PLC 用户手册

目录

1. SamSoarII.....7
2. 快速入门.....7
2.1 了解什么是 SamSoarII.....8
2.2 系统需求.....8
2.3 SamSoarII 的安装和卸载.....9
2.4 SamSoarII 的运行.....13
2.5 快速上手 SamSoarII.....14
2.6 SamSoarII 的基本操作.....23
3. 编辑界面.....27
3.1 菜单栏.....27
3.2 工具栏.....31
3.3 查找.....34
3.4 替换.....37
3.5 软元件管理器.....40
3.6 软元件初始化.....41
3.7 编译错误表.....42
3.8 监视.....43
3.9 PLC 参数设置.....48
3.10 系统设置.....52
3.11 通信设置.....55
4. PLC 指令.....56
4.1 位逻辑.....57
4.1.1 -| |-.....57
4.1.2 -|||-.....58
4.1.3 -|/|-.....59
4.1.4 -|/||-.....59
4.1.5 -|↑|-.....60
4.1.6 -|↓|-.....61
4.1.7 -↑-.....62
4.1.8 -↓-.....62
4.1.9 -/-.....63
4.1.10 -( )-.....63
4.1.11 -(|)-.....64
4.1.12 -(S)-.....64
4.1.13 -(SI)-.....65
4.1.14 -(R)-.....65
4.1.15 -(RI)-.....66
4.1.16 ALT.....66
4.1.17 ALTP.....67
4.2 比较.....68
4.2.1 ==.....68
4.2.2 <>.....69

4.2.3 >=....71
4.2.4 <=....72
4.2.5 >....74
4.2.6 <....75
4.2.7 CMP/CMPD/CMPF....76
4.2.8 ZCP/ZCPD/ZCPF....77
4.3 转换....78
4.3.1 WTOD....79
4.3.2 DTOW....79
4.3.3 DTOF....80
4.3.4 BIN....80
4.3.5 BCD....81
4.3.6 ROUND....81
4.3.7 TRUNC....82
4.4 逻辑运算....83
4.4.1 INVW/INVD....83
4.4.2 ANDW/ANDD....84
4.4.3 ORW/ORD....84
4.4.4 XORW/XORD....85
4.4.5 NEGW/NEGD....86
4.5 传送....87
4.5.1 MOV/MOVD/MOVF....87
4.5.2 MVBLK/MVDBLK....88
4.5.3 FMOV/FMOVD....88
4.5.4 SMOV....89
4.5.5 XCH/XCHD/XCHF....90
4.6 浮点数运算....90
4.6.1 ADDF....91
4.6.2 SUBF....91
4.6.3 MULF....92
4.6.4 DIVF....93
4.6.5 SQRT....93
4.6.6 SIN....94
4.6.7 COS....94
4.6.8 TAN....95
4.6.9 LN....95
4.6.10 EXP....96
4.6.11 LOG....97
4.6.12 POW....97
4.7 整数运算....100
4.7.1 ADD/ADDD....100
4.7.2 SUB/SUBD....101
4.7.3 MUL/MULW/MULD....102
4.7.4 DIV/DIVW/DIVD....103

4.7.5 INC/INCD....104
4.7.6 DEC/DECD....104
4.7.7 FACT....105
4.8 定时器....106
4.8.1 TON....106
4.8.2 TONR....109
4.8.3 TOF....109
4.9 计数器....110
4.9.1 CTU....111
4.9.2 CTD....111
4.9.3 CTUD....112
4.10 程序控制....113
4.10.1 FOR/NEXT....113
4.10.2 JMP/LBL....114
4.10.3 CALL....115
4.10.4 CALLM....116
4.10.5 STL....117
4.11 移位指令....136
4.11.1 SHL/SHLD....137
4.11.2 SHR/SHRD....138
4.11.3 SHLB....139
4.11.4 SHRB....140
4.11.5 ROL/ROLD....141
4.11.6 ROR/RORD....142
4.12 中断指令....143
4.12.1 ATCH....143
4.12.2 DTCH....145
4.12.3 EI....146
4.12.4 DI....146
4.13 实时时钟....146
4.13.1 TRD....147
4.13.2 TWR....147
4.14 通信指令....148
4.14.1 MBUS....156
4.14.2 SEND....162
4.14.3 REV....164
4.15 脉冲输出指令....170
4.15.1 PLSF....171
4.15.2 PWM....173
4.15.3 PLSY....176
4.15.4 PLSR/PLSA....180
4.15.5 PLSNEXT....188
4.15.6 PLSSTOP....190
4.15.7 ZRN....190

4.15.8 PTO....193
4.15.9 DRVI/DRVA....197
4.15.10 TBL....199
4.15.11 POLYLINEF/POLYLINEI....203
4.15.12 LINEF/LINEI....209
4.15.13 ARCF/ARCI....213
4.15.14 BLOCK....217
4.15.15 HMIBLOCK....220
4.15.16 PAUSE....227
4.15.17 ZRNR....228
4.15.18 JOG....238
4.15.19 FOLLOW....240
4.15.20 平面系统设置....241
4.16 高速计数指令....298
4.16.1 HCNT....298
4.16.2 CAM....303
4.17 PID 指令....311
5. 函数功能块....330
5.1 C 语言入门....330
5.1.1 C 语言程序类型....330
5.1.2 变量初值和类型转换....334
5.1.3 基本运算符和表达式....335
5.1.4 一维数值组....336
5.1.5 小结....338
5.2 C 语言程序设计初步....339
5.2.1 C 程序的语句....339
5.2.2 分支结构程序....340
5.2.3 Switch 语句....342
5.2.4 循环结构顺序....343
5.2.5 break 语句....344
5.2.6 continue 语句....345
5.2.7 小结....345
5.3 函数功能块的设计....346
5.3.1 创建一个新的函数功能块....346
5.3.2 能被 CALLM 指令调用的函数的条件....349
5.3.3 兼容一代 SAMSOAR Developer 的设计....350
5.3.4 通过构造块来编辑指令表....350
5.3.5 软元件的使用....354
6. 软元件的使用....355
6.1 字取位....356
6.2 位合成字....357
6.3 局部变量....358
7. 仿真界面....359
7.1 快速开始....359

7.2 仿真控制.....360
7.3 单步调试.....361
7.4 程序断点.....362
7.5 软元件断点.....365
8. SFC 编程.....366
8.1 SFC 简介.....367
8.2 SFC 编辑模式.....375
8.3 SFC 检查编译.....379
8.4 SFC 查看模式.....381
9. 寄存器.....382
10. 通信协议.....385
11. 扩展模块.....388
12. 硬件手册.....401
13. 联系方式.....406

1. SamSoarII

SAMSOARII 第二代编辑软件是在第一代编辑软件 SAMSOAR Developer 的基础上，针对显控推出的新一代编译型可编程控制器（PLC）的组态程序的图形界面编辑工具。与之前的版本相比，在保留了其原有功能的基础上，对运行性能和用户体验等方面上做了大幅度的改进。

本公司推出的 PLC 产品简单易学，应用范围广泛，其帮助系统能够帮助您掌握 SAMSOAR II 软件的基本操作与用法。

帮助文档包括：

1、 快速入门
2、 编辑界面
3、 指令系统
4、 寄存器
5、 仿真界面
6、 通信协议
7、 硬件手册
8、 系统设置
9、 联系方式

2. 快速入门

内容如下：

1、 了解什么是 SamSoarII
2、 SamSoarII 对系统的最低要求
3、 SamSoarII 的安装和卸载
4、 SamSoarII 的运行
5、 SamSoarII 快速开始
6、 SamSoarII 的基本操作

2.1 了解什么是 SamSoarII

SAMSOARII 是针对深圳市显控科技股份有限公司推出的可编程逻辑控制器（PLC）的梯形图编辑软件。该软件可以帮你快速准确的编辑一个 PLC 的梯形图工程。在梯形图中支持主程序、子程序、C 语言函数功能块的编写，每个程序中包含了若干个网络，每个函数块中也包含了若干个函数。用户可以根据需要自己添加网络和函数。

在编辑梯形图的时候支持使用鼠标点击工具栏输入，也支持直接键盘键入命令或者通过双击树形目录处的命令，将选中的命令添加到图标指定的位置。在编写 C 语言的函数块时也支持自动缩进，代码补全等功能。

SAMSOARII 支持三种编辑模式分别是：写入模式、注释编辑模式、监视模式。其中，写入模式支持正常的编辑、修改、转换梯形图等操作；注释编辑模式支持修改元件的注释，要注意的是此时只能修改梯形图的注释，不能插入或修改梯形图元件；监视模式支持监视梯形图元件在 PLC 中的值，另外除了支持梯形图监视以外还支持软元件登陆监视，可以批量监视软元件的值。在执行监视模式的时候可以选择串口或者 USB 口进行监视。

SAMSOARII 支持 USB 和串口的上下载，在上下载和执行监控之前可以通过点击通信测试按钮来检测 PLC 是否连接到计算机。在通过串口进行通信的时候支持自动检测波特率的功能。

SAMSOARII 还支持设置自动定时保存功能，通过在系统设置对话框中可以设置自动保存的时间，使您的工程更加安全无忧。

SAMSOARII 对上下载工程做了严格的保密措施，可以在 PLC 设置对话中配置上下载密码，若不输入正确的密码无法进行上载操作。

2.2 系统需求

SamSoarII 对系统的最低要求

1. SAMSOARII 支持的运行环境

本软件支持 WinXP，Win7，Win8 和 Win10 操作系统，并安装了.NET Framework 4.0 或以上版本，DirectX 9.0 或以上版本的电脑。

2. SAMSOARII 安装操作须知

需特别注意的是，安装此软件的电脑中必须含有.NET Framework 4.0 或以上的版本，否则软件将拒绝安装（若安装后卸载.NET Framework 4.0，本软件将无法正常运行）。因此.NET Framework 4.0 会随着安装包一起发布（在目录 DotNetFX40 下）。安装过程中若检测到您电脑上没有.NET Framework 4.0 或以上的版本，则会自动帮您安装，耐心等待安装过程结束即可继续运行本软件的安装程序，若随安装包一起发布的软件必备。丢失或与安装软件不在同一目录则无法顺利安装。您也可以访问微软官网，搜索.NET Framework 4.0 并下载对应的安装包进行安装。

对于 XP 系统，可能会因为缺少安装.NET Framework 4.0 所必须的图像处理程序 wic_x86_chs.exe 而安装失败，因此我们也将其一起与安装程序发布，同样，安装过程中会自动检测是否需要安装该程序。

2.3 SamSoarII 的安装和卸载

1. 软件的安装

SamSoarII 是专为标准 Microsoft Windows 系统设计的 32 位应用软件。安装软件前，请确认该计算机的软件和硬件是否符合安装的最低需求。

使用随产品配送的安装盘来安装 SamSoarII。具体安装步骤如下：

1. 启动 Windows 并运行安装包程序 SamSoarII Setup。

2. 在弹出的“欢迎使用”对话框中点击“下一步”，如图 1-2 所示：



图 1.2

3. 在弹出的“选择目标位置”对话框中可以选择程序的安装目录， 在安装之前请确定磁盘是否有有足够的剩余空间，否则的话会导致安装失败。程序默认路径为“C:\Program Files\SamSoarII”，用户可以根据自己实际情况选择安装，如图 1.3 所示，然后点击下一步：

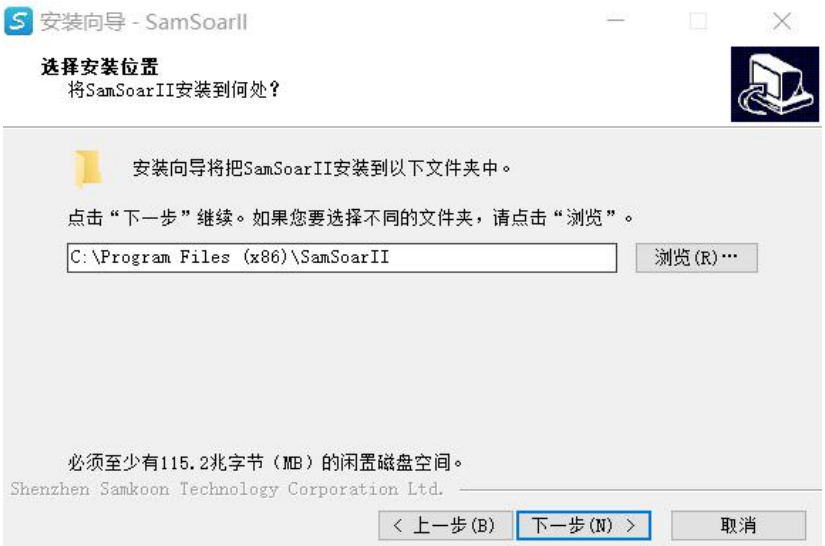


图 1.3

4. 在弹出的“选择开始菜单”对话框中点击“下一步”，如图 1.4:



图 1.4

5. 然后在弹出的对话框中选择下一步，如图 1.5:

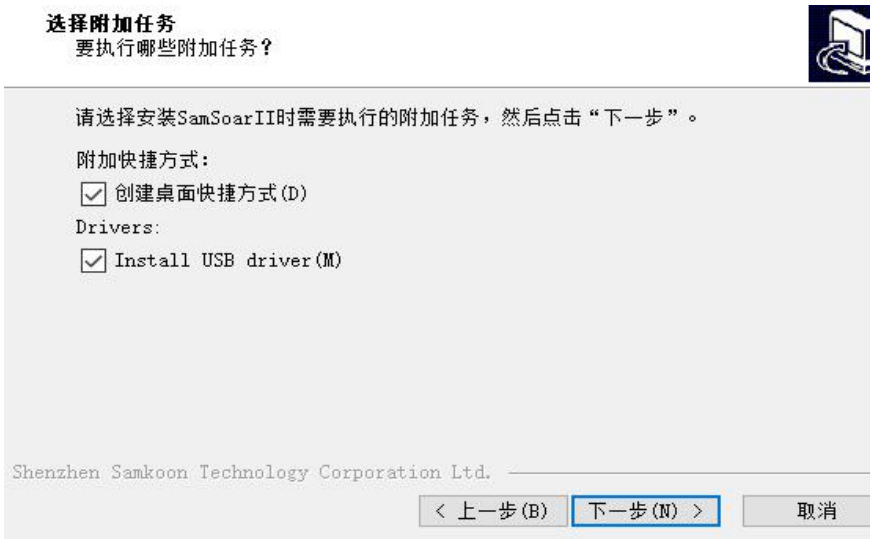


图 1.5

6. 此时会弹出提示选择安装的对话框，在确认选择无误的情况下请点击“安装”按钮，如图 1.6 所示:

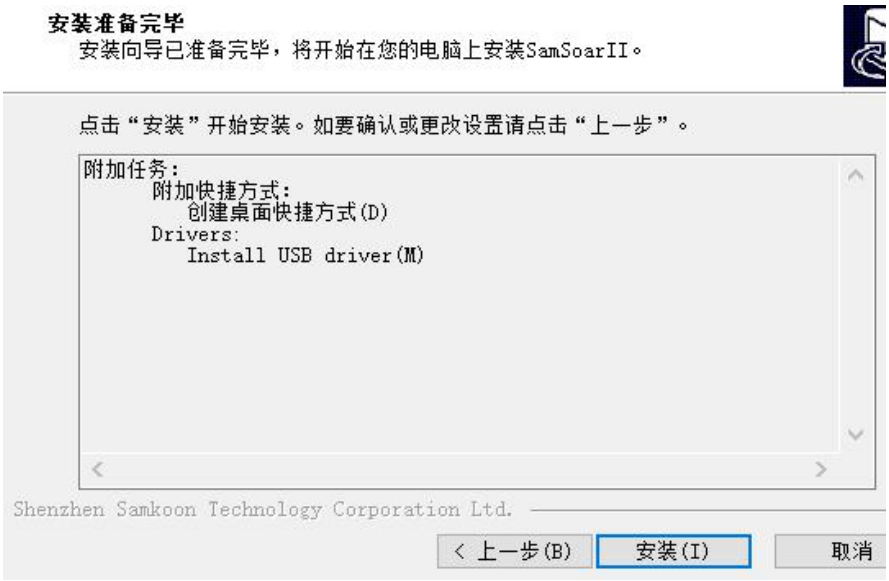


图 1.6

7. 此时会弹出正在安装的界面，如图 1.7 所示:

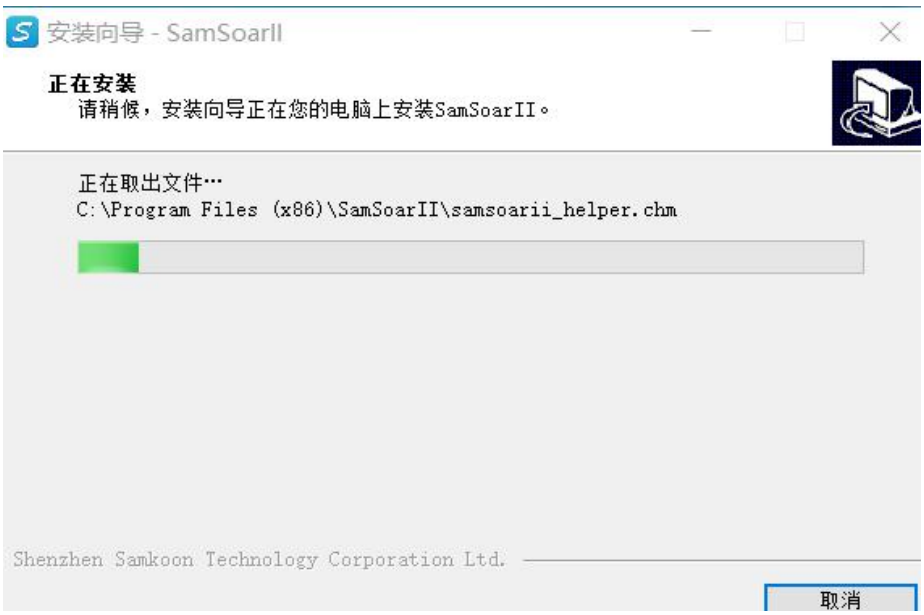


图 1.7

8. 安装程序正在将 SamSoarII 安装到指定目录下，请等待直到程序安装完毕，此时会弹出安装完成对话框，如图 1-8 所示：



图 1.8

9. 点击完成按钮，至此 SamSoarII 已经成功安装到您的计算机当中。

2. 软件的卸载

1. 想要卸载 SamSoarII 的话可以点击开始->程序->SamSoarII 下的卸载 SamSoarII 选项；如图 1-9 所示：



图 1.9

2. 此时会弹出是否确定卸载对话框，如图 1.10 所示：



图 1.10

3. 点击确定的话会弹出卸载进度对话框 如图 1-11 所示：

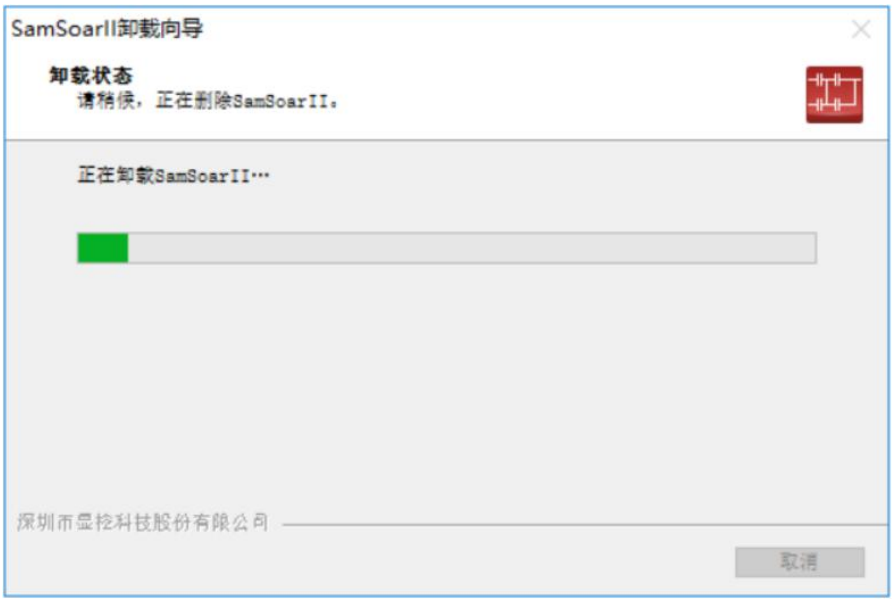


图 1.11

4. 至此软件将从本地计算机移除。

2.4 SamSoarII 的运行

SamSoarII.exe 在安装完成后，会在桌面上放置运行快捷方式，如图 1.1 所示：



图 1.1

同时 Windows 开始菜单中也添加了相应的 SamSoarII 程序，双击桌面上的快捷方式或已经保存的工程文件，或者单击开始菜单中 SamSoarII 程序组的 SamSoarII.exe 运行组态软件，运行界面如下图 1.2：

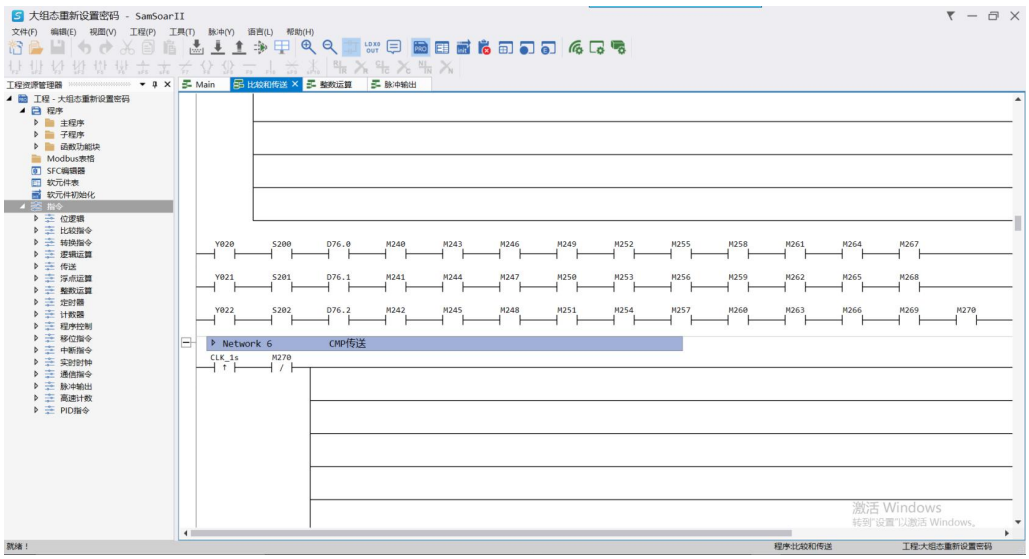


图 1.2 软件运行画面

2.5 快速上手 SamSoarII

SamSoarII 快速开始

1. 简介

如果您首次接触 FGs 产品，我们建议采用一个简便的方法：从简单实用的应用开始。即使没有任何有关 FGs 产品的经验，仍然可以很简单的实现操作。一切都从快速开始章节开始。

这个例子用于展示如何使用 FGs-16MR-A，使用 SAMSOAR Developer 编程工具进行编程和测试。

随后的各章节将进行更详尽的描述，为您提供更多的信息。

2. 准备硬件

范例：楼梯照明

通过楼梯照明控制的案例说明使用 FGs 实现一个应用。此建筑具有地面一层和地上三层，每层都有按钮控制电灯开关。快速按下这些按钮中的一个，楼梯内所有四盏灯都会开启并持续照明 5 分钟。

这些按钮分别与 FGs 的 4 个输入点 X0, X1, X2 和 X3 连接。4 盏灯通过 FGs 上的一个继电器输出点 Y0 进行控制。

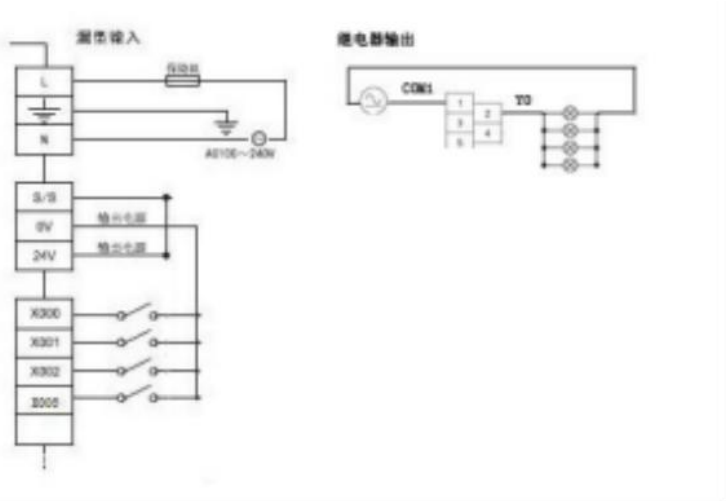


图 1.1

4. FGs-32MT-A 安装

DIN 导轨安装

1. 保持导轨固定点的间隔为 75 mm。
2. 打开模块底部的 DIN 夹子，将 FGs 背部卡在 DIN 导轨上。
3. 将 FGs 贴近 DIN 导轨，合上 DIN 夹子。检查模块上 DIN 夹子与 DIN 导轨是否紧密固定好。

为避免 FGs 损坏，不要直接按压 FGs 正面，而要按压安装孔的部分。

参照图 1-1 接线：

将 FGs 的端子 L 和端子 N 分别与 AC220V 电源的火线和零线连接。

将 FGs 的端子 S/S 与 FGs 的端子 24V 直接相连。

将 4 个按钮开关一端连接到 FGs 的端子 X0 到 X3，另一端统一接到 FGs 的端子 0V。

将 4 个灯的一端统一接到 FGs 的端子 Y0，另一端统一和 AC220V 电源零线连接。AC220V 电源火线与 FGs 的端子 COM1 连接。

将电脑的 USB 接口连接到 FGs。

注意：更详细的硬件安装以及接线信息，请查阅 FGs 硬件手册。

4. 编辑程序

4.1 启动 SAMSOARII 编程工具

双击桌面上的快捷方式，或者单击开始菜单中 SAMSOARII 程序组的 SAMSOARII.exe 运行组态软件。

进入 SAMSOARII 编程工具的运行界面如图。

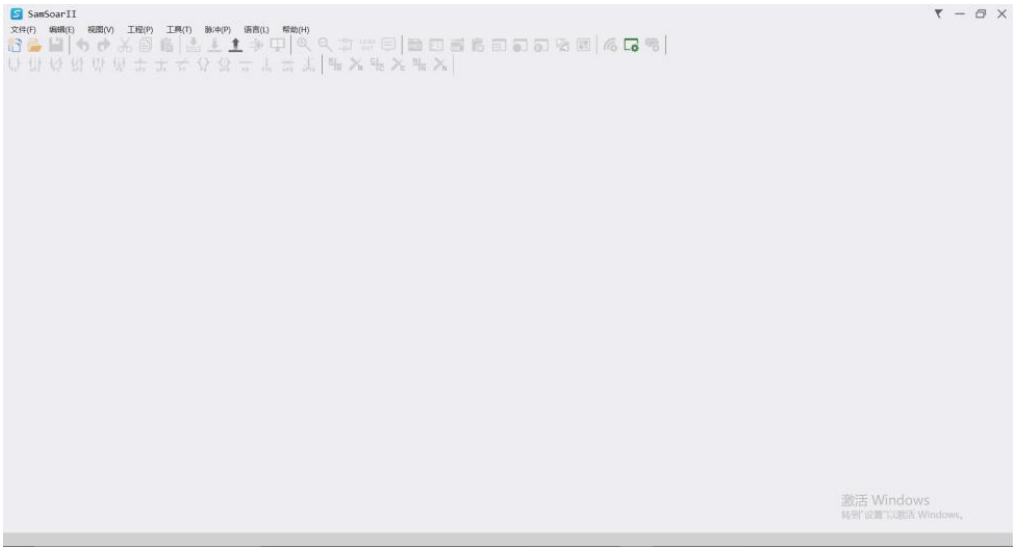


图 1.2 SamSoarII 运行界面

开始写新程序之前，必须新建一个新的或打开已存在的项目，这个项目包含必要的定义、少量配置参数。如果项目尚未存在，选择菜单栏中的“工程”-->“新建工程”，如图 1.3。



图 1.3

弹出新建工程定义画面如图 1.4。



图 1.4

点击下拉按钮选择 PLC 的型号，本项目选择 FGs_32MT_A。可以选择设置工程名以定义本工程的文件名称和存储位置，并点击“确定”按钮。

4.2 打开一个已存在的项目

打开一个已存在的项目，选择菜单栏中的“工程”-->“打开工程”，弹出打开缺省目录下的所有项目文件（*.ssr，*.ssp）如图 1.5 所示，

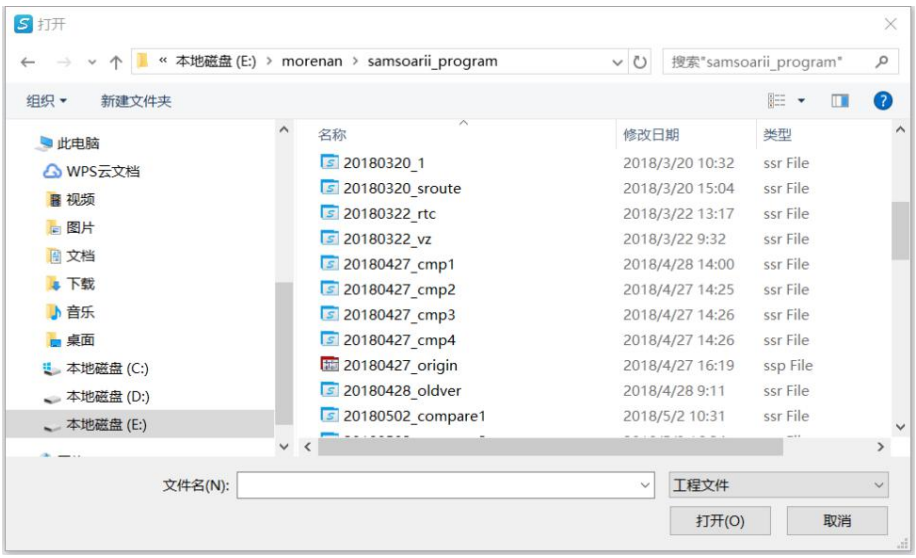


图 1.5

所有项目文件以列表的形式显示。双击列表内的所选项目文件名称，或者选择项目然后点击“打开”按钮。你还可以点击“工程”菜单下所列的项目文件名称，直接打开所选的项目文件。

注意，新版软件的项目文件以 ssr 结尾，而旧版软件以 ssp 结尾，为了向前兼容，新版软件支持旧版工程文件的转换并打开，转换工具可在菜单栏点击“工具”->“文件转换”来打开，如图 1.6 所示：

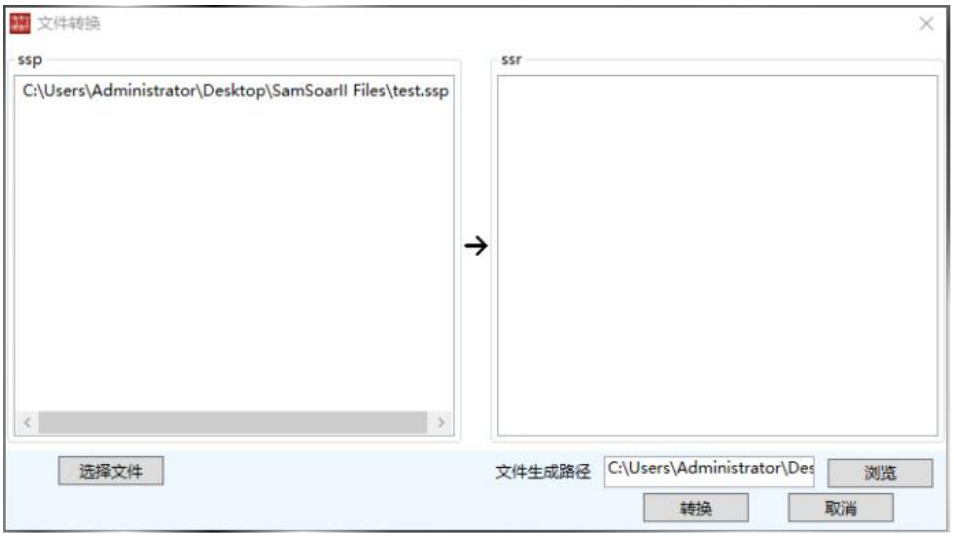


图 1.6

如果选择是“新建工程”，界面变成如下图 1.7：

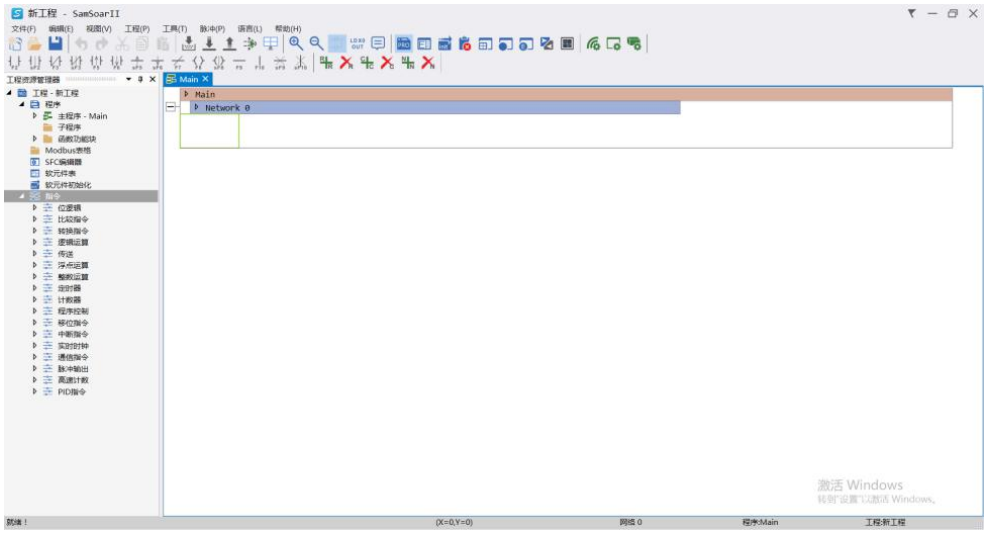


图 1.7

如果选择是“打开工程”，界面变成如下图 1.8：

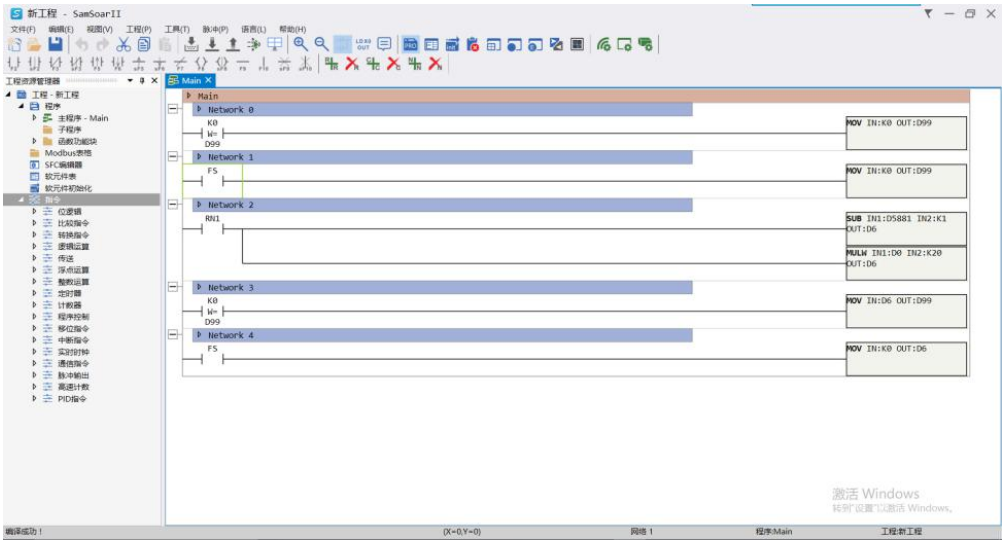


图 1.8

现在就可以编辑工程了，编写自己的工程梯形图。

5. 工程编辑

5.1 网络、行增加与删除

在编辑区可编辑位置右键菜单中可以选择网络插入、网络删除，行删除、行插入等选项，增加或者删除网络或行。



图 1.9

5.2 图元的添加和删除

在任一网络的可编辑位置左键点击选中图元，在左侧工程管理窗口中展开指令及其子项，如下图 2-2 示。双击某指令即能添加指令在选中位置。单击选中位置上的指令标识可以输入指令相关的参数。每个指令具体用法请参阅相关的帮助文档。



图 1.10

可以在选中位置，直接输入命令添加命令，比如”LD M0”，如下图所示。

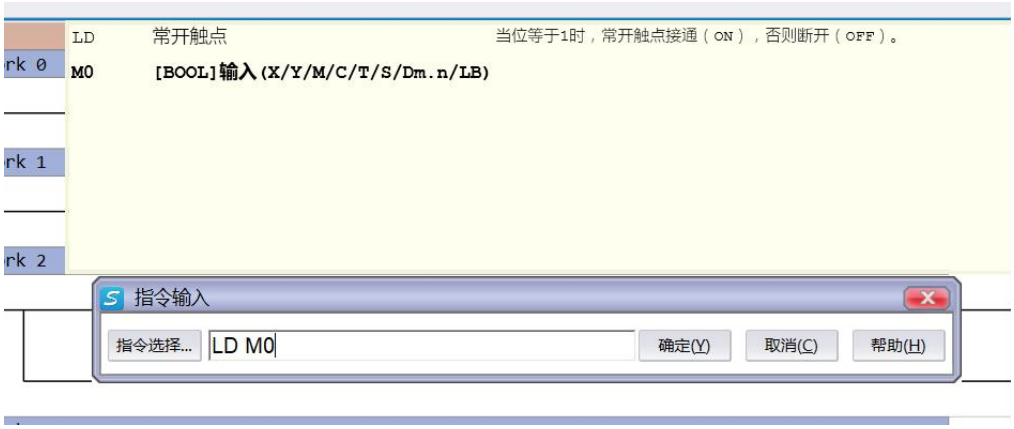


图 1.11

点击选中添加好的图元，按 F1 即能打开这条命令的帮助文档。按 Delete 键即可删除已添加好的图元。

5.3 查找与替换

用户可以通过菜单栏中的查找/替换功能，选择菜单栏中的“编辑”-->“查找/替换”，查找或者替换某个或多个软元件和指令。查看详情。

5.4 软元件初始化

软元件赋初始值，选择菜单栏中的“视图”-->“软元件初始化”，或者点击工具栏中的软元件初始化按钮，进入软元件初始化界面进行设置，查看详情。

5.5 软元件管理器

选择菜单栏中的“视图”-->“软元件管理器”，，或者点击工具栏中的软元件管理器按钮，可以查看软元件的使用状况，并且修改软元件的名称和注释。查看详情。

5.6 查看指令表

点击工具栏中的指令表模式，可以查看梯形图对应的 PLC 指令表。这样就能在主界面的上方显示梯形图，下方显示对应的指令表了。要想关闭梯形图界面只查看指令表，需要再点击工具栏中的梯形图模式按钮。

一般情况下，用户在修改梯形图之后不会立即更新指令表，可以通过点击界面来更新，或者在“系统设置”-->“其它”-->“自动更新 PLC 指令表”中设置来自动更新。

5.7 注释模式

点击工具栏中的注释模式按钮，可以在每个梯形图元件的下方显示每个软元件的注释。要注意注释的长度不能太长，超过显示范围的部分将会被忽略。

5.8 子程序的添加和使用

工程管理窗口中选择子程序的文件夹，右键菜单可以选择添加子程序以及文件夹。双击添加的子程序，就能在主界面中显示并进行编辑。

5.9 中断程序的使用

符合条件的子程序可以当作中断程序来使用。中断程序的相关命令有 4 个：ATCH（中断绑定）/DTCH（中断释放）/EI（中断允许）/DI（中断禁止）。

ATCH（中断绑定）指令将中断事件与中断子程序相关联，当中断事件发生的时候，执行相关联的中断子程序。

DTCH（中断释放）指令可以断开中断事件与中断子程序之间的关联，从而禁止单个中断事件。

中断允许 EI：指令全局性启用所有附加中断事件进程。要执行中断子程序，必须先开启全局中断。

中断禁止 DI：指令全局性禁止所有中断事件进程。

5.10 函数功能块的使用

工程管理窗口中选择函数功能块的文件夹，右键菜单可以选择添加或者删除函数功能块、导入导出程序和文件夹等功能。新版本的单个 C 语言函数块中可以编写多个函数，软件会自动识别所有实现的函数，合法的可以被 CALLM 指令调用。

6. 程序实例

梯形图如图 2.1：

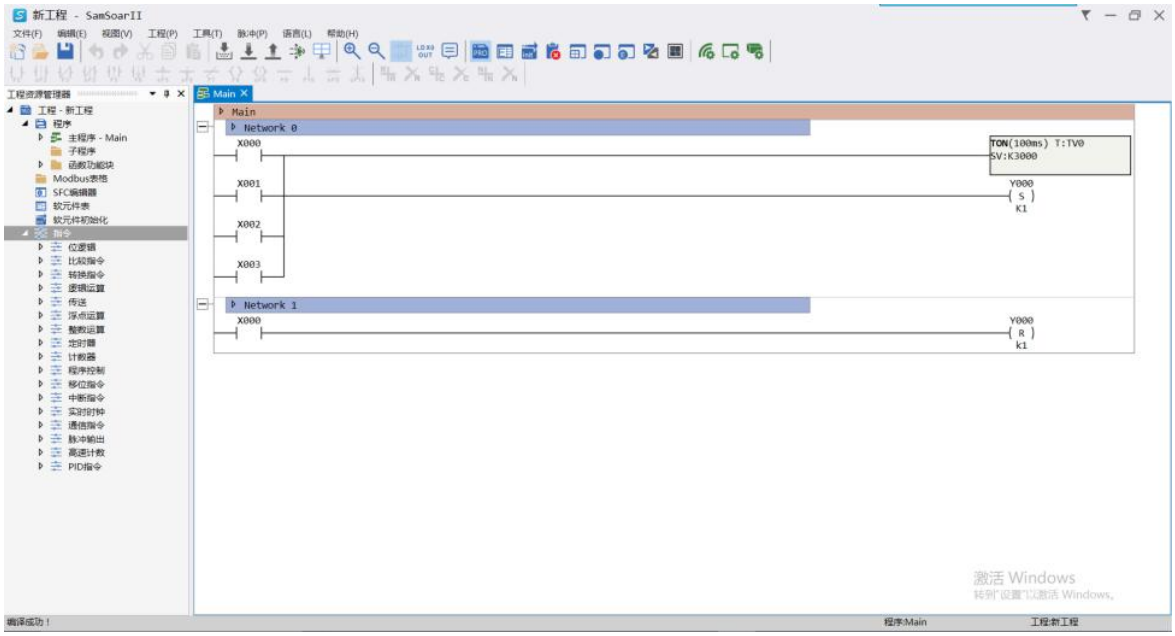


图 2.1

Network 000：启动定时器，线圈 Y000 置位。

点击展开左侧工程管理窗口中展开指令及其子项。

点击选中 Network 000 可编辑位置左侧第一个图元位置，选择并双击常开触点添加指令在选中位置，双击添加的图元，弹出梯形图输入对话框，输入 X000 并确认。

选择标准工具栏内“行插入”按钮，增加一行。点击选中第 1 行左侧第二个图元位置，在梯形图工具栏内选择“竖线”按钮添加一条竖线。

点击选中第 2 行左侧图元位置，选择并双击常开触点添加指令在选中位置，双击添加的图元，弹出梯形图输入对话框，输入 X001 并确认。

依次添加第 3、4 行，梯形图输入对话框分别输入 X002、X003 并确认。

点击选中第 1 行最右侧图元位置，选择并双击接通延时定时器 TON 添加指令在选中位置，双击添加的图元，弹出定时器输入对话框。

输入参数 T 的数据为 0，输入参数 SV 的数据为 K3000 并确认。

点击选中第 2 行最右侧图元位置，在梯形图工具栏内选择“竖线”按钮添加一条竖线。

选择并双击置位输出线圈添加指令在选中位置，双击添加的图元，弹出输出线圈输入对话框。

输入参数 BIT 的操作数为 Y000，输入参数 N 的数据为 K1 并确认。

Network 001：线圈 Y000 复位。

点击展开左侧工程管理窗口中展开指令及其子项。

点击选中 Network 001 可编辑位置左侧第一个图元位置，选择并双击常开触点添加指令在选中位置，双击添加的图元，弹出梯形图输入对话框，输入 T0 并确认。

点击选中最右侧图元位置，选择并双击复位输出线圈添加指令在选中位置，双击添加的图元，弹出输出线圈输入对话框。

输入参数 BIT 的操作数为 Y000，输入参数 N 的数据为 K1 并确认。

至此，梯形图工程文件编辑完成，选择标准工具栏内“保存工程”按钮，保存梯形图工程文件。项目存储之后，您可以下载程序到。

7. 工程下载及上载

7.1 工程下载

工程文件编辑完成后，就可以将工程文件下载到 PLC 中了。选择菜单栏中的“工具”-->“下载”，或者点击标准工具栏内的 PLC 下载图标，弹出对话框图 2.2。

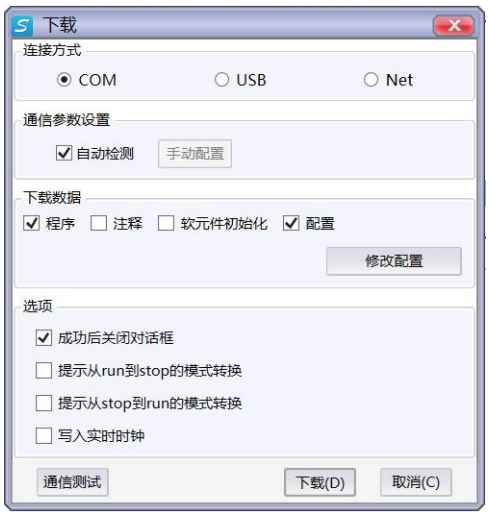


图 2.2

FGs 下载提供 com 口下载和 usb 下载两种方式。选择 Com 口下载需要设置通信参数，可以选择自动检测或者手动配置。

可以根据需要选择要下载的数据。必须勾选程序和配置信息的复选框，如果工程文件中选用了软元件初始化功能，必须勾选软元件表复选框。

开始下载前，可以点击“通信测试”进行通信测试，如果测试正常会提示通信成功。如图 2.3 所示。

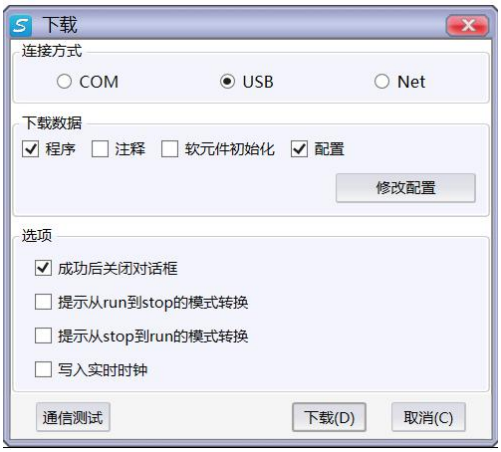


图 2.3

配置完成后，点击开始下载。按照提示操作就可以完成下载了。如果 PLC 参数设置不正确，点击“修改”按钮进入 PLC 参数设置对话框进行修改。

程序下载时 FGs 应置于 STOP 模式。程序下载后将 FGs 置于 RUN 状态，CPU 将执行程序。此时，如果按一下连接在 X0，X1，X2，X3 四个按钮中的任何一个，与 Y0 连接在一起的四个灯将会同时亮起来，五分钟之后自动关闭。

