功能及作用说明】

ITOC 指令将整数转换为字符串，D.ITOC 将长整数转换为字符串。输出会自动占用 6 个连续的寄存器，共可以表示 12 个字符，如果转换的结果不足 12 个字符则后面寄存器内容以 0 填充。

【指令示例】

//Network 1 整数转换为字符



【程序说明】

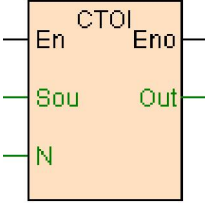
当 M0=ON,ITOC 把 V1000 的整数值转换为字符串输出到 V0~V5,D.ITOC 把 V1001V1002 的长整数值转换为字符串放在 V10~V15，如下表。

In 元件	初始值	转换结果
V1000	286	V0~V5 ="286"
V1001V1002	-2584810	V10~V15 =" -2584810"

CTOI（字符转换为整数）

指令格式及参数说明

语言	LD	FBD	IL
----	----	-----	----

语言	LD	FBD	IL
指令格式			CTOI En, Sou, N, Out

参数	参数定义	输入	输出	说明
En	使能	√		
Sou	源起始元件	√		占用 6 个连续元件
N	转换字符个数	√		N 范围：1~11
Eno	使能输出		√	
Out	输出		√	占用 2 个连续元件

【指令功能及作用说明】

- 1、CTOI 将以 Sou 起始的 N 个字符转换为长整数，如果转换结果超过长整数的范围则不转换并且 Eno 为 0（OFF），Out 保持原来不变。
- 2、N 为要转换的字符个数，有效范围 1~11，如果超过范围则不转换并且 Eno=OFF，Out 保持原来不变。
- 3、如果要转换的字符串中包括非法字符（除了 0~9、+、- 以外的字符），对前面的用空格替换，对后面的则截去。例如：字符串'123'、'123dfg'、'A123'转换结果都是整数 123。

【指令示例】

//Network 1 字符串转换为整数



【程序说明】

当 M0=ON，CTOI 把字符串 V1000~V1005 中的 7 个字符转换为整数值输出到 V0V1，把字符串 V1010~V1015 中的 9 个字符转换为整数值输出到 V2V3，如下表。

In 元件	初始值	转换结果
V1000~V1005	"1234567890"	V0V1 =1234567
V1010~V1015	"-987654321"	V2V3 =-98765432

FTOC（浮点数转换为字符）

指令格式及参数说明

语言	LD	FBD	IL
----	----	-----	----

语言	LD	FBD	IL
指令格式			FTOC En, In, Out

参数	参数定义	输入	输出	说明
En	使能	√		
In	输入	√		占用 2 个连续元件
Eno	使能输出		√	
Out	输出		√	占用 6 个连续元件

【指令功能及作用说明】

FTOC 指令将浮点数转换为字符串。输出会自动占用 6 个连续的寄存器，共可以表示 12 个字符，如果转换的结果不足 12 个字符则后面寄存器内容以 0 填充。

【指令示例】

//Network 1 浮点数转换为字符



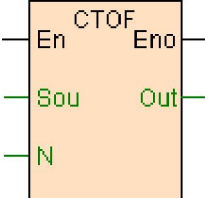
【程序说明】

当 M0=ON，FTOC 把浮点数 V1000V1001 转换为字符串输出到 V0~V5，把浮点数 V1002V1003 转换为字符串放在 V10~V15，如下表。

In 元件	初始值	转换结果
V1000V1001	23.4567	V0~V5 ="23.4567"
V1002V1003	-2987.56	V10~V15 ="-2987.56"

CTOF（字符转换为浮点数）

指令格式及参数说明

语言	LD	FBD	IL
指令格式			CTOF En, Sou, N, Out

参数	参数定义	输入	输出	说明
----	------	----	----	----

参数	参数定义	输入	输出	说明
En	使能	√		
Sou	源起始元件	√		占用 6 个连续元件
N	转换字符个数	√		N 范围：1~11
Eno	使能输出		√	
Out	输出		√	占用 2 个连续元件

【指令功能及作用说明】

- 1、CTOF 将以 Sou 起始的 N 个字符转换为浮点数，如果转换结果超过浮点数的范围则不转换并且 Eno 为 0（OFF），Out 保持原来不变。
- 2、N 为要转换的字符个数，有效范围 1~11，如果超过范围则不转换并且 Eno=OFF，Out 保持原来不变。
- 3、如果要转换的字符串中包括非法字符（除了 0 ~ 9、.、+、- 以外的字符），对前面的用空格替换，对后面的则截去。例如：字符串'1.23'、'1.23dfg'、'A1.23'转换结果都是浮点数 1.23。

【指令示例】

//Network 1 字符串转换为浮点数



【程序说明】

当 M0=ON，CTOF 把字符串 V1000~V1005 中的 7 个字符转换为浮点数输出到 V0V1，把字符串 V1010~V1015 中的 9 个字符转换为浮点数输出到 V2V3，如下表。

In 元件	初始值	转换结果
V1000~V1005	"1234.67890"	V0V1 =1234.67
V1010~V1015	"-98.654321"	V2V3 =-98.65432

4.12 数学运算指令

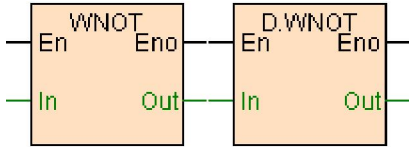
数学运算指令列表

指令名称	8 位模式	32 位模式	指令功能	支持语言		
				LD	FBD	IL
WNOT		D.WNOT	取反	√	√	√
WAND		D.WAND	与运算	√	√	√
WOR		D.WOR	或运算	√	√	√
WXOR		D.WXOR	异或运算	√	√	√
ADD		D.ADD	加法	√	√	√
SUB		D.SUB	减法	√	√	√

指令名称	8 位模式	32 位模式	指令功能	支持语言		
				LD	FBD	IL
INC		D.INC	加 1	√	√	√
DEC		D.DEC	减 1	√	√	√
MUL		D.MUL	乘法	√	√	√
DIV		D.DIV	除法	√	√	√
ACCU		D.ACCU	累加	√	√	√
AVG		D.AVG	平均值	√	√	√
ABS		D.ABS	绝对值	√	√	√
NEG		D.NEG	求 2 的补码	√	√	√

WNOT、D.WNOT（取反）

指令格式及参数说明

语言	LD	FBD	IL
16、32 位 指令格式			WNOT En, In, Out D.WNOT En, In, Out

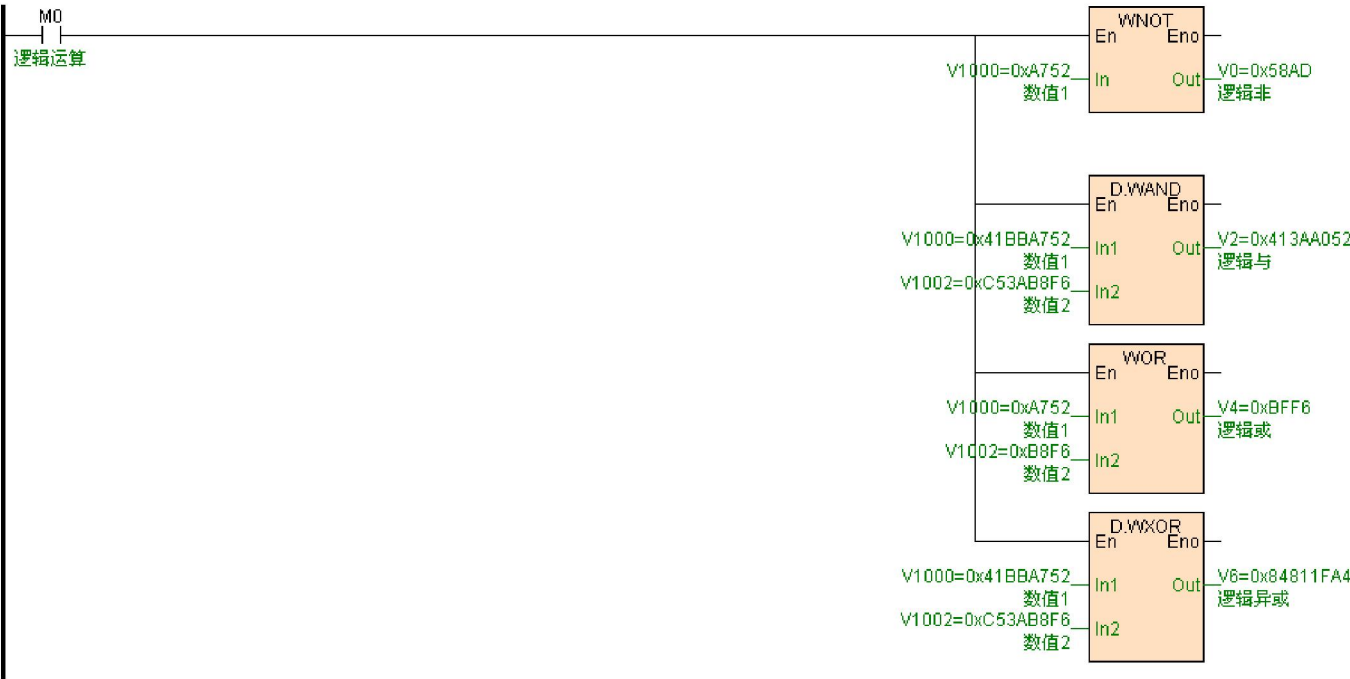
参数	参数定义	输入	输出	说明
En	使能	√		
In	输入	√		
Eno	使能输出		√	
Out	输出		√	

【指令功能及作用说明】

WNOT 指令将 In 按位取反后输出。

【指令示例】

//Network 1 逻辑运算



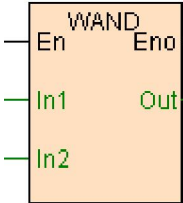
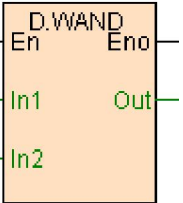
【程序说明】

当 M0=ON, 进行如下逻辑运算, V0= not V1000 ;V2V3=V1000V1001 and V1002V1003 ;V4=V1000 or V1002 ;V6V7=V1000V1001 xor V1002V1003, 如下表。

元件	初始值	运算结果
V1000	10100111 01010010	V0 =01011000 10101101
V1000V1001	01000001 10111011 10100111 01010010	V2V3 =01000001 00111010 1010000001010010
V1002	10111000 11110110	V4 =10111111 11110110
V1002V1003	11000101 00111010 10111000 11110110	V6V7 =10000100 10000001 00011111 10100100

WAND、D.WAND（与运算）

指令格式及参数说明

语言	LD	FBD	IL
16、32 位 指令格式			WAND En, In1, In2, Out D.WAND En, In1, In2, Out

参数	参数定义	输入	输出	说明
En	使能	√		
In1	输入 1	√		
In2	输入 2	√		
Eno	使能输出		√	
Out	输出		√	

【指令功能及作用说明】

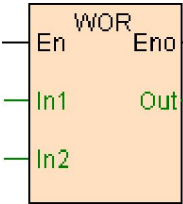
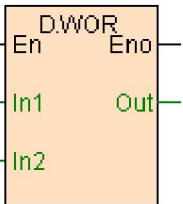
WAND 指令将 In1、In2 按位进行逻辑与运算后输出。

【指令示例】

参见 WNOT 指令示例。

WOR、D.WOR（或运算）

指令格式及参数说明

语言	LD	FBD	IL
16、32 位 指令格式			WOR En, In1, In2, Out D.WOR En, In1, In2, Out

参数	参数定义	输入	输出	说明
En	使能	√		
In1	输入 1	√		
In2	输入 2	√		

参数	参数定义	输入	输出	说明
Eno	使能输出		√	
Out	输出		√	

【指令功能及作用说明】

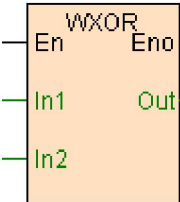
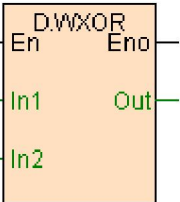
WOR 指令将 In1、In2 按位进行逻辑或运算后输出。

【指令示例】

参见 WNOT 指令示例。

WXOR、D.WXOR（异或运算）

指令格式及参数说明

语言	LD	FBD	IL
16、32 位 指令格式			WXOR En, In1, In2, Out D.WXOR En, In1, In2, Out

参数	参数定义	输入	输出	说明
En	使能	√		
In1	输入 1	√		
In2	输入 2	√		
Eno	使能输出		√	
Out	输出		√	

【指令功能及作用说明】

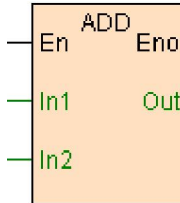
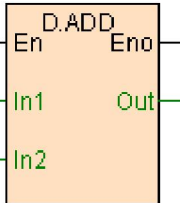
WXOR 指令将 In1、In2 按位进行逻辑异或运算后输出。

【指令示例】

参见 WNOT 指令示例。

ADD、D.ADD（加法）

指令格式及参数说明

语言	LD	FBD	IL
16、32 位 指令格式			ADD En, In1, In2, Out D.ADD En, In1, In2, Out

参数	参数定义	输入	输出	说明
En	使能	√		
In1	被加数	√		
In2	加数	√		
Eno	使能输出		√	

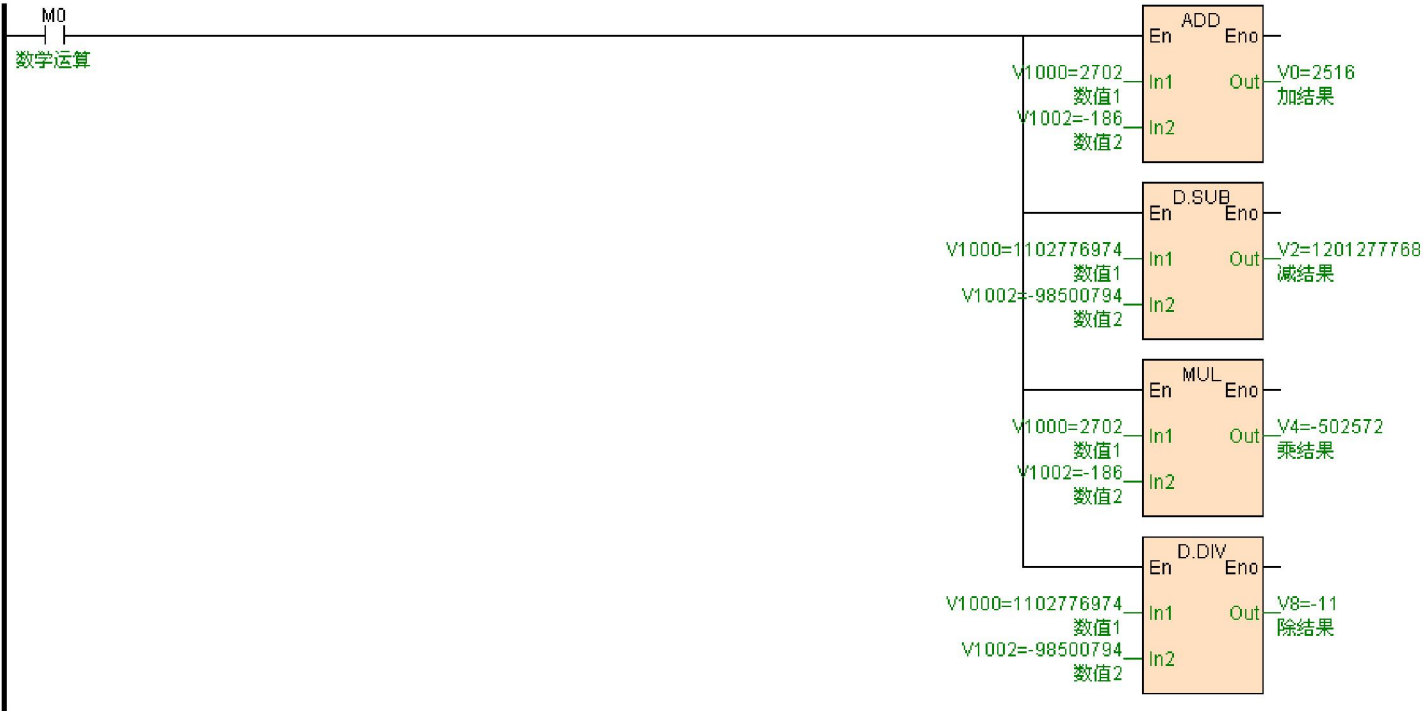
参数	参数定义	输入	输出	说明
Out	输出		√	

【指令功能及作用说明】

ADD 指令将 In1 加上 In2 进后输出到 Out。

【指令示例】

//Network 1 数学运算



【程序说明】

当 M0=ON, 进行如下逻辑运算, V0= V1000+V1002 ;V2V3=V1000V1001-V1002V1003 ;V4V5=V1000* V1002 ;V8V9=V1000V1001\ V1002V1003, 如下表。

元件	初始值	运算结果
V1000	2702	V0 =2516
V1000V1001	1102776974	V2V3 =1201277768
V1002	-186	V4V5 =-502572
V1002V1003	-98500794	商 V8V9 =-11, 余数 V10V11=19268240

SUB、D.SUB（减法）

指令格式及参数说明

语言	LD	FBD	IL
16、32 位 指令格式			SUB En, In1, In2, Out D.SUB En, In1, In2, Out

参数	参数定义	输入	输出	说明
En	使能	√		

参数	参数定义	输入	输出	说明
In1	被减数	√		
In2	减数	√		
Eno	使能输出		√	
Out	输出		√	

【指令功能及作用说明】

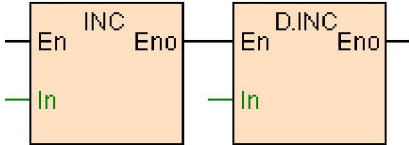
SUB 指令将 In1 减去 In2 后输出到 Out。

【指令示例】

参见 ADD 指令示例

INC、D.INC（加 1）

指令格式及参数说明

语言	LD	FBD	IL
16、32 位 指令格式			INC En, In D.INC En, In

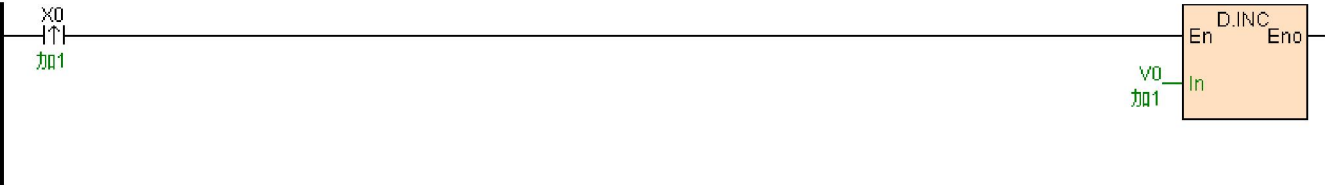
参数	参数定义	输入	输出	说明
En	使能	√		
In	被加数	√		
Eno	使能输出		√	

【指令功能及作用说明】

- 1、INC 指令将 In 加 1 后重新存放到 In 中。
- 2、16 位指令 INC：若 In=32767，加 1 后变为-32768。
- 3、32 位指令 D.INC：若 In=2147483647，加 1 后变为-2147483648。

【指令示例】

//Network 1



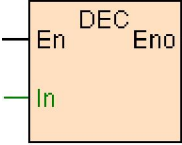
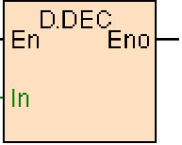
【程序说明】

当 X0=ON，则 V0V1=V0V1+1

DEC、D.DEC（减 1）

指令格式及参数说明

语言	LD	FBD	IL
----	----	-----	----

语言	LD	FBD	IL
16、32 位 指令格式			DEC En, In D.DEC En, In

参数	参数定义	输入	输出	说明
En	使能	√		
In	被减数	√		
Eno	使能输出		√	

【指令功能及作用说明】

- 1、DEC 指令将 In 减 1 后重新存放到 In 中。
- 2、16 位指令 DEC：若 In=-32768，加 1 后变为 32767。
- 3、32 位指令 D.DEC：若 In=-2147483648，加 1 后变为 2147483647。

【指令示例】

//Network 2

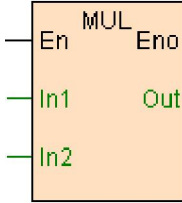
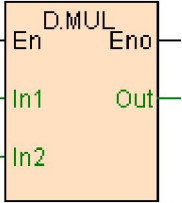


【程序说明】

当 X1=ON，则 V2=V2-1

MUL、D.MUL（乘法）

指令格式及参数说明

语言	LD	FBD	IL
16、32 位 指令格式			MUL En, In1, In2, Out D.MUL En, In1, In2, Out

参数	参数定义	输入	输出	说明
En	使能	√		
In1	被乘数	√		
In2	乘数	√		
Eno	使能输出		√	
Out	输出		√	MUL：占用 2 个连续元件，D.MUL：占用 4 个连续元件

【指令功能及作用说明】

MUL 指令将 In1 乘以 In2 输出到 Out、Out+1，32 位指令 D.MUL 将（In1、In1+1）乘以（In2、In2+1），运算结果 输出到（Out、

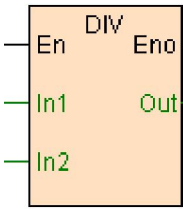
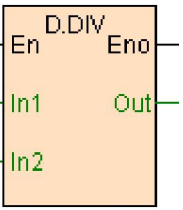
Out+1、Out+2、Out+3)。

【指令示例】

参见 ADD 指令示例

DIV、D.DIV（除法）

指令格式及参数说明

语言	LD	FBD	IL
16、32 位 指令格式			DIV En, In1, In2, Out D.DIV En, In1, In2, Out

参数	参数定义	输入	输出	说明
En	使能	√		
In1	被除数	√		
In2	除数	√		
Eno	使能输出		√	
Out	输出		√	DIV：占用 2 个连续元件，D.DIV：占用 4 个连续元件

【指令功能及作用说明】

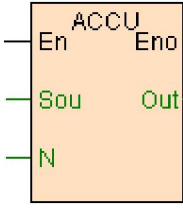
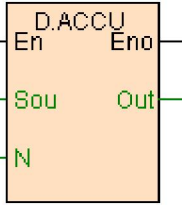
- 1、16 位指令（DIV），将 In1 除以 In2，运算结果的商数存放到 Out 中，余数存放到 Out+1 中。
- 2、32 位运算（D.DIV），将（In1、In1+1）除以（In2、In2+1），运算结果的商数存放到（Out、Out+1），余数存放到（Out+2、Out+3）。

【指令示例】

参见 ADD 指令示例

ACCU、D.ACCU（累加）

指令格式及参数说明

语言	LD	FBD	IL
16、32 位 指令格式			ACCU En, Sou, N, Out D.ACCU En, Sou, N, Out

参数	参数定义	输入	输出	说明
En	使能	√		
Sou	源起始元件	√		ACCU：占用 N 个连续元件，D.ACCU：占用 2N 个连续元件
N	数据个数	√		
Eno	使能输出		√	
Out	输出		√	ACCU：占用 2 个连续元件，D.ACCU：占用 4 个连续元件

【指令功能及作用说明】

- 1、ACCU 指令求以 Sou 起始 N 个 16 位整数的累加和，运算结果存放到 Out、Out+1。

2、D.ACCU 指令求以 Sou 为 N 个 32 位整数（2 个 16 位寄存器）的累加和，运算结果存放到 Out、Out+1、Out+2、Out+3。

【指令示例】

//Network 1 累加



【程序说明】

当 M0=ON，V0V1= V1000+V1001+V1002+V1003，如下表。

元件	初始值	运算结果
V1000	2702	V0V1 =17839
V1001	16827	
V1002	-186	
V1003	-1504	

AVG、D.AVG（平均值）

指令格式及参数说明

语言	LD	FBD	IL
16、32 位 指令格式			AVG En, Sou, N, Out D.AVG En, Sou, N, Out

参数	参数定义	输入	输出	说明
En	使能	√		
Sou	源起始元件	√		AVG：占用 N 个连续元件，D.AVG：占用 2N 个连续元件
N	数据个数	√		
Eno	使能输出		√	
Out	输出		√	AVG：占用 2 个连续元件，D.AVG：占用 4 个连续元件

【指令功能及作用说明】

- 1、AVG 指令求以 Sou 起始 N 个 16 位整数的平均值，运算结果存放到 Out，余数放在 Out+1。
- 2、D.AVG 指令求以 Sou 为 N 个 32 位整数（2 个 16 位寄存器）的平均值，运算结果存放到 Out、Out+1，余数放在 Out+2、Out+3。

【指令示例】

//Network 1 求平均值



【程序说明】

当 M0=ON，V0= (V1000+V1001+V1002+V1003)\4，如下表。

元件	初始值	运算结果
V1000	2702	V0=4459，余数 V1=3
V1001	16827	
V1002	-186	
V1003	-1504	

ABS、D.ABS（绝对值）

指令格式及参数说明

语言	LD	FBD	IL
16、32 位 指令格式			ABS En, In D.ABS En, In

参数	参数定义	输入	输出	说明
En	使能	√		
In	输入	√		
Eno	使能输出		√	

【指令功能及作用说明】

ABS 指令求 In 的绝对值后结果重新存放到 In。

【指令示例】

//Network 1 求绝对值



【程序说明】

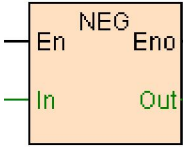
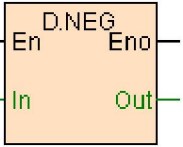
当 M0=ON，求 V1000、V1001V1002 的绝对值，如下表。

元件	初始值	取绝对值结果
V1000	2702	V1000=2702
V1001V1002	-12172869	V1001V1002=12172869

NEG、D.NEG（求 2 个补码）

指令格式及参数说明

语言	LD	FBD	IL
----	----	-----	----

语言	LD	FBD	IL
16、32 位 指令格式			NEG En, In, Out D.NEG En, In, Out

参数	参数定义	输入	输出	说明
En	使能	√		
In	输入	√		
Eno	使能输出		√	
Out	输出		√	

【指令功能及作用说明】

NEG 指令求 In 的 2 的补码。

【指令示例】

//Network 1 求2的补码



【程序说明】

当 M0=ON，求 V1000、V1001V1002 的 2 的补码，如下表。

元件	初始值	2 的补码
V1000	2702	V0=-2702
V1001V1002	-12172869	V2V3=12172869

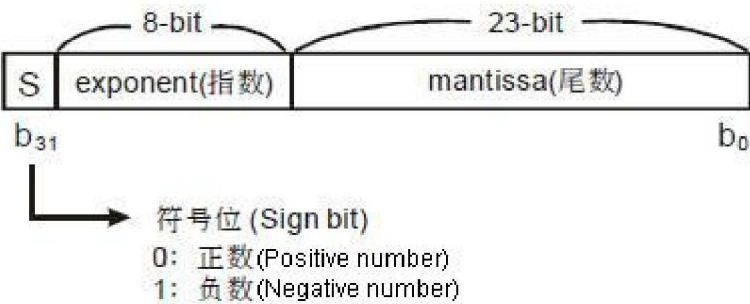
4.13 浮点数指令

浮点数指令列表

指令名称	8 位模式	32 位模式	指令功能	支持语言		
				LD	FBD	IL
FCMP			浮点数比较	√	√	√
FZCP			浮点数区域比较	√	√	√
FMOV			浮点数移动指令	√	√	√
FADD			浮点数加法	√	√	√
FSUB			浮点数减法	√	√	√
FMUL			浮点数乘法	√	√	√
FDIV			浮点数除法	√	√	√
FACCU			浮点数累加	√	√	√
FAVG			浮点数平均值	√	√	√

指令名称	8 位模式	32 位模式	指令功能	支持语言		
				LD	FBD	IL
FMAX			浮点数最大值	√	√	√
FMIN			浮点数最小值	√	√	√
FTOI			浮点数转换为整数	√	√	√
ITOF		D.ITOF	整数转换为浮点数	√	√	√
FABS			浮点数绝对值	√	√	√
FSQR			浮点数平方根	√	√	√
FSIN			正弦值	√	√	√
FCOS			余弦值	√	√	√
FTAN			正切值	√	√	√
FASIN			反正弦	√	√	√
FACOS			反余弦	√	√	√
FATAN			反正切	√	√	√
FLN			自然对数	√	√	√
FLOG			以 10 为底的对数	√	√	√
FEXP			自然指数	√	√	√
FRAD			角度转换为弧度	√	√	√
FDEG			弧度转换为角度	√	√	√
FXI			指数指令	√	√	√

Haiwell（海为）PLC 的浮点数采用 IEEE754 的标准，以 32 位表示浮点数（占用 2 个寄存器），数目范围为±2-126 到±2+128 相当于 ±1.1755e-38 到±3.4028e+38，格式如下：



FCMP（浮点数比较）

指令格式及参数说明

语言	LD	FBD	IL
指令格式			FCMP En, In1, In2, Out

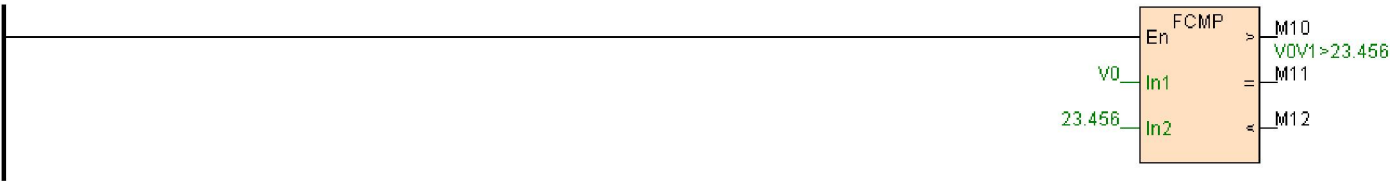
参数	参数定义	输入	输出	说明
En	使能	√		
In1	输入 1	√		
In2	输入 2	√		
Out	状态输出		√	占用 3 个连续元件

【指令功能及作用说明】

FCMP 是浮点数比较指令，它同时输出>、=、<这 3 个结果。

【指令示例】

//Network 1



【程序说明】

- 1、FCMP 指令从母排取电一直执行。
- 2、当 V0V1>23.456 则 M10=ON，当 V0V1=23.456 则 M11=ON，当 V0V1<23.456 则 M12=ON。

FZCP（浮点数区域比较）

指令格式及参数说明

语言	LD	FBD	IL
指令格式			FZCP En, In, Up, Down, Out

参数	参数定义	输入	输出	说明
En	使能	√		
In	输入	√		
Up	区域上限	√		
Down	区域下限	√		
Out	状态输出		√	占用 3 个连续元件

【指令功能及作用说明】

- 1、FZCP 是浮点数区域比较指令，它同时输出>、=、<这 3 个结果。
- 2、如果区域上限 Up<区域下限 Down，那么指令会自动将它们交换。

【指令示例】

//Network 1



【程序说明】

- 1、FZCP 指令从母排取电一直执行。
- 2、当 V0V1>9876.05 则 M10=ON，当 V0V1≤9876.05 并且 V0V1≥-23.12 则 M11=ON，当 V0V1<-23.12 则 M12=ON。

FMOV（浮点数移动）

指令格式及参数说明

语言	LD	FBD	IL
指令格式			FMOV En, In, Out

参数	参数定义	输入	输出	说明
En	使能	√		
In	输入	√		
Eno	使能输出		√	
Out	输出		√	

【指令功能及作用说明】

浮点数移动指令 FMOV 也叫浮点数赋值指令，用于将指定的浮点数赋值到输出寄存器 Out。

【指令示例】

//Network 1 程序开始赋初始值

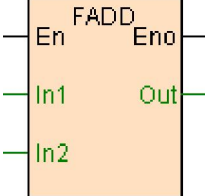


【程序说明】

程序首次扫描时赋初始值，V0V1=35.6，V10V11=-2.632。

FADD（浮点数加法）

指令格式及参数说明

语言	LD	FBD	IL
指令格式			FADD En, In1, In2, Out

参数	参数定义	输入	输出	说明
En	使能	√		

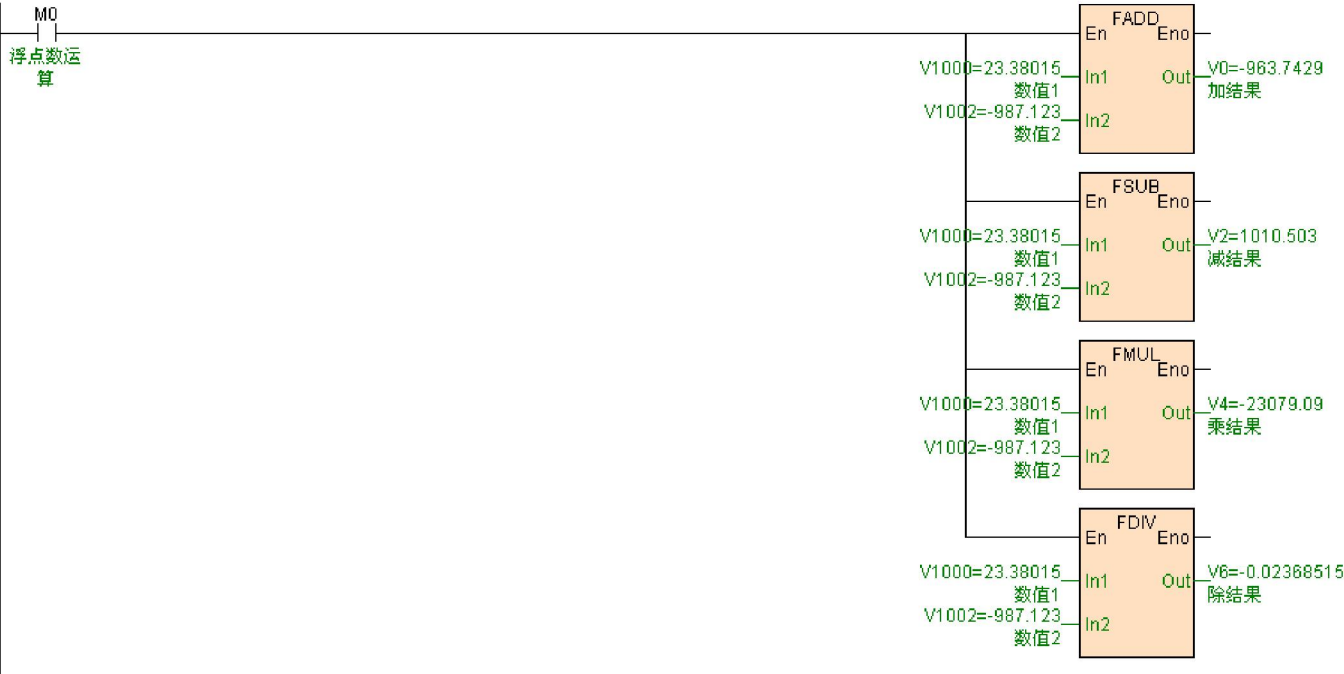
参数	参数定义	输入	输出	说明
In1	被加数	√		
In2	加数	√		
Eno	使能输出		√	
Out	输出		√	

【指令功能及作用说明】

FADD 指令将浮点数 In1 加上浮点数 In2 后输出到 Out。

【指令示例】

//Network 1 浮点数数学运算



【程序说明】

当 M0=ON，进行如下浮点数运算，V0V1= V1000V1001+V1002V1003；V2V3=V1000V1001-V1002V1003；V4V5=V1000V1001* V1002V1003；V6V7=V1000V1001\ V1002V1003，如下表。

元件	初始值	运算结果
V1000V1001	23.38015	V0V1 =-963.7429
		V2V3 =1010.503
V1002V1003	-987.123	V4V5 =-23079.09
		V6V7 =-0.02368515

FSUB（浮点数减法）

指令格式及参数说明

语言	LD	FBD	IL
指令格式			FSUB En, In1, In2, Out

参数	参数定义	输入	输出	说明
En	使能	√		
In1	被减数	√		
In2	减数	√		
Eno	使能输出		√	
Out	输出		√	

【指令功能及作用说明】

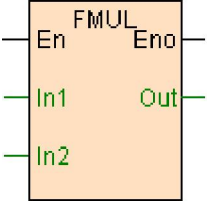
FSUB 指令将浮点数 In1 减去浮点数 In2 后输出到 Out。

【指令示例】

参见 FADD 指令示例。

FMUL（浮点数乘法）

指令格式及参数说明

语言	LD	FBD	IL
指令格式			FMUL En, In1, In2, Out

参数	参数定义	输入	输出	说明
En	使能	√		
In1	被乘数	√		
In2	乘数	√		
Eno	使能输出		√	
Out	输出		√	

【指令功能及作用说明】

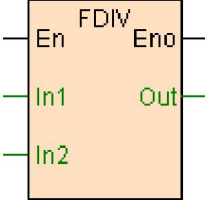
FMUL 指令将浮点数 In1 乘以浮点数 In2 后输出到 Out。

【指令示例】

参见 FADD 指令示例。

FDIV（浮点数除法）

指令格式及参数说明

语言	LD	FBD	IL
指令格式			FDIV En, In1, In2, Out

参数	参数定义	输入	输出	说明
En	使能	√		

参数	参数定义	输入	输出	说明
In1	被除数	√		
In2	除数	√		
Eno	使能输出		√	
Out	输出		√	

【指令功能及作用说明】

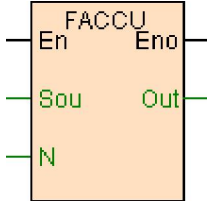
FDIV 指令将浮点数 In1 除以浮点数 In2 后输出到 Out。

【指令示例】

参见 FADD 指令示例。

FACCU（浮点数累加）

指令格式及参数说明

语言	LD	FBD	IL
指令格式			FACCU En, Sou, N, Out

参数	参数定义	输入	输出	说明
En	使能	√		
Sou	源起始元件	√		
N	数据个数	√		
Eno	使能输出		√	
Out	输出		√	

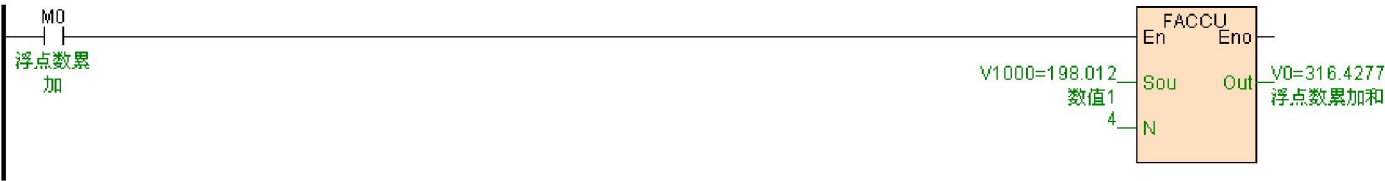
【指令功能及作用说明】

FACCU 指令求以 Sou 起始 N 个 浮点数的累加和，运算结果存放到 Out。

注：E/S 系列主机不支持 FACCU 指令。

【指令示例】

//Network 1 浮点数累加



【程序说明】

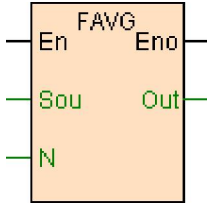
当 M0=ON，V0V1= V1000V1001+V1002V1003+V1004V1005+V1006V1007，如下表。

元件	初始值	运算结果
V1000V1001	198.012	V0V1 =316.4277
V1002V1003	23.781	
V1004V1005	-3.714	

V1006V1007	98.3487	
------------	---------	--

FAVG（浮点数平均值）

指令格式及参数说明

语言	LD	FBD	IL
指令格式			FAVG En, Sou, N, Out

参数	参数定义	输入	输出	说明
En	使能	√		
Sou	源起始元件	√		
N	数据个数	√		
Eno	使能输出		√	
Out	输出		√	

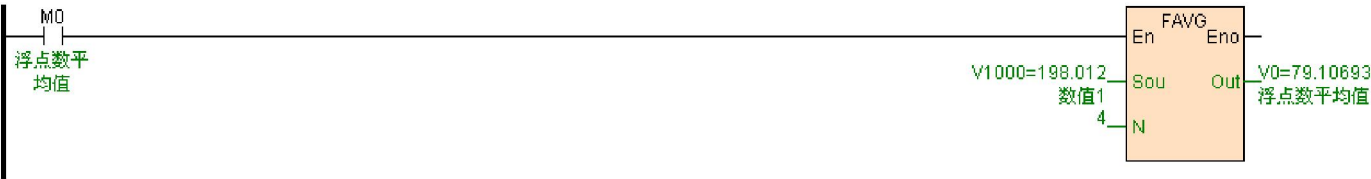
【指令功能及作用说明】

FAVG 指令求以 Sou 起始 N 个 浮点数的平均值，运算结果存放到 Out。

注：E/S 系列主机不支持 FAVG 指令。

【指令示例】

//Network 1 浮点数平均值



【程序说明】

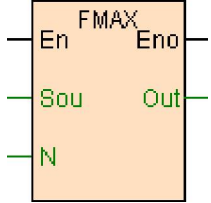
当 M0=ON，V0V1= (V1000V1001+V1002V1003+V1004V1005+V1006V1007)\4，如下表。

元件	初始值	运算结果
V1000V1001	198.012	V0V1=79.10693
V1002V1003	23.781	
V1004V1005	-3.714	
V1006V1007	98.3487	

FMAX（浮点数最大值）

指令格式及参数说明

语言	LD	FBD	IL
----	----	-----	----

语言	LD	FBD	IL
指令格式			FMAX En, Sou, N, Out

参数	参数定义	输入	输出	说明
En	使能	√		
Sou	源起始元件	√		
N	数据个数	√		
Eno	使能输出		√	
Out	比较结果输出		√	

【指令功能及作用说明】

FMAX 指令将以 Sou 起始的 N 个浮点数进行比较，将最大值输出到 Out。

【指令示例】

//Network 1 浮点数最大值

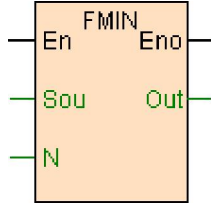


【程序说明】

- 1、M0=ON，假如 V1100V1101=198.012、V1102V1103=23.781、V1104V1105=-3.714、V1106V1107=98.3487，那么 V0V1=198.012。
- 2、当 M0=OFF，指令停止执行，Out 保持不变。

FMIN（浮点数最小值）

指令格式及参数说明

语言	LD	FBD	IL
指令格式			FMIN En, Sou, N, Out

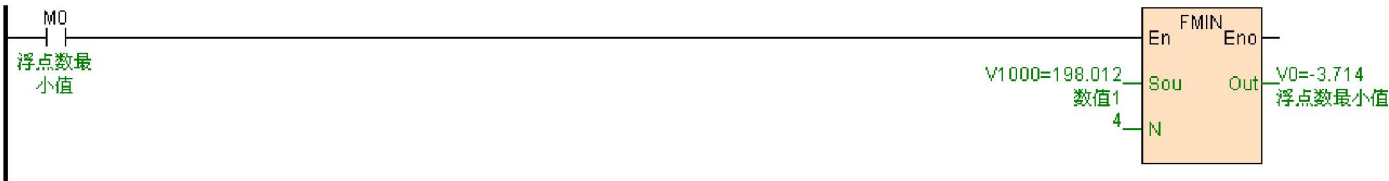
参数	参数定义	输入	输出	说明
En	使能	√		
Sou	源起始元件	√		
N	数据个数	√		
Eno	使能输出		√	
Out	比较结果输出		√	

【指令功能及作用说明】

FMIN 指令将以 Sou 起始的 N 个浮点数进行比较，将最小值输出到 Out。

【指令示例】

//Network 1 浮点数最小值



【程序说明】

- 1、M0=ON，假如 V1100V1101=198.012、V1102V1103=23.781、V1104V1105=-3.714、V1106V1107=98.3487，那么 V0V1=-3.714。
- 2、当 M0=OFF，指令停止执行，Out 保持不变。

FTOI（浮点数转换为整数）

指令格式及参数说明

语言	LD	FBD	IL
指令格式			FTOI En, In, Out

参数	参数定义	输入	输出	说明
En	使能	√		
In	输入	√		
Eno	使能输出		√	
Out	输出		√	

【指令功能及作用说明】

FTOI 指令把浮点数转换为整数。

【指令示例】

//Network 1 浮点数转换为整数



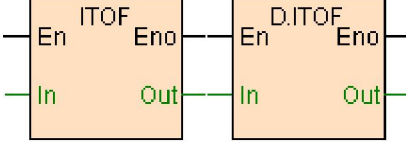
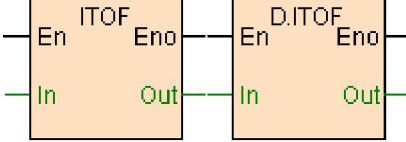
【程序说明】

当 M0=ON，FTOI 指令把浮点数 V1000V1001 转换为整数结果放在 V0V1，如下表。

In 元件	初始值	转换结果
V1000V1001	-2345.987	V0V1 = -2346

ITOF、D.ITOF（整数转换为浮点数）

指令格式及参数说明

语言	LD	FBD	IL
16、32 位 指令格式			ITOF En, In, Out D.ITOF En, In, Out

参数	参数定义	输入	输出	说明
En	使能	√		
In	输入	√		
Eno	使能输出		√	
Out	输出		√	

【指令功能及作用说明】

ITOF 指令把整数转换为浮点数。

【指令示例】

//Network 1 整数转换为浮点数



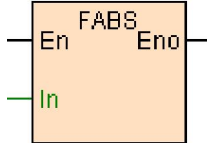
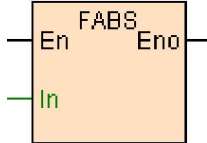
【程序说明】

当 M0=ON，ITOF 指令把 V1000 转换为浮点数结果放在 V0V1，把 V1000V1001 转换为浮点数结果放在 V2V3，如下表。

In 元件	初始值	转换结果
V1000	4648	V0V1 = 4648.0
V1000V1001	-257496	V2V3 = -257496.0

FABS（浮点数绝对值）

指令格式及参数说明

语言	LD	FBD	IL
指令格式			FABS En, In

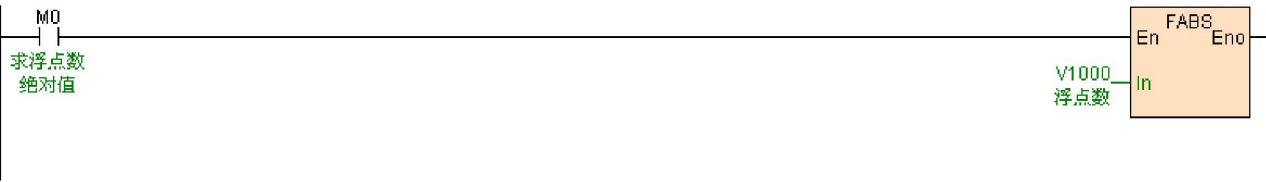
参数	参数定义	输入	输出	说明
En	使能	√		
In	输入	√		
Eno	使能输出		√	

【指令功能及作用说明】

FABS 指令求浮点数 In 的绝对值后结果重新存放到 In。

【指令示例】

//Network 1 求浮点数绝对值



【程序说明】

当 M0=ON，求浮点数 V1000V1001 的绝对值，如下表。

元件	初始值	取绝对值结果
V1000V1001	-23.3456	V1000V1001=23.3456

FSQR（浮点数平方根）

指令格式及参数说明

语言	LD	FBD	IL
指令格式			FSQR En, In, Out

参数	参数定义	输入	输出	说明
En	使能	√		
In	输入	√		
Eno	使能输出		√	
Out	输出		√	

【指令功能及作用说明】

FSQR 指令求浮点数的平方根。

【指令示例】

//Network 1 求浮点数平方根



【程序说明】

当 M0=ON，求浮点数 V1000V1001 的平方根，如下表。

元件	初始值	结果
V1000V1001	23.3456	V0V1=4.831728

FSIN（正弦值）

指令格式及参数说明

语言	LD	FBD	IL
指令格式			FSIN En, Angle, Out

参数	参数定义	输入	输出	说明
En	使能	√		
Angle	角度	√		
Eno	使能输出		√	
Out	输出		√	

【指令功能及作用说明】

FSIN 指令求角度的正弦值。

【指令示例】

//Network 1 求正弦、余弦、正切



【程序说明】

当 M0=ON，求浮点数 V1000V1001 的正弦值，浮点数 V1002V1003 的余弦值，浮点数 V1004V1005 的正切值，如下表。

元件	初始值	结果
V1000V1001	30.0	正弦值 V0V1=0.5
V1002V1003	45.0	余弦值 V2V3=0.7071068
V1004V1005	70.0	正切值 V4V5=2.747478

FCOS（余弦值）

指令格式及参数说明

语言	LD	FBD	IL
指令格式			FCOS En, Angle, Out

参数	参数定义	输入	输出	说明
----	------	----	----	----

参数	参数定义	输入	输出	说明
En	使能	√		
Angle	角度	√		
Eno	使能输出		√	
Out	输出		√	

【指令功能及作用说明】

FCOS 指令求角度的余弦值。

【指令示例】

参见 FSIN 指令示例

FTAN（正切值）

指令格式及参数说明

语言	LD	FBD	IL
指令格式			FTAN En, Angle, Out

参数	参数定义	输入	输出	说明
En	使能	√		
Angle	角度	√		
Eno	使能输出		√	
Out	输出		√	

【指令功能及作用说明】

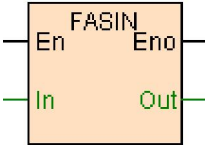
FTAN 指令求角度的正切值。

【指令示例】

参见 FSIN 指令示例

FASIN（反正弦）

指令格式及参数说明

语言	LD	FBD	IL
指令格式			FASIN En, In, Out

参数	参数定义	输入	输出	说明
En	使能	√		
In	输入	√		范围：-1 ~ 1
Eno	使能输出		√	
Out	输出弧度		√	

【指令功能及作用说明】

FASIN 指令求 In 的反正弦值。In 输入范围：-1 ~ 1 之间，如果 In 超范围，指令不执行，Eno=OFF。

【指令示例】

//Network 1 求反正弦、反余弦、反正切



【程序说明】

当 M0=ON，求浮点数 V1000V1001 的反正弦值，浮点数 V1002V1003 的反余弦值，浮点数 V1004V1005 的反正切值，如下表。

元件	初始值	结果
V1000V1001	0.8912	反正弦值 V0V1=1.099984
V1002V1003	-0.3409	反余弦值 V2V3=1.91867
V1004V1005	23.0912	反正切值 V4V5=1.527517

FACOS（反余弦）

指令格式及参数说明

语言	LD	FBD	IL
指令格式			FACOS En, In, Out

参数	参数定义	输入	输出	说明
En	使能	√		
In	输入	√		范围：-1 ~ 1
Eno	使能输出		√	
Out	输出弧度		√	

【指令功能及作用说明】

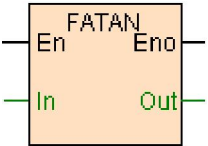
FACOS 指令求 In 的反余弦值。In 输入范围：-1 ~ 1 之间，如果 In 超范围，指令不执行，Eno=OFF。

【指令示例】

参见 FASIN 指令示例

FATAN（反正切）

指令格式及参数说明

语言	LD	FBD	IL
指令格式			FATAN En, In, Out

参数	参数定义	输入	输出	说明
En	使能	√		
In	输入	√		
Eno	使能输出		√	
Out	输出弧度		√	

【指令功能及作用说明】

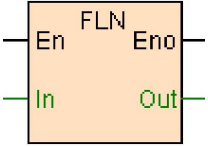
FATAN 指令求 In 的反正切值。

【指令示例】

参见 FASIN 指令示例

FLN（自然对数）

指令格式及参数说明

语言	LD	FBD	IL
指令格式			FLN En, In, Out

参数	参数定义	输入	输出	说明
En	使能	√		
In	输入	√		
Eno	使能输出		√	
Out	输出		√	

【指令功能及作用说明】

FLN 指令求 In 的自然对数。In 输入必须大于 0，否则指令不执行，Eno=OFF。

【指令示例】

//Network 1 求自然对数、10为底的对数、自然指数



【程序说明】

当 M0=ON，求浮点数 V1000V1001 的自然对数，浮点数 V1002V1003 的以 10 为底的对数，浮点数 V1004V1005 的自然指数，如下表。

元件	初始值	结果
V1000V1001	0.8912	自然对数 V0V1=-0.1151864
V1002V1003	3.8912	10 为底的对数 V2V3=0.5900835
V1004V1005	2.2398	自然指数 V4V5=9.391453

FLOG（以 10 为底的对数）

指令格式及参数说明

语言	LD	FBD	IL
指令格式			FLOG En, In, Out

参数	参数定义	输入	输出	说明
En	使能	√		
In	输入	√		
Eno	使能输出		√	
Out	输出		√	

【指令功能及作用说明】

FLOG 指令求 In 的以 10 为底的对数。In 输入必须大于 0，否则指令不执行，Eno=OFF。

【指令示例】

参见 [FLN](#) 指令示例

FEXP（自然指数）

指令格式及参数说明

语言	LD	FBD	IL
----	----	-----	----

语言	LD	FBD	IL
指令格式			FEXP En, In, Out

参数	参数定义	输入	输出	说明
En	使能	√		
In	输入	√		
Eno	使能输出		√	
Out	输出		√	

【指令功能及作用说明】

FEXP 指令求 In 的自然指数。In 输入必须小于 88.72284，否则指令不执行，Eno=OFF。

【指令示例】

参见 [FLN](#) 指令示例

FRAD（角度转换为弧度）

指令格式及参数说明

语言	LD	FBD	IL
指令格式			FRAD En, Angle, Out

参数	参数定义	输入	输出	说明
En	使能	√		
Angle	角度	√		
Eno	使能输出		√	
Out	输出弧度		√	

【指令功能及作用说明】

FRAD 指令把角度转换为弧度。

【指令示例】

//Network 1 角度、弧度转换



【程序说明】

当 M0=ON，把角度 V1000V1001 转换为弧度，弧度 V1002V1003 转换为角度，如下表。

元件	初始值	结果
V1000V1001	30.0	V0V1=0.5235988
V1002V1003	1.57	V2V3=89.95438

FDEG（弧度转换为角度）

指令格式及参数说明

语言	LD	FBD	IL
指令格式			FDEG En, In, Angle

参数	参数定义	输入	输出	说明
En	使能	√		
In	输入弧度	√		
Eno	使能输出		√	
Angle	角度		√	

【指令功能及作用说明】

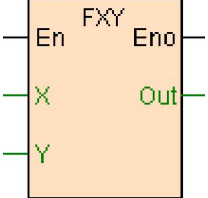
FDEG 指令把弧度转换为角度。

【指令示例】

参见 [FRAD](#) 指令示例。

FX Y（指数）

指令格式及参数说明

语言	LD	FBD	IL
指令格式			FX Y En, X, Y, Out

参数	参数定义	输入	输出	说明
En	使能	√		
X	底数	√		
Y	指数	√		
Eno	使能输出		√	
Out	输出		√	

【指令功能及作用说明】

FX Y 指令求 X^Y 的值。若 $X=0$ 且 $Y\leq 0$ 或 $X<0$ 且 Y 的尾数部分不为 0 时指令不执行，Eno=OFF。

【指令示例】



【程序说明】

当 M0=ON，求 V1000V1001^V1002V1003 的值。

元件	初始值	结果
V1000V1001	30.0	V0V1=208.4877
V1002V1003	1.57	

4.14 时钟指令

时钟指令列表

指令名称	8 位模式	32 位模式	指令功能	支持语言		
				LD	FBD	IL
TCMP			实时时钟比较	√	√	√
TACCU			时间累计	√	√	√
SCLK			设置时钟	√	√	√
TIME			时间开关	√	√	√
DATE			日期开关	√	√	√
INVT			倒计时	√	√	√

TCMP（实时时钟比较）

指令格式及参数说明

语言	LD	FBD	IL
指令格式			TCMP En, Clock, Out

参数	参数定义	输入	输出	说明
En	使能	√		
Clock	比较时间起始元件	√		占用 6 个连续元件
Out	状态输出		√	占用 3 个连续元件

【指令功能及作用说明】

- TCMP 将系统实时时钟与 Clock 设定的时间进行比较，如果系统实时时钟>Clock 输出>状态，如果系统实时时钟=Clock 输出=状态，如果系统实时时钟<Clock 输出<状态。
- Clock 占用 6 个寄存器，分别为年（0~9999）、月（1~12）、日（1~31）、时（0~23）、分（0~59）、秒（0~59）。如果年设定在 0~99 范围，系统会认为是 2000~2099 年。
- 如果 Clock 是无效时间，指令不执行。

【指令示例】

//Network 1



【程序说明】

- 1、TCMR 指令从母排取电一直执行。
- 2、假如 V1000=2012、V1001=12、V1002=25、V1003=8、V1004=0、V1005=0，表示设定时间为 2012-12-25 8:0:0。
- 3、系统时钟存在在 SV12~SV18, 当系统实时时钟>Clock 则 M10=ON, 当系统实时时钟=Clock 则 M11=ON, 当系统实时时钟<Clock 则 M12=ON。

TACCU（时间累计）

指令格式及参数说明

语言	LD	FBD	IL
指令格式			TACCU En, Rst, CT

参数	参数定义	输入	输出	说明
En	使能	√		
Rst	复位	√		
Eno	使能输出		√	
CT	累计时间		√	占用 6 个连续元件

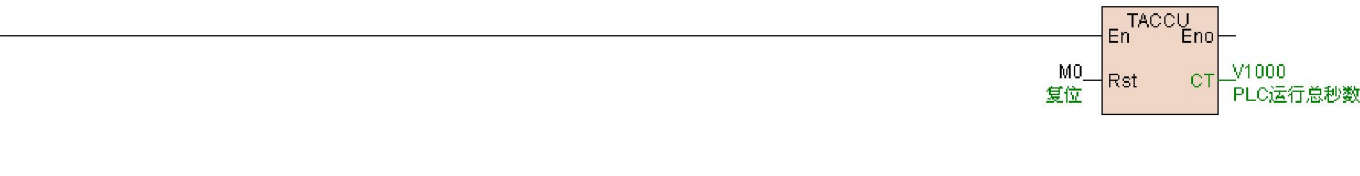
【指令功能及作用说明】

- 1、TACCU 指令以秒为单位累计 En=ON 的时间，它输出总累计秒数（CT、CT+1）以及总累计秒数相对应的天（CT+2）、时（CT+3）、分（CT+4）、秒（CT+5）。
- 2、累计时间 CT 需要用停电保持区的寄存器，默认停电保持寄存器为 V1000~V2047。
- 3、累计总秒数（CT、CT+1）达最大值 2147483647 后自动清零。

注：E/S 系列主机不支持 TACCU 指令。

【指令示例】

//Network 1 累计PLC运行时间



//Network 2 累计设备x运行时间



【程序说明】

- 1、Network 1 从母排取电，即累计 PLC 的运行时间。
- 2、Network 2 假如 X0 为设备 x 的运行反馈信号，X0=ON，TACCUM 执行计时，X0=OFF 则不计时。

SCLK（设置时钟）

指令格式及参数说明

语言	LD	FBD	IL
指令格式			SCLK En, Clock

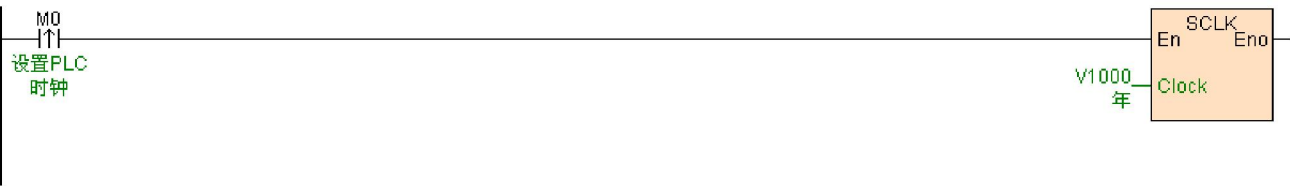
参数	参数定义	输入	输出	说明
En	使能	√		
Clock	时钟数据起始元件	√		占用 6 个连续元件
Eno	使能输出		√	

【指令功能及作用说明】

- 1、SCLK 指令用设定的时钟数据 Clock 去修改 PLC 的实时时钟。
- 2、Clock 占用 6 个寄存器，分别为年（0~9999）、月（1~12）、日（1~31）、时（0~23）、分（0~59）、秒（0~59）。如果年设定在 0~99 范围，系统会认为是 2000~2099 年。
- 3、如果 Clock 是无效时间，指令不执行。
- 4、SCLK 指令为边沿执行。
- 5、也可以通过编程软件菜单【PLC/设置 PLC 时钟】修改，无需编程。参见[“设置 PLC 时钟”](#)

【指令示例】

//Network 1

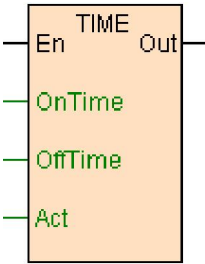


【程序说明】

- 1、假如 V1000=2012、V1001=12、V1002=25、V1003=8、V1004=0、V1005=0，表示设定时间为 2012-12-25 8:0:0。
- 2、当 M0=ON，PLC 系统时钟设置为 2012-12-25 8:0:0。

TIME（时间开关）

指令格式及参数说明

语言	LD	FBD	IL
指令格式			TIME En, OnTime, OffTime, Act, Out

参数	参数定义	输入	输出	说明
En	使能	√		
OnTime	ON 动作时间	√		
OffTime	OFF 动作时间	√		
Act	控制方式	√		
Out	状态输出		√	

【指令功能及作用说明】

- 1、TIME 指令以星期为控制周期，通过设定 ON、OFF 动作时间控制输出。
- 2、Act 为控制方式，其位 b0~b6 分别表示星期一~ 星期日的控制方式，当其相应位为 1 则表示该日对 ON、OFF 动作时间有效，为 0 则表示该日不动作。
- 3、如果 ON 动作时间<OFF 动作时间，表示在当日开和关。如果 ON 动作时间>OFF 动作时间，表示跨日开和关。
- 4、OnTime、OffTime：若以寄存器 输入，则高字节为小时（0~23）低字节为分钟（0~59）；若是常数输入，输入格式为：hh:mm（如 8 点 5 分，输入 08:05）。
- 5、若 OnTime 为无效时间则 ON 动作无效。若 OffTime 为无效时间则 OFF 动作无效。

【指令示例】



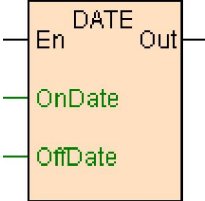
【程序说明】

Act=127 (01111111) 表示星期一~星期日都有效，当 M0=ON 时，TIME 指令执行，每天 8:30~16:30 之间 Y0=ON，其他时间 Y0=OFF。

DATE（日期开关）

指令格式及参数说明

语言	LD	FBD	IL
----	----	-----	----

语言	LD	FBD	IL
指令格式			DATE En, OnDate, OffDate, Out

参数	参数定义	输入	输出	说明
En	使能	√		
OnDate	ON 动作日期	√		
OffDate	OFF 动作日期	√		
Out	状态输出		√	

【指令功能及作用说明】

- 1、DATE 指令以年为控制周期，通过设定 ON、OFF 动作时间控制输出。
- 2、如果 ON 动作日期<OFF 动作日期，表示在本年度开和关。如果 ON 动作日期>OFF 动作日期，表示跨年度开和关。
- 3、OnDate、OffDate：若以寄存器输入，则高字节为月份（1~12）低字节为日期（1~31）；若 是常数输入，输入格式为：mm-dd（如 8 月 5 日，输入 08-05）。
- 4、若 OnDate 为无效日期则 ON 动作无效。若 OffDate 为无效日期则 OFF 动作无效。

【指令示例】

//Network 1 日期开关



【程序说明】

当 M0=ON 时，DATE 指令执行，在本年 8 月 1 日至明年 1 月 31 日之间 Y0=ON，其他时间 Y0=OFF。

INVT（倒计时）

指令格式及参数说明

语言	LD	FBD	IL
指令格式			INVT En, Clock, Out, Rtv

参数	参数定义	输入	输出	说明
En	使能	√		
Clock	倒计时时间起始元件	√		占用 6 个连续元件
Out	倒计时到输出		√	
Rtv	剩余时间		√	占用 4 个连续元件

【指令功能及作用说明】

- 1、INVT 指令根据设定的倒计时时间 Clock 计算出当前时间到设定时间之间还剩余的天（Rtv）、时（Rtv+1）、分（Rtv+2）、秒（Rtv+3），在计时时间到达后输出 Out。
- 2、Clock 占用 6 个寄存器，分别为年（0~9999）、月（1~12）、日（1~31）、时（0~23）、分（0~59）、秒（0~59）。如果年设定在 0~99 范围，系统会认为是 2000~2099 年。
- 3、如果 Clock 是无效时间，指令不执行。

【指令示例】



【程序说明】

- 1、假如 V1000=2013、V1001=12、V1002=25、V1003=8、V1004=0、V1005=0，表示设定 倒计时时间为 2013-12-25 8:0:0。
- 2、当 M0=ON，INVT 指令开始倒计时，自动计算出当前时间到 2013-12-25 8:0:0 之间还剩余的天、时、分、秒，在计时时间到达后输出 Y0=ON。

4.15 通讯指令

通讯指令列表

指令名称	8 位模式	32 位模式	指令功能	支持语言		
				LD	FBD	IL
SUM	SUM.LB		SUM 累加和校验	√	√	√
BCC	BCC.LB		BCC 校验	√	√	√
CRC	CRC.LB		CRC 校验	√	√	√
LRC	LRC.LB		LRC 校验	√	√	√
COMM	COMM.LB		串行通讯	√	√	√
MODR			Modbus 读	√	√	√
MODW			Modbus 写	√	√	√
HWRD			Haiwellbus 读	√	√	√
HWWR			Haiwellbus 写	√	√	√
RCV			接收通讯数据	√	√	√
XMT	XMT.LB		发送通讯数据	√	√	√
FROM			扩展模块 CR 数据读取	√	√	√
TO			扩展模块 CR 数据写入	√	√	√

注意：Haiwell（海为）PLC 支持标准 Modbus 协议、Haiwellbus 协议、自由协议通讯，PLC 作为从站时不需要写任何通讯程序，上位机（组态软件、触摸屏、文本显示器等 HMI）可以直接使用 Modbus 协议访问 Haiwell（海为）PLC。参见["通讯地址代码表"](#)

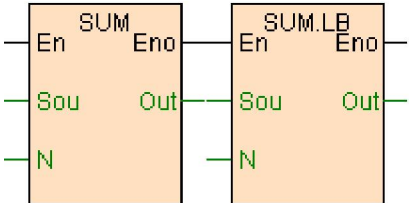
Haiwell（海为）PLC 通讯功能特点：

- 1、Haiwellbus 协议是一种高效、高速的通讯协议，它支持离散、混合数据类型的传输，通讯效率极高。
- 2、Haiwell（海为）PLC 支持最大 5 个通讯口，所有通讯口功能相同，都可以编程、上下载程序、监控、联网。

- 3、5 个通讯口完全独立，并行工作，支持不同波特率、不同协议格式、不同厂商的设备连在同一个网络。
- 4、无需担心通讯口冲突问题，无需控制通讯指令执行的时序，所有通讯指令可以并行得电运行。
- 5、通过通讯指令的 Out 端子可以直观判断通讯成功与否，可以用它实现对每个从机通讯失败的报警。
- 6、Haiwell（海为）PLC 的通讯在网络物理层面是零空闲的，如果从站设备无法支持如此高速的通讯，可以通过给 SV141 赋值使得通讯指令之间插入一个时间间隔。

SUM、SUM.LB（SUM 累加和校验）

指令格式及参数说明

语言	LD	FBD	IL
16、8 位 指令格式			SUM En, Sou, N, Out SUM.LB En, Sou, N, Out

参数	参数定义	输入	输出	说明
En	使能	√		
Sou	源起始元件	√		SUM：占用(N+1)\2 个连续元件，SUM.LB：占用 N 个连续元件
N	校验字节数	√		
Eno	使能输出		√	
Out	SUM 校验码输出		√	

【指令功能及作用说明】

SUM 累加和校验码的计算方法：将以 Sou 为起始的 N 个字节数据求累加和，取低字节，超过 256 部分溢出。SUM.LB 为低字节模式，只计算低字节的 SUM 累加和校验码，高字节不使用。

注：E/S 系列主机不支持 SUM 指令。

【指令示例】

//Network 1 计算SUM校验码



【程序说明】

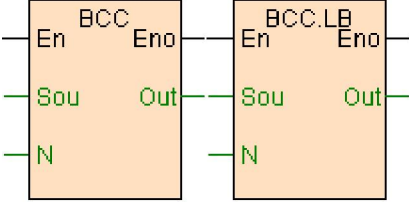
当 M0=ON，计算 SUM 校验码，如下表。

Sou 元件	初始值	SUM 输出	SUM.LB 输出
V1000	0x8181	V0=0xB3	V2=0x30
V1001	0x0152		

V1002	0x0000		
V1003	0x0153		
V1004	0x0D0A		

BCC、BCC.LB（BCC 校验）

指令格式及参数说明

语言	LD	FBD	IL
16、8 位 指令格式			BCC En, Sou, N, Out BCC.LB En, Sou, N, Out

参数	参数定义	输入	输出	说明
En	使能	√		
Sou	源起始元件	√		BCC：占用(N+1)\2 个连续元件，BCC.LB：占用 N 个连续元件
N	校验字节数	√		
Eno	使能输出		√	
Out	BCC 码输出		√	

【指令功能及作用说明】

BCC 校验码的计算方法：将以 Sou 为起始的 N 个字节数据进行异或运算。BCC.LB 为低字节模式，只计算低字节的 BCC 校验码，高字节不使用。

【指令示例】

//Network 1 计算BCC校验码



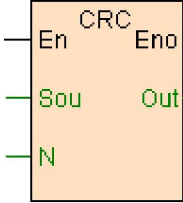
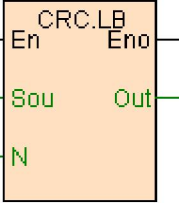
【程序说明】

当 M0=ON，计算 BCC 校验码，如下表。

Sou 元件	初始值	BCC 输出	BCC.LB 输出
V1000	0x8181	V0=0xB	V2=0x8A
V1001	0x0152		
V1002	0x0000		
V1003	0x0153		
V1004	0x0D0A		

CRC、CRC.LB（CRC 校验）

指令格式及参数说明

语言	LD	FBD	IL
16、8 位 指令格式			CRC En, Sou, N, Out CRC.LB En, Sou, N, Out

参数	参数定义	输入	输出	说明
En	使能	√		
Sou	源起始元件	√		CRC：占用(N+1)\2 个连续元件，CRC.LB：占用 N 个连续元件
N	校验字节数	√		
Eno	使能输出		√	
Out	CRC 码输出		√	

【指令功能及作用说明】

CRC 计算以 Sou 为起始的 N 个字节数据的 CRC 校验码。CRC.LB 为低字节模式，只计算低字节的 CRC 校验码，高字节不使用。

【指令示例】

//Network 1 计算CRC校验码



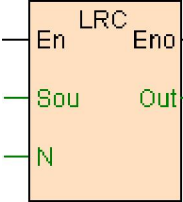
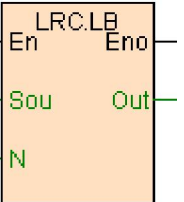
【程序说明】

当 M0=ON，计算 CRC 校验码，如下表。

Sou 元件	初始值	CRC 输出	CRC.LB 输出
V1000	0x8181	V0=0x98AC	V2=0xB4 V3=0x51
V1001	0x0152		
V1002	0x0000		
V1003	0x0153		
V1004	0x0D0A		

LRC、LRC.LB（LRC 校验）

指令格式及参数说明

语言	LD	FBD	IL
16、8 位 指令格式			LRC En, Sou, N, Out LRC.LB En, Sou, N, Out

参数	参数定义	输入	输出	说明
En	使能	√		
Sou	源起始元件	√		LRC：占用(N+1)\2 个连续元件，LRC.LB：占用 N 个连续元件
N	校验字节数	√		
Eno	使能输出		√	
Out	LRC 码输出		√	

【指令功能及作用说明】

LRC 校验码的计算方法：将以 Sou 为起始的 N 个字节数据求和再取其 2 的补码。LRC.LB 为低字节模式，只计算低字节的 LRC 校验码，高字节不使用。

【指令示例】

//Network 1 计算LRC校验码



【程序说明】

当 M0=ON，计算 LRC 校验码，如下表。

Sou 元件	初始值	LRC 输出	LRC.LB 输出
V1000	0x8181	V0=0xFE4D	V2=0xD0 V3=0xFE
V1001	0x0152		
V1002	0x0000		
V1003	0x0153		
V1004	0x0D0A		

COMM、COMM.LB（串行通讯）

指令格式及参数说明

语言	LD	FBD	IL
16、8 位 指令格式	COMM EnOut TxdErr TnRxd Rn Protocol Port	COMM.LB EnOut TxdErr TnRxd Rn Protocol Port	COMM En, Txd, Tn, Rn, Ptotocol, Port, Out, Err, Rxd COMM.LB En, Txd, Tn, Rn, Ptotocol, Port, Out, Err, Rxd

参数	参数定义	输入	输出	说明
En	使能	√		

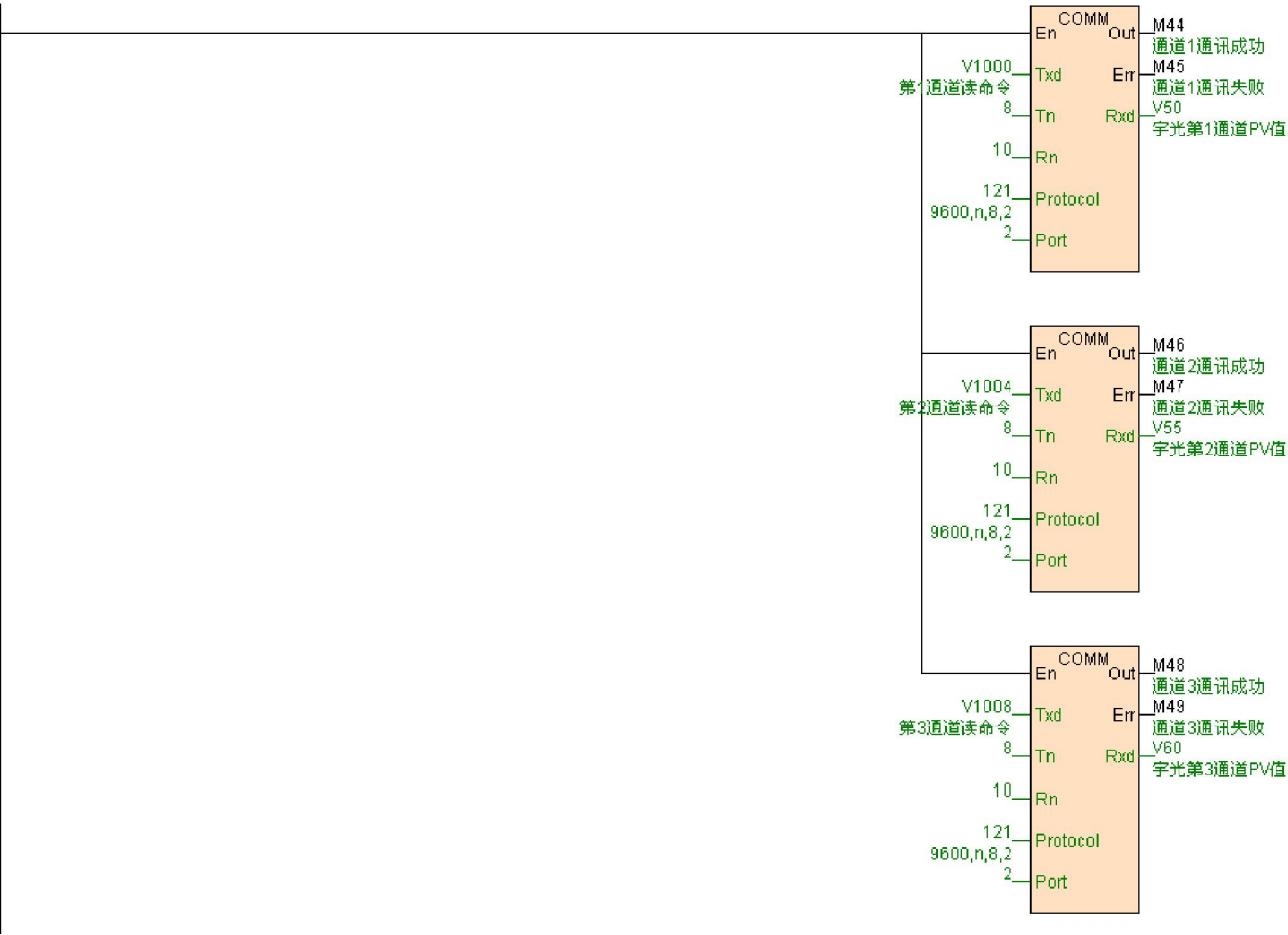
参数	参数定义	输入	输出	说明
Txd	发送数据起始元件	√		
Tn	发送数据字节数	√		
Rn	接收数据字节数	√		
Protocol	通讯协议	√		
Port	通讯端口	√		
Out	通讯完成输出		√	
Err	错误指示		√	
Rxd	接收数据起始元件		√	

【指令功能及作用说明】

- 1、当 PLC 与外部设备用自由协议进行通讯时，使用 COMM 指令发送与接收数据。此时 PLC 作为主站，外部设备作为从站。
- 2、当 Tn=0 时，COMM 指令只接收数据而不发送数据。当 Rn=0 时，COMM 指令只发送数据而不接收数据。当 Tn=Rn=0 时，COMM 指令不执行。
- 3、COMM 指令执行时，按 Protocol 定义的通讯格式（波特率、数据位、停止位、校验方式）将 Txd 为起始的 Tn 个字节数据发送到 Port 所指定的串口，发送完毕后，如果 Rn>0 则自动转为接收状态，接收完成 Out=ON，接收的数据放在 Rxd；如果 Rn=0 则不接收数据 Out=ON，系统执行下一条通讯指令。如果通讯指令未完成则 Err=ON。
- 4、COMM 指令有两种发送方式：高低字节发送方式(COMM)与仅发送低字节方式(COMM.LB)
- 5、COMM 指令可以与 XMT、MODR、MODW、HWRD、HWWR 指令同时使用。但不能与 RCV 指令使用同一个通讯口。

【指令示例】

//Network 1 读字光AI-708M的3路MV值：站号3-5、通信格式：9600,n,8,2，读出数据放于V50/V55/V60。读命令放于V1000~V1011



【程序说明】

1、根据 AI-708M 巡检仪的通讯协议，把读取 AI-708M 巡检仪的 3 个通道测量值的通讯命令放在初始寄存器值表“读 AI-708M 巡检仪命令”中：

元件	初始值	说明
V1000	0x8383	第 1 通道读命令
V1001	0x0152	
V1002	0x0000	
V1003	0x0155	
V1004	0x8484	第 2 通道读命令
V1005	0x0152	
V1006	0x0000	
V1007	0x0156	
V1008	0x8585	第 3 通道读命令
V1009	0x0152	
V1010	0x0000	
V1011	0x0157	

2、3 条 COMM 指令从母排取电一直执行，PLC 自动依次发送与 AI-708M 巡检仪的通讯命令并且将返回的测量值输出到指令的 Rxd。

3、无需担心通讯口冲突，无需控制通讯指令的执行时序，完全由系统自动完成。

MODR（Modbus 读）

指令格式及参数说明

语言	LD	FBD	IL
指令格式	MODR EnOut SlaveRxd Code Read N Protocol Port		MODR En, Slave, Code, Read, N, Ptotocol, Port, Out, Rxd

参数	参数定义	输入	输出	说明
En	使能	√		
Slave	从机设备地址	√		
Code	功能码	√		
Read	读数据起始地址	√		
N	数据个数	√		
Protocol	通讯协议	√		
Port	通讯端口	√		
Out	通讯完成输出		√	
Rxd	接收数据起始元件		√	占用 N 个连续元件

【指令功能及作用说明】

- 1、MODR 指令用于与所有支持 Modbus 协议的第三方设备进行通讯。
- 2、当 PLC 与外部设备用串口以 Modbus 协议进行相互通信时，用 MODR 指令读取外部设备的数据。此时 PLC 作为主站，外部设备作为从站。
- 3、MODR 指令无需写任何校验码，它自动对返回数据进行校验，校验正确 Out=ON，读取的数据放到 Rxd。
- 4、MODR 指令可以与 XMT、COMM、MODW、HWRD、HWWR 指令同时使用。但不能与 RCV 指令使用同一个通讯口。

【指令示例】

//Network 1 读远程模块的4个通道测量值，假如模块站地址为1，通讯格式19200,N,8,2 RTU格式，结果存放在v0~v3



【程序说明】

- 1、MODR 指令读取站号为 1 的远程模块（假如模块型号 S04AI）的 4 个通道测量值（CR 参数号 16~19），读取数据存放到 V0~V3。

2、不同型号的模块其 CR 号不尽相同，详细信息请参考[硬件手册](#)的["扩展模块参数"](#)部分。

MODW（Modbus 写）

指令格式及参数说明

语言	LD	FBD	IL
指令格式	MODW EnOut Slave Code Write Val N Protocol Port		MODW En, Slave, Code, Write, Val, N, Ptotocol, Port, Out

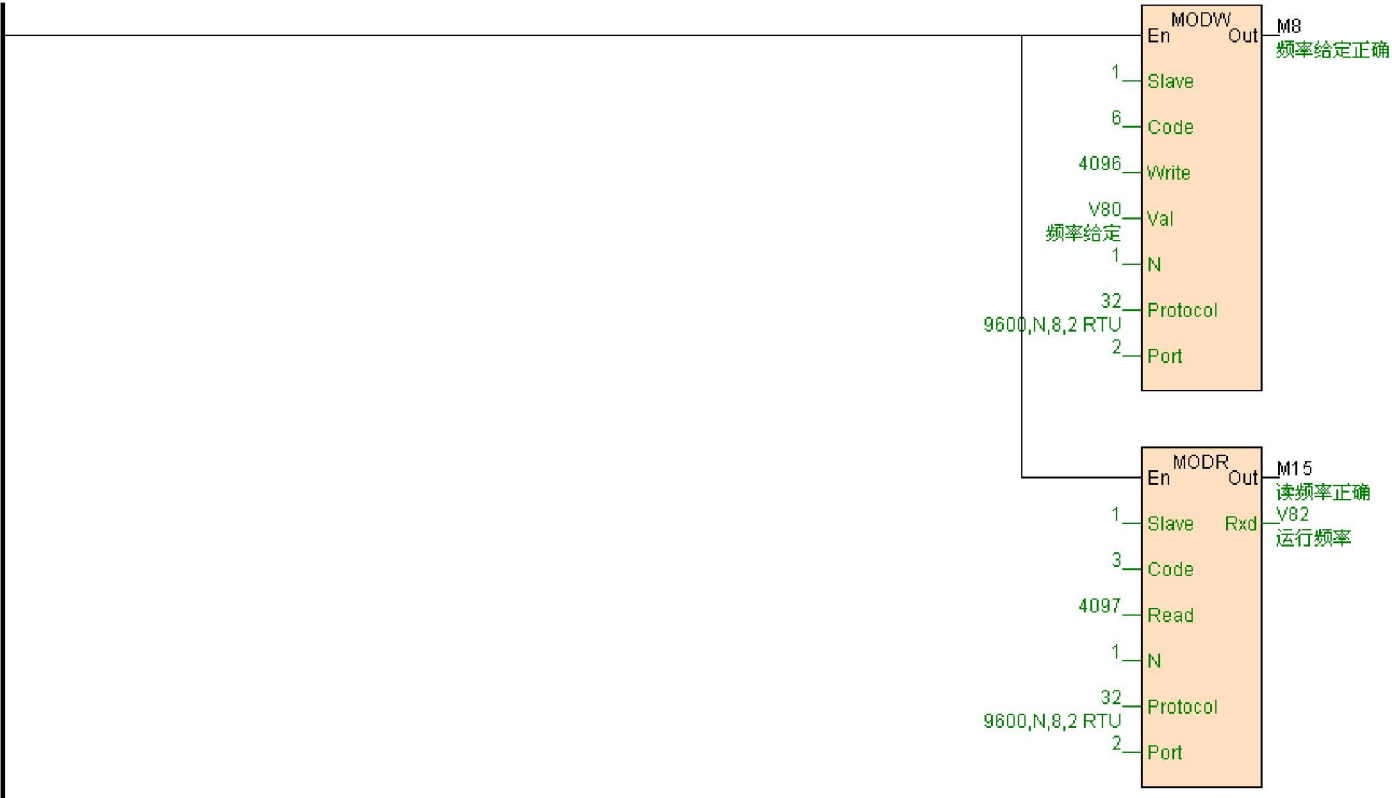
参数	参数定义	输入	输出	说明
En	使能	√		
Slave	从机设备地址	√		
Code	功能码	√		
Write	写入目标起始地址	√		
Val	欲写数据起始元件	√		占用 N 个连续元件
N	数据个数	√		
Protocol	通讯协议	√		
Port	通讯端口	√		
Out	通讯完成输出		√	

【指令功能及作用说明】

- 1、MODW 指令用于与所有支持 Modbus 协议的第三方设备进行通讯。
- 2、当 PLC 与外部设备用串口以 Modbus 协议进行相互通信时，用 MODW 指令把数据写到外部设备。此时 PLC 作为主站，外部设备作为从站。
- 3、MODW 指令无需写任何校验码，它自动对返回数据进行校验，校验正确 Out=ON 表示写成功。
- 4、MODW 指令可以与 XMT、COMM、MODR、HWRD、HWWR 指令同时使用。但不能与 RCV 指令使用同一个通讯口。

【指令示例】

#Network 1 与汇川变频器通信：站号为1，通信格式为9600,N,8,2 RTU。频率给定放于V80，读出的运行频率放在V82



【程序说明】

- 1、根据汇川变频器的通讯协议（请参考汇川变频器说明书的通讯部分），给定频率的 Modbus 地址为 4096，MODW 指令把 V80 的值实时写到变频器。
- 2、运行频率的 Modbus 地址为 4097，MODR 指令把变频器的当前频率读出存放到 V82 中。

HWRD（Haiwellbus 读）

指令格式及参数说明

语言	LD	FBD	IL
指令格式			HWRD En, Slave, Table, Port, Out

参数	参数定义	输入	输出	说明
En	使能	√		
Slave	从机设备地址	√		
Table	Haiwellbus 读通讯表	√		
Port	通讯端口	√		
Out	通讯完成输出		√	

【指令功能及作用说明】

- 1、HWRD 指令根据定义的“[Haiwellbus 读通讯表](#)”自动完成与从机的数据交换。

- 2、Haiwellbus 协议支持离散、混合数据类型的传输，通讯效率极高。
- 3、HWRD 指令可以与 XMT、COMM、MODR、MODW、HWWR 指令同时使用。但不能与 RCV 指令使用同一个通讯口。

【指令示例】



【程序说明】

- 1、定义 Haiwellbus 读通讯表“读 2#PLC 数据”如下：

序号	从机读出数据元件	主机写入数据元件
1	X0	M10
2	X3	M11
3	V11	V80
4	V12	V81
5	AI0	V20
6	AI1	V21

- 2、定义 Haiwellbus 写通讯表“写 2#PLC 数据”如下：

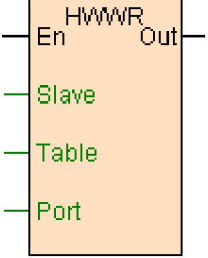
序号	主机读出数据元件	从机写入数据元件
1	X0	M100
2	X1	M101
3	V0	V100
4	V50	V102
5	Y4	M0
6	Y5	Y0
7	V60	V200
8	V61	V201

- 3、HWRD、HWWR 指令从母排取电一直执行，根据上面定义的 Haiwellbus 读（写）通讯表，自动与 2#PLC 完成数据交换。

HWWR（Haiwellbus 写）

指令格式及参数说明

语言	LD	FBD	IL
----	----	-----	----

语言	LD	FBD	IL
指令格式			HWWR En, Slave, Table, Port, Out

参数	参数定义	输入	输出	说明
En	使能	√		
Slave	从机设备地址	√		
Table	Haiwellbus 写通讯表	√		
Port	通讯端口	√		
Out	通讯完成输出		√	

【指令功能及作用说明】

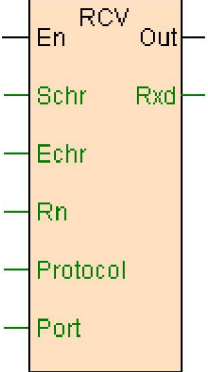
- 1、HWWR 指令根据定义的“[Haiwellbus 写通讯表](#)”自动完成与从机的数据交换。
- 2、Haiwellbus 协议支持离散、混合数据类型的传输，通讯效率极高。
- 3、HWWR 指令可以与 XMT、COMM、MODR、MODW、HWRD 指令同时使用。但不能与 RCV 指令使用同一个通讯口。

【指令示例】

参见 [HWRD](#) 指令示例。

RCV（接收通讯数据）

指令格式及参数说明

语言	LD	FBD	IL
指令格式			RCV En, Schr, Echr, Rn, Ptotocol, Port, Out, Rxd

参数	参数定义	输入	输出	说明
En	使能	√		
Schr	起始符	√		
Echr	结束符	√		
Rn	接收数据字节数	√		
Protocol	通讯协议	√		
Port	通讯端口	√		
Out	通讯完成输出		√	

参数	参数定义	输入	输出	说明
Rxd	接收数据起始元件		√	

【指令功能及作用说明】

- 1、在上位机作为通讯主站，PLC 作为通讯从站，并且上位机需要以自由协议方式与 PLC 通讯时才需要用到 RCV 指令。
- 2、RCV 指令被动接收上位机发送来的数据，如果需要给上位机答复则使用 XMT 指令发送答复数据。
- 3、Schr 为起始符定义，如果 Schr=0 表示没有起始符，如果 Schr 的高字节=0 低字节<>0 表示只有一个起始符 （如 Schr=0x003A，起始符是 0x3A），如果 Schr 的高字节<>0 表示 2 个起始符 （如 Schr=0x833A，起始符是 0x3A、0x83）。
- 4、Echr 为结束符定义，如果 Echr=0 表示没有结束符，如果 Echr 的高字节=0 低字节<>0 表示只有一个结束符（如 Echr=0x000D，结束符是 0x0D），如果 Echr 的高字节<>0 表示 2 个结束符（如 Echr=0x0A0D，结束符是 0x0D、0x0A）。
- 5、如果定义了起始符或者结束符，RCV 会根据起始符、结束符进行匹配，只有匹配正确才认为通讯成功，接收到的数据输出 Rxd；
当 Schr 和 Echr 都为 0 时，即起始符和结束符都没有定义，那么 RCV 指令将根据相应通讯口的通讯超时时间来判断通讯帧的起止。
当接收到新帧的第一个字节或者通讯超时 RCV 指令会自动复位 Out 和 Rxd。
- 6、Rn 为接收数据字节数，例如：RCV 指令需要接收上位机发送下来的 22 个字节内容，则指定 Rn=22。注意：如果上位机发送下来的命令是不固定长度，那么将接收字节数定义为 0，表示不计接收字节数。
- 7、一个通讯口只能使用一条 RCV 指令，并且不能与 COMM、MODR、MODW、HWRD、HWWR 指令使用同一个通讯口。

注：E/S 系列主机不支持 RCV 指令。

【指令示例】

//Network 1 RCV接收上位机发送来的数据



//Network 2 如果接收数据完成，根据接收到命令不同，给上位机返回不同数值



【程序说明】

- 1、Network 1 启动一条 RCV 被动接收通讯指令,起始符匹配 0x3A,结束符匹配 0x0D 0x0A,接收 4 个字节数据,通讯格式 19200,n,8,2，通讯口 2，接收数据放在 V0V1，接收数据正确则每次通讯 M0=ON。
- 2、当 M0=ON 接收数据正确，比较第 2 字节内容（V0 的高字节），如果为 1 则执行 XMT 指令把 V1000V1001 返回，如果为 2 则执行 XMT 指令把 V100~V102 返回，如果为 3 则执行 XMT 指令把 V200~V203 返回。

XMT、XMT.LB（发送通讯数据）

指令格式及参数说明

语言	LD	FBD	IL
16、8 位指令格式			XMT En, Txd, Tn, Ptotocol, Port, Out XMT.LB En, Txd, Tn, Ptotocol, Port, Out

参数	参数定义	输入	输出	说明
En	使能	√		
Txd	发送数据起始元件	√		
Tn	发送数据字节数	√		
Protocol	通讯协议	√		
Port	通讯端口	√		
Out	通讯完成输出		√	

【指令功能及作用说明】

- 1、在上位机作为通讯主站，PLC 作为通讯从站，并且上位机需要以自由协议方式与 PLC 通讯时才需要用到 XMT 指令。
- 2、XMT 指令一般用于配合 RCV 指令使用，在 RCV 指令接收到上位机发送来的数据，如果需要给上位机答复则使用 XMT 指令发送答复数据。
- 3、XMT 指令有两种 发送方式：高低字节发送方式(XMT)与仅发送低字节方式(XMT.LB)。
- 4、XMT 指令可以重复，XMT 指令与 COMM 指令不同，它只能发送数据而不接收数据。

注：E/S 系列主机不支持 XMT 指令。

【指令示例】

参见 [RCV](#) 指令示例。

FROM（读取扩展模块 CR 数据）

指令格式及参数说明

语言	LD	FBD	IL
指令格式			FROM En, Slot, CR, N, Out

参数	参数定义	输入	输出	说明
En	使能	√		
Slot	模块位置号	√		
Cr	欲读取 CR 号	√		
N	读取 CR 个数	√		
Eno	使能输出		√	
Out	返回数据起始元件		√	占用 N 个连续元件

【指令功能及作用说明】

- 1、FROM 用于通过并行总线在程序中读取扩展模块的参数值。
- 2、PLC 会自动分配并行总线上扩展模块的 IO 通道对应的元件地址，并且实时更新扩展模块的 IO，所以一般不需要使用 FROM 指令。
详细参考“[PLC 硬件配置](#)”部分。
- 3、不同型号的扩展模块其 CR 号不尽相同，详细信息请参考[硬件手册](#)的“[扩展模块参数](#)”部分。

【指令示例】

//Network 1 读取S04AI模块的4个输入通道的测量值，第一通道测量值的参数号是0x10（十进制16）



//Network 2 把S04AI模块通道1的零点修正值（参数号为0x25，十进制37）改为30



【程序说明】

- 1、当 M0=ON，读取第 1 个扩展模块（假如模块型号 S04AI）的 4 个输入通道测量值。
- 2、当 M1=ON，把第 1 个扩展模块的通道 1 的零点修正值改为 30（假如 V1000=30）。

TO（扩展模块 CR 数据写入）

指令格式及参数说明

语言	LD	FBD	IL
指令格式			TO En, Slot, CR, Val, N

参数	参数定义	输入	输出	说明
En	使能	√		
Slot	模块位置号	√		
Cr	欲写入 CR 号	√		
Val	欲写数据起始元件	√		占用 N 个连续元件
N	写入 CR 个数	√		
Eno	使能输出		√	

【指令功能及作用说明】

- 1、TO 用于通过并行总线在程序中去修改扩展模块的参数。可以在“PLC 硬件配置”窗口配置模块参数，当 PLC 程序下载时会自动下载，所以一般不需要使用 TO 指令。
- 2、不同型号的扩展模块其 CR 号不尽相同，详细信息请参考[硬件手册](#)的“[扩展模块参数](#)”部分。

【指令示例】

参见 [FROM](#) 指令示例。

4.16 中断指令

中断指令列表

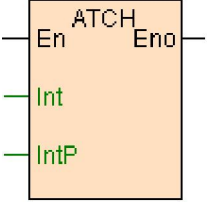
指令名称	8 位模式	32 位模式	指令功能	支持语言		
				LD	FBD	IL

指令名称	8 位模式	32 位模式	指令功能	支持语言		
				LD	FBD	IL
ATCH			中断绑定	√	√	√
DTCH			中断释放	√	√	√
ENI			允许中断	√	√	√
DISI			禁止中断	√	√	√

- 1、Haiwell（海为）PLC 提供多达 52 个系统中断资源，包括脉冲输出、边沿捕捉、高速计数和计时器中断。参见“[系统中断表](#)”。
- 2、中断程序只有用 ATCH 指令绑定到相应的中断，并且在系统产生该中断时才被执行一次，其他任何时候都不会被执行。参见“[中断程序](#)”。
- 3、中断程序应尽量短小，只有必要指令放在中断程序执行。

ATCH（中断绑定）

指令格式及参数说明

语言	LD	FBD	IL
指令格式			ATCH En, Int, IntP

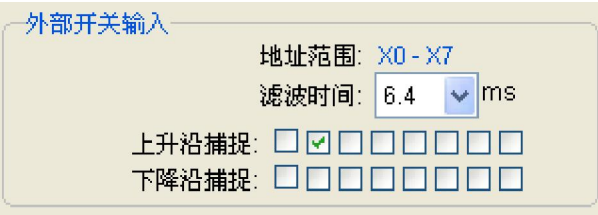
参数	参数定义	输入	输出	说明
En	使能	√		
Int	中断号	√		
IntP	中断程序名称	√		
Eno	使能输出		√	