用户编辑的程序下载运行就算完成了。

7.2 工程上载

选择菜单栏中的“在线”-->“PLC 上载”，或者点击标准工具栏内的“PLC 上载”，打开上载对话框，如图 2.4。用户可以使用程序上载来上载已经下载到 PLC 中的程序。



图 2.4

PLC 上载与 PLC 下载相同。配置好通讯方式后，直接点击上载按钮，自动上载 PLC 中程序到 PC 并打开上载工程。

2.6 SamSoarII 的基本操作

1. 鼠标的用法

在编辑梯形图的时候，用鼠标点击【图片】的元件工具栏之后，之后将弹出梯形图输入对话框，如图 1.1 所示：



图 1.1 梯形图输入对话框

在该对话框中填入元件地址并点击下方的【确定】，即可将元件添加到梯形图光标所在处。

另外也可以通过双击树形目录处的命令或者直接键盘键入命令来编辑梯形图。

下面是对鼠标的几种常用的操作和意义

◎左键单击

按左侧的键，然后松开在本书中称为【单击】，在选择菜单，选择对象，选择工具按钮和确认设定时使用。

◎左键双击

连续快速的按左侧的健两次在本书中称为【双击】，在梯形图编辑模式、注释编辑模式、监控模式下鼠标在编辑区域双击已经存在的元件时是打开修改对话框，这时会对元件的信息进行修改、编辑注释或者修改监视的值。

◎右键单击

按右侧的键，然后松开在本书中称为【右击】，对某部件进行右击操作可以打开右键下拉菜单，或者在空白处右击操作显示右键菜单。还可以取消编辑状态，恢复鼠标形状。

2. 快捷键

下表 2-1 中罗列的是一些常用的快捷键，方便工程设计人员快速的进行工程画面的组态：

按键	功能
Enter	指令输入或修改
Delete	删除选定的元件
↑↓←→	梯形图光标移动
Ctrl+↑↓←→	梯形图光标移动并画线
Ctrl+鼠标滚轮向上	界面放大
Ctrl+鼠标滚轮向下	界面缩小
Ctrl+Enter	末尾添加行
Ctrl+Delete	删除当前行
Ctrl+X	剪切
Ctrl+C	复制
Ctrl+V	粘贴

Ctrl+Z	撤销
Ctrl+Y	恢复
Shift+↑↓←→	选择范围移动
Shift+Ctrl+↑↓←→	梯形图光标处推动
Shift+Enter	添加网络
Shift+Delete	删除网络
Shift+鼠标左键	梯形图多选
Shift+Esc	多选还原到单选光标
Ctrl+N	新建一个工程
Ctrl+O	打开一个工程
Ctrl+S	保存当前工程
Ctrl+F	查找
Ctrl+H	替换

表 1.1 快捷键组合

3. 用键盘代替鼠标

大部分的 Windows 软件，包括 SamSoarII，都带有一些通过特殊的系统按键，替代鼠标来点击操作顶部菜单栏和右键菜单。

1. 通过按下键盘的 Alt 键，将焦点转移到顶部菜单栏上，这个时候菜单项会有鼠标移上去时一样的深色标记，表示这个项被预选中，这个时候通过键盘的左右方向键可以移动这个预选标记，Enter 键或者下方向键可以展开当前菜单项的下拉子菜单。并且，括号内的关键字母会出现下划线。在键盘按下相应按钮也会移动到对应的项并展开子菜单。

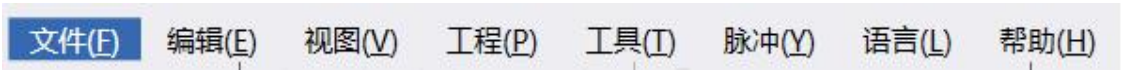


图 1.2

2. 展开子菜单后，同样也可以通过键盘上下方向键移动所选子项，键盘右键展开下一级菜单，按下 Enter 键或者对应的关键字母可以执行这个操作。这里举个实际的使用案例，按下 Alt 键时，顶部菜单出现深色标记表示正在操作状态，这时可以按下 F 键来展开文件菜单，按下 E 键展开编辑菜单。按下 F 键后，再按下 N 键就可以新建工程，按下 O 键就可以打开工程了。这样的话，就可以将 Alt,F,N 的按键组合视为新建工程的快捷键，Alt,F,O 视为打开工程的快捷键了。

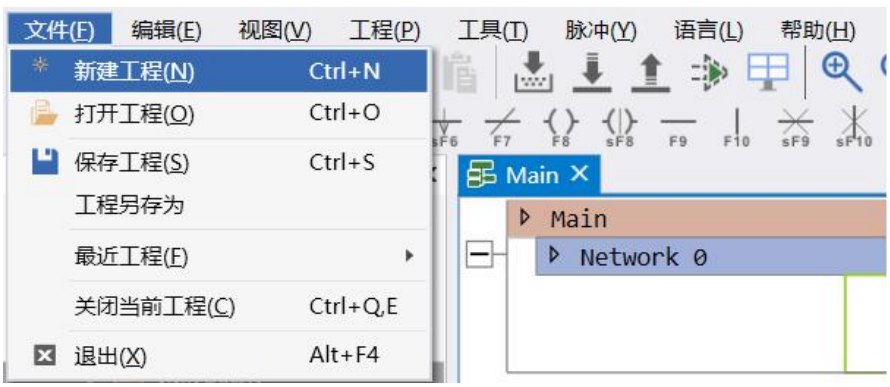


图 1.3

3.同理，编辑梯形图时，也可以按下键盘的菜单键（一般在右侧 Alt 键和 Ctrl 键之间，一个列表菜单形状的图标的键），这时会和点击鼠标右键一样弹出右键菜单，可以和上述操作一样来使用这个右键菜单。



图 1.4

4. 三键系统（Three Hot Key）

除了常用的快捷键外，本软件针对所有的菜单栏项都添加了三键系统。如图 1.2 所示:



图 1. 5

例如上载，可用快捷键 Ctrl+T,U 快速触发，具体操作如下，首先按下快捷键 Ctrl+T(手指按下后即可松开)，此时软件底部任务栏会出现如下提示，如图 1.2:



图 1.6

然后继续按下 U 键即可，若遗忘第二个键，可将鼠标移动到附近，此时会出现快捷键提示，如图 1.3 所示:

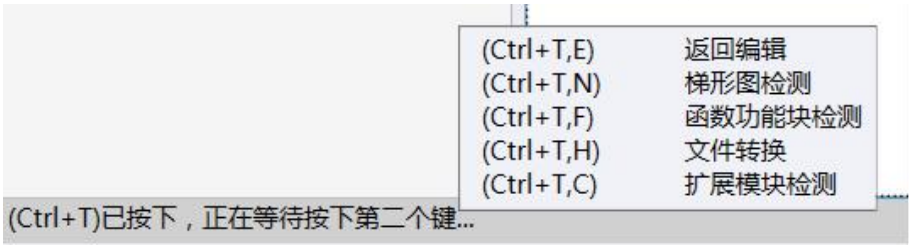


图 1.7

一般来说，以 Ctrl+T 开头的快捷三键都是功能性质的，方便熟悉。
以下是本软件支持的所有快捷三键:

按键	功能
视图菜单	
Ctrl+T,I	梯形图放大
Ctrl+T,O	梯形图缩小
Ctrl+E,C	注释模式
Ctrl+F1,O	打开工程资源管理器
Ctrl+F2,O	打开软元件管理器
Ctrl+F3,O	打开软元件初始化
Ctrl+F4,O	打开监视列表
Ctrl+F5,O	打开错误列表
Ctrl+F8,O	打开断点列表
Ctrl+F9,O	打开软元件断点列表
工程菜单	
Ctrl+W,R	添加子程序
Ctrl+W,F	添加函数功能块
Ctrl+W,M	添加 Modbus 表格
Ctrl+F7,O	打开属性窗口
工具菜单	
Ctrl+T,U	打开上载窗口
Ctrl+T,D	打开下载窗口
Ctrl+E,S	梯形图仿真
Ctrl+E,M	梯形图监视
Ctrl+T,E	返回监视模式
Ctrl+T,N	梯形图检测
Ctrl+T,F	函数功能块检测
Ctrl+T,H	打开文件转换窗口
Ctrl+T,C	扩展模块检测
Ctrl+F6,O	打开选项窗口
脉冲菜单	
Ctrl+P,O	打开脉冲平面配置窗口
Ctrl+P,I	打开图形导入窗口
语言菜单	
Ctrl+U,C	切换为中文

Ctrl+U,E	切换为英文
帮助菜单	
Ctrl+F10,O	打开帮助文档
程序编辑	
Ctrl+F1,A	向前插入行
Ctrl+F1,B	向后插入行
Ctrl+F2,A	向前插入列
Ctrl+F2,B	向后插入列
Ctrl+F3,B	向前插入网络
Ctrl+F3,E	末尾添加网络

表 1.2 三键快捷键

3. 编辑界面

SAMSOARII 编辑接口简洁友好，易学易用。本节介绍 SAMSOARII 的组态画面以及软件的各个功能按钮。

1、 SamSoarII 菜单栏的操作
2、 SamSoarII 工具栏的操作
3、 查找梯形图软元件以及函数功能块文本
4、 对软元件或文本进行替换
5、 打开并操作软元件管理器
6、 对软元件的初始化进行设置
7、 查看编译错误的报告
8、 对 PLC 进行监视
9、 PLC 参数的设置
10、SamSoarII 上位软件的设置
11、与 PLC 的通信设置

3.1 菜单栏

SamSoarII 菜单栏

SAMSOARII 编辑软件的菜单栏为使用者提供了丰富的菜单选项菜单栏，如下图 1.1 所示。

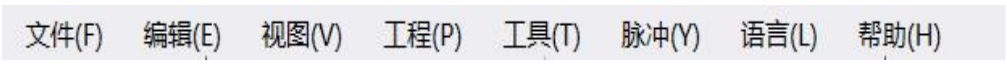


图 1.1 SamSoarII 菜单栏

1. 文件（F）

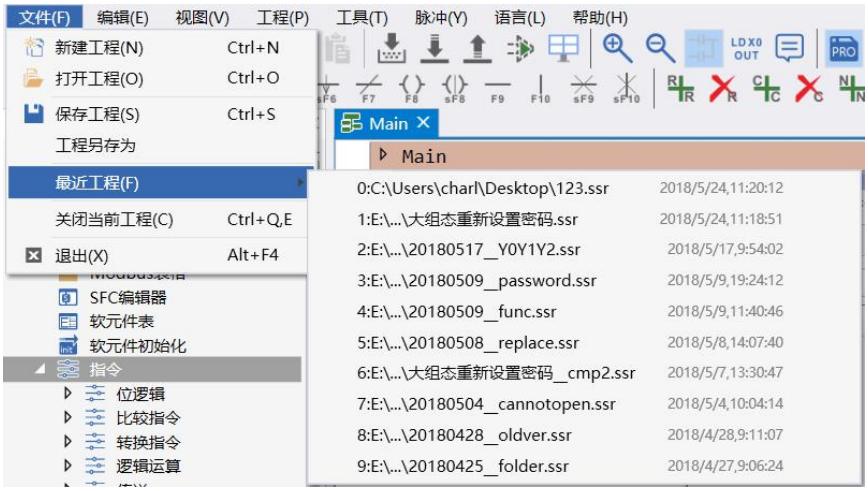


图 1.2 **文件菜单**

新建工程：用于新建一个新的 SAMSOARII 工程文件

打开工程：打开一个已经存在的 SAMSOARII 工程文件

保存工程：保存正在编辑的 SAMSOARII 工程文件

另存为：自己选择路径保存工程文件

最近工程：查看最近使用的工程文件的列表，选择其中一个打开

关闭当前工程：关闭当前正在编辑的 SAMSOARII 工程文件

退出：退出 SAMSOARII 编辑软件

2. 编辑（E）



图 1.3 编辑菜单

撤销：　　撤销刚才的操作步骤；注意,在执行以下操作之后，将不能进行撤销操作。

1. 进行梯形图检查，编译，仿真，监视都会进行一次检查。
2. 切换 PLC 型号。
3. 保存退出再打开。

恢复： 恢复刚才已经撤销的操作步骤

剪切： 对于选中的目标执行剪切操作

复制： 对于选中的目标执行复制操作

粘贴： 在光标的指定位置粘贴剪切板中的内容

全选： 全选当前的梯形图或者文本。

查找： 查找梯形图中的指令或者软元件，以及文本中的关键字。

替换： 替换梯形图中的指令或者软元件，以及文本中的关键字。

3. 视图（V）



图 1.4 视图菜单

放大：将当前梯形图的尺寸变大。

缩小：将当前梯形图的尺寸变小。

注释模式：显示软元件的注释信息。

工程资源管理器：查看并管理当前工程各个部分的窗口

软元件管理器：查看或编辑软元件信息的窗口

软元件初始化：设置在 PLC 运行之前，各个软元件要预设的值。

错误列表：显示梯形图和函数块检查后，检查到的所有错误以及警告。

监视列表：通过该对话框进行软元件变量的监视以及对软元件的值进行修改，注意，只有切换到梯形图监视模式的情况下才允许执行软元件的变量监视。

断点列表：**仅在仿真模式下使用。**在梯形图指令或者函数块语句中设置断点，仿真程序运行到该位置会自动暂停，以便于查看运行的即时情况。

软元件断点：**仅在仿真模式下使用。**对软元件设置条件表达式，当符合这个条件时仿真程序会自动暂停，并标出当前运行到的位置。

4. 工程（P）



图 1.5 工程菜单

添加子程序：单击这个按钮后，会在“工程资源管理器”-->“工程”-->“程序”-->“子程序”的文件夹中创建一个新的子程序，默认名称叫“新建子程序”，若存在重名则会在后面加入代号。添加的子程序的名称可以在编辑模式下修改。

添加函数功能块：在工程资源管理器中创建一个新的函数功能块。

添加 MODBUS 表格：将 MODBUS 编辑界面显示在当前主界面上，并且弹出创建 MODBUS 表格的对话框。

属性：弹出 PLC 设置的对话框，在此对话框内可以设置设备型号，通信参数，通信密码，模拟量，滤波参数，掉电保持以及扩展模块。

5. 工具（T）



图 1.6 工具菜单

上载：弹出 PLC 读取对话框，在对话框中可以实现对 plc 内部数据的上载操作。

下载：弹出 PLC 下载对话框，在对话中实现将编辑的梯形图下载到 PLC 中。

仿真：单击这个按钮后，会进入仿真模式，在仿真模式下会模拟工程程序的运行。**要注意，仿真不支持脉冲，通信以及高速计数。**

监视：单击这个按钮后，会进入监视模式，此时可以进行软元件变量的监视以及对软元件的值进行修改。

返回编辑：在仿真模式和监视模式下使用，单击以返回到编辑模式。

编译：对整个工程进行合法性检查，确保各个部分的正确。编译时发现的错误信息会报告到错误列表中。

程序检测：对梯形图和函数块进行检测，确保其正确性。

文件转换：将一代 SAMSOAR Developer 的工程文件，转换为二代 SAMSOARII 的工程文件。

扩展模块检测：检测连接 PLC 的扩展模块数量及信息。

PLC 恢复出厂：清除 PLC 里的工程程序和数据，将 PLC 恢复到刚出厂的状态。

选项：在该对话框内对梯形图的编程界面以及风格等进行相应的设置，使得软件使用更符合用户自己编程习惯。

6. 脉冲（Y）



图 1.7 脉冲菜单

脉冲平面：弹出对平面系统进行设置的对话框进行操作。
图形导入：弹出对 HMI 图形的管理的对话框。可以进行导入，参数设置等操作。

7. 语言（L）

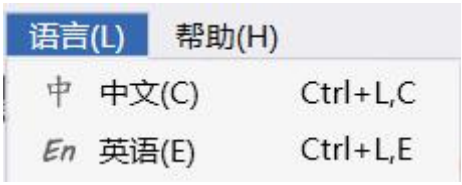


图 1.8 语言菜单

中文：将当前语言切换为简体中文。
英文：将当前语言切换为 English.

8. 帮助（H）

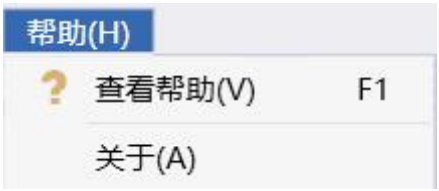


图 1.9 帮助菜单

帮助：点击该按钮可以调处帮助文档查看。
关于：点击该按钮可以查看软件的版本号以及版权归属。

3.2 工具栏

SamSoarII 工具栏

2.1 标准工具栏

标准工具栏中包含了针对 plc 工程项目的一些基本操作的按钮，如新建、打开、保存以及对于梯形图编辑过程中用到的剪切、拷贝、黏贴等基本操作。



图 1.1 标准工具栏


新建工程：用于新建一个 SAMSOARII 工程文件


打开工程：打开一个已经存在的 SAMSOARII 工程文件


保存工程：保存正在编辑的 SAMSOARII 工程文件


撤销：撤销刚才的操作步骤；

 恢复：恢复刚才已经撤销的操作步骤

 剪切：对于选中的目标执行剪切操作

 复制：对于选中的目标执行复制操作

 粘贴：在光标的指定位置粘贴剪切板中的内容

 编译：检查当前工程是否合法，包括梯形图和函数功能块。

 上载：弹出 PLC 读取对话框，在对话框中可以实现对 plc 内部数据的上载操作。

 下载：弹出 PLC 下载对话框，在对话中实现将编辑的梯形图下载到 PLC 中。

 仿真：单击这个按钮后，会进入仿真模式，在仿真模式下会模拟工程程序的运行。

 监视：单击这个按钮后，会进入监视模式，此时可以进行软元件变量的监视以及对软元件的值进行修改。

 放大：将当前界面放大显示。

 缩小：将当前界面缩小显示。

 梯形图显示模式：此按钮处于选定状态（深色）下，在主界面显示梯形图。当指令表模式未被选定时，不能取消梯形图模式。

 指令表显示模式：此按钮处于选定状态（深色）下，在主界面显示指令表。当梯形图模式未被选定时，不能取消指令表模式。当梯形图和指令表同时显示时，梯形图位于主界面的上方，指令表在下方，可以拖动边界来调整二者的高度比。

 注释编辑模式：此按钮处于选定状态（深色）下，会在梯形图元件下方和指令表指令的右方显示每个软元件的注释。

 工程资源管理器：管理工程目录结构，打开项目并编辑的管理器窗口。

 软元件管理器：查看或编辑软元件信息的窗口。

 软元件初始化：设置在 PLC 运行之前，各个软元件要预设的值。

 错误列表：每次检查后报告错误的窗口。

 监视列表：监视软元件的值的变化的窗口。

 断点列表：**仅在仿真模式下使用。**在程序的运行位置处注册断点后，程序运行到此处会自动暂停，并将控制权交给用户。

 软元件断点：**仅在仿真模式下使用。**注册对某个软元件的中断条件的软元件断点，中断条件符合时会自动暂停，并将控制权交给用户。

 COM 设置：对通信参数进行设置。可以设置或者自动检测串口的波特率，校验位，停止位，超时时间，并进行对下位的通信测试。

 SYS 设置：对软件系统进行设置。包括界面风格的设置以及其他设置，以符合用户的编程习惯。

 PLC 设置：弹出 PLC 设置的对话框，在此对话框内可以设置设备型号，通信参数，通信密码，模拟量，滤波参数，掉电保持以及扩展模块。

2.2 编辑工具栏

该工具栏中包含编辑梯形图时经常用到的基本元件和操作，通过单击该工具栏上的按钮可以快速进行梯形图符号的输入。



图 1.2 编辑工具栏

常开触点：点击这个按钮，会弹出梯形图命令输入对话框，可以快速插入一个常开触点，快捷键为（F2）

立即常开：快速插入立即常开触点，快捷键为（Shift+F2）

常闭：快速插入常闭触点，快捷键为（F3）

立即常闭：快速插入立即常闭触点，快捷键为（Shift+F3）

上升沿脉冲：快速插入上升沿触点，快捷键为（F5）

下降沿脉冲：快速插入下降沿触点，快捷键为（F6）

运算结果上升沿化：前面的运算结果取上升沿，快捷键为（Shift+F5）

运算结果下降沿化：前面的运算结果取下降沿，快捷键为（Shift+F6）

运算结果取反：前面的运算结果取反，快捷键为（F7）

输出线圈：快速插入输出线圈，快捷键为（F8）

立即输出线圈：快速插入立即输出线圈，快捷键为（Shift+F8）

横线：绘制一条横向连接线，快捷键为（F9）

竖线：绘制一条竖向连接线，快捷键为（F10）

横线删除：删除点中的横线，快捷键为（Shift+F9）

竖线删除：删除点中的竖线，快捷键为（Shift+F10）

行插入：在当前光标处插入一条新行。

行删除：删除当前选择范围内的所有行。

列插入：在当前光标处插入一条新列。

列删除：删除当前选择范围内的所有列。

网络插入：在当前光标处插入新的网络。

网络删除：删除与选择范围关联的所有网络。

3.3 查找

查找操作

1. 对梯形图的查找

当前正在查看并编辑梯形图时，选择菜单栏中的“编辑”-->“查找”按钮，或者按下 Ctrl+F，打开查找窗口，如图 1.1。

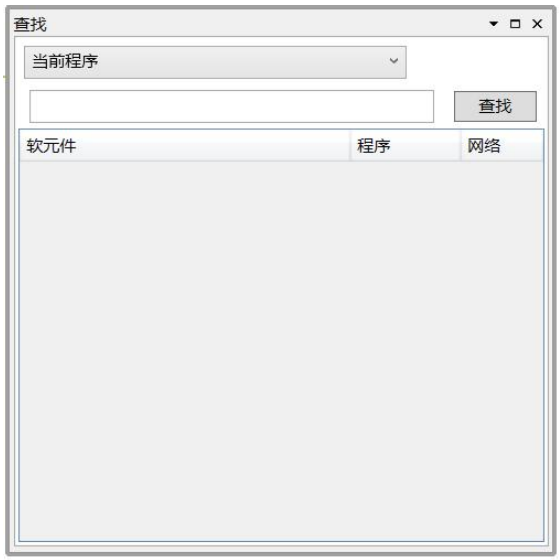


图 1.1



选择查找的程序范围，当前程序是指当前显示的并正在编辑的程序，所有程序是指工程文件内的主程序和所有子程序。

在下方的文本框内输入要查找的内容并按下 

或者回车键，会在下方的表格中显示所有符合条件的指令项。输入非法时文本框背景会变红，输入合法待查找时显示绿色，查找后恢复白色。可以按照以下规则来输入：

1.查找单个指令名称：例如 LD, OUT, ADD。

2.查找单个软元件名称：例如 D0, M0V0, K1。

3.查找选定范围内的软元件：例如 D[0..2]，将会找到所有使用了 D0,D1,D2 的指令。同样变址和常量也可以选择范围。例如 M10V[1..3]代表 M10V1,M10V2,M10V3，M[10..100]V1 代表 M10V0 到 M100V0，也可以同时选择二者的范围 M[10..100]V[1..3]。K[1.5..4.5]会查找在 K1.5 到 K4.5 范围内的所有常量，包括整数和十六进制数。位合成字和字取位也符合以上规则。

4.查找子程序，C 函数，MODBUS 表格的名称。

5.以上元素的任意组合，以组成指令的架构。例如 LD M0，ADD K[0..10] K[10..16]M[10..20] D[20..30]等。可以使用*符号来对任意项匹配，例如 LD *同样可以匹配 LD M0,LD C0, LD D0.0 等指令，ADD * * D0 可以只匹配符合 out 参数的指令。

在输入关键字并按下回车后，若匹配到符合条件的指令，会在下方表格中显示。

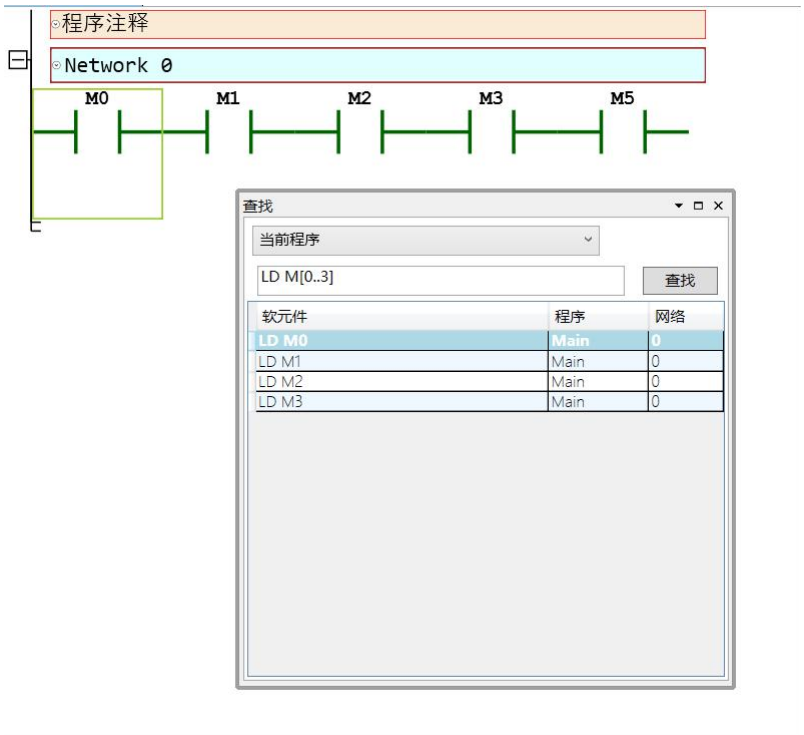


图 1.2

在表格中双击其中一项，梯形图会自动导航到选定的位置。

2. 对文本的查找

查找功能同样可以对函数功能块文本进行查找。在上方可以选择查找的范围为当前文本或者全部文本。输入关键词并按下回车，即可在下方列表查看找到的所有项，双击以选中匹配的文本内容。

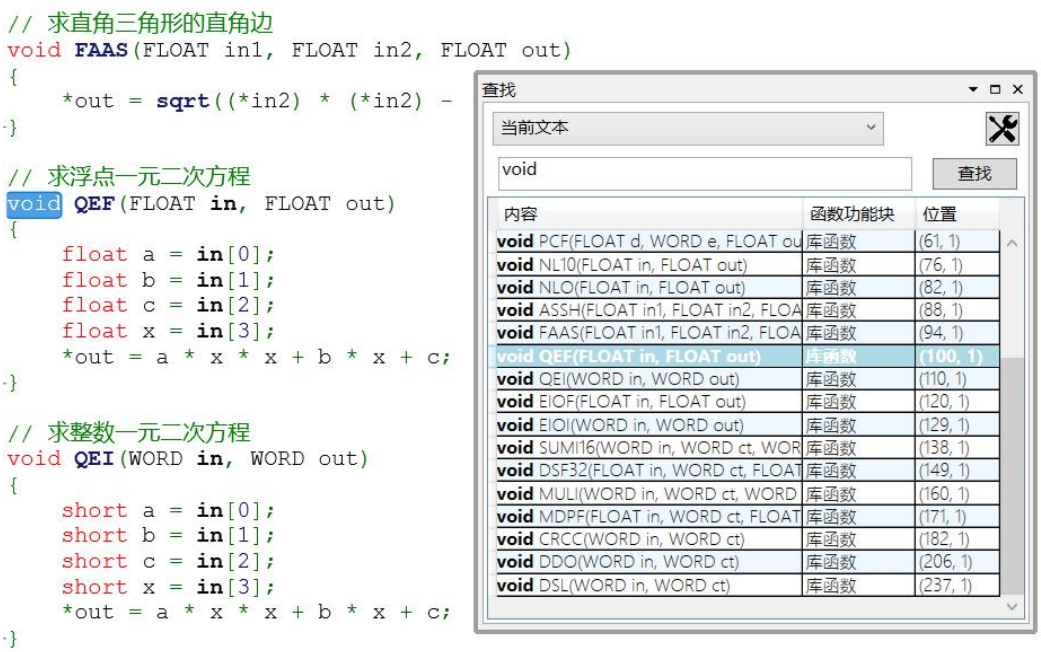


图 1.3

点击右上角的设置按钮，来进入文本查找的设置。

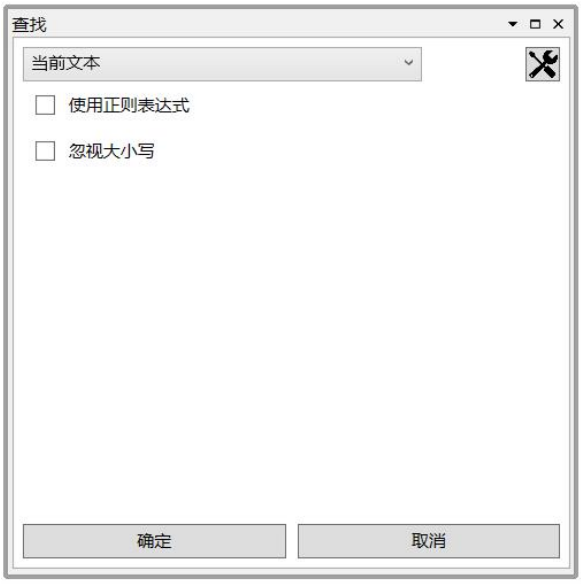


图 1.4 设置窗口

使用正则表达式：允许用户输入符合正则表达式规则的关键字格式。如何使用正则表达式？
忽视大小写：查找时无视大小写，输入 a 的话 a 和 A 都会被匹配到。

3.4 替换

替换操作

1. 对梯形图的替换

当前正在查看并编辑梯形图时，选择菜单栏中的“编辑”-->“替换”按钮，或者按下 Ctrl+R，打开替换窗口。



图 1.1



选择查找的程序范围，当前程序是指当前显示的并正在编辑的程序，所有程序是指工程文件内的主程序和所有子程序。

在查找文本框内输入要查找的内容并按下 

（或者回车键），会在下方的表格中显示所有符合条件的指令项。输入非法时文本框背景会变红，输入合法待查找时显示绿色，查找后恢复白色。如何输入查找关键字？

在替换文本框内输入要将查找到的指令进行替换的关键字，然后进行后续替换。替换关键字的输入规则和查找一致，但查找和替换的关键字之间存在配对规则：

1. 一般情况下，替换关键字只替换**明确**的部分，例如 ADD D0 * D[10..20]，如果将 SUB K2 K1 D5 进行替换，那么会替换为 ADD D0 K1 D10，因为只有 ADD，D0 和 D[10..20]是明确的，并且 D[10..20]将视作 D10，而中间的 K1 将会被保留。

2. 如果查找只输入了单个软元件，那么替换也可以只输入单个软元件，来仅对软元件进行替换。例如查找输入了 D0，替换输入 D1，若要替换 LD D0.8, MUL K1 D0 D1，那么会被替换为 LD D1.8 和 MUL K1 D1 D1，只进行了 **D0 ---> D1** 的替换而保留了其他部分。**在替换输入的软元件名称前面添加$符号，表示强制替换为所选的软元件，不考虑原来的软元件的变址，字取位，位合成字的情况。**例如查找输入了 D0，替换输入$D1，那么 MOV K10 D0V0 会被替换为 MOV K10 D1，变址部分会被省略。相比较而言，如果替换输入的是 D1，那么还是会被替换为 MOV K10 D1V0，变址部分会被保留。

3. 如果查找输入了软元件的范围，例如 M[0..2]，那么替换也可以输入软元件的范围，例如 C[10..12]，这样就会进行 **M0-->C10，M1-->C11，M2-->C12** 的替换。

4. 若查找输入了指令格式，替换也输入了指令格式，那么会对每个对应位置的参数进行一一匹配，若符合上述的范围匹配则按照上述规则执行。例如 ADD D0 D[0..2] D10 替换为 SUB D0 CV[10..12] D20，那么中间的参数会进行 **D0-->CV10，D1-->CV11，D2-->CV12** 的替换。

对查找到的指令进行替换可以通过以下三种方式来进行：

1. 双击表格项，对单个指令进行替换。

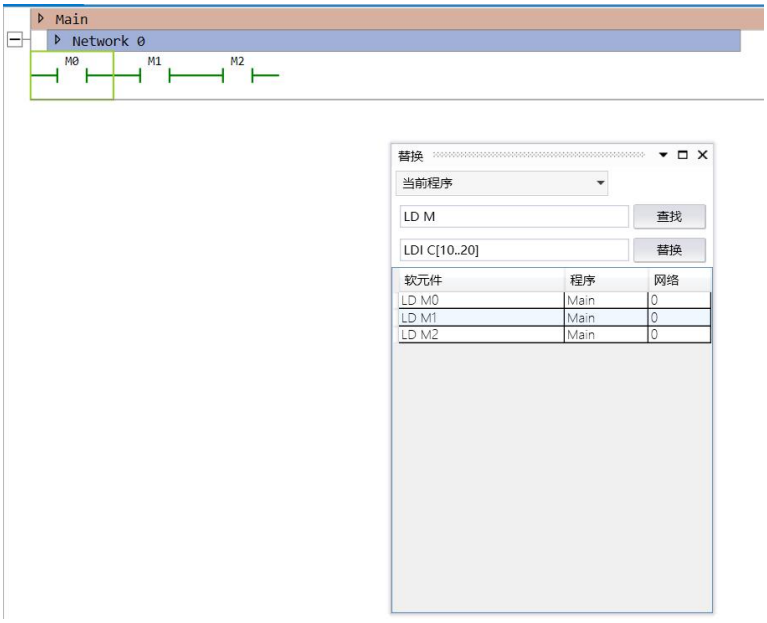


图 1.2

2. 通过鼠标拖动来范围选择，Shift+鼠标左键来调整选择范围，Ctrl+鼠标左键来添加或者移除项的选择。选择好范围后，按下回车键对范围内的所有指令进行替换。



图 1.3

如果替换不成功，会将错误信息报告到对话框中。

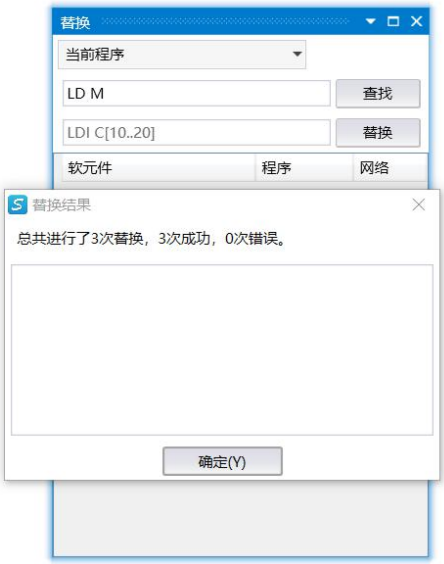


图 1.4

3. 未选中下方表格中的任意项，直接点击替换按钮，会将下表中所有项进行替换。

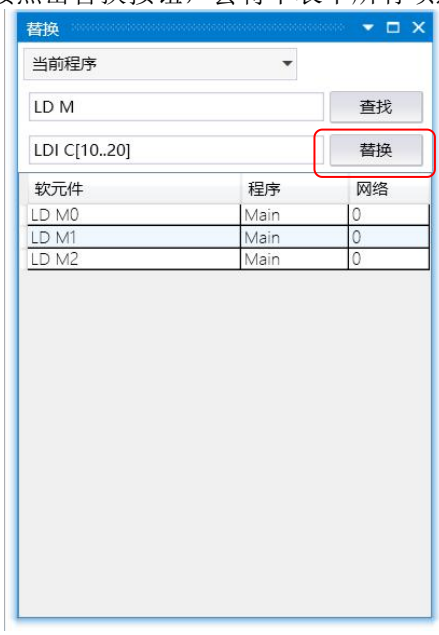


图 1.5

替换报告对话框弹出时，替换的图元部分会在屏幕上暂时消失，这属于正常现象，关闭对话框后会替换成新的图元。

替换后可以在替换窗口点击  进行撤销，或者点击  进行恢复操作。**要注意，之后对梯形图直接修改会清除掉替换窗口的撤销记录，请谨慎修改。**

2. 对文本的替换

当前正在编辑文本时，选择菜单栏中的“编辑”-->“替换”按钮，或者按下 Ctrl+R，打开替换窗口。在查找输入框内输入关键字来查找，在替换输入框内输入关键字来替换。

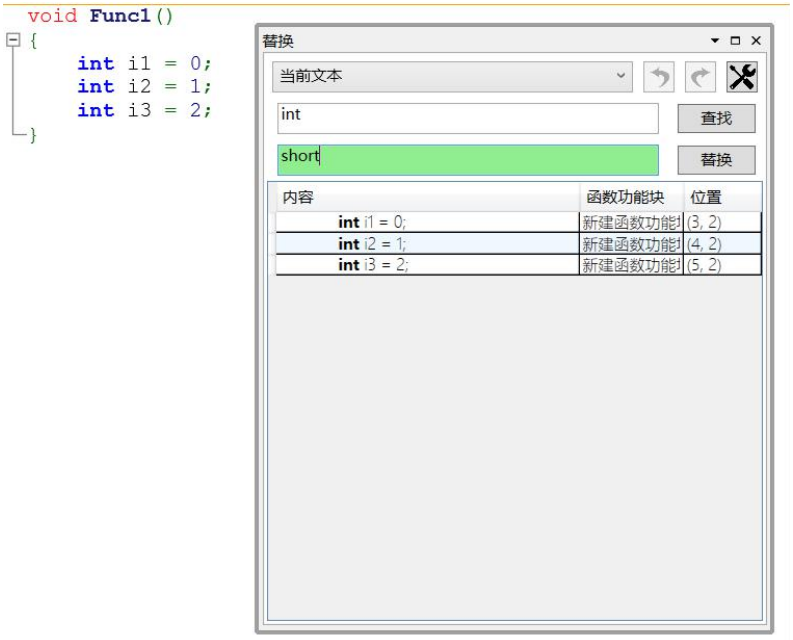


图 1.5

3.5 软元件管理器

选择菜单栏中的“视图”-->“软元件管理器”，或者按下工具栏的软元件管理器按钮，打开软元件管理器窗口。



图 1.1

选择软元件区间：可以将只符合某个基地址的软元件在表格中显示，点击以选中，再次点击选中的以取消。

搜索：输入软元件名称并按回车键来搜索符合的软元件。

显示所选项细节：这个按钮处于按下状态时，选中的表格项的下方会显示使用这个软元件的指令列表，单击其中的指令，梯形图的界面可以导航到此处。

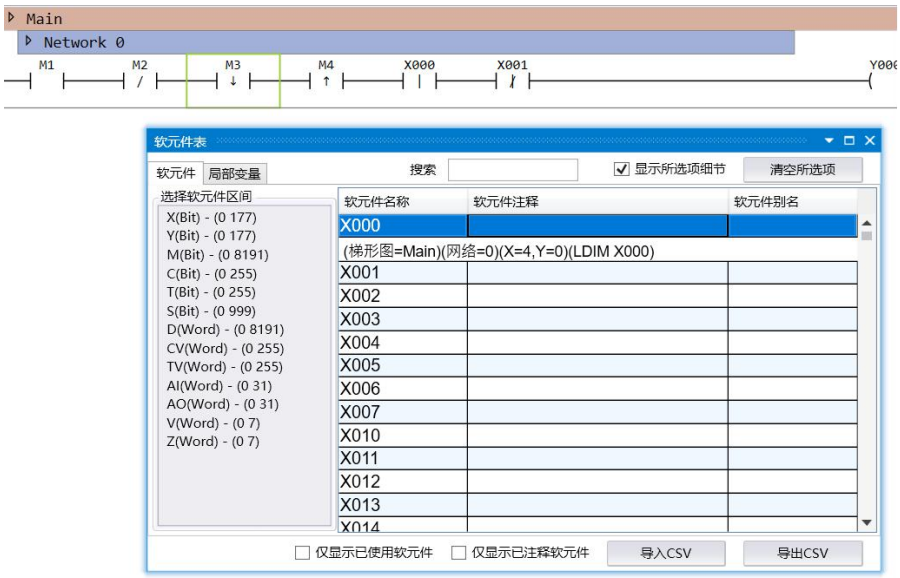


图 1.2

清空所选项：取消对表格项的选择，从单选或者多选转换到未选择的状态。

已使用软元件：这个按钮处于按下状态时，只会显示梯形图程序用到的软元件。

已注释软元件：这个按钮处于按下状态时，只会显示拥有非空白注释的软元件。

导入 CSV：从 CSV 文件中的软元件信息导入到工程内。

导出 CSV：将工程内的软元件信息导出到 CSV 文件。

3.6 软元件初始化

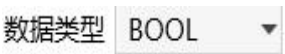
选择菜单栏中的“视图”-->“软元件初始化”，或者按下工具栏的软元件初始化按钮，打开软元件初始化窗口。



图 1.1



输入要添加的软元件的地址。



选择要添加的数据类型。



输入要批量添加的软元件的数量，若输入 10，会将 D0~D9 依次添加到表格中，**若数据类型选择 INT32,POS_INT32 或者 FLOAT，再输入 10，会将 D0,D2,D4 一直到 D18 依次添加到表格中。但是 CV200 以上的双字计数器还会保持一个间隔，即 CV200,CV201...CV209。**

添加要初始化的软元件之后，可以在列表中对应的行项双击【ON/OFF/值】列项，会出现输入框来输入要初始化的值。

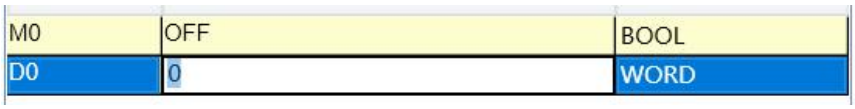


图 1.2 双击之后的输入框

应遵守以下规则来输入：
1.对于类型为 BOOL 的软元件，可以输入 ON,on 或者 1 来代表置位，输入 OFF,off 或者 0 来代表复位。
2.对于其他类型的软元件，输入的值应符合当前类型的范围。

类型	范围
INT16	[-32768~32767]之间的整数
POS_INT16	[0~65535]之间的整数
INT32	[-2147483648~2147483647]之间的整数
POS_INT32	[0~4294967295]之间的整数
BCD	[0~9999]之间的整数
FLOAT	[-3.4e38~3.4e38]之间的实数

表 1.1 类型的范围

1. 导出和导入 CSV

选中表格中要导出的软元件初始化信息（单击以选中，Shift+单击以连选，Ctrl+单击以多选），单击“导出 CSV”按钮，就可以将这些初始化值导出到本地文件。

要导入上次导出的文件时，单击“导入 CSV”按钮，就可以选择本地 CSV 文件导入。

3.7 编译错误表

错误列表

工程编辑完毕后，可以通过编译来验证工程程序的正确性。通过 F4 键来进行编译，编译包括梯形图检查以及函数块检查这两个步骤。

在梯形图检查或者函数块检查中，发现的错误信息会报告到错误列表中，这时错误列表的窗口会自动弹出。也可以通过选择菜单栏中的“视图”-->“错误列表”来打开窗口。

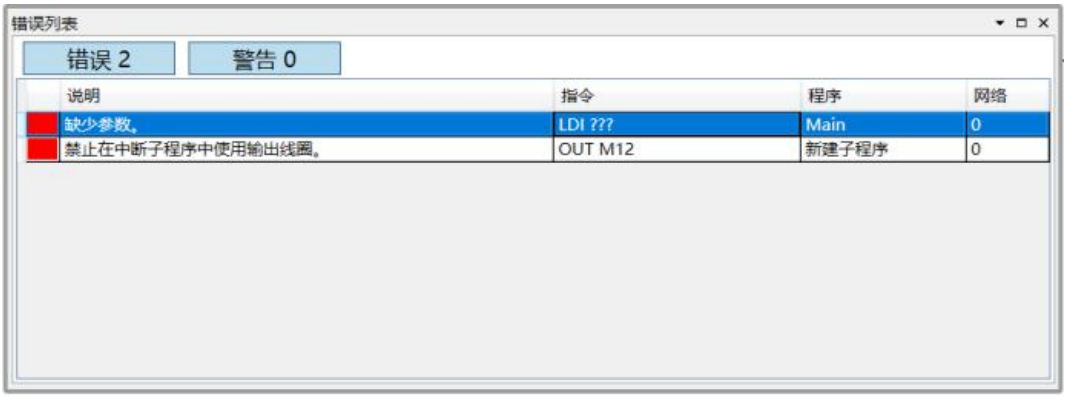


图 1.1

列表中的每个错误项都有相应的说明，双击该项以导航到梯形图对应的位置，将错误更改后这个项就会消失。

函数块的错误通过将编译器给的报错信息，经过整理显示在列表中。

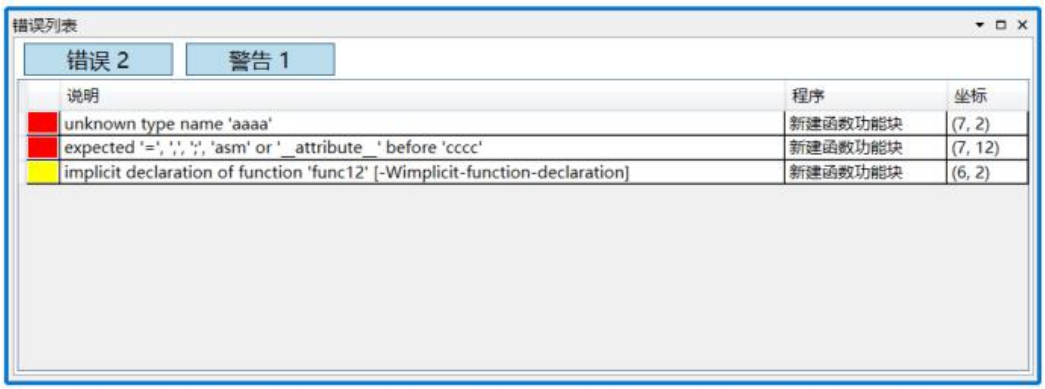


图 1.2

和梯形图错误一样，双击会导航到对应页面，但是更改的过程中错误信息不会消失，必须全部更改完毕后再一次进行函数块检查才可通过。

3.8 监视

变量监视表

选择菜单栏中的“视图”-->“监视列表”，监视变量对话框将会打开。

注意：只有在监视模式或者仿真模式下才能在该对话框进行元件监视，“监视开始”按钮才会变为有效状态。

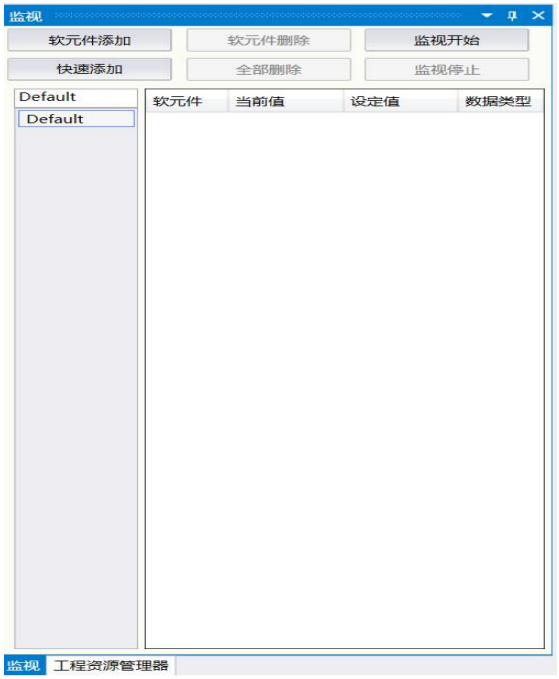


图 1.1 变量监视表

1. 监视变量表的添加和删除

左侧列表为监视变量表的名称，左键列表名可切换监视变量表,右键列表框可对变量表进行增减(最大可增加 13 个表格)，每一个表下可添加不同监视变量,在同一时刻只可监视其中一个表格内的变量。

右键打开菜单，可以选择新建或者删除变量表。



图 1.2 表格编辑

点击【新建表格】会在最下方新建一个新的变量表。默认名称为 New，重名会在后面标序号。



此处显示选中的监视变量表名称,修改此处内容可修改对应表格的名称，点击别处修改的名称会立即生效。

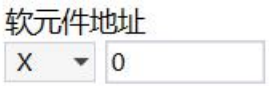
点击【删除表格】会删除选中的变量表。**点击【全部删除】后，之前的监视变量表以及里面的内容均会被删除，请谨慎操作。**

2. 添加监视元件

单击“软元件添加”按钮可以弹出软元件添加对话框：



图 1.3 添加软元件



在此处可以输入软元件的地址。



在此处可以选择软元件的数据类型。

要注意，位寄存器（X,Y,M,S,T,C）若选择字类型视为位组成字，同理字寄存器选择 BOOL 视为字取位。在输入软元件地址的下方会要求输入位组成字的长度或者字取位的位置。

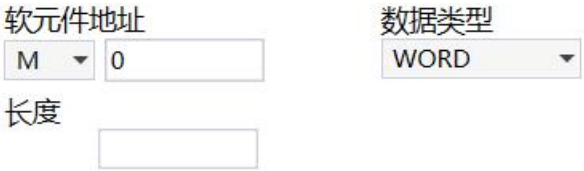


图 1.4 添加位合成字

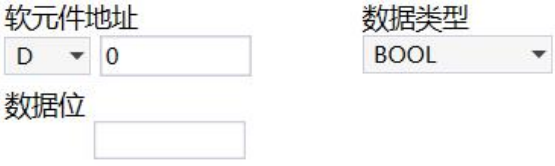


图 1.5 添加字取位

勾选☑变址后，会在右侧显示 偏移元件选择 V 0 ，这时输入的软元件带变址偏移。

勾选☑批量后，会在右侧显示 添加个数 1 (1 ~ 64) ，输入个数后会批量添加软元件。**要注意，若单字寄存器(D,V,Z,AI,AO,TV,CV0~199)选择了双字的类型(DWORD,UDWORD, DHEX,FLOAT)，则会间隔为 2 批量添加。**

3. 快速添加

单击【快速添加】按钮会弹出快速添加对话框。

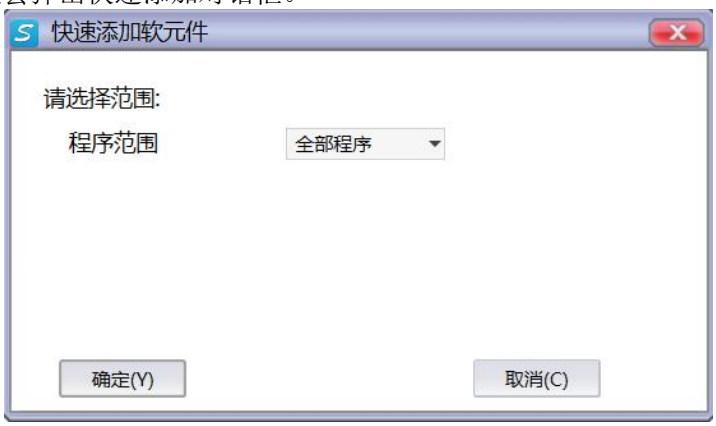


图 1.6 快速添加

这个添加模式下，会识别出所有用到的软元件，按照使用的信息（数据类型，变址，字取位，位合成字）自动添加到监视列表中。



软元件	当前值	设定值	数据类型
Y000	???		BOOL
Y001	???		BOOL
Y002	???		BOOL
Y003	???		BOOL
Y004	???		BOOL
Y005	???		BOOL
M0	???		BOOL
M1	???		BOOL
M2	???		BOOL
M3	???		BOOL
M4	???		BOOL
M5	???		BOOL
CLK_100ms	???		BOOL
D0	???		WORD
D1	???		WORD
D2	???		WORD
D3	???		WORD
D4	???		WORD
D5	???		WORD
D10	???		FLOAT
D12	???		FLOAT
D14	???		FLOAT
D16	???		FLOAT
D18	???		FLOAT
D20	???		FLOAT

图 1.7

在 程序范围 全部程序 选择了【全部程序】后，点击确定按钮，每个程序都会分配给一个对应的监视变量表中，以程序的名称来命名（主程序设为 Main）。

要注意，进行快速添加后，之前的监视变量表以及里面的内容均会被删除，请谨慎操作。

若选择【当前程序】，会要求在下方给定网络范围。这样只会创建一个监视变量表，表内只包含网络范围内的软元件。



图 1.8

4. 删除监视元件

在监视表中选中项后，【软元件删除】按钮可以点击。这时会删除掉所选择的软元件。

点击【全部删除】后，当前的监视变量表里面的内容均会被删除，请谨慎操作。

5. 监视开始和停止

在监视模式和仿真模式下，可以打开或者关闭监视。在仿真模式下打开监视程序也会立即运行，关闭监视后程序会立即停止。开始监视后，就能在梯形图和监视表中查看软元件的当前值了。这时点击监视停止，会立即停止监视。

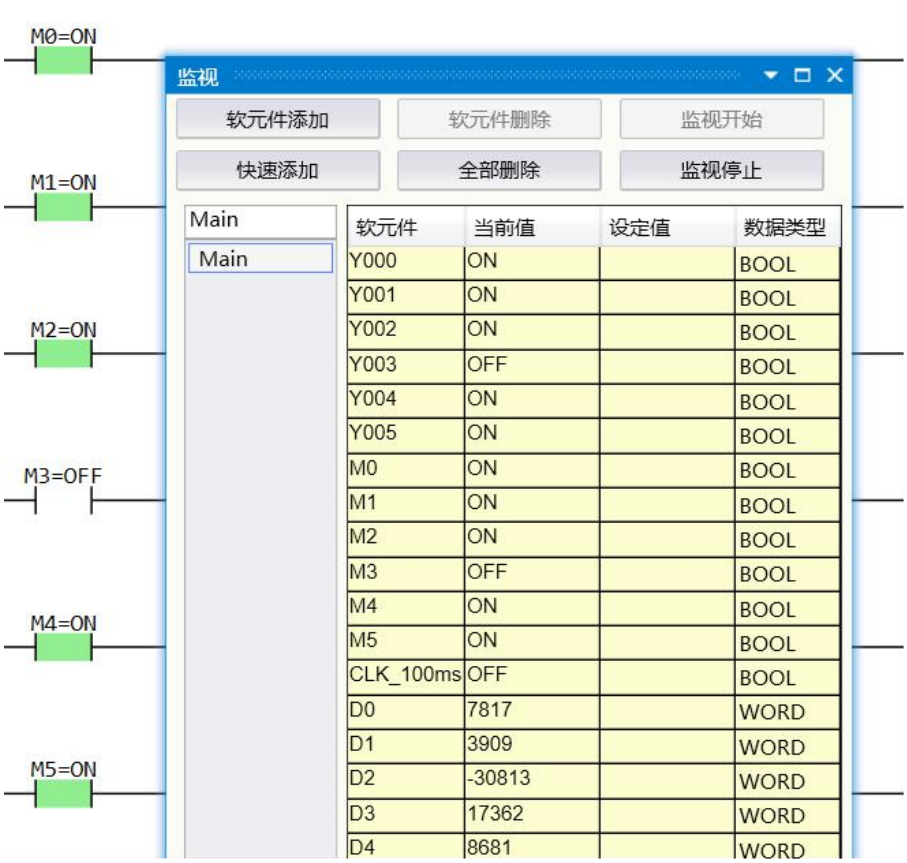


图 2.1 正在监视

这时双击梯形图或者监视表时，会弹出软元件值修改的对话框。

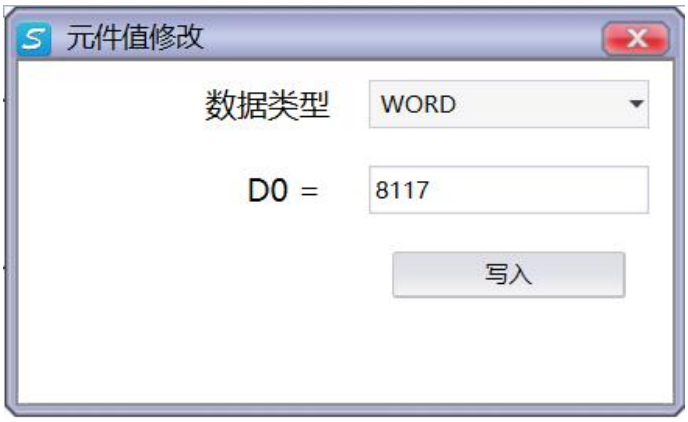


图 2.2 修改值

对于字软元件，输入要修改的值点击【写入】就会立即生效。对于一般的位软元件，只能进行置位和复位操作。对于 X,Y，可以强制设为 ON 和 OFF，也可以取消单个或者全部强制。



图 2.3 修改位



图 2.4 强制位

3.9 PLC 参数设置

单击工具栏上的 PLC 设置图标，或者点击菜单栏“工程”-->“属性”，PLC 设置画面将会打开。

1. 设备型号设置

这里选择当前使用的 PLC 设备的型号，目前分为基本型 FGs 和 mini 型 FGr 这两个系列，基本型有 FGs_16MR_A，FGs_16MT_A，FGs_32MR_A，FGs_32MT_A，FGs_64MR_A，FGs_64MT_A，FGm_64MT_A 这七种，mini 型有 FGRB_C8X8T，FGRS_C8X8T，FGRB_C8X6R 这三种。



图 1.1

注意：

1、用户若无法分清当前所连 PLC 的实际型号，那么在创建新工程时也可任意选择其中一种，因为在下载时会有型号校验功能，帮助用户自动切换型号；

2、不同型号之间可能会有不同的功能，某些指令仅能在某种型号中使用；

2. 通信参数设置



图 1.2

COM 口参数设置：在该页面可以对 PLC 的通信参数进行相应的设置。PLC 串口的默认通信参数为：波特率 9600，数据位 8，停止位 1，校验码 NONE；

串口通信协议类型：包括 Modbus 主站、Modbus 从站、以及自由口通信，默认协议为 Modbus 从站；

站号：设置 PLC 的站号，不同的 COM 口其站号可以不同；

超时时间：该参数仅在 Modbus 主站通信时生效，对于从站和自由口通信而言没有实际意义，表示 PLC 作为主站通信时每次接收从站一包数据的最大等待时间，默认值为 20*10ms；

重传次数：该参数仅在 Modbus 主站通信时生效，对于从站和自由口通信而言没有实际意义，表示当接收从站数据超时后会再次进行重传的次数，默认值为 3；

缓冲位：该参数仅在自由口通信时生效，对于 Modbus 主站和从站而言没有实际意义，表示通信时数据以 8 位还是 16 位格式进行缓冲；

注意：

1、若想使用 MODBUS 主站通信必须选择 Modbus 主站协议，且建立并编辑 Modbus 主站表格，在梯形图中使用 Modbus 指令；

2、当该 COM 口的通讯协议设置为 MODBUS 主站或自由口通讯用后,无法再用来下载工程,也无法进行监视，仅当选择 Modbus 从站协议并下载至 PLC 中才具有下载和监视功能；

3、除了可通过配置信息下载更新串口信息，对于串口的波特率、数据位、停止位、校验码、站号等设置还可通过特殊寄存器 D8066~D8075、M8184 设置；

3. 密码设置

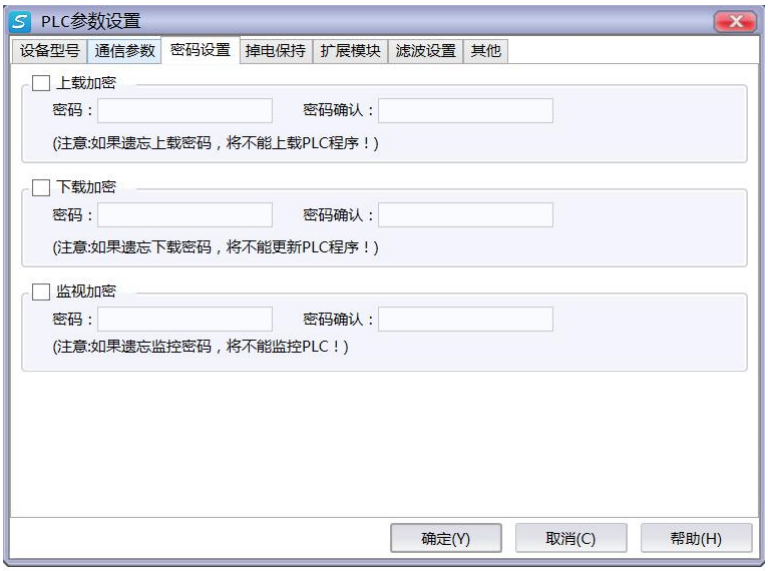


图 1.3

·上载加密：设置 PLC 程序的上载密码，在上载程序的时候回提示输入该密码，遗忘该密码的话会导致程序无法上载

·下载加密：设置 PLC 程序的下载密码，在下载程序的时候回提示输入该密码，遗忘该密码的话会导致无法更新 PLC 内部的程序

·监控加密：设置 PLC 监控的密码，在上位梯形图监控该模式的时候会用到，遗忘该密码将不能监控 PLC 程序

注意：密码必须以字母或数字开头，长度为 4 至 12 位。

4. 保持区设置



图 1.6

掉电保持区范围最大可设置为：M7488~M7999、D5952~D7999、S600~S999、CV100~CV199、CV220~CV234、CV235~CV255；

·在该页面中可以对掉电保持区的容量进行设置，点击默认设置的时候会清空当前设置并将掉电保持区的起始地址设置为最小值，将长度设置为最大值

·勾选下载时不清除掉电保持数据，在下载梯形图程序时将不清除下载之前掉电保持区的数据。

·掉电保持功能不受【下载时不清除】这个设置的影响，始终按照用户设置的表格来执行。

5. 扩展模块设置

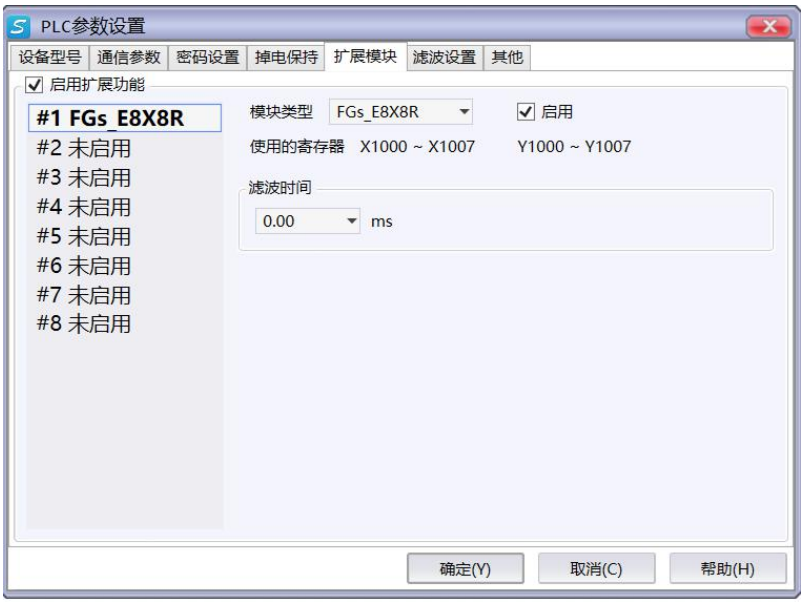


图 1.7

·勾选使用扩展模块，开始扩展模块设置

·具体使用请查看扩展模块使用说明

6. 滤波设置



图 1.5

·在该页面可以进行滤波设置，滤波设置范围从 0.20~51.20ms 之间。

1、滤波设置共有三种模式：不启用、统一模式、其它模式

当选择不启用时，则不开启对输入信号 X 的软件滤波；

当选择统一模式时，则可配置滤波时间，该滤波参数适用于所有 X 输入通道；

当选择其它时，则可单独配置每一路 X 输入通道的滤波时间；

2、对于多脉冲型 FGm 系列 PLC，除了以上的滤波设置外，另外还可选择是否启用硬件滤波，但硬件滤波只可配置前 16 路 X 通道；



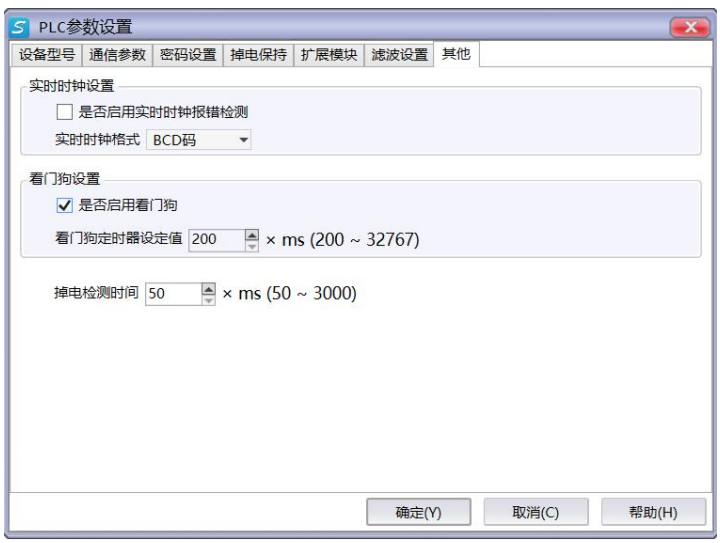
分频系数是以 50MHz 的采样时钟为基准，另外 PLC 内部默认对分频过后的输入信号采样时钟进行连续 8 次采样才确定唯一高低电平；

如当分频系数选择为 2 时，表示先对 50MHz 的基准时钟进行二分频后，以 25MHz 的采样频率对输入信号进行采样，除此之外内部默认需连续采样 8 次的电平都是唯一的才确定该电平是有效的，若以 25MHz 连续采样的 8 次电平中有一次不相同的，　　则该电平认为是因干扰引起的无效电平，因此该方法可有效地抑制干扰抖动等不确定因数；

需小心使用该硬件滤波，如若实际输入信号的频率较高时，请不要将该分频系数设置地过大，否则分频过后的采样时钟无法对高频率的输入信号进行有效采样；

如当使用高速计数功能时，实际计数最大频率若可达到 500KHz，则最大可配置的分频系数应<=50MHz/(500KHz*8*2)≈3，即分频系数最大可设置为 2；

7. 其他



实时时钟的设置：

1、可配置实时时钟读写格式为 BCD 码或 10 进制；

2、当启用实时时钟报错检测时，若实时时钟异常则 PLC 会进入 ERROR、STOP 状态，PLC 将停止运行，可通过查看 D8176 是否等于 9，或者通过上位软件监视查看状态栏信息；

3、当不启用实时时钟报错检测时，即便实时时钟异常也不会影响 PLC 的正常运行，仍可通过 D8176 或监视信息状态栏查看 PLC 是否异常；

看门狗的设置：

1、可由用户选择是否配置看门狗定时器，并且可设定看门狗定时器值，该功能一般用于检测用户编写梯形图或函数功能块时造成死循环的问题；

2、在 PLC 运行的每个扫描周期结束时内部会默认对看门狗进行复位，也可通过在梯形图程序中使用 WDT 指令手动复位看门狗；

3、若超过设定值时间后若没有复位则 PLC 会进入 ERROR、STOP 状态，PLC 将停止运行，可通过查看 D8176 是否等于 10，或者通过上位软件监视查看状态栏信息；

掉电保持时间的设置：

1、该参数用来设置 PLC 判断 PLC 掉电软件滤波时间，如当设置为默认值 50ms 时，表明电网波动低于 50ms 时，将认为该 PLC 处于正常供电状态，若超过 50ms 后，则认为该 PLC 进入了掉电状态，此时 PLC 将停止运行并存储掉电保持数据；

2、当外部干扰比较严重时导致 PLC 运行时误进 STOP 掉电状态，则可适当增加该掉电检测时间；

3、当增大该值时，可能会影响掉电保持数据的正常存储，造成掉电数据不保持，因此该掉电检测时间仅在外部供电电压不稳定导致 PLC 可能误进掉电模式时才适当地略微增大，其它情况下请不要擅自修改该参数值；

3.10 系统设置

选择菜单栏中的“工具”-->“选项”，或者直接单击工具栏上的系统设置图标，系统设置对话框将会打开。

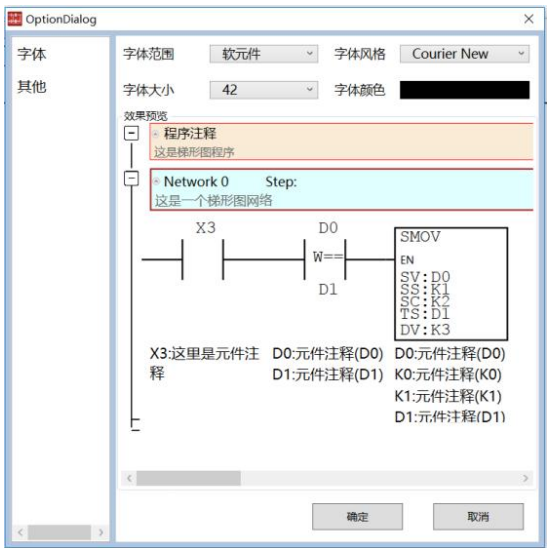


图 1.1

1. 字体设置

选择“字体设置”选项卡可以对编程界面上的字体风格、字体颜色、字体大小等信息进行设置。

字体范围 软元件 可以选择软元件，梯形图标题，注释以及函数功能块。

软元件是指软元件上除了 W== 这种作为图元符号以外的图元上的所有字体，梯形图标题是指 程序注释 以及 Network 0 Step: ，注释是指标题以及图元下方的注释。函数功能块是指编写功能块的 C 语言代码的字体。

字体风格 Courier New 可以设置字体的显示风格，常用的有微软雅黑和宋体。

字体大小 42 设置字体的大小，一般来说，软元件注释的字体不要设的太大，否则会导致界面空间不足。

单击 字体颜色 会弹出画板框，可以对颜色进行编辑。

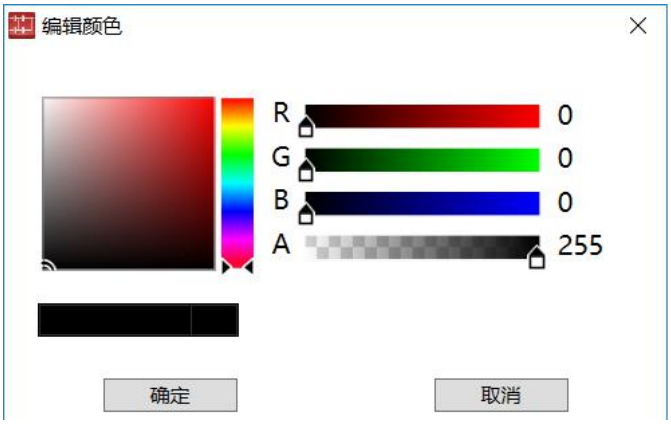


图 1.2

2. 其他设置

选择“其他”选项卡可以设置更强大的功能。

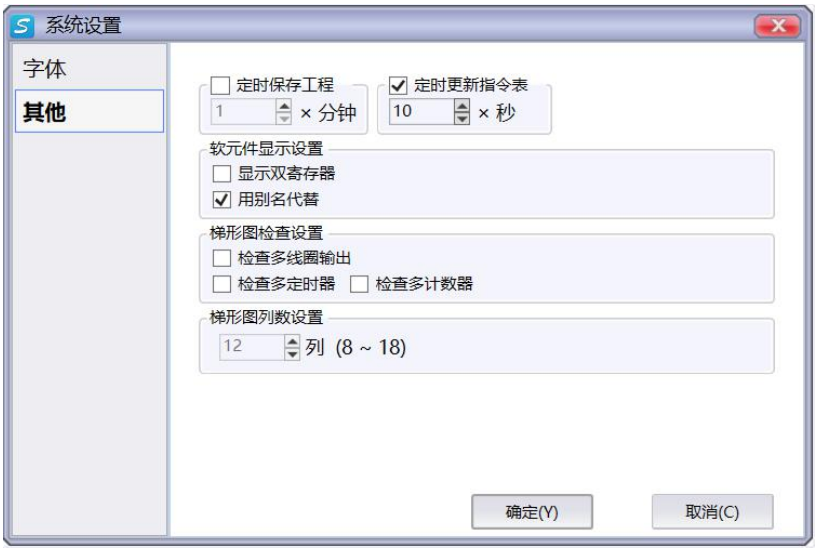


图 1.3

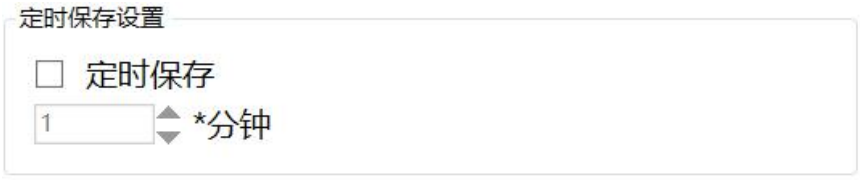


图 1.4

勾选此选项可以设定任意分钟间隔内自动保存工程保存到所在路径中。

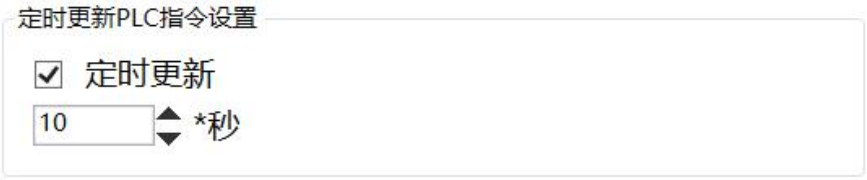


图 1.5

勾选此选项可以设定任意秒间隔内自动更新修改过的梯形图的指令表。

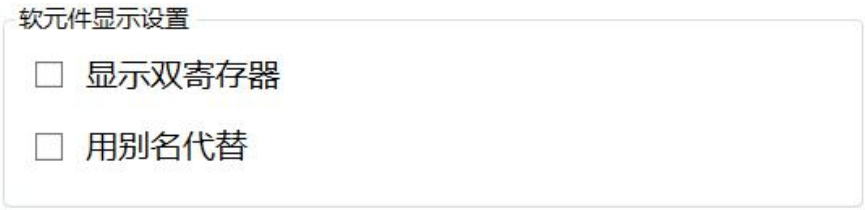


图 1.6

勾选【显示双寄存器】时，会将被占用为双字的单字寄存器的下一个字显示出来。例如双字的 D0 会显示为 D0D1。由于会占显示空间并且破坏美观，建议不要勾选。

勾选【用别名代替】时，会将设置了别名(Alias)的软元件用别名来代替显示。例如 1s 时钟 M8160 的系统默认别名为 CLK_1s，会在梯形图和监视表中显示。



图 1.7



图 1.8

勾选【检查多线圈输出】后，对同一个软元件的多个线圈输出将不会被允许，梯形图检查时会报错。

勾选【检查多定时器】，【检查多计数器】同理，不允许同一个计时器或者计数器使用在多个位置上。

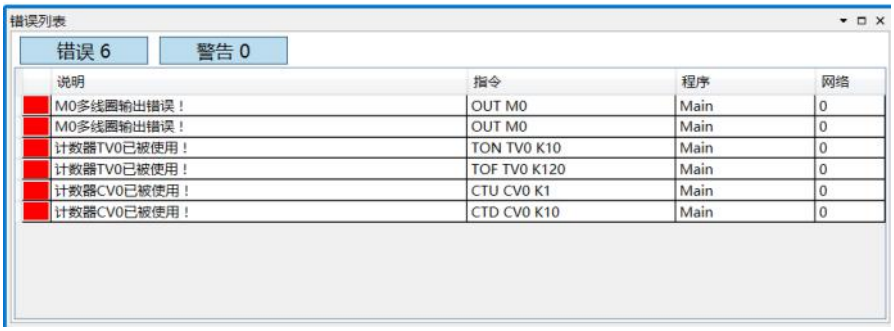


图 1.9

3.11 通信设置

单击标准工具栏上的通信设置按钮，可以打开通信设置对话框。



图 1.1

选择 USB 口时可以直接进行通信测试，选择 COM 口时需要设置通信参数。可以自动检测，也可以手动设置。



图 1.2

点击左下角通信测试，上位软件会尝试和下位进行一次通信，若是手动配置的参数，会根据配置的参数来进行通信，若选择的是自动检测，软件会自动切换识别通信参数。

当通信成功后，会显示对话框：



否则会显示：



4. PLC 指令

内容如下：

1、 位逻辑
2、 比较
3、 转换
4、 逻辑运算
5、 传送
6、 浮点数计算
7、 整数计算
8、 定时器
9、 计数器
10、程序控制
11、移位指令
12、中断指令
13、实时时钟
14、通信指令
15、脉冲输出指令
16、高速计数指令
17、辅助指令
PID 指令

4.1 位逻辑

位逻辑指令

内容如下：

1、 常开输入指令 -| |-

2、 立即常开输入指令 -|||-

3、 常闭输入指令 -|/|-

4、 立即常闭输入指令 -|/||-

5、 上升沿脉冲 -|↑|-

6、 下降沿脉冲 -|↓|-

7、 结果上升沿 -↑-

8、 结果下降沿 -↓-

9、 运算结果取反指令 -/-

10、普通输出线圈 -()

11、立即输出线圈 –(|)

12、置位输出线圈 –(S)

13、立即置位输出线圈 –(SI)

14、复位输出线圈 –(R)

15、立即复位输出线圈 –(RI)

16、连续交替输出 ALT

17、脉冲交替输出 ALTP

4.1.1 -| |-

对应命令	注释/说明	操作数（线圈）
LD、AND、OR	常开触点	X/Y/M/T/C/S

1. 指令说明

当指定的位等于 1（ON）时，常开触点闭合(接通)，当指定的位等于 0（OFF）时，常闭触点闭合(断开)。

在指令表中，常开触点用 LD（载入）、AND（与）和 OR（或）指令表示，这些指令 LD（载入）、AND（与）或 OR（或）将地址位数值置于堆栈顶部。

2. 指令的有效操作数

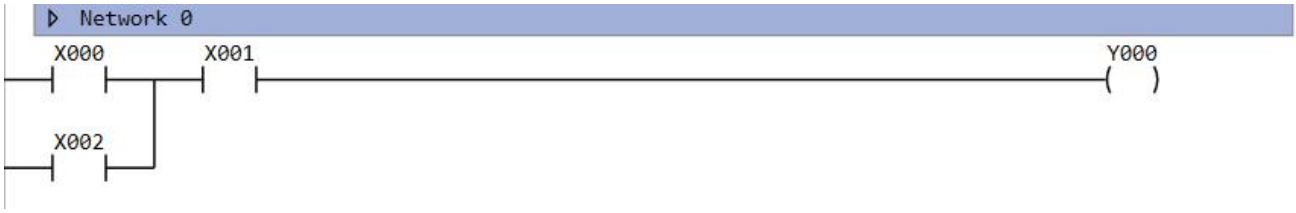
输入输出	类型	操作数	描述
无	BOOL	**X/Y/M/T/C/S**	**指定的操作数**

3. 实例

指令表：

```
LD      X000    // 加载输入线圈 X000，X000 必须等于 1 时才能激活 X000
OR      X002    // 与输入线圈 X002 进行‘或’操作
AND     X001    // 与输入线圈 X001 进行‘与’操作
OUT     Y000    // 输出线圈 Y000
```

梯形图：



4.1.2 -|||-

对应命令	注释/说明	操作数（线圈）
LDIM、ANDIM、ORIM	立即常开触点	X/Y/M/T/C/S

1. 指令说明

当指定的位等于 1（ON）时，立即常开触点闭合(接通)，当指定的位等于 0（OFF）时，立即常开触点闭合(断开)。指定的位不依靠 FGs 扫描周期进行更新，它会立即更新。

在指令表中，立即常开触点用 LDIM（立即载入）、ANDIM（立即与）和 ORIM（立即或）指令表示，这些指令 LDIM（立即载入）、ANDIM（立即与）或 ORIM（立即或）将地址位数值置于堆栈顶部，并立即执行。

2. 指令的有效操作数

输入输出	类型	操作数	描述
无	BOOL	**X/Y/M/T/C/S**	**指定的操作数**

3. 实例

指令表：

```
LDIM    X000    // 立即加载输入线圈 X000，X000 必须等于 1 时才能激活 X000
ORIM    X002    // 与输入线圈 X002 进行‘立即或’操作
ANDIM   X001    // 与输入线圈 X001 进行‘立即与’操作
OUT     Y000    // 输出线圈 Y000
```

梯形图：



4.1.3 -|/|-

对应命令	注释/说明	操作数（线圈）
LDI、ANDI、ORI	常闭触点	X/Y/M/T/C/S

1. 指令说明

当指定的位等于 0（OFF）时，常开触点闭合(接通)，当指定的位等于 1（ON）时，常闭触点闭合(断开)。

在指令表中，常闭触点用 LD　NOT（载入非）、AND　NOT（与非）和 OR　NOT（或非）指令表示，这些指令 LD（载入）、AND（与）或 OR（或）将地址位数值的逻辑 NOT（非）置于堆栈顶部。

2. 指令的有效操作数

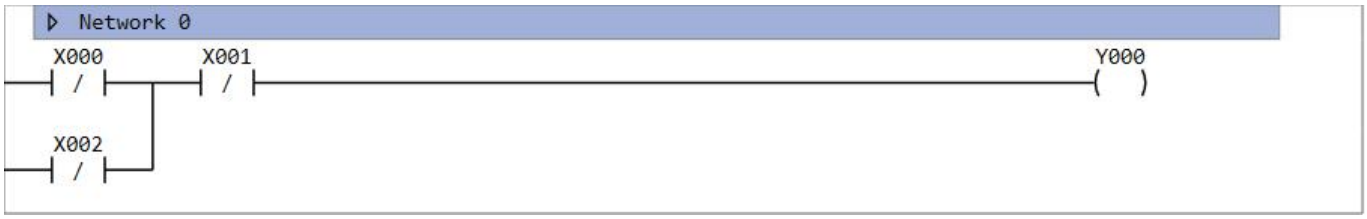
输入输出	类型	操作数	描述
无	BOOL	X/Y/M/T/C/S	指定的操作数

3. 实例

指令表：

```
LDI     X000    // 加载输入线圈 X000 的逻辑 NOT，X000 必须等于 0 时才能激活 X000
ORI     X002    // 与输入线圈 X002 进行‘或非’操作
ANDI    X001    // 与输入线圈 X001 进行‘与非’操作
OUT     Y000    // 输出线圈 Y000
```

梯形图：



4.1.4 -|/||-

对应命令	注释/说明	操作数（线圈）
LDIIM、ANDIIM、ORIIM	立即常闭触点	X/Y/M/T/C/S

1. 指令说明

当指定的位等于 0（OFF）时，立即常闭触点闭合(接通)，当指定的位等于 1（ON）时，立即常闭触点闭合(断开)。指定的位不依靠 FGs 扫描周期进行更新,它会立即更新。

在指令表中，立即常闭触点用 LDIIM（立即载入非）、ANDIIM（立即与非）和 ORIIM（立即或非）指令表示，这些指令 LDIM（立即载入）、ANDIM（立即与）或 ORIM（立即或非）将地址位数值的逻辑 NOT（非）置于堆栈顶部，并立即执行。

2. 指令的有效操作数

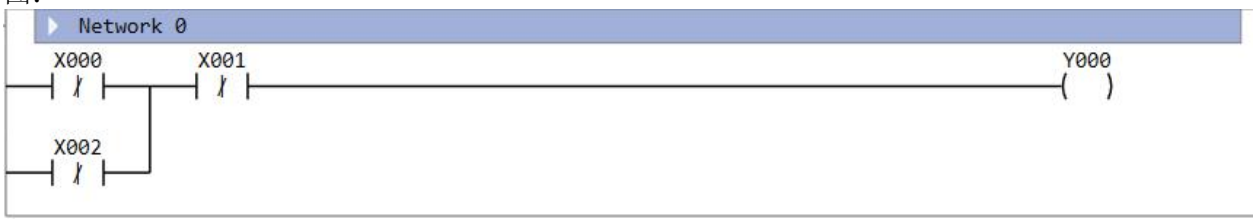
输入输出	类型	操作数	描述
无	BOOL	**X/Y/M/T/C/S**	**指定的操作数**

3. 实例

指令表：

```
LDIIM    X000     // 立即加载输入线圈 X000 的逻辑 NOT，X000 必须等于 0 时才能激活 X000
ORIIM    X002     // 与输入线圈 X002 进行‘立即或非’操作
ANDIIM   X001     // 与输入线圈 X001 进行‘立即与非’操作
OUT      Y000     // 输出线圈 Y000
```

梯形图：



4.1.5 -|↑|-

对应命令	注释/说明	操作数（寄存器）
LDP、ANDP、ORP	上升沿脉冲	X/Y/M/T/C/S

1. 指令说明

1） 当指定的位检测到正跳变（断到通）时，触点闭合一个扫描周期。

2） 对于上升沿指令，指定的位允许一次扫描中每次执行 0 到 1（OFF → ON）转换时，将堆栈顶值设为 1；否则，将其设为 0；

3） 指令每次只执行一瞬间（PLC 扫描周期）。常用于顺控中信号触发用，不能当作输出（外部线圈可能不吸，这点要注意）；

4） 在使用该指令时要格外注意，因为只有在系统检测到上升沿变化时触点才会接通，实际使用时处理不当可能会错过致使不能发生。

2. 指令的有效操作数

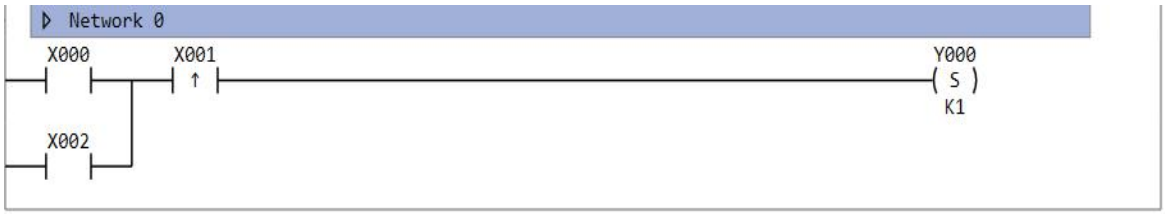
输入输出	类型	操作数	描述
无	**BOOL**	**X/Y/M/T/C/S**	**指定的操作数**

3. 实例

指令表：

```
LD       X000          // 加载输入线圈 X000，X000 必须等于 1 时才能激活 X000
OR       X002          // 与输入线圈 X002 进行‘或’操作
ANDP     X001          // 与输入线圈 X001 进行‘与’操作
SET      Y000    K1    // 输出线圈 Y000
```

梯形图：



4.1.6 -|↓|-

对应命令	注释/说明	操作数（寄存器）
LDF、ANDF、ORF	下降沿脉冲	X/Y/M/T/C/S

1. 指令说明

1） 当指定的位检测到负跳变（通到断）时，触点闭合一个扫描周期；

2） 对于下降沿指令，指定的位触点允许一次扫描中每次执行 1 到 0（ON → OFF）转换时，将堆栈顶值设为 1；否则，将其设为 0；

3） 指令每次只执行一瞬间（PLC 扫描周期）。常用于顺控中信号触发用，不能当作输出（外部线圈可能不吸，这点要注意）；

4） 在使用该指令时要格外注意，因为只有在系统检测到下降沿变化时触点才会接通，实际使用时处理不当可能会错过致使不能发生。

2. 指令的有效操作数

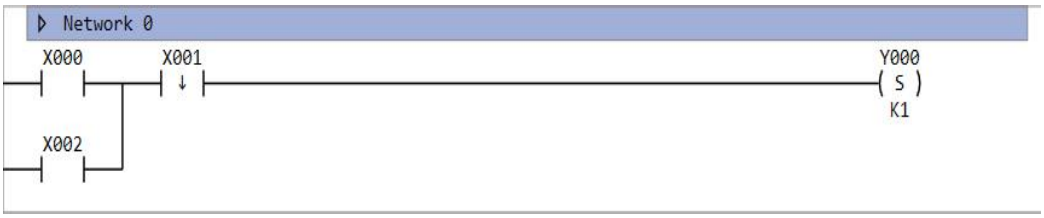
输入输出	类型	操作数	描述
无	BOOL	**X/Y/M/T/C/S**	**指定的操作数**

3. 实例

指令表：

```
LD      X000          // 加载输入线圈 X000，X000 必须等于 1 时才能激活 X000
OR      X002          // 与输入线圈 X002 进行‘或’操作
ANDF    X001          // 与输入线圈 X001 进行‘与’操作
SET     Y000    K1    // 输出线圈 Y000
```

梯形图：



4.1.7 -↑-

结果上升沿 -↑-

对应命令	注释/说明	操作数（寄存器）
MEP	结果上升沿	None

1. 指令说明

1） MEP 指令前的运算结果为上升沿（OFF→ ON）时，该指令将变为 ON（导通状态）；否则该指令将变为 OFF（非导通状态）。即使得结果脉冲化。

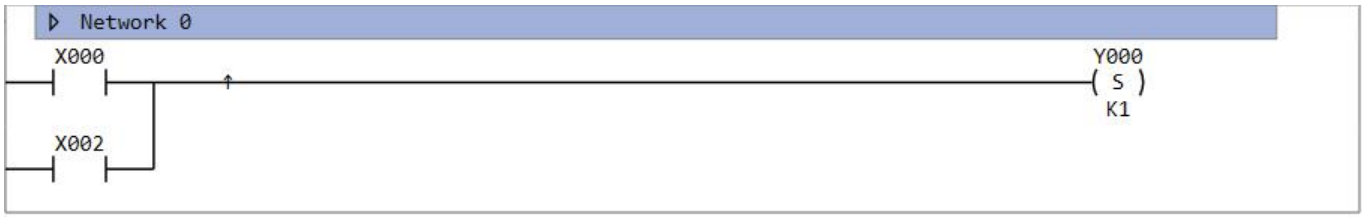
2） 输出将会在边沿信号导通的瞬间执行一次，除非再次有相应的边沿信号产生时才会再执行一次。

2. 实例

指令表：

```
LD      X000
OR      X002
MEP                     // 如果前方触点结果有上升沿，则执行后方输出
SET     Y000    K1
```

梯形图：



4.1.8 -↓-

结果下降沿 -↓-

对应命令	注释/说明	操作数（寄存器）
MEF	结果下降沿	None

1. 指令说明

1） MEP 指令前的运算结果为下降沿（ON → OFF）时，该指令将变为 ON（导通状态）；否则该指令将变为 OFF(非导通状态)。即使得结果脉冲化。

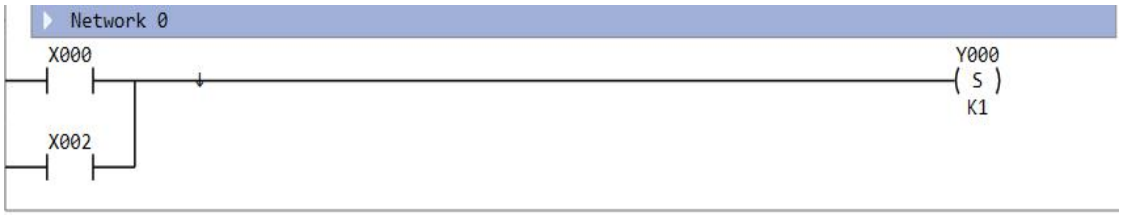
2） 输出将会在边沿信号导通的瞬间执行一次，除非再次有相应的边沿信号产生时才会再执行一次。

2. 实例

指令表：

```
LD      X000
OR      X002
MEF                     // 如果前方触点结果有下降沿，则执行后方输出
SET     Y000    K1
```

梯形图：



4.1.9 -/-

运算结果取反指令 -/-

对应命令	注释/说明	操作数（寄存器）
INV	结果取反	None

1. 指令说明

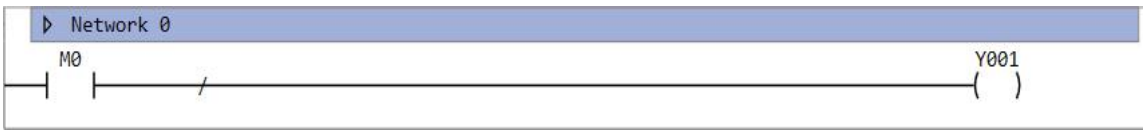
将至 INV 之前的运算结果取反后作为输出的使能。

2. 实例

指令表：

```
LD      M000
INV             // 将 M000 结果取反输出到 Y001
OUT     Y001
```

梯形图：



4.1.10 -( )-

普通输出线圈 -()

对应命令	注释/说明	操作数（寄存器）
OUT	普通输出线圈	Y/M/S

1. 指令说明

将新值写入指定的位寄存器。指令执行时，FGs 将指定的位寄存器中的位接通或者断开；

2. 指令的有效操作数

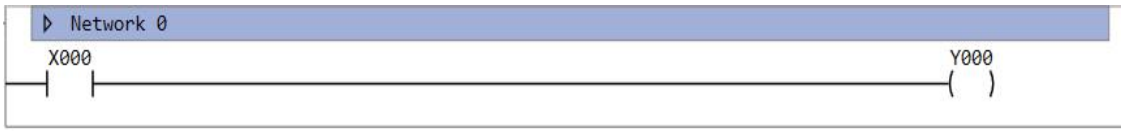
输入输出	类型	操作数	描述
无	BOOL	Y/M/S	指定的操作数

3. 实例

指令表：

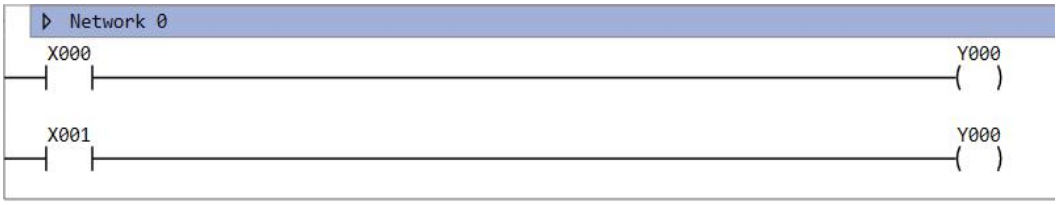
```
LD      X000
OUT     Y000    // 输出 Y000
```

梯形图：



4. 错误实例

注意：禁止线圈重复输出，如错误实例，X0 导通而 X1 未导通，则梯形图顺序执行过程中，X0 先导通 Y0 有输出，X1 未导通 Y0 禁止输出，所以 Y0 始终无法输出，可采用辅助继电器 M 解决该问题。



4.1.11 -(|)-

立即输出线圈 –(|)