【指令功能及作用说明】

ATCH 指令将 IntP 中断程序与 Int 中断号绑定，当系统产生 Int 中断时，自动执行绑定的 IntP 中断程序，指令一般边沿执行。

【指令示例】

- 1、在“[PLC 硬件配置](#)”窗口配置“[X 开关量输入参数](#)”打开 X1 的上升沿捕捉功能。

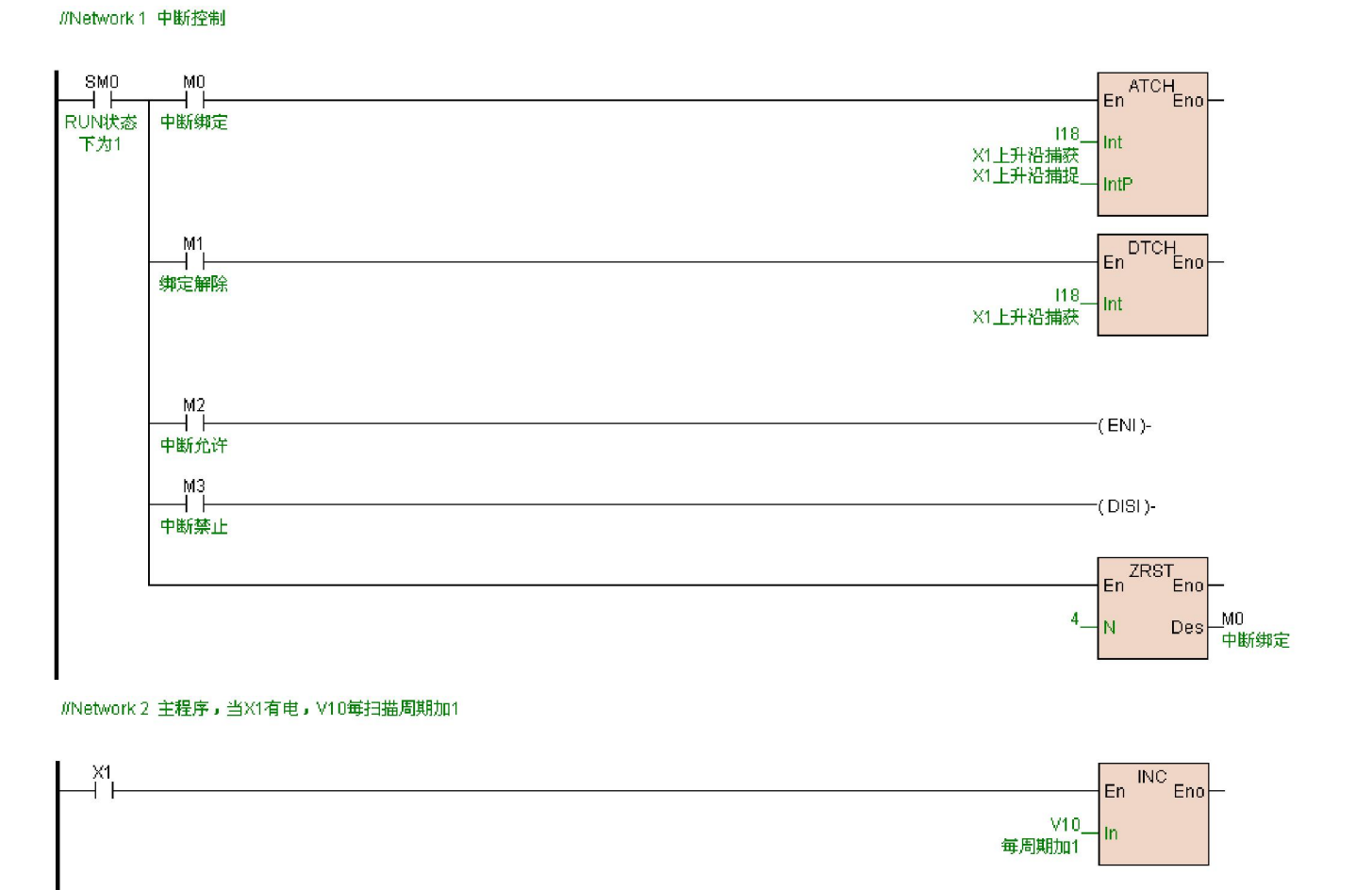


- 2、编写一个中断程序“X1 上升沿捕捉”，仅一条指令，V0 加 1，如下。

//Network 1



3、编写主程序“ATCH”如下：



【程序说明】

- 1、M0=ON，将中断程序“X1 上升沿捕捉”与中断号 I18（X1 上升沿中断）绑定，这样当 X1 由 OFF 变为 ON 时，系统产生 I18 中断并且调用中断程序“X1 上升沿捕捉”。
- 2、M1=ON，解除中断号 I18 的绑定，这时即使系统产生 I18 中断也不会执行中断程序。
- 3、M2=ON，允许系统中断，这样当 X1 由 OFF 变为 ON 时，系统产生 I18 中断。
- 4、M3=ON，禁止系统中断，这样系统不会产生任何中断。

DTCH（中断释放）

指令格式及参数说明

语言	LD	FBD	IL
指令格式			DTCH En, Int

参数	参数定义	输入	输出	说明
En	使能	√		
Int	中断号	√		
Eno	使能输出		√	

【指令功能及作用说明】

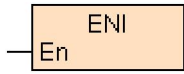
DECH 指令解除 Int 中断号绑定的中断程序，指令一般边沿执行。

【指令示例】

参见 [ATCH](#) 指令示例。

ENI（允许中断）

指令格式及参数说明

语言	LD	FBD	IL
指令格式	—(ENI)—		ENI En

参数	参数定义	输入	输出	说明
En	使能	√		

【指令功能及作用说明】

ENI 指令开启系统自动功能，系统默认为开启中断，指令一般边沿执行。

【指令示例】

参见 [ATCH](#) 指令示例。

DISI（禁止中断）

指令格式及参数说明

语言	LD	FBD	IL
指令格式	—(DISI)—		DISI En

参数	参数定义	输入	输出	说明
En	使能	√		

【指令功能及作用说明】

DISI 指令屏蔽系统中断功能，指令执行后将不再产生系统中断，指令一般边沿执行。

【指令示例】

参见 [ATCH](#) 指令示例。

4.17 程序控制指令

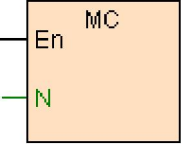
程序控制指令列表

指令名称	8 位模式	32 位模式	指令功能	支持语言		
				LD	FBD	IL
MC			主控	√	√	√
MCR			主控清除	√	√	√
FOR			循环指令	√	√	√
NEXT			循环结束	√	√	√
WAIT			延时等待	√	√	√
CALL			调用子程序	√	√	√
EXIT			条件返回	√	√	√
REWD			扫描时间复位	√	√	√

指令名称	8 位模式	32 位模式	指令功能	支持语言		
				LD	FBD	IL
JMPC			条件跳转	√	√	√
LBL			跳转标号	√	√	√

MC（主控指令）

指令格式及参数说明

语言	LD	FBD	IL
指令格式			MC En, N

参数	参数定义	输入	输出	说明
En	使能	√		
N	标号	√		

【指令功能及作用说明】

- 1、MC 为主控起始指令，N 为主控号，当 En=ON 时，主控 N 内（位于 MC N 与 MCR N 指令之间）的指令照常执行。
- 2、当 En=OFF 时，主控 N 内（位于 MC 与 MCR 指令之间）的指令被跳过，并且同时复位其中的 OUT 指令输出、计时器线圈 T 及当前值 TV、计数器线圈 C 及当前值 CV。
- 3、MC-MCR 指令支持嵌套结构，最多可达 8 层。

注：E/S 系列主机不支持 MC 指令。

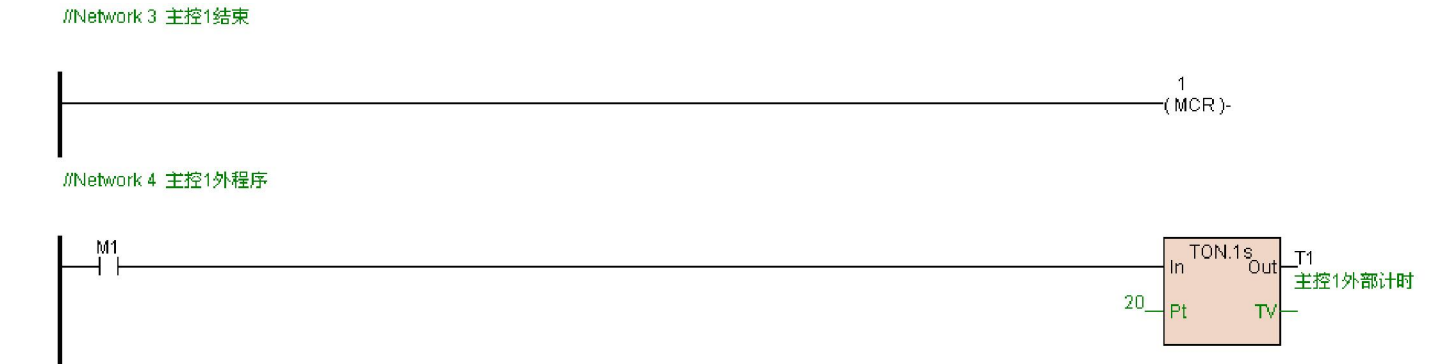
【指令示例】

//Network 1 主控1开始



//Network 2 主控1内程序





【程序说明】

- 1、Network 1 定义主控 1 开始，Network 3 定义主控 1 结束。
- 2、当 M0=ON，主控 1 内程序（Network 2）正常执行，当 M1=ON，Y0=ON，计时器 T0 开始计时，计数器 C0 加 1，CV0=1，C0=ON。
主控 1 外部 Network 4 的计时器 T1 启动计时。
- 3、当 M0=OFF，则主控 1 内程序（Network 2）的指令被跳过，并且复位 Y0=OFF、T0=OFF、TV0=0、C0=OFF、CV0=0。主控 1 外部 Network 4 的计时器 T1 继续计时。

MCR（主控指令清除）

指令格式及参数说明

语言	LD	FBD	IL
指令格式	N —(MCR)—	MCR N	MCR N

参数	参数定义	输入	输出	说明
N	标号	√		

【指令功能及作用说明】

MCR 为主控结束指令，N 为主控号，只能与 MC 指令配对使用。

注：E/S 系列主机不支持 MCR 指令。

FOR（循环指令）

指令格式及参数说明

语言	LD	FBD	IL
指令格式	FOR EnIndexInitFinalEno		FOR En, Index, Init, Final

参数	参数定义	输入	输出	说明
En	使能	√		
Index	循环索引	√		
Init	起始值	√		

参数	参数定义	输入	输出	说明
Final	终止值	√		
Eno	使能输出		√	

【指令功能及作用说明】

- 1、FOR 指令和 NEXT 指令必须成对使用，其中 FOR 标记了循环的开始，NEXT 标记了循环的结束。在 FOR 和 NEXT 之间的语句被称为循环体。
- 2、FOR/NEXT 指令反复执行循环体内的语句直至达到指定的循环次数，其中，循环次数的计数值存放在参数 Index 内，而 Init 指定了 Index 的初始值，Final 指定了 Index 的终值。
- 3、一个循环体内又包含另一个完整的循环结构，称为循环的嵌套。FOR/NEXT 循环支持嵌套，嵌套深度最大为 8 层。
- 4、在循环体内，用户可以改变终值 Final，从而改变循环的结束条件。
- 5、若循环次数比较多，则程序的执行时间可能就会超过系统看门狗的定时值，从而可能会触发系统的看门狗动作引起保护停机，这时就必须在循环内增加 **REWD** 指令复位看门狗。

【指令示例】

//Network 1 循环外部



//Network 2 循环控制



//Network 3 循环内部程序



【程序说明】

- 1、M0=ON，每个扫描周期 V10 加 1，Network 2 启动一个循环，循环索引 V0，循环次数 30（0~29），Network 3 循环内指令，V2 每次加 1（一个循环执行完成则加 30），这样 V2 一定是 V10 的 30 倍。
- 2、REWD 指令复位看门狗，NEXT 为循环结束定义。

NEXT（循环结束）

指令格式及参数说明

语言	LD	FBD	IL
指令格式	—(NEXT)—	NEXT	NEXT

【指令功能及作用说明】

定义一个循环的结束，只能与 **FOR** 指令配对使用。

WAIT（延时等待）

指令格式及参数说明

语言	LD	FBD	IL
指令格式	—En— —Tms— —Eno—	WAIT En, Tms	

参数	参数定义	输入	输出	说明
En	使能	√		
Tms	延时时间	√		单位 0.1ms
Eno	使能输出		√	

【指令功能及作用说明】

- 1、WAIT 指令使 PLC 暂停程序的执行，暂停时间由参数 Tms 设定。
- 2、WAIT 指令延长了扫描周期，可能会触发看门狗动作，应尽量不使用。

CALL（调用子程序）

指令格式及参数说明

语言	LD	FBD	IL
指令格式	—En— —SubP— —Par1— —Par2— —Par3— —Par4— —Par5— —Par6— —Par7— —Par8—	CALL En, SubP (0-8 个输入参数名称) (0-3 个输出参数名称)	

参数	参数定义	输入	输出	说明
En	使能	√		
SubP	子程序名称	√		
Par1~8	输入参数（可不带参数）	√		

参数	参数定义	输入	输出	说明
Eno	使能输出		√	
Out1~3	输出 参数（可不带参数）		√	

【指令功能及作用说明】

- 1、CALL 指令用于调用子程序。对子程序的调用可带参数或不带参数，利用编程软件的新建子程序窗口或子程序的属性窗口，可对该子程序的参数进行修改与编辑；当 SubP 参数端输入子程序名称后，编程软件将该子程序对参数的定义情况给出用于调用该子程序所需的输入、输出参数端脚及参数名称，指令的输入输出参数端允许输入的数据类型与子程序定义的参数数据类型一致。
- 2、用 CALL 调用子程序时，若被调用的子程序未定义参数，则为无参数调用，CALL 指令的输入参数端与输出参数端省略。对于带参数调用子程序时，参数是以按顺序一对一的方式传递与赋值的。
- 3、使用 CALL 指令达到调用子程序的程序称为属主程序，子程序可以重复调用，支持子程序嵌套调用，最大嵌套 8 级。
- 4、子程序的参数说明：

A、一个子程序最多可定义 8 个输入（IN）或输入输出（IN_OUT）参数及 3 个输出（OUT）参数；

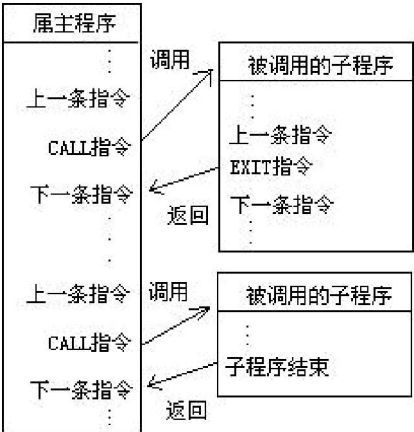
B、每个参数都有其参数类型（IN-输入参数，OUT-输出参数，IN_OUT—输入输出参数）与数据类型，在用 CALL 指令对子程序调用时，指令的输入输出参数端允许输入的数据类型要与子程序定义的参数数据类型一致。
- 5、子程序参数的参数类型如下表所示：

参数类型	含义
IN	输入参数，由 CALL 指令将参数数据传入子程序
IN_OUT	输入输出参数，先由 CALL 指令将参数数据传入子程序，子程序在将执行完成后再将结果传回 CALL 指令参数
OUT	输出参数，子程序在将执行完成后再将结果传回 CALL 指令参数

- 6、子程序的数据类型如下：

数据类型	含义
BOOL	又称布尔类型，ON 或 OFF
WORD	16 位字，可表达的数据范围为-32768~32767
DWORD	双字，32 位，可表达的数据范围为-2147483648~2147483647，占用 2 个连续元件
INT	整数，16 位，可表达的数据范围为-32768~32767
DINT	双字，32 位，可表达的数据范围为-2147483648~2147483647，占用 2 个连续元件
REAL	单精度浮点数，32 位，占用 2 个连续元件

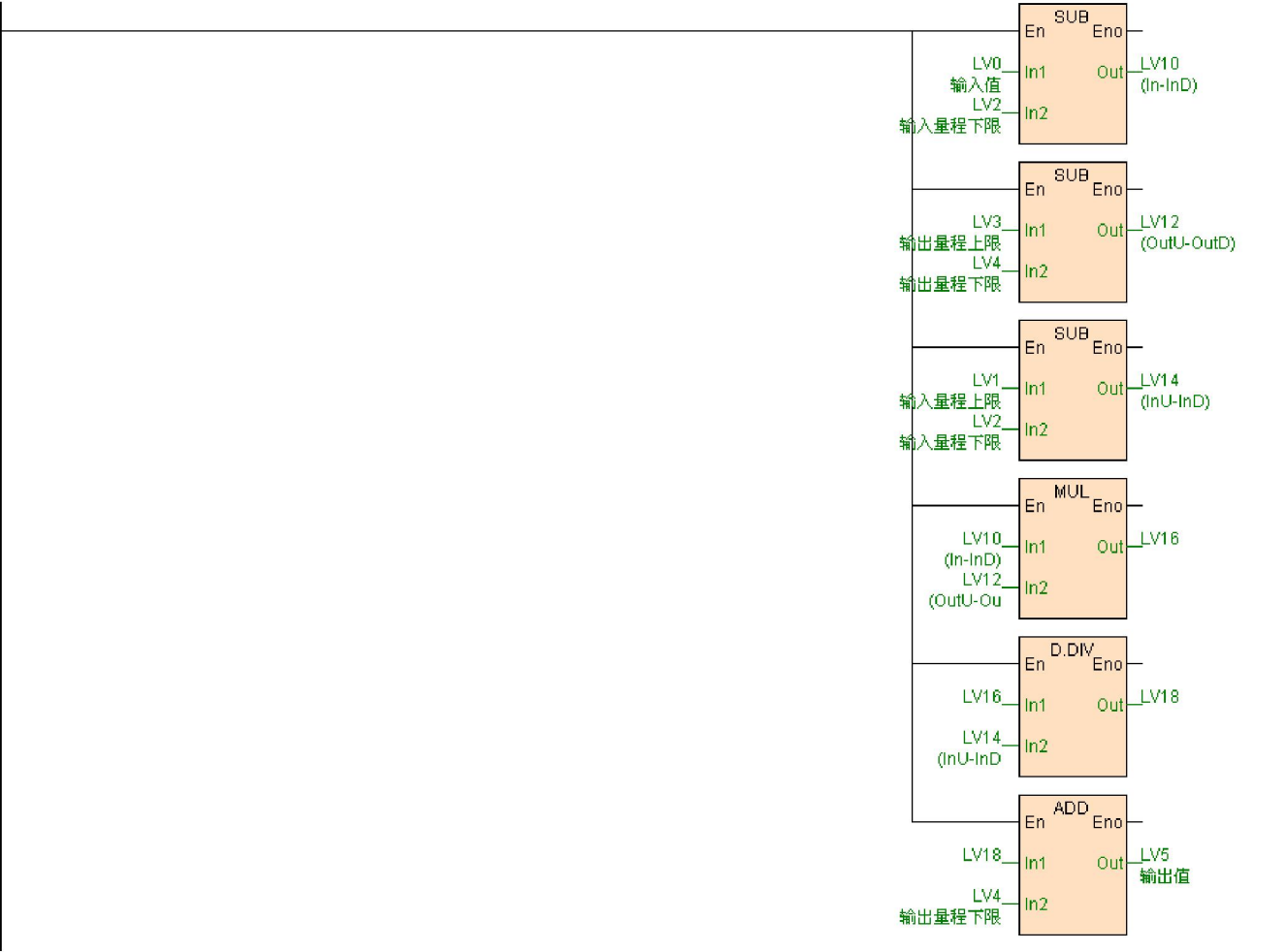
- 7、调用子程序时 PLC 程序的执行顺序示意图如下：



【指令示例】

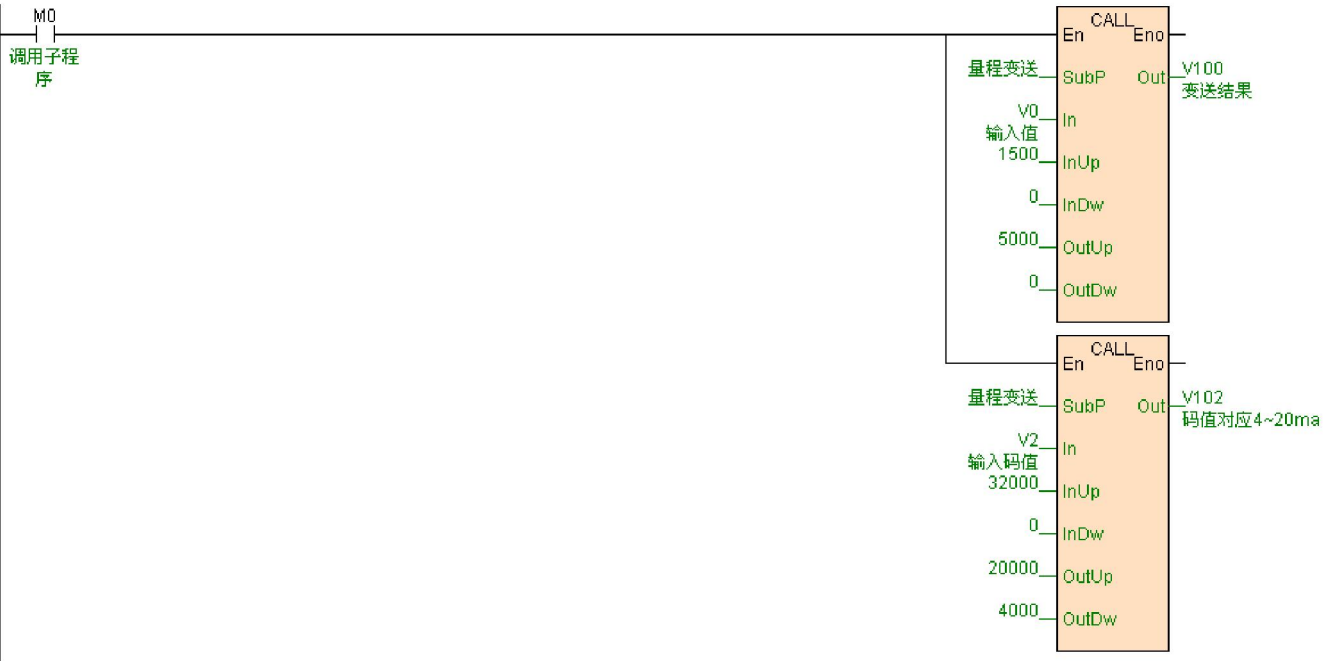
- 1、编写一个子程序“量程变送”，这个子程序实现 **SC** 指令相同的功能，如下。

//Network 1 Out = (In - InDw) * (OutUp- OutDw) / (InUp- InDw) + OutDw



2、编写主程序“CALL”如下：

//Network 1 把V0从0~1500变送到0~5000输出，把V2从0~32000变送到4000~20000输出



【程序说明】

- 子程序带 4 个输入参数和 1 个输出参数，计算 $Out = (In - InDw) * (OutUp - OutDw) / (InUp - InDw) + OutDw$ 。
- 主程序当 $M0=ON$ ，2 条 CALL 指令调用同一个子程序“量程变送”，因为传入参数值不同，实现不同量程的变送功能，如下图：

参数	参数值	调用子程序结果
InUp	1500	当 V0=0, 则 V100 =0 当 V0=100, 则 V100 =333 当 V0=1200, 则 V100 =4000 当 V0=1250, 则 V100 =4166
InDw	0	
OutUp	5000	
OutDw	0	
InUp	32000	当 V2=0, 则 V102 =4000 当 V2=1600, 则 V102 =4800 当 V2=16000, 则 V102 =12000 当 V2=28000, 则 V102 =18000
InDw	0	
OutUp	20000	
OutDw	4000	

EXIT（条件退出）

指令格式及参数说明

语言	LD	FBD	IL
指令格式	—(EXIT)-		EXIT En

参数	参数定义	输入	输出	说明
En	使能	√		

【指令功能及作用说明】

EXIT 为条件退出指令，它仅在需要从子程序或中断程序中提前退出时使用，正常退出子程序或中断程序无需增加该指令。

REWD（扫描时间复位）

指令格式及参数说明

语言	LD	FBD	IL
指令格式	—(REWD)-		REWD En

参数	参数定义	输入	输出	说明
En	使能	√		

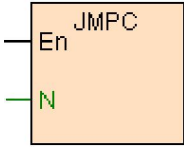
【指令功能及作用说明】

- 1、REWD 指令用于复位看门狗。
- 2、当程序在某种情况下需要循环执行，因此可能会触发系统的看门狗动作引起保护停机，这时就必须在循环内增加 REWD 指令复位看门狗。
- 3、REWD 指令使看门狗失去作用，应尽量不使用。

JMPC（条件跳转）

指令格式及参数说明

语言	LD	FBD	IL
----	----	-----	----

语言	LD	FBD	IL
指令格式			JMPC En, N

参数	参数定义	输入	输出	说明
En	使能	√		
N	跳转标号	√		

【指令功能及作用说明】

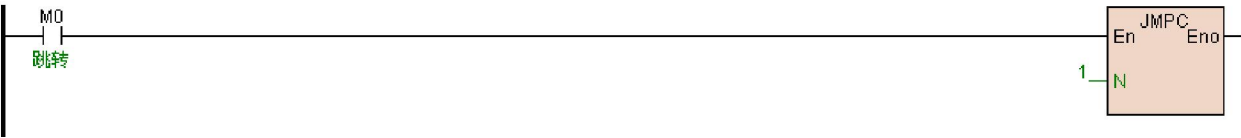
- 1、JMPC 为有条件跳转指令，当 En=ON 时，程序将跳转到标号 N 指定位置的下一条指令继续执行，如果 En=OFF 则不跳转，按照原来指令顺序执行。
- 2、N 跳转标号须用 LBL 指令定义，若标号不存在，则本指令不执行。

【指令示例】

//Network 1 正常程序



//Network 2 M0=ON时跳转



//Network 3 定义跳转标号1

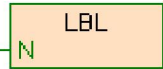


【程序说明】

- 1、Network 1 的 INC 指令从母排取电，每个扫描周期将 V0 加 1。
- 2、当 M0=OFF，JMPC 指令不执行，Network 3 的 INC 指令从母排取电，每个扫描周期将 V2 加 1。
- 3、当 M0=ON，JMPC 指令执行，跳转到标号 1 的后面继续执行，这时 V2 不会被加 1。

LBL（跳转标号）

指令格式及参数说明

语言	LD	FBD	IL
指令格式	N —(LBL)—		LBL N

参数	参数定义	输入	输出	说明
N	跳转标号	√		

【指令功能及作用说明】

LBL 指令用于定义跳转标号，供 **JMPC** 指令使用。

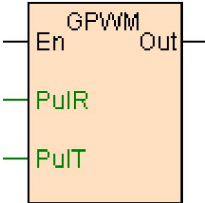
4.18 特殊功能指令

特殊功能指令列表

指令名称	8 位模式	32 位模式	指令功能	支持语言		
				LD	FBD	IL
GPWM			通用脉宽调制	√	√	√
FTC			模糊温度控制	√	√	√
PID			PID 控制	√	√	√
HAL		D.HAL	上限报警	√	√	√
LAL		D.LAL	下限报警	√	√	√
LIM		D.LIM	范围限制	√	√	√
SC		D.SC	线性变换	√	√	√
VC			阀门控制	√	√	√
TTC			温度曲线控制	√	√	√

GPWM（通用脉宽调制）

指令格式及参数说明

语言	LD	FBD	IL
指令格式			GPWM En, PulR, PulT, Out

参数	参数定义	输入	输出	说明
En	使能	√		
PulR	脉冲占空比	√		单位 0.1%，范围 0~1000
PulT	脉冲输出周期	√		单位 ms
Out	脉宽调制输出		√	

【指令功能及作用说明】

- 1、GPWM 指令使用普通晶体管输出点 Yn 输出可变脉冲周期、占空比的脉冲。
- 2、最大脉冲周期 32767，当 PulT≤0 不输出脉冲。
- 3、当 PulT>0，若 PulR>0 并且 PulR<1000 时，Out 输出占空比为 PulR、周期为 PulT 的脉冲，若 PulR=0 则 Out 输出低电平，若 PulR≥1000 则 Out 输出高电平信号。
- 4、PulR、PulT 的值可实时修改。

注：E/S 系列主机的 GPWM 指令参见“**GPWM**”。

【指令示例】



【程序说明】

- 1、当 M0=ON，从 Y20 输出占空比为 30%、周期为 30ms 的脉冲信号。
- 2、当 M0=OFF，脉冲输出停止。

FTC（模糊温度控制）

指令格式及参数说明

语言	LD	FBD	IL
指令格式			FTC En, PV, SV, Out, MV

参数	参数定义	输入	输出	说明
En	使能	√		
PV	测量值	√		单位 0.1℃
SV	设定值	√		单位 0.1℃
Out	脉宽调制输出		√	
MV	控制输出		√	

【指令功能及作用说明】

- 1、FTC 指令是用于模糊温度控制的专用指令，该指令不需要设置任何参数完全自动调节输出，使用简单。
- 2、指令提供 2 种输出方式，Out 为脉宽调制输出（频率 1Hz），MV 为控制输出（范围 0~1000）。
- 3、En 为指令的使能端，当 En=ON 时指令执行；当 En=OFF 时指令停止执行，Out=OFF，MV=0。
- 4、影响温度控制效果的外部因素很多，比如：传感器精度、传感器安装位置、加热功能大小等，如果 FTC 控制温度效果不理想，可以使用 PID 或者自己编写经验算法控制。

【指令示例】



【程序说明】

当 M0=ON，FTC 指令执行，根据 AI0 与 V1000 的偏差用模糊控制算法控制输出。当 M0=OFF，则 Y3=OFF，V100=0。

PID（PID 控制）

指令格式及参数说明

语言	LD	FBD	IL
指令格式			
			PID En, Act, PV, SV, P, I, D, T, Span, PVH, PVL, MVH, MVL, MV

参数	参数定义	输入	输出	说明
En	使能	√		
Act	控制方式	√		OFF-反作用，ON-正作用
PV	测量值	√		
SV	设定值	√		
P	比例系数	√		单位%
I	积分时间	√		单位 10ms
D	微分时间	√		单位 10ms
T	取样周期	√		单位 10ms
Span	调节死区	√		
PVH	PV 量程上限	√		
PVL	PV 量程下限	√		
MVH	MV 量程上限	√		
MVL	MV 量程下限	√		
Eno	使能输出		√	
MV	控制输出		√	

【指令功能及作用说明】

- 1、Act 为控制方式，Act=OFF 为反作用（PV 增大时，MV 输出趋向减小，一般用于加热控制），Act=ON 为正作用（PV 增大时，MV 输出趋向增大，一般用于冷却控制）。
- 2、En 为指令的使能端，当 En=ON 时指令执行；当 En=OFF 时指令停止执行，MV=0。
- 3、影响控制效果的外部因素很多，需要反复调整 P、I、D 这 3 个参数以满足控制对象，有经验者也可以自己编写经验算法控制。

4、P、I、D 三者的参数将直接影响到实际控制效果的好坏，下面是三个参数在 PID 控制中的作用：

- 比例系数（P）调节作用：为一百分比数据。是按比例反应系统的偏差，系统一旦出现了偏差，比例调节立即产生调节作用用以减少偏差。比例作用大，可以加快调节，减少误差，但是过大的比例，使系统的稳定性下降，甚至造成系统的不稳定。
- 积分时间（I）调节作用：是使系统消除稳态误差，提高无差度。因为有误差，积分调节就进行，直至无差，积分调节停止，积分调节输出一常值。积分作用的强弱取决于积分时间 I，I 越大，积分作用就越强，调节回应就越慢。反之 I 小则积分作用弱，调节响应就越快，加入积分调节可使系统稳定性下降，动态回应变慢。积分作用常与另两种调节规律结合，组成 PI 调节器或 PID 调节器。当积分时间设定为 0 时，不积分。
- 微分时间（D）调节作用：微分作用反映系统偏差信号的变化率，具有预见性，能预见偏差变化的趋势，因此能产生超前的控制作用，在偏差还没有形成之前，已被微分调节作用消除。因此，可以改善系统的动态性能。在微分时间选择合适情况下，可以减少超调，减少调节时间。微分作用对噪声干扰有放大作用，因此过强的微分调节，对系统抗干扰不利。此外，微分反应的是变化率，而当输入没有变化时，微分作用输出为零。微分作用不能单独使用，需要与另外两种调节规律相结合，组成 PD 或 PID 控制器。当微分时间设定为 0 时，不微分。

5、增量式 PID 算法：

$$\begin{aligned}\Delta u(n) &= K_p * \Delta e(n) + K_i * e(n) + K_d (\Delta e(n) - \Delta e(n-1)) \\ &= K_p * \Delta e(n) + K_i * e(n) + K_d * (e(n) - 2 * e(n-1) + e(n-2))\end{aligned}$$

$$u(n) = u(n-1) + \Delta u(n) \text{ 反作用}$$

$$u(n) = u(n-1) - \Delta u(n) \text{ 正作用}$$

其中：u(n)：MV 输出 MV

u(n-1)：上次输出

$\Delta u(n)$ ：输出增量

$K_p = P / 100$ ：比例系数

$K_i = K_p * T / I$ ：积分系数

$K_d = D / T$ ：微分系数

$e(n) = (SV - PV)$ ：偏差

$e(n-1)$ ：上次偏差

$e(n-2)$ ：上上次偏差

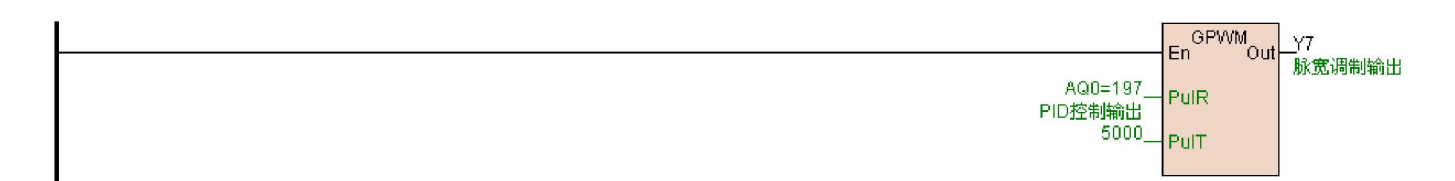
注：E/S 系列主机的 PID 指令参见“[PID](#)”。

【指令示例】

//Network 1 因为AI0的量程是0~1500，所以PVH=1500 PVL=0，AQ0的量程是0~1000表示百分比输出（精度0.1），所以MVH=1000 MVL=0



//Network 2 如果需要将PID指令的模拟量输出变为开关量占空比输出，使用GPWM指令即可，PulT=5000，表示脉冲周期5秒



- 【程序说明】
- 1、当 M0=ON，PID 指令执行，根据 AI0 与 V1000 的偏差通过 PID 算法控制输出。当 M0=OFF，指令不执行，AQ0=0。
 - 2、如果需要将 PID 指令的模拟量输出变为开关量占空比输出，使用 GPWM 指令即可，PulT=5000 表示脉冲周期 5 秒，注意 PID 指令 MV 输出的量程要设为 0~1000。

HAL、D.HAL（上限报警）

指令格式及参数说明

语言	LD	FBD	IL
16、32 位 指令格式	HAL EnInUpSpanOut	D.HAL EnInUpSpanOut	HAL En, In, Up, Span, Out D.HAL En, In, Up, Span, Out

参数	参数定义	输入	输出	说明
En	使能	√		
In	输入	√		
Up	上限报警值	√		
Span	死区	√		
Out	状态输出		√	

【指令功能及作用说明】

HAL 指令为上限报警指令。若 $In > (Up + Span)$ ，则 $Out = ON$ ；若 $In < (Up - Span)$ ，则 $Out = OFF$ ，当 In 与 Up 的偏差值小于等于死区值 $Span$ 时， Out 保持不变。

【指令示例】



【程序说明】

当 $M0 = ON$ 启动上限报警功能。假如 $AI0$ 为测量温度值， $V1000 = 3000$ (300°)， $V1001 = 20$ (2°)，当 $AI0 > 3020$ (302°) 时 $Y3 = ON$ ；
当 $AI0 < 2980$ (298°) 时 $Y3 = OFF$ 。

LAL、D.LAL（下限报警）

指令格式及参数说明

语言	LD	FBD	IL
16、32 位 指令格式			LAL En, In, Down, Span, Out D.LAL En, In, Down, Span, Out

参数	参数定义	输入	输出	说明
En	使能	√		
In	输入	√		
Down	下限报警值	√		
Span	死区	√		
Out	状态输出		√	

【指令功能及作用说明】

LAL 指令为下限报警指令。若 $In < (Down - Span)$ ，则 $Out = ON$ ；若 $In > (Down + Span)$ ，则 $Out = OFF$ ，当 In 与 $Down$ 的偏差值小于等于死区值 $Span$ 时， Out 保持不变。

【指令示例】

//Network 1 下限报警



【程序说明】

当 M0=ON 启动下限报警功能。假如 AI0 为测量温度值，V1000=300（30°），V1001=20（2°），当 AI0<280（28°）时 Y3=ON；当 AI0>320（32°）时 Y3=OFF。

LIM、D.LIM（范围限制）

指令格式及参数说明

语言	LD	FBD	IL
16、32 位 指令格式			LIM En, In, Up, Down, Out D.LIM En, In, Up, Down, Out

参数	参数定义	输入	输出	说明
En	使能	√		
In	输入	√		
Up	上限值	√		
Down	下限值	√		
Eno	使能输出		√	
Out	输出		√	

【指令功能及作用说明】

- 1、LIM 为范围限制指令，如果 In>Up 则 Out=Up，如果 Down≤In≤Up 则 Out=In，如果 In<Down 则 Out=Down。
- 2、如果 Up≤Down，则本指令不执行。

【指令示例】

//Network 1 范围限制



【程序说明】

当 M0=ON 启动范围限制。假如 V1000=800,V1001=20,当 V0<20 时 AQ0=20 ;当 V0>800 时 AQ0=800,当 20≤V0≤800 时 AQ0=V0。

SC、D.SC（线性变换）

指令格式及参数说明

语言	LD	FBD	IL
16、32 位 指令格式	SC EnEno InOut InUp InDown OutUp OutDown	D.SC EnEno InOut InUp InDown OutUp OutDown	SC En, In, InUp, InDown, OutUp, OutDown, Out D.SC En, In, InUp, InDown, OutUp, OutDown, Out

参数	参数定义	输入	输出	说明
En	使能	√		
In	输入	√		
InUp	输入上限	√		
InDown	输入下限	√		
OutUp	输出上限	√		
OutDown	输出下限	√		
Eno	使能输出		√	
Out	变送输出		√	

【指令功能及作用说明】

- 1、SC 指令依照输入、输出的上下限把 In 输入进行线性变送后输出到 Out。
- 2、线性变送公式： $Out = (In - InDown) * (OutUp - OutDown) / (InUp - InDown) + OutDown$ 。
- 3、如果 InUp=InDown，则本指令不执行。

注：E/S 系列主机的 SC 指令参见“SC”。

【指令示例】

//Network 1 量程变送，从0~32000变送为1000~5000输出



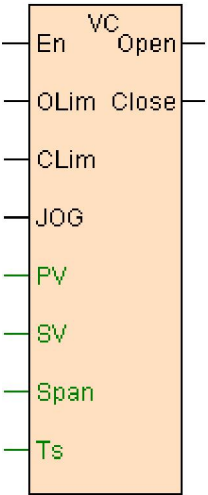
【程序说明】

当 M0=ON，SC 把 V0 变送输出到 V10，如下表。

元件	初始值	变送结果
V1000	32000	当 V0=0，则 V10 =1000 当 V0=100，则 V10 =1012 当 V0=12000，则 V10 =2500 当 V0=28000，则 V10 =4500
V1001	0	
V1002	5000	
V1003	0	

VC（阀门控制）

指令格式及参数说明

语言	LD	FBD	IL
指令格式			VC En, OLim, CLim, JOG, PV, SV, Span, Ts, Open, Close

参数	参数定义	输入	输出	说明
En	使能	√		
OLim	阀门开限位	√		
CLim	阀门关限位	√		
JOG	阀门点动	√		
PV	测量值	√		
SV	设定值	√		
Span	死区	√		
Ts	马达运行时间	√		单位秒
Open	阀门开输出		√	
Close	阀门关输出		√	

【指令功能及作用说明】

- 1、VC 为针对阀门进行控制的专用指令，相当于一台阀门控制器。
- 2、En 为指令的使能端，当 En=ON 时指令执行，当 En=OFF 指令停止工作，Open=OFF，Close=OFF。
- 3、VC 指令根据阀门的当前开度 PV 与阀门设定开度 SV 进行比较，若两者的差值大于死区 Span，则对阀门控制输出（开或关），每次控制电机运行的时间最长为 Ts（秒）。
- 4、Open 阀门开控制：当(SV - PV)> Span 时，Open=ON 阀门开输出；当(SV - PV)≤ Span 时，Open=OFF 阀门开停止。在控制阀门开的过程中，若阀门开限位 Olim=ON 时，Open=OFF。
- 5、Close 阀门关控制：当(PV - SV)> Span 时，Close=ON 阀门关输出；当(PV - SV)≤ Span 时，Close=OFF 阀门关停止。在控制阀门关的过程中，若阀门关限位 Clim=ON 时，Close=OFF。

- 6、阀门点动控制：当 JOG 从 OFF 变为 ON 时，将根据 SV、PV、Span 三者的当前值作出对阀门的开或关控制（控制依据与开关控制同），每次点动控制电机运行的时间最长为 Ts（秒）。
- 7、若 Ts>0 时，阀门保护时间有效，在 Ts 时间内阀门仍未到达预定位置（设定值），开关动作将停止，以防阀门出现机械性故障而使电机运转过长时间被损坏。
- 8、若 Ts=0 时，表示不计时间输出，注意这将失去对阀门的保护功能。
- 9、若 Span<0 或者 Ts<0 时，则指令不执行。
- 10、如果阀门本身没有限位保护输出，Olim、Clim 可以使用 SM1 输入，表示不用限位保护功能。
- 11、VC 指令可以配合 PID 等指令实现更复杂的控制功能。

【指令示例】

//Network 1 阀门控制



TTC（温度曲线控制）

指令格式及参数说明

语言	LD	FBD	IL
指令格式			TTC En, Begin, End, Ts, Act, Out, Val, Ct

参数	参数定义	输入	输出	说明
En	使能	√		
Begin	起点值	√		
End	终点值	√		
Ts	控制时间	√		Ts>0 时单位为秒，Ts<0 时单位为分钟
Act	控制方式	√		
Eno	使能输出		√	
Out	状态输出		√	
Val	输出值		√	

参数	参数定义	输入	输出	说明
Ct	当前时间		√	单位为秒，占用 2 个连续元件

【指令功能及作用说明】

- 1、TTC 指令用于控制一个数值在给定的时间内从起点值（Begin）到终点值（End）的变化过程，即斜率控制。
- 2、Ts 端子为控制时间，Ts>0 时单位为秒，Ts<0 时单位为分钟，Ts=0 则无效指令不执行，无论单位是秒还是分指令都是每秒运算输出一次。
- 3、当 Act=0（重后模式），在 TTC 指令启动时，复位 Out=OFF，然后从 Begin 开始计时控制输出 Val。
- 4、当 Act=1（记忆模式），在 TTC 指令启动时，复位 Out=OFF 并且读取当前输出值 Val，根据 Val 在 Begin~End 区间的如下情况执行。

1)、Begin<End（上升阶段），如果 Val≤Begin 则从 Begin 开始计时控制输出 Val；如果 Val≥End 则计时时间到，Val 不变且 Out=ON；如果 Val 在 Begin~End 区间内则从 Val 开始计时控制输出 Val。

2)、Begin>End（下降阶段），如果 Val≥Begin 则从 Begin 开始计时控制输出 Val；如果 Val≤End 则计时时间到，Val 不变且 Out=ON；如果 Val 在 Begin~End 区间内则从 Val 开始计时控制输出 Val。

3)、Begin=End（保持阶段），如果 Val=Begin 则从起点开始计时控制输出 Val；如果 Val≠Begin 则计时时间到，Val 不变且 Out=ON。
- 5、指令在执行过程中，参数 Begin、End、Ts 不能被修改（修改不会实时生效，需要重新执行指令）。
- 6、使用 TTC 指令可以生成多段曲线，可以配合 PID 等其他指令实现更复杂的控制功能。
- 7、在 Act=1（记忆模式）时，Val 输出端子应该使用停电保持型的元件。

注：E/S 系列主机的 TTC 指令参见“TTC”。

【指令示例】

//Network 1 第1段为300~1200，时间2分钟，上升



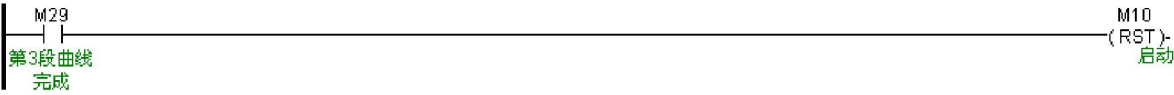
//Network 2 第2段为1200~1200，时间30秒，保持



//Network 3 第3段为1200~300时间60秒，下降

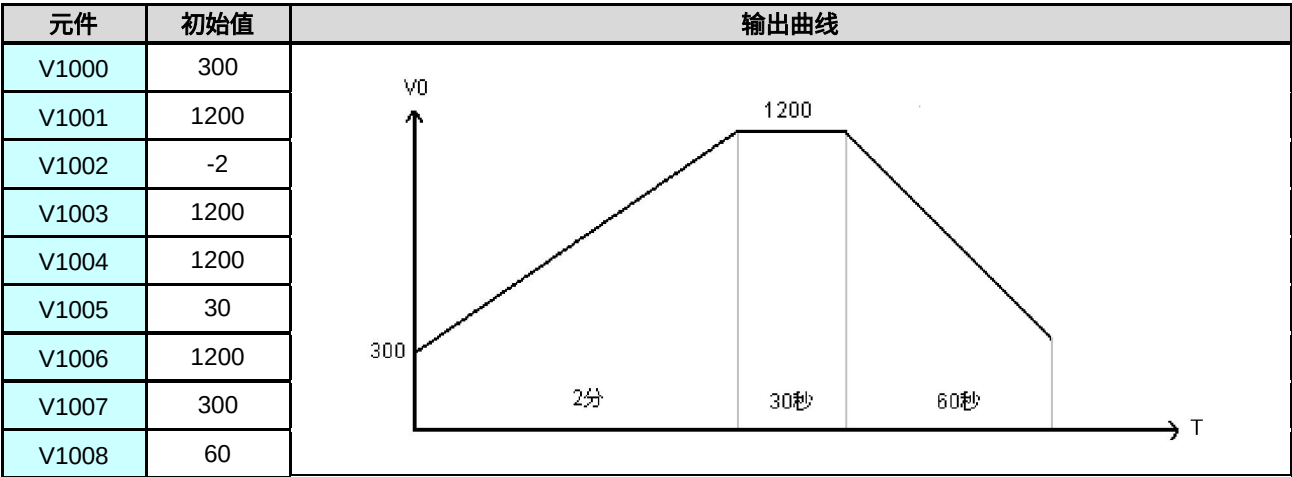


//Network 4



【程序说明】

假如“3 段曲线初始数据”如下表。



- 1、当 M10=ON，启动第 1 段曲线，从 300 到 1200 历时 2 分钟，完成后 M27=ON。
- 2、当 M27=ON，启动第 2 段曲线，从 1200 到 1200 历时 30 秒，完成后 M28=ON。
- 3、当 M28=ON，启动第 3 段曲线，从 1200 到 300 历时 60 秒，完成后 M29=ON。
- 4、当 M29=ON，复位 M10=OFF。

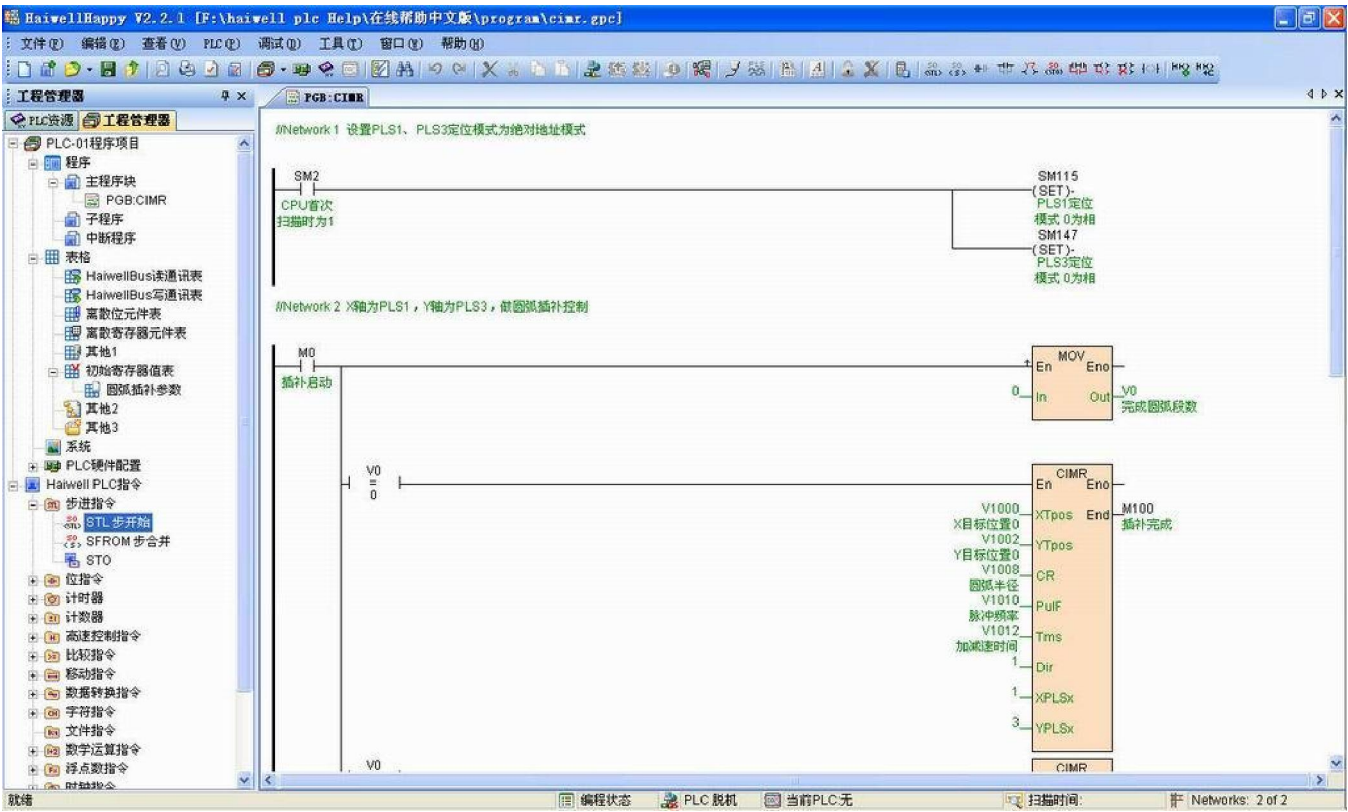
第五章、编程操作手册

本章详细介绍 HaiwellHappy 编程软件的操作 。

5.1 编程环境

概述

编程软件的主界面包括：菜单、工具栏、工程管理器、工作区和状态栏等内容。



菜单

菜单包含了编程软件的所有功能，由主菜单和多级子菜单组成。

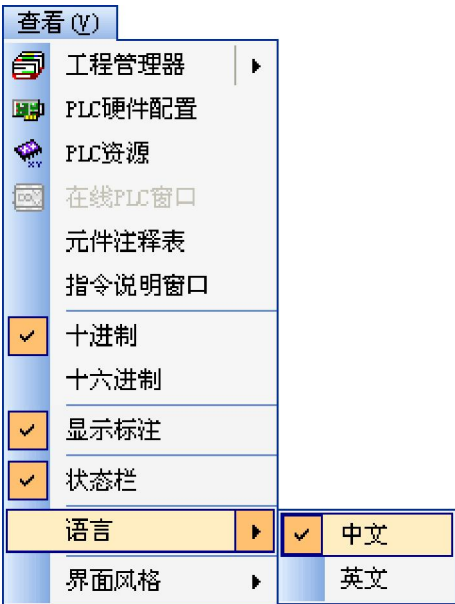
1、“文件”子菜单：用于对程序项目文件进行管理的相关操作。



2、“编辑”子菜单：用于实现程序编辑的相关操作。



3、“查看”子菜单：用于打开各种资源窗口，改变语言、显示方式等操作。



4、“PLC”子菜单：用于对 PLC 进行各种控制。



5、“调试”子菜单：提供一组方便调试程序的相关工具。



6、“工具”子菜单：提供一组便利工具，如：管理远程模块。



7、“帮助”子菜单：提供在线帮助功能。



工具栏

工具栏包含了 PLC 编程软件的常用功能，以便于用户快速编程。当鼠标移到工具栏按钮上面时会提示该按钮的名称。

1、标准工具栏：



2、LD 语言工具栏：



3、FBD 语言工具栏：



4、仿真器工具栏：



5、“插补仿真器”窗口工具栏：



6、“远程模块”窗口工具栏：



7、“批量元件注释”窗口工具栏：



8、打印预览工具栏：



右键菜单

为提高编程效率，编程软件使用了大量的右键菜单。通过鼠标右键 点击即可弹出的右键菜单。

1、在 LD 语言的编程环境下，选中开关时的右键菜单：



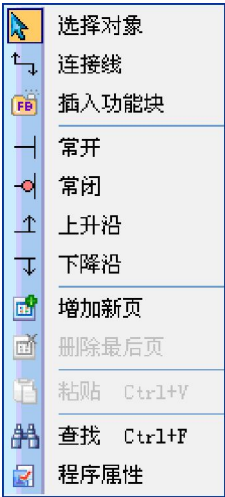
2、 在 LD 语言的编程环境下，选中指令或指令端子时的右键菜单：



3、 在 FBD 语言的编程环境下，选中指令时的右键菜单：



4、 在 FBD 语言的编程环境下，选中指令端子时的右键菜单：



5、 在 IL 语言的编程环境下的右键菜单：



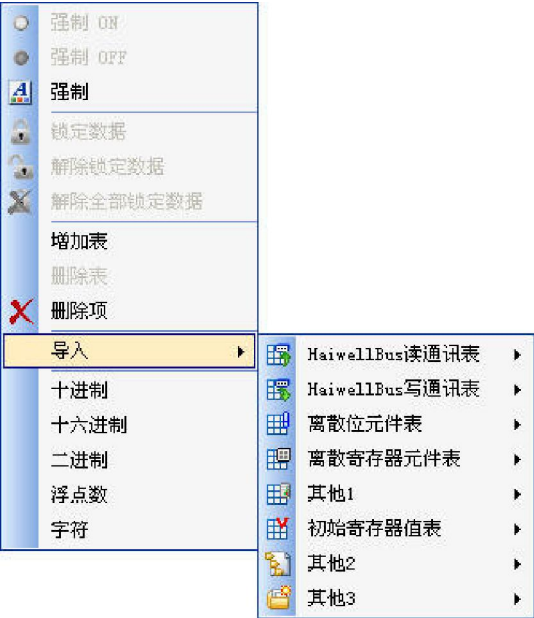
6、 在仿真环境下的右键菜单：



7、 在在线监控环境下的右键菜单：



8、 在“元件状态表”窗口的右键菜单：



9、“PLC 硬件配置”窗口、“工程管理器”窗口、“在线 PLC”窗口等也提供右键菜单。

快捷键

HaiwellHappy 编程软件提供了丰富的快捷键，用户在编程操作的过程中， 可以使用快捷键来提高编程速度和效率，使得编程操作简单而高效，快捷键列表 如下：

类别	快捷功能键	功能及用途
菜单操作	Alt + F	打开“文件”菜单
	Alt + E	打开“编辑”菜单
	Alt + V	打开“查看”菜单
	Alt + P	打开“PLC” 菜单
	Alt + D	打开“调试”菜单
	Alt + T	打开“工具”菜单
	Alt + W	打开“窗口”菜单
	Alt + H	打开“帮助”菜单
程序项目操作	Ctrl + N	新建程序项目
	Ctrl + O	打开程序项目
	Ctrl + S	存储程序项目
	Ctrl + P	打印
	Alt + F4	退出编程软件
编辑	Ctrl + Z	恢复

	Ctrl + R	重做
	Ctrl + X	剪切
	Ctrl + C	复制
	Ctrl + V	粘贴
	Ctrl + G	Go to ...
	Ctrl + F	查找
	F3	查找下一个
	Ctrl + A	全选
调试	F5	启动“在线监控”
	F6	停止“在线监控”
程序编辑	Ctrl + 1	LD 程序：常开开关
	Ctrl + 2	LD 程序：常闭开关
	Ctrl + 3	LD 程序：上升沿开关
	Ctrl + 4	LD 程序：下降沿开关
	Ctrl + 5	LD 程序：等于 开关(16 位比较)
	Ctrl + 6	LD 程序：不等于 开关(16 位比较)
	Ctrl + 7	LD 程序：大于 开关(16 位比较)
	Ctrl + 8	LD 程序：大于等于 开关(16 位比较)
	Ctrl + 9	LD 程序：小于 开关(16 位比较)
	Ctrl + 0	LD 程序：小于等于 开关(16 位比较)
	Ctrl + Alt + 5	LD 程序：等于 开关(32 位比较)
	Ctrl + Alt + 6	LD 程序：不等于 开关(32 位比较)
	Ctrl + Alt + 7	LD 程序：大于 开关(32 位比较)
	Ctrl + Alt + 8	LD 程序：大于等于 开关(32 位比较)
	Ctrl + Alt + 9	LD 程序：小于 开关(32 位比较)
	Ctrl + Alt + 0	LD 程序：小于等于 开关(32 位比较)
	Ctrl + Shift + 5	LD 程序：等于 开关(浮点数比较)
	Ctrl + Shift + 6	LD 程序：不等于 开关(浮点数比较)
	Ctrl + Shift + 7	LD 程序：大于 开关(浮点数比较)
	Ctrl + Shift + 8	LD 程序：大于等于 开关(浮点数比较)
	Ctrl + Shift + 9	LD 程序：小于 开关(浮点数比较)
	Ctrl + Shift + 0	LD 程序：小于等于 开关(浮点数比较)
	Alt + 5	LD 程序：等于 开关(低字节比较)
	Alt + 6	LD 程序：不等于 开关(低字节比较)
	Alt + 7	LD 程序：大于 开关(低字节比较)
	Alt + 8	LD 程序：大于等于 开关(低字节比较)
	Alt + 9	LD 程序：小于 开关(低字节比较)
	Alt + 0	LD 程序：小于等于 开关(低字节比较)
	Alt + Shift + 5	LD 程序：等于 开关(高字节比较)
	Alt + Shift + 6	LD 程序：不等于 开关(高字节比较)
	Alt + Shift + 7	LD 程序：大于 开关(高字节比较)
	Alt + Shift + 8	LD 程序：大于等于 开关(高字节比较)
	Alt + Shift + 9	LD 程序：小于 开关(高字节比较)
	Alt + Shift + 0	LD 程序：小于等于 开关(高字节比较)

	Ctrl + U	LD、IL 程序：应用指令上移
	Ctrl + D	LD、IL 程序：应用指令下移
	Ctrl + Up	LD 程序：网络上移
	Ctrl + Down	LD 程序：网络下移
	F7	LD 程序：STL 步开始
	F8	LD 程序：SFROM 步合并
	F9	LD 程序：串联开关
	F10	LD 程序：并联开关
	F11	LD 程序：输出线圈
	F12	LD 程序：STO 步转移指令
	Ctrl+B	LD 程序：输出分支
	Ctrl+H	LD 程序：删除输出分支
	Ctrl+L	LD 程序：添加网络
	Ctrl+I	LD 程序：插入网络
在线帮助	F1	打开在线帮助
通用	ESC	取消操作
	DEL	删除选中的对象

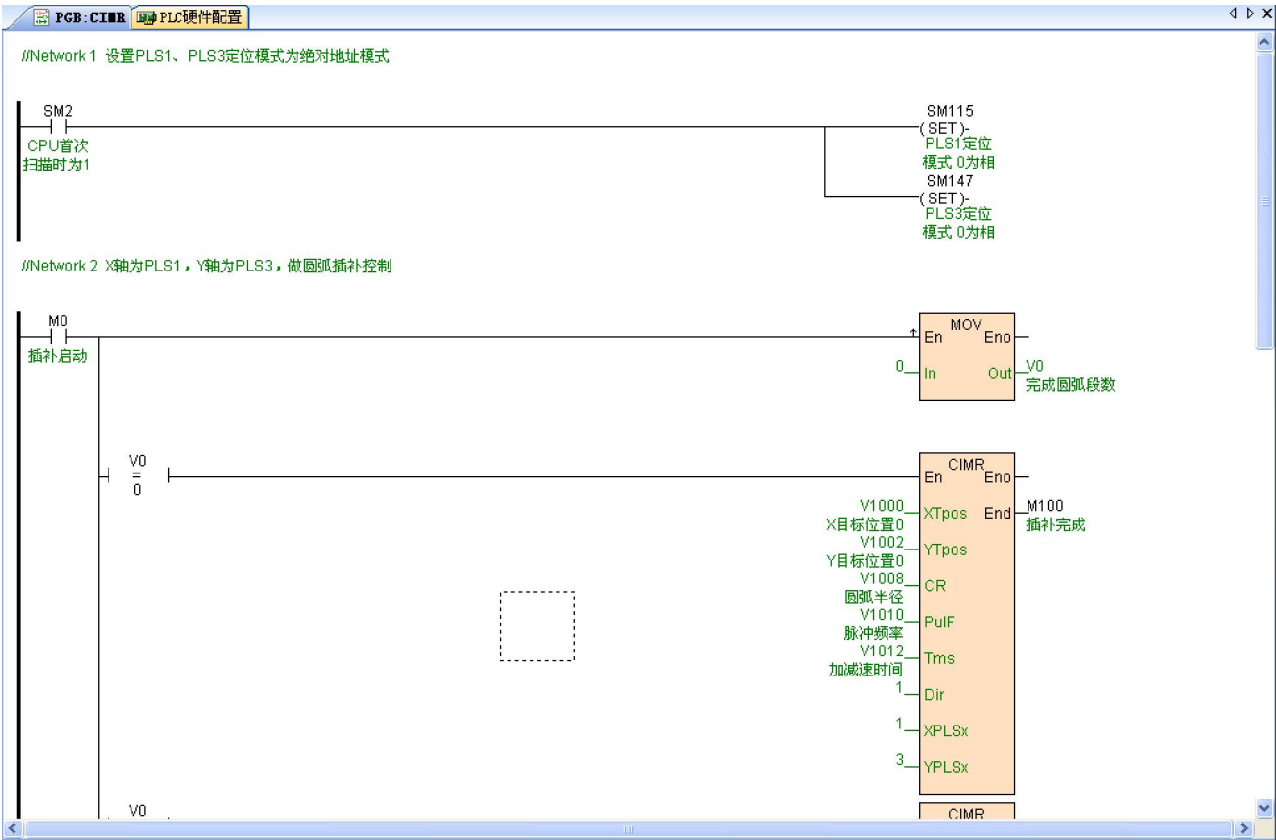
状态栏

状态栏用于指示编程软件当前的状态及显示有关操作的提示信息，可通过菜单【查看/状态栏】打开或关闭它。



工作区

用户工作区是用于编写控制程序、配置 PLC 硬件 信息的区域。在此区域内，用户可对当前的程序或 PLC 硬件配置信息进行编辑。



工程管理器

可通过菜单【查看/工程管理器】打开“工程管理器”页面，它以树形结构显示整个工程项目的全部内容：项目名称、主程序块、子程序、中断程序、表格、PLC 硬件配置、各种指令等。工程管理器支持右键菜单，方便用户可对工程项目的管理操作。



PLC 资源

可通过菜单【查看/PLC 资源】打开“PLC 资源”页面，“PLC 资源”页面共有 5 个子页面，列出了当前程序项目使用的 CPU 主机的所有资源，以及系统状态位、系统寄存器、中断和错误代码。

PLC资源	
工程管理器	PLC资源
PLC资源	系统状态位
类型	PLC资源
CPU 类型	N16S2T N系列
程序空间	48000
扩展模块数	7
高速计数器(HSC)	8点 HSC0 - HSC7
脉冲输出(PLS)	8点 PLS0 - PLS7
外部开关输入(X)	1024点 X0 - X1023
外部开关输出(Y)	1024点 Y0 - Y1023
计时器(T)	1024点 T0 - T1023
计数器(C)	256点 C0 - C255
32位计数器(C)	32点 C48 - C79
内部状态位(M)	12288点 M0 - M12287
步进状态位(S)	2048点 S0 - S2047
系统状态位(SM)	216点 SM0 - SM215
外部模拟输入(AI)	256点 AI0 - AI255
外部模拟输出(AQ)	256点 AQ0 - AQ255
系统寄存器(SV)	701点 SV0 - SV700
内部寄存器(V)	14848点 V0 - V14847
局部状态位(LM)	32点 LM0 - LM31
局部寄存器(LV)	32点 LV0 - LV31
变址指针(P)	10点 P0 - P9
中断(I)	52点 I1 - I52
停电保持(T)	32点 T96 - T127
停电保持(C)	64点 C64 - C127
停电保持(M)	512点 M1536 - M2047
停电保持(S)	100点 S156 - S255
停电保持(V)	1048点 V1000 - V2047

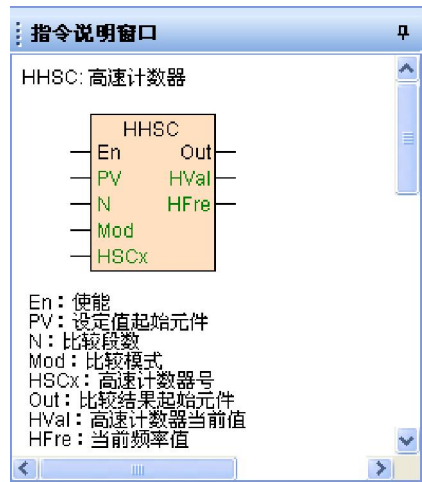
元件注释表

可通过菜单【查看/元件注释表】打开“元件注释表”页面，它列出了当前程序项目中使用的所有元件的注释。



指令说明窗口

可通过菜单【查看/指令说明窗口】打开“指令说明窗口”页面，它列出了当前选中的指令的简要说明，方便用户对指令的了解。



指令属性窗口

双击程序中的应用指令，即可打开该指令的属性窗口，不同指令显示内容会有所不同。



- 1、当选择“指令禁用”时，该指令不会被执行且变为灰色显示。
- 2、当选择“低字节模式”时，该指令为 8 位模式；当选择“32Bits”时，该指令为 32 位模式。
- 3、在本窗口还可以修改计时器的时基、计数器号、指令的输入参数和输出参数等。

端子定义窗口

在“指令属性窗口”双击输入或输出端子可以打开“端子”窗口，不同指令的输入或输出端子显示的内容会有所不同，在“端子”窗口可以通过配置方式定义参数，无需查阅资料即可方便输入复杂的组合参数，如：通讯指令 COMM 的 Protocol 端子。



5.2 程序结构

用户程序由主程序、子程序、中断程序三类程序块构成。

- 1、主程序块、子程序、中断程序三者的总数不能超过 31 个。
- 2、所有程序块都可以单独设置口令，实现局部加密功能。

主程序

主程序是用户程序的主体和框架，Haiwell（海为）PLC 允许有多个主程序块，多个主程序块之间属同级关系，系统处于运行状态时，所有主程序块都会被循环执行，执行顺序为在“工程管理器”目录树上的顺序，从上到下执行。

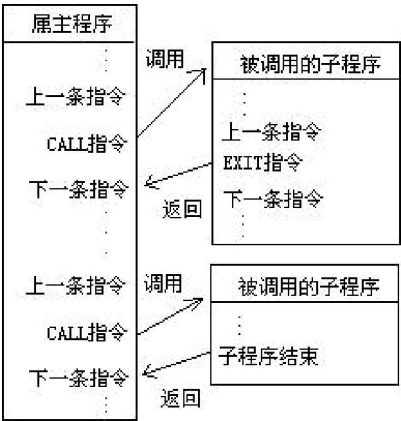
- 1、用户程序至少必须包含一个主程序块。
- 2、可以通过菜单【调试/程序块执行顺序】调整它们的执行顺序。

子程序

子程序是功能独立的、可以被其它程序块调用的程序块，子程序可带参数或不带参数，一个子程序最多可定义 8 个输入（IN）或输入输出（IN_OUT）参数及 3 个输出（OUT）参数。

- 1、用户程序可以没有子程序，也可包含一个或多个子程序。
- 2、子程序只有在被 CALL 指令调用时才执行。
- 3、子程序可调用别的子程序，但不能调用自己，也不能循环调用。
- 4、子程序允许嵌套调用，包括程序块调用子程序在内，其嵌套调用的深度不能超过 8 级。

子程序 调用示意图如下：



中断程序

中断程序是处理系统中中断事件的程序。系统为每个中断事件分配一个中断号，参考“[系统中断表](#)”。

- 1、中断程序的执行必须满足 2 个条件：一是使用 ATCH 指令将中断程序与中断事件（中断号）绑定，二是系统发生该中断事件。其中，对 X0~X7 上升沿或下降沿的捕捉中断必须在“[PLC 硬件配置](#)”窗口的“[X 开关量输入参数](#)”中开启。
 - 2、当系统中断事件发生时，程序执行被打断，程序自动跳转到与该中断事件绑定的中断程序执行，直至中断程序执行完毕，系统才会返回到正常的程序执行。
 - 3、用户程序可以没有中断程序，也可包含一个或多个中断程序。
 - 4、系统中断受 ENI（中断允许）和 DISI（中断禁止）指令的控制，在中断禁止的情况下，系统不会发生中断事件。默认为允许中断。
- 注意：**中断程序只在相应中断事件发生时执行一次。

5.3 程序项目建立

新建程序项目

通过点击菜单【文件/新建程序项目】、点击工具栏的按钮或用快捷键 Ctrl+N，打开“新建程序项目”窗口。

新建程序项目

PLC 系列: S系列CPU 类型: S16S2T

自动保存时间间隔: 0 分

S16S2T (V0-V2047 M0-M2047 T0-T127 C0-C127 S0-S255)
CPU模块 8*DI 8*DO晶体管 AC220V供电 1路20KHz脉冲输入 1路10KHz脉冲输出 2个通讯端口 可扩展7个模块

停电保持 (V1000-V2047 M1536-M2047 T96-T127 C64-C127 S156-S255)

起始元件	长度	起始元件	长度
V 1000	1048	T 96	32
M 1536	512	C 64	64
S 156	100		

项目名: Haiwell PLC程序项目

用户名:

设计: 版本:

公司:

口令: 确认口令:

创建日期: 2013-4-15 15:34:46 修改日期:

注释:

确定 取消

- 1、在“PLC 系列”下拉列表中选择“PLC 系列”。
- 2、在“CPU 类型”下拉列表中选择“CPU 即主机的型号”。
- 3、停电保持区可以由用户任意定义，可设置 V、M、S、T、C 五种元件类型的停电保持区的起始元件及长度。系统默认的停电保持区域列表：

元件类型	停电保持区	元件个数
V	V1000~V2047	1048
M	M1536~M2047	512
S	S156~S255	100
T	T96~T127	32
C	C64~C127	64

- 4、在“项目名”中输入新建项目的名称，此项目名称将显示于主画面的工程管理器中。
- 5、在“用户名”、“设计”、“版本”、“公司”等栏目中输入该项目的有关信息。
- 6、如需对新建的程序项目做口令保护，则在“口令”及“确认口令”中输入保护口令。
- 7、在“注释”中可以输入该项目的相关备注信息。
- 8、按“确定”按钮建立了一个空白的程序项目。

新建程序块

用户程序由主程序、子程序、中断程序三类程序块构成。用户根据控制要求建立各种程序块。

- 1、点击菜单 【文件/新建...../主程序块】 或点击工具栏的  按钮下的“主程序块”，打开“新建程序块”窗口。



- 2、在“块名称”中输入该新建程序块的名称（此名称将显示于工程管理器中），选择该程序块的类型及编程语言，可在注释文本框中写入一些注释说明。
- 3、口令”及“确认口令”两个文本框是用于设置该程序块（程序块、子程序、中断程序）的保护密码。若对程序块设置了密码，则要求用户输入正确的密码后方可对其进行查看或编辑，对每个程序块都可以设置独立的密码，实现局部加密功能。
- 4、用户可以选择自己喜欢的语言编写程序，按“确定”按钮即可建立一个空白的程序块。

新建子程序

当新建一个子程序或打开子程序的属性窗口时，窗口的下部将列出该子程序参数的表格，如下图所示。

子程序 程序项目属性

块名称: 循环时间累计

口令:

确认口令:

类型

☐ 主程序块

☒ 子程序

☐ 中断程序

编程语言

☒ LD

☐ FBD

☐ IL

☐ SFC

☐ ST

注释

参数 (Input Max.=8 Output Max.=3)

地址	参数名称	参数类型	数据类型	注释
LM0		IN	BOOL	
LM1		IN_OUT	BOOL	
LM2		OUT	BOOL	
LV0		OUT	WORD	
LV1		OUT	INT	
LV2		OUT	DWORD	
LV4		OUT	REAL	

添加

插入

删除

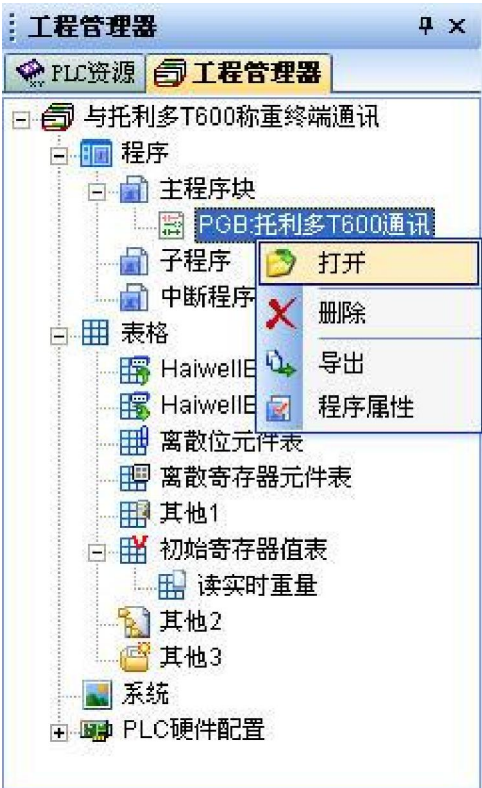
确定

取消

- 1、一个子程序最多可定义 8 个输入（IN）或输入输出（IN_OUT）参数及 3 个输出（OUT）参数。每个参数都有其参数类型（IN、OUT、IN_OUT）与数据类型（BOOL、WORD、DWORD、INT、DINT、REAL）。
- 2、用 CALL 指令调用子程序时，CALL 指令输入输出参数的数据类型要与子程序定义的参数数据类型一致。
- 3、参数的地址是系统自动分配的，当数据选择 BOOL 型时，地址从 LM0 开始分配，其它参数类型的地址从 LV0 开始分配。
- 4、按“添加”按钮添加一个新参数行，新参数类型默认为 IN，数据类型默认为 BOOL。
- 5、点击该参数的参数名称栏，在文本框中输入该参数的名称，参数名称最多 5 个字符，对每个参数必须命名不同的名称，该名称将在 CALL 指令上显示。
- 6、点击该参数的参数类型栏，在下拉列表中选择该参数的参数类型（IN、IN_OUT、OUT），系统会自动根据选择的参数类型对参数排序。
- 7、点击该参数的数据类型栏，在下拉列表中选择该参数的数据类型（BOOL、WORD、DWORD、INT、DINT、REAL）。
- 8、按“插入”按钮，在当前行的前面将插入一个新参数。
- 9、点击要删除的参数行，按“删除”按钮，该参数被删除。
- 10、可在注释栏输入该参数的注释。

打开程序块

双击“工程管理器”目录树上的 程序块名称或者通过右键菜单“打开”可以打开该程序块。



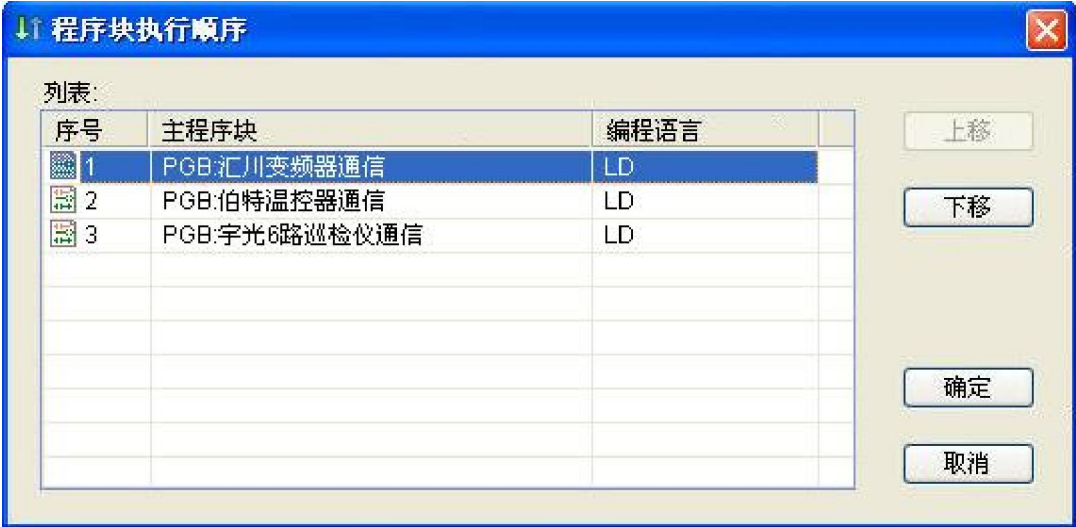
删除程序块

右键点击“工程管理器”窗口中要删除的程序块的名称，在弹出的右键菜单中点击“删除”即可。



程序块执行顺序调整

- 1、点击菜单【调试/程序块执行顺序】，打开”程序块执行顺序“窗口。



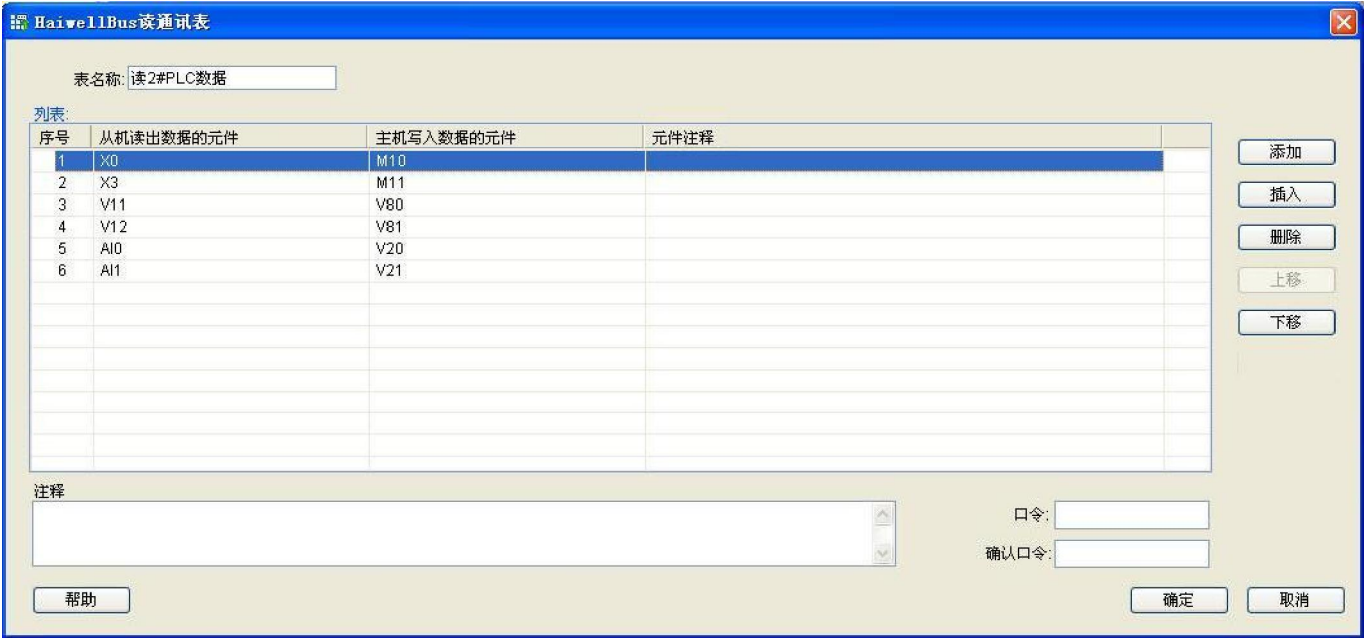
- 2、列表框列出了当前程序项目中的所有主程序块，序号为当前的执行顺序，选中要调整执行顺序的程序块，点击“上移”或“下移”按钮，即可调整程序块的执行顺序。
- 3、子程序由 CALL 指令调用，中断程序由系统中断触发执行，无法调整执行顺序。

5.4 指令使用表格

HaiwellHappy 编程软件提供多种指令使用表格，使用这些表格可减少许多程序量，节约程序空间，实现如初始化数据等功能。每个表格可单独设置保护口令，亦可导入或导出 。

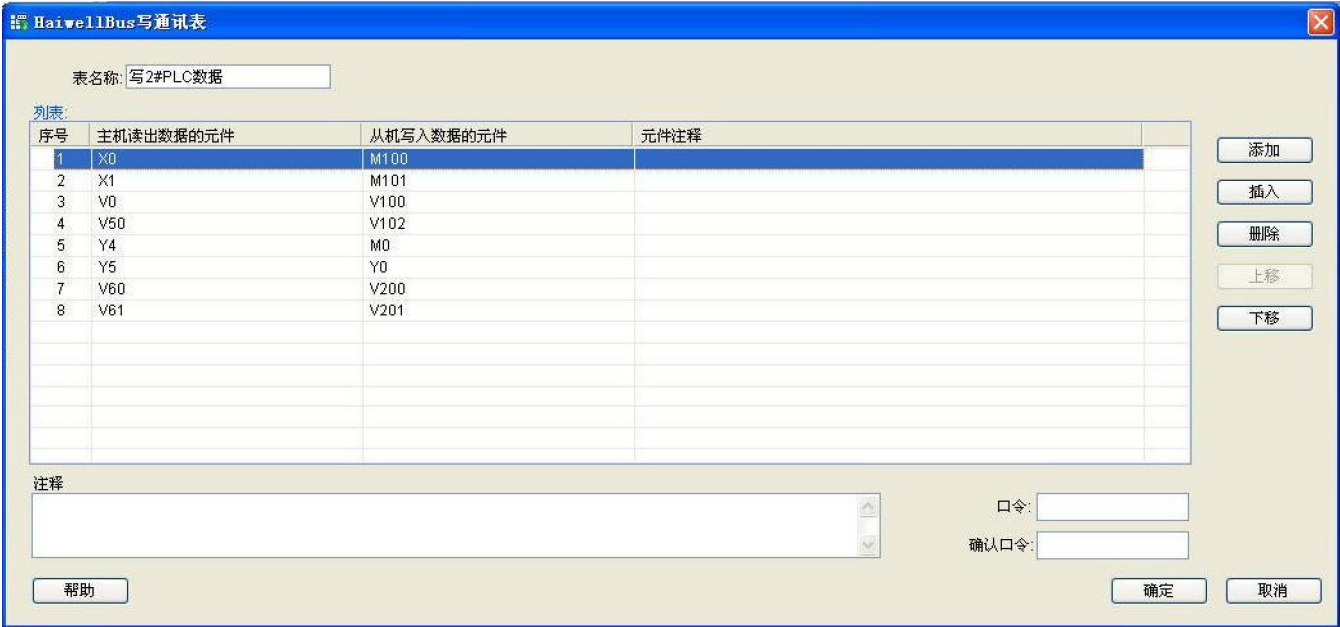
Haiwellbus 读通讯表

“Haiwellbus 读通讯表”作为 **HWRD** 指令的 Table 端参数 ，当 PLC 与 PLC 之间使用 Haiwellbus 协议通讯时，HWRD 指令根据指定的“Haiwellbus 读通讯表”自动读取从 PLC 的数据再写到主 PLC。如下图：



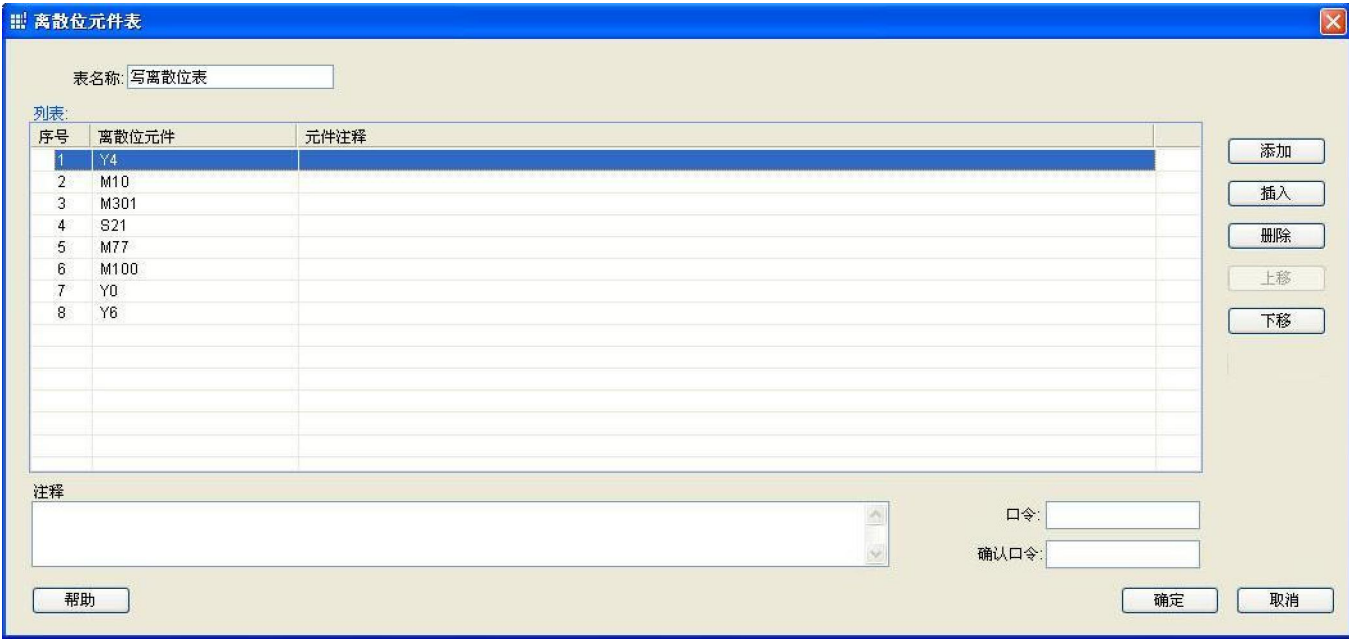
Haiwellbus 写通讯表

“Haiwellbus 写通讯表”作为 **HWWR** 指令的 Table 端参数 ，当 PLC 与 PLC 之间使用 Haiwellbus 协议通讯时，HWWR 指令根据指定的“Haiwellbus 写通讯表”自动读取主 PLC 的数据再写到从 PLC。如下图：



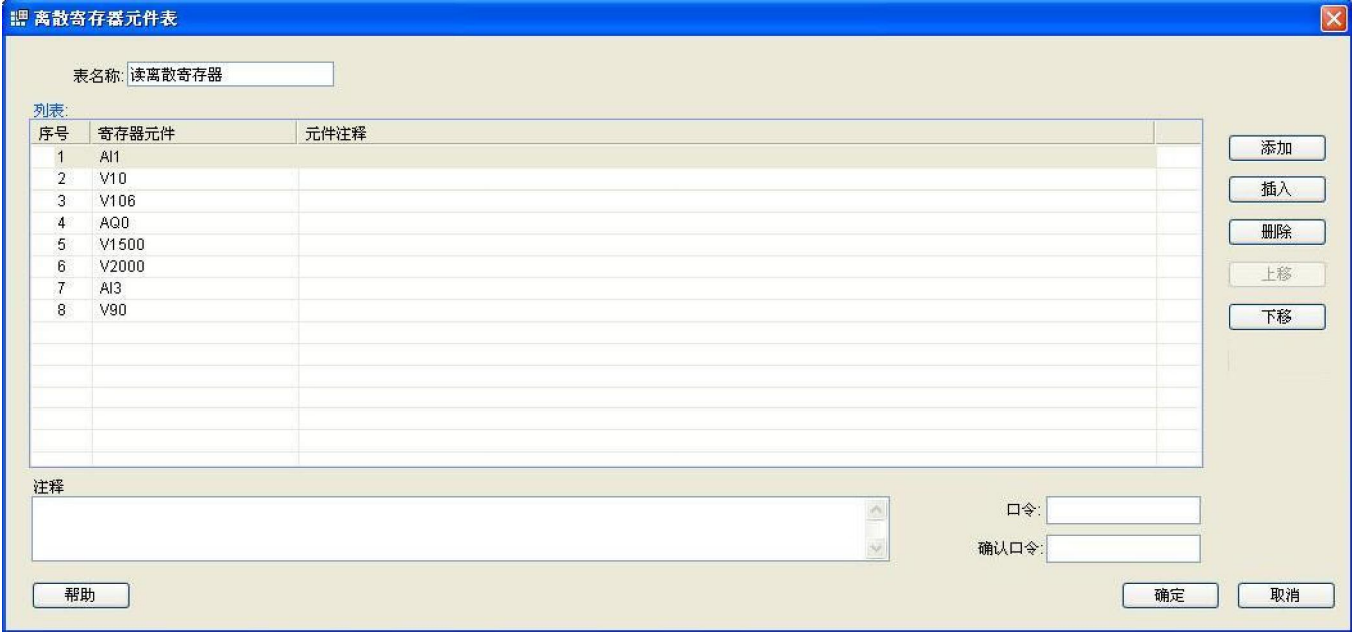
离散位元件表

“离散位元件表”定义一组离散的位元件，作为 [BUNB](#)、[BUNW](#)、[BDIB](#)、[WDIB](#) 指令的 Table 端参数，快速实现位元件的组合或分散，方便上位机访问。如下图：



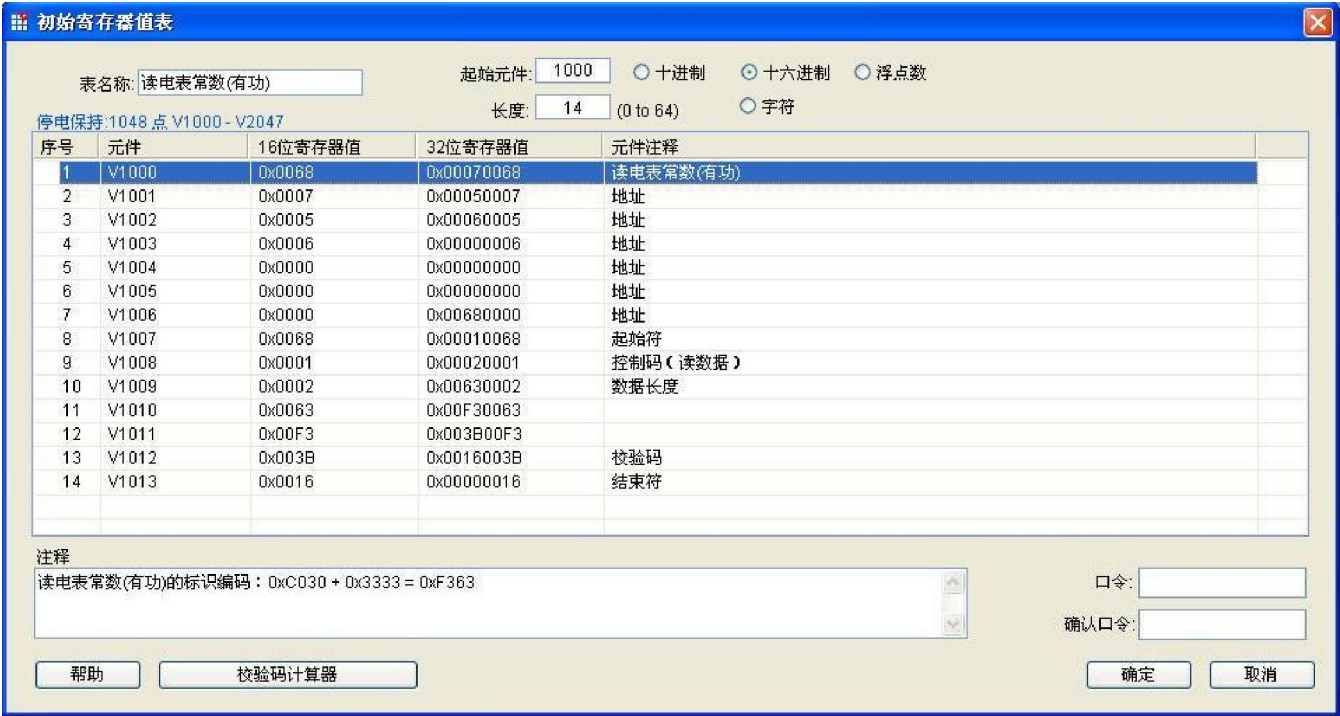
离散寄存器元件表

“离散寄存器元件表”定义一组离散的寄存器元件，作为 [WUNW](#)、[WDIW](#) 指令的 Table 端参数，快速实现寄存器元件的组合或分散，方便上位机访问。如下图：



初始寄存器值表

用于给寄存器元件定义初始值，当程序项目下载到 PLC 中时，表中所设定的 初始值会随程序一起下载，可定义整数、长整数、浮点数、字符格式数据。如下图：



- 1、“起始元件”指定了 V 元件的起点，“长度”指定了使用元件的个数。
- 2、“十进制”、“十六进制”、“浮点数”、“字符”选择数值的输入方式。
- 3、在“十进制”、“十六进制”模式下，在“16 位寄存器值”栏输入 16 位整数值，在“32 位寄存器值”栏输入 32 位整数值。注意：“32 位寄存器值”一栏显示的是相邻 2 个连续寄存器的值，如上图第一行 V1000 元件的“32 位寄存器值”是 V1000V1001 的数值。
- 4、在“浮点数”模式下直接输入浮点数值。注意：浮点数为 32 位单精度值，占用 2 个连续寄存器。
- 5、在“字符”模式下直接输入字符即可。

新建表格

- 1、右键点击“工程管理器”窗口中要新建的表格类型名称，在弹出的右键菜单中点击“新建”；点击菜单【文件/新建.....】或点击工具栏的

 按钮选择要新建的表格类型， 即可新建相应种类的表格。 不同种类表格界面和内容会有所不同。



- 2、在“表名称”中输入该新建表格的名称（此名称将显示于工程管理器中），可在注释文本框中写入一些注释说明。
- 3、口令及“确认口令”两个文本框是用于设置该表格的保护密码。若对表格设置了密码，则要求用户输入正确的密码后方可对其进行查看或编辑，对每个表格都可以设置独立的密码，实现局部加密功能。
- 4、对“初始寄存器值表”定义表格的“起始元件”和“长度”，然后输入元件的数值。对其他种类表格则通过“添加”、“插入”、“删除”、“上移”和“下移”按钮来编辑表格内容。
- 5、可以在“注释”中写入表格注释，在“元件注释”栏编辑元件的注释。
- 6、按“确定”完成表格建立。

打开表格

双击“工程管理器”目录树上的表格名称或者通过右键菜单“打开”可以打开该表格。



删除表格

右键点击“工程管理器”窗口中要删除的表格的名称，在弹出的右键菜单中点击“删除”即可。



5.5 PLC硬件配置

双击“工程管理器”目录树上的“PLC 硬件配置”节点或者菜单【查看/PLC 硬件配置】，打开“PLC 硬件配置”窗口。用户根据实际硬件型号添加扩展模块，可以添加、删除或者调整扩展模块的位置，可配置主机或扩展模块的参数（一般情况保持默认值即可）。



PLC 主机型号

PLC 主机型号在新建程序项目时就可以选择确定，如果需要修改主机型号则通过菜单【文件/程序项目属性】打开“程序项目属性”窗口，选择正确的“PLC 系列”和“CPU 类型”，按“确定”按钮即可。

程序项目属性

PLC 系列: N系列

CPU 类型: N16S2T

自动保存时间间隔: 6 分

N16S2T (V0-V14847 M0-M12287 T0-T1023 C0-C255 S0-S2047)
CPU模块 8*DI 8*DO 晶体管 AC220V供电 4路200KHz脉冲输入 4路200KHz脉冲输出 2个通讯端口 可扩展7个模块

停电保持 (V1000-V2047 M1536-M2047 T96-T127 C64-C127 S156-S255)

起始元件	长度	起始元件	长度
V 1000	1048	T 96	32
M 1536	512	C 64	64
S 156	100		

项目名称: PLC-01程序项目

用户名:

设计:

版本:

公司:

口令:

确认口令:

创建日期: 2012-9-26 13:16:12

修改日期: 2013-5-11 10:18:17

注释:

确定 取消

扩展模块编辑

- 1、展开工程管理器中的“Haiwell PLC 模块”目录树，找到要添加的模块型号，双击此模块或将此模块拖拉到 PLC 硬件配置表中，模块将被添加到 PLC 硬件配置表的当前位置。
- 2、调出右键菜单，可以添加、插入或者删除相应的扩展模块，也可以通过上移（Ctrl + U）或下移（Ctrl + D）调整扩展模块的位置。
- 3、点击要删除的模块，按“Del”键， 即可删除该模块。

PLC硬件配置

序号	模块型号	X元件	Y元件	AI元件	AQ元件	其他
0	N16S2T	X0 - X7	Y0 - Y7			COM
1	H40XDT	X8 - X27	Y8 - Y27			
2	H64XDT	X32 - X63	Y32 - Y63			
3	S04XA			AI0 - AI1	AQ0 - AQ1	

删除

上移 Ctrl+U

下移 Ctrl+D

添加

插入

开关量模块

模拟量模块

扩展接口模块

S01RS

H01RS

H02RS

S04XA

帮助 默认值

外部模拟输入

信号类型 使用工程量 下限值

外部 I/O 元件的分配

Haiwell（海为）PLC 的外部 I/O 元件（X、Y、AI、AQ）由系统自动分配，元件编号为十进制，当增加、删除或者调整模块位置时系统会自动重新分配。

注意：AI 和 AQ 元件的编号为连续编号，而 X 和 Y 元件的编号有可能不是连续的。因为 X 和 Y 元件以 8 个为一组，当模块的 I/O 点数非 8 的倍数时就会产生不连续的编号。

X 开关量输入参数

- 1、可以设置外部开关量输入的“滤波时间”，滤波时间越大那么响应越慢、滤波效果越好，滤波时间越小那么响应越快、滤波效果越差。
- 2、CPU 主机带的 X0~X7 开关量输入支持边沿捕捉功能，如果需要使用边沿捕捉，需勾选相应 X 通道的“上升沿捕捉”或“下降沿捕捉”选项。

外部开关输入

地址范围: X0 - X7

滤波时间: 6.4 ms

上升沿捕捉: ☐ ☒ ☐ ☐ ☐ ☐ ☐ ☐

下降沿捕捉: ☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐

Y 开关量输出参数

当配置 Y 通道为“停机输出”时，在 PLC 停机时该 Y 通道会保持 ON 输出。

外部开关输出

元件范围: Y0 - Y19

停机输出 ☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐

☐ 全选 ☐ ☐ ☐

AI 模拟量输入参数

- 1、线性 AI 输入支持 6 种信号类型：[4,20]mA、[0,20]mA、[1,5]V、[0,5]V、[0,10]V、[-10,10]V。

外部模拟输入

	信号类型	使用工程量	下限值	上限值	采样次数	零点修正
AI0	[4,20]mA	☒	0	5000	64	0
AI1	[0,20]mA	☒	0	27000	64	0
AI2	[1,5]V	☐			64	0
AI3	[-10,10]V	☐			64	0

- 2、热电偶模块支持：S、K、T、E、J、B、N、R、Wre3/25、Wre5/26、[0,20]mV、[0,50]mV、[0,100]mV。

外部模拟输入

	信号类型	使用工程量	下限值	上限值	采样次数	零点修正
AI4	E热电偶	☒	-2000	10000	64	0
AI5	J热电偶	☒	-2000	12000	64	0
AI6	K热电偶	☒	-2000	13000	64	0
AI7	B热电偶	☒	0	18000	64	0

- 3、热电阻模块支持：Pt100、Pt1000、Cu50、Cu100。

外部模拟输入

	信号类型	使用工程量	下限值	上限值	采样次数	零点修正
AI8	Pt100热电阻	☒	-2000	8500	64	0
AI9	Pt1000热电阻	☒	-500	3000	64	0
AI10	Cu100热电阻	☒	-500	1500	64	0
AI11	Cu50热电阻	☒	-500	1500	64	0