对应命令	注释/说明	操作数（寄存器）
OUTIM	立即输出线圈	Y/M/S

1. 指令说明

将新值立即写入指定的位寄存器并输出。指令执行时，指定的位不依靠 FGs 扫描周期进行更新。

2. 指令的有效操作数

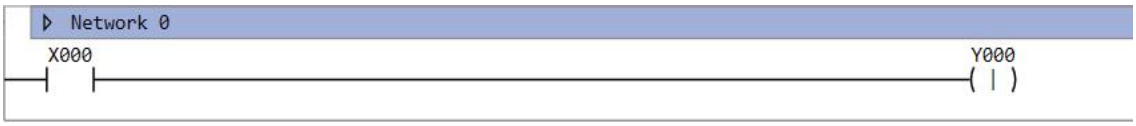
输入输出	类型	操作数	描述
无	BOOL	Y/M/S	指定的操作数

3. 实例

指令表：

```
LD      X000
OUTIM   Y000   // 立即输出 Y000
```

梯形图：



4.1.12 -(S)-

置位输出线圈 –(S)

对应命令	注释/说明	操作数（寄存器）
SET	置位输出线圈	Y/M/S/T/C, K/H

1. 指令说明

置位从指定地址位（BIT）开始的（N）个操作数，最大点数根据操作数类型设定。

2. 指令的有效操作数

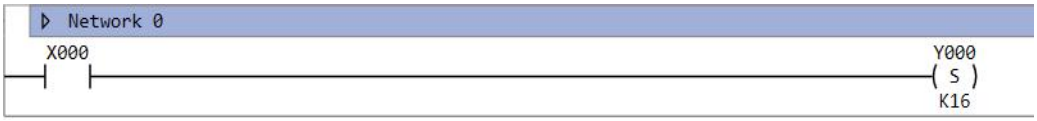
输入输出	类型	操作数	描述
（上）	BOOL	Y/M/S/T/C	指定的置位起始地址
（下）	BOOL	K/H	置位的操作数个数

3. 实例

指令表：

```
LD      X000
SET     Y000    K16   // 置位 Y000~Y017
```

梯形图：



4.1.13 -(SI)-

对应命令	注释/说明	操作数（寄存器）
SETIM	立即置位输出线圈	Y/M/S/T/C, K/H

1. 指令说明

立即置位从指定地址位（BIT）开始的（N）个操作数，最大点数根据操作数类型设定。置位不依靠 FGs 扫描周期进行更新。

2. 指令的有效操作数

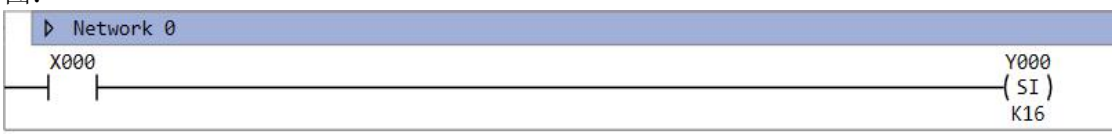
输入输出	类型	操作数	描述
（上）	**BOOL**	**Y/M/S/T/C**	**指定的置位起始地址**
（下）	**BOOL**	**K/H**	**置位的操作数个数**

3. 实例

指令表：

```
LD      X000
SETIM   Y000    K16   // 立即置位 Y000~Y017
```

梯形图：



4.1.14 -(R)-

复位输出线圈 –(R)

对应命令	注释/说明	操作数（寄存器）
RST	复位输出线圈	Y/M/S/T/C, K/H

1. 指令说明

复位从指定地址位（BIT）开始的（N）个操作数，最大点数根据操作数类型设定。

2. 指令的有效操作数

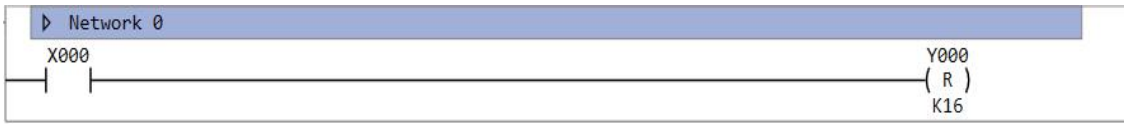
输入输出	类型	操作数	描述
（上）	BOOL	Y/M/S/T/C	指定的复位起始地址
（下）	BOOL	K/H	复位的操作数个数

3. 实例

指令表：

```
LD      X000
RST     Y000    K16   // 复位 Y000~Y017
```

梯形图：



4.1.15 -(RI)-

对应命令	注释/说明	操作数（寄存器）
RSTIM	立即复位输出线圈	Y/M/S/T/C, K/H

1. 指令说明

立即复位从指定地址位（BIT）开始的（N）个操作数，最大点数根据操作数类型设定。置位不依靠 FGs 扫描周期进行更新。

2. 指令的有效操作数

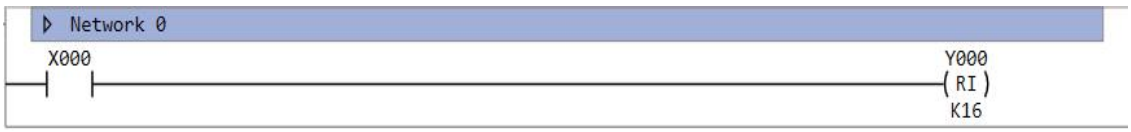
输入输出	类型	操作数	描述
（上）	BOOL	Y/M/S/T/C	指定的置位起始地址
（下）	BOOL	K/H	置位的操作数个数

3. 实例

指令表：

```
LD      X000
RSTIM   Y000    K16   // 立即置位 Y000~Y017
```

梯形图：



4.1.16 ALT

1. 指令说明

当指定的操作数由 OFF 变为 ON 时，指令让指定的操作数在 1（ON）和 0（OFF）之间连续交替改变。

2. 指令的有效操作数

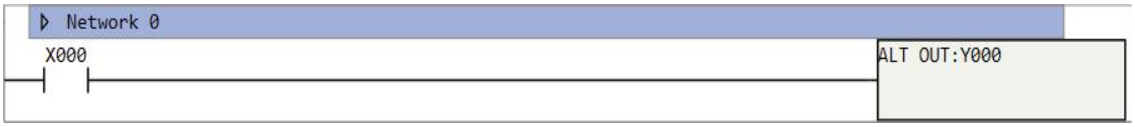
输入输出	类型	操作数	描述
EN			启用输入端接通时，指令执行；否则，不执行
控制位	BOOL	Y/M/S	指定的操作数

3. 实例

指令表：

```
LD      X000
ALT     Y000   //Y000 在 0 和 1 交替变化，周期为当前扫描周期
```

梯形图：



4.1.17 ALTP

1. 指令说明

当指定的操作数由 OFF 变为 ON 时，指令让指定的操作数在 1（ON）和 0（OFF）之间交替改变一次。

2. 指令的有效操作数

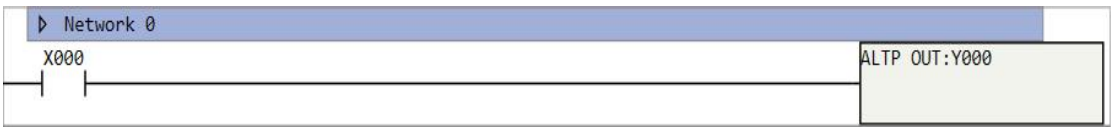
输入输出	类型	操作数	描述
EN			启用输入端接通时，指令执行；否则，不执行
控制位	BOOL	Y/M/S	指定的操作数

3. 实例

指令表：

```
LD      X000
ALTP    Y000   //  当 X000 从 0 变为 1 时，Y000 在 0 和 1 交替变化
```

梯形图：



4.2 比较

比较指令

内容如下：

1、 比较指令==(查看 [标题编号])
2、 比较指令<>(查看 [标题编号])
3、 比较指令>=(查看 [标题编号])
4、 比较指令<=(查看 [标题编号])
5、 比较指令>(查看 [标题编号])
6、 比较指令<(查看 [标题编号])

4.2.1 ==

比较指令==

对应命令	注释/说明	操作数（寄存器）
W=	LDW=、AW=、ORW=	K/H/D/CV/TV/AI/AO/V/Z,K/H/D/CT/TV/AI/AO/V/Z
D=	LDD=、AD=、ORD=	K/H/D/CV32,K/H/D/CV32
F=	LDF=、AF=、ORF=	K/D/CV32,K/D/CV32

1. 指令说明

等于指令用于比较两个数值是否相等，如果上方的数据等于下方的数据，比较结果为真，触点接通。

比较的两个数是带符号的

W= 比较双字节整数

D= 比较四字节整数

F= 比较实数

在指令表中，当 1 位于堆栈顶端时，指令执行载入（LD）、与（AND）和或（OR）操作。

2. 指令的有效操作数

输入输出	类型	操作数	描述
（上）	字/双字/实数	K/H/D/CV/TV/AI/AO/V/Z	要参加比较的操作数
（下）	字/双字/实数	同上	同上

3. 实例

指令表：

```
LDW=    D0      D1    // 加载 D0 和 D1，并比较 D0 是否等于 D1
ORW=    D4      H0123 // 比较 D4 和 H0123 是否相等，并将比较结果和栈顶值进行或操作
AW=     D2      D3 // 比较 D2 和 D3 是否相等，并将比较结果和栈顶值进行与操作
LDD=    D6      D8 // 加载 D6D7 和 D8D9，并比较 D6D7 是否等于 D8D9
ORD=    D14     K888 // 比较 D14D15 和 K888 是否相等，并将比较结果和栈顶值进行或操作
AD=     D10V0   D12Z0 // 比较（D10D11+地址偏移 V0）和（D12D13+地址偏移 Z0）是否相等，并将比较结果和栈顶值进行‘与’操作
LDF=    D18     D20 // 加载浮点数 D18D19 和浮点数 D20D21，并比较 浮点数 D18D19 是否等于浮点数 D20D21
ORF=    D26     K1.234567 // 比较浮点数 D26D27 和 K1.234567 是否相等，并将比较结果和栈顶值进行‘或’操作
AF=     D22     D24 // 比较浮点数 D22D23 和浮点数 D24D25 是否相等，并将比较结果和栈顶值进行‘与’操作
ORB             // 将栈顶值与当前能流值进行‘或’操作
ORB             // 将栈顶值与当前能流值进行‘或’操作
OUT     Y000    // 输出 Y000
```

梯形图：



4.2.2 <>

比较指令<>

对应命令	注释/说明	操作数（寄存器）
W<>	LDW<>、AW<>、ORW<>	K/H/D/CV/TV/AI/AO/V/Z,K/H/D/CV/TV/AI/AO/V/Z
D<>	LDD<>、AD<>、ORD<>	K/H/D/CV32,K/H/D/CV32
F<>	LDF<>、AF<>、ORF<>	K/D/CV32,K/D/CV32

1. 指令说明

不等于指令用于比较两个数值是否不相等，如果上方的数据不等于下方的数据，比较结果为真，触点接通。

比较的两个数是带符号的

W<> 比较双字节整数

D<> 比较四字节整数

F<> 比较实数

在指令表中，当 1 位于堆栈顶端时，指令执行载入（LD）、与（AND）和或（OR）操作。

2. 指令的有效操作数

输入输出	类型	操作数	描述
（上）	字/双字/实数	K/H/D/CV/TV/AI/AO/V/Z	要参加比较的操作数
（下）	字/双字/实数	同上	同上

3. 实例

指令表：

```
LDW<>   D0      D1   // 加载 D0 和 D1，并比较 D0 是否‘不等于’D1
ORW<>   D4      D5   // 比较 D4 是否‘不等于’D5，并将比较结果和栈顶值进行‘或’操作
AW<>    D2      D3   // 比较 D2 是否‘不等于’D3，并将比较结果和栈顶值进行‘与’操作
LDD<>   D6      D8   // 加载 D6D7 和 D8D9，并比较 D6D7 是否‘不等于’D8D9
ORD<>   D14     D16 // 比较 D14D15 是否‘不等于’D16D17，并将比较结果和栈顶值进行‘或’操作
AD<>    D10     D12 // 比较 D10D11 是否‘不等于’D12D13，并将比较结果和栈顶值进行‘与’操作
LDF<>   D18     D20 // 加载浮点数 D18D19 和 D20D21，并比较 浮点数 D18D19 是否‘不等于’浮点数 D20D21
ORF<>   D26     D28 // 比较浮点数 D26D27 是否‘不等于’浮点数 D28D29，并将比较结果和栈顶值进行‘或’操作
AF<>    D22     D24 // 比较浮点数 D22D23 是否‘不等于’浮点数 D24D25，并将比较结果和栈顶值进行‘与’操作
ORB                 // 将栈顶值与当前能流值进行‘或’操作
ORB                 // 将栈顶值与当前能流值进行‘或’操作
OUT     Y000        // 输出 Y000
```

梯形图：



4.2.3 >=

比较指令>=

对应命令	注释/说明	操作数（寄存器）
W>=	LDW>=、AW>=、ORW>=	K/H/D/CV/TV/AI/AO/V/Z,K/H/D/CT/TV/AI/AO/V/Z
D>=	LDD>=、AD>=、ORD>=	K/H/D/CV32,K/H/D/CV32
F>=	LDF>=、AF>=、ORF>=	K/D/CV32,K/D/CV32

1. 指令说明

大于等于指令用于比较两个数值的大小，如果上方的数据大于或者等于下方的数据，比较结果为真，触点接通。

比较的两个数是带符号的

W >= 比较双字节整数

D >= 比较四字节整数

F >= 比较实数

在指令表中， 当 1 位于堆栈顶端时，指令执行载入（LD）、与（AND）和或（OR）操作。

2. 指令的有效操作数

输入输出	类型	操作数	描述
（上）	字/双字/实数	K/H/D/CV/TV/AI/AO/V/Z	要参加比较的操作数
（下）	字/双字/实数	同上	同上

3. 实例

指令表：

```
LDW>=   D0      D1   // 加载 D0 和 D1，并比较 D0 是否‘大于等于’D1
ORW>=   D4      D5   // 比较 D4 是否‘大于等于’D5，并将比较结果和栈顶值进行‘或’操作
AW>=    D2      D3   // 比较 D2 是否‘大于等于’D3，并将比较结果和栈顶值进行‘与’操作
LDD>=   D6      D8   // 加载 D6D7 和 D8D9，并比较 D6D7 是否‘大于等于’D8D9
ORD>=   D14     D16 // 比较 D14D15 是否‘大于等于’D16D17，并将比较结果和栈顶值进行‘或’操作
AD>=    D10     D12 // 比较 D10D11 是否‘大于等于’D12D13，并将比较结果和栈顶值进行‘与’操作
LDF>=   D18     D20 // 加载浮点数 D18D19 和 D20D21，并比较 浮点数 D18D19 是否‘大于等于’浮点数 D20D21
ORF>=   D26     D28 // 比较浮点数 D26D27 是否‘大于等于’浮点数 D28D29，并将比较结果和栈顶值进行‘或’操作
AF>=    D22     D24 // 比较浮点数 D22D23 是否‘大于等于’浮点数 D24D25，并将比较结果和栈顶值进行‘与’操作
ORB                 // 将栈顶值与当前能流值进行‘或’操作
ORB                 // 将栈顶值与当前能流值进行‘或’操作
OUT     Y000        // 输出 Y000
```

梯形图：



4.2.4 <=

比较指令<=

对应命令	注释/说明	操作数（寄存器）
W<=	LDW<=、AW<=、ORW<=	K/H/D/CV/TV/AI/AO/V/Z,K/H/D/CT/TV/AI/AO/V/Z
D<=	LDD<=、AD<=、ORD<=	K/H/D/CV32,K/H/D/CV32
F<=	LDF<=、AF<=、ORF<=	K/D/CV32,K/D/CV32

1. 指令说明

小于等于指令用于比较两个数值的大小，如果上方的数据小于或者等于下方的数据，比较结果为真，触点接通。

比较的两个数是带符号的

W<= 比较双字节整数

D <= 比较四字节整数

F <= 比较实数

在指令表中， 当 1 位于堆栈顶端时，指令执行载入（LD）、与（AND）和或（OR）操作。

2. 指令的有效操作数

输入输出	类型	操作数	描述
（上）	字/双字/实数	K/H/D/CV/TV/AI/AO/V/Z	要参加比较的操作数
（下）	字/双字/实数	同上	同上

3. 实例

指令表：

```
LDW<=   D0      D1   // 加载 D0 和 D1，并比较 D0 是否‘小于等于’D1
ORW<=   D4      D5   // 比较 D4 是否‘小于等于’D5，并将比较结果和栈顶值进行‘或’操作
AW<=    D2      D3   // 比较 D2 是否‘小于等于’D3，并将比较结果和栈顶值进行‘与’操作
LDD<=   D6      D8   // 加载 D6D7 和 D8D9，并比较 D6D7 是否‘小于等于’D8D9
ORD<=   D14     D16 // 比较 D14D15 是否‘小于等于’D16D17，并将比较结果和栈顶值进行‘或’操作
AD<=    D10     D12 // 比较 D10D11 是否‘小于等于’D12D13，并将比较结果和栈顶值进行‘与’操作
LDF<=   D18     D20 // 加载浮点数 D18D19 和 D20D21，并比较 浮点数 D18D19 是否‘小于等于’浮点数 D20D21
ORF<=   D26     D28 // 比较浮点数 D26D27 是否‘小于等于’浮点数 D28D29，并将比较结果和栈顶值进行‘或’操作
AF<=    D22     D24 // 比较浮点数 D22D23 是否‘小于等于’浮点数 D24D25，并将比较结果和栈顶值进行‘与’操作
ORB                 // 将栈顶值与当前能流值进行‘或’操作
ORB                 // 将栈顶值与当前能流值进行‘或’操作
OUT     Y000        // 输出 Y000
```

梯形图：



4.2.5 >

比较指令>

对应命令	注释/说明	操作数（寄存器）
W>	LDW>、AW>、ORW>	K/H/D/CV/TV/AI/AO/V/Z,K/H/D/CT/TV/AI/AO/V/Z
D>	LDD>、AD>、ORD>	K/H/D/CV32,K/H/D/CV32
F>	LDF>、AF>、ORF>	K/D/CV32,K/D/CV32

1. 指令说明

大于指令用于比较两个数值的大小，如果上方的数据大于下方的数据，比较结果为真，触点接通。

比较的两个数是带符号的

W > 比较双字节整数

D > 比较四字节整数

F > 比较实数

在指令表中， 当 1 位于堆栈顶端时，指令执行载入（LD）、与（AND）和或（OR）操作。

2. 指令的有效操作数

输入输出	类型	操作数	描述
（上）	字/双字/实数	K/H/D/CV/TV/AI/AO/V/Z	要参加比较的操作数
（下）	字/双字/实数	同上	同上

3. 实例

指令表：

```
LDW>    D0      D1   // 加载 D0 和 D1，并比较 D0 是否‘大于’D1
ORW>    D4      D5   // 比较 D4 是否‘大于’D5，并将比较结果和栈顶值进行‘或’操作
AW>     D2      D3   // 比较 D2 是否‘大于’D3，并将比较结果和栈顶值进行‘与’操作
LDD>    D6      D8   // 加载 D6D7 和 D8D9，并比较 D6D7 是否‘大于’D8D9
ORD>    D14     D16 // 比较 D14D15 是否‘大于’D16D17，并将比较结果和栈顶值进行‘或’操作
AD>     D10     D12 // 比较 D10D11 是否‘大于’D12D13，并将比较结果和栈顶值进行‘与’操作
LDF>    D18     D20 // 加载浮点数 D18D19 和 D20D21，并比较 浮点数 D18D19 是否‘大于’浮点数 D20D21
ORF>    D26     D28 // 比较浮点数 D26D27 是否‘大于’浮点数 D28D29，并将比较结果和栈顶值进行‘或’操作
AF>     D22     D24 // 比较浮点数 D22D23 是否‘大于’浮点数 D24D25，并将比较结果和栈顶值进行‘与’操作
ORB                 // 将栈顶值与当前能流值进行‘或’操作
ORB                 // 将栈顶值与当前能流值进行‘或’操作
OUT     Y000        // 输出 Y000
```

梯形图：



4.2.6 <

比较指令<

对应命令	注释/说明	操作数（寄存器）
W<	LDW<、AW<、ORW<	K/H/D/CV/TV/AI/AO/V/Z,K/H/D/CT/TV/AI/AO/V/Z
D<	LDD<、AD<、ORD<	K/H/D/CV32,K/H/D/CV32
F<	LDF<、AF<、ORF<	K/D/CV32,K/D/CV32

1. 指令说明

小于指令用于比较两个数值的大小，如果上方的数据小于下方的数据，比较结果为真，触点接通。

比较的两个数是带符号的

W < 比较双字节整数

D < 比较四字节整数

F < 比较实数

在指令表中， 当 1 位于堆栈顶端时，指令执行载入（LD）、与（AND）和或（OR）操作。

2. 指令的有效操作数

输入输出	类型	操作数	描述
（上）	字/双字/实数	K/H/D/CV/TV/AI/AO/V/Z	要参加比较的操作数
（下）	字/双字/实数	同上	同上

3. 实例

指令表：

```
LDW<    D0    D1   // 加载 D0 和 D1，并比较 D0 是否‘小于’D1
ORW<    D4    D5   // 比较 D4 是否‘小于’D5，并将比较结果和栈顶值进行‘或’操作
AW<     D2    D3   // 比较 D2 是否‘小于’D3，并将比较结果和栈顶值进行‘与’操作
LDD<    D6    D8   // 加载 D6D7 和 D8D9，并比较 D6D7 是否‘小于’D8D9
```

```
ORD<      D14      D16 // 比较 D14D15 是否‘小于’D16D17，并将比较结果和栈顶值进行‘或’操作
AD<       D10      D12 // 比较 D10D11 是否‘小于’D12D13，并将比较结果和栈顶值进行‘与’操作
LDF<      D18      D20 // 加载浮点数 D18D19 和 D20D21，并比较 浮点数 D18D19 是否‘小于’浮点数 D20D21
ORF<      D26      D28 // 比较浮点数 D26D27 是否‘小于’浮点数 D28D29，并将比较结果和栈顶值进行‘或’操作
AF<       D22      D24 // 比较浮点数 D22D23 是否‘小于’浮点数 D24D25，并将比较结果和栈顶值进行‘与’操作
ORB                    // 将栈顶值与当前能流值进行‘或’操作
ORB                    // 将栈顶值与当前能流值进行‘或’操作
OUT       Y000         // 输出 Y000
```

梯形图：



4.2.7 CMP/CMPD/CMPF

对应命令	注释/说明	操作数（寄存器）
CMP	单字比较	D/H/K/CV/TV/AI/AO/V/Z, D/H/K/CV/TV/AI/AO/V/Z, Y/M/S(占用连续 3 个位)
CMPD	双字比较	D/H/K, D/H/K, Y/M/S(占用连续 3 个位)
CMPF	浮点比较	D/K, D/K, Y/M/S(占用连续 3 个位)

1. 指令说明

比较 IN1 和 IN2 的大小，比较结果送到 OUT：

若 OUT 为 Y000

当 IN1 > IN2: Y000 为 1

当 IN1 == IN2: Y001 为 1

当 IN1 < IN2: Y002 为 1

2. 指令的有效操作数

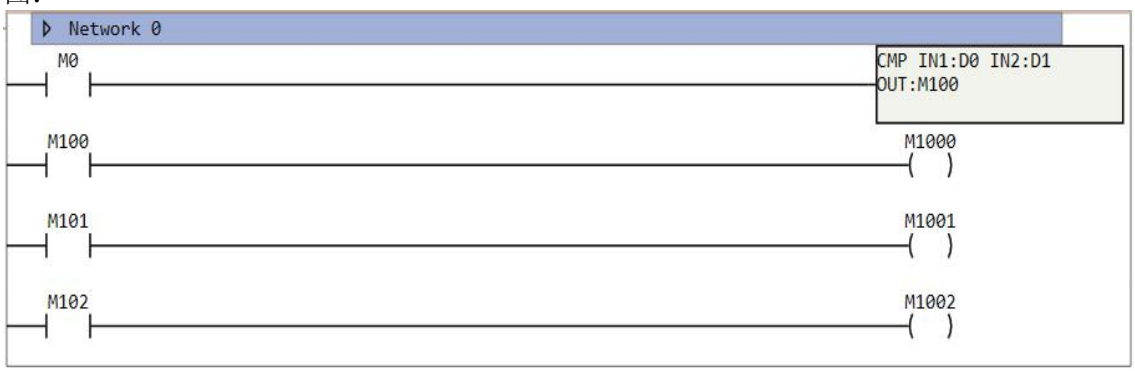
输入输出	类型	操作数	描述
EN	BOOL		启用输入端接通时，指令执行；否则，不执行
IN1，IN2	单字 双字 浮点	D/H/K/TV/CV/AO/AI/V/Z D/H/K, D/K	要参加运算的操作数 1，操作数 2
OUT	BOOL	Y/M/S	计算结果寄存器

3. 实例

指令表：

```
LD      M0
CMP     D0      D1      M100    // 比较 D0 和 D1 的大小
POP
LD      M100
OUT     M1000   // 如果 D0 >  D1, M100= ON
POP
LD      M101
OUT     M1001   // 如果 D0 == D1, M101= ON
POP
LD      M102
OUT     M1002   // 如果 D0 <  D1, M102= ON
POP
```

梯形图：



4.2.8 ZCP/ZCPD/ZCPF

对应命令	注释/说明	操作数（寄存器）
ZCP	单字区间比较	D/H/K/TV/CV/AI/AO/V/Z, D/H/K/TV/CV/AI/AO/V/Z, D/H/K/TV/CV/AI/AO/V/Z, Y/M/S(占用连续 3 个位)
ZCPD	双字区间比较	D/H/K,D/H/K,D/H/K, Y/M/S(占用连续 3 个位)
ZCPF	浮点区间比较	D/K,D/K,D/K, Y/M/S(占用连续 3 个位)

1. 指令说明

比较值 IN3 和下限 IN1 及上限 IN2 作比较，其比较结果在 O 作表示。

下限 IN1 不可以大于上限 IN2。

大小比较是以代数来进行，全部的数据是以有符号数二进制数值来作比较。

2. 指令的有效操作数

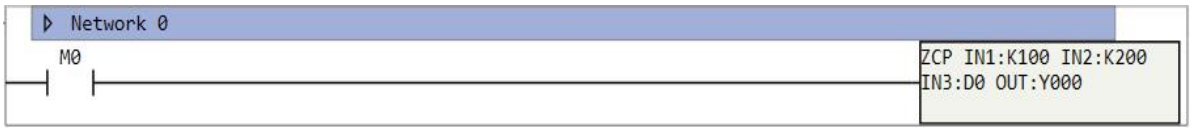
输入输出	类型	操作数	描述
EN	BOOL		**启用输入端接通时，指令执行；否则，不执行**
S1,S2,S3	单字 双字 浮点	**D/H/K/TV/CV/AI/AO/V/Z,** **D/H/K,** **D/K**	**下限设定值，上限设定值，比较值**
D	BOOL	**Y/M/S**	**结果寄存器**

3. 实例

指令表：

```
LD      M0
ZCP     K100    K200    D0      Y000    //    当 D0 < K100 时  Y000 = ON ；
                                              当 K100 <= D0 <= K200 时  Y001 = ON ；
                                              当 D0 > K200 时  Y002 = ON。
```

梯形图：



4.3 转换指令

内容如下：

1、 整数转换为双整数 WTOD

2、 双整数转换为整数 DTOW

3、 双整数转换为浮点数 DTOF

4、 BCD 码转换为整数 BIN

5、 整数转换为 BCD BCD

6、 取整 ROUND

7、 截位 TRUNC

4.3.1 WTOD

1. 指令说明

整数至双整数（WTOD）指令将整数值（IN）转换成双整数值，并将结果置入（OUT）指定的寄存器中。

2. 指令的有效操作数

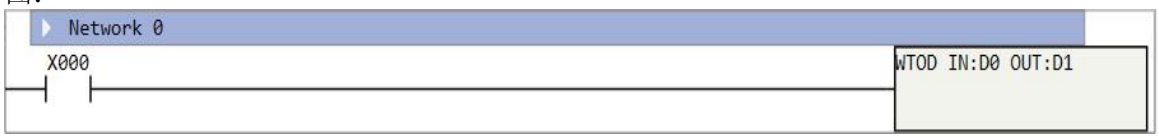
输入输出	类型	操作数	描述
EN	BOOL		**启用输入端接通时，指令执行；否则，不执行**
IN	整数	**D/K/H/TV/CV/AI/AO/V/Z**	**要转换的操作数**
OUT	双整数	**D**	**转换结果寄存器**

3. 实例

指令表：

```
LD      X000
WTOD    D0      D1    // 将整数 D0 转换为双整数并存储在寄存器 D1D2
```

梯形图：



4.3.2 DTOW

1. 指令说明

双整数至整数（DTOW）指令将双整数值（IN）转换成整数值，并将结果置入（OUT）指定的寄存器中。

如果转换的数值过大，数值溢出标志位被置位，且无法在输出中表示。

M8169　　被置位时表示数值溢出。

2. 指令的有效操作数

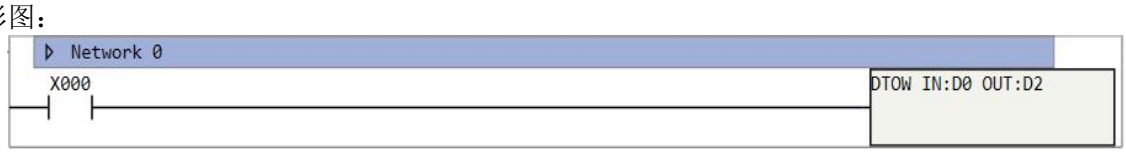
输入输出	类型	操作数	描述
EN	BOOL		**启用输入端接通时，指令执行；否则，不执行**
IN	双整数	**D/K/H**	**要转换的操作数**
OUT	整数	**D/TV/CV/AO/V/Z**	**转换结果寄存器**

3. 实例

指令表：

```
LD      X000
DTOW    D0      D2    // 将双整数 D0D1 转换为整数并存储在寄存器 D2
```

梯形图：



4.3.3 DTOF

1. 指令说明

双整数至浮点数（DTOF）指令将双整数值（IN）转换成浮点数，并将结果置入（OUT）指定的寄存器中。

2. 指令的有效操作数

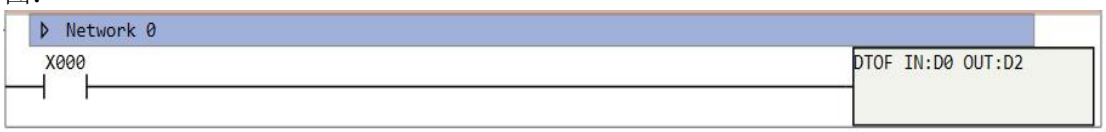
输入输出	类型	操作数	描述
EN	BOOL		**启用输入端接通时，指令执行；否则，不执行**
IN	双整数	**D/K/H**	**要转换的操作数**
OUT	浮点数	**D**	**转换结果寄存器**

3. 实例

指令表：

```
LD      X000
DTOF    D0      D2      // 将双整数 D0D1 转换为整数并存储在寄存器 D2
```

梯形图：



4.3.4 BIN

1. 指令说明

BCD 至整数指令（BIN）将二进制编码的十进制数值（IN）转换成整数，并将结果载入（OUT）指定的寄存器中。IN 的有效范围是 0 至 9999 BCD。

M8168　　被置位时表示无效的 BCD 码转换。

2. 指令的有效操作数

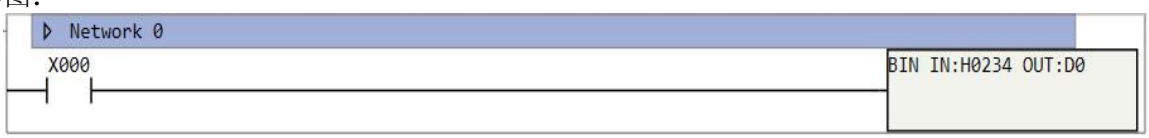
输入输出	类型	操作数	描述
EN	BOOL		**启用输入端接通时，指令执行；否则，不执行**
IN	BCD 码	**D/K/H/TV/CV/AI/AO/V/Z**	**要转换的操作数**
OUT	整数	**D/TV/CV/AO/V/Z**	**转换结果寄存器**

3. 实例

指令表：

```
LD      X000
BIN     H0234   D0 // 将 BCD 码 234 转换为整数并存储在寄存器 D0
```

梯形图：



4.3.5 BCD

1. 指令说明

整数至 BCD 指令（BCD）将整数（IN）转换成二进制编码的十进制数值，并将结果载入（OUT）指定的寄存器中。OUT 的有效范围是 0 至 9999 BCD。

M8168　　被置位时表示无效的 BCD 码转换

2. 指令的有效操作数

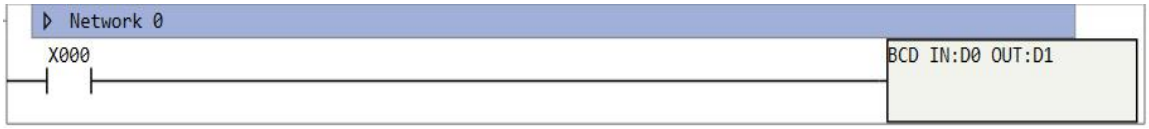
输入输出	类型	操作数	描述
EN	BOOL		**启用输入端接通时，指令执行；否则，不执行**
IN	整数	**D/K/H/TV/CV/AI/AO/V/Z**	**要转换的操作数**
OUT	BCD 码	**D/TV/CV/AO/V/Z**	**转换结果寄存器**

3. 实例

指令表：

```
LD      X000
BCD     D0      D1 // 将整数 D0 转换为 BCD 码并存储在寄存器 D1
```

梯形图：



4.3.6 ROUND

对应命令	注释/说明	操作数（寄存器）
ROUND	实数转换成双整数值	D/K,D

1. 指令说明

指令将实值（IN）转换成双整数值，并将结果置入（OUT）指定的寄存器中。如果小数部分等于或大于 0.5，则进位为整数。

2. 指令的有效操作数

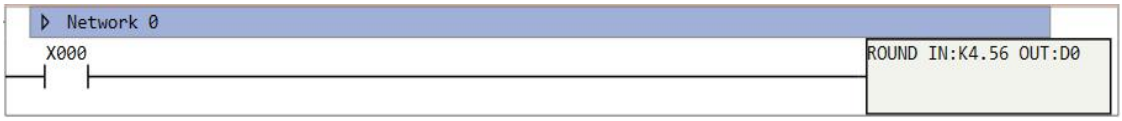
输入输出	类型	操作数	描述
EN	BOOL		**启用输入端接通时，指令执行；否则，不执行**
IN	实数	**D/K**	**要转换的操作数**
OUT	双整数	**D**	**转换结果寄存器**

3. 实例

指令表：

```
LD      X000
ROUND   K4.56   D0 // 将实数 4.56 转换为双整数并存储在寄存器 D0D1 0.56 > 0.5 进位为 1 所以转换结果为 5
```

梯形图：



4.3.7 TRUNC

对应命令	注释/说明	操作数（寄存器）
TRUNC	截位将 32 位实数转换为 32 位双整数	D/K,D

1. 指令说明

指令将 32 位实数（IN）转换成 32 位双整数，并将结果的整数部分置入（OUT）指定的寄存器中。只有实数的整数部分被转换，小数部分被丢弃。

2. 指令的有效操作数

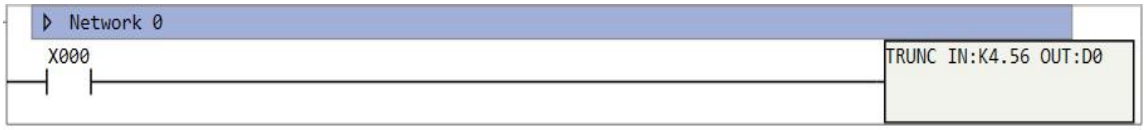
输入输出	类型	操作数	描述
EN	BOOL		**启用输入端接通时，指令执行；否则，不执行**
IN	实数	**D/K**	**要转换的操作数**
OUT	双整数	**D**	**转换结果寄存器**

3. 实例

指令表：

```
LD      X000
TRUNC   K4.56   D0 // 将实数 4.56 转换为双整数并存储在寄存器 D0D1 转换结果为 4
```

梯形图：



4.4 逻辑运算

逻辑运算指令

内容如下：

1、 反转字 INVW
2、 反转双字 INVD
3、 字相与 ANDW
4、 双字相与 ANDD
5、 字相或 ORW
6、 双字相或 ORD
7、 字异或 XORW
8、 双字异或 XORD

4.4.1 INVW/INVD

取反运算

应命令	注释/说明	操作数（寄存器）
INVW	单字取反	K/H/D/TV/CV/AI/AO/V/Z, D/TV/CV/AO/V/Z
INVD	双字取反	K/H/D,D

1. 指令说明

指令对输入（IN）执行取反操作，并将结果载入输出（OUT）寄存器。

2. 指令的有效操作数

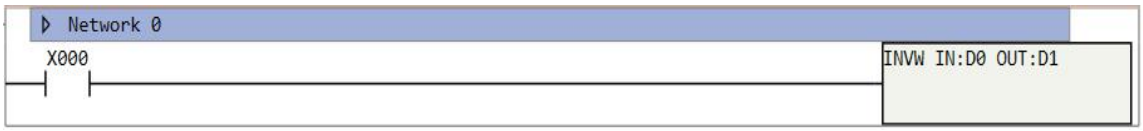
输入输出	类型	操作数	描述
EN	BOOL		**启用输入端接通时，指令执行；否则，不执行**
IN	字	**K/H/D/TV/CV/AI/AO/V/Z**	**要参加运算的操作数 1 和操作数 2**
	双字	**K/H/D**	
OUT	字	**D/TV/CV/AO/V/Z**	**计算结果寄存器**
	双字	**D**	

3. 实例

指令表：

```
LD      X000
INVW    D0      D1 // 将 D0 取反输出到 D1
```

梯形图：



4.4.2 ANDW/ANDD

相与运算

对应命令	注释/说明	操作数（寄存器）
ANDW	单字相与	K/H/D/TV/CV/AI/AO/V/Z, K/H/D/TV/CV/AI/AO/V/Z, D/TV/CV/AO/V/Z
ANDD	双字相与	K/H/D,K/H/D,D

1. 指令说明

指令对输入（IN1）和（IN2）执行与操作，并将结果载入输出（OUT）寄存器。

2. 指令的有效操作数

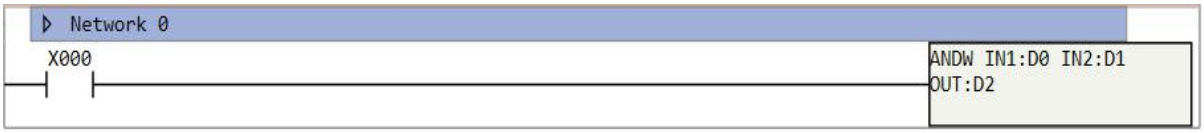
输入输出	类型	操作数	描述
EN	BOOL		启用输入端接通时，指令执行；否则，不执行
IN1,IN2	字	K/H/D/TV/CV/AI/AO/V/Z	要参加运算的操作数 1 和操作数 2
	双字	K/H/D	
OUT	字	D/TV/CV/AO/V/Z	计算结果寄存器
	双字	D	

3. 实例

指令表：

```
LD      X000
ANDW    D0      D1      D2    // 将 D0 与 D1 进行与操作，并将结果输出到 D2
```

梯形图：



4.4.3 ORW/ORD

相或运算

对应命令	注释/说明	操作数（寄存器）
ORW	单字相或	K/H/D/TV/CV/AI/AO/V/Z, K/H/D/TV/CV/AI/AO/V/Z, D/TV/CV/AO/V/Z
ORD	双字相或	K/H/D,K/H/D,D

1. 指令说明

指令对输入（IN1）和（IN2）执行或操作，并将结果载入输出（OUT）寄存器。

2. 指令的有效操作数

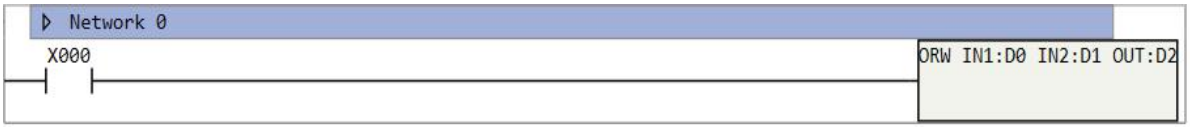
输入输出	类型	操作数	描述
EN	BOOL		**启用输入端接通时，指令执行；否则，不执行**
IN1,IN2	字	**K/H/D/TV/CV/AI/AO/V/Z**	**要参加运算的操作数 1 和操作数 2**
	双字	**K/H/D**	
OUT	字	**D/TV/CV/AO/V/Z**	**计算结果寄存器**
	双字	**D**	

3. 实例

指令表：

```
LD      X000
ORW     D0      D1      D2      // 将 D0 与 D1 进行与操作，并将结果输出到 D2
```

梯形图：



4.4.4 XORW/XORD

异或运算

对应命令	注释/说明	操作数（寄存器）
XORW	单字异或	K/H/D/TV/CV/AI/AO/V/Z, K/H/D/TV/CV/AI/AO/V/Z, D/TV/CV/AO/V/Z
XORD	双字异或	K/H/D,K/H/D,D

1. 指令说明

指令对输入（IN1）和（IN2）执行异或操作，并将结果载入输出（OUT）寄存器。

2. 指令的有效操作数

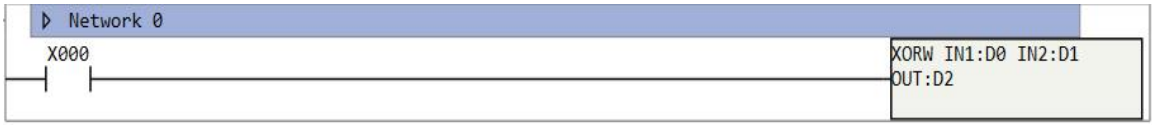
输入输出	类型	操作数	描述
EN	BOOL		**启用输入端接通时，指令执行；否则，不执行**
IN1,IN2	字	**K/H/D/TV/CV/AI/AO/V/Z**	**要参加运算的操作数 1 和操作数 2**
	双字	**K/H/D**	
OUT	字	**D/TV/CV/AO/V/Z**	**计算结果寄存器**
	双字	**D**	

3. 实例

指令表：

```
LD      X000
XORW    D0      D1      D2    // 将 D0 与 D1 进行异或操作，并将结果输出到 D2
```

梯形图：



4.4.5 NEGW/NEGD

求补运算

对应命令	注释/说明	操作数（寄存器）
NEGW	单字求补	D/K/H/AI/AO/CV/TV/V/Z, D/CV/TV/AO/V/Z
NEGD	双字求补	D/K/H, D

1. 指令说明

指令对输入（IN）求出补码，并将结果载入输出（OUT）寄存器。通常情况下，整数的补码即为其相反数。

2. 指令的有效操作数

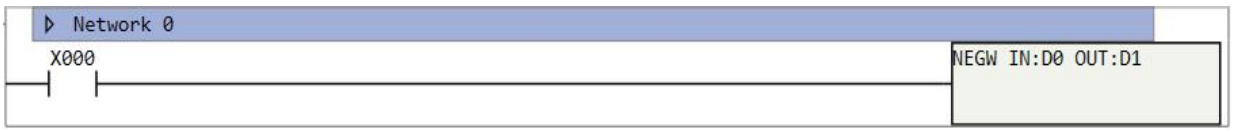
输入输出	类型	操作数	描述
EN	BOOL		**启用输入端接通时，指令执行；否则，不执行**
IN	字	**K/H/D/TV/CV/AI/AO/V/Z**	**要参加运算的操作数 1 和操作数 2**
	双字	**K/H/D**	
OUT	字	**D/TV/CV/AO/V/Z**	**计算结果寄存器**
	双字	**D**	

3. 实例

指令表：

```
LD      X000
NEGW    D0      D1 // 将 D0 求补输出到 D1
```

梯形图：



4.5 传送

传送指令

内容如下：

1、 移动字 MOV

2、 传送双字 MOVD

3、 传送实数 MOVF

4、 成块移动字 MVBLK

5、 成块移动双字 MVDBLK

4.5.1 MOV/MOVD/MOVF

移动操作

对应命令	注释/说明	操作数（寄存器）
MOV	移动字	D/TV/CV/K/H/AI/AO/V/Z,D/TV/CV/AO/V/Z
MOVD	传送双字	D/K/H/CV32,D/CV32
MOVF	传送实数	D/K/AI/AO,D

1. 指令说明

指令将输入值（S）复制至输出寄存器（D），不改变原来的数值。

2. 指令的有效操作数

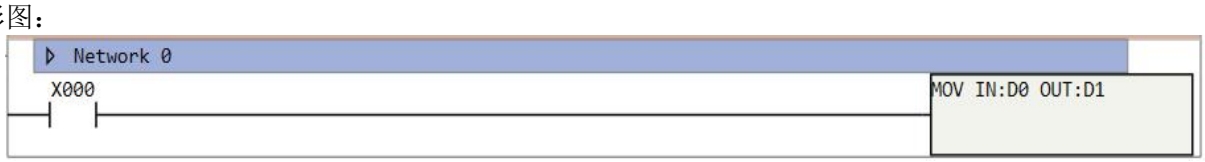
输入输出	类型	操作数	描述
EN	BOOL		**启用输入端接通时，指令执行；否则，不执行**
S	字 双字 浮点	**D/TV/CV/K/H/AI/AO/V/Z** **D/K/H/CV** **D/K/AI/AO**	**要复制传送的值**
D	字 双字 浮点	**D/TV/CV/AO/V/Z** **D/CV** **D**	**输出的目标寄存器**

3. 实例

指令表：

```
LD      X000
MOV     D0      D1    // D0 复制到 D1
```

梯形图：



4.5.2 MVBLK/MVDBLK

1. 指令说明

指令将寄存器数目（N）从输入地址（S）复制至输出地址（D）。N 的范围是 1 至 1024。

2. 指令的有效操作数

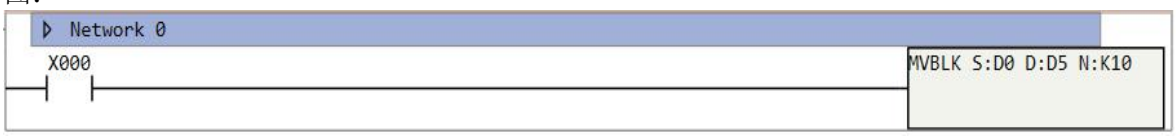
输入输出	类型	操作数	描述
EN	BOOL		启用输入端接通时，指令执行；否则，不执行
S	字 双字	D/TV/CV/AI/AO/V/Z D/TV/CV/AI/AO/V/Z	要复制传送的字的起始地址
D	字 双字	D/TV/CV/AO/V/Z D/TV/CV/AO/V/Z	输出的目标寄存器的起始地址
N	整数	D/K/H	复制传送的字的个数

3. 实例

指令表：

```
LD      X000
MVBLK   D0      D5      K10   // 将 D0~D9 这 10 个字复制到 D5~D14 的 10 个字
```

梯形图：



4.5.3 FMOV/FMOVD

多点传送

对应命令	注释/说明	操作数（寄存器）
FMOV	单字多点传送	D/H/K/AI/AO/TV/CV/V/Z, D/AO/CV/TV/V/Z, D/K/H
FMOVD	双字多点传送	D/H/K,　D, D/K/H