- 4、可以配置 AI 通道使用工程量，如果使用工程量则该通道量程由“上限值”和“下限值”决定，如果不使用工程量则采用 0~32000 的码值。
- 5、“采样次数”影响该 AI 通道的响应时间和滤波效果，采样次数越大那么响应时间越长、滤波效果越好；采样次数越小那么响应时间越短、滤波效果越差。
- 6、如果外部传感器产生零点偏差，这时可以通过设置“零点修正”来调整而无须更换传感器。传感器负偏差时零点修正值要设定为正值，传感器正偏差时零点修正值要设定为负值。

AQ 模拟量输出参数

- 1、AQ 输出支持类型：[4,20]mA、[0,20]mA、[1,5]V、[0,5]V、[0,10]V、[-10,10]V，每通道可以单独设置。
- 2、可以配置 AQ 通道使用工程量，如果使用工程量则该通道量程由“上限值”和“下限值”决定，如果不使用工程量则采用 0~32000 的码值。
- 3、当配置 AQ 通道“停机输出”时，可以给它设定一个输出值，这样在 PLC 停机时该 AQ 通道会保持设定的值输出。

外部模拟输出

	信号类型	使用工程量	下限值	上限值	停机输出	输出
AQ0	[4,20]mA	☒	0	1000	☐	
AQ1	[-10,10]V	☒	0	1000	☐	

HSC 高速计数器参数

- 1、高速计数器支持：脉冲/方向、正/反脉冲、A/B 相脉冲输入模式，支持 1、2、4 倍频计数模式。
- 2、高速计数器通道用 HSCx 表示，每个通道使用 2 个高速脉冲输入点。
- 3、如果在程序中没有使用高速计数器 **HHSC**，那么无需配置参数，相应的高速脉冲输入点可以当做普通 X 点使用。

高速计数器号

	计数模式	
HSC0	0 - 脉冲/方向 1倍频	脉冲:X0 方向:X1
HSC1	1 - 脉冲/方向 2倍频	脉冲:X2 方向:X3
HSC2	2 - 正转/反转 1倍频	正转脉冲:X4 反转脉冲:X5
HSC3	4 - A相/B相 1倍频	A相脉冲:X6 B相脉冲:X7

0 - 脉冲/方向 1倍频

1 - 脉冲/方向 2倍频

2 - 正转/反转 1倍频

3 - 正转/反转 2倍频

4 - A相/B相 1倍频

5 - A相/B相 2倍频

6 - A相/B相 4倍频

PLS 高速脉冲输出参数

- 1、高速脉冲输出支持：单脉冲、脉冲/方向、正/反脉冲、A/B 相脉冲、同步脉冲输出，共 5 种输出模式

- 2、高速脉冲输出通道用 PLSx 表示，每个通道使用 2 个高速脉冲输出点。
- 3、如果在程序中没有使用高速脉冲输出指令（[PLSY](#)、[PLSR](#)、[PPMR](#)、[CIMR](#) 等），那么无需配置参数，相应的高速脉冲输出点可以当做普通 Y 点使用。

脉冲输出通道号

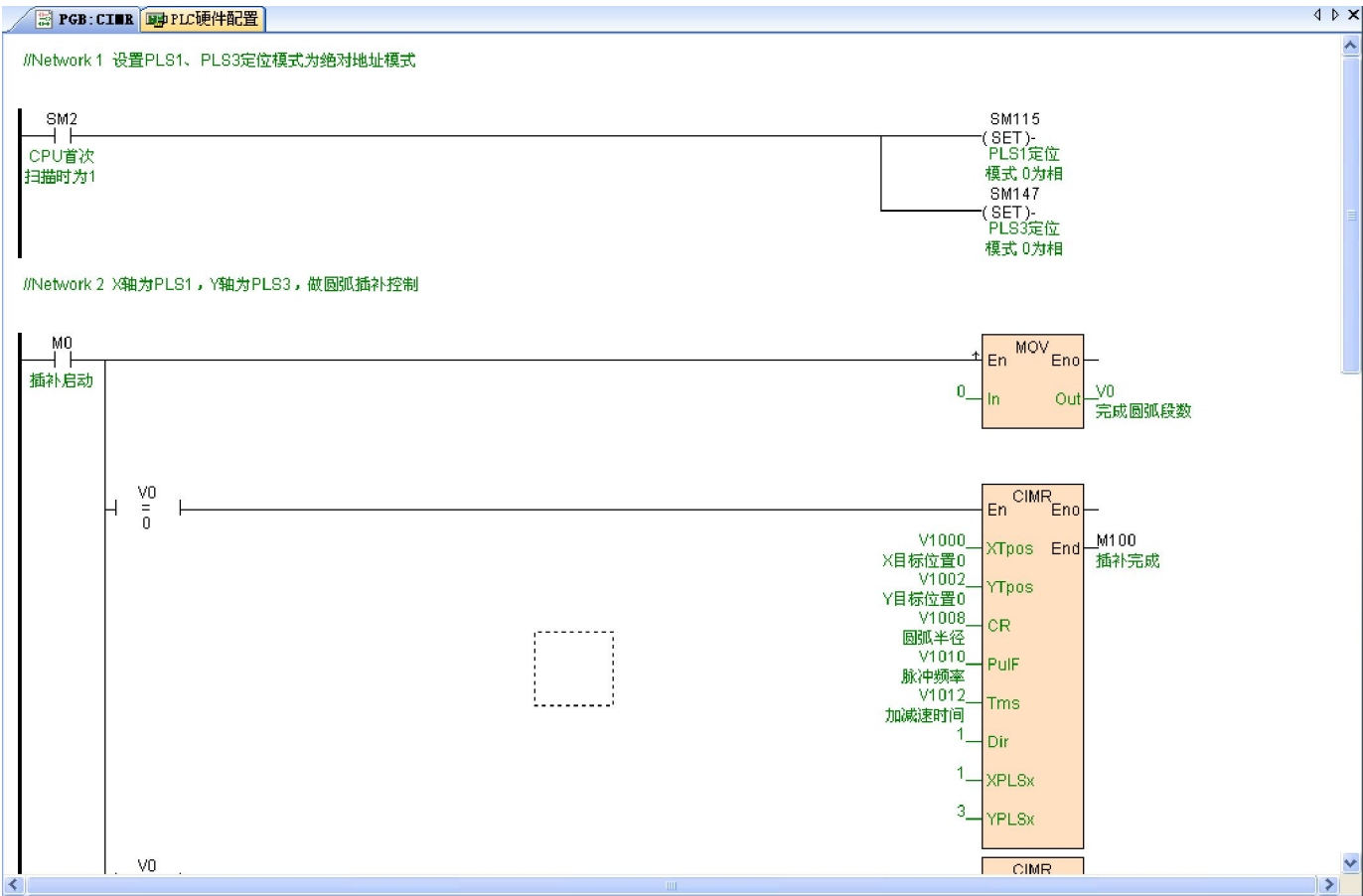
输出模式	
PLS0	0 - 单脉冲输出 脉冲:Y0
PLS1	2 - 正转/反转脉冲 正转脉冲:Y2 反转脉冲:Y3
PLS2	2 - 正转/反转脉冲 正转脉冲:Y4 反转脉冲:Y5
PLS3	3 - A相/B相脉冲 A相脉冲:Y6 B相脉冲:Y7

0 - 单脉冲输出
1 - 脉冲/方向输出
2 - 正转/反转脉冲
3 - A相/B相脉冲
4 - 同步脉冲输出

5.6 LD梯形图程序的编写

LD 工作区

在 LD 工作区进行 LD 语言的编辑，如增加网络、删除网络、调整网络顺序，串联或并联开关，增加、删除指令等。

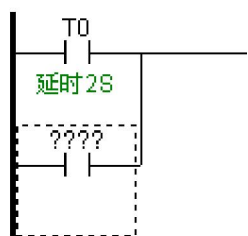


开关编辑

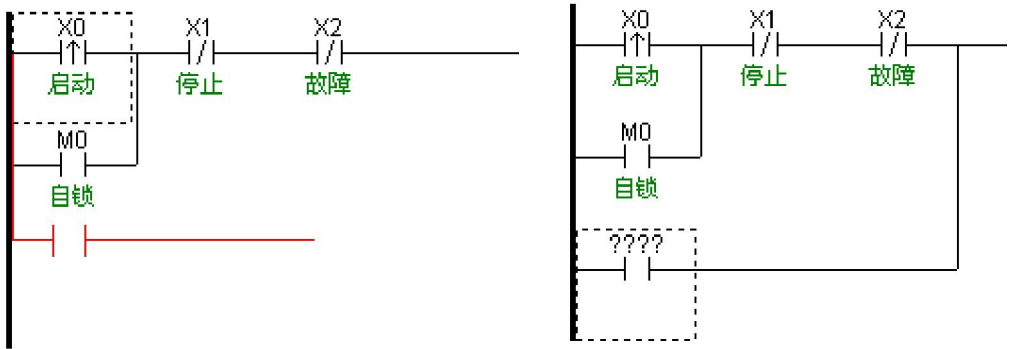
- 1、串联一个开关：点击按钮或者按“F9”。



2、并联一个开关：点击按钮或者按“F10”。



3、并联分支：点击工具栏按钮，鼠标变为，鼠标移到要 并联分支的起始位置点击（X0 开关位置），再移动鼠标到终点位置点击（X2 开关位置），即可并联一个分支。



4、鼠标点击或用键盘将虚框移到 开关位置，即可选择该开关。可输入开关元件、改变开关状态，可拷贝、剪切、删除等操作。

开关状态改变

鼠标点击或用键盘将虚框移到要改变状态的开关位置，用快捷键或鼠标右键调出右键菜单即可改变开关的状态。

SM0

M0

V8

RUN

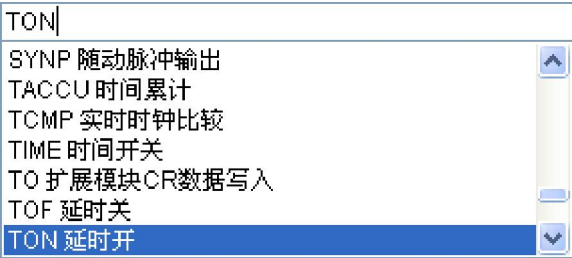
下步

	常开	Ctrl+1
	常闭	Ctrl+2
	上升沿	Ctrl+3
	下降沿	Ctrl+4
	STL 步开始	F7
	SFB 步合并	F8
16位比较开关		
	等于	Ctrl+5
	不等于	Ctrl+6
	大于	Ctrl+7
	大于等于	Ctrl+8
	小于	Ctrl+9
	小于等于	Ctrl+0
	网络上移	Ctrl+Up
	网络下移	Ctrl+Down
	删除	
	剪切	Ctrl+X
	复制	Ctrl+C
	粘贴	Ctrl+V
	查找	Ctrl+F
	程序属性	

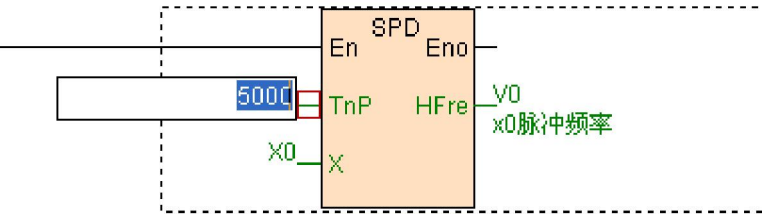
LD 指令编辑

- 1、双击工程管理器中指令树的指令名称即可将该指令添加到网络，也可使用拖拉方式将指令拖到相应位置 。
- 2、支持键盘输入指令 ，输入指令名称首字母时自动弹出指令输入窗口，输入指令名称按回车键即可增加指令，可随时按“ESC”关闭窗

□。



3、鼠标点击或用键盘将虚框移到指令位置，按回车键即可输入 或修改该指令的参数。



4、选择指令后，可以拷贝、剪切、删除指令，亦可上移、下移指令顺序。

5、双击指令可以打开“指令属性”窗口，完成改变指令模式、设置指令禁用、设置计时器时基等操作。双击输入、输出端子可以打开“端子”窗口，以配置方式输入端子参数。

指令输入端状态改变

当鼠标移到指令输入端时会出现一个红色方框，用鼠标右键调出右键菜单即可改变指令输入端的状态（常开、常闭、上升沿、下降沿）。



分支编辑

LD 语言支持并联输出和分支输出方式执行指令，如果执行指令的逻辑条件相同时使用并联输出；如果执行指令的逻辑条件不同时则必须使用分支输出。

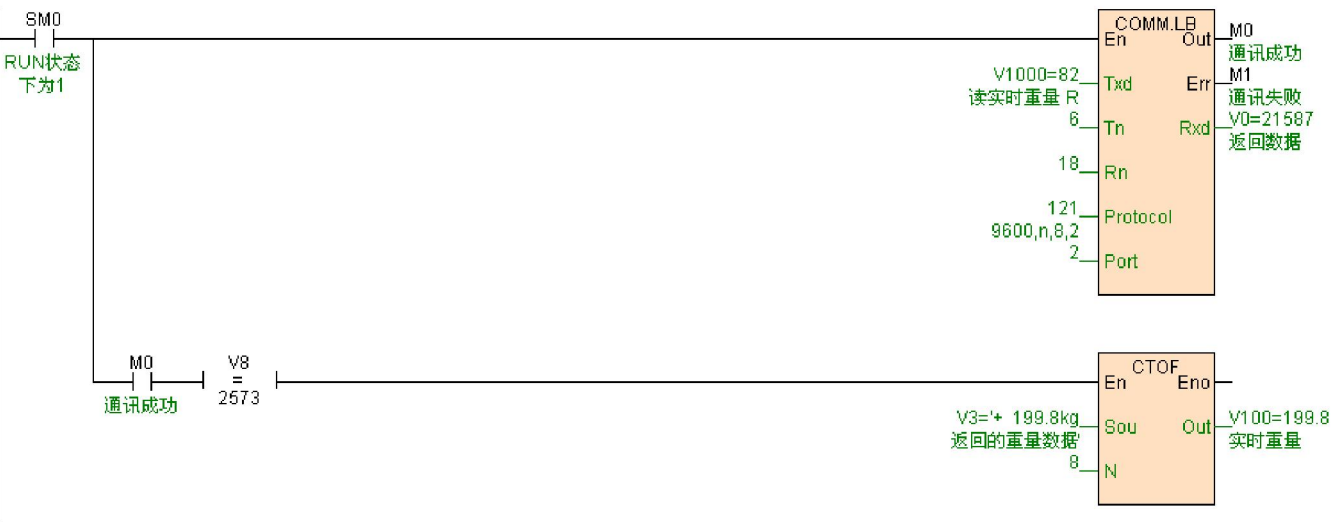
1、如下图，执行指令 FSIN、FCOS 和 FTAN 的逻辑条件相同使用并联输出。

//Network 1 求正弦、余弦、正切



2、如下图，执行指令 COMM.LB 和 CTOF 的逻辑条件不同使用分支输出。

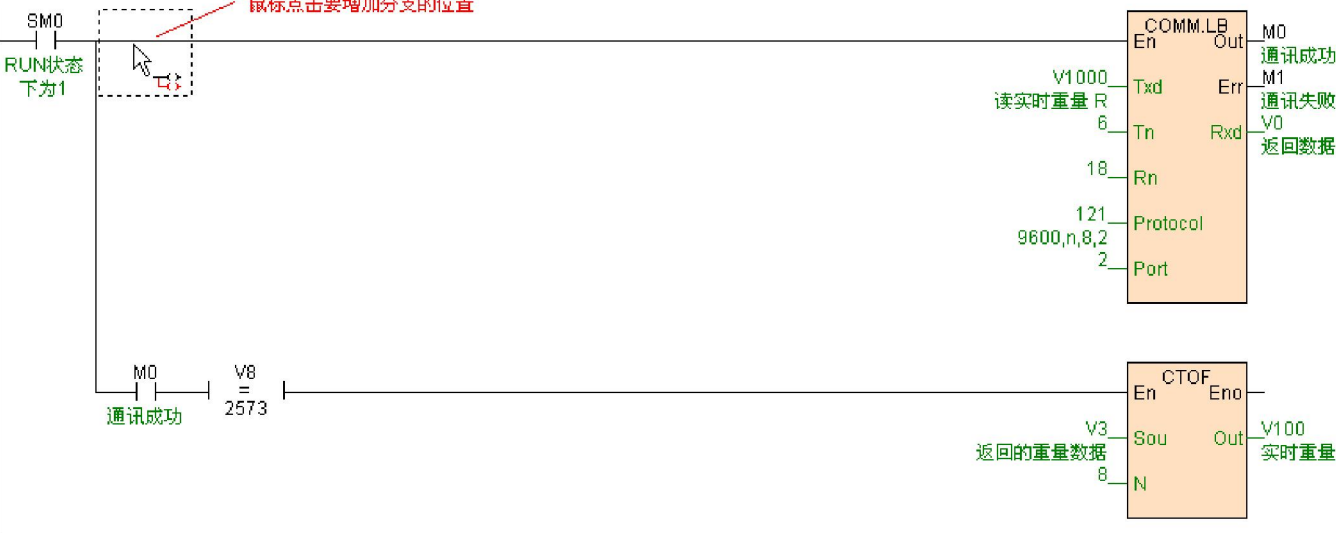
//Network 1 采用CB920通讯协议，读实时重量，通讯成功则M0=ON，同时判断最后2字节内容是否为0x0A0D，正确则解析出实时重量到V100



3、添加指令默认为并联输出，分支输出需要先增加新分支后再增加指令。

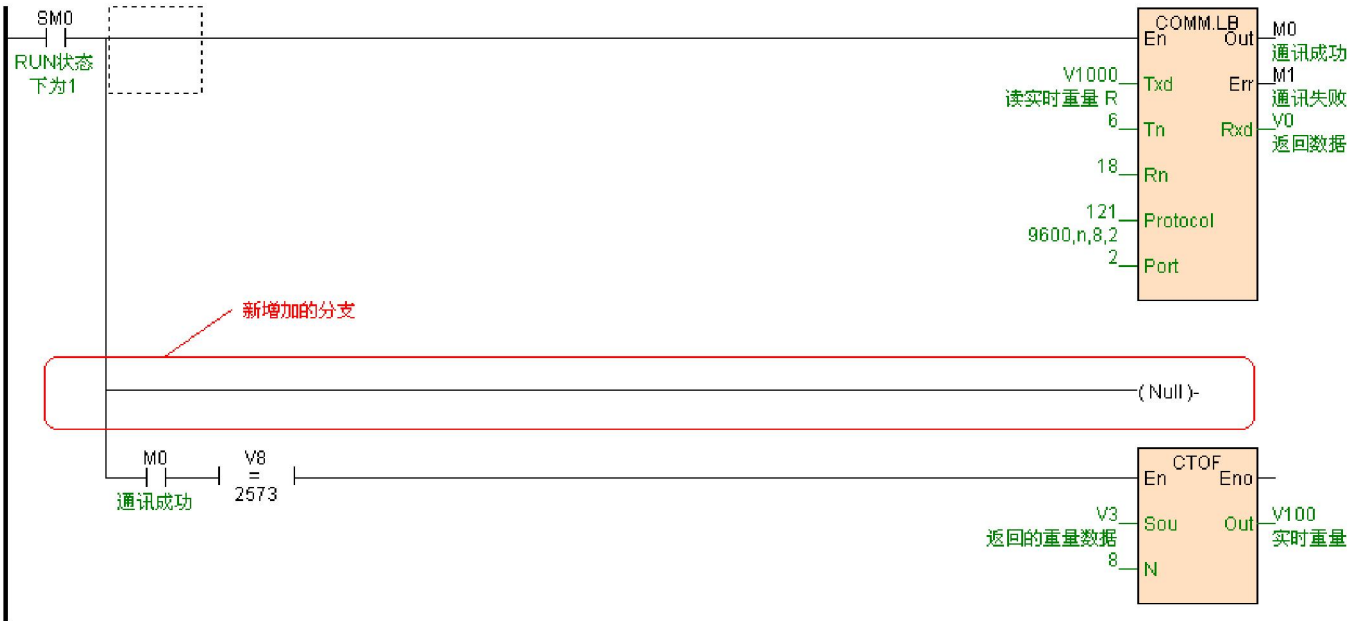
4、点击工具栏  按钮，鼠标变为 ，鼠标移到要增加分支的位置点击即可增加新分支，新分支将添加在当前分支的后面。
增加分支前如下图：

//Network 1 采用CB920通讯协议，读实时重量，通讯成功则M0=ON，同时判断最后2字节内容是否为0x0D0A，正确则解析出实时重量到V100



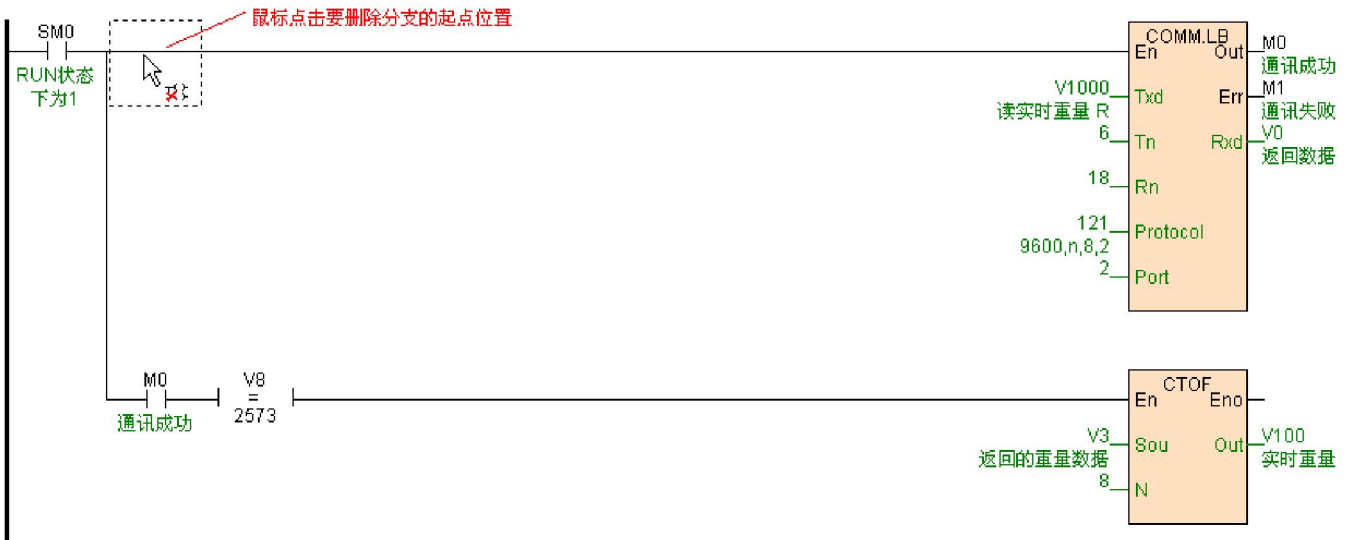
增加分支后如下图：

//Network 1 采用CB920通讯协议，读实时重量，通讯成功则M0=ON，同时判断最后2字节内容是否为0x0D0A，正确则解析出实时重量到V100



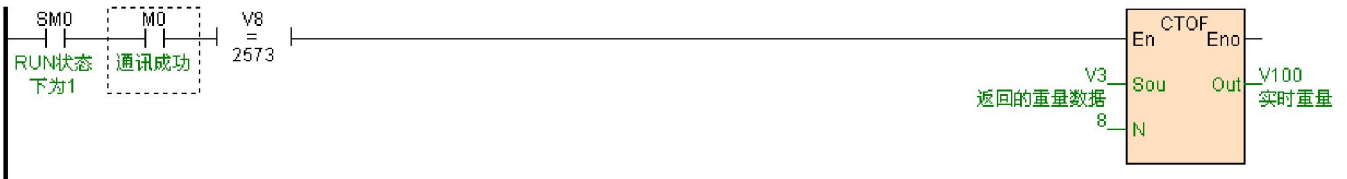
- 5、点击工具栏按钮，鼠标变为，鼠标移到要删除分支的起点位置点击即可删除该分支。
删除分支前如下图：

//Network 1 采用CB920通讯协议，读实时重量，通讯成功则M0=ON，同时判断最后2字节内容是否为0x0D0A，正确则解析出实时重量到V100



删除分支后如下图：

//Network 1 采用CB920通讯协议，读实时重量，通讯成功则M0=ON，同时判断最后2字节内容是否为0x0D0A，正确则解析出实时重量到V100



网络编辑

- 1、点击按钮或者用快捷键 Ctrl+L 添加一个新网络。
- 2、点击按钮或者用快捷键 Ctrl+I 插入一个新网络。

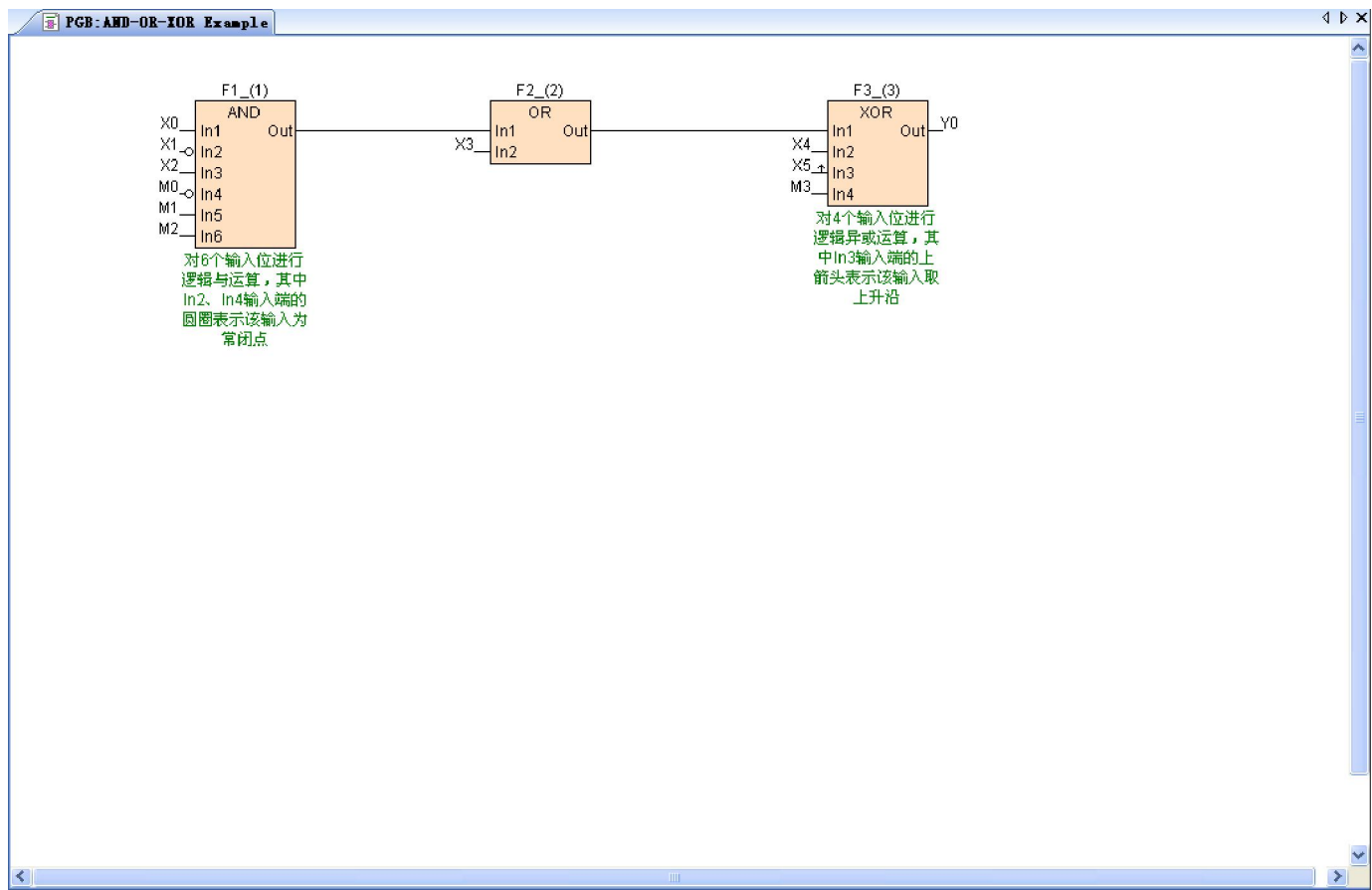
- 3、鼠标在网络上下拉，可以选择当前网络或多个网络，被选择的网络内容以反白显示，按“Del”键即可删除，亦可剪切、拷贝、粘贴等操作。
- 4、鼠标双击网络标题可以编辑网络注释，当虚框在网络标题位置时可以直接输入网络注释。

5.7 FBD功能块程序编写

FBD 工作区

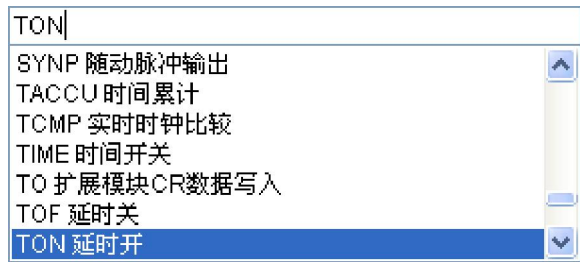
在 FBD 工作区进行 FBD 语言的编辑，如增加新页、增加、删除指令、连接指令、改变指令执行顺序等。

- 1、FBD 指令的顶部功能块序号和它的执行顺序号，如：“F3_(3)”，“F3”表示第 3 个功能块，“_(3)”表示该功能块的执行顺序号为 3。
- 2、改变指令间的连接线系统会自动调整功能块的执行顺序，亦可手动调整功能块执行顺序。

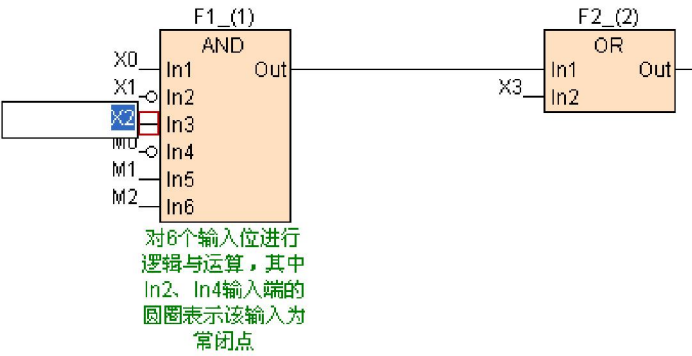


FBD 指令编辑

- 1、双击工程管理器中指令树的指令名称即可添加该指令到程序中，也可使用拖拉方式将指令拖到相应位置。
- 2、支持键盘输入指令，输入指令名称首字母时自动弹出指令输入窗口，输入指令名称按回车键即可增加指令，可随时按“ESC”关闭窗口。



- 3、鼠标点击指令，按回车键即可输入或修改该指令的参数。

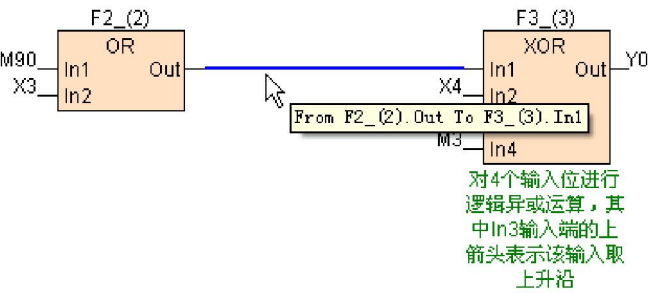


- 4、用鼠标拖拉方框可以任意选择指令，可以拷贝、剪切、删除指令。
- 5、双击指令可以打开“指令属性”窗口，完成改变指令模式、设置指令禁用、设置计时器时基等操作。双击输入、输出端子可以打开“端子”窗口，以配置方式输入端子参数。

FBD 指令间连接

FBD 指令间可以用能流线连接。

- 1、点击工具栏按钮，鼠标变为，鼠标移到 指令的输入或输出端子，如果允许连接鼠标变为，如果不允许连接鼠标变为。
- 2、指令输出端不能连接到自身的输入端，输入端只能有一个能流输入，输出端可以输出多个能流。
- 3、能流线的连接会自动改变功能块指令的执行顺序。
- 4、指令间可以不使用能流线连接，直接输入指令的参数即可。
- 5、鼠标点击能流线会提示该能流的详细连接信息。



改变 FB 执行顺序

鼠标右键点击指令，调出右键菜单点击【改变 FB 执行顺序】，输入新执行顺序，按“确定”即可。如果 FB 之间是通过能流连接，新执行顺序不能改变能流顺序。



指令输入端状态改变

当鼠标移到指令输入端时会出现一个兰色方框，用鼠标右键调出右键菜单即可改变指令输入端的状态（常开、常闭、上升沿、下降沿）。



页面编辑

- 1、点击按钮添加一个新页。
- 2、点击按钮删除最后一页，如果该页有指令存在则不能删除。

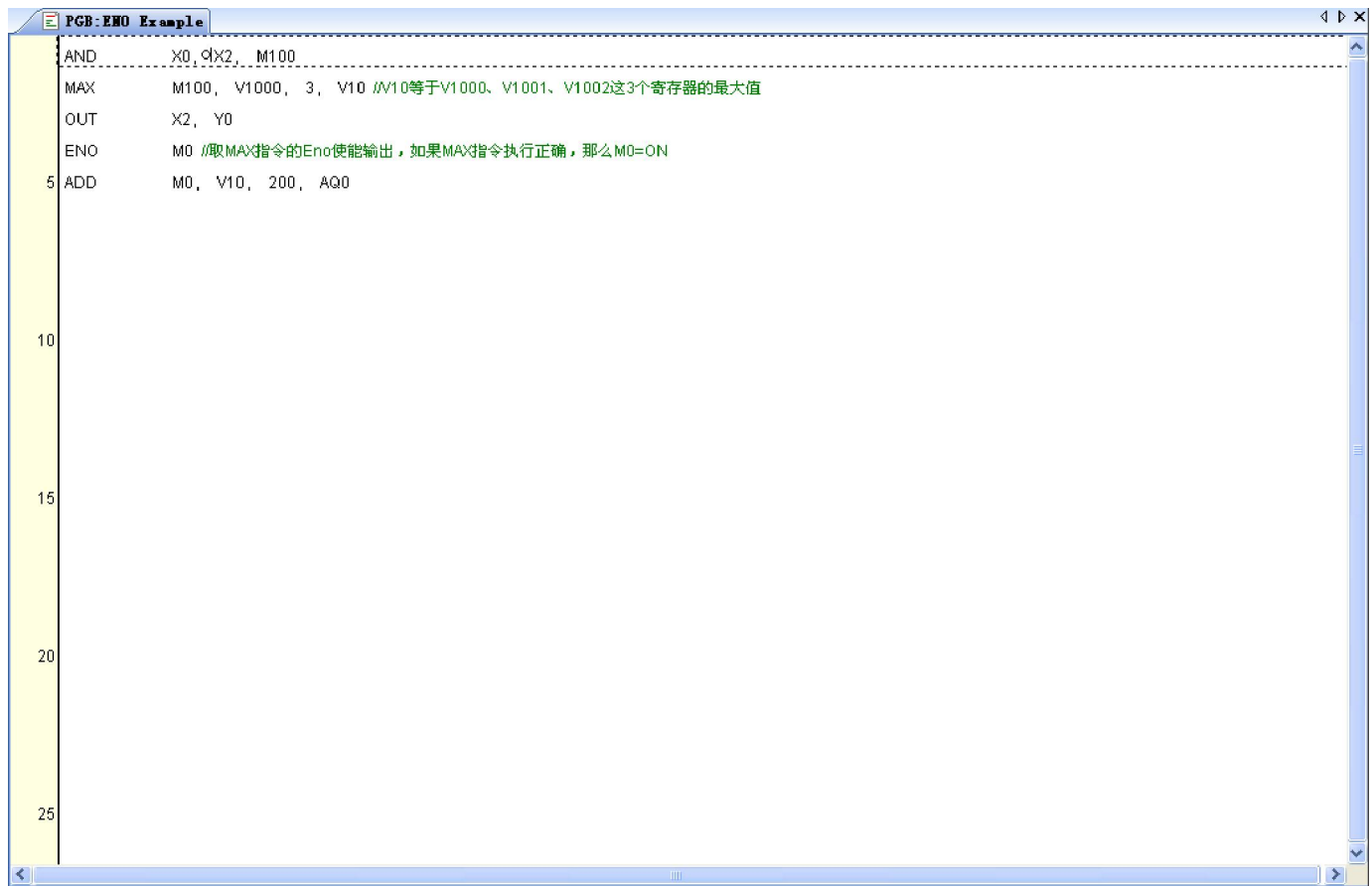
FBD 注释

FBD 语言可以给每一条指令写注释，双击指令可以打开“指令属性”窗口，在“注释”框写入注释即可。

5.8 IL指令表程序编写

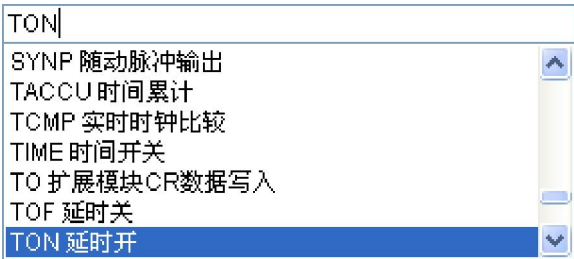
IL 工作区

在 IL 工作区进行 IL 语言的编辑，如增加、删除指令、上移或下移指令等。

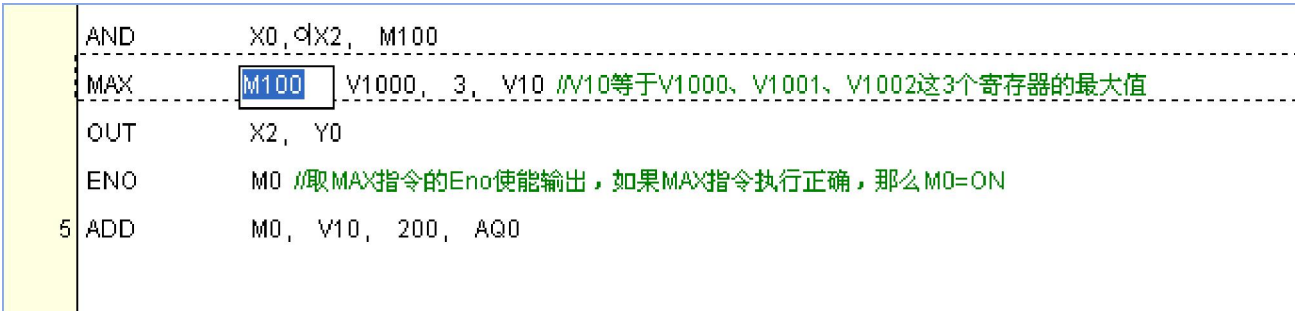


II. 指令编辑

- 1、双击工程管理器中指令树的指令名称即可添加该指令到程序尾部，也可使用拖拉方式将指令拖到相应位置。
- 2、支持键盘输入指令，输入指令名称首字母时自动弹出指令输入窗口，输入指令名称按回车键即可增加指令，可随时按“ESC”关闭窗口。



- 3、鼠标点击或用键盘将虚框移到指令位置，按回车键即可输入或修改该指令的参数。



- 4、用鼠标下拉可以任意选择指令，可以拷贝、剪切、删除指令，亦可上移、下移指令顺序。
- 5、双击指令可以打开“指令属性”窗口，完成改变指令模式、设置指令禁用、设置计时器时基等操作。双击输入、输出端子可以打开“端子”窗口，以配置方式输入端子参数。

指令输入状态改变

鼠标右键点击指令输入端，调出右键菜单即可改变指令输入端的状态（常开、常闭、上升沿、下降沿）。



II. 注释

IL 语言可以给每一条指令写注释，双击指令可以打开“指令属性”窗口，在“注释”框写入注释即可。

5.9 其他

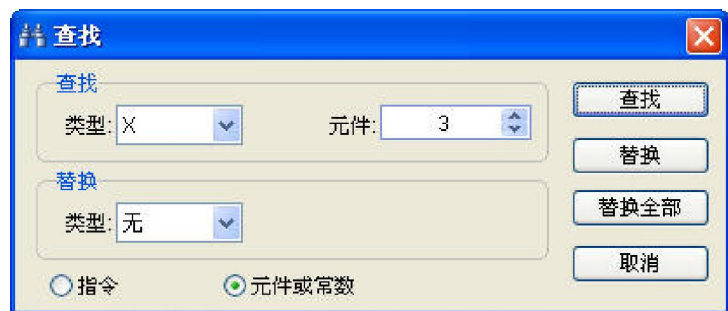
查找、替换

通过菜单【编辑/查找】或快捷键“Ctrl+F”打开“查找”窗口，可以任意查找指令、元件或常数，可以替换元件或常数。

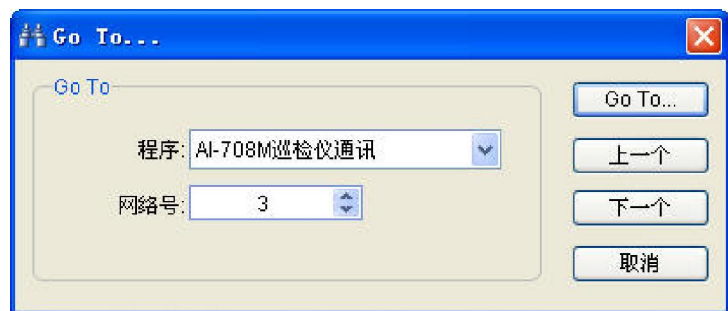
1、指令的查找。



2、元件或常数的查找或替换。

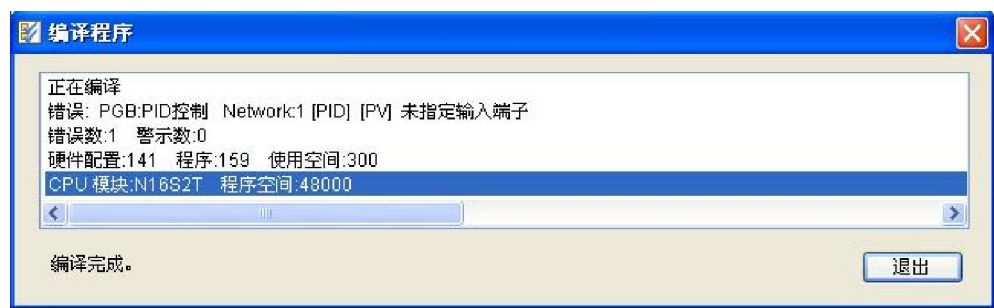


3、通过菜单【编辑/Go To...】或快捷键“Ctrl+G”打开“Go To...”窗口，可以实现快速定位。LD 语言快速定位到网络，FBD 语言快速定位到功能块，IL 语言快速定位到行。



编译程序

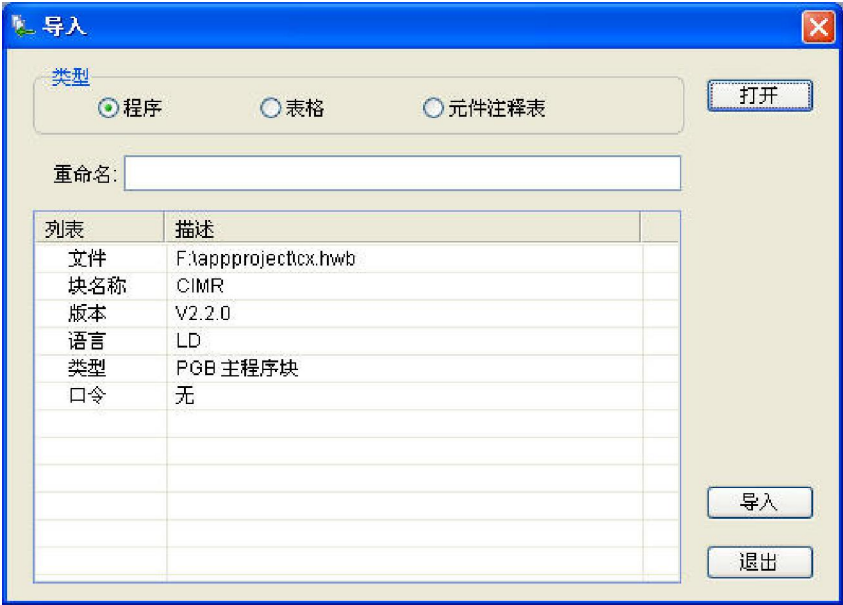
在编程过程中可以随时对程序进行编译操作，以检查程序是否存在错误。通过菜单【调试/编译程序】或点击工具栏按钮启动编译操作。编译窗口如下：



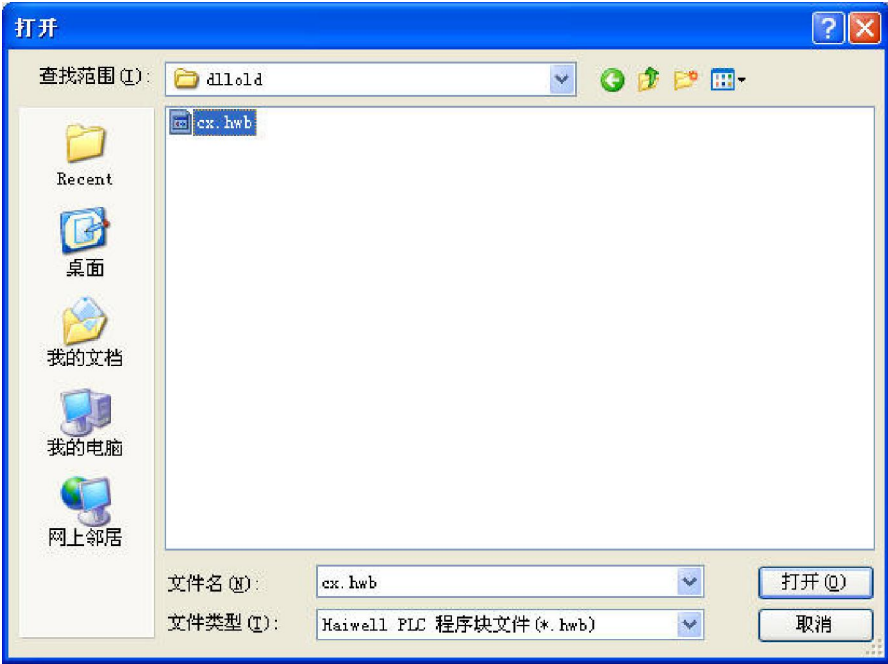
- 1、下载程序或者启动仿真器运行程序时，系统会自动启动编译操作，只有没有任何错误的程序才可以被下载或者启动仿真器运行。
- 2、编译窗口会列出编译结果的错误数和警示数以及它们的详细信息。双击错误或警示信息可以快速定位到产生错误或警示的程序位置，方便用户修改程序。
- 3、错误数为 0 表示无错误，警示数为 0 表示无警示。程序的错误必须被修改，警示信息可以忽略。

程序及表格的导入

- 1、通过菜单【文件/导入】打开“导入”窗口。



- 2、点击“程序”、“表格”、“元件注释表”选择要导入的文件类型，点击“打开”按钮选择要导入的文件。



- 3、点击“导入”按钮将文件导入到当前程序项目中。

程序及表格的导出

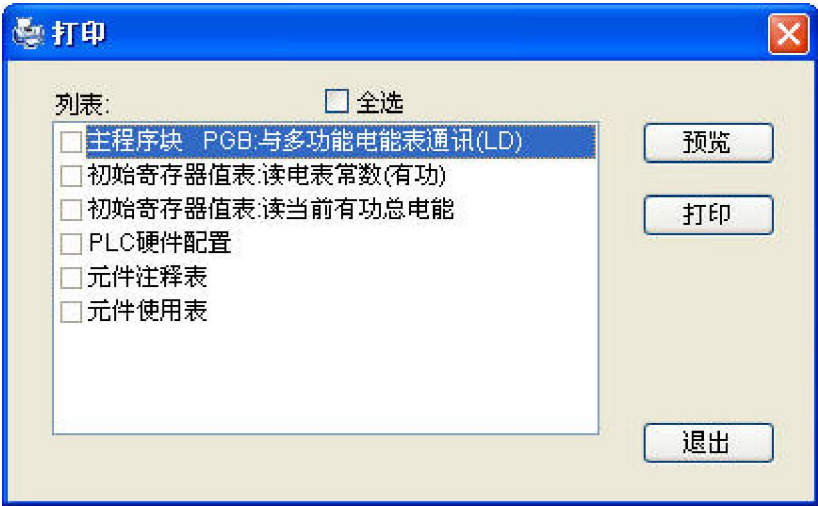
可以将当前程序项目中的程序块、表格及元件注释表导出，方便在其他程序项目中导入使用。通过菜单【文件/导出】打开“导出”窗口，选择要导出的内容按“导出”按钮即可。