1. 指令说明

将源元件（S）中的数据传送到指定目标（D）开始的（N）个目标元件中，这 N 个元件中的数据完全相同。

2. 指令的有效操作数

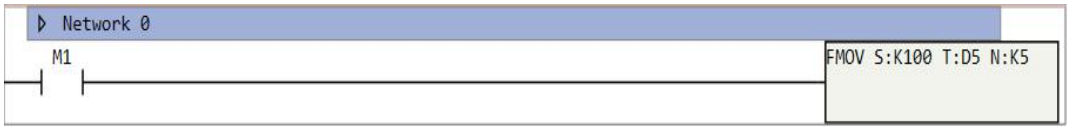
输入输出	类型	操作数	描述
EN	BOOL		启用输入端接通时，指令执行；否则，不执行
S	字 双字	D/H/K/AO/AI/CV/TV/V/Z D/H/K	要参加计算的操作数
D	字 双字	D/AO/CV/TV/V/Z D	传送目标寄存器起始地址
N	字 双字	D/H/K D/H/K	移位传送的目标寄存器的数量

3. 实例

指令表：

```
LD      M1
FMOV    K100    D5      K5      // 执行结果 D5—D9 的值均为 100
```

梯形图：



4.5.4 SMOV

1. 指令说明

指令将首先将二进制的数据（S）转换成 BCD 码，然后将 BCD 码移位传送，实现数据的分配、组合。源数据（S）的 BCD 码右起从第（N1）位开始的（N2）位移送到数据（D）的第（N3）位开始的（N2）位，而目数据（D）的其他位 BCD 码不变。然后，数据（D）中的 BCD 码自动转换成二进制数，即为 数据（D）的内容。BCD 码值超过 9999 时出错。

2. 指令的有效操作数

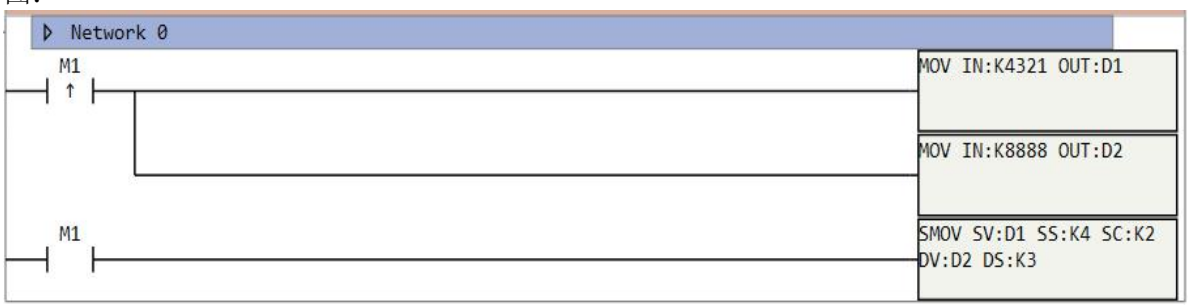
输入输出	类型	操作数	描述
EN	BOOL		启用输入端接通时，指令执行；否则，不执行
S	字	D/H/K/AO/AI/CV/TV/V/Z	要参加计算的操作数
D	字	D/AO/CV/TV/V/Z	计算结果寄存器
N1	字	H/K	源数据 S 的 BCD 码移位传送的起始位数
N2	字	H/K	源数据 S 的 BCD 码移位传送的位数
N3	字	H/K	目标数据 D 的 BCD 码接收的起始位数

3. 实例

指令表：

```
LDP     M1
MOV     K4321   D1
MOV     K8888   D2
POP
LD      M1
SMOV    D1      K4      K2      D2      K3      //当 M1 为‘1’时，执行结果 D2 = 8438
```

梯形图：



4.5.5 XCH/XCHD/XCHF

对应命令	注释/说明	操作数（寄存器）
XCH	单字交换	OUT1：D/AO/CV/TV/V/Z, OUT2：D/AO/CV/TV/V/Z
XCHD	双字交换	OUT1：D OUT2：D
XCHF	浮点交换	OUT1：D OUT2：D

1. 指令说明

交换寄存器（O1），寄存器（O2）的值。

2. 指令的有效操作数

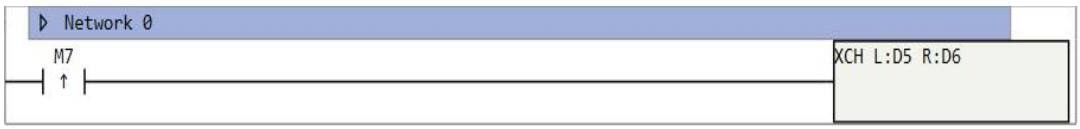
输入输出	类型	操作数	描述
EN	BOOL		**启用输入端接通时，指令执行；否则，不执行**
O1，O2	单字 双字 浮点	**D/AO/CV/TV/V/Z** **D** **D**	**要参加计算的寄存器 1，寄存器 2**

3. 实例

指令表：

```
    LDP     M7
    XCH     D5      D6              // 若该指令执行之前 D5 = 100， D6 = 50， 则执行该指令后 D5 = 50, D6 = 100;
```

梯形图：



4.6 浮点数运算

内容如下：

1、 浮点加运算 ADDF

2、 浮点减运算 SUBF

3、 浮点乘运算 MULF

4、 浮点除运算 DIVF

5、 平方根 SQRT

6、 正弦 SIN

7、 余弦 COS

8、 正切 TAN

9、 自然对数 LN

10、自然指数 EXP

4.6.1 ADDF

1. 指令说明

指令将两个 32 位实数相加，并产生一个 32 位实数结果（OUT）。
IN1 + IN2 = OUT
M8170　　负结果
M8171　　零结果

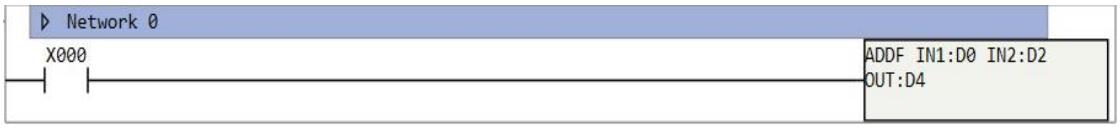
2. 指令的有效操作数

输入输出	类型	操作数	描述
EN	BOOL		启用输入端接通时，指令执行；否则，不执行
IN1	实数	K/D	要参加运算的操作数 1
IN2	实数	K/D	要参加运算的操作数 2
OUT	实数	D	计算结果寄存器

3. 实例
指令表：

```
LD      X000
ADDF    D0      D2      D4    // D0D1 + D2D3 = D4D5
```

梯形图：



4.6.2 SUBF

1. 指令说明

指令将两个 32 位实数相减，并产生一个 32 位实数结果（OUT）。
IN1 - IN2 = OUT
M8170　　负结果
M8171　　零结果

2. 指令的有效操作数

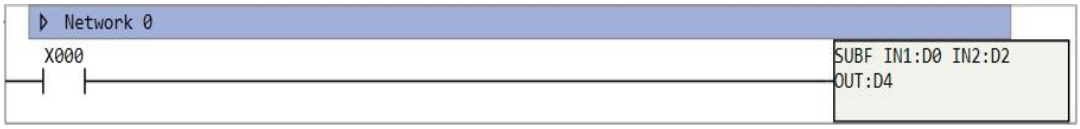
输入输出	类型	操作数	描述
EN	BOOL		启用输入端接通时，指令执行；否则，不执行
IN1	实数	K/D	要参加运算的操作数 1
IN2	实数	K/D	要参加运算的操作数 2
OUT	实数	D	计算结果寄存器

3. 实例

指令表：

```
LD      X000
SUBF    D0      D2      D4    // D0D1 - D2D3 = D4D5
```

梯形图：



4.6.3 MULF

1. 指令说明

指令将两个 32 位实数相乘，并产生一个 32 位实数结果（OUT）。

IN1 × IN2 = OUT

M8170　　负结果

M8171　　零结果

2. 指令的有效操作数

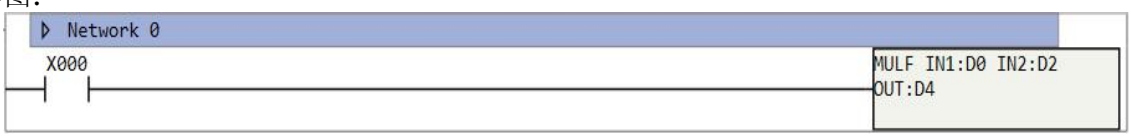
输入输出	类型	操作数	描述
EN	BOOL		启用输入端接通时，指令执行；否则，不执行
IN1	实数	K/D	要参加运算的操作数 1
IN2	实数	K/D	要参加运算的操作数 2
OUT	实数	D	计算结果寄存器

3. 实例

指令表：

```
LD      X000
MULF    D0      D2      D4    // D0D1 × D2D3 = D4D5
```

梯形图：



4.6.4 DIVF

1. 指令说明

指令将两个 32 位实数相除，并产生一个 32 位实数结果（OUT）。
IN1 ÷ IN2 = OUT
M8170　负结果
M8171　零结果
M8172　除数为零

2. 指令的有效操作数

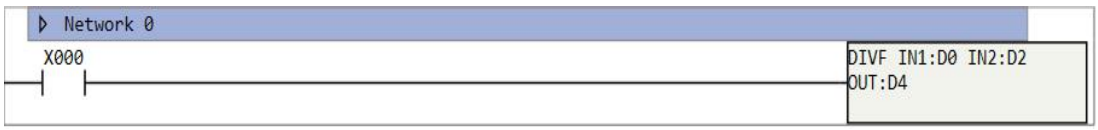
输入输出	类型	操作数	描述
EN	BOOL		启用输入端接通时，指令执行；否则，不执行
IN1	实数	K/D	要参加运算的操作数 1
IN2	实数	K/D	要参加运算的操作数 2
OUT	实数	D	计算结果寄存器

3. 实例

指令表：

```
LD      X000
DIVF    D0      D2      D4    // D0D1 ÷ D2D3 = D4D5
```

梯形图：



4.6.5 SQRT

1. 指令说明

指令对 32 位实数（IN）取平方根，并产生一个 32 位实数结果（OUT）
SQRT（IN）= OUT
M8171　零结果

2. 指令的有效操作数

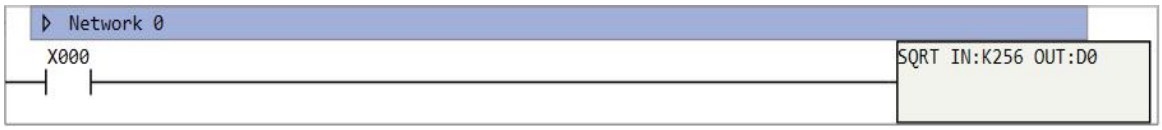
输入输出	类型	操作数	描述
EN	BOOL		启用输入端接通时，指令执行；否则，不执行
IN	实数	K/D/AI/AO	要参加运算的操作数
OUT	实数	D	计算结果寄存器

3. 实例

指令表：

```
LD      X000
SQRT    K256    D0    // SQRT(256) = D0D1 = 16
```

梯形图：



4.6.6 SIN

1. 指令说明

指令对角度值（IN）进行三角运算，并将结果放置在（OUT）中。输入角以角度为单位。欲将输入角从弧度转换成角度，用弧度除以 1.745329E-2（约等于 PI/180，PI = 3.1415926）。

SIN（IN）= OUT

M8170　　负结果

M8171　　零结果

2. 指令的有效操作数

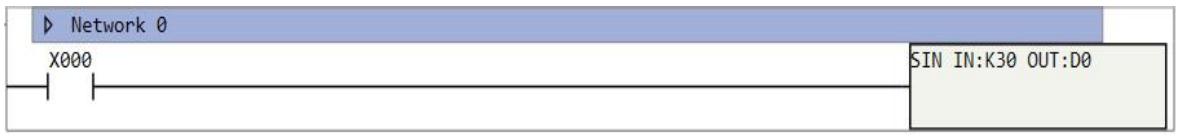
输入输出	类型	操作数	描述
EN	BOOL		**启用输入端接通时，指令执行；否则，不执行**
IN	实数	**K/D/AI/AO**	**要参加运算的操作数**
OUT	实数	**D**	**计算结果寄存器**

3. 实例

指令表：

```
LD      X000
SIN     K30     D0    // SIN(30°) = D0D1 = 0.5
```

梯形图：



4.6.7 COS

1. 指令说明

指令对角度值（IN）进行三角运算，并将结果放置在（OUT）中。输入角以角度为单位。欲将输入角从弧度转换成角度，用弧度除以 1.745329E-2（约等于 PI/180，PI = 3.1415926）。

COS（IN）= OUT

M8170　　负结果

M8171　　零结果

2. 指令的有效操作数

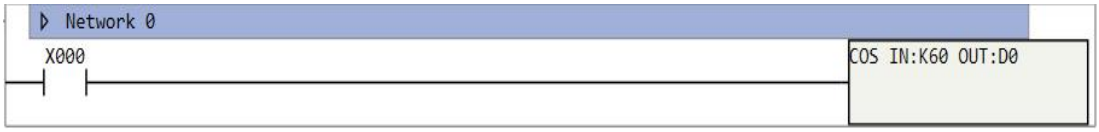
输入输出	类型	操作数	描述
EN	BOOL		**启用输入端接通时，指令执行；否则，不执行**
IN	实数	**K/D/AI/AO**	**要参加运算的操作数**
OUT	实数	**D**	**计算结果寄存器**

3. 实例

指令表：

```
LD      X000
COS     K60     D0    // COS(60°) = D0D1 = 0.5
```

梯形图：



4.6.8 TAN

1. 指令说明

指令对角度值（IN）进行三角运算，并将结果放置在（OUT）中。输入角以角度为单位。欲将输入角从弧度转换成角度，用弧度除以 1.745329E-2（约等于 PI/180，PI = 3.1415926）。

TAN（IN）= OUT

M8169　溢出

M8170　负结果

M8171　零结果

2. 指令的有效操作数

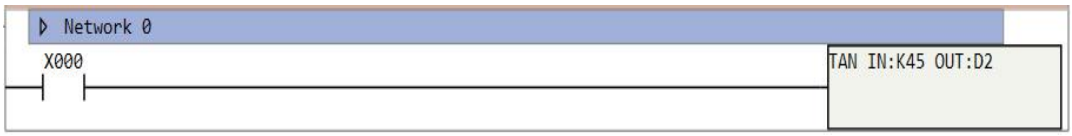
输入输出	类型	操作数	描述
EN	BOOL		**启用输入端接通时，指令执行；否则，不执行**
IN	实数	**K/D/AI/AO**	**要参加运算的操作数**
OUT	实数	**D**	**计算结果寄存器**

3. 实例

指令表：

```
LD      X000
TAN     K45     D2    // TAN(45°) = D2D3 = 1
```

梯形图：



4.6.9 LN

1. 指令说明

指令对（IN）中的数值进行自然对数计算，并将结果置于（OUT）中。

LN(IN) = OUT

欲从自然对数获得以 10 为底数的对数，用自然对数除以 2.302585（约等于 10 的自然对数）。

欲将任何实数提升为另一个实数的乘幂，包括分数指数：将"自然指数"指令与"自然对数"指令相结合。例如，欲将 X 提升为 Y 乘幂，输入以下指令：EXP (Y * LN (X))。

2. 指令的有效操作数

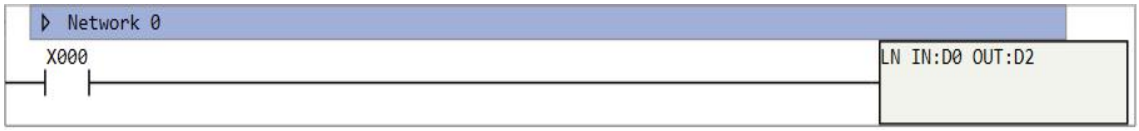
输入输出	类型	操作数	描述
EN	BOOL		**启用输入端接通时，指令执行；否则，不执行**
IN	实数	**K/D/AI/AO**	**要参加运算的操作数**
OUT	实数	**D**	**计算结果寄存器**

3. 实例

指令表：

```
LD      X000
LN      D0      D2    // LN（D0D1） = D2D3
```

梯形图：



4.6.10 EXP

1. 指令说明

指令进行 e 的 IN 次方指数计算，并将结果置于 OUT 中。

EXP(IN) = OUT

欲将任何实数提升为另一个实数的乘幂，包括分数指数：将"自然指数"指令与"自然对数"指令相结合。例如欲将 X 提升为 Y 乘幂，输入以下指令：

EXP (Y * LN (X))。

例：5 的立方 = 5^3=EXP(3*LN(5))=125

2. 指令的有效操作数

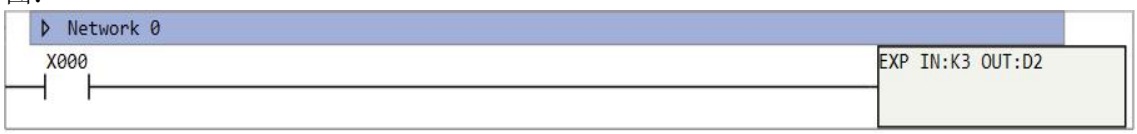
输入输出	类型	操作数	描述
EN	BOOL		**启用输入端接通时，指令执行；否则，不执行**
IN	实数	**K/D/AI/AO**	**要参加运算的操作数**
OUT	实数	**D**	**计算结果寄存器**

3. 实例

指令表：

```
LD      X000
EXP     K3      D2    // EXP（3） = D2D3 = e^3
```

梯形图：



4.6.11 LOG

1. 指令说明

对寄存器（I）的浮点数值进行以 10 为底的对数运算，将浮点数结果传送到寄存器（O）。

2. 指令的有效操作数

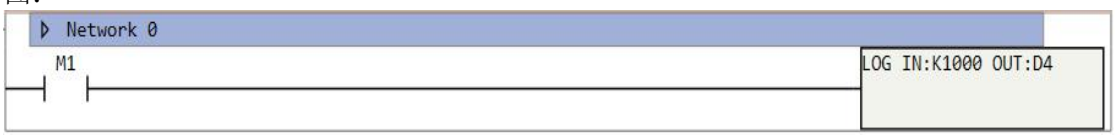
输入输出	类型	操作数	描述
EN	BOOL		启用输入端接通时，指令执行；否则，不执行
I	浮点	D/K/AI/AO	要参加运算的操作数
O	浮点	D	计算结果寄存器

3. 实例

指令表：

```
LD      M1
LOG     K1000    D4// 执行结果 D4（D5） = 3.0
```

梯形图：



4.6.12 POW

1. 指令说明

以寄存器（B）的浮点数值为底，寄存器（E）的浮点数值为指数进行幂运算，将结果传送到寄存器（P）。

2. 指令的有效操作数

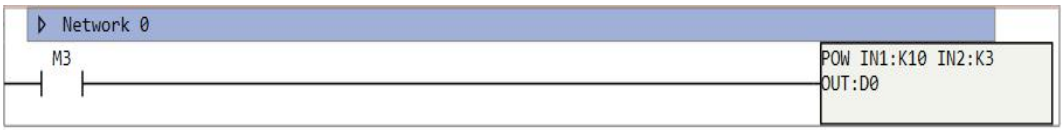
输入输出	类型	操作数	描述
EN	BOOL		启用输入端接通时，指令执行；否则，不执行
B	浮点	D/K	要参加运算的操作数(底数)
E	浮点	D/K	要参加运算的操作数(指数)
P	浮点	D	计算结果（幂）寄存器

3. 实例

指令表：

```
LD      M3
POW     K10     K3      D0      // 执行结果 D0(D1) = 1000.0
```

梯形图：



4.6.13 ABSF

1. 指令说明

指令（ABSF）对给定的 32 位的浮点数输入（IN），输出一个 32 位的浮点数结果（OUT）。根据输入的正负性，输入为正时输出原值，输入为负时输出相反值。

2.指令的有效操作数

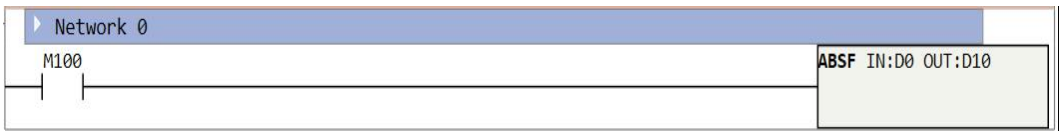
输入输出	类型	操作数	描述
EN	BOOL		**启用输入端接通时，指令执行；否则，不执行**
IN	浮点	**K/D**	**要参加运算的操作数**
OUT	浮点	**D**	**计算结果寄存器**

3. 实例

指令表：

```
LD      M100
ABSF    D0      D10
```

梯形图：



4.6.14 ASIN/ACOS/ATAN

1. 指令说明

ASIN 指令对正弦值（IN）进行反正弦运算，并将结果放置在（OUT）中。输入的值的范围是[-1, 1]，输出的值的范围是[-90, 90]。

ACOS 指令对正弦值（IN）进行反余弦运算，并将结果放置在（OUT）中。输入的值的范围是[-1, 1]，输出的值的范围是[0, 180]。

ATAN 指令对正弦值（IN）进行反正切运算，并将结果放置在（OUT）中。输入的值的范围是[-∞, +∞]，输出的值的范围是[-90, 90]。

ASIN，ACOS，ATAN 指令比较耗时，请谨慎使用。

2. 指令的有效操作数

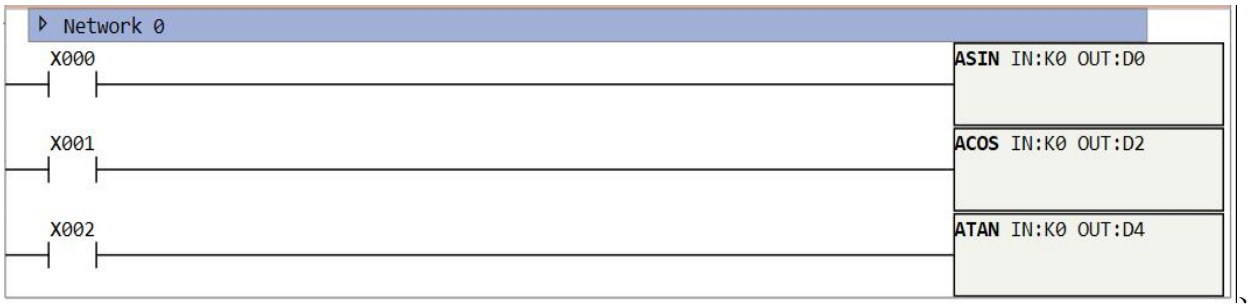
输入输出	类型	操作数	描述
EN	BOOL		**启用输入端接通时，指令执行；否则，不执行**
IN	实数	**K/D/AI/AO**	**要参加运算的操作数**
OUT	实数	**D**	**计算结果寄存器**

3. 实例

指令表：

```
LD      X000
ASIN K0      D0    // ASIN(0) = 0°
POP
LD      X001
ACOS K0      D2    // ACOS(0) = 90°
POP
LD      X002
ATAN K0      D4    // ATAN(0) = 0°
POP
```

梯形图：



4.6.15 RAD/DEG

1. 指令说明

从 0 开始，角度 360 转一圈，弧度 2π转一圈。
RAD 指令将角度值（IN）转换成弧度值（OUT）。转换公式 OUT = IN ×π/ 180。
DEG 指令对弧度值（IN）转换成角度值（OUT）。转换公式 OUT = IN / π×180。

2. 指令的有效操作数

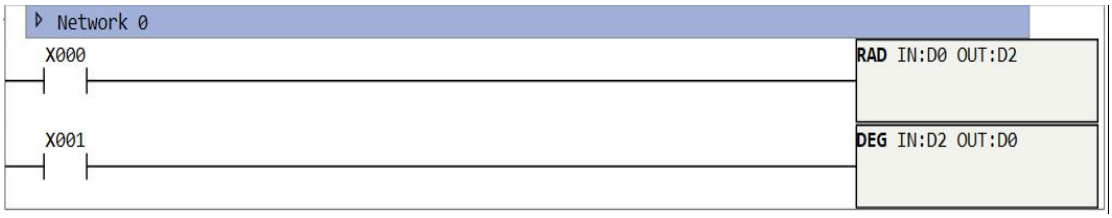
输入输出	类型	操作数	描述
EN	BOOL		**启用输入端接通时，指令执行；否则，不执行**
IN	实数	**K/D/AI/AO**	**要参加运算的操作数**
OUT	实数	**D**	**计算结果寄存器**

3. 实例

指令表：

```
LD      X000
RAD     D0      D2
POP
LD      X001
DEG     D2      D0
POP
```

梯形图:



4.7 整数运算

整数运算指令

内容如下：

1、 16 位整数加 ADD
2、 双整数加 ADDD
3、 16 位整数减 SUB
4、 双整数减 SUBD
5、 乘 MUL
6、 16 位乘 MULW
7、 32 位乘 MULD
8、 除 DIV
9、 16 位除 DIVW
10、32 位除 DIVD
11、字加一 INC
12、双字加一 INCD
13、字减一 DEC
14、双字减一 DECD

4.7.1 ADD/ADDD

加法运算

对应命令	注释/说明	操作数（寄存器）
ADD	16 位整数加得到 16 位整数	K/H/D/TV/CV/AI/AO/V/Z, K/H/D/TV/CV/AI/AO/V/Z, D/TV/CV/AO/V/Z
ADDD	32 位整数加	K/H/D/CV32,K/H/D/CV32,D/CV32

1. 指令说明

指令（ADD）将两个 16 位的整数相加，并产生一个 16 位的整数结果（OUT）。指令（ADDD）的操作数为 32 位。

IN1 ＋ IN2 ＝ OUT

M8169 溢出

M8170 负结果

M8171 零结果

2. 指令的有效操作数

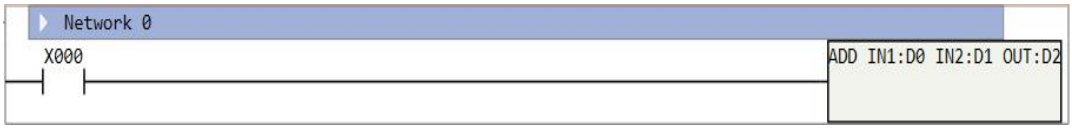
输入输出	类型	操作数	描述
EN	BOOL		**启用输入端接通时，指令执行；否则，不执行**
IN1,IN2	字 双字	**K/H/D/TV/CV/AI/AO/V/Z** **K/H/D/CV**	**要参加运算的操作数 1 和操作数 2**
OUT	字 双字	**D/TV/CV/AO/V/Z** **D/CV**	**计算结果寄存器**

3. 实例

指令表：

```
LD      X000
ADD     D0      D1      D2      // D0 + D1 = D2
```

梯形图：



4.7.2 SUB/SUBD

减法运算

对应命令	注释/说明	操作数（寄存器）
SUB	16 位整数减得到 16 位整数	K/H/D/TV/CV/AI/AO/V/Z, K/H/D/TV/CV/AI/AO/V/Z, D/TV/CV/AO/V/Z
SUBD	32 位整数减	K/H/D/CV32,K/H/D/CV32,D/CV32

1. 指令说明

指令（SUB）将两个 16 位的整数相减，并产生一个 16 位的整数结果（OUT）。指令（SUBD）的操作数为 32 位。

IN1 - IN2 = OUT

M8169 溢出

M8170 负结果

M8171 零结果

2. 指令的有效操作数

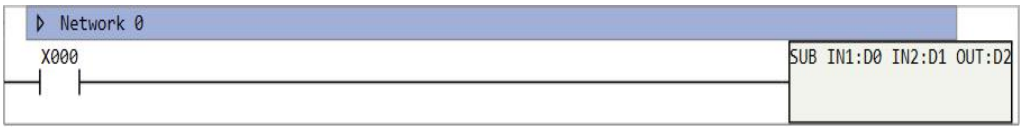
输入输出	类型	操作数	描述
EN	BOOL		启用输入端接通时，指令执行；否则，不执行
IN1,IN2	字 双字	K/H/D/TV/CV/AI/AO/V/Z K/H/D/CV	要参加运算的操作数 1 和操作数 2
OUT	字 双字	D/TV/CV/AO/V/Z D/CV	计算结果寄存器

3. 实例

指令表：

```
LD      X000
SUB     D0      D1      D2      // D0 - D1 = D2
```

梯形图：



4.7.3 MUL/MULW/MULD

乘法运算

对应命令	注释/说明	操作数（寄存器）
MUL	乘	K/H/D/TV/CV/AI/AO/V/Z,K/H/D/TV/CV/AI/AO/V/Z,D
MULW	16 位乘	K/H/D/TV/CV/AI/AO/V/Z, K/H/D/TV/CV/AI/AO/V/Z, D/TV/CV/AO/V/Z
MULD	32 位乘	K/H/D/CV32,K/H/D/CV32,D/CV32

1. 指令说明

指令（MUL）将两个 16 位整数相乘，并产生一个 32 位的结果（OUT）。
指令（MULW）将两个 16 位整数相乘，并产生一个 16 位的结果。
指令（MULD）将两个 32 位整数相乘，并产生一个 32 位的结果。
IN1 × IN2 = OUT
M8170　　负结果
M8171　　零结果

2. 指令的有效操作数

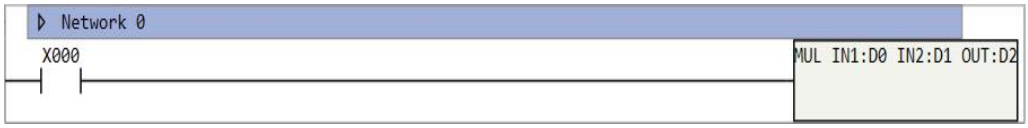
输入输出	类型	操作数	描述
EN	BOOL		启用输入端接通时，指令执行；否则，不执行
IN1,IN2	字 双字	K/H/D/TV/CV/AI/AO/V/Z K/H/D/CV	要参加运算的操作数 1 和操作数 2
OUT	字 双字	D/TV/CV/AO/V/Z D/CV	计算结果寄存器

3. 实例

指令表：

```
LD      X000
MUL     D0      D1      D2      // D0 × D1 = D2D3
```

梯形图：



4.7.4 DIV/DIVW/DIVD

除法运算

对应命令	注释/说明	操作数（寄存器）
DIV	除	K/H/D/TV/CV/AI/AO/V/Z,K/H/D/TV/CV/AI/AO/V/Z,D
DIVW	16 位除	K/H/D/TV/CV/AI/AO/V/Z, K/H/D/TV/CV/AI/AO/V/Z, D/TV/CV32/AO/V/Z
DIVD	32 位除	K/H/D/CV32,K/H/D/CV32,D/CV32

1. 指令说明

指令（DIV）将两个 16 位整数相除，并产生一个 32 位的结果（OUT），其中低 16 位存商，高 16 位存余数。

指令（DIVW）将两个 16 位整数相除，并产生一个 16 位的商，余数被省略。

指令（DIVD）将两个 32 位整数相除，并产生一个 32 位的商。

IN1 ÷ IN2 = OUT

M8169　溢出

M8170　负结果

M8171　零结果

M8172　除数为零

2. 指令的有效操作数

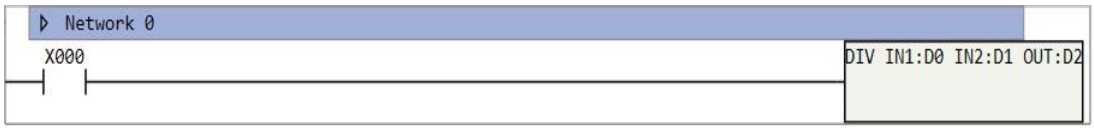
输入输出	类型	操作数	描述
EN	BOOL		**启用输入端接通时，指令执行；否则，不执行**
IN1,IN2	字 双字	**K/H/D/TV/CV/AI/AO/V/Z K/H/D/CV**	**要参加运算的操作数 1 和操作数 2**
OUT	字 双字	**D/TV/CV/AO/V/Z D/CV**	**计算结果寄存器**

3. 实例

指令表：

```
LD      X000
DIV     D0      D1      D2    // D0 ÷ D1 = D2D3 D2 存商  D3 存余数
```

梯形图：



4.7.5 INC/INCD

整数加一

对应命令	注释/说明	操作数（寄存器）
INC	字加 1	K/H/D/TV/CV/AO/V/Z, D/TV/CV/AO/V/Z
INCD	双字加 1	K/H/D/CV32,D/CV32

1. 指令说明

指令（INC）16 位整数加一，并产生一个 16 位的结果（OUT）。指令（INCD）操作 32 位整数。

IN + 1 = OUT

M8169　　溢出

M8170　　负结果

M8171　　零结果

2. 指令的有效操作数

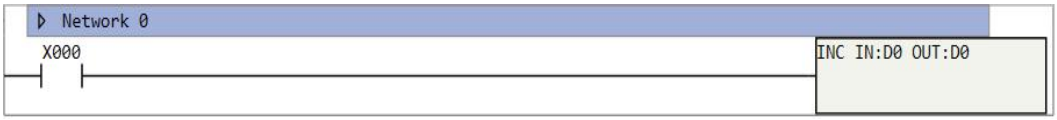
输入输出	类型	操作数	描述
EN	BOOL		**启用输入端接通时，指令执行；否则，不执行**
S	字 双字	**D/TV/CV/K/H/AI/AO/V/Z** **D/K/H/CV**	**要参加运算的操作数**
D	字 双字	**D/TV/CV/AO/V/Z** **D/CV**	**计算结果寄存器**

3. 实例

指令表：

```
LD      X000
INC     D0      D0    // D0 + 1 = D0
```

梯形图：



4.7.6 DEC/DECD

整数减一

对应命令	注释/说明	操作数（寄存器）
DEC	字减 1	K/H/D/TV/CV/AO/V/Z, D/TV/CV/AO/V/Z
DECD	双字减 1	K/H/D/CV32,D/CV32

1. 指令说明

指令（DEC）16 位整数减一，并产生一个 16 位的结果（OUT）。指令（INCD）操作 32 位整数。

IN - 1 = OUT

M8169　　溢出

M8170　　负结果

M8171　　零结果

2. 指令的有效操作数

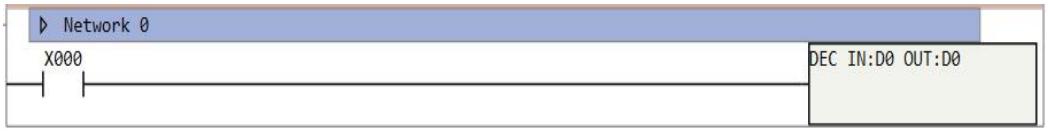
输入输出	类型	操作数	描述
EN	BOOL		启用输入端接通时，指令执行；否则，不执行
S	字 双字	D/TV/CV/K/H/AI/AO/V/Z D/K/H/CV	要参加运算的操作数
D	字 双字	D/TV/CV/AO/V/Z D/CV	计算结果寄存器

3. 实例

指令表：

```
LD      X000
DEC     D0      D0    // D0 - 1 = D0
```

梯形图：



4.7.7 FACT

1. 指令说明

计算寄存器（I）值的阶乘，N 的最大值为 12，将结果传送到寄存器（O）。

2. 指令的有效操作数

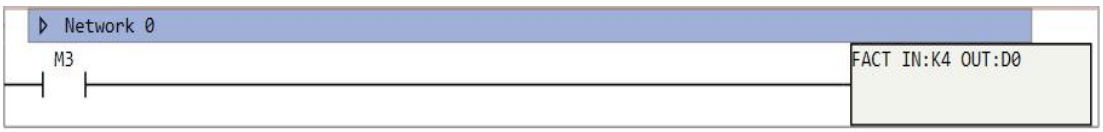
输入输出	类型	操作数	描述
EN	BOOL		启用输入端接通时，指令执行；否则，不执行
I	整数	D/H/K/TV/CV/AO/AI/V/Z	要参加运算的操作数
O	双整数	D	计算结果寄存器

3. 实例

指令表：

```
LD      M3
FACT    K4      D0      //执行结果 D0（D1） = 24
```

梯形图：



4.7.8 ABS/ABSD

1. 指令说明

指令（ABS）对给定的 16 位的整数输入（IN），输出一个 16 位的整数结果（OUT）。根据输入的正负性，输入为正时输出原值，输入为负时输出相反值。
指令（ABSD）的操作数为 32 位。

2. 指令的有效操作数

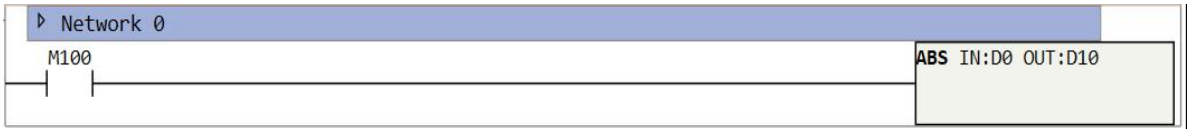
输入输出	类型	操作数	描述
EN	BOOL		**启用输入端接通时，指令执行；否则，不执行**
IN	字 双字	**D/H/K/TV/CV/AO/AI/V/Z K/H/D/CV**	**要参加运算的操作数**
OUT	字 双字	**D/TV/CV/AO/V/Z D/CV**	**计算结果寄存器**

3. 实例

指令表：

```
LD      M100
ABS     D0      D10
```

梯形图：



4.8 定时器

定时器指令

内容如下：

1. 接通延时定时器 TON
2. 接通延时记忆定时器 TONR
3. 断开延时定时器 TOF

4.8.1 TON

1. 指令说明

启用输入端为“接通”时，定时器位（Tx）为“断开”，当前值（TVx）从 0 开始计时。当前值（TVx）大于或等于预设时间（SV）时，定时器位（Tx）为"接通"，且定时器停止计时；启用输入端为"断开"时，接通延时定时器自动复位，即定时器位（Tx）为 OFF，当前值（TVx）为 0。Tx = T0 至 T255，TVx = TV0 至 TV255。

TON、TONR 和 TOF 定时器有三种分辨率。分辨率由下图所示的定时器编号决定。每一个当前值都是时间基数的倍数。例如，10 毫秒定时器中的计数 50 表示 500 毫秒

定时器类型	分辨率	最大值	定时器编号
TON、TONR、TOF	100ms	3276.7 秒	T0 – T199
	10ms	327.67 秒	T200 – T249
	1ms	32.767 秒	T250 – T255

注意：

TOF，TONR 及 TON 不能共享相同的定时器号码。例如，不能有 TON TV32 D0 和 TOF TV32 D0。

您可以将 TON 用于单间隔计时。

执行复位操作可以复位定时器，即定时器位置 0，定时器当前值置 0。

定时器的刷新方式：

FGs 系列 PLC 对于不同分辨率的定时器，其刷新方式是不同的，从而在使用方法上也有很大的不同，使用时请一定要注意根据使用场合和要求来选择不同分辨率的定时器

（1）**1ms 定时器**：1ms 定时器由系统每隔 1ms 刷新一次，与扫描周期及程序处理无关，它采用的是中断刷新方式，因此当扫描周期大于 1ms 时，在一个周期中可能被多次刷新，其当前值（TVx）及（Tx）在一个扫描周期内不一定保持一致；

（2）**10ms 定时器**：10ms 定时器由系统在每个扫描周期开始时自动刷新，由于在每隔扫描周期只刷新一次，因此在一个扫描周期内定时器位（Tx）和当前值（TVx）始终保持一致；

（3）**100ms 定时器**：100ms 定时器是在定时器指令执行时才被刷新，因此，若 100ms 定时器被激活后，如果不是每个扫描周期都执行定时器指令或在一个扫描周期内多次执行定时器指令，则都会造成计时失准。100ms 定时器仅用在定时器指令在每个扫描周期执行一次的程序中；

定时器的正确使用举例：

（1）**1ms 定时器**：在使用错误方法时，只有当定时器的刷新发生在 T250 的常闭触点执行以后到 T250 的常开触点执行以前的区间时，才能产生宽度为一个扫描周期的 Y0 输出，而这种可能性是极小的；

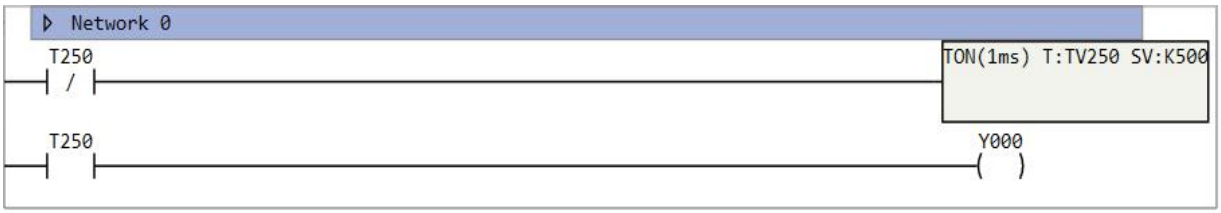


图 1.1 1ms 定时器**错误**用法

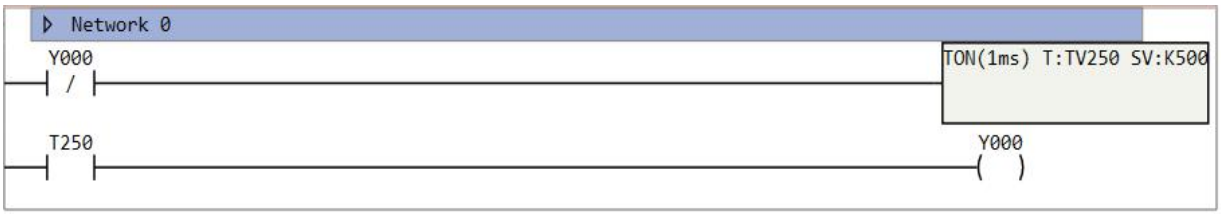


图 1.2 1ms 定时器**正确**用法

（2）**10ms 定时器**：在使用错误方法时，则永远也产生不了 Y0 输出，因为当 10ms 的定时器计时到时，定时器是在每次扫描周期的开始时才刷新，例中当 T200 被置位，但执行到定时器指令时，定时器将被复位（当前值和定时器位都被置 0）；当常开触点 T200 被执行时，T200 永远为 OFF，Y0 也将为 OFF，即永远不会被置位 ON；

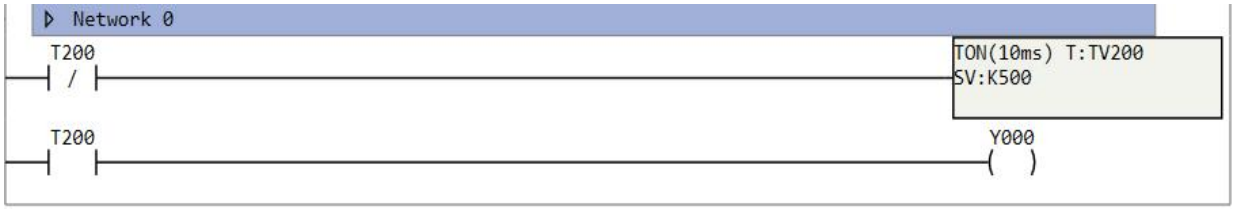


图 1.3 10ms 定时器**错误**用法

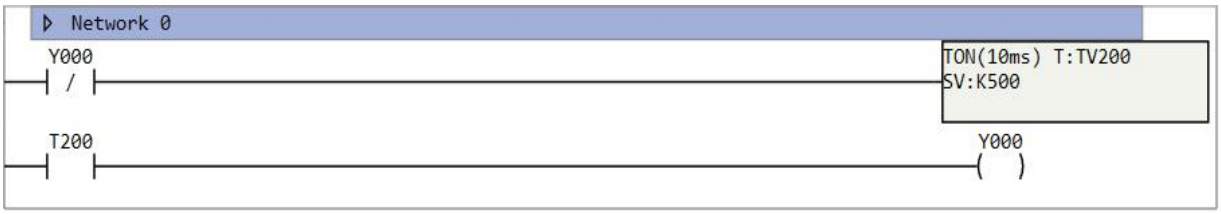


图 1.4 10ms 定时器**正确**用法

（3）**100ms 定时器**：由于 100ms 定时器是在执行指令时刷新，因此，当定时器 T0 到达设定值时，必定会产生一个 Y0 输出；

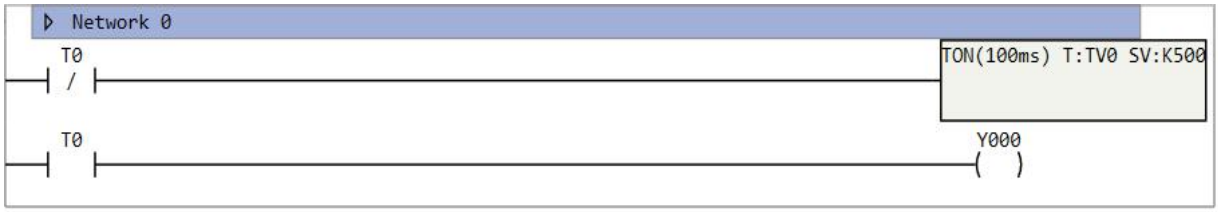


图 1.5 100ms 定时器**错误**用法

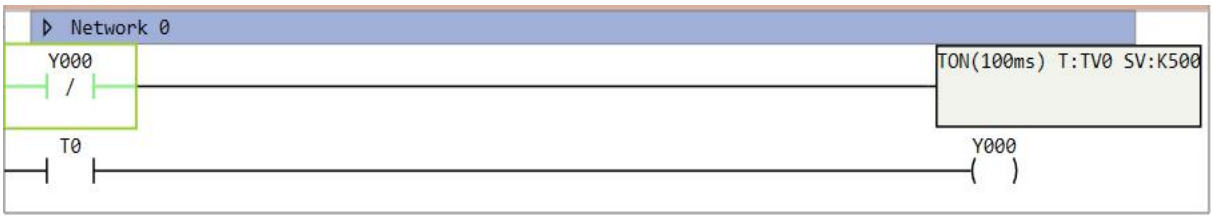


图 1.6 100ms 定时器**正确**用法

2. 指令的有效操作数

输入输出	类型	操作数	描述
EN	BOOL		**启用输入端接通时，指令执行；否则，不执行**
T	双整数	**TV**	**选定的定时寄存器**
SV	双整数	**K/H/D**	**预设定时时间**

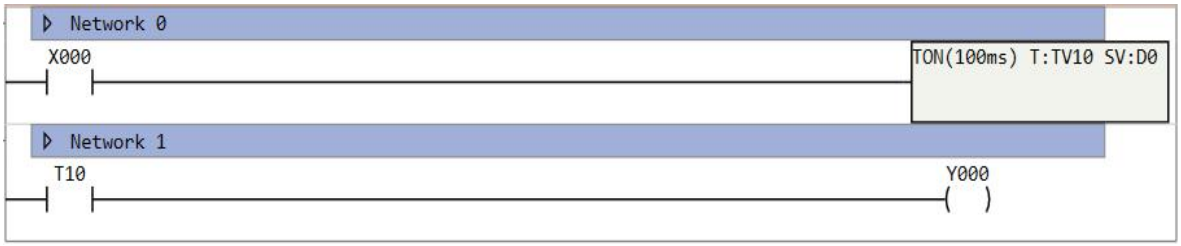
3. 实例

指令表：

```
Network 000
    LD      X000
    TON     TV10    D0      // 定时器定时到 D0 时，T10 = 1 （D0 为变量,也可以用常量代替，如 K1）

Network 001
    LD      T10
    OUT     Y000
```

梯形图：



4.8.2 TONR

1. 指令说明

启用输入端为“接通”时，定时器位（Tx）为“断开”，当前值（TVx）从 0 开始计时。启用输入端为"断开"时，定时器状态位和当前值保持最后状态。启用输入端再次为“接通”时，当前值（TVx）从上次的保持值继续计数，当累计当前值（TVx）大于或等于预设时间（SV）时，定时器状态位为“接通”，当前值连续计数到 32 767。Txxx = T0 至 T255，TVxxx = TV0 至 TV255。

TON、TONR 和 TOF 定时器有三种分辨率。分辨率由下图所示的定时器编号决定。每一个当前值都是时间基准的倍数。例如，10 毫秒定时器中的计数 50 表示 500 毫秒。

定时器类型	分辨率	最大值	定时器编号
TON、TONR、TOF	100ms	3276.7 秒	T0 – T199
	10ms	327.67 秒	T200 – T249
	1ms	32.767 秒	T250 – T255

注意：

（1）TOF，TONR 及 TON 不能共享相同的定时器号码。例如，不能有 TONR TV32 D0 和 TON TV32 D0。
（2）您可以将 TONR 用于累积多个计时间隔。
（3）执行复位操作可以复位定时器，即定时器位置 0，定时器当前值置 0。
（4）只能用复原操作复位 TONR 定时器。

2. 指令的有效操作数

输入输出	类型	操作数	描述
EN	BOOL		**启用输入端接通时，指令执行；否则，不执行**
T	双整数	**TV**	**选定的定时寄存器**
SV	双整数	**K/H/D**	**预设定时时间**

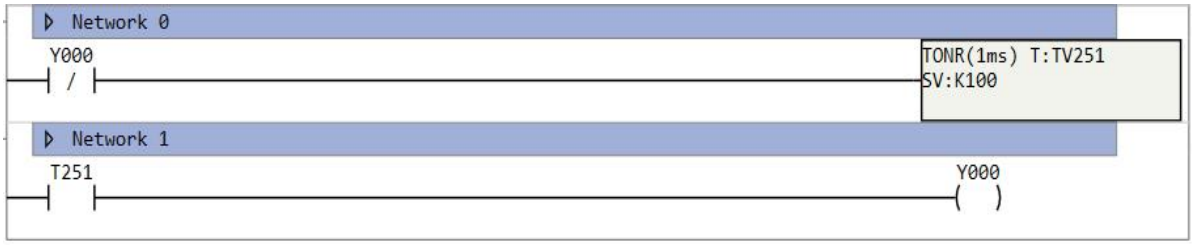
3. 实例

指令表：

```
 Network 000
     LD        X000
     TONR      TV251     K100   // 定时器定时到 K100 时，T251 = 1 （如 K100 表示定时到达 100*1ms,也可以用变量代替）

 Network 001
     LD        T251
     OUT       Y000
```

梯形图：



4.8.3 TOF

1. 指令说明

启用输入端为“接通”时，定时器位（Tx）为“接通”，当前值（TVx）被设为 0。启用输入端从“接通”至“断开”跳变时，当前值（TVx）从 0 开始计时。当前值（TVx）大于或等于预设时间（SV）时，定时器位（Tx）

为“断开”，且定时器停止计时，当前值（TVx）保持。如果启用输入端“断开”的时间短于预设数值，则定时器位仍保持在“接通”状态。如果启用输入端再“接通”时，定时器当前值仍设为 0。Tx = T0 至 T255，TVx = TV0 至 TV255。

TON、TONR 和 TOF 定时器有三种分辨率。分辨率由下图所示的定时器编号决定。每一个当前值都是时间基准的倍数。例如，10 毫秒定时器中的计数 50 表示 500 毫秒。

定时器类型	分辨率	最大值	定时器编号
TON、TONR、TOF	100ms	3276.7 秒	T0 – T199
	10ms	327.67 秒	T200 – T249
	1ms	32.767 秒	T250 – T255

注意：

（1）TOF，TONR 及 TON 不能共享相同的定时器号码。例如，不能有 TONR TV32 D0 和 TOF TV32 D0。

（2）您可以将 TOF 用于延长时间以超过关闭（或假）条件，例如在电机关闭后使电机冷却。

（3）执行复位操作可以复位定时器，即定时器位置 0，定时器当前值置 0。

（4）复位后，TOF 定时器要求启用输入端从“接通”转换为“断开”，以便重新启动。

2. 指令的有效操作数

输入输出	类型	操作数	描述
EN	BOOL		**启用输入端接通时，指令执行；否则，不执行**
T	双整数	**TV**	**选定的定时寄存器**
SV	双整数	**K/H/D**	**预设定时时间**

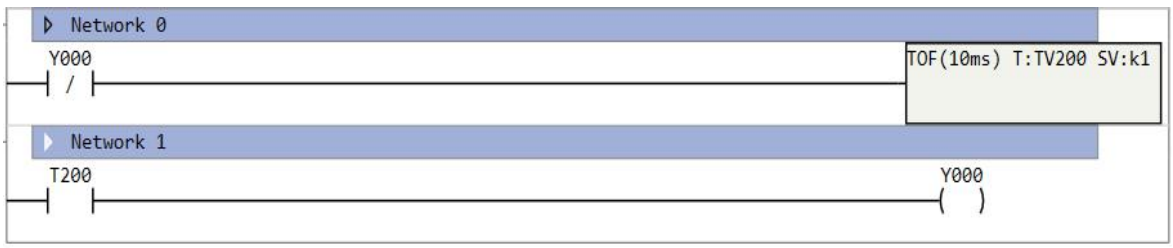
3. 实例

指令表：

```
Network 000
    LD      X000
    TOF     TV200   K1   // 定时器定时到 K1 时，T200 = 1（如 K1 表示定时到达 1*10ms，K2 表示 2*10ms 等,也可以用变量代替）

Network 001
    LD      T200
    OUT     Y000
```

梯形图：



4.9 计数器

计数器指令

内容如下：

1、 向上计数器 CTU

2、 向上向下计数器 CTUD

3、 向下计数器 CTD

4.9.1 CTU

1. 指令说明

每次向上计数输入能流从关闭向打开转换时，向上计数（CTU）指令从当前值向上计数。当前值（CVxxx）大于或等于预设值（SV）时，计数器位（Cxxx）打开。达到最大值（32,767 或 2147483647）时，计数器停止计数。执行复位操作可以复位计数器，即计数器位（Cxxx）置 0，计数器当前值（CVxxx）置 0。

计数器范围： CVxxx=CV0 至 CV255； Cxxx= C0 至 C255。

计数器类型	计数器位数	最大值	计数器编号
CTU	16 位	32767	C0 – C199
	32 位	2147483647	C200 – C234
	32 位（高速）	2147483647	C235 – C255

注意：因为每个计数器有一个当前值，请勿将相同的计数器号码设置给一个以上计数器（号码相同的向上计数器和向下计数器存取相同的当前值）。

2. 指令的有效操作数

输入输出	类型	操作数	描述
EN	BOOL		**启用输入端接通时，指令执行；否则，不执行**
C	整数/双整数	**CV**	**选定的计数寄存器**
SV	整数/双整数	**K/H/D**	**预设计数值**

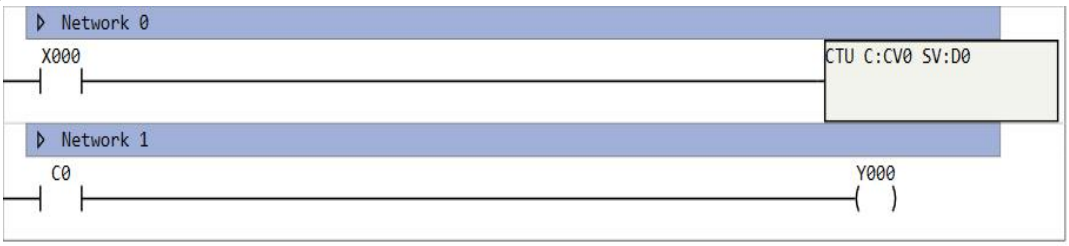
3. 实例

指令表：

```
Network 000
    LD      X000
    CTU     CV0     D0      // 计数器从当前值 CV0 开始向上计数，设定值（SV）为 D0

Network 001
    LD      C0
    OUT     Y000
```

梯形图：



4.9.2 CTD

1. 指令说明

每次向下计数输入能流从关闭向打开转换时，向下计数（CTD）指令从当前值向下计数。当前值 CVxxx 等于 0 时，计数器位（Cxxx）打开。向下计数器继续向下计数到最小值（16 位-32768，,32 位-2147483648），停止计数。执行复位操作可以复位计数器，即计数器位（Cxxx）置 0，计数器当前值（CVxxx）复位为预设值（SV）。

计数器范围： CVxxx=CV0 至 CV255；Cxxx= C0 至 C255。

计数器类型	计数器位数	最大值	计数器编号
CTD	16 位	32767	C0 – C199
	32 位	2147483647	C200 – C234
	32 位（高速）	2147483647	C235 – C255

注意：因为每个计数器有一个当前值，请勿将相同的计数器号码设置给一个以上计数器（号码相同的向上计数器和向下计数器存取相同的当前值）。

2. 指令的有效操作数

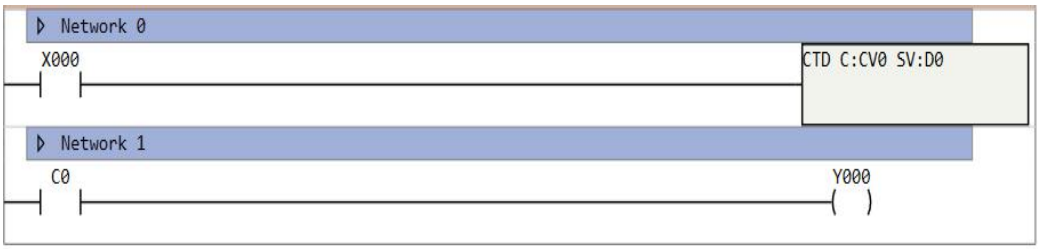
输入输出	类型	操作数	描述
EN	BOOL		**启用输入端接通时，指令执行；否则，不执行**
C	整数/双整数	**CV**	**选定的计数寄存器**
SV	整数/双整数	**K/H/D**	**预设计数值**

3. 实例

指令表：

```
Network 000
    LD      X000
    CTD     CV0     D0      // 计数器从当前值 CV0 开始向下计数，设定值（SV）为 D0
Network 001
    LD      C0
    OUT     Y000
```

梯形图：



4.9.3 CTUD

1. 指令说明

每次计数输入能流从关闭向打开转换时，计数器（CTUD）指令判断当前值与设定值的大小。当前值（CVxxx）大于（小于）预设值（SV）时，计数器做向下（向上）计数，当前值（CVxxx）等于预设值（SV）时，计数器（Cxxx）置为 1，当前值（CVxxx）不动作。执行复位操作可以复位计数器，即计数器位（Cxxx）置 0，预设值（SV）复位。

计数器范围：　　CVxxx=CV0 至 CV255；Cxxx= C0 至 C255。

计数器类型	计数器位数	最大值	计数器编号
CTUD	16 位	32767	C0 – C199
	32 位	2147483647	C200 – C234
	32 位（高速）	2147483647	C235 – C255

注意：因为每个计数器有一个当前值，请勿将相同的计数器号码设置给一个以上计数器（编号相同的计数器存取相同的当前值）。

2. 指令的有效操作数

输入输出	类型	操作数	描述
EN	BOOL		**启用输入端接通时，指令执行；否则，不执行**
C	整数/双整数	**CV**	**选定的计数寄存器**
SV	整数/双整数	**K/H/D**	**预设计数值**

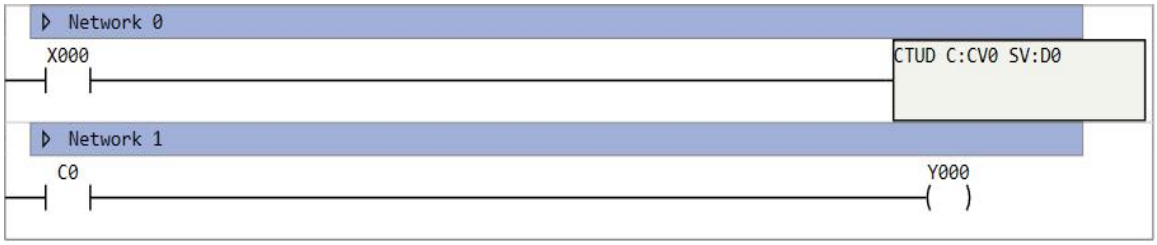
3. 实例

指令表：

```
Network 000
    LD      X000
    CTUD    CV0     D0      // 计数器判断 CV0 与 D0 的大小关系，并从当前值 CV0 开始向上向下计数，设定值（SV）为 D0

Network 001
    LD      C0
    OUT     Y000
```

梯形图：



4.10 程序控制

程序控制指令

内容如下：

1、 FOR--NEXT 循环指令

2、 程序内部跳转 JMP/LBL

3、 调用子程序 CALL

4、 顺控指令 STL\STLE\ST\SET\RST

5、 C 函数功能块调用 CALLM

4.10.1 FOR/NEXT

对应命令	注释/说明	操作数（寄存器）
FOR-NEXT	For 循环	D/K/H

1. 指令说明

启用输入端为“接通”时，执行此指令。每条 FOR 指令必须和 NEXT 指令配对使用，FOR 和 NEXT 指令中间的所有指令程序段将被重复执行（CNT）次，指令必须指定循环次数 CNT 值。NEXT 指令标志着 FOR 循环

的结束。

FOR/NEXT 循环执行的过程中可以修改 CNT 值。当循环再次允许时，循环次数按照新的 CNT 值循环执行。当下一次允许时，FOR/NEXT 指令复位它自己。

启用输入端为“断开”时，则程序将跳过 FOR—NEXT 中间的所有指令程序段，直接执行 NEXT 指令后面的指令。

注意：FOR--NEXT 循环嵌套层数最大为 8 层。

2. 指令的有效操作数

输入输出	类型	操作数	描述
EN	BOOL		**启用输入端接通时，指令执行；否则，不执行**
CNT	整数/双整数	**K/H/D**	**预设循环次数**

3. 实例

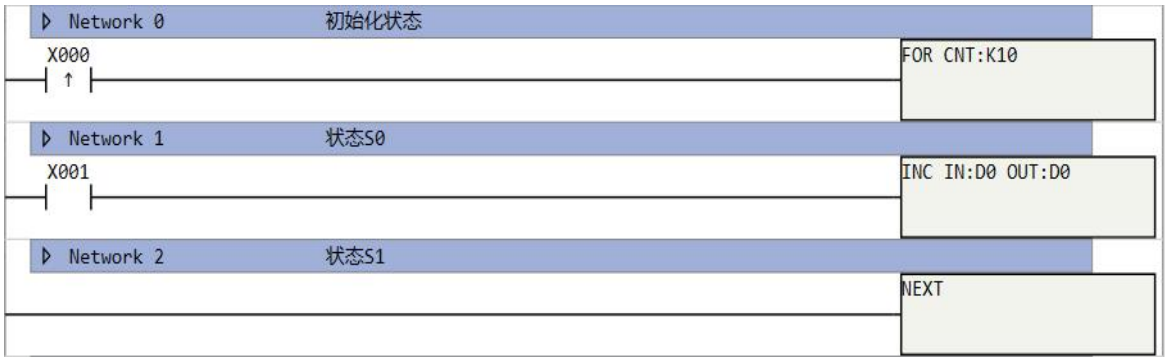
指令表：

```
Network 000
    LDP     X000
    FOR     K10   // 循环 10 次

Network 001
    LD      X001
    INC     D0      D0   // 每循环一次 D0 累加 1

Network 002
    NEXT
```

梯形图：



4.10.2 JMP/LBL

程序内部跳转 CJ/LBL

对应命令	注释/说明	操作数（寄存器）
CJ	程序内部跳转	K/H

1. 指令说明

跳转（CJ）指令指程序跳转到标签（LBL）指定的程序分支的操作。标签 LBL 指令与 CJ 指令配合使用，标识跳转的位置。

可以在主程序、子程序或者中断程序中，使用跳转指令。跳转和与之相应的标号指令必须位于同一段程序代码(无论是主程序、子程序还是中断程序)。不能从主程序跳到子程序或中断程序， 同样不能从子程序或中断程序跳出。

跳接接受时，堆栈顶值始终为逻辑 1。

2. 指令的有效操作数

输入输出	类型	操作数	描述
EN	BOOL		**启用输入端接通时，指令执行；否则，不执行**
LBL	整数	**K/H**	**JMP 指令指定的程序跳转目的地位置标号**
L	整数	**K/H**	**LBL 指令对应的 JMP 指令程序跳转目的地的位置标号**

3. 实例

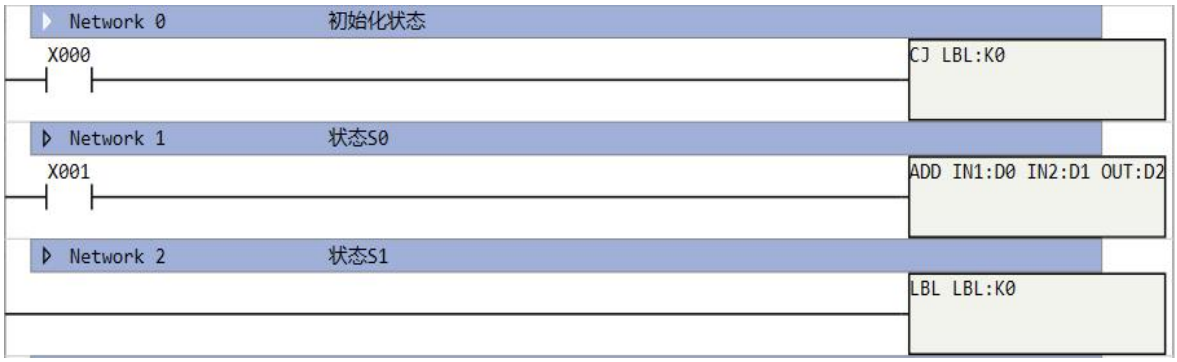
指令表：

```
Network 000
    LD      X000
    CJ      K0   // 假如栈顶值为 1，跳转到 K0 标签处

Network 001
    LD      X001
    ADD     D0      D1      D2   // 如果上面没有跳转，则栈顶为 1 时会执行 D2=D0+D1

Network 002
    LBL     K0   // K0 标签
```

梯形图：



4.10.3 CALL

1. 指令说明

当 CALL 指令有效时，CALL 指令将程序控制权交给子程序。子程序执行完成后，控制权返回到调用子程序的指令的下一条指令。

当 CALL 指令无效时，程序会自动跳过此条 CALL 指令，执行下一条指令。

要添加一个子程序可以在左侧工程管理器通过路径“工程”-->“程序”-->“子程序”：右键点击子程序文件夹，点击选择“添加子程序”。

要调用一个子程序除了输入 CALL 指令以外，还可以通过从工程管理器中将子程序项拖动到梯形图中来插入 CALL 指令。

注意：CALL 子程序调用最大嵌套 8 层。

2. 指令的有效操作数

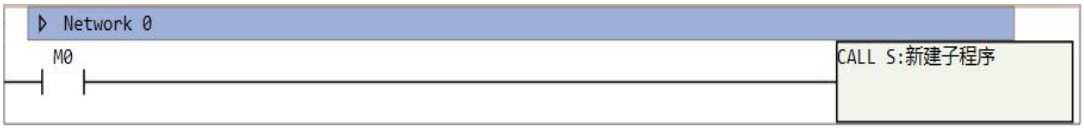
输入输出	类型	操作数	描述
EN	BOOL		**启用输入端接通时，指令执行；否则，不执行**
			子程序名称

3. 实例

指令表：

```
LD      X000
CALL    新建子程序   // 调用子程序“新建子程序”
```

梯形图：



4.10.4 CALLM

1. 指令说明

当 CALLM 指令有效时，CALLM 指令将调用函数功能块，启动一个函数功能块子程序。

当 CALLM 指令无效时，程序会自动跳过此条 CALLM 指令，执行下一条指令。

注意：（1）.采用 CALLM 指令调用函数功能块时，要妥善处理好调用的入口和出口判断，防止造成时序或者逻辑的错误，导致程序执行出现错误。

（2）.函数功能块的设计详见“函数功能块的设计”部分。

2. 实例(排序)

函数块：

```
// 数据初始化
[Array(data, size)]
void Data_Init(WORD data, WORD size)
{
    short i = 0 ;
    for ( i = 0 ; i < *size ; i++)
        data[i] = (i ^ 0x4321);          // 每个数都异或 0x4321
}

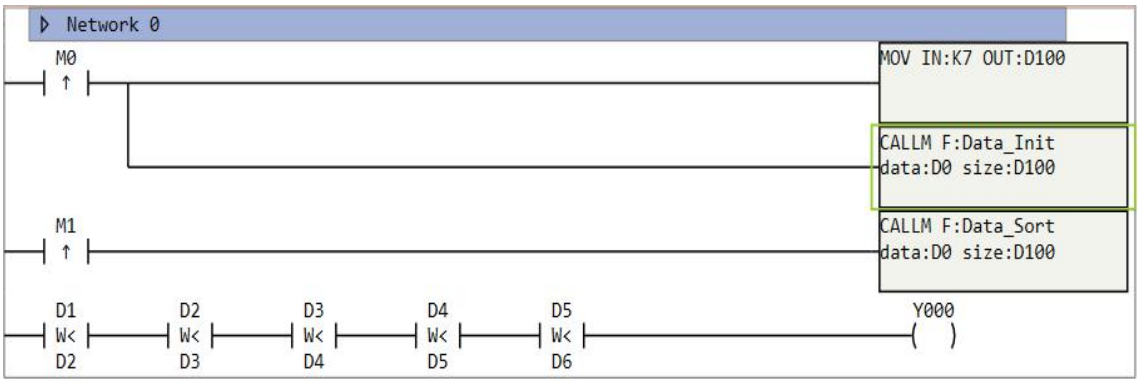
// 数据从小到大排序
[Array(data, size)]
void Data_Sort(WORD data, WORD size)
{
    short i = 0;
    short j = 0;
    short temp = 0;
    for ( i = 0 ; i < *size ; i++)
    {
        temp = data[i];
        for (j = i ; j > 0 && data[j-1] > temp ; j--)
            data[j] = data[j-1];      // 不断向前冒泡
        data[j] = temp;
    }
}
```

指令表：

```
LDP     M0    // M0 上升沿来临时，执行函数功能块 Data_Init，使 D[0]=0x4321，D[1]=0x4320，D[2]=0x4323，D[3]=0x4322，D[4]=0x4325，D[5]=0x4324，D[6]=0x4327
MOV     K7      D100   // 将长度设置到 D100
CALLM   Data_Init   D0      D100   // 调用 Data_Init，参数为 data=D0, size=D100
POP   // 弹出上一个栈值
LDP     M1    // M1 上升沿来临时，执行函数功能块 Data_Sort，对 D[0]到 D[6]进行从小到大的排序，结果为 D[0]=0x4320，D[1]=0x4321，D[2]=0x4322，D[3]=0x4323，D[4]=0x4324，D[5]=0x4325，D[6]=0x4327
CALLM   Data_Sort   D0      D100   // 调用 Data_Sort，参数为 data=D0, size=D100
POP   // 弹出上一个栈值
LDW<    D1      D2
AW<     D2      D3
AW<     D3      D4
AW<     D4      D5
```

```
AW<     D5      D6
OUT     Y000    // 如果 D1<D2<D3<D4<D5<D6，Y000 置 1
POP     // 弹出上一个栈值
```

梯形图：



3. 实例（求一组浮点的标准差）

函数块：

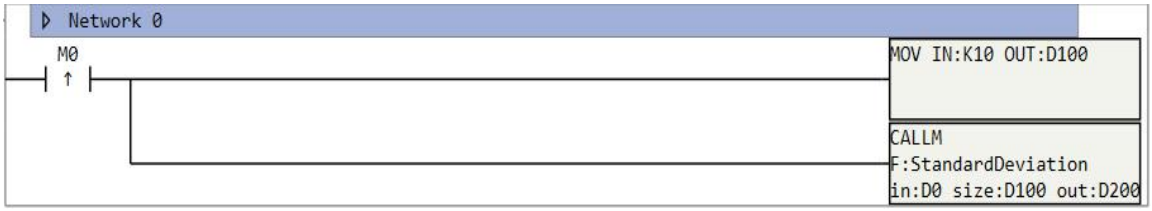
```
// 求平均值
[Array(in, size)]
void Averange(FLOAT in, WORD size, FLOAT out)
{
    *out = 0.0;
    short i = 0;
    for ( ; i < *size ; i++)
        *out += in[i];
    *out /= *size;
}

// 求标准差
[Array(in, size)]
void StandardDeviation(FLOAT in, WORD size, FLOAT out)
{
    float ave = 0.0;
    Averange(in, size, &ave);
    short i = 0;
    *out = 0.0;
    for ( ; i < *size ; i++)
        *out += (in[i] - ave) * (in[i] - ave);
    *out /= *size;
    *out = sqrt(*out);
}
```

指令表：

```
    LDP     M0
    MOV     K10     D100    // 将长度设置到 D100
    CALLM   StandardDeviation   D0      D100    D200    // 调用 StandardDeviation，参数为 in=D0, size=D100,out=D200
```

梯形图：



4.10.5 STL

1. 指令说明

1) STL 与 STLE 必须一一配对使用，STL 表示一个状态的开始，STLE 表示一个状态的结束；
2) 各个状态之间都是独立的，不能互相嵌套使用，且在一个扫描周期的执行过程中，不要重复使用相同编号的 STL 段；

3) 初始状态可以按需求任意指定，不一定非要按照 S0、S1、S2 等顺序执行；
4) 执行 SET Sx K1 指令后，该 Sx 状态被打开，相应 STL Sx 梯形图段将被执行；
5) 执行 RST Sx K1 指令后，该 Sx 状态被关闭，相应 STL Sx 梯形图段将被复位；
6) 状态从 ON 变为 OFF 时，会将状态内的 OUT、定时器、脉冲等输出类指令复位，对于 SET、接通延时记忆定时器等指令保持原有状态；
7) SET 与 ST 均为状态转移指令，但需注意 SET 与 ST 两个状态转移指令的区别：

① 当使用 SET 指令当状态转移条件成立时，下一个待转移状态被置为 ON，在状态转移的过程中，仅在 1 个扫描周期内该两个状态会同时接通，当前 STL 段中的状态在转移后的下一个扫描周期被复位 OFF；

② 当使用 ST 指令当状态转移条件成立时，下一个待转移状态被置为 ON，但当前 STL 段中的状态不会被复位 OFF， ST 指令一般在程序需要同时运行多个状态程序时使用；

8) 由于不同型号的 PLC 内部储存区大小不同，因此该状态寄存器 S 范围有不同的使用限制：

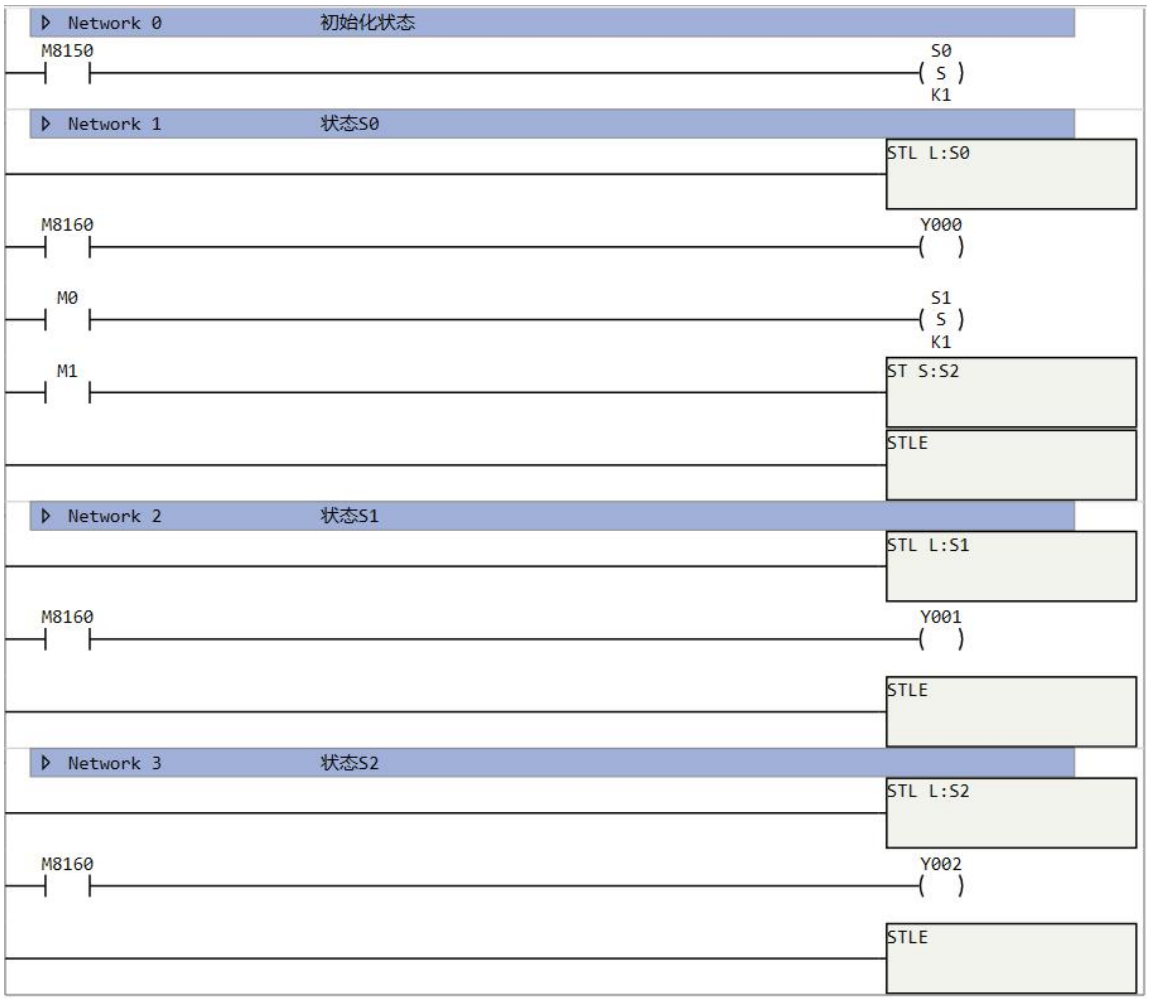
① 对于 GC043 系列一体机，该状态 S 范围为 0~63；
② 对于 GC070 系列一体机、FGs_16M 与 FGs_32M 系列 PLC，该状态 S 范围为 0~999；
③ 对于 FGs_64M 与 FGm_64M 系列 PLC，该状态 S 范围为 0~999；

2. 实例 1：SET 与 ST 的区别

指令表：

```
Network 0
0    LD        M8150
1    SET       S0        K1   // 初始化状态S0为ON
2    POP
Network 1
0    STL       S0
1    LD        M8160
2    OUT       Y000
3    POP
4    LD        M0
5    SET       S1        K1   // 当条件M0导通时，状态S1被置为ON，同时下个扫描周期该S0状态被复位
6    POP
7    LD        M1
8    ST        S2   // 当条件M1导通时，状态S2被置为ON，但不会复位当前S0状态，即接下来程序会同时执行S0和
9    POP             S2两个状态程序
10   STLE
Network 2
0    STL       S1
1    LD        M8160
2    OUT       Y001
3    POP
4    STLE
Network 3
0    STL       S2
1    LD        M8160
2    OUT       Y002
3    POP
4    STLE
```

梯形图：



3. 可选择的分支和连接

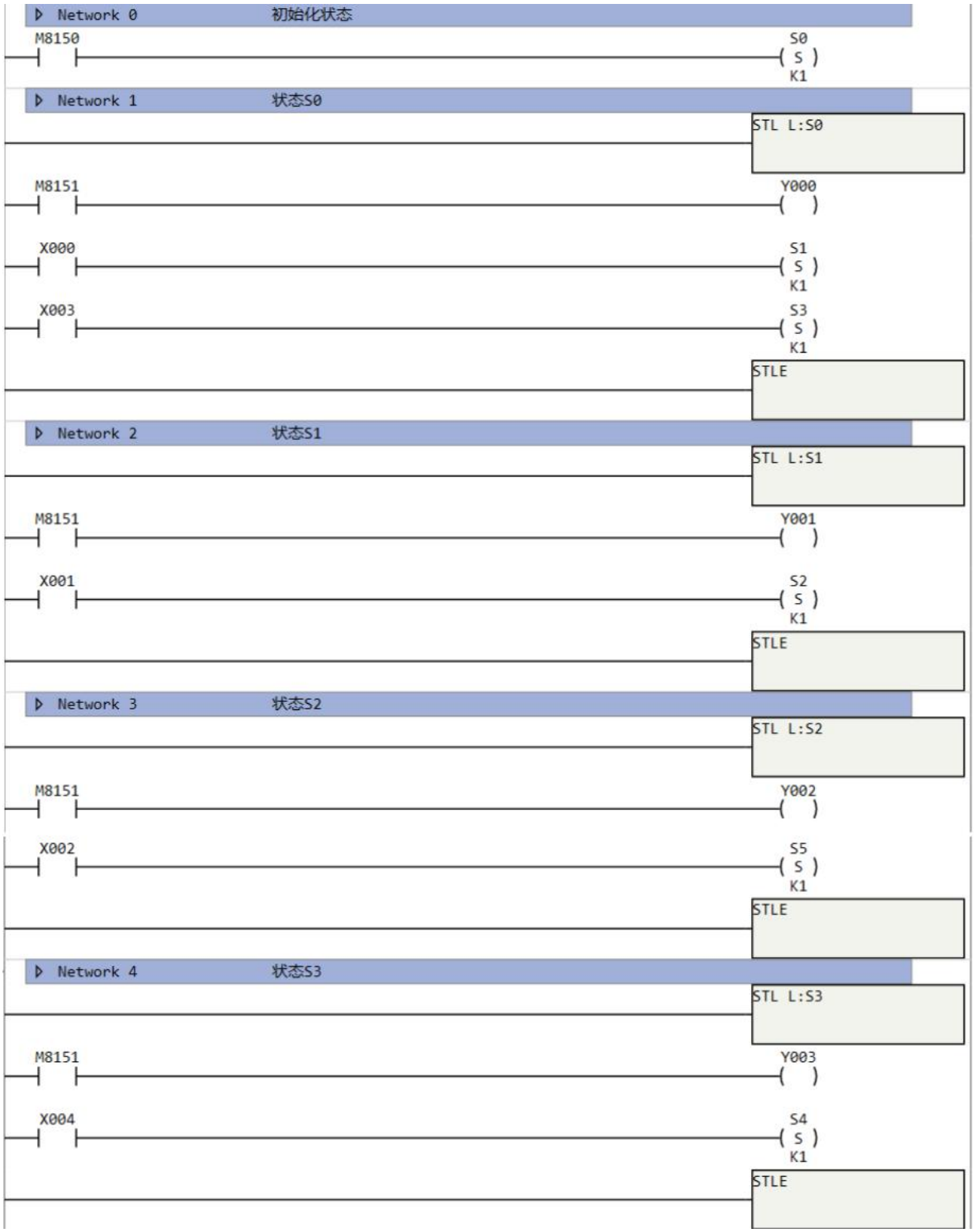
在生产实际中，对具有多流程的工作要进行流程选择或者分支选择，即一个控制流可能转入多个可能的控制流中的某一个，但不允许多路分支同时执行；到底进入哪一个分支，取决于控制流前面的转移条件哪一个为真。

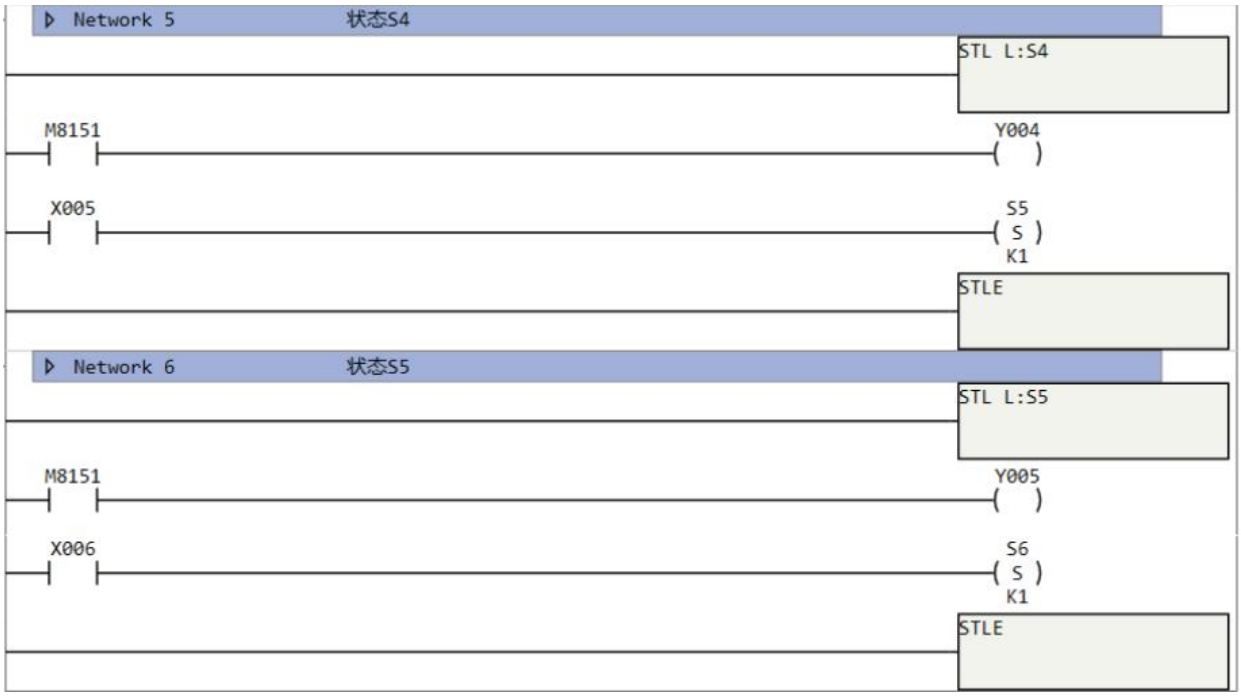
指令表：

```
Network 0
0    LD      M8150
1    SET     S0      K1
2    POP
Network 1
0    STL     S0
1    LD      M8151
2    OUT     Y000
3    POP
4    LD      X000
```

```
5    SET     S1      K1
6    POP
7    LD      X003
8    SET     S3      K1
9    POP
10   STLE
Network 2
0    STL     S1
1    LD      M8151
2    OUT     Y001
3    POP
4    LD      X001
5    SET     S2      K1
6    POP
7    STLE
Network 3
0    STL     S2
1    LD      M8151
2    OUT     Y002
3    POP
4    LD      X002
5    SET     S5      K1
6    POP
7    STLE
Network 4
0    STL     S3
1    LD      M8151
2    OUT     Y003
3    POP
4    LD      X004
5    SET     S4      K1
6    POP
7    STLE
Network 5
0    STL     S4
1    LD      M8151
2    OUT     Y004
3    POP
4    LD      X005
5    SET     S5      K1
6    POP
7    STLE
Network 6
0    STL     S5
1    LD      M8151
2    OUT     Y005
3    POP
4    LD      X006
5    SET     S6      K1
6    POP
7    STLE
```

梯形图：


Network 0 初始化状态
M8150
S0
S
K1
Network 1 状态S0
STL L:S0
M8151
Y000
X000
S1
S
K1
X003
S3
S
K1
STLE
Network 2 状态S1
STL L:S1
M8151
Y001
X001
S2
S
K1
STLE
Network 3 状态S2
STL L:S2
M8151
Y002
X002
S5
S
K1
STLE
Network 4 状态S3
STL L:S3
M8151
Y003
X004
S4
S
K1
STLE



4. 并行分支和连接

在许多实例中，一个顺序控制状态流必须分成两个或多个不同分支控制状态流，这就是并行分支或并发分支。当一个控制状态流分成多个分支时，所有的分支控制状态流必须同时激活。当多个控制流产生的结果相同时，可以把这些控制流合并成一个控制流，即并行分支的连接。在合并控制流时，所有的分支控制流必须都是完成了的。这样，在转移条件满足时才能转移到下一个状态。

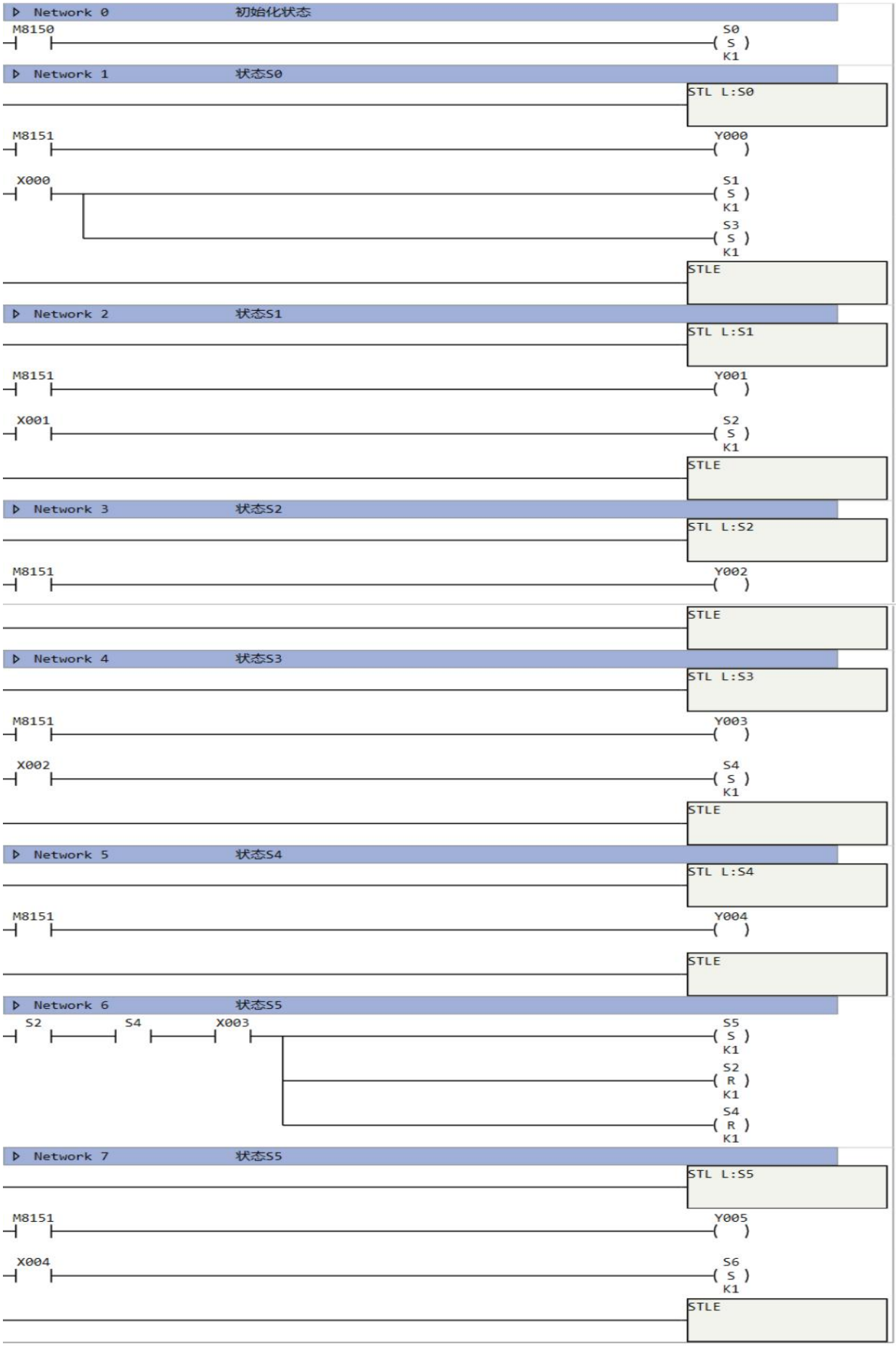
以下这个示例需特别说明的是：当并行分支连接时，要同时使所有含状态转移到新的状态，完成新状态的启动。另外在状态 S2 和 S4 的 STL 程序段中，由于没有使用 SET 指令，所以 S2 和 S4 的复位不能自动进行，最后要用复位指令对其进行复位。这种处理方法在并行分支的连接合并时会经常用到，而且在并行分支连接合并前的最后一个状态往往是“等待”过度状态，它们要等待所有并行分支都为“活动状态”后一起转移到新的状态。这些“等待”状态不能自动复位，它们的复位就要使用复位指令来完成。

指令表：

```
Network 0
0    LD      M8150
1    SET     S0      K1
2    POP
Network 1
0    STL     S0
1    LD      M8151
2    OUT     Y000
3    POP
4    LD      X000
5    SET     S1      K1
6    SET     S3      K1
7    POP
8    STLE
```

```
Network 2
0    STL     S1
1    LD      M8151
2    OUT     Y001
3    POP
4    LD      X001
5    SET     S2      K1
6    POP
7    STLE
Network 3
0    STL     S2
1    LD      M8151
2    OUT     Y002
3    POP
4    STLE
Network 4
0    STL     S3
1    LD      M8151
2    OUT     Y003
3    POP
4    LD      X002
5    SET     S4      K1
6    POP
7    STLE
Network 5
0    STL     S4
1    LD      M8151
2    OUT     Y004
3    POP
4    STLE
Network 6
0    LD      S2
1    AND     S4
2    AND     X003
3    SET     S5      K1
4    RST     S2      K1
5    RST     S4      K1
6    POP
Network 7
0    STL     S5
1    LD      M8151
2    OUT     Y005
3    POP
4    LD      X004
5    SET     S6      K1
6    POP
7    STLE
```

梯形图：


Network 0 初始化状态
M8150
S0
S
K1
Network 1 状态S0
STL L:S0
M8151
Y000
X000
S1
S
K1
S3
S
K1
STLE
Network 2 状态S1
STL L:S1
M8151
Y001
X001
S2
S
K1
STLE
Network 3 状态S2
STL L:S2
M8151
Y002
STLE
Network 4 状态S3
STL L:S3
M8151
Y003
X002
S4
S
K1
STLE
Network 5 状态S4
STL L:S4
M8151
Y004
STLE
Network 6 状态S5
S2
S4
X003
S5
S
K1
S2
R
K1
S4
R
K1
Network 7 状态S5
STL L:S5
M8151
Y005
X004
S6
S
K1
STLE