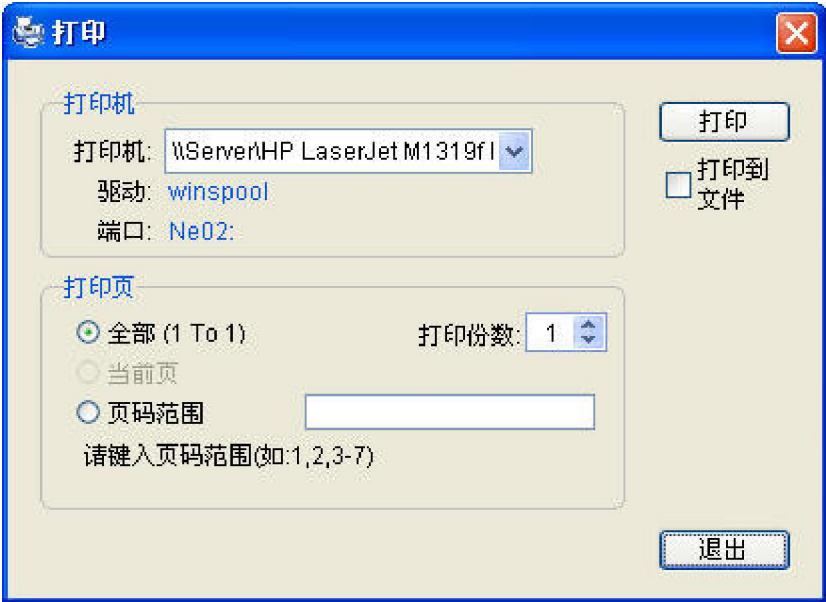
注：如果程序块、表格有设置保护口令，则导出的文件具同样的保护口令。

打印及预览

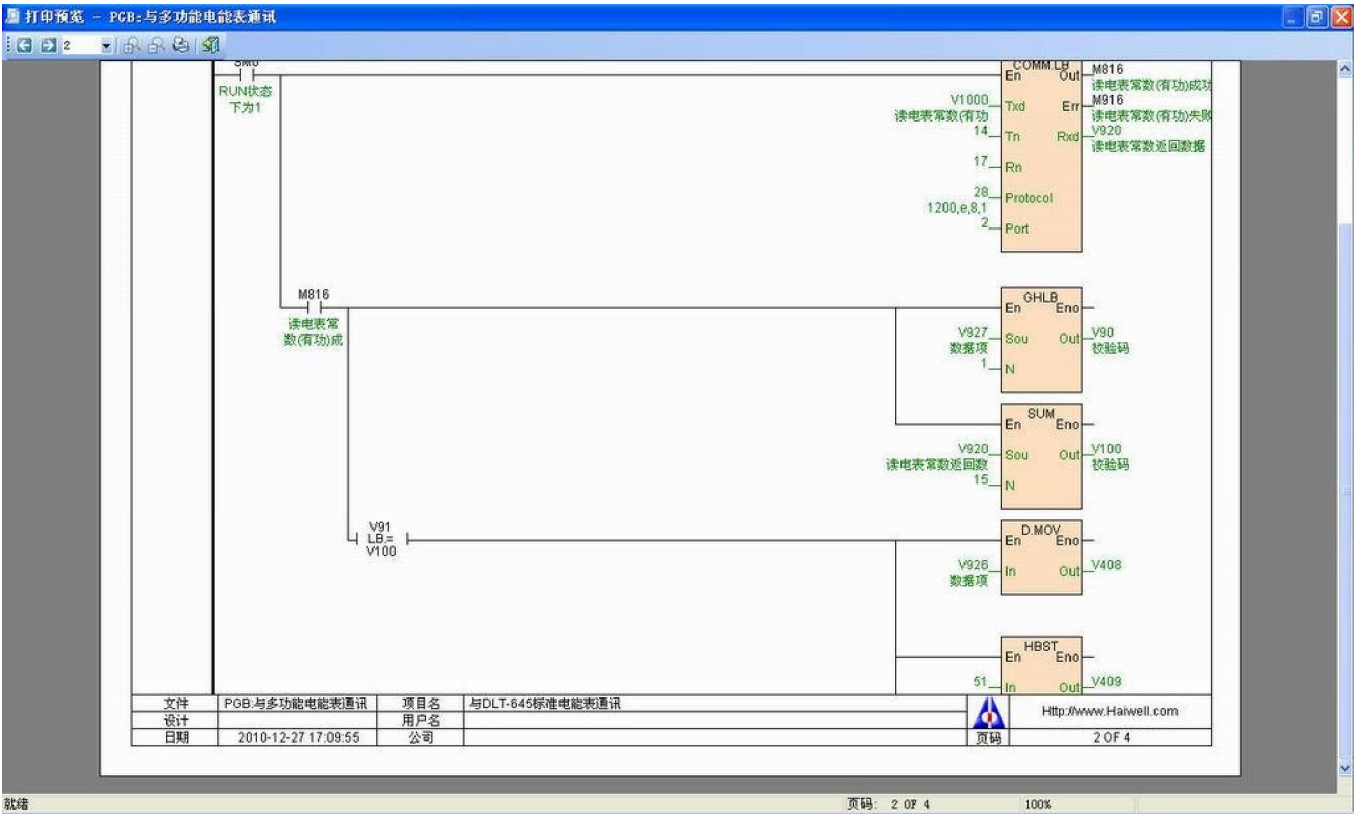
如果需要将程序内容以工程文件方式打印输出，通过菜单【文件/打印】或点击工具栏按钮，打开“打印”窗口。



- 1、选择要打印的内容，点击“打印”按钮打开“打印”属性窗口，在本窗口可以选择打印的份数、页数等信息，也可以选择“打印到文件”输出。



2、通过菜单【文件/打印预览】或点击工具栏按钮对要打印内容进行预览。



元件使用表

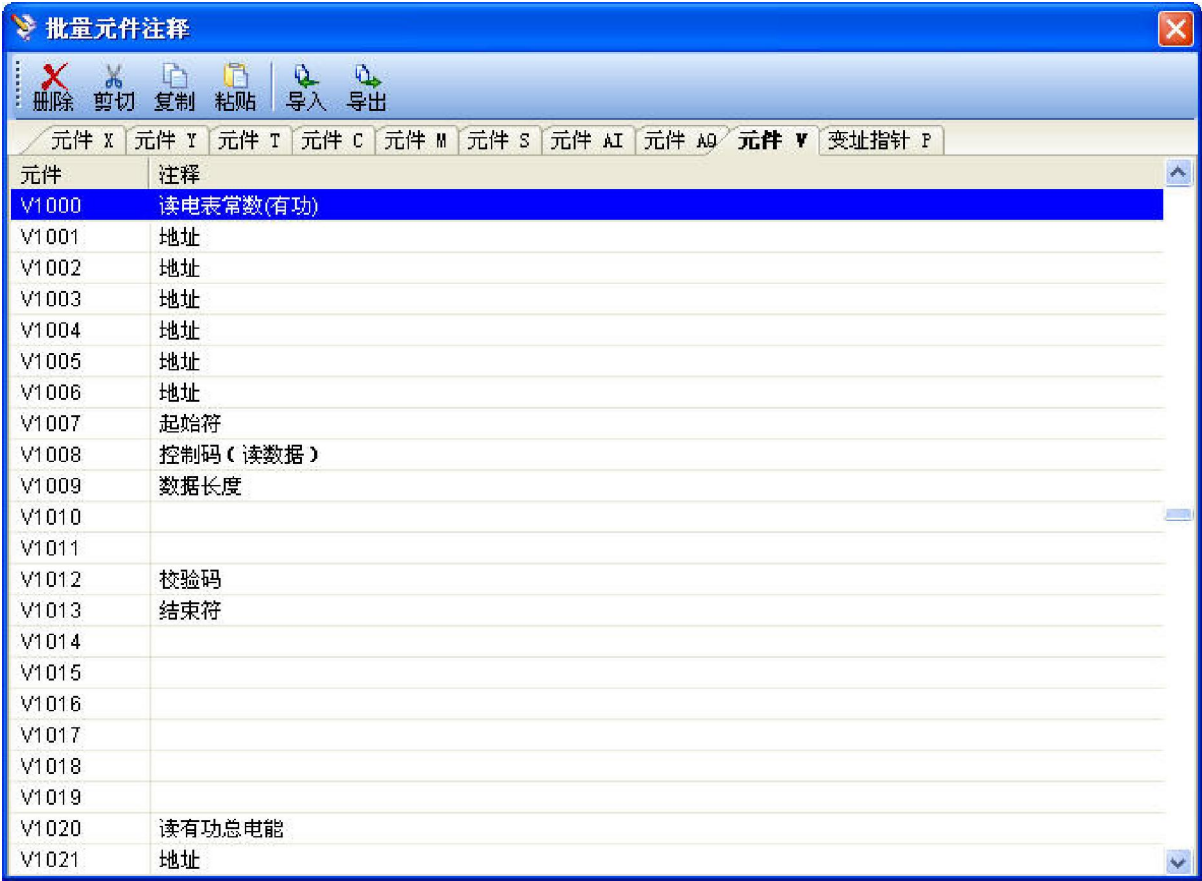
“元件使用表”列出当前程序项目中所有被使用的元件，并且详细统计出元件的读次数、写次数及指令自动占用连续元件（读/写）次数。
注：如果“写次数”加上“指令自动占用连续元件的写次数”大于 1，那么表示该元件存在多输出情况。

元件使用表 已使用元件 60													
X	Y	T	C	M	SM	AI	AQ	SV	V	TV	CV	S	P
元件	读次数		写次数		指令自动占用连续元件(读/写)				元件注释				
V90				1						校验码			
V91	1					0 / 1				校验码			
V100	1			1						校验码			
V102	1			1						校验码			
V408	1			1									
V409				1		1 / 1							
V520				1						电表常数(有功)			
V521						0 / 1							
V522				1						(当前)有功总电能			
V523						0 / 1							
V920	1			1						读电表常数返回数据			
V921						1 / 1							
V922						1 / 1							
V923						1 / 1							
V924						1 / 1							
V925						1 / 1							
V926	1					1 / 1				数据项			
V927	1					2 / 1				数据项			
V928						0 / 1							
V929	1			1									
V930						1 / 1							
V940	1			1						读有功总电能返回数据			

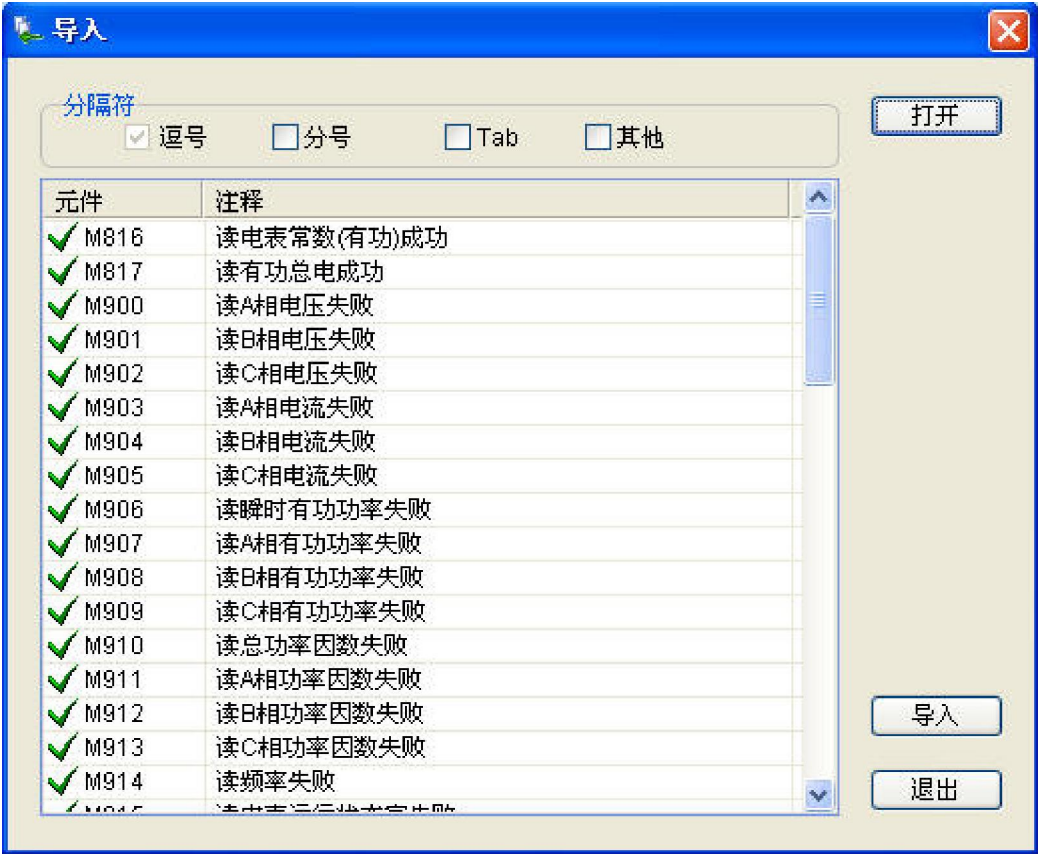
元件注释

HaiwellHappy 编程软件提供了丰富的注释功能，如：元件注释、网络注释、指令注释、程序块注释、表格注释和程序项目注释等，注释可选择下载到 PLC 中，方便日后上载程序的阅读或修改。

- 1、元件注释可通过在元件后跟“//”直接输入，如：X0//电机启动，X0 为元件，“电机启动”为 X0 的注释。
- 2、元件注释可以批量编辑，通过菜单【工具/批量元件注释】打开“批量元件注释”窗口。



- 3、按“导出”按钮可以将所有元件注释导出到文件存储。
- 4、按“导入”按钮可以将外部元件注释文件导入。



第六章、仿真与在线调试

本章介绍编程软件内置仿真器的使用和在线调试程序的方法。

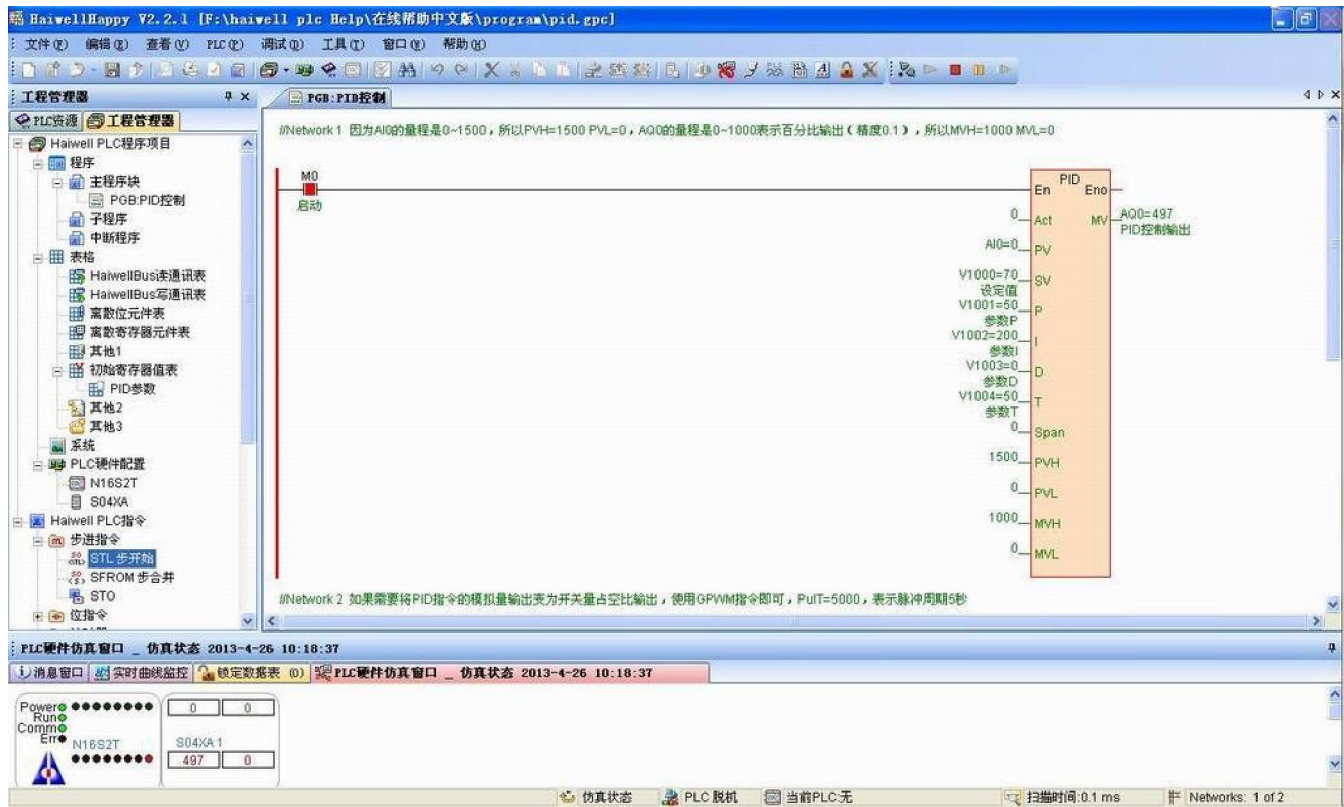
6.1 概述

编程软件内置仿真器（离线仿真），全面实现了 PLC 程序的仿真运行。您在编程过程中间或程序编写完成后，可用仿真器在完全脱离 PLC 的情况下仿真运行 PLC 程序，以检查程序执行是否正确，极大地减少现场调试时间，降低调试难度，提高调试效率。

编程软件还提供丰富的调试工具，方便进行在线联机调试程序，可搜索出与计算机连接的所有 PLC，显示出所有在线 PLC 的运行状态、故障状态、RUN/STOP 开关位置、硬件配置、通讯端口参数等详尽信息，可选择对任意一台 PLC 进行在线监控，查看 PLC 所有元件的数值或状态。

6.2 仿真环境

在仿真或在线监控状态下，主体窗口的底部将出现一个“PLC 硬件仿真窗口”。共分为四个页面窗口：消息窗口、实时曲线监控、锁定数据表、PLC 硬件仿真窗口，同时在左上角会出现仿真工具栏，配合右键菜单，使仿真更加接近真实环境。



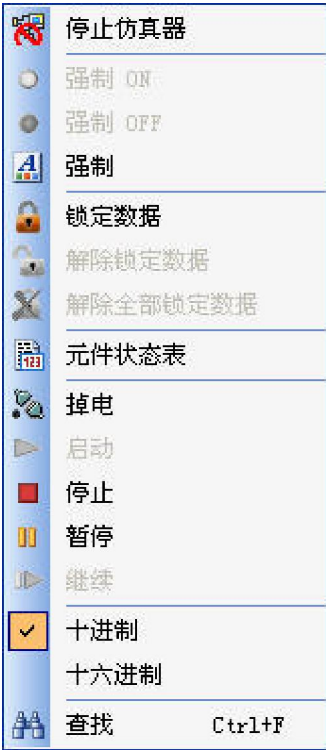
仿真工具栏

仿真工具栏用于控制仿真器的掉电、上电、启动、停止、暂停和继续操作。



右键菜单

在仿真或监视状态下，单击右键可调出右键菜单。右键菜单提供更加全面的仿真器操控和其他调试工具。



- 1、停止仿真器：结束当前的仿真状态，返回编辑状态。
- 2、强制 ON：强制鼠标位置的位元件状态为 ON。
- 3、强制 OFF：强制鼠标位置的位元件状态为 OFF。
- 4、强制：打开“强制”窗口，强制元件的状态或数值。
- 5、锁定数据：打开“锁定”窗口，锁定元件的状态或数值。
- 6、解除锁定数据：对鼠标位置的元件解除锁定数据。
- 7、解除全部锁定数据：解除全部被锁定数据的元件。
- 8、元件状态表：打开“元件状态表”，可任意监控 PLC 所有元件的状态或数值。
- 9、掉电/上电：仿真 PLC 掉电/上电的情景。
- 10、启动：控制仿真器启动运行。
- 11、停止：控制仿真器停止运行。
- 12、暂停：暂停仿真器运行。
- 13、继续：让仿真器继续运行。
- 14、十进制：以十进制显示数值。
- 15、十六进制：以十六进制显示数值。
- 16、查找：打开“查找”窗口，查找程序中的元件或指令。

硬件仿真窗口

“硬件仿真窗口”列出当前程序项目的硬件配置，显示主机和模块的名称以及所有 X（DI）或 Y（DO）通道的状态和 AI、AQ 通道的数值。



1、点击 X 或 Y 通道可以强制该通道的状态，可按“ESC”取消强制。



2、点击 AI 或者 AQ 通道可以打开“AI/AO 模拟窗口”修改该通道数值。

AI/AQ 模拟窗口

点击“硬件仿真窗口”中的 AI 或者 AQ 通道可以打开“AI/AO 模拟窗口”，窗口显示该通道的信号类型、信号值、信号值对应的工程量值或码值等，可修改该通道的数值（仿真状态可修改输 AI 入和 AQ 输出，在线监控只可修改 AQ 输出），可按“ESC”关闭窗口。



数据锁定窗口

“数据锁定窗口”列出所有被锁定数据的元件以及它的状态或数值。

锁定数据表 (5)						
序号	元件	16位寄存器值	32位寄存器值	浮点数	字符	元件注释
1	X0	On				启动
2	V0	0(0x0000)	1092616192(0x41200000)	10.0	'□□□□'	
3	AI0	500(0x01F4)	500(0x000001F4)	0.0	'?□□'	

实时曲线窗口

“实时曲线窗口”以实时曲线形式监控寄存器元件数值的变化趋势，方便用户对某些重要参数进行动态的观察与调试。



消息窗口

“消息窗口”显示在仿真或调试过程中出现的系统信息，如中断信息、系统错误信息、通讯错误信息等，用户通过查看该窗口中的信息可得知当前程序的运行情况。

消息窗口	
2013-4-28 10:01:01	中断:33 HSC0当前值=设定值(每段设定值都会产生)
2013-4-28 10:01:08	中断:34 HSC0输入方向改变
2013-4-28 10:01:17	中断:34 HSC0输入方向改变

仿真状态 PLC 脱机 当前PLC:无 扫描时间:15.0 ms

6.3 仿真操作

脱机仿真的一般步骤

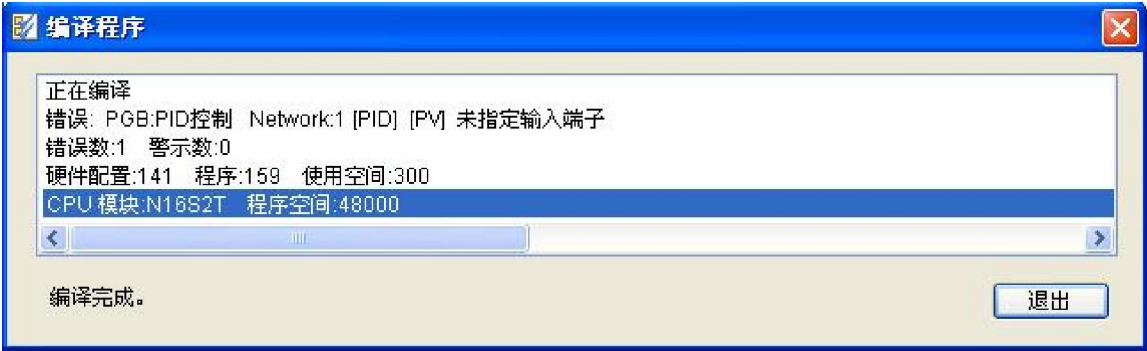
- 1、启动仿真器，进入仿真状态。
- 2、根据需要对元件进行强制，修改元件的状态或数值，使程序尽可能地模拟现场实际的情况执行。在仿真过程中，可用“元件状态表”、

“实时曲线监控”或“PLC 硬件仿真”等众多仿真调试工具对程序仿真运行的结果进行实时监控，以验证程序运行的结果是否正确。

- 3、如果程序有使用通讯指令，那么可以启动“通讯仿真器”仿真从机设备返回的数据。
- 4、如果程序有使用插补等运动控制指令，那么可以启动“插补仿真器”以图形仿真出插补输出的结果。
- 5、仿真 PLC 停电/上电过程，测试程序是否在经过停电后仍然能够保持正常工作。
- 6、停止仿真器，返回编辑状态。如果程序运行结果不正确，那么修改程序然后重新仿真运行。

启动仿真器

点击菜单【调试/启动仿真器】或点击工具栏的按钮启动仿真器，系统将自动对当前程序项目进行编译，若程序编译有错误或警告，将弹出如下窗口。

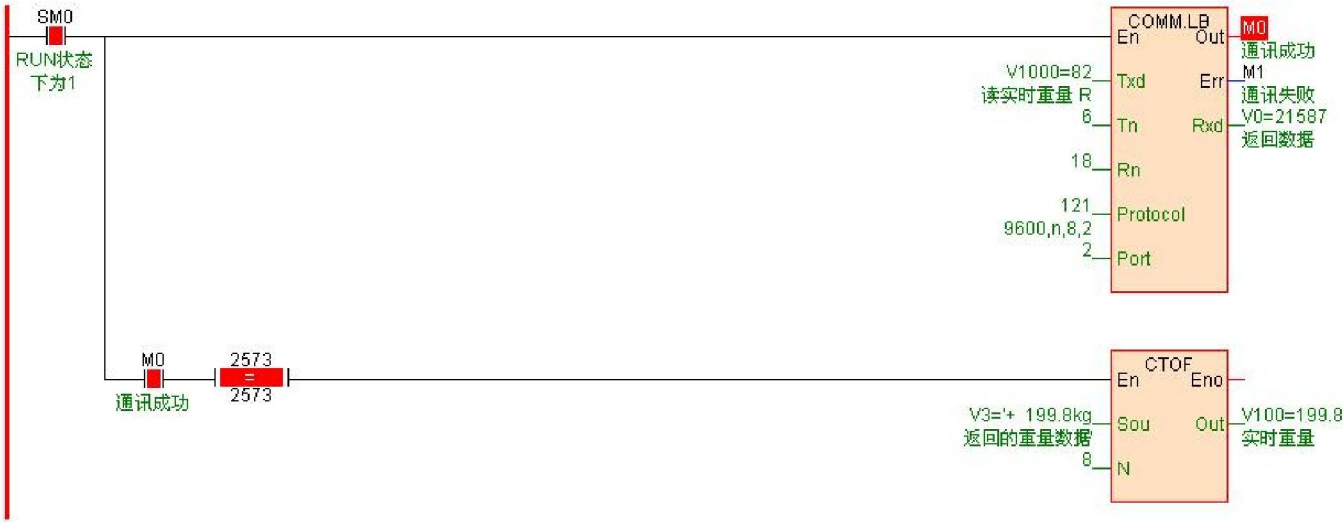


- 1、在列表框中列出所有的错误和警告信息，双击其中的错误或警告信息，程序将精确定位到产生该错误或警告的位置，便于用户修改程序。
- 2、若程序编译错误，仿真器不能启动，直到用户将程序修改至无任何错误后方可启动仿真器。
- 3、若程序编译无错误或仅有警告时，可以启动仿真器。如果有警告信息，建议您将程序修改至无任何警告消息后再仿真。
- 4、一旦启动仿真器后，编程软件便进入脱机仿真状态。在仿真状态下，不可对程序进行修改，只有在停止仿真器返回到程序编辑状态后，方可对程序进行修改。

元件监控

在仿真或在线监控状态下，可以通过程序编辑工作区、元件状态表、硬件仿真窗口等地方监控元件的状态或数值，以及指令的运行情况。下图为在仿真或在线监控状态下的程序编辑工作区。

//Network 1 采用CB920通讯协议，读实时重量，通讯成功则M0=ON，同时判断最后2字节内容是否为0x0D0A，正确则解析出实时重量到V100



- 1、指令的外框为红色时，表示该指令执行正确；为蓝色时，表示该指令未执行或者执行有错误（指令参数有错误，如 DIV 指令的除数为 0）。

- 2、寄存器元件将显示出它的当前数值，可通过菜单【查看\十进制、十六进制】改变寄存器显示的形式。
- 3、浮点数或者字符格式的元件会根据指令端子的定义自动显示。如上图：CTOF 指令，Sou 输入的是字符串，所以字符串显示 V3='+ 199.8kg'，Out 端子输出的是浮点数，所以浮点数显示 V100=199.8。
- 4、常开开关和比较开关内带红色方块，表示该开关接通，否则不通。注意：常闭开关与常开开关状态相反；上升沿和下降沿开关因为是边沿，仅产生边沿是通一次，其他时间都不通。
- 5、位元件带红色方块，表示该元件状态为 ON，否则为 OFF。
- 6、如果元件带，表示该元件被锁定数据。
- 7、双击元件可以强制该元件状态或数值。

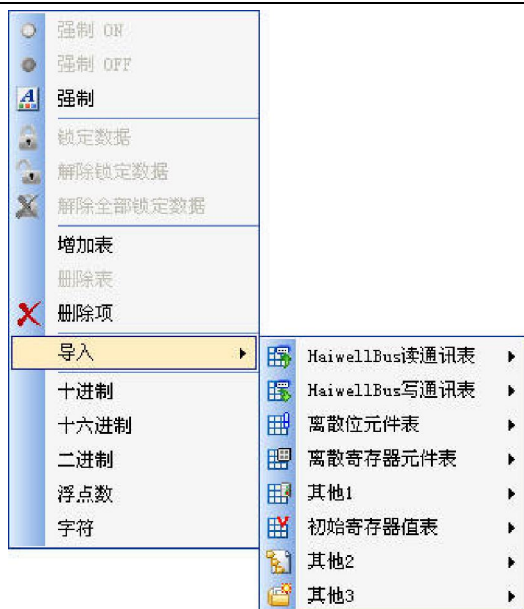
元件状态表

“元件状态表”可以任意监控 PLC 所有元件的当前状态或数值，最大可建立 10 个状态表，允许以不同数据格式显示寄存器元件的数值。它只有在仿真或在线监控的状态可用。

- 1、点击菜单【调试/元件状态表】或点击工具栏按钮，打开“元件状态表”



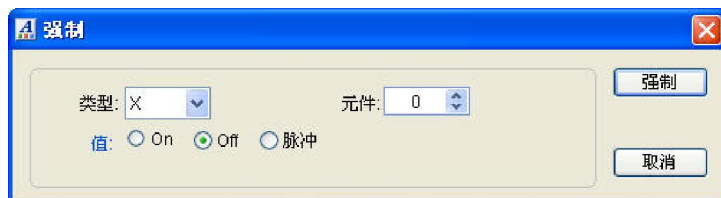
- 2、单击“元件”栏下面的空白处，在输入框中输入需要监控的元件或者元件范围，例如：上图分别输入 V0-8、V100、V1000-1005、M0。
- 3、对寄存器元件同时显示它的“16 位寄存器值”和“32 位寄存器值”。注意：“32 位寄存器值”一栏显示的是相邻 2 个连续寄存器的值，如上图第一行 V0 元件的“32 位寄存器值”是 V1V0 的数值。
- 4、如果元件带，表示该元件被锁定数据。
- 5、双击行可以强制该元件状态或数值。
- 6、右键单击可弹出右键菜单，通过右键菜单可进行强制、锁定、解除锁定、增加或删除状态表、导入各种数据表格中的元件等操作，可以改变寄存器的显示格式（包括进制、浮点、字符）等。



强制

强制改变元件的状态或数值。

- 1、位元件的强制：在“类型”中选择元件类型，在“元件”中输入元件号，选择 ON 或 OFF 状态，如果 X 元件是外部高速脉冲输入点则可以选择输入脉冲频率值，点击“强制”按钮。



- 2、寄存器元件的强制：在“类型”中选择元件类型，在“元件”中输入元件号，输入要 强制的数值，可选择输入 16 位整数、32 位整数、浮点数或字符，“HEX”选中 表示输入十六进制整数，点击“强制”按钮。



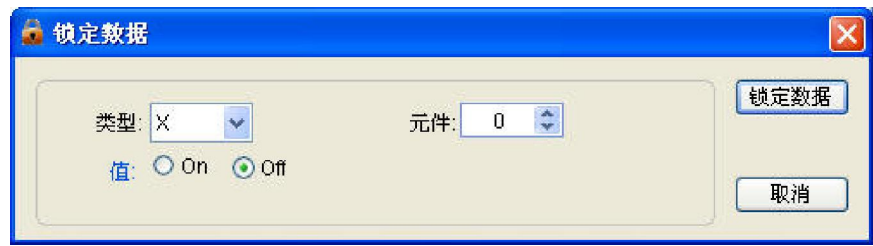
- 3、字符输入如下图。每个寄存器存储两个字符， 当勾选“低字节模式”时，每个寄存器只存储一个字符。



锁定数据

锁定元件的状态或数值，使元件值不发生改变，直至解除锁定。

1、 位元件的锁定：在“类型”中选择元件类型，在“元件”中输入元件号，选择 ON 或 OFF 状态，点击“锁定数据”按钮。



2、 寄存器元件的锁定：在“类型”中选择元件类型，在“元件”中输入元件号，输入要锁定的数值，可选择输入 16 位整数、32 位整数、浮点数或字符，“HEX”选中 表示输入十六进制整数，点击“锁定数据”按钮。



3、 字符输入如下图。每个寄存器存储两个字符， 当勾选“低字节模式”时，每个寄存器只存储一个字符。



强制和锁定数据的区别：

- 1、 强制只是赋值给元件，程序运算输出、外部设备通讯写入等都会改变元件的状态或数值。
- 2、 锁定是将元件固定为某个状态或数值，无论是程序运算输出、外部信号输入、外部设备通讯写入都无法更新，直至解除锁定。
- 3、 在线监控状态下，不能强制外部输入元件的状态或数值（如开关量输入 X 和模拟量输入 AI）。可以锁定外部输入元件的状态或数值。

【注意】

- 1、 请谨慎使用锁定数据功能，锁定数据可能会带来不可预见的后果。
- 2、 如果 PLC 内存在锁定数据，将产生错误代码 142，即“SV3=142 有锁定数据”，以提醒用户注意。
- 3、 如果确实需要运用锁定功能，方便处理某些现场的问题，请在现场问题处理完毕后及时解除锁定数据。

实时曲线

在仿真或在线监控 PLC 状态下，对于寄存器元件（如 V、AI、AQ 等）数值的实时变化可用实时曲线图的方式形象地表现出来，方便用户对某些重要参数进行动态的观察与调试。



- 1、在“元件”栏输入要监控的寄存器元件。
- 2、在“上限值”和“下限值”栏分别输入该寄存器的上限值和下限值。
- 3、将最左边的选择框打勾，表示绘制该元件的实时曲线。
- 4、寄存器元件默认是 16 位整数，如果是 32 位整数将“32Bits”打勾，如果是浮点数将“浮点数”打勾。
- 5、“暂停”选项暂停曲线的描绘。在暂停时，可通过鼠标左键单击曲线上的点来读取相应的数值。

仿真停电

所有控制系统或设备都可能会停电，强壮的控制程序应该充分考虑设备停电的因素，确保程序在停电后重新上电时依然能够正常工作。

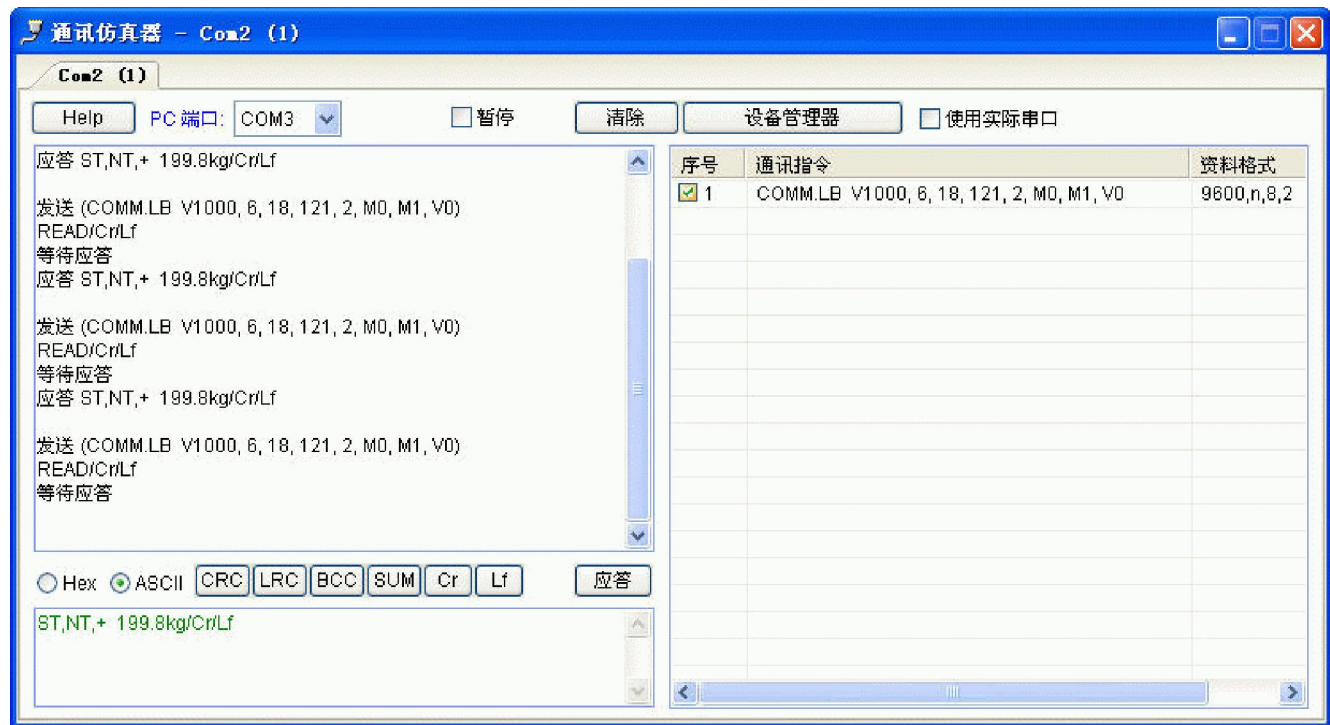
- 1、点击仿真工具栏的 按钮，仿真 PLC 停电，此时程序停止运行。
- 2、点击仿真工具栏的 按钮，仿真 PLC 重新上电，此时程序重新开始执行。
- 3、通过仿真停电/上电过程，可以测试程序是否在经过停电后仍然能够保持正常工作，可以检查出参数是否会丢失等问题。

6.4 通讯仿真器

“通讯仿真器”是专用于调试通讯指令的仿真工具。它可以手动模拟输入从机返回给 PLC 的应答信息，也可以使用计算机的实际串口与从机真实通讯，真实仿真 PLC 执行通讯指令的过程以及对从机返回数据的处理。

启动通讯仿真器

在仿真状态下，点击菜单【调试/通讯仿真器】或单击工具栏 按钮即可启动“通讯仿真器”，默认为“手动输入从机应答信息”模式。



- 1、左上消息框显示正在执行的通讯指令、该指令发出的命令帧数据、从机应答返回的数据等，消息框滚动显示所有通讯指令执行时产生的发送和接收数据。按“清除”按钮可以清除所有信息，勾选“暂停”可以停止更新通讯指令产生的发送和接收数据。
- 2、左下输入框用于手动模拟输入从机返回给 PLC 的应答信息。
- 3、右边列出使用同一个通讯口的所有通讯指令，对正在执行的通讯指令加标识，双击指令会定位到该指令在程序中的位置。
- 4、对每个有通讯指令使用的通讯口都会产生一个单独页面，页标题显示通讯口号和使用该通讯口的通讯指令条数。

手动输入从机应答信息

- 1、当左上消息框显示“等待应答”时，在左下输入框输入从机返回的应答数据帧（数据内容请依据从机的通讯协议规范），按“回车键”即可，应答数据会自动写到通讯指令的输出寄存器。
- 2、“Hex”、“ASCII”用于选择以十六进制或 ASCII 码格式输入和显示数据。
- 3、“CRC”、“LRC”、“BCC”和“SUM”按钮用于对输入数据计算相应的校验码。
- 4、“Cr”输入回车符，“Lf”输入换行符。

使用实际串口与从机通讯

- 1、勾选“使用实际串口”进入使用计算机的实际串口与从机真实通讯模式，通讯指令通过计算机串口给从机发送命令帧数据，并且接收从机返回的应答数据，应答数据会自动写到通讯指令的输出寄存器。实际发送和接收的数据都会在消息框中滚动显示。
- 2、“PC 端口”列表框中列出了计算机可以使用的所有串口，从中选择与从机实际连接的串口。

6.5 高速计数仿真

- 1、高速计数器支持：脉冲/方向、正/反脉冲、A/B 相脉冲输入模式，支持 1、2、4 倍频计数模式。

高速计数器号

计数模式

HSC0

0 - 脉冲/方向 1倍频

脉冲:X0 方向:X1

HSC1

1 - 脉冲/方向 2倍频

脉冲:X2 方向:X3

HSC2

2 - 正转/反转 1倍频

正转脉冲:X4 反转脉冲:X5

HSC3

4 - A相/B相 1倍频

A相脉冲:X6 B相脉冲:X7

0 - 脉冲/方向 1倍频

1 - 脉冲/方向 2倍频

2 - 正转/反转 1倍频

3 - 正转/反转 2倍频

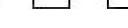
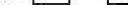
4 - A相/B相 1倍频

5 - A相/B相 2倍频

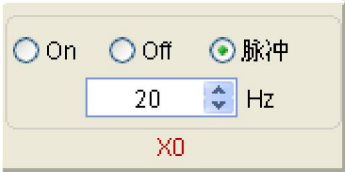
6 - A相/B相 4倍频

【高速计数模式与脉冲波形图】

计数模式		脉冲波形图	
模式	倍频数	加计数	减计数
0 --脉冲/方向	1	脉冲 方向	
1 --脉冲/方向	2	脉冲 方向	

计数模式		脉冲波形图	
模式	倍频数	加计数	减计数
2 --正转/反转	1	正转脉冲  反转脉冲 	
3 --正转/反转	2	正转脉冲  反转脉冲 	
4 -- A 相/B 相	1	A相脉冲  B相脉冲 	
5 -- A 相/B 相	2	A相脉冲  B相脉冲 	
6 -- A 相/B 相	4	A相脉冲  B相脉冲 	

2、高速计数器通道用 HSCx 表示，每个通道使用 2 个高速脉冲输入点。对高速脉冲输入点可以强制该通道输入脉冲的频率值。



3、仿真脉冲输入的方法，以 HSC0 通道为例。

A、脉冲/方向（X0 为脉冲信号，X1 为方向信号）：强制 X0 为脉冲值如 20Hz，X1=OFF，此时为加计数
强制 X0 为脉冲值如 20Hz，X1=ON，此时为减计数

B、正转/反转（X0 为正转脉冲，X1 为反转脉冲）：强制 X0 为脉冲值如 20Hz，X1=OFF，此时为加计数
强制 X0=OFF，X1 为脉冲值如 20Hz，此时为减计数

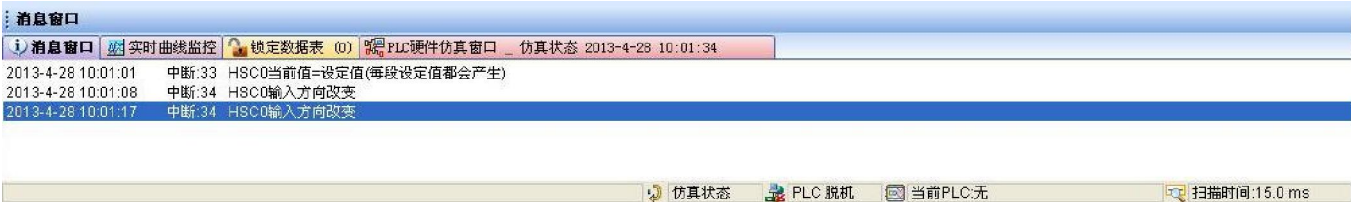
C、A 相/B 相（X0 为 A 相脉冲，X1 为 B 相脉冲）：先强制 X0 为脉冲值如 20Hz，后强制 X1 为脉冲值如 20Hz，此时为加计数（A 相脉冲在前）
先强制 X1 为脉冲值如 20Hz，后强制 X0 为脉冲值如 20Hz，此时为减计数（B 相脉冲在前）

4、高速计数器仿真运行过程中会产生相应的系统中断，如下图。

程序示例：

```
//Network 1 单段比较模式，比较段数设定为4，初始段号为1
```





6.6 脉冲输出仿真

1、高速脉冲输出支持：单脉冲、脉冲/方向、正/反脉冲、A/B 相脉冲、同步脉冲输出，共 5 种输出模式，高速脉冲输出通道用 PLSx 表示，每个通道使用 2 个高速脉冲输出点。

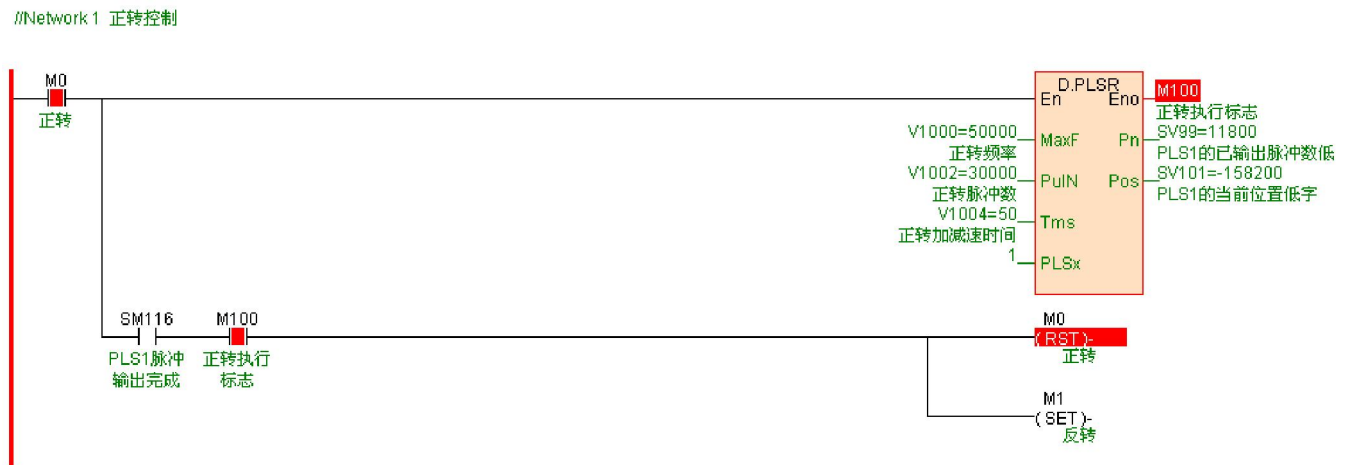


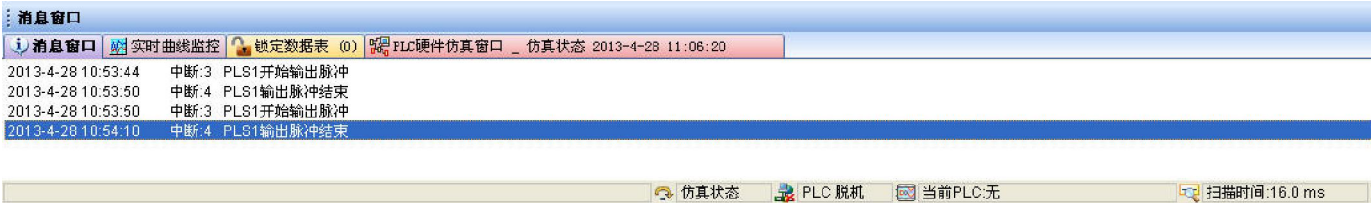
2、仿真脉冲输出的显示，以 PLS1 通道为例。

- A、单脉冲输出（Y2 为脉冲信号）：脉冲输出时 Y2 闪烁；无脉冲输出时 Y2=OFF。
- B、脉冲/方向输出（Y2 为脉冲信号，Y3 为方向信号）：正转脉冲输出时 Y2 闪烁，Y3=OFF。
反转脉冲输出时 Y2 闪烁，Y3=ON。
无脉冲输出时 Y2=OFF，Y3=OFF。
- C、正转/反转 脉冲（Y2 为正转脉冲，Y3 为反转脉冲）：正转脉冲输出时 Y2 闪烁，Y3=OFF。
反转脉冲输出时 Y2=OFF，Y3 闪烁。
无脉冲输出时 Y2=OFF，Y3=OFF。
- D、A 相/B 相脉冲（Y2 为 A 相脉冲，Y3 为 B 相脉冲）：脉冲输出时 Y2、Y3 同时闪烁；无脉冲输出时 Y2=OFF、Y3=OFF。
- E、同步脉冲输出（Y2 为脉冲，Y3 为同步脉冲）：脉冲输出时 Y2、Y3 同时闪烁；无脉冲输出时 Y2=OFF、Y3=OFF。