4.10.6 RET

1. 指令说明

输入条件导通时，放弃执行当前子程序内这条指令之后的指令，直接跳出到调用子程序的位置，继续执行父程序之后的指令。

主程序 Main 内不允许使用 RET 指令。

2. 指令的有效操作数

输入输出	类型	操作数	描述
EN	BOOL		启用输入端接通时，指令执行；否则，不执行

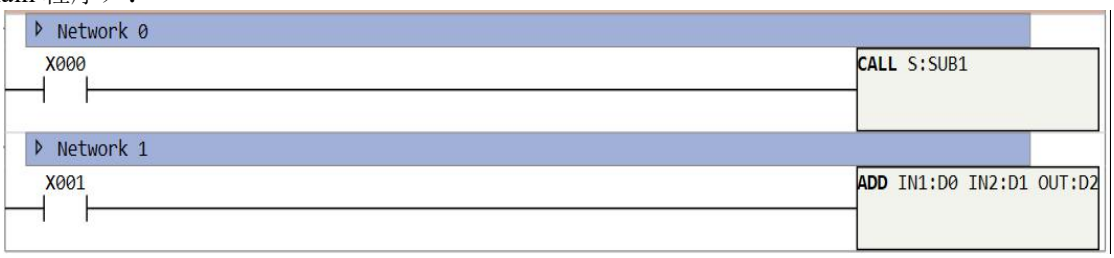
3. 实例

指令表（Main 程序）：

```
Network 000
LD        X000
CALL      SUB1   // 假如栈顶值为 1，调用名为 SUB1 的子程序

Network 001
LD        X001
ADD       D0        D1        D2   // 如果上面没有跳转，则栈顶为 1 时会执行 D2=D0+D1
```

梯形图（Main 程序）：

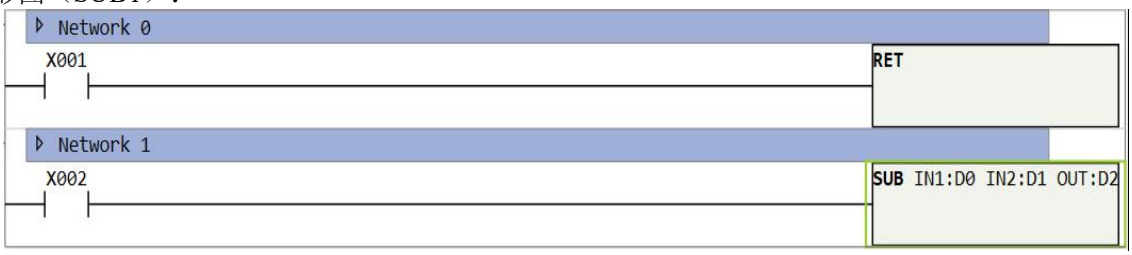


指令表（SUB1）：

```
Network 000
LD        X001
RET    // 假如栈顶值为 1，从这里跳出来，继续执行 Main 程序的网络 1

Network 001
LD        X002
SUB       D0        D1        D2   // 如果上面没有跳转，会继续执行这里的程序
```

梯形图（SUB1）：



4.10.7 BREAK

1. 指令说明

输入条件导通时，如果这个指令在 FOR 循环内部，会跳出 FOR 循环继续执行 NEXT 之后的指令。如果这个指令在 STL-STLE 块的内部，会跳出这个 STL 块继续执行 STLE 之后的指令。

FOR 循环和 STL 块同时存在的场合使用这个指令，会跳出当前的 FOR 循环。

2. 指令的有效操作数

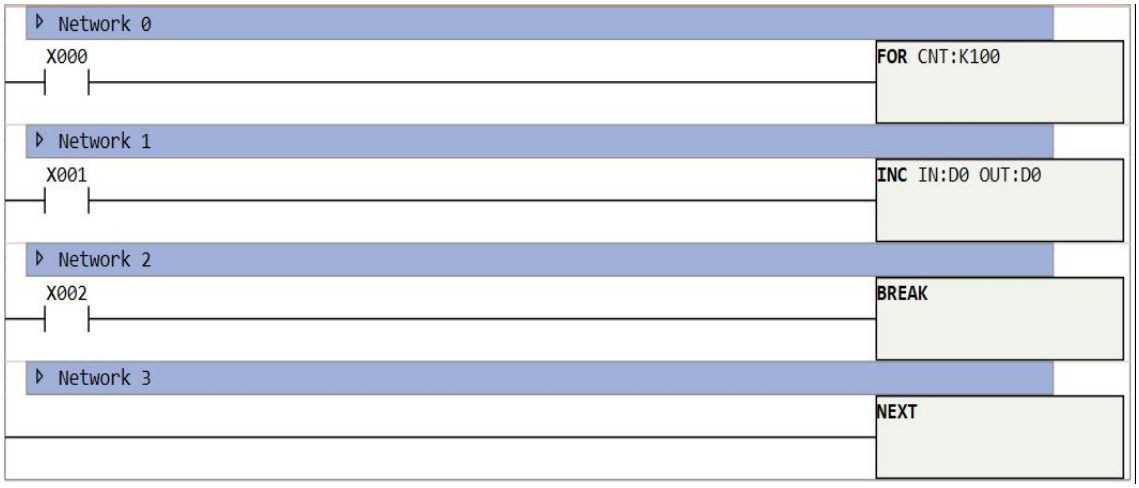
输入输出	类型	操作数	描述
EN	BOOL		启用输入端接通时，指令执行；否则，不执行

3. 实例 1（FOR 循环）

指令表：

```
Network 000
LD          X000
FOR         K100    // FOR 循环的开始，循环 100 次
Network 001
LD          X001
INC         D0          D0      // 每一次循环累加一次 D0
Network 002
LD          X002
BREAK       // X002 为 ON 时，从 FOR 循环中跳出
Network 003
NEXT
```

梯形图：

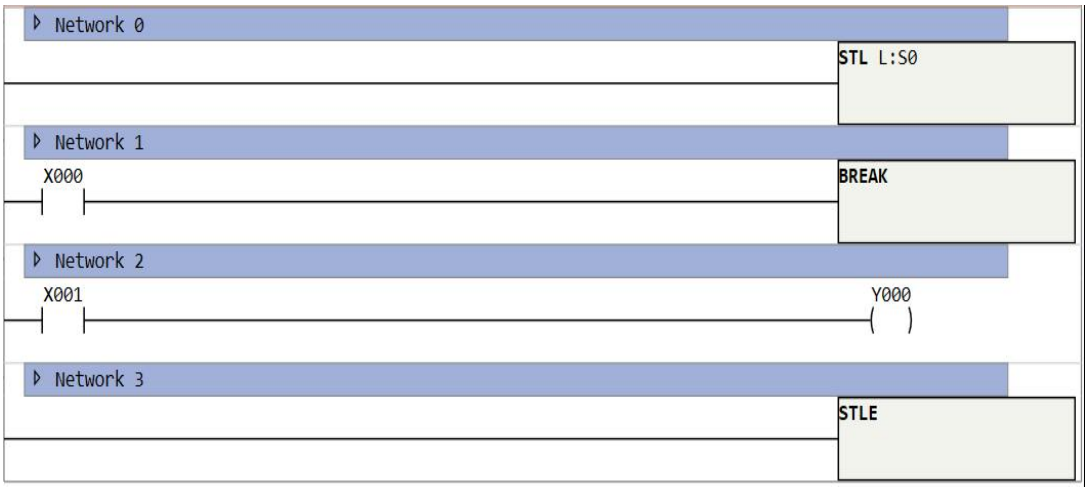


4. 实例 2（STL 块）

指令表：

```
Network 000
STL         S0
Network 001
LD          X000
BREAK   // 如果 X0 为 ON，从 STL 块中跳出
Network 002
LD          X001
OUT         Y0   // 上面的指令不跳出，则继续执行这里的指令
Network 003
STLE
```

梯形图：



4.10.8 IST

1. 指令说明

指令 IST 给步进程序定制了一套多模式自动控制的综合系统，用户输入给定的开关位来进行模式切换和程序控制。

给定一个作为输入开关的起始位（以 M20 为例）,M20~M27 都会被 IST 所使用，这些位寄存器作为开关，来操控 IST 指令来控制步进程序。这些位只有在触发上升沿才有效。2

M20：切换到手动模式。 M21：切换到回原点模式。 M22：切换到单步模式。 M23：切换到单周期模式。M24：切换到自动模式。 M25：原点回归启动。 M26：启动。 M27：停止。

当触发 M20 上升沿时，IST 会立即进入手动模式。同理，在 M21,M22,M23,M24 的上升沿分别进入回原点模式，单步模式，单周期模式和自动模式。

2. 寄存器使用

IST 会涉及到下面的寄存器，请勿将以下的寄存器当作普通的寄存器来编程：

IST相关寄存器		
存器名称	功能	读写权限
数据寄存器D		
D_8048	IST当前模式（0:未启用，1:手动，2:回原点，3:单步，4:单周期，5:自动）	Read
线圈寄存器M		
M_8021	IST开始转移	Read & Write
M_8022	IST启动脉冲	Read
M_8023	IST原点回归完成	Read
M_8024	IST原点条件	Read & Write
M_8025	IST所有输出禁止复位	Read & Write
状态寄存器S		
S_0	手动操作的初始化状态	Read & Write
S_1	原点回归的初始化状态	Read & Write
S_2	自动运行的初始化状态	Read & Write

3. 注意要点

- 请在主程序的开头使用这条指令，输入条件是 M8151。
- IST 指令在程序中只能编写 1 个。

4. IST 指令的等价梯形图

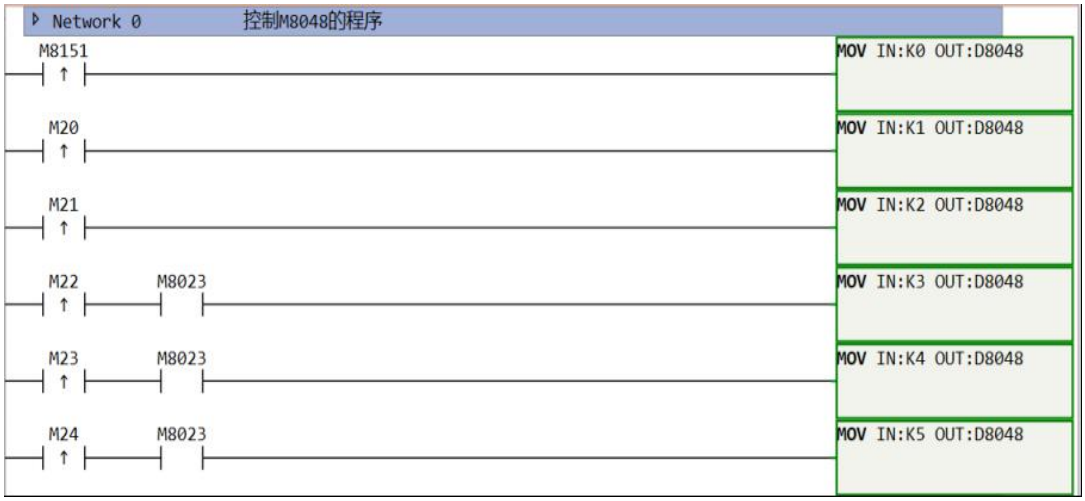
有关使用 IST 指令被自动控制的线圈寄存器 M 和状态寄存器的详细内容，如下面的等价梯形图所示。（作为参考知识，请阅读一遍）

此外，该等价梯形图不能编写为程序。

a) IST 当前模式（D8048）相关

D8048 与 IST 的当前模式（手动，回原点，单步，单周期，自动）相关。初始值为 0，在一定条件下根据 IST 输入信号的上升沿来被控制。

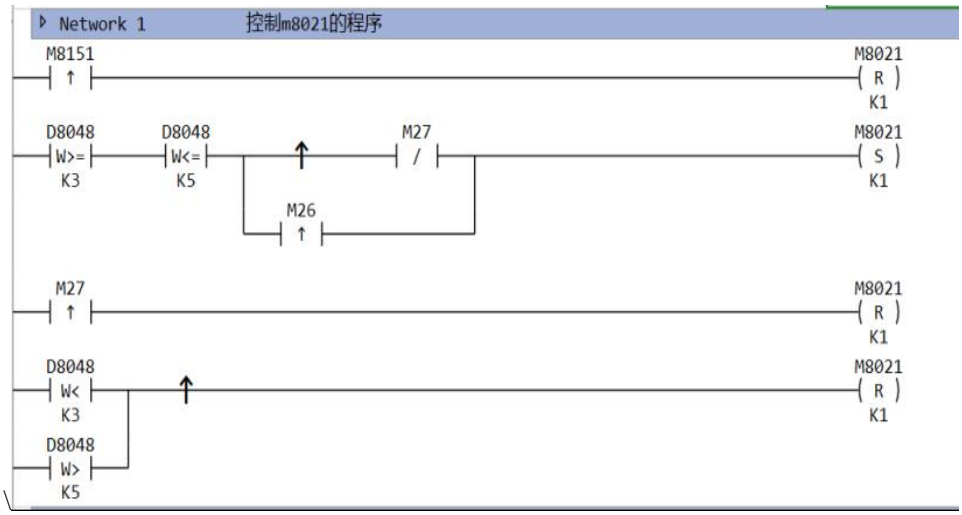
单周期，单步，自动这三种模式使用的是同一个程序，所以被归为一个自动模式类。这类模式只有在原点回归条件下才能被切换。



b) 开始转移（M8021）相关

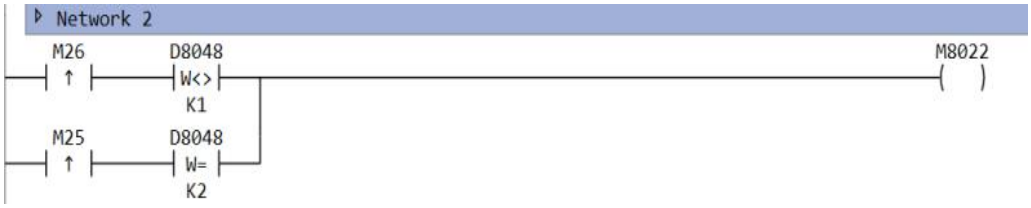
在自动模式类下，按下开始按键(M26)后 M8021 才动作，尤其是在连续模式时，执行自保持动作，按停止按键(M27)后被解除。

进入到手动模式或者回原点模式下，会复位 M8021。



c) 起动脉冲（M8022）相关

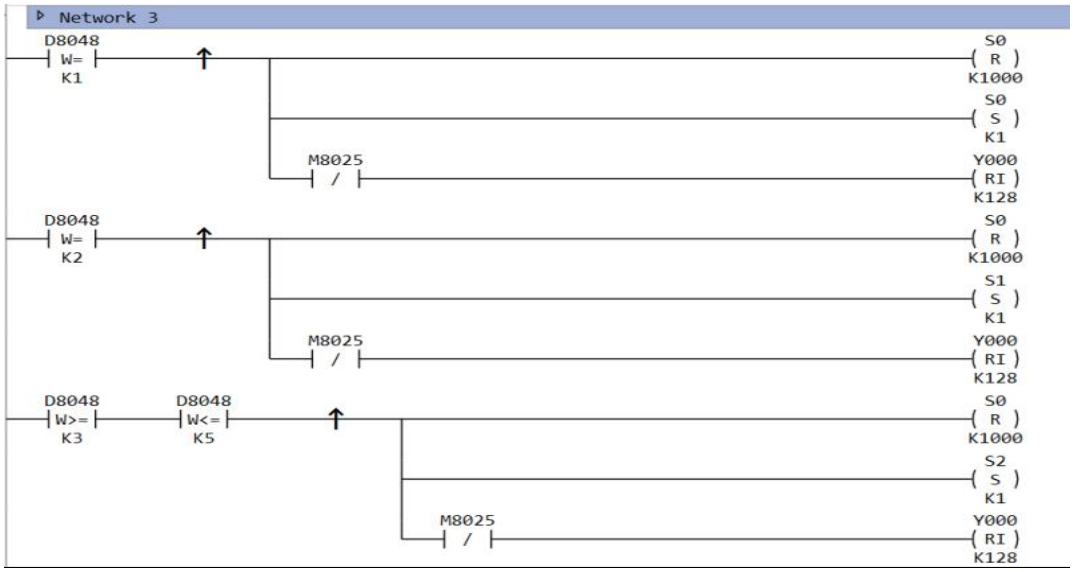
起动脉冲在单个周期的上升沿，用来标识自动模式或者原点回归模式的起动。所以启动信号(M65)和原点回归启动信号(M25)只有在对应的模式下才有效。



d) 顺控状态（S0,S1,S2）相关

S0，S1 和 S2 分别是手动模式，原点回归模式和自动模式的起始状态，每进入其中一个模式，就要复位所有的其他状态，并置位对应的起始状态。

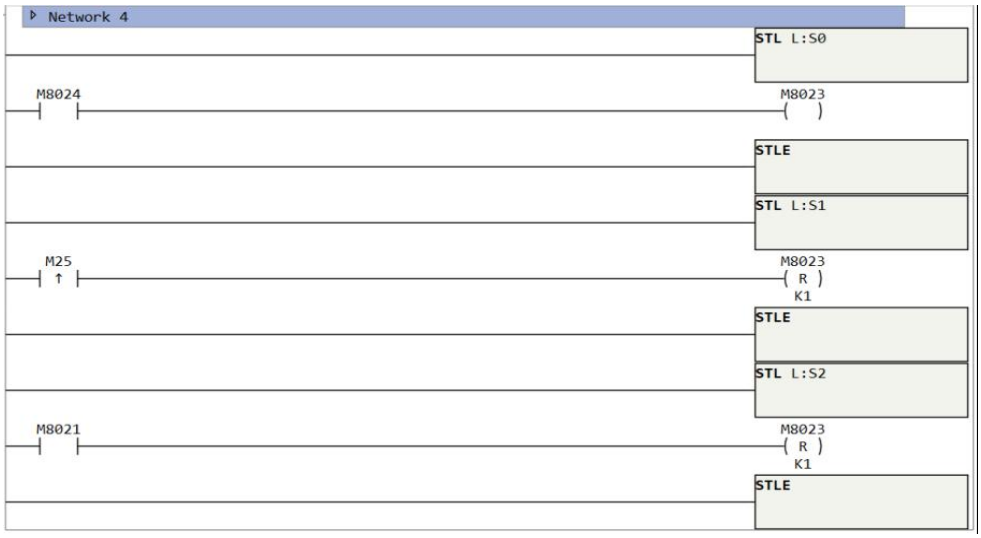
这时，默认会将所有的 Y 输出复位，可以将 M8025 置为 ON 来取消这个操作。



e) 原点回归完成标志(M8023)相关

根据各个模式的输入，执行初始化状态的切换的同时，还可以执行对 M8023 的控制。

但是，M8043 和 M8044（原点回归条件）需要用户程序等控制。



5. 使用实例（机械手臂控制）

1. 动作要求：分开大小两个皮球，并搬到不同的箱子里存放。配置触摸屏或者控制台以供控制。
2. 机械手臂操作：下降，夹取，上升，右移，下降，释放，上升，左移，依序完成皮球的搬运。

3. IO 设置: PLC 输出 Y0 来让手臂上升，输出 Y1 下降，输出 Y2 右移，输出 Y3 左移，输出 Y4 来抓紧把手。X0 来感应球的重量（过重置为 ON，否则 OFF）,手臂到达夹取球的位置相同垂直线时 X1 置位 ON，到达大球箱子垂直线时 X2 置位 ON，到达小球箱子垂直线时 X3 置位 ON，到达最顶部的水平线时 X4 置位 ON，到达最底部的水平线时 X5 置位 ON

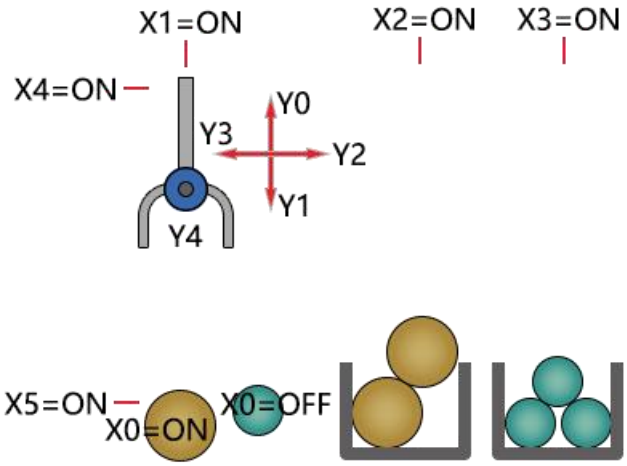


图 1.1 动作流程示意图

运行模式		动作内容
手动	手动操作：	使用各按钮对各负载启停操作的模式。
	原点回归：	按下原点回归用按键，使机械自动返回到原点的模式。
自动	单步操作：	每次按下开始按键就前进一个工序。
	单周期操作：	在原点位置按下开始按键后，执行一个循环的自动运行后，在原点位置停止。如果在中途按下停止按键，就在该工序停止，再次按开始按键后会从刚才的位置开始继续运行，然后再原点处自动停止。
	自动模式：	再原点的位置按下开始按键后，开始连续的重复运行。按下停止按键后，会运行到原点位置后停下。

表 1.1 IST 动作说明

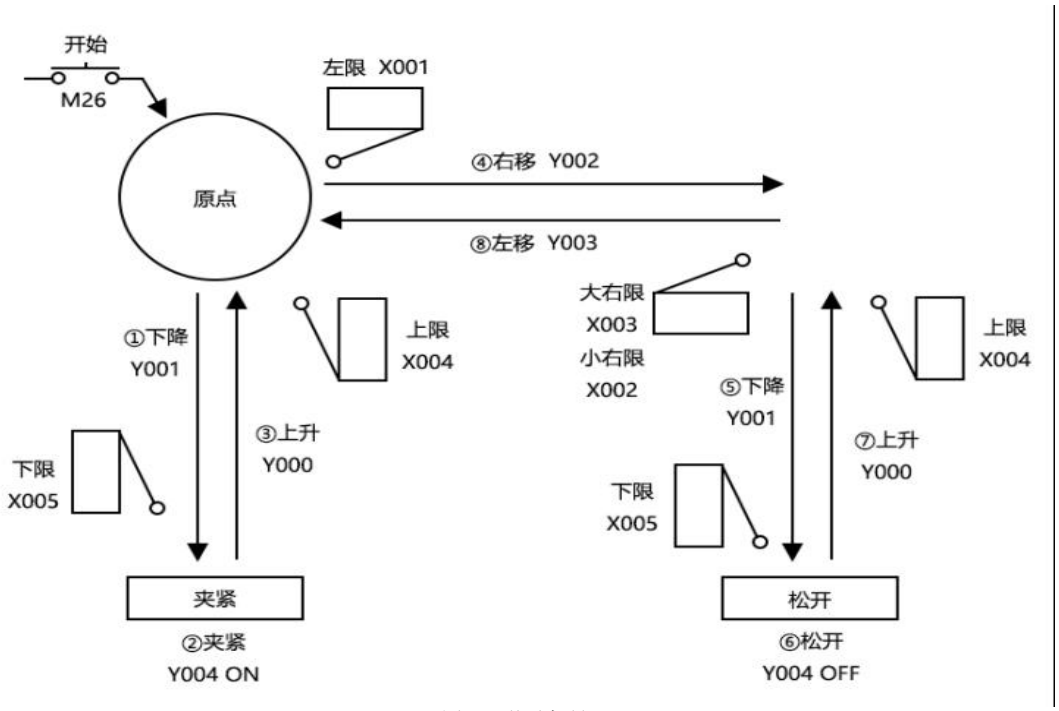


图 1.2 传动机械

以左上为原点，按照下降，夹紧，上升，右移，下降，松开，上升，左移的顺序将工件从左移到右。下降/上升，左移/右移时使用双电磁阀（驱动/非驱动的 2 个输入），夹紧时使用单电磁阀（仅通电时动作）。

6. IST 指令用的特殊辅助继电器(M)

在 IST 指令中使用的辅助继电器(M)，区分为：根据指令本身具体情况自动进行控制，以及根据运行的准备和控制目的，需要用程序进行控制。

a) M8020：禁止转移

一旦这个辅助继电器工作，则所有的状态转移被禁止。

M8020 和 IST 的在单步和单周期的中断无关。实际上，IST 使用的是和 M8020 不同的机制。使用 ISTNEXT 指令可以起到和触发启动信号相同的作用，同时也复位了 M8020。

b) M8021：开始转移

作为从初始化状态 S2 开始转移到下一个状态的转移条件的辅助继电器。

场合	动作
手动模式	不动作。
原点回归模式	
步进模式	仅当按下开始键时动作。
单周期模式	在初始状态下，按下开始键时动作，循环一次后返回初始状态。 在一个流程周期完成之前动作会被保持，按下停止键后会暂停。
自动模式	按下开始键后动作被保持，按下停止键后被解除。

c) M8022：起始脉冲

仅在按下开始键时的一瞬间动作。

d) M8027：STL 监控有效

将 M8027 置为 ON 后，使 STL 监控变为有效，动作中的状态编号(S0~S999)按照从小到大的顺序被保存在特殊寄存器 D8040~D8047 中。

出于性能考虑，IST 不会默认将 M8027 置为 ON，如有需要则在用户程序里手动置位。

因此，最多可以监控 8 个动作状态编号。

此外，如果这些状态中任意一个动作，特殊辅助继电器 M8026 也动作。

e) M8023：原点回归结束

在原点回归模式下，当机械回归到原点时，请使用用户程序使这个特殊复制继电器(M)动作。

f) M8024：原点条件

请在检测出机械的原点条件后，驱动这个特殊辅助继电器，所有模式下都为有效信号。

g) M8025：禁止所有输出复位

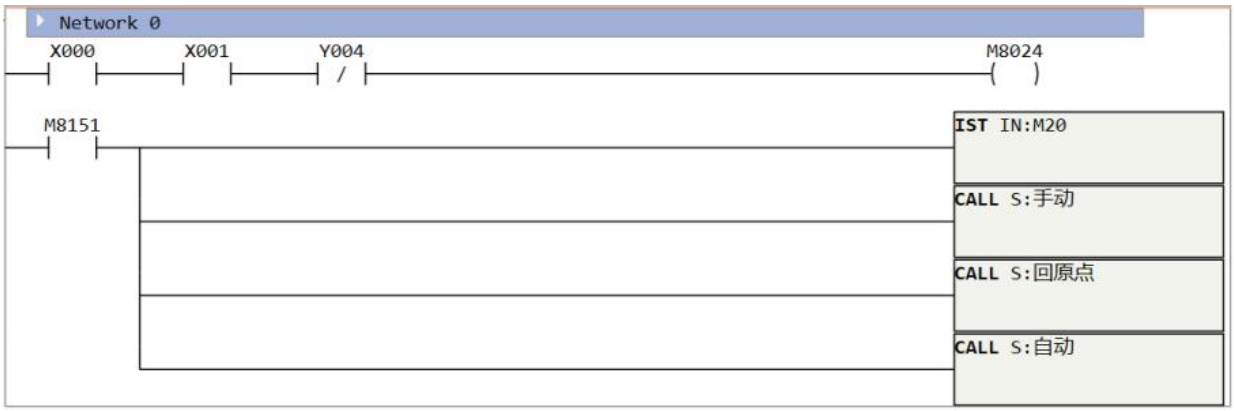
在手动模式，原点回归模式，自动模式之间执行切换后，当机械不在原点位置时，所有输出和动作状态被复位。但是如果先驱动 M8045，则仅动作状态被复位。

7. 程序示例

a) 主程序

主程序需要完成三个步骤：

1. 检测原点条件，控制 M8024 这个寄存器。
2. 调用 IST 指令，这里以 M20~M27 作为 8 个输出。
3. 实现三个模式类对应的三个程序，便于清晰管理，这里可以实现三个子程序来调用。



b) 手动模式

手动模式的状态下，根据不同的输入执行不同的动作。

手动模式的状态位为 S0，进入这个模式后这个状态位打开（S0=ON）。这个状态不会转移到任何其他的状态位，仅根据用户在屏里的操控，通过 Modbus 将控制位传到 PLC 来进行相应的操作。按一次 M30 按钮来闭合抓手，按一次 M31 按钮来松开抓手，按下 M32 按钮来上升，按下 M33 按钮来下降，按下 M34 按钮来左移，按下 M35 按钮来右移。

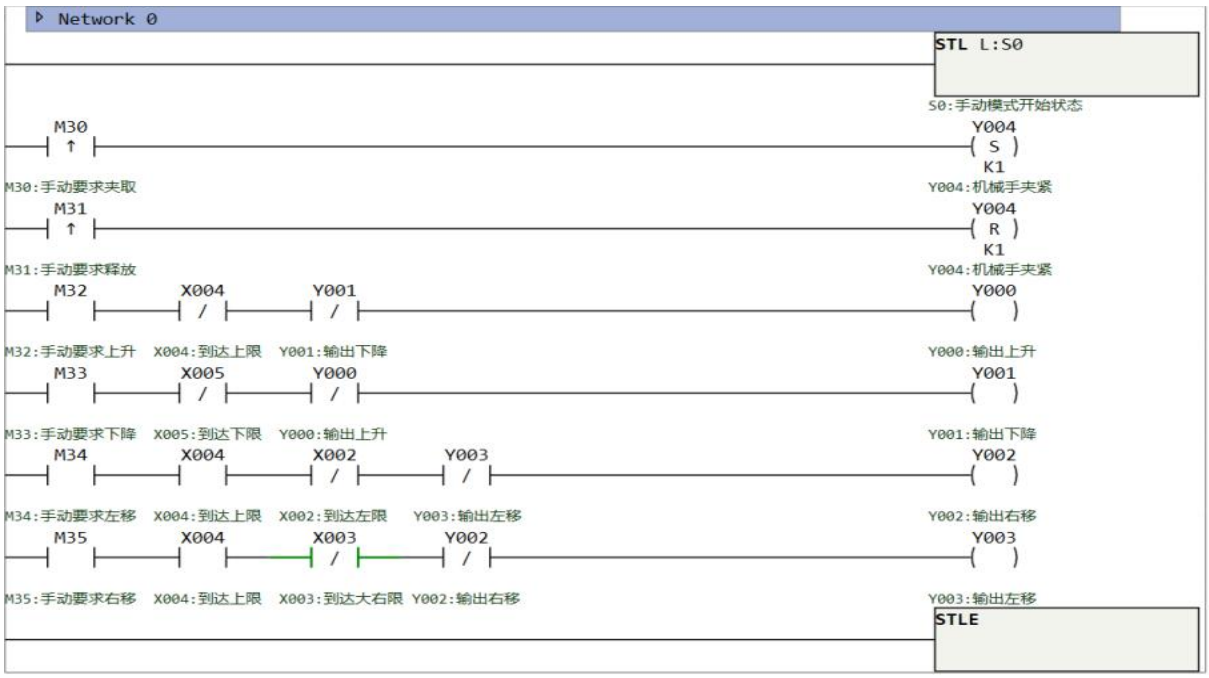


图 1.2 手动模式程序

c) 回原点模式

当 IST 处于这个模式时，需要操纵机械手臂至原位。机械手臂的默认位置是左上角，所以到达最左边和最上面就完成任务了。这里编写 SFC 程序，以 S1 为回原点模式的起始状态位。

回原点完成后复位最后一个状态，记得要置位 M8023（回原点完成）。



图 1.3 回原点模式程序

d) 单步/单周期/自动模式

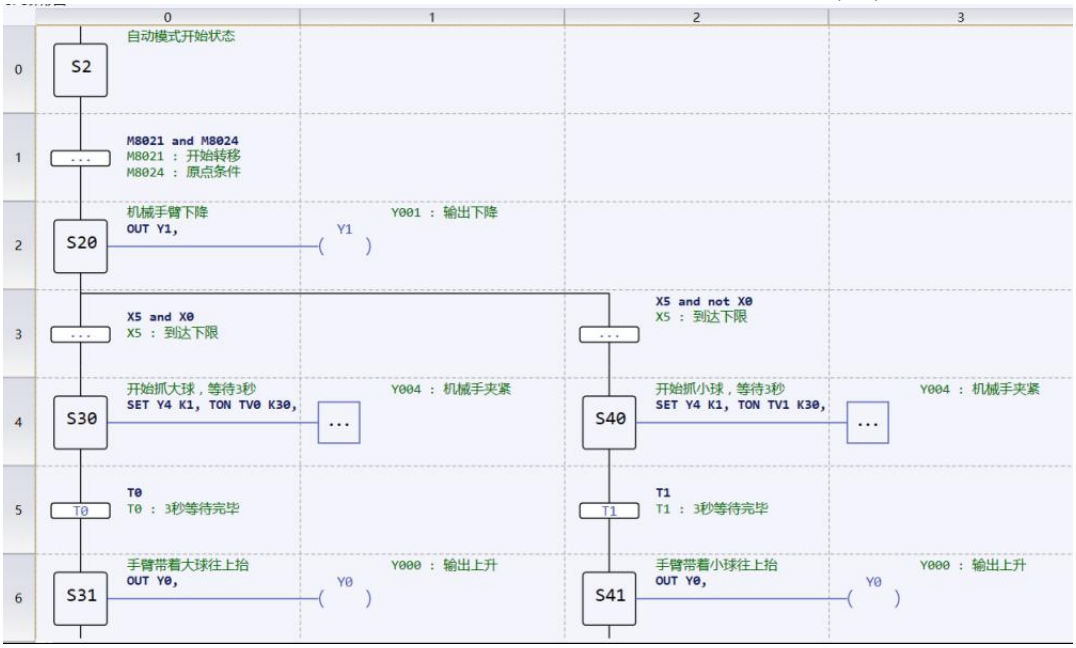
这三个模式共用一个 SFC 程序，从状态 S2 开始，具体的控制由 IST 来处理。区别如下：

1. 单步模式：当进入到某一个状态位后，即使满足条件也不会转移到下一个状态位，只有用户按下启动按钮后（M25 上升沿）才会转移到下一个状态。

2. 单周期模式：这个模式下，从 S2 开始会自动运行一个周期并返回到 S2，之后会等待启动信号（M25 上升沿），才会继续运行下一个周期。

3. 自动模式：这个模式下会自动运行整个流程，全自动的完成工作。

这三个模式来回切换不需要重启这个自动系统，活动状态会保持在当前状态而不是返回 S2。但是三个程序（手动程序，回原点程序，自动程序）之间的切换需要返回初始状态 S0,S1,S2。



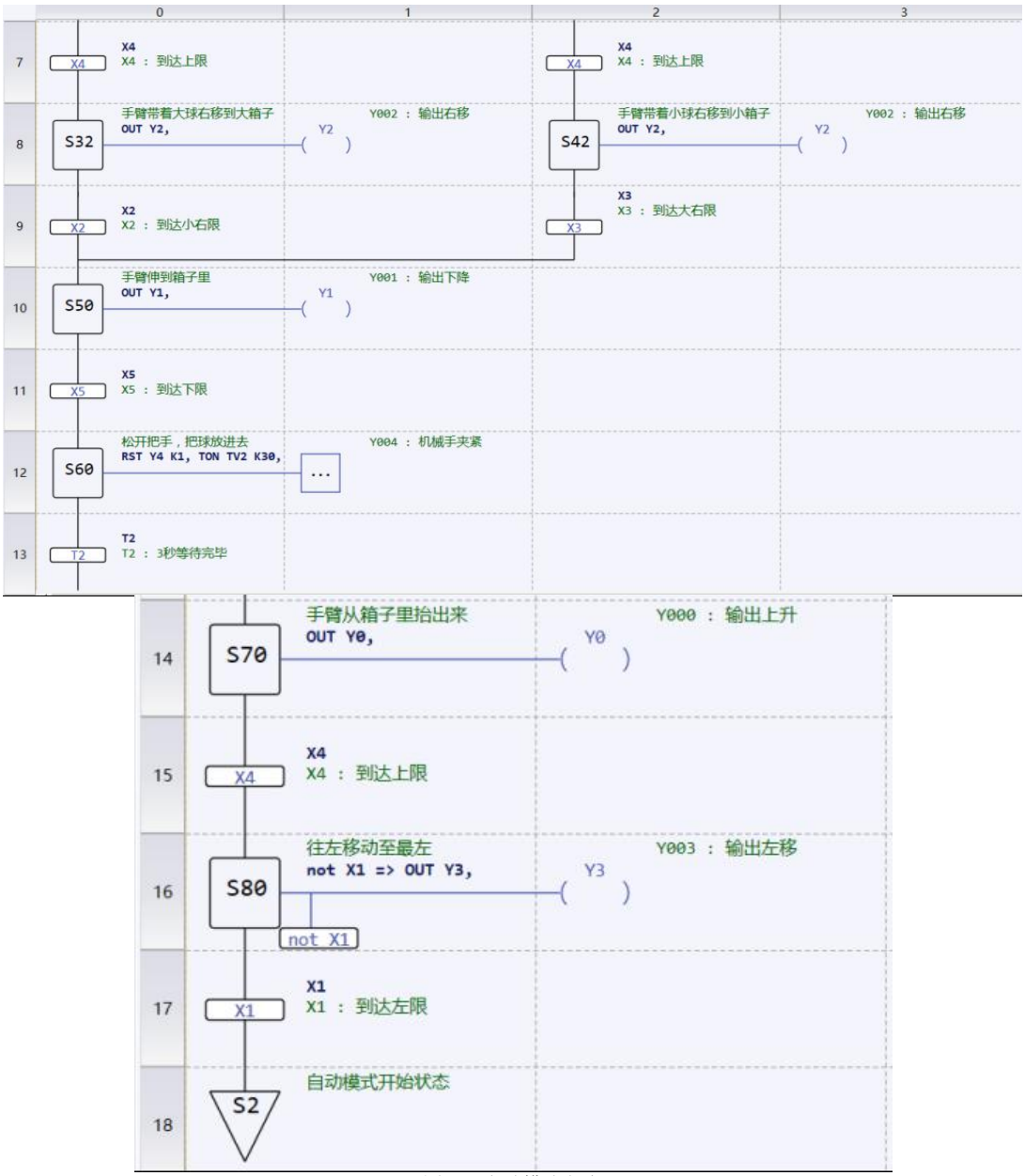


图 1.4 自动模式程序

4.10.9 ISTNEXT

1. 指令说明

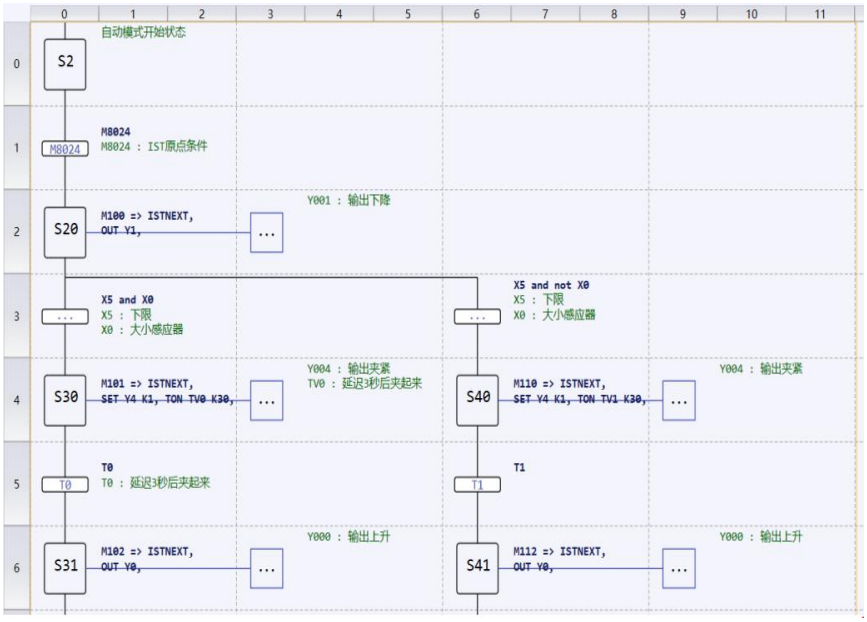
指令 ISTNEXT 在 IST 的单步模式或者单周期模式下可以使用。在单步模式下使用这个指令，可以不用触发起动位就可以继续下一步的状态。在单周期模式下可以继续下一个周期。

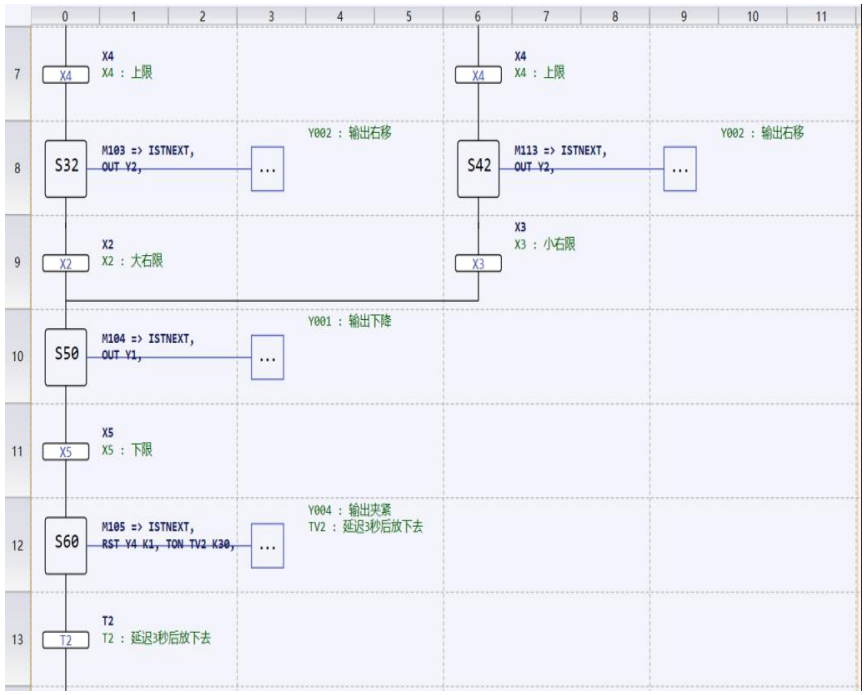
使用 ISTNEXT 指令后，状态转移禁止位 M8020 也会被复位，确保能转移到下一个状态。

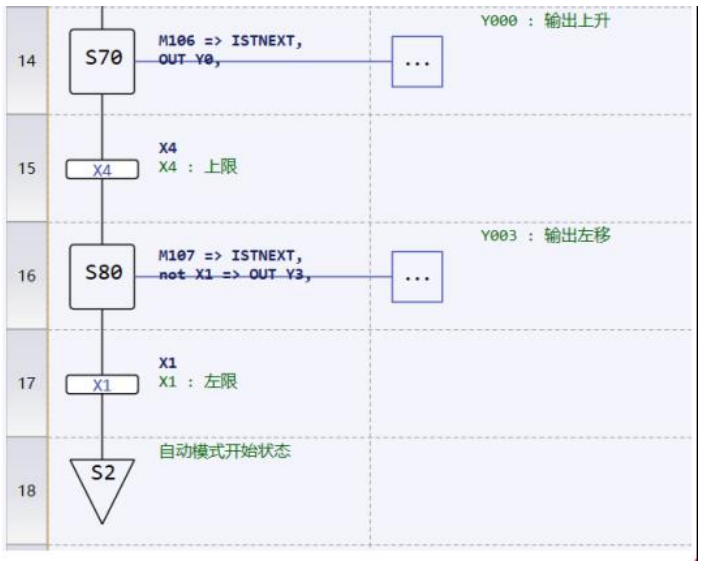
2. 注意要点

l ISTNEXT 使用后，M8020 会被复位为 OFF。在使用 ISTNEXT 之前 M8020 为 ON 的情况，如有需要则重新置位。

3. 使用案例


自动模式开始状态
S2
M8024
M8024 : IST原点条件
S20
M100 => ISTNEXT,
OUT Y1,
Y001 : 输出下降
X5 and X0
X5 : 下限
X0 : 大小感应器
X5 and not X0
X5 : 下限
X0 : 大小感应器
S30
M101 => ISTNEXT,
SET Y4 K1, TON TV0 K30,
Y004 : 输出夹紧
TV0 : 延迟3秒后夹起来
S40
M110 => ISTNEXT,
SET Y4 K1, TON TV1 K30,
Y004 : 输出夹紧
T0
T0 : 延迟3秒后夹起来
T1
S31
M102 => ISTNEXT,
OUT Y0,
Y000 : 输出上升
S41
M112 => ISTNEXT,
OUT Y0,
Y000 : 输出上升


X4
X4 : 上限
S32
M103 => ISTNEXT,
OUT Y2,
Y002 : 输出右移
S42
M113 => ISTNEXT,
OUT Y2,
Y002 : 输出右移
X2
X2 : 大右限
X3
X3 : 小右限
S50
M104 => ISTNEXT,
OUT Y1,
Y001 : 输出下降
X5
X5 : 下限
S60
M105 => ISTNEXT,
RST Y4 K1, TON TV2 K30,
Y004 : 输出夹紧
TV2 : 延迟3秒后放下去
T2
T2 : 延迟3秒后放下去

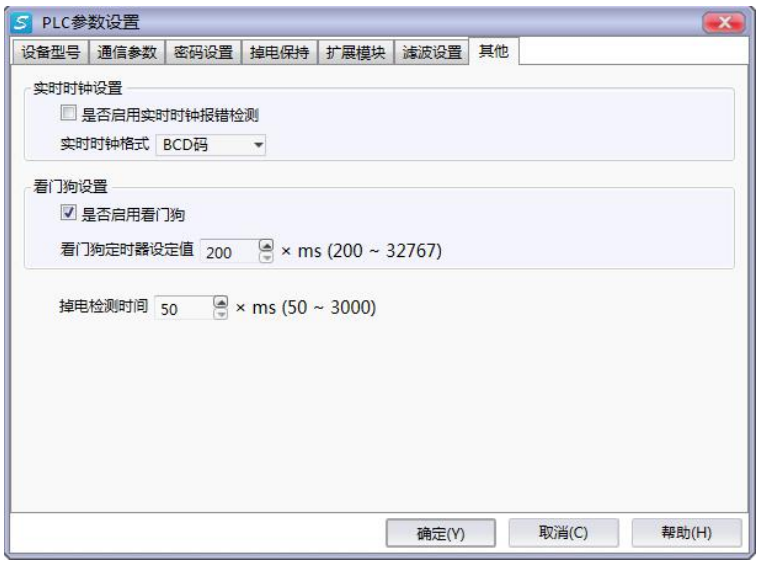

S70
M106 => ISTNEXT,
OUT Y0,
Y000 : 输出上升
X4
X4 : 上限
S80
M107 => ISTNEXT,
not X1 => OUT Y3,
Y003 : 输出左移
X1
X1 : 左限
自动模式开始状态
S2

以上的 SFC 程序是在 IST 指令程序案例的基础上，在自动模式程序里给每个状态添加一个 ISTNEXT 指令，来独立控制这个状态的单步进行。比如说，S20 有了单独的控制信号 M100 来独立控制到 S30 或者 S40 的转移，这个转移可以直接进行而不受 IST 的 M26 的控制。其他状态和配合的控制信号也一样如此

4.10.10 WDT

1. 指令说明

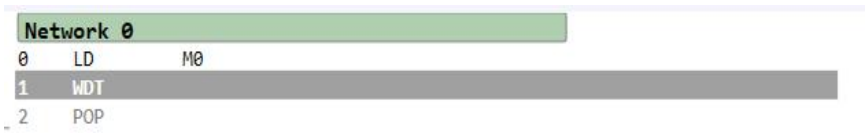
用户可以在 PLC 参数设置界面的其他选项卡中设置启用看门狗功能。看门狗可以检测梯形图或者函数功能块中是否存在死循环导致 PLC 卡死。