3、脉冲输出指令仿真运行过程中会产生相应的系统中断，如下图。

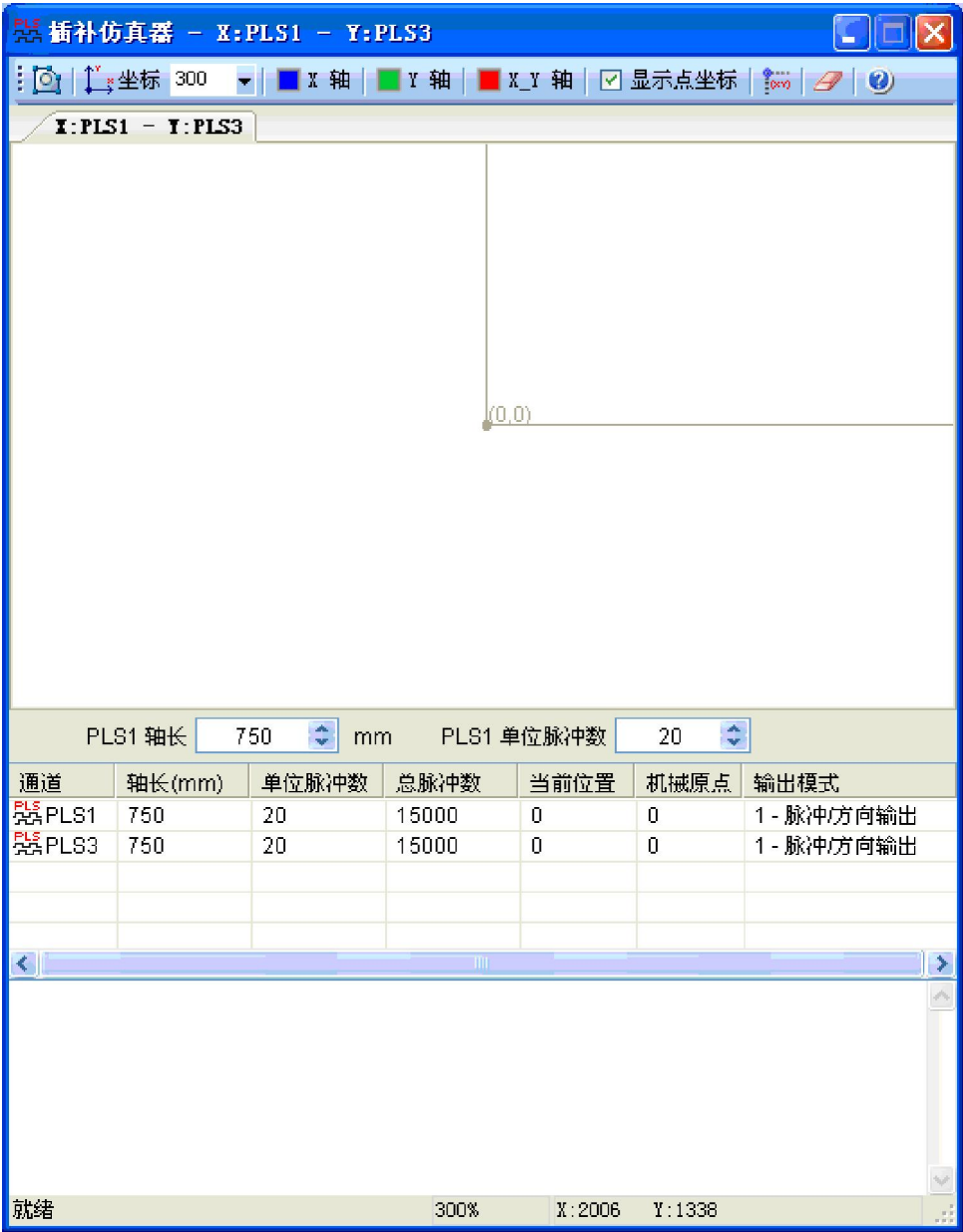
程序示例：





6.7 插补仿真器

- 1、在程序中如有使用插补指令，在仿真状态下，可以启动“插补仿真器”来观察插补指令生成的运动轨迹。
- 2、点击菜单【调试/插补仿真器】或点击工具栏按钮，启动“插补仿真器”。

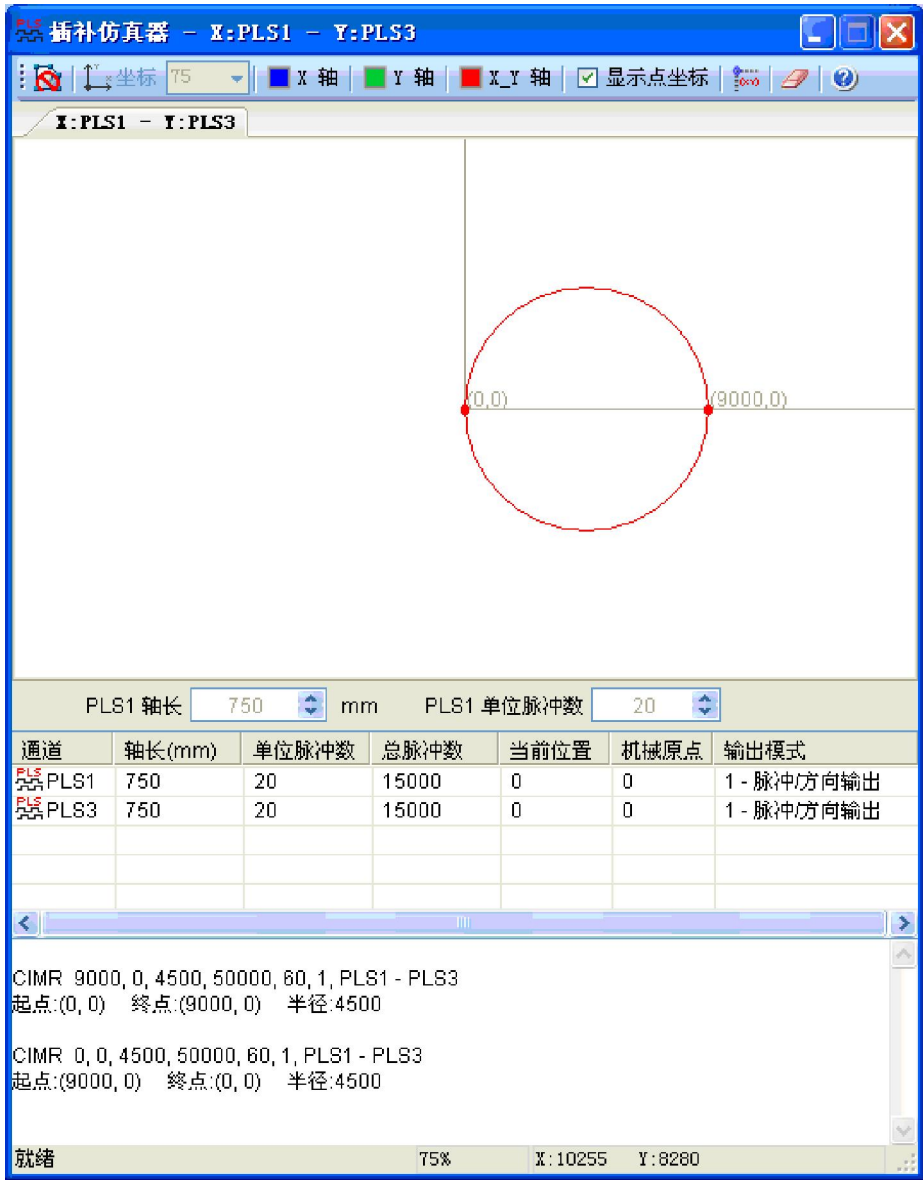


- 3、“插补仿真器”的工具栏。

- A、点击后，插补指令执行时生成的运动轨迹才会被绘制。点击停止绘图。注意：插补指令的执行是由程序控制的，如果指令没有执行就不会生成运动轨迹。
- B、点击坐标，可选择不同的平面坐标。选择与实际运动平台一致的坐标，更方便用户观察运动轨迹。
- C、点击，可选择各坐标轴绘制线条的颜色。

- D、勾选  显示点坐标 则显示运动轨迹点的坐标，不勾选则隐藏点坐标。
- E、点击 ，可将坐标原点拉回到绘图区的中心。
- F、点击 ，可擦除已绘制的运动轨迹。
- 4、插补指令执行时生成的运动轨迹在平面坐标上绘制，不同运动平面会分页面显示。
- 5、在中部列出运动平面各轴对应的脉冲输出通道的参数，显示该通道的当前位置、机械原点位置、输出模式等，可以设定轴长、单位脉冲数。
- 6、下部是消息框，显示正在执行的插补指令及轨迹描述。

圆弧插补程序示例：



6.8 在线调试与脱机仿真的区别

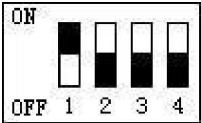
- 1、脱机仿真：不需要实际 PLC，程序在仿真器内运行，对元件状态或数值的修改以及程序运行结果的输出只发生在仿真器中。
- 2、在线调试：编程软件必须与 PLC 联机，程序下载到 PLC 中运行，对元件状态或数值的修改以及程序运行结果的输出发生在联机的 PLC 中，实时响应外部输入的信号（DI 或 AI），输出信号（DO 或 AO）对外部设备产生实际控制效果。

第七章、在线控制 PLC

本章介绍使用编程软件在线控制 PLC。

7.1 PLC站号设置

通过硬件上的 4 位 DIP 拨码开关设置 PLC 站号（也叫站地址、通讯地址）。合法的 PLC 站号为 1~254（0 为广播地址）。4 位 DIP 拨码开关可用于表示十进制数 1~15（二进制的数从 0001 ~ 1111），通过拨码开关修改 PLC 站号的方法如下：



上图为用于设定 PLC 站号的 4 位拨码开关，上面为 ON，下面为 OFF，图中黑色部分表示拨码开关的位置，把其中的一位拨到 ON 时表示该位为 1，拨到 ON 时表示该位为 0，上图中第 1 位为 ON，其它位为 OFF。拨码开关的第 1 位表示二进制的第 0 位（b0），拨码开关的第 4 位表示二进制的第 3 位（b3），由此，如上图所示的拨码开关的位置表示 0001，即 10 进制数为 1，表示 PLC 的站号为 1（出厂默认 为 1）；若把第 1、2 位拨到 ON 位置，而其它位拨到 OFF 位置，则为 0011，即 10 进制数为 3，表示 PLC 的站号为 3。

如果 4 位 DIP 拨码开关表示的硬地址不够使用，可以通过菜单【PLC/设置 PLC 参数】设置 PLC 软地址。如下图：



勾选“使用 PLC 软地址”，输入 PLC 地址数值，点击“确定”即可。使用 PLC 软地址后硬件上的 4 位 DIP 拨码开关失效。

7.2 与 PLC联机

只有在与 PLC 联机成功后，才可以对连接在网络上的 PLC 进行控制，比如上下载程序、在线监控等。

- 1、点击菜单【PLC/PLC 联机】或点击工具栏上按钮，打开“PLC 联机”窗口。



- 2、相关参数设置，一般情况使用默认参数即可无需设置修改。

- PC 端口：选择计算机与 PLC 通讯的串口。列表中可用的 COM 串口数目依不同计算机而不同，系统会自动查找出本计算机所有可用的 COM 串口供用户选择。
- 波特率：选择通讯波特率，系统默认为 19200。
- 资料格式：选择通讯数据格式，系统默认为"N,8,2 RTU"。
- 起始地址、终止地址：如果只是与单台 PLC 联机，使用"单机查找"选项即可；如果是与多台 PLC 联机，则"起始地址"填网络上 PLC 的最小站号，"终止地址"填网络上 PLC 的最大站号。
- 添加到列表、覆盖列表：选择把已建立联机的 PLC 以添加或者覆盖方式加到列表框中。
- 超时时间：设置计算机与 PLC 建立联机通讯的超时时间。RS232 或 RS485 等有线方式联机使用默认的 200ms 即可，无线方式联机（如通过 GPRS）应根据无线通讯延时情况设置一个比较大的数值，一般建议在 5000ms 左如果右。
- 单机查找： 如果只是与单台 PLC 联机，勾起"单机查找"选项；如果是与多台 PLC 联机，则必须取消"单机查找"选项并且设置"起始地址"和"终止地址"。

3、联机操作

- 联机：直接点"联机"按钮即可，找到的 PLC（联机成功）会在列表框列出。
- 查找：如果忘记了波特率等通讯参数，这时可以点"查找"按钮进行查找，查找功能会尝试以所有波特率、所有资料格式来与 PLC 通讯，所以需要花费比较长时间等待，找到的 PLC（联机成功）会在列表框列出。

4、点击"退出"，关闭"PLC 联机"窗口。

5、如果联机不成功，有可能是以下情况所造成，仔细检查即可排除问题，如果无法解决请联系海为公司技术支持 。

- A、选择的 PC 端口不对 。
- B、选择的通讯参数与目标 PLC 的通讯参数不对 。
- C、目标 PLC 未上电 。
- D、通讯电缆连接有误或通讯接头未连接好 。
- E、使用"USB 转 RS232 串口数据线"时未正确安装它的驱动 。

7.3 在线 PLC窗口

在"PLC 联机"窗口如果有找到 PLC（联机成功），退出"PLC 联机"窗口后会自动打开"在线 PLC 窗口"，如下图。



- 1、窗口上部为在线的 PLC 列表，列出已经联机的所有 PLC。可通过双击将列表中任意一台 PLC 选为当前 PLC，所有对 PLC 的在线操作如：上下载程序、固件升级、设置 PLC 口令等都只是对当前 PLC 的操作，不影响其他 PLC。
- 2、窗口下部为当前 PLC 的详细状态，如：PLC 运行状态、程序大小、版本、有无口令、串口参数、主机及所有扩展模块的型号版本等信息。

7.4 下载程序

将当前程序项目（硬件配置、程序、注释）下载到目标 PLC 中。下载前系统会自动对当前程序项目进行编译，若程序编译有错误，则列出所有错误，用户将程序修改至无任何错误方可下载。警告信息只是提示用户注意，如果无错误仅有警告信息时，可进行下载。

- 1、点击菜单【PLC/PLC 程序下载】或点击工具栏上的按钮，打开"PLC 程序下载"窗口。



- 2、可以选择下载内容：硬件配置、程序、注释、初始寄存器值表
- 3、禁止上载：如勾选此选项，下载的程序无法被上载。该方式安全保护了用户 程序的知识产权。
- 4、带清除功能下载：如勾选此选项，在下载时会自动对 PLC 进行初始化，包括清除程序、停电保持区等，然后再进行下载程序。
- 5、不停机下载：如勾选此选项，在下载时程序不会停止运行，即在线修改程序。**请谨慎使用此项功能，未经调试的程序可能会带来不可预见的后果。**
- 6、硬件匹配：列表详细列出"程序项目配置"与"目标 PLC 配置"的匹配情况，硬件配置信息应该相同。如果有硬件配置不同，程序下载到目标 PLC 后将产生错误代码 140，即"SV3=140 硬件配置不匹配"。用户需根据列表提示将不同的硬件配置修改为相同，再重新下载以确保 PLC 正常工作。
- 7、修改硬件配置信息：如果是"CPU 模块"不同，请通过菜单【文件/程序项目属性】修改 CPU 模块型号；如果是"扩展模块数量或者型号"不同，请通过菜单【查看/PLC 硬件配置】打开"PLC 硬件配置"窗口进行增减或修改模块。
- 8、点击"下载"按钮将程序下载到 PLC 中。

7.5 上载程序

将目标 PLC 内的程序上载到计算机中。如果该程序在下载时选择了“禁止上载”，那么程序无法上载。

- 1、点击菜单【PLC/PLC 程序上载】或点击工具栏上的按钮，打开"PLC 程序上载"窗口。



2、点击“上传”按钮，将目标 PLC 内的程序上传到计算机。

7.6 生成 PLC执行文件

将 PLC 程序项目 生成一个 PLC 的执行文件，该文件可以单独发布和下载到 PLC 执行，但是无法打开编辑。

1、点击菜单【文件/生成 PLC 执行文件】，打开“生成 PLC 执行文件”窗口。



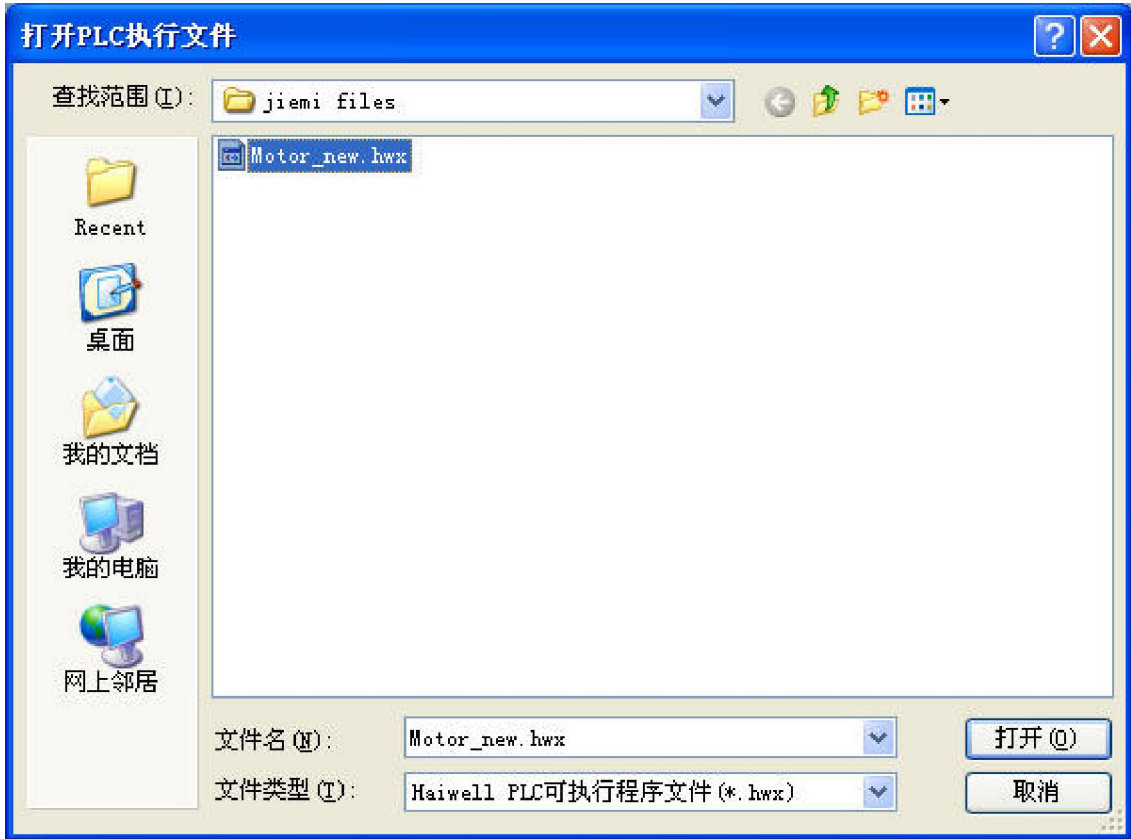
2、“PLC 口令”、“PLC 确认口令”： 如果 PLC 的硬件设置有保护口令，那么输入 PLC 的口令，这样生成的 PLC 执行文件才可以被下载到 PLC 中。

3、点击“生成”按钮将 生成 PLC 执行文件。

7.7 下载 PLC执行文件

将生成的 PLC 执行文件下载到 PLC 中，该程序无法上传。

1、点击菜单【PLC/下载 PLC 执行文件】或点击工具栏上的按钮，打开“PLC 程序上传”窗口。



2、选择要下载的 PLC 执行文件，打开“下载 PLC 执行文件”窗口。



3、点击“下载”按钮，将 PLC 执行文件下载到 PLC 中，如果硬件不匹配则无法下载，如果 PLC 设定了保护口令则当口令不正确时无法下载。

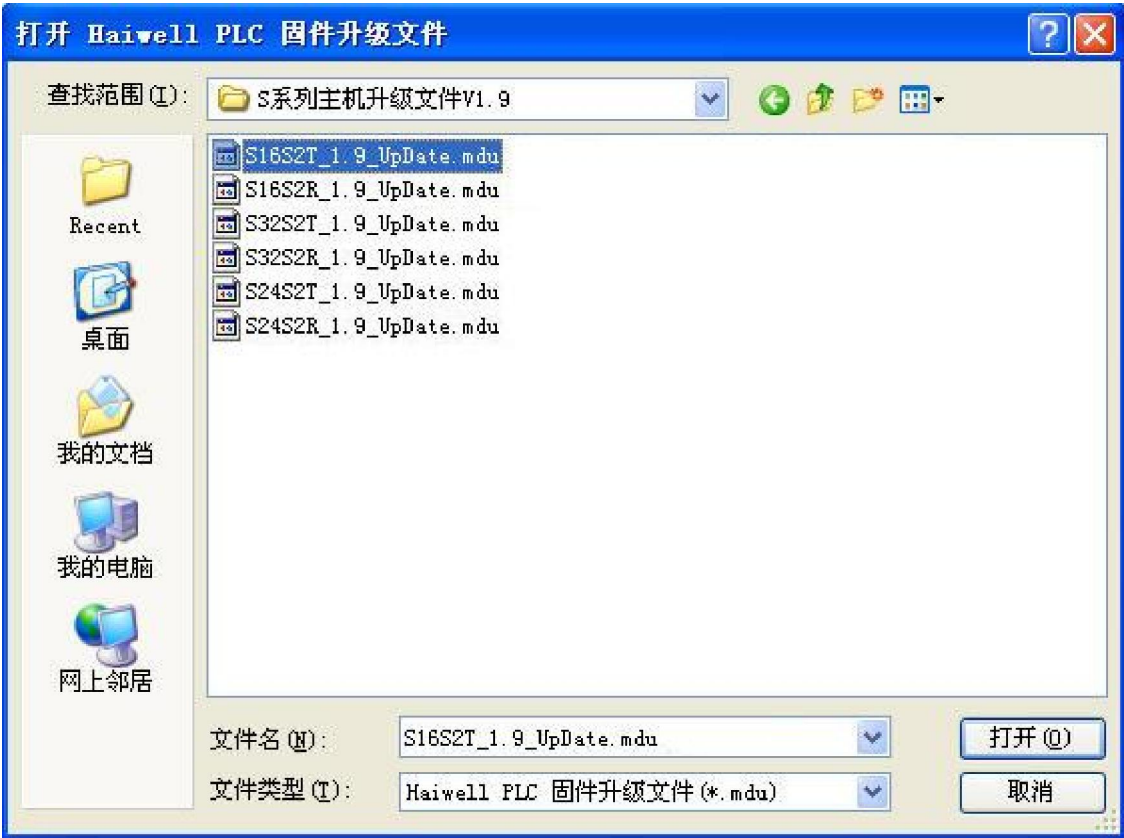
7.8 PLC固件升级

更新 PLC 的 CPU 主机或扩展模块的固件程序。使 CPU 主机或扩展模块支持新的功能。

1、点击菜单【PLC/PLC 固件升级】，打开“PLC 固件升级”窗口



2、点击“打开”按钮，选择与被升级模块相对应的升级文件。



3、在“PLC 固件升级”窗口，上部表格中列出了固件升级文件的版本、模块型号等信息，下部列出了目标 PLC 的名称、CPU 模块和扩展模块等硬件配置信息，在“选择升级模块”框内选择要升级的模块序号，模块序号 0 表示 CPU 模块，扩展模块从 1 开始按照从左到右顺序排列。

4、点击“升级”按钮，即可对 PLC 的 CPU 主机或扩展模块的固件程序升级。

注意：如果升级过程中由于断电或者其他原因造成升级中断，需重新执行 PLC 固件升级直至升级成功。

7.9 启动、停止 PLC

通过编程软件来控制 PLC 的启动或停止，使其处于 RUN/STOP 状态，实现远程控制 PLC 的启动或停止。

- 1、点击菜单【PLC/启动 PLC 运行】即可启动 PLC 运行。
- 2、点击菜单【PLC/停止 PLC 运行】即可停止 PLC 运行。

【注意】

- 1、在 PLC 的 CPU 主机设有一个启停开关，当开关拨到“STOP”位置时，PLC 处于停止状态；当开关拨到“RUN”位置时，PLC 处于运行状态。
- 2、启停开关与编程软件的“启动或停止 PLC 运行”功能的关系：以后停开关优先，在 PLC 断电后重新上电时，若启停开关处于 STOP 位置，则此时 PLC 将处于停止状态；若启停开关处于 RUN 位置，则此时 PLC 将自动转入运行状态。仅若 PLC 处于上电状态时，编程软件才可进行“启动 PLC 运行”或“停止 PLC 运行”控制，不管启停开关处于什么位置。
- 3、若启停开关处于 STOP 位置，在下载程序或固件升级完成后，PLC 将处于 STOP 状态；若启停开关处于 RUN 位置，在下载程序或模块固件升级完成后，PLC 将自动处于 RUN 状态。
- 4、用户在用编程软件进行“启动 PLC 运行”或“停止 PLC 运行”控制时，要确保现场安全，以避免对人身和设备损伤。

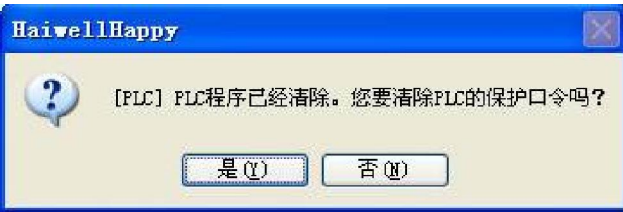
7.10 清除 PLC程序

初始化目标 PLC，清除包括程序、硬件配置、停电保持区等。

- 1、点击菜单【PLC/清除 PLC 程序】

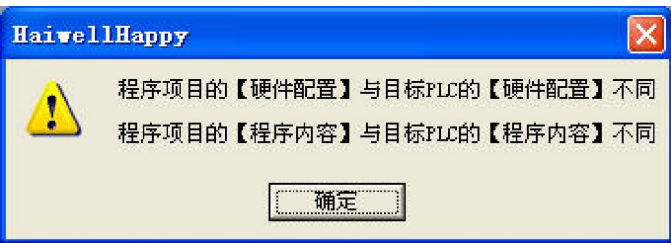


- 2、确认是否清除 PLC 程序，如果选择“是”，将对 PLC 进行初始化。选择“否”，不对 PLC 进行任何操作。
- 3、若 PLC 设置了口令保护，将提示用户输入的口令，只有口令正确时，才可对目标 PLC 进行程序的清除程序工作。并在清除 PLC 程序后提示用户是否清除 PLC 口令，如果想清除 PLC 口令，选择是。否则 PLC 口令将保留。



7.11 程序比较

将当前程序项目与目标 PLC 进行比较。 分别列出“程序内容”和“硬件配置”是否相同。



7.12 设置 PLC 口令

设置目标 PLC 的口令（密码），以保护 PLC 内的程序及硬件配置等信息。设置口令后，包括程序上载、程序下载、清除程序等都需要输入正确的口令才能执行。

- 1、点击菜单【PLC/设置 PLC 口令】，打开“设置 PLC 口令”窗口。



- 2、在口令和确定口令栏里，输入相同的口令，点击“确认”完成口令设置。
- 3、如果 PLC 已经设置了口令，用户需要修改或者取消口令，提供点击菜单【PLC/设置 PLC 口令】，打开“口令验证”窗口，输入“原来口令”，口令验证正确后会打开“设置 PLC 口令”窗口，重新设置新口令即可，口令为空表示不设口令（即清除口令）。



7.13 设置 PLC 时钟

设置目标 PLC 的实时时钟。

- 1、点击菜单【PLC/设置 PLC 时钟】，打开“设置 PLC 时钟”窗口。



- 2、点击“确定”按钮即可将计算机的当前日期和时间设置为 PLC 的实时时钟。
- 3、如果需要调整计算机的当前时钟，点击《时钟》按钮，即可修改计算机的日期和时间。



7.14 设置 PLC通讯参数

设置目标 PLC 的各个通讯口参数。

1、点击菜单【PLC/设置 PLC 通讯参数】，打开“设置 PLC 通讯参数”窗口。



2、PLC 端口：选择被设置的 PLC 通讯口。CPU 主机带 2 个通讯口（RS232 口是 COM1，RS485 口是 COM2）。扩展通讯口按实际顺序分别为 COM3、COM4、COM5。

3、波特率：选择通讯波特率。

4、资料格式：选择通讯资料格式。

5、超时时间：设定通讯超时时间。

6、点击“确定”按钮，即可设置 PLC 通讯口的通讯参数。

注意：所有通讯口默认参数为 19200 N,8,2 RTU，建议使用默认参数。

7.15 设置 PLC参数

设置目标 PLC 的名称、程序扫描超时时间和 PLC 软地址。

- 1、点击菜单【PLC/设置 PLC 参数】，打开“设置 PLC 参数”窗口。



- 2、设置 PLC 名称可以方便对网络上多个 PLC 加以区分，PLC 名称支持最多 6 个汉字或 12 个英文字母、数字。
- 3、PLC 扫描超时时间 即看门狗时间，单位为毫秒(ms)。当 PLC 程序运行的实际扫描时间大于该值时，将产生错误代码 141，即“SV3=141 扫描超时，看门狗动作”。PLC 停止运行。
- 4、勾选“使用 PLC 软地址”，输入 PLC 地址数值，点击“确定”即可。使用 PLC 软地址后硬件上的 4 位 DIP 拨码开关失效。

第八章、联网通讯功能

本章介绍 Haiwell（海为）PLC 的联网通讯功能。

8.1 特点

- 1、**支持多种通讯协议**：内置 Modbus RTU/ASCII 协议、自由通讯协议以及海为公司的 Haiwellbus 高速通讯协议。
- 2、**支持 5 个通讯口**：所有型号主机均自带 2 个通讯口（RS-232 + RS-485），可扩展至 5 个通讯口，每个通讯口互相独立可同时并行工作，所有通讯端口功能相同都可用于编程、上下载程序、监控、联网，所有通讯口都支持主或者从通讯模式。
- 3、**组网灵活**：支持 1:N、N:1、N:N 联网方式，支持各种人机界面和组态软件，可与任何带通讯功能的第三方设备（如变频器、仪表、条码扫描器等）联网。
- 4、**极为便利的通讯指令**：使您无论使用何种通讯协议都只需一条通讯指令便可完成复杂的通讯功能，编程简单，多条通讯指令可以同时得电并行执行，无须再为通讯端口冲突、发送接收控制、通讯中断处理等问题烦恼，可以在程序中混合使用各种协议轻松完成您所需的通讯功能。
- 5、**所有通讯指令带 Out 输出**：可以正确指示通讯指令执行的成功与失败，可以明确指示与哪一个从站通讯失败，方便现场调试和故障判断。
- 6、**可方便地与第三方设备联网通讯**：支持不同厂商、不同协议、不同波特率和通讯格式的设备挂在同一个 485 网络。
- 7、**扩展模块带通讯口可作为远程 IO 模块**：Haiwell（海为）PLC 的扩展模块带有一个 RS485 通讯口，既支持并行总线（用扩展总线挂到 PLC 主机的并行接口）也支持串行总线（用 RS485 通讯口与 PLC 主机的通讯口联网，主机使用通讯指令控制远程模块），当用串行总线进行扩展时（即远程 IO 模块），不受系统点数的扩展限制，可分布式安装。这一点对于有大量分散的开关量或者模拟量信号（温度、湿度、压差、风量、流量、风机转速、阀门开度等）需要进行采集和监控的系统极为重要，轻松实现分布式安装控制且可无限制 点数扩展，极大地提高了控制系统的配置灵活度及日后的控制扩展能力，减少了各种信号的布线量，同时也减小了

因模拟量信号线过长带来的干扰问题，节省工程投资成本。

8、上位机（HMI 人机界面、组态软件等）使用 Modbus 协议访问：Haiwell（海为）PLC 作为从机不需要写任何通讯程序，各元件对应的 Modbus 通讯地址码见“[通讯地址代码表](#)”

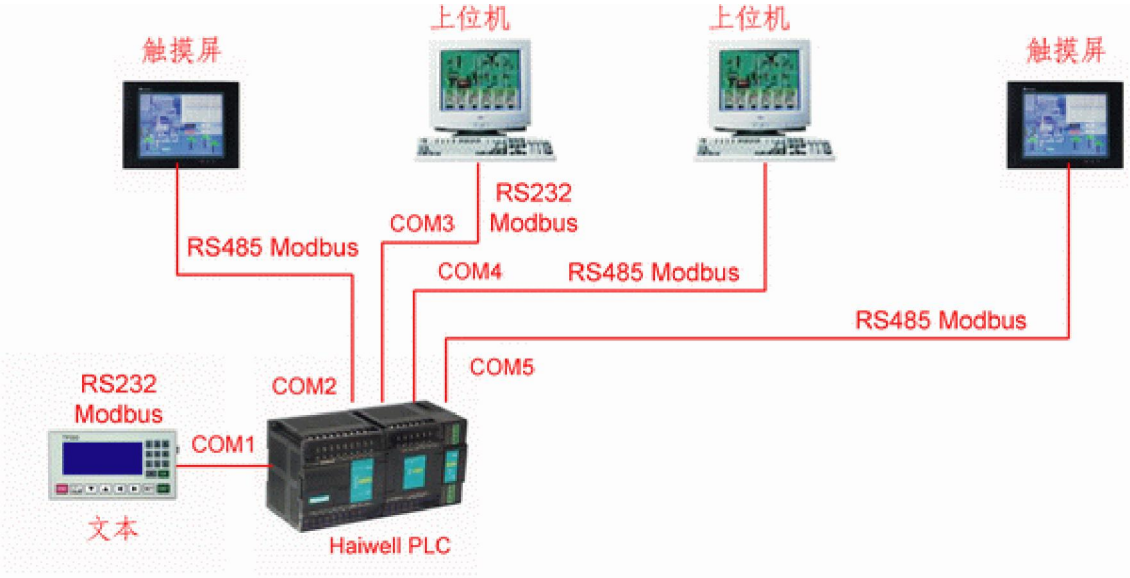
8.2 联网示意图

Haiwell（海为）PLC 组网方便，支持 1：N、N：1、N：N 联网方式，联网示意图如下：

一、1：N 联网示意图：



二、N：1 联网示意图：



三、N:N 多级联网示意图：



8.3 Modbus通讯

Modbus 是由 Modicon（现为施耐德电气公司的一个品牌）在 1979 年发明的，是全球第一个真正用于工业现场的总线协议。为了更好地普及和推动 Modbus 在基于以太网上的分布式应用，目前施耐德公司已将 Modbus 协议的所有权移交给 IDA（Interface for Distributed Automation，分布式自动化接口）组织，并成立了 Modbus-IDA 组织，为 Modbus 今后的发展奠定了基础。在中国，Modbus 已经成为国家标准 GB/T19582-2008。Modbus 协议是应用于电子控制器上的一种通用语言。通过此协议，控制器相互之间、控制器经由网络（例如以太网）和其它设备之间可以通信。它已经成为一通用工业标准。有了它，不同厂商生产的控制设备可以连成工业网络，进行集中监控。

Haiwell（海为）PLC 内置 Modbus RTU/ASCII 协议，当第三方设备（如变频器、伺服控制器、仪表等）支持 Modbus 协议时，可以使用 Modbus 协议与它们通讯，用 **MODR**（Modbus 读）指令读取从机设备数据和用 **MODW**（Modbus 写）指令把数据写入从机设备。仅需此 2 条指令，无需编写任何校验程序，它根据通讯模式（RTU 或 ASCII）自动对返回数据进行校验（CRC 或 LRC）。

【示例】

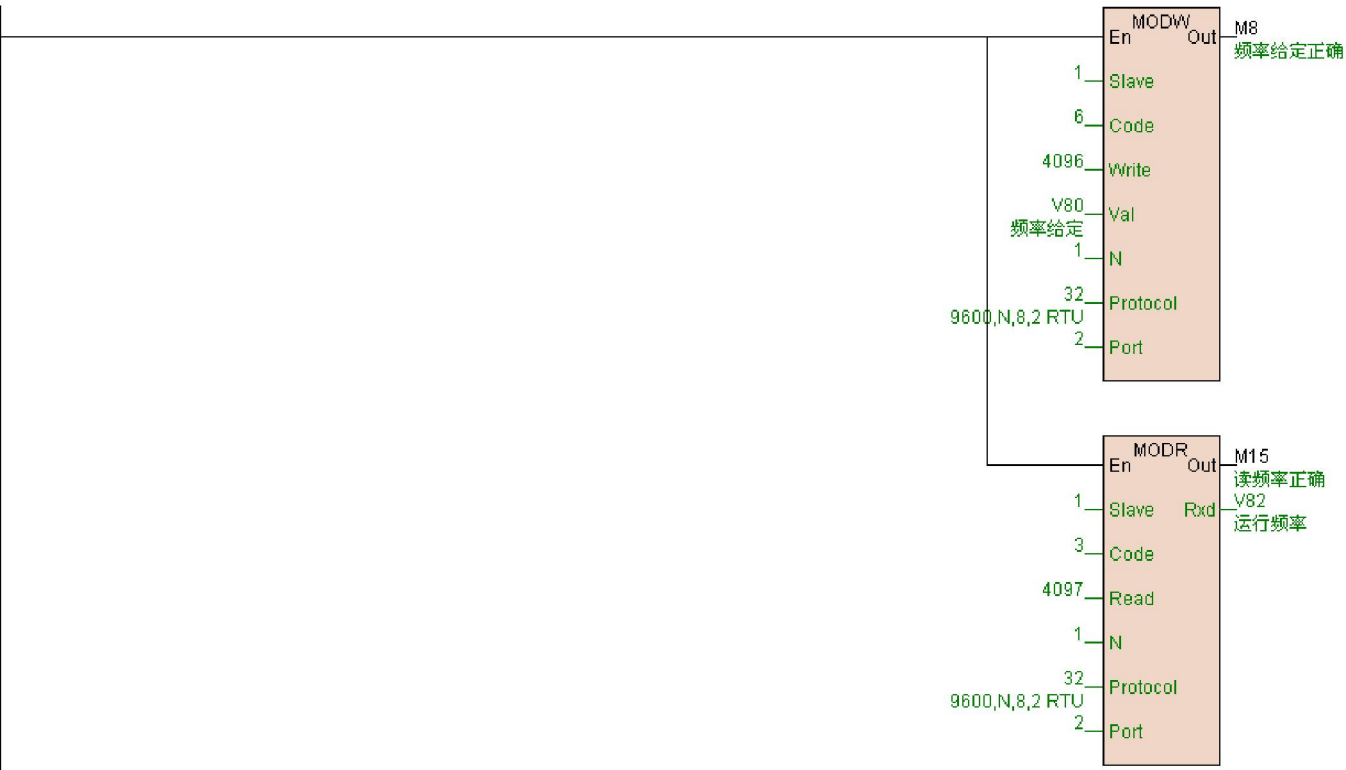
PLC 通过 485 通讯口与汇川 MD320 系列变频器联网，用通讯方式设定变频器的给定频率和读取变频器的运行频率。

【示例程序】

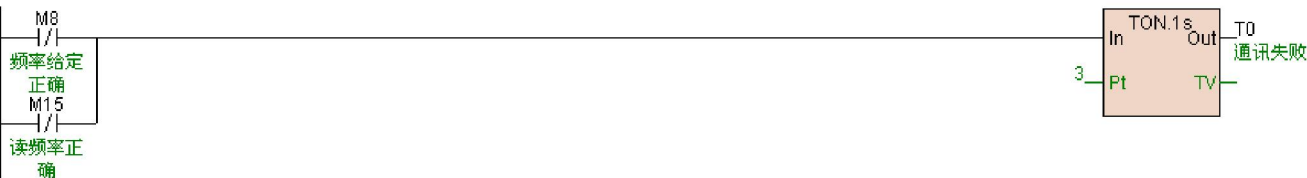
- 1、根据汇川变频器的通讯协议（请参考汇川变频器说明书的通讯部分），给定频率的 Modbus 地址为 4096，MODW 指令把 V80 的值实时写到变频器。

- 2、运行频率的 Modbus 地址为 4097，MODR 指令把变频器的当前频率读出存放到 V82 中。
- 3、MODW、MODR 指令从母排取电一直执行，如果连续 3 秒通讯不成功那么产生通讯失败报警。

//Network 1 与汇川变频器通信：站号为1，通信格式为9600,N,8,2 RTU。频率给定放于V80，读出的运行频率放在V82



//Network 2 如果连续3秒通讯不成功那么产生通讯失败报警



8.4 Haiwellbus通讯

Haiwellbus 是海为公司定义的一种高效高速通讯协议，具有离散或连续的、混合数据（位元件、寄存器元件）传输能力，有很高的通讯速度和通讯效率，一次通讯最大可完成 30 笔数据的交互，Haiwellbus 协议的通讯指令有二条，分别是 **HWRD**（Haiwellbus 读指令，须定义“**Haiwellbus 读通讯表**”）和 **HWWR**（Haiwellbus 写指令，须定义“**Haiwellbus 写通讯表**”）。当单台 PLC 的控制能力不足或控制设备之间距离比较分散时，往往需要采用多台 PLC 进行分站控制，各 PLC 分站之间根据需要进行数据交互。

Haiwell（海为）PLC 具有强大的联网功能，站与站之间既可以采用 Haiwellbus 协议进行数据交互，也可以采用标准 Modbus 协议进行数据交互，无论采用何种协议，从站 PLC 都不需要编写任何程序，只需要在主站 PLC 中编写读或写命令。图示如下：



【示例】

2 台 PLC 通过 485 通讯口联网，它们之间使用 Haiwellbus 协议交换数据。

【示例程序】

1、只要 1#PLC 写通讯程序，2#PLC 不需要写任何通讯程序。如果连续 3 秒通讯不成功那么产生通讯失败报警。

2、定义 Haiwellbus 读通讯表“读 2#PLC 数据”如下：

序号	从机读出数据元件	主机写入数据元件
1	X0	M10
2	X3	M11
3	V11	V80
4	V12	V81
5	AI0	V20
6	AI1	V21

3、定义 Haiwellbus 写通讯表“写 2#PLC 数据”如下：

序号	主机读出数据元件	从机写入数据元件
1	X0	M100
2	X1	M101
3	V0	V100
4	V50	V102
5	Y4	M0
6	Y5	Y0
7	V60	V200
8	V61	V201

4、HWRD、HWWR 指令从母排取电一直执行，根据上面定义的 Haiwellbus 读/写通讯表，自动与 2#PLC 完成数据交换。

//Network 1 与 2#PLC 交换数据



//Network 2 如果连续3秒通讯不成功那么产生通讯失败报警



8.5 自由通讯

Haiwell（海为）PLC 除了支持标准 Modbus 协议和 Haiwellbus 协议外，还支持灵活的自由协议通讯，无论从机设备使用什么协议，只要知道其协议文本就可以与其通讯。

自由通讯的核心在于看懂从机的通讯协议文本，根据从机的协议要求发送正确的内容（字节串）给从机，并对从机返回的数据依协议进行校验、分解、解释出正确结果，具体参见 [COMM](#)、[RCV](#)、[XMT](#) 指令。

下面以 2 个实际通讯例子详细讲解如何使用第三方通讯协议实现自由通讯。

【示例一】

PLC 与符合 DLT-645 标准的多功能电能表联网通讯，读取它的电表常数(有功)和当前有功总电能数值。

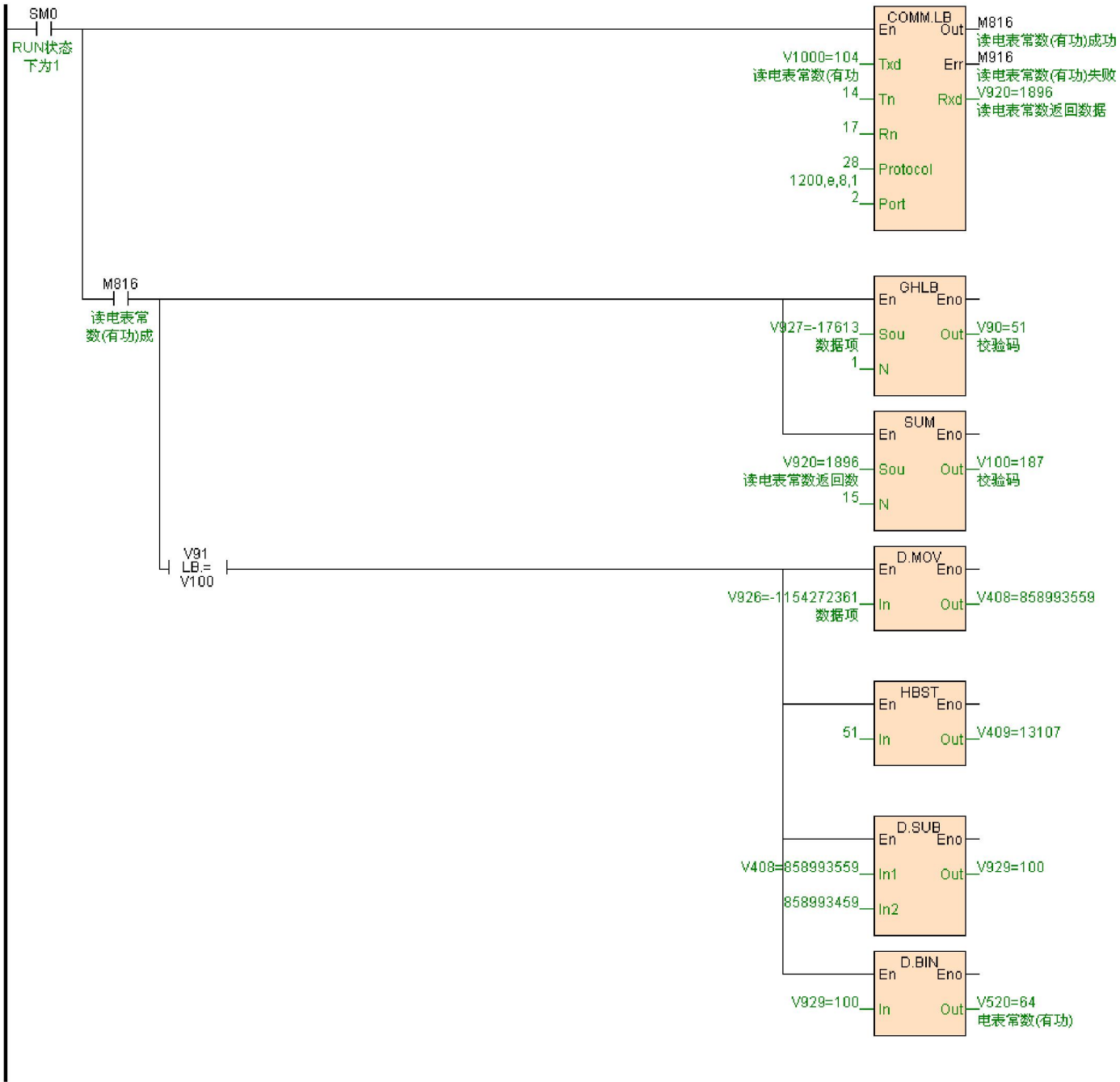
【示例一程序】

1、按照 DLT-645 协议规约， 读取电表常数(有功)发送命令帧：68 07 05 06 00 00 00 68 01 02 63 F3 3B 16 共 14 个字节，电表常数(有功)的标识编码：0xC030 + 0x3333 = 0xF363，命令存放在“读电表常数(有功)”表。

序号	元件初始值	说明
1	V1000=0x0068	起始符
2	V1001=0x0007	地址
3	V1002=0x0005	地址
4	V1003=0x0006	地址
5	V1004=0x0000	地址
6	V1005=0x0000	地址
7	V1006=0x0000	地址
8	V1007=0x0068	起始符
9	V1008=0x0001	控制码
10	V1009=0x0002	数据长度
11	V1010=0x0063	标识编码
12	V1011=0x00F3	标识编码
13	V1012=0x003B	校验码
14	V1013=0x0016	结束符

程序使用 COMM.LB 指令（低字节模式）发送 V1000~V1013 元件的低字节，接收的 17 个字节返回数据放在 V920~V928。 假如返回的 17 个字节数据为：68 07 05 06 00 00 00 68 81 05 63 F3 97 33 33 BB 16，那么电表常数(有功)V520=64。

//Network 1 读电能表电常数(有功)数据，通讯成功则M816=ON，则计算返回数据的校验码到V100，与返回校验码比较，正确则解析出电表常数数据到V520



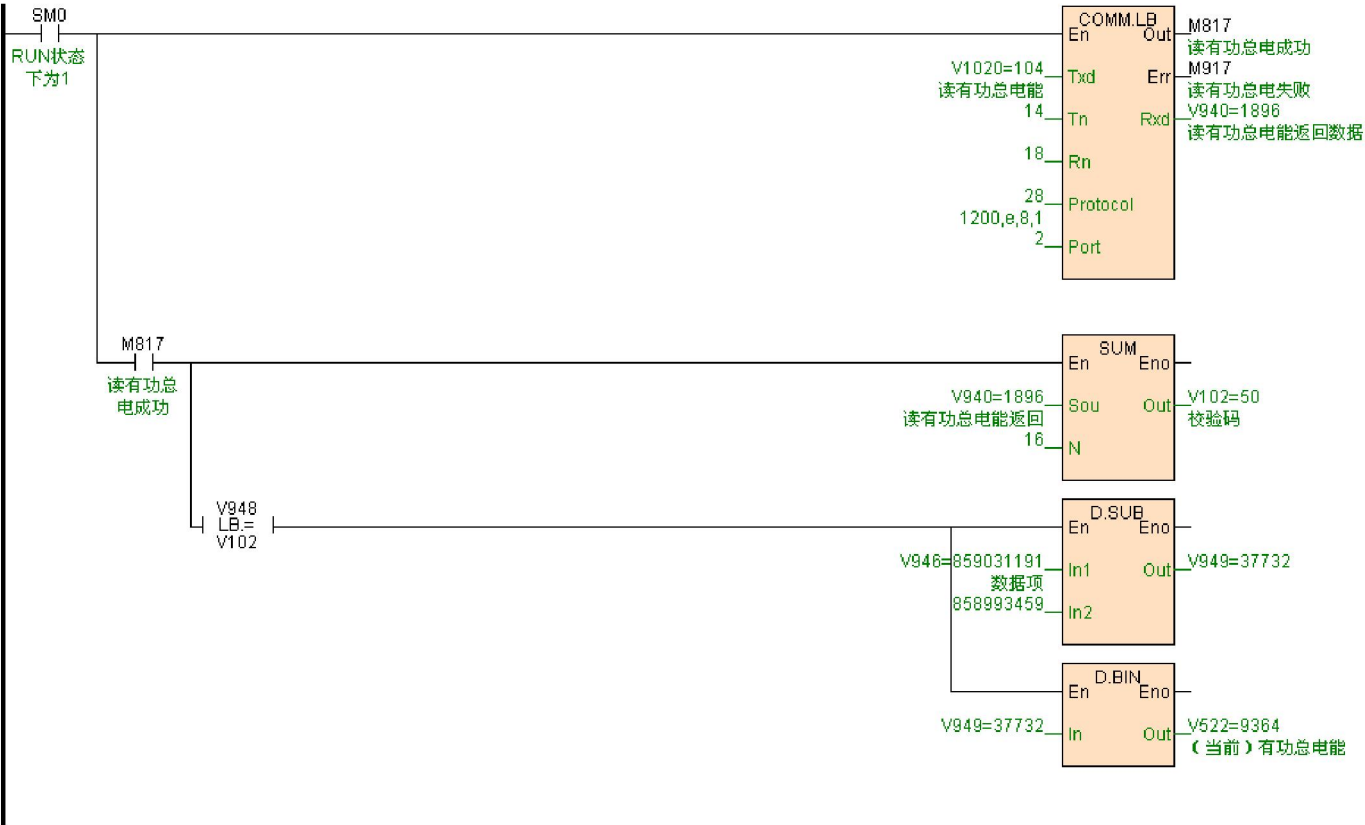
2、按照 DLT-645 协议规约，读取当前有功总电能发送命令帧：68 07 05 06 00 00 00 68 01 02 43 C3 EB 16 共 14 个字节，当前有功总电能的标识编码：0x9010 + 0x3333 = 0xC343，存放在“读当前有功总电能”表。

序号	元件初始值	说明
1	V1020=0x0068	起始符
2	V1021=0x0007	地址
3	V1022=0x0005	地址
4	V1023=0x0006	地址
5	V1024=0x0000	地址
6	V1025=0x0000	地址
7	V1026=0x0000	地址
8	V1027=0x0068	起始符
9	V1028=0x0001	控制码
10	V1029=0x0002	数据长度
11	V1030=0x0043	标识编码
12	V1031=0x00C3	标识编码

13	V1032=0x00EB	校验码
14	V1033=0x0016	结束符

程序使用 COMM.LB 指令（低字节模式）发送 V1020~V1033 元件的低字节，接收的 18 个字节返回数据放在 V940~V948。假如返回的 18 个字节数据为：68 07 05 06 00 00 00 68 81 06 43 C3 97 C6 33 33 32 16，那么当前有功总电能 V522=9364。

//Network 2 读电能表《当前》有功总电能数据，通讯成功则M817=ON，则计算返回数据的校验码到V102，与返回校验码比较，正确则解析出有功总电能数据到V522



【示例二】

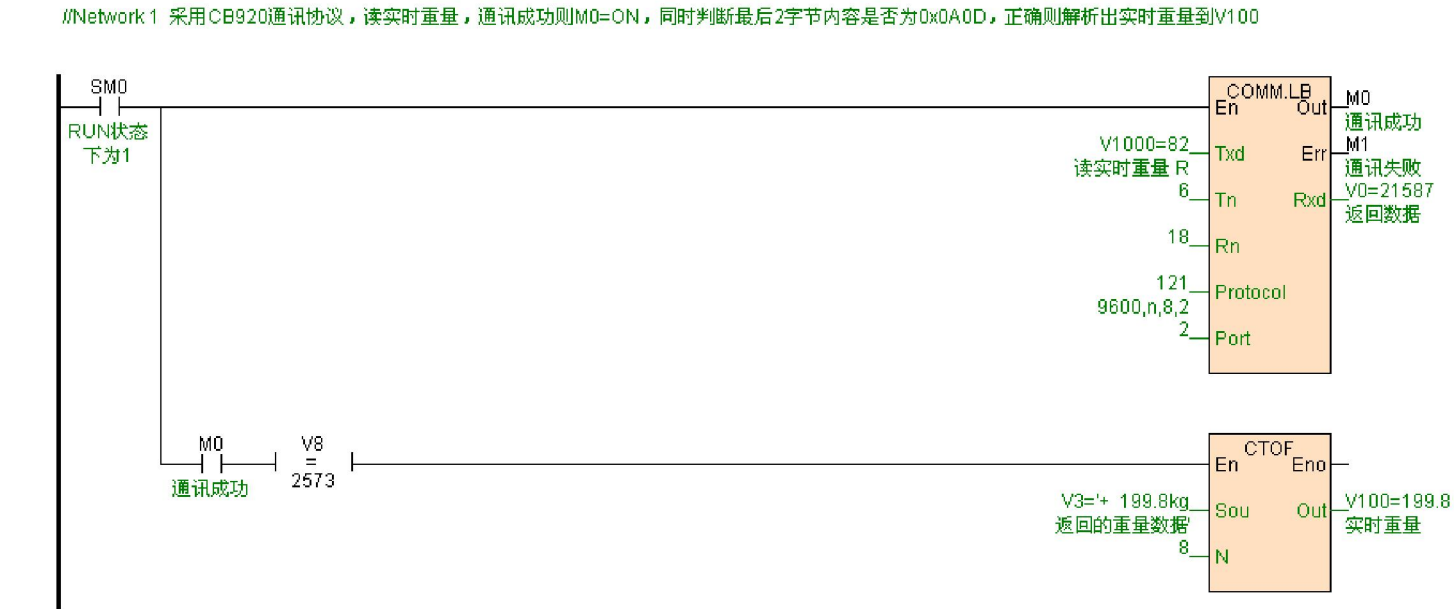
PLC 与托利多 T600 称重终端通讯，使用它的 CB920 通讯协议读取实时重量值。

【示例二程序】

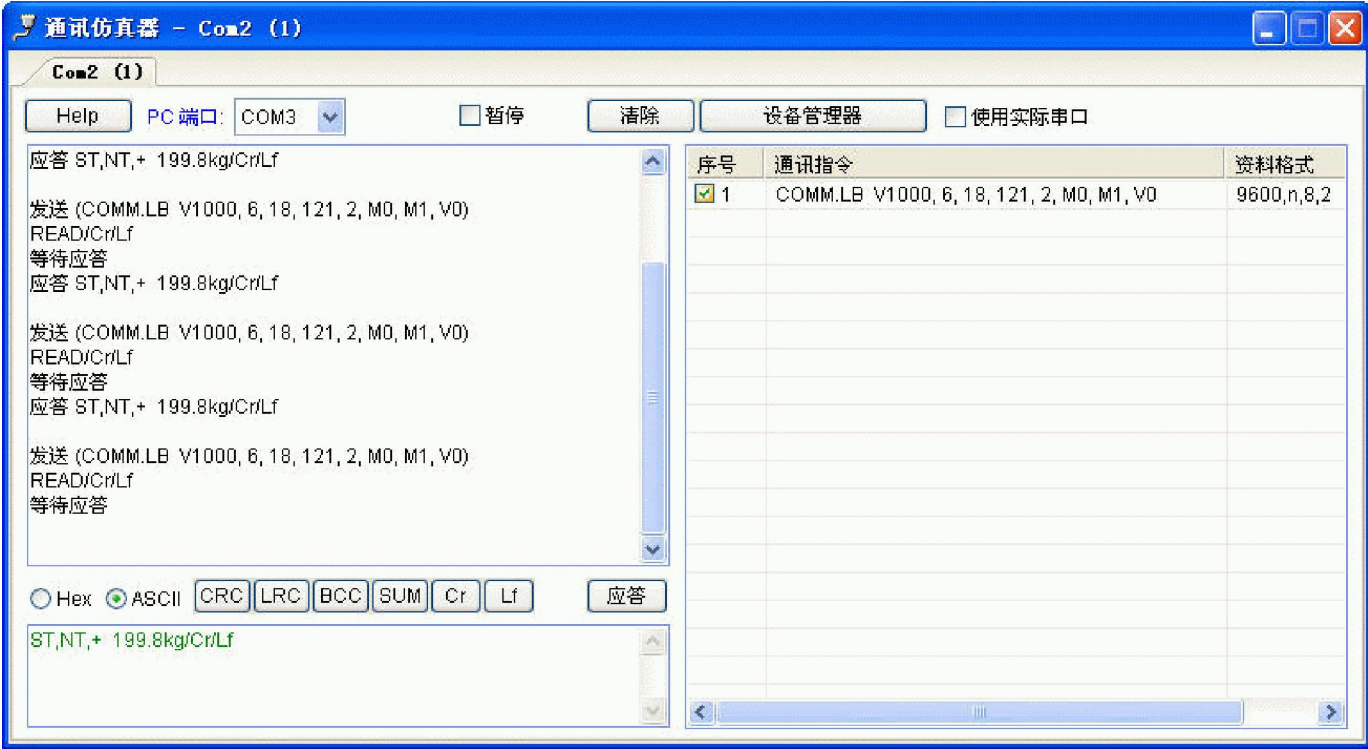
1、按照 CB920 通讯协议， 读实时重量的 ASCII 码命令：READ/CR/LF，即 6 字节命令帧：52 45 41 44 0D 0A，命令存放在“读实时重量”表。

序号	元件初始值	ASCII 码
1	V1000=0x0052	R
2	V1001=0x0045	E
3	V1002=0x0041	A
4	V1003=0x0044	D
5	V1004=0x000D	/CR
6	V1005=0x000A	/LF

程序使用 COMM.LB 指令（低字节模式）发送 V1000~V1006 元件的低字节，接收的 18 个字节返回数据放在 V0~V8。假如返回的 18 个字节数据为：53 54 2C 4E 54 2C 2B 20 20 31 39 39 2E 38 6B 67 0D 0A（返回 ASCII 码：ST,NT,+ 199.8kg/Cr/Lf），那么实时重量 V0=199.8。



2、通讯仿真器。



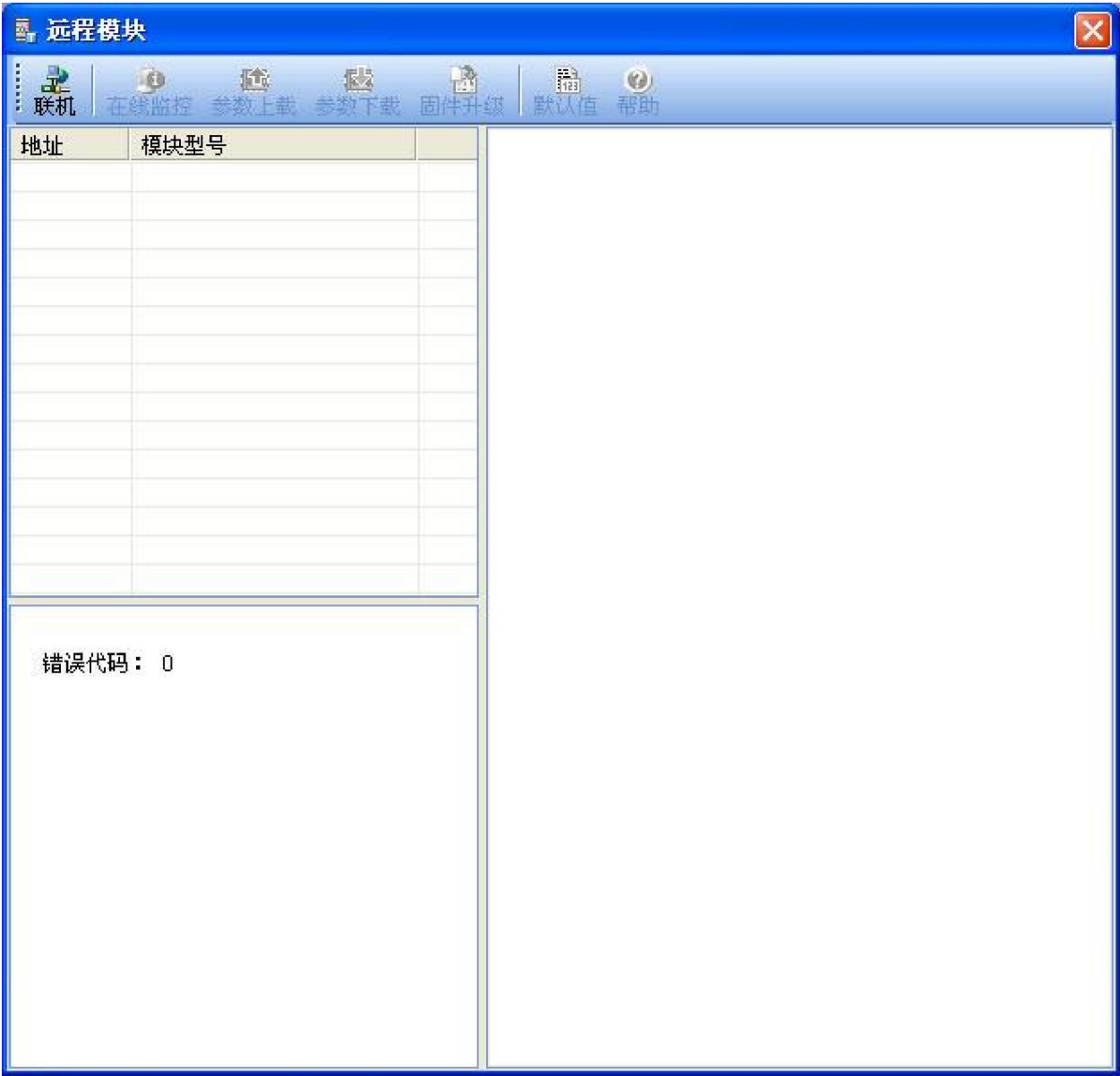
第九章、远程模块

本章介绍 Haiwell （海为）PLC 的扩展模块作为远程 IO 模块的使用方法

9.1 概述

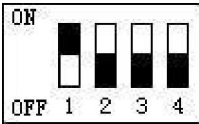
Haiwell（海为）PLC 的扩展模块带有一个 RS485 通讯口，既支持并行总线（用扩展总线挂到 PLC 主机的并行接口）也支持串行总线（用 RS485 通讯口与 PLC 主机的通讯口联网，主机使用通讯指令控制远程模块），当用串行总线进行扩展时（即远程 IO 模块），不受系统点数的扩展限制，可分布式安装。这一点对于有大量分散的开关量或者模拟量信号（温度、湿度、压差、风量、流量、风机转速、阀门开度等）需要进行采集和监控的系统极为重要，轻松实现分布式安装控制且可无限制点的扩展，极大地提高了控制系统的配置灵活度及日后的控制扩展能力，减少了各种信号的布线量，同时也减小了因模拟量信号线过长带来的干扰问题，节省工程投资成本。

点击菜单【工具/远程模块】或点击工具栏的  按钮，打开“远程模块”管理窗口。对远程模块的所有控制都通过本窗口完成。



9.2 模块站号设置

通过硬件上的 4 位 DIP 拨码开关设置远程模块站号（也叫站地址、通讯地址）。合法的站号为 1~254（0 为广播地址）。4 位 DIP 拨码开关可用于表示十进制数 1~15（二进制的数从 0001 ~ 1111），通过拨码开关修改站号的方法如下：



上图为用于设定模块站号的 4 位拨码开关，上面为 ON，下面为 OFF，图中黑色部分表示拨码开关的位置，把其中的一位拨到 ON 时表示该位为 1，拨到 ON 时表示该位为 0，上图中第 1 位为 ON，其它位为 OFF。拨码开关的第 1 位表示二进制的第 0 位（b0），拨码开关的第 4 位表示二进制的第 3 位（b3），由此，如上图所示的拨码开关的位置表示 0001，即 10 进制数为 1，表示 模块的站号为 1（出厂默认为 1）；若把第 1、2 位拨到 ON 位置，而其它位拨到 OFF 位置，则为 0011，即 10 进制数为 3，表示模块的站号为 3。

如果 4 位 DIP 拨码开关表示的硬地址不够使用或模块没有 DIP 拨码开关，可通过编程软件设置站号。与模块联机后即可在“通讯参数”栏的“地址”修改模块站号，然后点击“参数下载”，改变模块的站号。



注：如果模块有外部 DIP 拨码开关，以开关设置的地址为准（4 位拨码开关可设置的地址范围 1~15）；16 以上的地址或者模块没有 DIP 拨码开关则以编程软件设置的地址为准。

9.3 与模块联机

与网络上的一个或多个模块建立连接。因为模块是 RS485 通讯口，所以必须使用 RS232/485 转换器把计算机的 RS232 通讯口转换为 RS485 才可以与模块联机。只有在与 模块联机成功后，才可以对连接在网络上的模块进行控制，比如上下载 参数、在线监控等。

- 

1、点击“远程模块”窗口中的 **联机** 按钮，打开“联机”窗口。



- 2、相关参数设置，一般情况使用默认参数即可无需设置修改。

- PC 端口：选择计算机与模块通讯的串口。列表中可用的 COM 串口数目依不同计算机而不同，系统会自动查找出本计算机所有可用的 COM 串口供用户选择。

- 波特率：选择通讯波特率，系统默认为 19200。
- 资料格式：选择通讯数据格式，系统默认为"N,8,2 RTU"。
- 起始地址、终止地址：如果只是与单个模块联机，使用"单机查找"选项即可；如果是与多个模块联机，则"起始地址"填网络上模块的最小站号，"终止地址"填网络上模块的最大站号。
- 添加到列表、覆盖列表：选择把已建立联机的模块以添加或者覆盖方式加到列表框中。
- 单机查找： 如果只是与单个模块联机，勾起"单机查找"选项；如果是与多个模块联机，则必须取消"单机查找"选项并且设置"起始地址"和"终止地址"。

3、联机操作

- 联机：直接点"联机"按钮即可，找到的模块（联机成功）会在列表框列出。
- 查找：如果忘记了波特率等通讯参数，这时可以点"查找"按钮进行查找，查找功能会尝试以所有波特率、所有资料格式来与模块通讯，所以需要花费比较长时间等待，找到的模块（联机成功）会在列表框列出。

4、点击"退出"，关闭"联机"窗口。

5、如果联机不成功，有可能是以下情况所造成，仔细检查即可排除问题，如果无法解决请联系海为公司技术支持 。

- A、选择的 PC 端口不对
- B、选择的通讯参数与模块的通讯参数不对
- C、模块未上电
- D、通讯电缆连接有误或通讯接头未连接好
- E、使用"USB 转 RS232 串口数据线"时未正确安装它的驱动

9.4 参数修改

1、联机成功后，模块的参数会自动上载上来，如下画：



- A、左边上部为模块列表，此处显示已连接上的所有模块。可通过双击将列表中任意一个模块选为当前模块，所有对模块的在线操作如：上下载参数、固件升级等都只是对当前模块的操作，不影响其他模块。
- B、左边下部为监控区，在线监控时用于显示当前模块实时数据。
- 2、右边的模块参数会依模块型号的不同而不同，主要包括信号类型、使用工程量、工程量上下限、采样次数、零点修正、滤波时间等。
参见硬件手册的“[扩展模块参数](#)”。
- 3、用户可以任意修改模块参数，然后通过“参数下载”按钮下载到模块。
注意：信号类型的选择应与外部传感器的信号类型一致。

9.5 参数上传



点击参数上传按钮即可将当前模块的参数上传并显示在“远程模块”窗口中。

9.6 参数下载



点击参数下载按钮即可将修改过的参数下载到目标模块。

注意：修改过的任何参数，只有通过“参数下载”下载到目标模块才生效。没有执行“参数下载”操作，目标模块的参数将不会发生改变。

9.7 固件升级

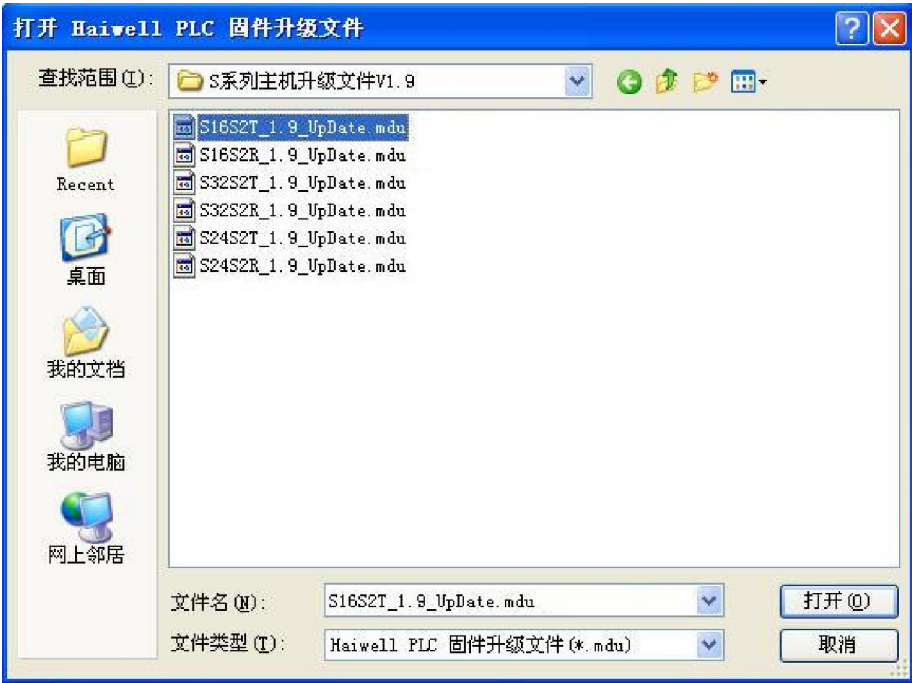
更新模块的固件程序，使模块支持新的功能。



- 1、点击固件升级按钮，，打开“固件升级”窗口



- 2、点击“打开”按钮，选择与被升级模块相对应的升级文件。



- 3、点击“升级”按钮，即可对模块的固件程序升级。
- 注：**如果升级过程中由于断电或者其他原因造成升级中断，需重新执行固件升级直至升级成功。

9.8 在线监控

- 1、点击  按钮进入监控状态。实时监控当前模块输入通道的输入数据，也可强制改变模块输出通道的输出值。



- 2、点击  按钮退出监控状态。

第十章、附录

附录列出了 Haiwell（海为）PLC 的各种系统资源。

10.1 SM系统状态位

SM 系统状态位是系统定义的一组特殊的内部继电器，在程序中使用的次数无限制，每一个系统状态位均有其特定的作用，未列出的请勿使用。

系统状态位	功能说明	属性	停电保持	出厂值
SM0	RUN 状态下为 1，STOP 状态下为 0	R	否	0
SM1	RUN 状态下为 0，STOP 状态下为 1	R	否	0
SM2	CPU 首次扫描时为 1，之后清 0	R	否	0
SM3	10ms 时钟脉冲	R	否	0
SM4	100ms 时钟脉冲	R	否	0
SM5	1s 时钟脉冲	R	否	0
SM8	扫描超时	R	否	0
SM9	PLC 开关状态	R	否	0
SM10	运行状态	R	否	0
SM11	系统故障	R	否	0
SM12	硬件配置表与模块不匹配	R	否	0
SM13	电池电压太低	R	否	0
SM14	被零除标志	R	否	0
SM15	数据溢出标志	R	否	0
SM16	COM1 通讯错误	R	否	0
SM17	COM2 通讯错误	R	否	0
SM18	COM3 通讯错误	R	否	0
SM19	COM4 通讯错误	R	否	0
SM20	COM5 通讯错误	R	否	0
SM25	HSC0 学习使能控制，0 为正常状态，1 为学习状态	R/W	否	0
SM26	HSC0 学习确认控制	R/W	否	0
SM30	HSC0 方向指示，0 为加，1 为减	R	否	0
SM31	HSC0 错误指示	R	否	0
SM33	HSC1 学习使能控制，0 为正常状态，1 为学习状态	R/W	否	0
SM34	HSC1 学习确认控制	R/W	否	0
SM38	HSC1 方向指示，0 为加，1 为减	R	否	0
SM39	HSC1 错误指示	R	否	0
SM41	HSC2 学习使能控制，0 为正常状态，1 为学习状态	R/W	否	0
SM42	HSC2 学习确认控制	R/W	否	0
SM46	HSC2 方向指示，0 为加，1 为减	R	否	0
SM47	HSC2 错误指示	R	否	0
SM49	HSC3 学习使能控制，0 为正常状态，1 为学习状态	R/W	否	0
SM50	HSC3 学习确认控制	R/W	否	0
SM54	HSC3 方向指示，0 为加，1 为减	R	否	0
SM55	HSC3 错误指示	R	否	0
SM57	HSC4 学习使能控制，0 为正常状态，1 为学习状态	R/W	否	0
SM58	HSC4 学习确认控制	R/W	否	0

SM62	HSC4 方向指示, 0 为加, 1 为减	R	否	0
SM63	HSC4 错误指示	R	否	0
SM65	HSC5 学习使能控制, 0 为正常状态, 1 为学习状态	R/W	否	0
SM66	HSC5 学习确认控制	R/W	否	0
SM70	HSC5 方向指示, 0 为加, 1 为减	R	否	0
SM71	HSC5 错误指示	R	否	0
SM73	HSC6 学习使能控制, 0 为正常状态, 1 为学习状态	R/W	否	0
SM74	HSC6 学习确认控制	R/W	否	0
SM78	HSC6 方向指示, 0 为加, 1 为减	R	否	0
SM79	HSC6 错误指示	R	否	0
SM81	HSC7 学习使能控制, 0 为正常状态, 1 为学习状态	R/W	否	0
SM82	HSC7 学习确认控制	R/W	否	0
SM86	HSC7 方向指示, 0 为加, 1 为减	R	否	0
SM87	HSC7 错误指示	R	否	0
SM93	PLS0 正转脉冲禁止	R/W	是	0
SM94	PLS0 反转脉冲禁止	R/W	是	0
SM95	PLS0 刹车功能禁止	R/W	是	0
SM96	PLS0 脉冲输出指示	R	是	0
SM97	PLS0 脉冲输出方向指示, 0 为正向, 1 为反向	R	是	0
SM98	PLS0 错误指示	R	是	0
SM99	PLS0 定位模式 0 为相对地址 1 为绝对地址	R/W	是	0
SM100	PLS0 脉冲输出完成	R	是	0
SM109	PLS1 正转脉冲禁止	R/W	是	0
SM110	PLS1 反转脉冲禁止	R/W	是	0
SM111	PLS1 刹车功能禁止	R/W	是	0
SM112	PLS1 脉冲输出指示	R	是	0
SM113	PLS1 脉冲输出方向指示, 0 为正向, 1 为反向	R	是	0
SM114	PLS1 错误指示	R	是	0
SM115	PLS1 定位模式 0 为相对地址 1 为绝对地址	R/W	是	0
SM116	PLS1 脉冲输出完成	R	是	0
SM125	PLS2 正转脉冲禁止	R/W	是	0
SM126	PLS2 反转脉冲禁止	R/W	是	0
SM127	PLS2 刹车功能禁止	R/W	是	0
SM128	PLS2 脉冲输出指示	R	是	0
SM129	PLS2 脉冲输出方向指示, 0 为正向, 1 为反向	R	是	0
SM130	PLS2 错误指示	R	是	0
SM131	PLS2 定位模式 0 为相对地址 1 为绝对地址	R/W	是	0
SM132	PLS2 脉冲输出完成	R	是	0
SM141	PLS3 正转脉冲禁止	R/W	是	0
SM142	PLS3 反转脉冲禁止	R/W	是	0
SM143	PLS3 刹车功能禁止	R/W	是	0
SM144	PLS3 脉冲输出指示	R	是	0
SM145	PLS3 脉冲输出方向指示, 0 为正向, 1 为反向	R	是	0
SM146	PLS3 错误指示	R	是	0

SM147	PLS3 定位模式 0 为相对地址 1 为绝对地址	R/W	是	0
SM148	PLS3 脉冲输出完成	R	是	0
SM157	PLS4 正转脉冲禁止	R/W	是	0
SM158	PLS4 反转脉冲禁止	R/W	是	0
SM159	PLS4 刹车功能禁止	R/W	是	0
SM160	PLS4 脉冲输出指示	R	是	0
SM161	PLS4 脉冲输出方向指示，0 为正向，1 为反向	R	是	0
SM162	PLS4 错误指示	R	是	0
SM163	PLS4 定位模式 0 为相对地址 1 为绝对地址	R/W	是	0
SM164	PLS4 脉冲输出完成	R	是	0
SM173	PLS5 正转脉冲禁止	R/W	是	0
SM174	PLS5 反转脉冲禁止	R/W	是	0
SM175	PLS5 刹车功能禁止	R/W	是	0
SM176	PLS5 脉冲输出指示	R	是	0
SM177	PLS5 脉冲输出方向指示，0 为正向，1 为反向	R	是	0
SM178	PLS5 错误指示	R	是	0
SM179	PLS5 定位模式 0 为相对地址 1 为绝对地址	R/W	是	0
SM180	PLS5 脉冲输出完成	R	是	0
SM189	PLS6 正转脉冲禁止	R/W	是	0
SM190	PLS6 反转脉冲禁止	R/W	是	0
SM191	PLS6 刹车功能禁止	R/W	是	0
SM192	PLS6 脉冲输出指示	R	是	0
SM193	PLS6 脉冲输出方向指示，0 为正向，1 为反向	R	是	0
SM194	PLS6 错误指示	R	是	0
SM195	PLS6 定位模式 0 为相对地址 1 为绝对地址	R/W	是	0
SM196	PLS6 脉冲输出完成	R	是	0
SM205	PLS7 正转脉冲禁止	R/W	是	0
SM206	PLS7 反转脉冲禁止	R/W	是	0
SM207	PLS7 刹车功能禁止	R/W	是	0
SM208	PLS7 脉冲输出指示	R	是	0
SM209	PLS7 脉冲输出方向指示，0 为正向，1 为反向	R	是	0
SM210	PLS7 错误指示	R	是	0
SM211	PLS7 定位模式 0 为相对地址 1 为绝对地址	R/W	是	0
SM212	PLS7 脉冲输出完成	R	是	0

10.2 SV系统寄存器

SV 系统寄存器是系统定义的一组特殊的内部寄存器，在程序中使用的次数无限制，每一个系统寄存器均有其特定的作用，未列出的请勿使用。

系统寄存器	功能说明	属性	停电保持	出厂值
SV0	当前扫描时间，单位 0.1ms	R	否	0
SV1	最小扫描时间，单位 0.1ms	R	否	0
SV2	最大扫描时间，单位 0.1ms	R	否	0
SV3	系统故障代码，见系统故障代码表	R	否	0
SV4	COM1 通讯错误代码	R	否	0
SV5	COM2 通讯错误代码	R	否	0

SV6	COM3 通讯错误代码	R	否	0
SV7	COM4 通讯错误代码	R	否	0
SV8	COM5 通讯错误代码	R	否	0
SV9	编译错误指令行号	R	否	0
SV11	CPU 主机 AI 输入断线报警，每位表示 1 个通道，0-正常，1-断线	R	否	0
SV12	年	R	否	0
SV13	月（1-12）	R	否	0
SV14	日（1-31）	R	否	0
SV15	时（0-23）	R	否	0
SV16	分（0-59）	R	否	0
SV17	秒（0-59）	R	否	0
SV18	星期（1-7，表示星期一至星期日）	R	否	0
SV19	PLC 站名称	R/W	是	0
SV20	PLC 站名称	R/W	是	0
SV21	PLC 站名称	R/W	是	0
SV22	PLC 站名称	R/W	是	0
SV23	PLC 站名称	R/W	是	0
SV24	PLC 站名称	R/W	是	0
SV25	程序扫描超时定时器，单位 ms	R/W	是	200 ms
SV26	PLC 地址 1~254	R	是	1
SV27	低字节为扩展模块数 0~31，高字节为类型	R	是	0
SV28	低字节为 CPU 模块代码，高字节为版本号	R	是	0
SV29	低字节为扩展模块 1 代码，高字节为版本号	R	是	0
SV30	扩展模块 2，定义同上	R	是	0
SV31	扩展模块 3，定义同上	R	是	0
SV32	扩展模块 4，定义同上	R	是	0
SV33	扩展模块 5，定义同上	R	是	0
SV34	扩展模块 6，定义同上	R	是	0
SV35	扩展模块 7，定义同上	R	是	0
SV36	扩展模块 8，定义同上	R	是	0
SV37	扩展模块 9，定义同上	R	是	0
SV38	扩展模块 10，定义同上	R	是	0
SV39	扩展模块 11，定义同上	R	是	0
SV40	扩展模块 12，定义同上	R	是	0
SV41	扩展模块 13，定义同上	R	是	0
SV42	扩展模块 14，定义同上	R	是	0
SV43	扩展模块 15，定义同上	R	是	0
SV44	COM1 通讯协议： 低字节低 4 位：0 - N,8, 2 For RTU 1 - E,8, 1 For RTU 2 - O 8, ,1 For RTU 3 - N,7, 2 For ASCII 4 - E,7, 1 For ASCII 5 - O,7, 1 For ASCII 6 - N,8, 1 For RTU（H/N 系列支持） 低字节高 4 位：0 - 2400 1 - 4800 2 - 9600 3 - 19200	R/W	是	30H, 19200, N,8, 2 RTU

	4 - 38400 5 - 57600 6 - 115200 (H/N 系列支持)			
SV45	COM1 通讯超时时间, 单位 ms	R/W	是	200ms
SV46	COM2 通讯协议, 同 COM1	R/W	是	30H, 19200, N, 8, 2 RTU
SV47	COM2 通讯超时时间, 单位 ms	R/W	是	200ms
SV48	PLC 程序大小	R	是	0
SV49	系统时钟低字 单位 16us	R	是	
SV50	系统时钟高字 单位 16us	R	是	
SV54	COM3 通讯协议, 同 COM1	R/W	是	30H, 19200, N, 8, 2 RTU
SV55	COM3 通讯超时时间, 单位 ms	R/W	是	200ms
SV56	COM4 通讯协议, 同 COM1	R/W	是	30H, 19200, N, 8, 2 RTU
SV57	COM4 通讯超时时间, 单位 ms	R/W	是	200ms
SV58	COM5 通讯协议, 同 COM1	R/W	是	30H, 19200, N, 8, 2 RTU
SV59	COM5 通讯超时时间, 单位 ms	R/W	是	200ms
SV60	HSC0 的当前段号	R	是	0
SV61	HSC0 的当前值低字	R	是	0
SV62	HSC0 的当前值高字	R	是	0
SV63	HSC0 的错误代码	R	是	0
SV64	HSC1 的当前段号	R	是	0
SV65	HSC1 的当前值低字	R	是	0
SV66	HSC1 的当前值高字	R	是	0
SV67	HSC1 的错误代码	R	是	0
SV68	HSC2 的当前段号	R	是	0
SV69	HSC2 的当前值低字	R	是	0
SV70	HSC2 的当前值高字	R	是	0
SV71	HSC2 的错误代码	R	是	0
SV72	HSC3 的当前段号	R	是	0
SV73	HSC3 的当前值低字	R	是	0
SV74	HSC3 的当前值高字	R	是	0
SV75	HSC3 的错误代码	R	是	0
SV76	HSC4 的当前段号	R	是	0
SV77	HSC4 的当前值低字	R	是	0
SV78	HSC4 的当前值高字	R	是	0
SV79	HSC4 的错误代码	R	是	0
SV80	HSC5 的当前段号	R	是	0
SV81	HSC5 的当前值低字	R	是	0
SV82	HSC5 的当前值高字	R	是	0
SV83	HSC5 的错误代码	R	是	0
SV84	HSC6 的当前段号	R	是	0
SV85	HSC6 的当前值低字	R	是	0
SV86	HSC6 的当前值高字	R	是	0
SV87	HSC6 的错误代码	R	是	0
SV88	HSC7 的当前段号	R	是	0
SV89	HSC7 的当前值低字	R	是	0
SV90	HSC7 的当前值高字	R	是	0

SV91	HSC7 的错误代码	R	是	0
SV92	PLS0 的当前段号	R	是	0
SV93	PLS0 的已输出脉冲数低字	R	是	0
SV94	PLS0 的已输出脉冲数高字	R	是	0
SV95	PLS0 的当前位置低字	R	是	0
SV96	PLS0 的当前位置高字	R	是	0
SV97	PLS0 的错误代码	R	是	0
SV98	PLS1 的当前段号	R	是	0
SV99	PLS1 的已输出脉冲数低字	R	是	0
SV100	PLS1 的已输出脉冲数高字	R	是	0
SV101	PLS1 的当前位置低字	R	是	0
SV102	PLS1 的当前位置高字	R	是	0
SV103	PLS1 的错误代码	R	是	0
SV104	PLS2 的当前段号	R	是	0
SV105	PLS2 的已输出脉冲数低字	R	是	0
SV106	PLS2 的已输出脉冲数高字	R	是	0
SV107	PLS2 的当前位置低字	R	是	0
SV108	PLS2 的当前位置高字	R	是	0
SV109	PLS2 的错误代码	R	是	0
SV110	PLS3 的当前段号	R	是	0
SV111	PLS3 的已输出脉冲数低字	R	是	0
SV112	PLS3 的已输出脉冲数高字	R	是	0
SV113	PLS3 的当前位置低字	R	是	0
SV114	PLS3 的当前位置高字	R	是	0
SV115	PLS3 的错误代码	R	是	0
SV116	PLS4 的当前段号	R	是	0
SV117	PLS4 的已输出脉冲数低字	R	是	0
SV118	PLS4 的已输出脉冲数高字	R	是	0
SV119	PLS4 的当前位置低字	R	是	0
SV120	PLS4 的当前位置高字	R	是	0
SV121	PLS4 的错误代码	R	是	0
SV122	PLS5 的当前段号	R	是	0
SV123	PLS5 的已输出脉冲数低字	R	是	0
SV124	PLS5 的已输出脉冲数高字	R	是	0
SV125	PLS5 的当前位置低字	R	是	0
SV126	PLS5 的当前位置高字	R	是	0
SV127	PLS5 的错误代码	R	是	0
SV128	PLS6 的当前段号	R	是	0
SV129	PLS6 的已输出脉冲数低字	R	是	0
SV130	PLS6 的已输出脉冲数高字	R	是	0
SV131	PLS6 的当前位置低字	R	是	0
SV132	PLS6 的当前位置高字	R	是	0
SV133	PLS6 的错误代码	R	是	0
SV134	PLS7 的当前段号	R	是	0

SV135	PLS7 的已输出脉冲数低字	R	是	0
SV136	PLS7 的已输出脉冲数高字	R	是	0
SV137	PLS7 的当前位置低字	R	是	0
SV138	PLS7 的当前位置高字	R	是	0
SV139	PLS7 的错误代码	R	是	0
SV140	当值为-23206 时禁止 Y 输出	R/W	是	0
SV141	通讯指令执行间隔单位毫秒	R/W	是	0
SV142	PLC 软地址(1~254)	R	是	0
SV143	PLC 外部 DIP 设定地址	R	是	0
SV144	流水号低字	R	是	0
SV145	流水号高字	R	是	0
SV146	方向早于脉冲输出的时间(5~100us)	R/W	是	5
SV151	锁定数据个数	R	是	0
SV152	模拟输入 1	R	是	0
SV153	模拟输入 2	R	是	0
SV154	模拟输出 1	R/W	是	0
SV155	模拟输出 2	R/W	是	0
SV156	PLS0 的机械原点低字	R	是	0
SV157	PLS0 的机械原点高字	R	是	0
SV158	PLS0 的反向间隙补偿脉冲数	R/W	是	0
SV160	PLS1 的机械原点低字	R	是	0
SV161	PLS1 的机械原点高字	R	是	0
SV162	PLS1 的反向间隙补偿脉冲数	R/W	是	0
SV164	PLS2 的机械原点低字	R	是	0
SV165	PLS2 的机械原点高字	R	是	0
SV166	PLS2 的反向间隙补偿脉冲数	R/W	是	0
SV168	PLS3 的机械原点低字	R	是	0
SV169	PLS3 的机械原点高字	R	是	0
SV170	PLS3 的反向间隙补偿脉冲数	R/W	是	0
SV172	PLS4 的机械原点低字	R	是	0
SV173	PLS4 的机械原点高字	R	是	0
SV174	PLS4 的反向间隙补偿脉冲数	R/W	是	0
SV176	PLS5 的机械原点低字	R	是	0
SV177	PLS5 的机械原点高字	R	是	0
SV178	PLS5 的反向间隙补偿脉冲数	R/W	是	0
SV180	PLS6 的机械原点低字	R	是	0
SV181	PLS6 的机械原点高字	R	是	0
SV182	PLS6 的反向间隙补偿脉冲数	R/W	是	0
SV184	PLS7 的机械原点低字	R	是	0
SV185	PLS7 的机械原点高字	R	是	0
SV186	PLS7 的反向间隙补偿脉冲数	R/W	是	0
SV801	HSC0 的频率值低字	R	是	0
SV802	HSC0 的频率值高字	R	是	0
SV803	HSC1 的频率值低字	R	是	0

SV804	HSC1 的频率值高字	R	是	0
SV805	HSC2 的频率值低字	R	是	0
SV806	HSC2 的频率值高字	R	是	0
SV807	HSC3 的频率值低字	R	是	0
SV808	HSC3 的频率值高字	R	是	0
SV809	HSC4 的频率值低字	R	是	0
SV810	HSC4 的频率值高字	R	是	0
SV811	HSC5 的频率值低字	R	是	0
SV812	HSC5 的频率值高字	R	是	0
SV813	HSC6 的频率值低字	R	是	0
SV814	HSC6 的频率值高字	R	是	0
SV815	HSC7 的频率值低字	R	是	0
SV816	HSC7 的频率值高字	R	是	0
SV817	历史故障代码	R	是	0
SV818	历史故障代码	R	是	0
SV819	历史故障代码	R	是	0
SV820	历史故障代码	R	是	0
SV821	历史故障代码	R	是	0
SV822	历史故障代码	R	是	0
SV823	历史故障代码	R	是	0
SV824	历史故障代码	R	是	0
SV825	历史故障代码	R	是	0
SV826	历史故障代码	R	是	0
SV827	历史故障代码	R	是	0
SV828	历史故障代码	R	是	0
SV829	历史故障代码	R	是	0
SV830	历史故障代码	R	是	0
SV831	历史故障代码	R	是	0
SV832	历史故障代码	R	是	0

10.3 系统中断表

Haiwell（海为）PLC 提供多达 52 个系统中断资源，包括脉冲输出、边沿捕捉、高速计数和计时器中断。

中断号	中断类型	说明	优先级
1	脉冲输出中断	PLS0 开始输出脉冲	由高到低 (即中断号小的 优先级高于中断 号大的)
2		PLS0 输出脉冲结束	
3		PLS1 开始输出脉冲	
4		PLS1 输出脉冲结束	
5		PLS2 开始输出脉冲	
6		PLS2 输出脉冲结束	
7		PLS3 开始输出脉冲	
8		PLS3 输出脉冲结束	
9		PLS4 开始输出脉冲	
10		PLS4 输出脉冲结束	
11		PLS5 开始输出脉冲	
12		PLS5 输出脉冲结束	
13		PLS6 开始输出脉冲	
14		PLS6 输出脉冲结束	
15		PLS7 开始输出脉冲	
16		PLS7 输出脉冲结束	
17	边沿捕捉中断	X0 上升沿捕捉	
18		X1 上升沿捕捉	

中断号	中断类型	说明	优先级
19	高速计数中断	X2 上升沿捕捉	
20		X3 上升沿捕捉	
21		X4 上升沿捕捉	
22		X5 上升沿捕捉	
23		X6 上升沿捕捉	
24		X7 上升沿捕捉	
25		X0 下降沿捕捉	
26		X1 下降沿捕捉	
27		X2 下降沿捕捉	
28		X3 下降沿捕捉	
29		X4 下降沿捕捉	
30		X5 下降沿捕捉	
31		X6 下降沿捕捉	
32		X7 下降沿捕捉	
33		HSC0 当前值=设定值(每段设定值都会产生)	
34		HSC0 输入方向改变	
35		HSC1 当前值=设定值(每段设定值都会产生)	
36		HSC1 输入方向改变	
37		HSC2 当前值=设定值(每段设定值都会产生)	
38		HSC2 输入方向改变	
39		HSC3 当前值=设定值(每段设定值都会产生)	
40		HSC3 输入方向改变	
41		HSC4 当前值=设定值(每段设定值都会产生)	
42		HSC4 输入方向改变	
43		HSC5 当前值=设定值(每段设定值都会产生)	
44		HSC5 输入方向改变	
45		HSC6 当前值=设定值(每段设定值都会产生)	
46		HSC6 输入方向改变	
47		HSC7 当前值=设定值(每段设定值都会产生)	
48		HSC7 输入方向改变	
49	计时器中断	T252 计时到	
50		T253 计时到	
51		T254 计时到	
52		T255 计时到	

10.4 通讯地址代码表

一、Haiwell（海为）PLC 位元件表（相当于 Modbus 地址类型 0、1，支持 Modbus 功能码 1、2、5、15）

元 件	名称	元件范围		读写属性	Modbus 通讯地址码		说 明
		E/S 系列	H/N 系列		16 进制	10 进制	
X	开关量输入	X0~X255	X0~X1023	只读	0x0000~0x03FF	0~1023	
Y	开关量输出	Y0~Y255	Y0~Y1023	可读/写	0x0600~0x09FF	1536~2559	
M	内部继电器	M0~M2047	M0~M12287	可读/写	0x0C00~0x3BFF	3072~15359	
T	计时器（输出线圈）	T0~T127	T0~T1023	可读/写	0x3C00~0x3FFF	15360~16383	
C	计数器（输出线圈）	C0~C127	C0~C255	可读/写	0x4000~0x40FF	16384~16639	
SM	系统状态位	SM0~SM215	SM0~SM215	全部可读部分可写	0x4200~0x42D7	16896~17111	
S	步进继电器	S0~S255	S0~S2047	可读/写	0x7000~0x77FF	28672~30719	

二、Haiwell（海为）PLC 寄存器元件表（相当于 Modbus 地址类型 3、4，支持 Modbus 功能码 3、4、6、16）

元 件	名称	元件范围		读写属性	Modbus 通讯地址码		说 明
		E/S 系列	H/N 系列		16 进制	10 进制	
CR	扩展模块参数寄存器	CR0~CR159		可读/部分可写	0x00~0x9F	0~159	通 过 Modbus 协 议 访 问 扩 展 模 块 参 数 时 用
AI	模拟量输入寄存器	AI0~AI63	AI0~AI255	只读	0x0000~0x00FF	0~255	
AQ	模拟量输出寄存器	AQ0~AQ63	AQ0~AQ255	可读/写	0x0100~0x01FF	256~511	

元 件	名称	元件范围		读写属性	Modbus 通讯地址码		说 明
		E/S 系列	H/N 系列		16 进制	10 进制	
V	内部寄存器	V0~V2047	V0~V14847	可读/写	0x0200~0x3BFF	512~15359	
TV	计时器（当前值寄存器）	TV0~TV127	TV0~TV1023	可读/写	0x3C00~0x3FFF	15360~16383	
CV	计数器（当前值寄存器）	CV0~CV127	CV0~CV255	可读/写	0x4000~0x40FF	16384~16639	16 位寄存器，其中 CV48~CV79 为 32 位寄存器
SV	系统寄存器	SV0~SV154	SV0~SV900	全部可读部分可写	0x4400~0x4784	17408~18308	

三、说明：

- 1、Haiwell（海为）PLC 采用标准 Modbus 协议（支持 RTU 和 ASCII 格式），能与所有支持 Modbus 协议的 HMI 和组态软件连接通讯。
- 2、Haiwell（海为）PLC 的 Modbus 地址号从 0 开始，有些 HMI 或组态软件则从 1 开始，如果 HMI 或组态软件的 Modbus 地址从 0 开始则直接使用通讯地址，如 M0 为 0x3072，V0 为 4x0512；如果 HMI 或组态软件的 Modbus 地址从 1 开始则需将 元件地址号加 1，如 M0 为 0x3073[3072+1]，V0 为 4x0513[512+1]。地址的首位是 Modbus 协议的元件类型（0/1 为位元件，3/4 为寄存器元件，类型 0/4 表示可读写，类型 1/3 表示只读）后面的其余数位才是元件地址号。

10.5 错误代码表

一、系统错误代码表：

错误代码	功能说明
0	系统正常
1	CPU 固件不完整
2	CPU 内存 1 访问异常
3	CPU 内存 2 访问异常
4	RTC 访问异常
5	访问 CPU I/O 失败
6	CPU 存储器 3 访问异常
7	I/O 板访问失败
8	增强总线工作异常
60	1 号扩展模块升级失败
61	2 号扩展模块升级失败
62	3 号扩展模块升级失败
63	4 号扩展模块升级失败
64	5 号扩展模块升级失败
65	6 号扩展模块升级失败
66	7 号扩展模块升级失败
67	8 号扩展模块升级失败
68	9 号扩展模块升级失败
69	10 号扩展模块升级失败
70	11 号扩展模块升级失败
71	12 号扩展模块升级失败
72	13 号扩展模块升级失败
73	14 号扩展模块升级失败
74	15 号扩展模块升级失败

错误代码	功能说明
75	扩展模块硬件故障
89	编程软件版本太低
90	用户程序损坏
91	步元件超出范围
92	步合并超出范围
93	表格记录项数超出范围
94	取边沿条数超出范围
95	掉电配置数据非法
96	非法指令 ID
97	非法操作数
98	同类指令条数超出范围
99	无结束指令
100	访问 1 号扩展模块 I/O 失败
101	访问 2 号扩展模块 I/O 失败
102	访问 3 号扩展模块 I/O 失败
103	访问 4 号扩展模块 I/O 失败
104	访问 5 号扩展模块 I/O 失败
105	访问 6 号扩展模块 I/O 失败
106	访问 7 号扩展模块 I/O 失败
107	访问 8 号扩展模块 I/O 失败
108	访问 9 号扩展模块 I/O 失败
109	访问 10 号扩展模块 I/O 失败
110	访问 11 号扩展模块 I/O 失败
111	访问 12 号扩展模块 I/O 失败
112	访问 13 号扩展模块 I/O 失败
113	访问 14 号扩展模块 I/O 失败
114	访问 15 号扩展模块 I/O 失败
131	RTC 电池失效
132	扩展模块供电不正常
140	硬件配置不匹配
141	扫描超时 看门狗动作
142	有锁定数据

二、通讯错误代码表：

错误代码	功能说明
0	正常
1	功能码 错误
2	资料地址错误
3	资料值错误
4	通讯帧太短或太长，最小 4 字节，最大 128 字节
5	内含非 ASCII 字符
6	从机忙接收资料超时
7	无结束符
8	写数据信息太长或太短
9	校验码错误

错误代码	功能说明
10	申请的资源被占用
11	固件与硬件不匹配
12	程序容量超限，禁止写入

10.6 编程电缆接线图

计算机侧（RS-232）

PLC 侧（COM1 口）

DB9 母插头

4 芯 S 端子 公插头