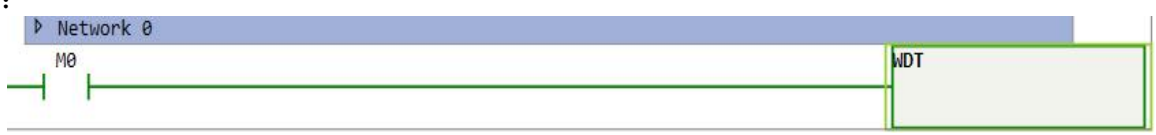
注：
1.在 PLC 运行的每个扫描周期结束时内部会默认对看门狗进行复位，也可通过在梯形图程序中使用 WDT 指令手动复位看门狗；
2.若超过设定值时间后若没有复位则 PLC 会进入 ERROR、STOP 状态，PLC 将停止运行，可通过查看 D8176 是否等于 10，或者通过上位软件监视查看状态栏信息；
3.解决梯形图的问题后重新下载可以恢复；

2. 梯形图实例

指令表：



梯形图：



注：M0 置位后，每次梯形图运行到该指令处都会复位看门狗，重新开始看门狗计时。

4.11 移位指令

内容如下：

1、 字左移 SHL

2、 双字左移 SHLD

3、 字右移 SHR

4、 双字右移 SHRD

5、 位左移 SHLB

6、 位右移 SHRB

7、 循环字左移 ROL

8、 循环双字左移 ROLD

9、 循环字右移 ROR

10、循环双字右移 RORD

4.11.1 SHL/SHLD

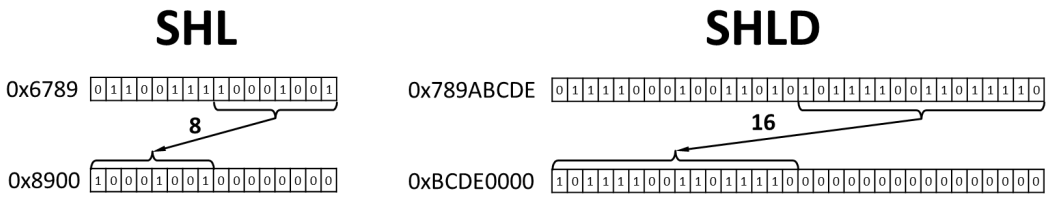
左移操作

对应命令	注释/说明	操作数（寄存器）
SHL	字左移	D/K/H/TV/CV/AI/AO/V/Z,D/K/H,D/TV/CV/AO/V/Z
SHLD	双字左移	D/K/H,D/K/H,D

1. 指令说明

指令将输入寄存器（S）数值向左移动（N）位，并将结果载入输出寄存器（D）。移位指令对每个移出位补 0。如果移位数目（N）大于或等于 16（SHLD 指令是 32），则数值最多被移位 16 次（SHLD 指令是 32）。如果移位次数大于 0，最后一个移位的值会复制到移位指令溢出标志位（M8166）。如果移位操作结果为 0，移位指令零标志（M8167）置 0。

SHL 用于单字，SHLD 用于双字。



2. 指令的有效操作数

输入输出	类型	操作数	描述
EN	BOOL		**启用输入端接通时，指令执行；否则，不执行**
S	字 双字	**D/K/H/TV/CV/AI/AO/V/Z** **D/K/H**	**要移位的操作数**
N	字 双字	**D/K/H** **D/K/H**	**移动的位数**

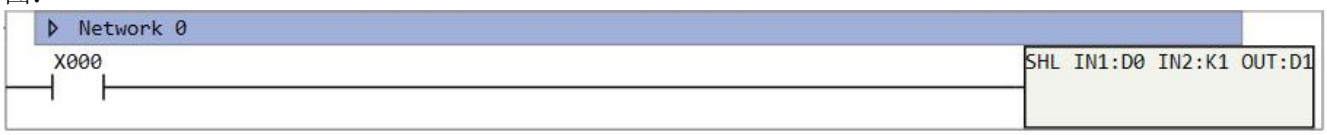
D	字 双字	**D/TV/CV/AO/V/Z D**	**结果寄存器**

3. 实例

指令表：

```
LD      X000
SHL     D0      K1      D1    // 将 D0 左移 1 位输出到 D1
```

梯形图：



4.11.2 SHR/SHRD

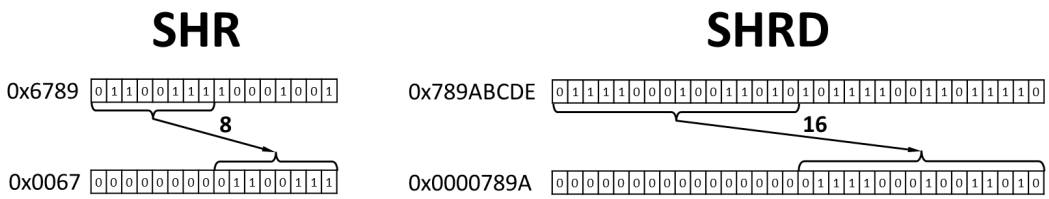
右移操作

对应命令	注释/说明	操作数（寄存器）
SHR	字右移	D/K/H/TV/CV/AI/AO/V/Z,D/K/H,D/TV/CV/AO/V/Z
SHRD	双字右移	D/K/H,D/K/H,D

1. 指令说明

指令将输入字（S）数值向右移动（N）位，并将结果载入输出字（D）。移位指令对每个移出位补 0。如果移位数目（N）大于或等于 16（SHRD 指令是 32），则数值最多被移位 16 次（SHRD 指令是 32）。如果移位次数大于 0，最后一个移位的值会复制到移位指令溢出标志位（M8166）。如果移位操作结果为 0，移位指令零标志（M8167）置 0。

SHR 用于单字　　SHRD 用于双字



2. 指令的有效操作数

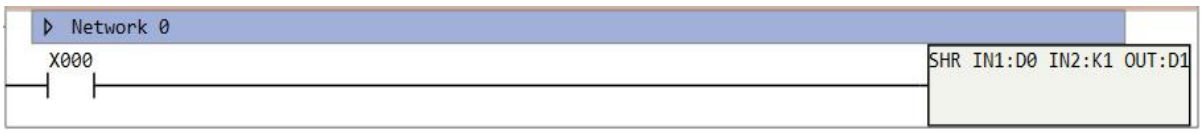
输入输出	类型	操作数	描述
EN	BOOL		**启用输入端接通时，指令执行；否则，不执行**
S	字 双字	**D/K/H/TV/CV/AI/AO/V/Z D/K/H**	**要移位的操作数**
N	字 双字	**D/K/H D/K/H**	**移动的位数**
D	字 双字	**D/TV/CV/AO/V/Z D**	**结果寄存器**

3. 实例

指令表：

```
LD      X000
SHR     D0      K1      D1    // 将 D0 左移 1 位输出到 D1
```

梯形图：



4.11.3 SHLB

1. 指令说明

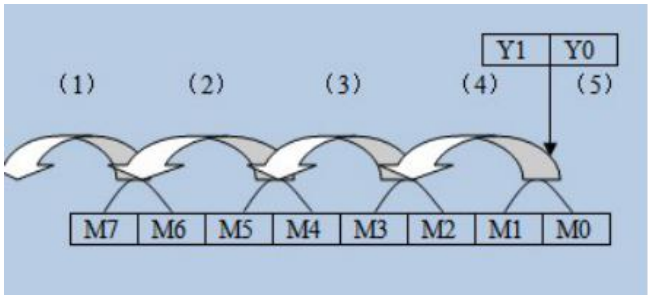
指令将操作数（S）左移（N2）位进入长度（N1）的操作数（D）。N2≤N1≤操作数（D）长度最大值，其中，操作数（S）指定的是其（N2）位的最低位，从操作数（S）指定的（N2）位的最高位开始移入操作数（D）。

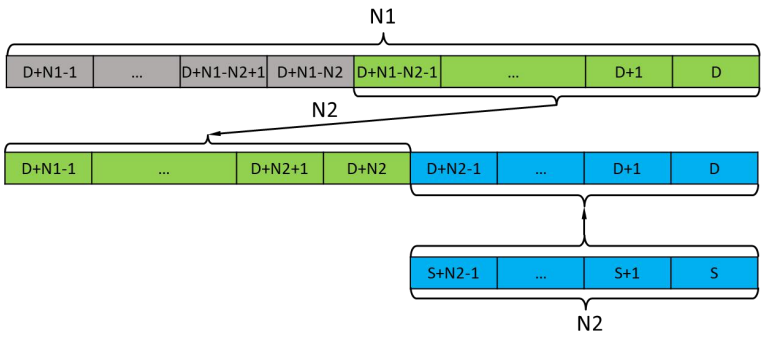
如下：

	[SRC.]	[DST.]	n1	n2
SHLB	Y0	M0	K8	K2

指令执行过程：

(1) M7,M6 → 移出
(2) M5,M4 → M7,M6
(3) M3,M2 → M5,M4
(4) M1,M0 → M3,M2
(5) Y1,Y0 → M1,M0





2. 指令的有效操作数

输入输出	类型	操作数	描述
EN	BOOL		启用输入端接通时，指令执行；否则，不执行
S	BOOL	X/Y/M/S/T/C	要移位的操作数
D	BOOL	Y/M/S/T/C	结果寄存器

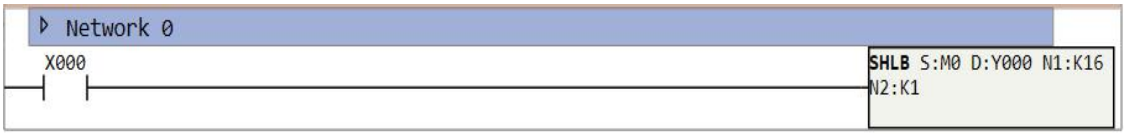
N1	字	**D/K/H**	**结果寄存器中操作数的长度**
N2	字	**D/K/H**	**移动的位数**

3. 实例

指令表：

```
LD      X000
SHLB    M0    Y0    K16    K1    // 将 Y017 移出，再将 Y016 移入 Y017，…，再将 M0 移入 Y0
```

梯形图：



4.11.4 SHRB

1. 指令说明

指令将操作数（S）右移（N2）位进入长度（N1）的操作数（D）。N2≤N1≤操作数（D）长度最大值，其中，操作数（S）指定的是其（N2）位的最低位，从操作数（S）指定的（N2）位的最低位开始移入操作数（D）。

如下：

	[SRC.]	[DST.]	n1	n2
SHRB	Y0	M0	K8	K2

指令执行过程：

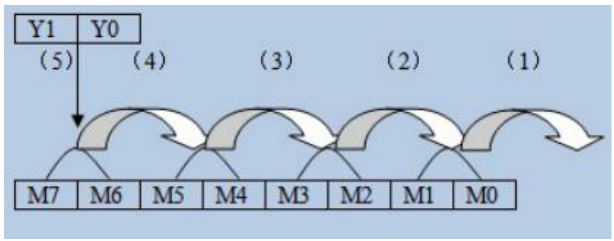
（1）M1,M0 → 移出

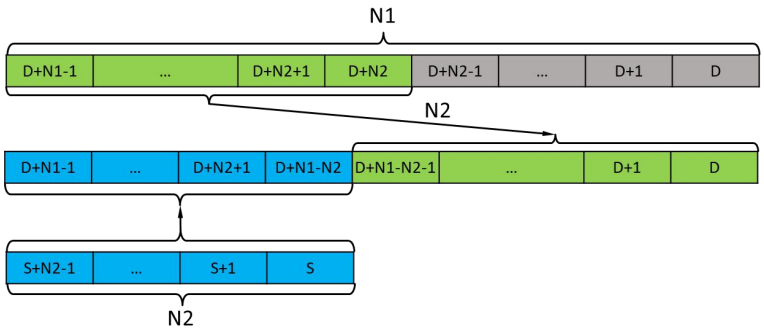
（2）M3,M2 → M1,M0

（3）M5,M4 → M3,M2

（4）M7,M6 → M5,M4

（5）Y1,Y0 → M7,M6





2. 指令的有效操作数

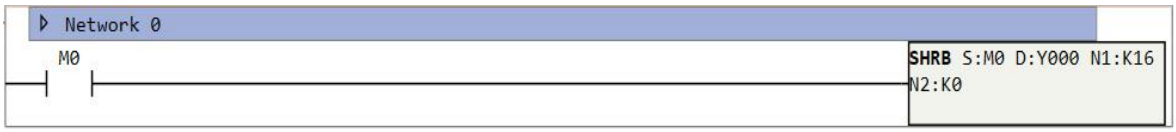
输入输出	类型	操作数	描述
EN	BOOL		**启用输入端接通时，指令执行；否则，不执行**
S	BOOL	**X/Y/M/S/T/C**	**要移位的操作数**
D	BOOL	**Y/M/S/T/C**	**结果寄存器**
N1	字	**D/K/H**	**结果寄存器中操作数的长度**
N2	字	**D/K/H**	**移动的位数**

3. 实例

指令表：

```
LD      X000
SHRB    M0    Y0    K16    K1    // 将 Y017 移出，再将 Y016 移入 Y017，…，再将 M0 移入 Y0
```

梯形图：



4.11.5 ROL/ROLD

循环左移

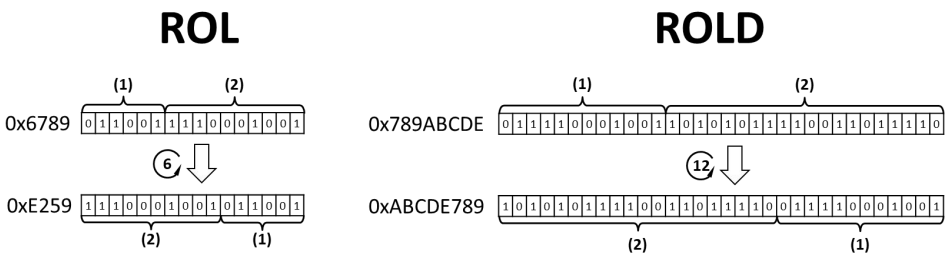
对应命令	注释/说明	操作数（寄存器）
ROL	循环字左移	D/K/H/TV/CV/AI/AO/V/Z,D/K/H,D/TV/CV/AO/V/Z
ROLD	循环双字左移	D/K/H,D/K/H,D

1. 指令说明

指令将输入字（S）数值向左循环移动（N）位，并将结果载入输出字（D）。如果移位数目（N）大于或等于 16（ROLD 是 32），则在执行循环移位之前会执行模数 16 的取模操作（ROLD 是 32），得到一个有效的移位次数（N），从而使移动位数在 0 至 15 之间（ROLD 是 0 至 31 之间）。如果移位次数大于 0，循环移位指令执行，最后一个移位的值会复制到移位指令溢出标志位（M8166）。如果移位次数为 0，循环移位指令不执行，移位指令零标志（M8167）置 0。

当使用有符号数据类型时，符号位也被移位。

ROL 用于单字 ROLD 用于双字



2. 指令的有效操作数

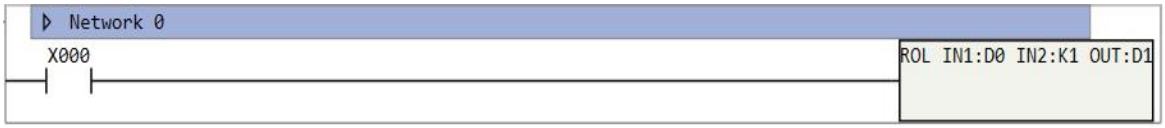
输入输出	类型	操作数	描述
EN	BOOL		**启用输入端接通时，指令执行；否则，不执行**
S	字 双字	**D/K/H/TV/CV/AI/AO/V/Z** **D/K/H**	**要移位的操作数**
N	字 双字	**D/K/H** **D/K/H**	**移动的位数**
D	字 双字	**D/TV/CV/AO/V/Z** **D**	**结果寄存器**

3. 实例

指令表：

```
LD      X000
ROL     D0      K1      D1    // 将 D0 循环左移 1 位输出到 D1
```

梯形图：



4.11.6 ROR/RORD

循环字右移 ROR

对应命令	注释/说明	操作数（寄存器）
ROR	循环字右移	D/K/H/TV/CV/AI/AO/V/Z,D/K/H,D/TV/CV/AO/V/Z
RORD	循环双字右移	D/K/H,D/K/H,D

1. 指令说明

指令将输入字（S）数值向右循环移动（N）位，并将结果载入输出字（D）。如果移位数目（N）大于或等于 16（RORD 是 32），则在执行循环移位之前会执行模数 16 的取模操作（RORD 是 32），得到一个有效的移位次数（N），从而使移动位数在 0 至 15 之间（RORD 是 0 至 31 之间）。如果移位次数大于 0，循环移位指令执行，最后一个移位的值会复制到移位指令溢出标志位（M8166）。如果移位次数为 0，循环移位指令不执行，移位指令零标志（M8167）置 0。

当使用有符号数据类型时，符号位也被移位。

ROR 用于单字 RORD 用于双字



2. 指令的有效操作数

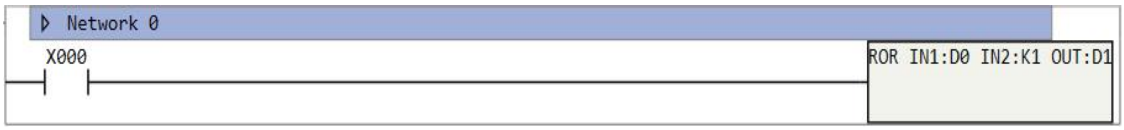
输入输出	类型	操作数	描述
EN	BOOL		启用输入端接通时，指令执行；否则，不执行
S	字	D/K/H/TV/CV/AI/AO/V/Z	要移位的操作数
N	字	D/K/H	移动的位数
D	字	D/TV/CV/AO/V/Z	结果寄存器

3. 实例

指令表：

```
LD      X000
ROR     D0      K1      D1      // 将 D0 循环右移 1 位输出到 D1
```

梯形图：



4.12 中断指令

内容如下：

1、 中断绑定 ATCH

2、 中断释放 DTCH

3、 中断允许 EI

4、 中断禁止 DI

4.12.1 ATCH

1. 指令说明

指令将中断事件号（EVNT）与中断子程序号码（INT）相关联，并启用中断事件。中断事件号（EVNT）见下表：

中断向量表：

中断号	中断事件
K0	X000 上升沿触发
K1	X000 下降沿触发
K2	X000 边沿变化触发
K3	X001 上升沿触发
K4	X001 下降沿触发
K5	X001 边沿变化触发

K6	中断定时器 0 **注：设定值为 D8173，定时范围（1~32767ms）（分辨率 1ms）**
K7	中断定时器 1 **注：设定值为 D8174，定时范围（1~32767ms）（分辨率 1ms）**
仅多脉冲型 PLC FGm_32MT_A, FGm_64MT_A 具有以下中断事件	
K8	Y0 脉冲发送完成中断
K9	Y1 脉冲发送完成中断
K10	Y2 脉冲发送完成中断
K11	Y3 脉冲发送完成中断
K12	Y4 脉冲发送完成中断
K13	Y5 脉冲发送完成中断
K14	Y6 脉冲发送完成中断
K15	Y7 脉冲发送完成中断
K16	Y10 脉冲发送完成中断
K17	Y11 脉冲发送完成中断
K18	C235 计数完成中断
K19	C236 计数完成中断
K20	C237 计数完成中断
K21	C238 计数完成中断
K22	C239 计数完成中断
K23	C240 计数完成中断
K24	C241 计数完成中断
K25	C242 计数完成中断
K26	C243 计数完成中断
K27	C244 计数完成中断
K28	C245 计数完成中断
K29	C246 计数完成中断
K30	X2 上升沿中断
K31	X2 下降沿中断
K32	X2 边沿中断
K33	X3 上升沿中断
K34	X3 下降沿中断
K35	X3 边沿中断
K36	X4 上升沿中断
K37	X4 下降沿中断
K38	X4 边沿中断
K39	X5 上升沿中断
K40	X5 下降沿中断
K41	X5 边沿中断

注意：

（1）输入端口边沿变化中断优先级和定时中断优先级小于其他中断优先级；
（2）当有高速计数或脉冲中断与外部中断同时输入时，外部中断不处理。
（3）当几个外部中断同时输入时，仅处理检测到的第一个外部中断。
（4）所有外部中断的输入总频率不要太高。

2. 指令的有效操作数

输入输出	类型	操作数	描述
EN	BOOL		**启用输入端接通时，指令执行；否则，不执行**
INT			**中断子程序 ID**
EVNT	字	**K**	**中断事件号**

3. 实例

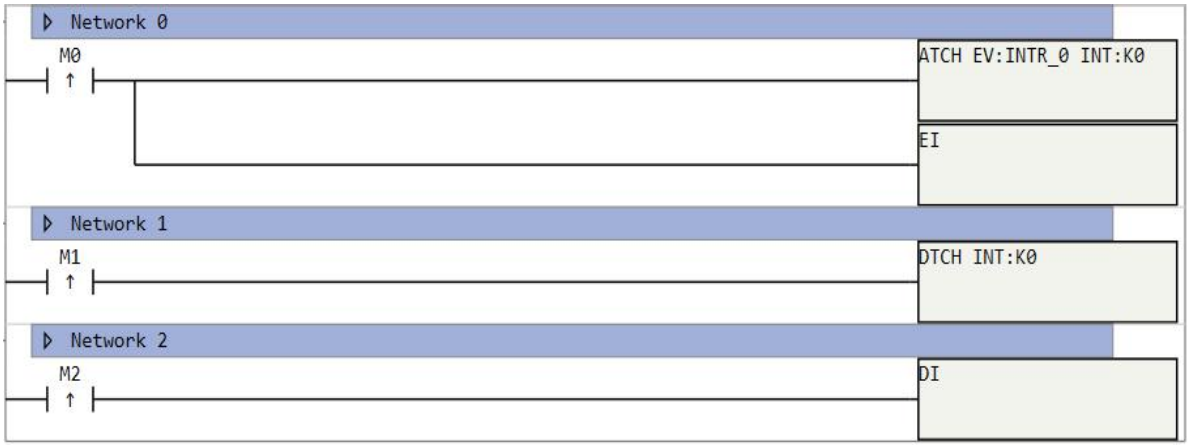
指令表：

```
Network 000
      LDP       M0
      ATCH      INTR_0   K0   // 将中断事件 K0 与中断子程序 INTR_0 关联
      EI                      // 中断开启

Network 001
      LDP       M1
      DTCH      K0             // 中断释放

Network 002
      LDP       M2
      DI                      // 中断禁止
```

梯形图：



4.12.2 DTCH

1. 指令说明

指令将解除中断事件号（EVNT）的关联，并禁用该中断事件。

2. 指令的有效操作数

输入输出	类型	操作数	描述
EN	BOOL		**启用输入端接通时，指令执行；否则，不执行**
EVNT	字	**K**	**中断事件号**

3. 实例

请参见中断绑定 ATCH 的实例(查看 [标题编号])。

4.12.3 EI

1. 指令说明

指令将全局性启用所有被关联的中断事件。要执行中断子程序，必须先开启全局中断。

当进入 RUN 模式时，初始状态为禁止中断。在 RUN 模式下，可以执行中断允许指令(EI)允许所有中断。执行禁用中断指令(DI)可禁止中断过程；但是，激活的中断事件仍继续排队。

2. 指令的有效操作数

输入输出	类型	操作数	描述
EN	BOOL		启用输入端接通时，指令执行；否则，不执行

3. 实例

请参见中断绑定 ATCH 的实例(查看 [标题编号])。

4.12.4 DI

1. 指令说明

指令将全局性禁止所有中断事件进程。

当进入 RUN 模式时，初始状态为禁止中断。在 RUN 模式下，可以执行中断允许指令(EI)允许所有中断。执行禁用中断指令(DI)可禁止中断过程；但是，激活的中断事件仍继续排队。

2. 指令的有效操作数

输入输出	类型	操作数	描述
EN	BOOL		启用输入端接通时，指令执行；否则，不执行

3. 实例

请参见中断绑定 ATCH 的实例(查看 [标题编号])。

4.13 实时时钟

实时时钟

内容如下：

1、 读实时时钟 TRD

2、 写实时时钟 TWR

3、 读实时秒数 TRDS

4、 写实时秒数 TWRS

5、 时钟转秒数 TSEC

6、 秒数转时钟 RSEC

7、 实时时钟比较 TCMP

8、 实时时窗判定 TZCP

9、 星期时间比较 WKCMP

10、 星期时窗判定 WKZCP

11、 刻钟时间比较 CKCMP

12、 刻钟时窗判定 CKZCP

4.13.1 TRD

1. 指令说明

指令从硬件时钟读取当前时间和日期，并将其载入以地址 T 起始的 8 个连续 D 的时间缓冲区。其中，年份用 2 字节表示。

所有日期和时间值必须采用 BCD 格式编码（例如，8211 代表 2013 年）。请参阅下表。

T（Dn – Dn+7）	说明	数据
0	年（2000 -- 2099）	当前年份（BCD 值）
1	月（1 -- 12）	当前月份（BCD 值）
2	日期（1 -- 31）	当前日期（BCD 值）
3	小时（0 -- 23）	当前小时（BCD 值）
4	分钟（0 -- 59）	当前分钟（BCD 值）
5	秒（0 -- 59）	当前秒 （BCD 值）
6	00	保留
7	星期几（0 -- 6）	当前是星期几，0=星期日（BCD 值）

2. 指令的有效操作数

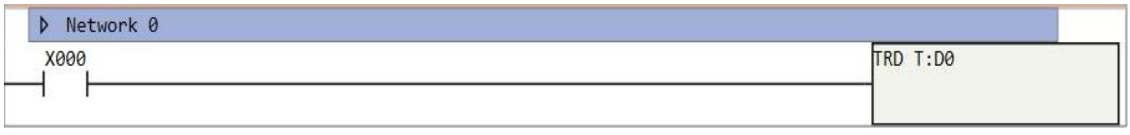
输入输出	类型	操作数	描述
EN	BOOL		启用输入端接通时，指令执行；否则，不执行
T	字	D	读入时间存储的寄存器起始地址

3. 实例

指令表：

```
LD      X000
TRD     D0   // 时间读入 D0~D7
```

梯形图：



4.13.2 TWR

1. 指令说明

指令将当前时间和日期写入硬件时钟，当前时钟存储在以地址 T 开始的 8 字节时间缓冲区中。其中，年份用 2 字节表示。

该指令请采用边沿信号触发。

所有日期和时间值必须采用 BCD 格式编码（例如，8211 代表 2013 年）。请参阅下表。

T（Dn – Dn+7）	说明	数据
0	年（2000 -- 2099）	当前年份（BCD 值）
1	月（1 -- 12）	当前月份（BCD 值）
2	日期（1 -- 31）	当前日期（BCD 值）
3	小时（0 -- 23）	当前小时（BCD 值）
4	分钟（0 -- 59）	当前分钟（BCD 值）
5	秒（0 -- 59）	当前秒（BCD 值）
6	00	保留
7	星期几（0 -- 6）	星期由年月日决定不可独立修改

2. 指令的有效操作数

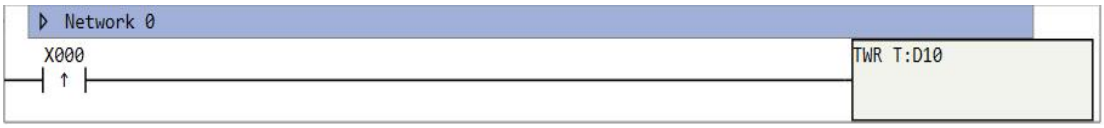
输入输出	类型	操作数	描述
EN	BOOL		**启用输入端接通时(边沿信号)，指令执行；否则，不执行**
T	字	**D**	**存储当前时间和日期的寄存器起始地址**

3. 实例

指令表：

```
LDP     X000
TWR     D0   // 存储在 D0~D7 的当前时间和日期写入硬件时钟
```

梯形图：



4.13.3 TRDS

1. 指令说明

指令从硬件时钟读取当前时间和日期，转换成从 2000/1/1 00:00:00 开始累计经过的秒数，写入指定的双字 D 寄存器里。

2. 指令的有效操作数

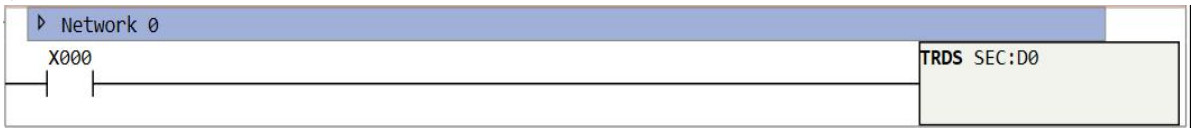
输入输出	类型	操作数	描述
EN	BOOL		**启用输入端接通时(边沿信号)，指令执行；否则，不执行**
T	双字	**D**	**读入秒数的寄存器地址**

3. 实例

指令表：

```
LD      X000
TRDS    D0   // 假设当前时间为 2001/1/1 12:00:00，经过的秒数为(366*24+12)*60*60=31,665,600
```

梯形图：



4.13.4 TWRS

1. 指令说明

给定从 2000/1/1 00:00:00 开始累计经过的秒数，指令计算出当前日期和时间，并写入硬件时钟。

2. 指令的有效操作数

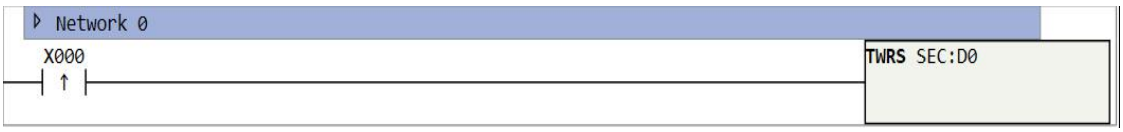
输入输出	类型	操作数	描述
EN	BOOL		**启用输入端接通时(边沿信号)，指令执行；否则，不执行**
T	双字	**D**	**存储当前时间和日期的寄存器起始地址**

3. 实例

指令表：

```
LDP     X000
TWRS    D0
```

梯形图：



4.13.5 TSEC

1. 指令说明

指令将以下格式的日期时间数据，转换为从 2000/1/1 00:00:00 开始累计经过的秒数，写入指定的双字 D 寄存器里。

如果在【PLC 设置】=>【其他】=>【实时时钟格式】里选择了 BCD 格式，则所有日期和时间值必须采用 BCD 格式编码（例如，8211 代表 2013 年）。请参阅下表。

T（Dn – Dn+7）	说明	数据
0	年（2000 -- 2099）	当前年份（BCD 值）
1	月（1 -- 12）	当前月份（BCD 值）
2	日期（1 -- 31）	当前日期（BCD 值）
3	小时（0 -- 23）	当前小时（BCD 值）
4	分钟（0 -- 59）	当前分钟（BCD 值）
5	秒（0 -- 59）	当前秒（ BCD 值）
6	00	保留
7	星期几（0 -- 6）	当前是星期几，0=星期日（BCD 值）

2. 指令的有效操作数

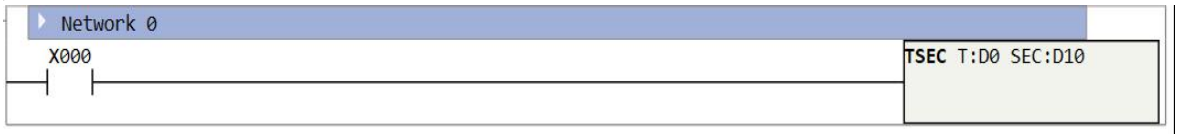
输入输出	类型	操作数	描述
EN	BOOL		启用输入端接通时，指令执行；否则，不执行
T	字	D	读入时间存储的寄存器起始地址
SEC	双字	D	写入秒数的寄存器

3. 实例

指令表：

```
LD      X000
TSEC    D0      D10   // 将时间 D0~D7 转换为秒数存至 D10。
```

梯形图：



4.13.6 RSEC

1. 指令说明

指令将以下格式的日期时间数据，转换为从 2000/1/1 00:00:00 开始累计经过的秒数，写入指定的双字 D 寄存器里。

如果在【PLC 设置】=>【其他】=>【实时时钟格式】里选择了 BCD 格式，则所有日期和时间值必须采用 BCD 格式编码（例如，8211 代表 2013 年）。请参阅下表。

T（Dn – Dn+7）	说明	数据
0	年（2000 -- 2099）	当前年份（BCD 值）
1	月（1 -- 12）	当前月份（BCD 值）
2	日期（1 -- 31）	当前日期（BCD 值）
3	小时（0 -- 23）	当前小时（BCD 值）
4	分钟（0 -- 59）	当前分钟（BCD 值）
5	秒（0 -- 59）	当前秒　（BCD 值）
6	00	保留
7	星期几（0 -- 6）	星期由年月日决定不可独立修改

2. 指令的有效操作数

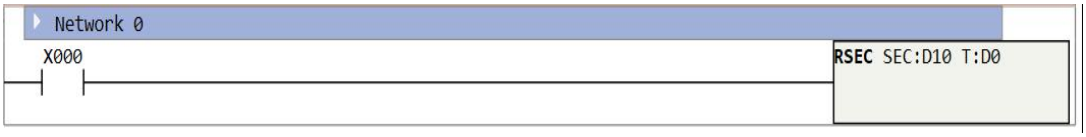
输入输出	类型	操作数	描述
EN	BOOL		启用输入端接通时，指令执行；否则，不执行
SEC	双字	D	读入秒数
T	字	D	写入转换后数据的寄存器起始地址

3. 实例

指令表：

```
LD      X000
RSEC    D10     D0   // 将秒数 D10 转换为时钟数据存储到 D0~D7。
```

梯形图：



4.13.7 TCMP

1. 指令说明

指令将从硬件时钟读取当前时间，和给定的时间数据进行比较，将各个情况的比较结果存入指定的位寄存器组里。

假设要输出到 Y000,Y001,Y002，当前时间没有到指定时间时 **Y000=1**，当前时间正好到达指定时间时 **Y001=1**，当前时间超过了指定时间时 **Y002=1**。

如果在【PLC 设置】=>【其他】=>【实时时钟格式】里选择了 BCD 格式，则所有日期和时间值必须采用 BCD 格式编码（例如，8211 代表 2013 年）。请参阅下表

T（Dn – Dn+7）	说明	数据
0	年（2000 -- 2099）	当前年份（BCD 值）
1	月（1 -- 12）	当前月份（BCD 值）
2	日期（1 -- 31）	当前日期（BCD 值）
3	小时（0 -- 23）	当前小时（BCD 值）
4	分钟（0 -- 59）	当前分钟（BCD 值）
5	秒（0 -- 59）	当前秒 （BCD 值）
6	00	保留
7	星期几（0 -- 6）	当前是星期几，0=星期日（BCD 值）

2. 指令的有效操作数

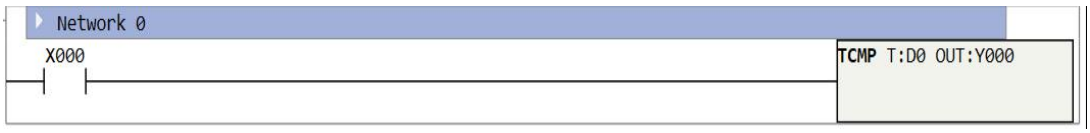
输入输出	类型	操作数	描述
EN	**BOOL**		**启用输入端接通时，指令执行；否则，不执行**
D	**双字**	**D**	**读入指定时间数据的寄存器起始地址**
OUT	**字**	**Y/M/S**	**写入比较结果的位寄存器起始地址**

3. 实例

指令表：

```
LD      X000
TCMP    D0      Y0   // 以 D0~D7 所存储的时间数据为指定时间，输出比较结果到 Y000,Y001,Y002。
```

梯形图：



4.13.8 TZCP

1. 指令说明

指令将从硬件时钟读取当前时间，和给定的时间窗口（由起始时间和结束时间组成的时间段）进行比较，将各个情况的比较结果存入指定的位寄存器组里。

假设要输出到 Y000,Y001,Y002，当前时间没有进入时窗时 **Y000=1**，进入时窗时 **Y001=1**，超过时窗时 **Y002=1**。

如果在【PLC 设置】=>【其他】=>【实时时钟格式】里选择了 BCD 格式，则所有日期和时间值必须采用 BCD 格式编码（例如，8211 代表 2013 年）。请参阅下表。

T（Dn – Dn+7）	说明	数据
0	年（2000 -- 2099）	当前年份（BCD 值）
1	月（1 -- 12）	当前月份（BCD 值）
2	日期（1 -- 31）	当前日期（BCD 值）
3	小时（0 -- 23）	当前小时（BCD 值）
4	分钟（0 -- 59）	当前分钟（BCD 值）
5	秒（0 -- 59）	当前秒（BCD 值）
6	00	保留
7	星期几（0 -- 6）	当前是星期几，0=星期日（BCD 值）

2. 指令的有效操作数

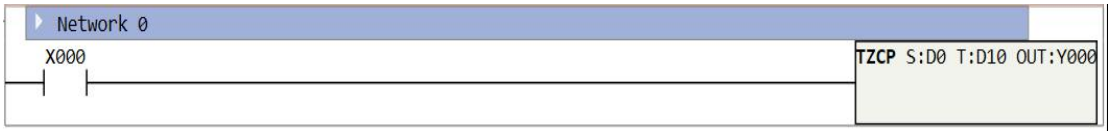
输入输出	类型	操作数	描述
EN	BOOL		**启用输入端接通时，指令执行；否则，不执行**
D	字	**D**	**时窗的起始时间**
D	字	**D**	**时窗的结束时间**
OUT	字	**Y/M/S**	**写入比较结果的位寄存器起始地址**

3. 实例

指令表：

```
LD      X000
TZCP    D0      D10     Y0   // 以 D0~D7 所存储的时间数据为起始时间，D10~D17 为结束时间，输出比较结果到 Y000,Y001,Y002。
```

梯形图：



4.13.9 WKCMP

1. 指令说明

指令将从硬件时钟读取当前时间，和给定的星期时间进行比较，将各个情况的比较结果存入指定的位寄存器组里。

仅比较星期/时/分/秒，日历部分会被忽略。所以时间格式跳过了之前的日历部分。

假设要输出到 Y000,Y001,Y002，当前时间没有到达指定时间时 **Y000=1**，刚好到达指定时间时 **Y001=1**，超过指定时间时 **Y002=1**。

以当前时间 2018/6/26 09:50:30 星期二举例，指定时间为星期二 09:50:30 时刚好相等，指定时间为星期日 08:00:00 时超过了指定时间，指定时间为星期四 11:00:00 时没有到达指定时间。

有以下注意事项：

- **如果在【PLC 设置】=>【其他】=>【实时时钟格式】里选择了 BCD 格式，则所有日期和时间值必须采用 BCD 格式编码（例如，8211 代表 2013 年）。请参阅下表。**
- **使用的时间格式是从小时开始的 5 个字的寄存器组，之前的 8 个字的完整时间格式，需要跳 3 个字作为起始寄存器来带入参数。**
-

T（Dn – Dn+7）	说明	数据
3	小时（0 -- 23）	当前小时（BCD 值）
4	分钟（0 -- 59）	当前分钟（BCD 值）
5	秒（0 -- 59）	当前秒（BCD 值）
6	00	保留
7	星期几（0 -- 6）	当前是星期几，0=星期日（BCD 值）

2. 指令的有效操作数

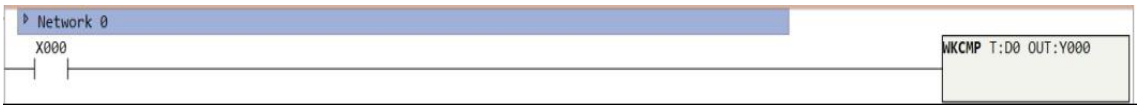
输入输出	类型	操作数	描述
EN	BOOL		**启用输入端接通时，指令执行；否则，不执行**
D	字	**D**	**时窗的起始时间**
D	字	**D**	**时窗的结束时间**

3. 实例

指令表：

```
LD      X000
WKCMP   D0      Y0    // 以 D3~D7 所存储的时间数据为指定时间，输出比较结果到 Y000,Y001,Y002。
```

梯形图：



4.13.10 WKZCP

1. 指令说明

指令将从硬件时钟读取当前时间，和给定的时间窗口（由起始时间和结束时间组成的时间段）进行比较，将各个情况的比较结果存入指定的位寄存器组里。

仅比较星期/时/分/秒，日历部分会被忽略。所以时间格式跳过了之前的日历部分。

假设要输出到 Y000,Y001,Y002，当前时间没有进入时窗时 **Y000=1**，进入时窗时 **Y001=1**，超过时窗时 **Y002=1**。

以当前时间 2018/6/26 09:50:30 星期二举例，时窗为[星期一 09:00:00 星期三 10:00:00]时在时窗里，时窗为[星期日 08:00:00 星期二 09:00:00]时超过了时窗，时窗为[星期二 10:00:00 星期四 11:00:00]时没有进入时窗。

有以下注意事项：

- **如果在【PLC 设置】=>【其他】=>【实时时钟格式】里选择了 BCD 格式，则所有日期和时间值必须采用 BCD 格式编码（例如，8211 代表 2013 年）。请参阅下表。**
- **使用的时间格式是从小时开始的 5 个字的寄存器组，之前的 8 个字的完整时间格式，需要跳 3 个字作为起始寄存器来带入参数。**
- **起始时间必须不超过结束时间，像[星期六 23:59:59 星期日 00:00:00]这样的时窗是非法的。**

T (Dn-Dn +7)	说明	数据
3	小时 （ 0 -- 23 ）	当前小时 （BCD 值）
4	分钟 （ 0 -- 59 ）	当前分钟 （BCD 值）
5	秒 （ 0 -- 59 ）	当前秒 （BCD 值）
6	00	保留
7	星期几 （ 0 -- 6）	当前是星期几， 0=星期日（BCD 值）

2. 指令的有效操作数

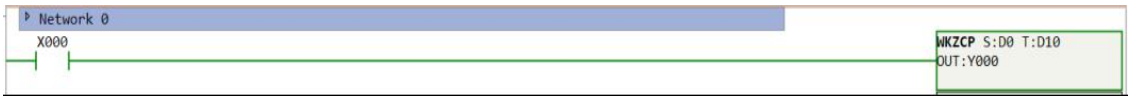
输入输出	类型	操作数	描述
EN	BOOL		**启用输入端接通时，指令执行；否则，不执行**
D	字	**D**	**时窗的起始时间**
D	字	**D**	**时窗的结束时间**
OUT	字	**Y/M/S**	**写入比较结果的位寄存器起始地址**

3. 实例

指令表：

```
LD      X000
WKZCP   D0      D10     Y0   // 以 D3~D7 所存储的时间数据为起始时间，D13~D17 为结束时间，输出比较结果到 Y000,Y001,Y002。
```

梯形图：



4.13.11 CKCMP

1. 指令说明

指令将从硬件时钟读取当前时间，和给定的星期时间进行比较，将各个情况的比较结果存入指定的位寄存器组里。

仅比较时/分/秒，日历部分会被忽略。所以时间格式跳过了之前的日历部分。

假设要输出到 Y000,Y001,Y002，当前时间没有到达指定时间时 **Y000=1**，刚好到达指定时间时 **Y001=1**，超过指定时间时 **Y002=1**。

以当前时间 2018/6/26 09:50:30 星期二举例，指定时间为 09:50:30 时刚好相等，指定时间为 08:00:00 时超过了指定时间，指定时间为 11:00:00 时没有到达指定时间。

有以下注意事项：

- 如果在【PLC 设置】=>【其他】=>【实时时钟格式】里选择了 BCD 格式，则所有日期和时间值必须采用 BCD 格式编码（例如，8211 代表 2013 年）。请参阅下表。
- 使用的时间格式是从小时开始的 3 个字的寄存器组，之前的 8 个字的完整时间格式，需要跳 3 个字作为起始寄存器来带入参数。

T (Dn-Dn +7)	说明	数据
3	小时 （ 0 -- 23 ）	当前小时 （BCD 值）
4	分钟 （ 0 -- 59 ）	当前分钟 （BCD 值）
5	秒 （ 0 -- 59 ）	当前秒 （BCD 值）

2. 指令的有效操作数

输入输出	类型	操作数	描述
EN	BOOL		**启用输入端接通时，指令执行；否则，不执行**
D	字	**D**	**指定的刻钟时间**
OUT	字	**Y/M/S**	**写入比较结果的位寄存器起始地址**

3. 实例

指令表：

```
LD      X000
CKCMP   D0      Y0   // 以 D3~D7 所存储的时间数据为指定时间，输出比较结果到
```

Y000,Y001,Y002。

梯形图：



4.13.12 CKZCP

1. 指令说明

指令将从硬件时钟读取当前时间，和给定的时间窗口（由起始时间和结束时间组成的时间段）进行比较，将各个情况的比较结果存入指定的位寄存器组里。

仅比较时/分/秒，日历和星期部分会被忽略。所以时间格式跳过了之前的日历部分，以及最后两个字，仅保留刻钟部分。

假设要输出到 Y000,Y001,Y002，当前时间没有进入时窗时 **Y000=1**，进入时窗时 **Y001=1**，超过时窗时 **Y002=1**。

以当前时间 2018/6/26 09:50:30 举例，时窗为[09:00:00 10:00:00]时在时窗里，时窗为[08:00:00 09:00:00]时超过了时窗，时窗为[10:00:00 11:00:00]时没有进入时窗。

有以下注意事项：

- **如果在【PLC 设置】=>【其他】=>【实时时钟格式】里选择了 BCD 格式，则所有日期和时间值必须采用 BCD 格式编码（例如，8211 代表 2013 年）。请参阅下表。**
- **使用的时间格式是从小时开始的 3 个字的寄存器组，之前的 8 个字的完整时间格式，需要跳 3 个字作为起始寄存器来带入参数。**
- **起始时间必须不超过结束时间，像[23:59:59 00:00:00]这样的时窗是非法的。**

T (Dn-Dn +7)	说明	数据
3	小时 （ 0 -- 23 ）	当前小时 （BCD 值）
4	分钟 （ 0 -- 59 ）	当前分钟 （BCD 值）
5	秒 （ 0 -- 59 ）	当前秒 （BCD 值）

2. 指令的有效操作数

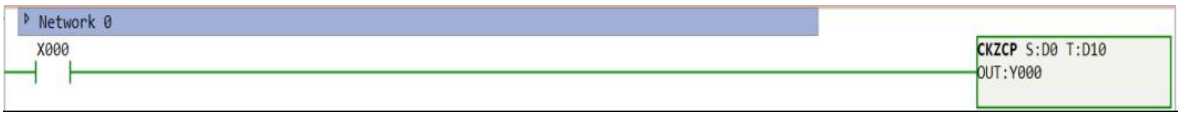
输入输出	类型	操作数	描述
EN	BOOL		**启用输入端接通时，指令执行；否则，不执行**
D	字	**D**	**时窗的起始时间**
D	字	**D**	**时窗的结束时间**
OUT	字	**Y/M/S**	**写入比较结果的位寄存器起始地址**

3. 实例

指令表：

```
LD      X000
CKZCP   D0      D10     Y0   // 以 D3~D7 所存储的时间数据为起始时间，D13~D17 为结束时间，输出比较结果到 Y000,Y001,Y002。
```

梯形图：



4.14 通信指令

内容如下：

1、 Modbus 主站通信 MBUS

2、 自由口通信 SEND

3、 自由口通信 REV

4、 以太网主站通信绑定远端 IP 指令 DESTIP

5、 ModbusTCP 以太网主站通信指令 MBUSTCP

6、 CRC 校验生成 CRC

7、 CRC 校验 CRCCHECK

4.14.1 MBUS

1. 指令说明

1、**COM_ID**：通信口选择，可选择 K0（232）或 K1（485）；

2、**Table**：Modbus 主站表格 ID，可在工程管理窗口中添加所需的 Modbus 主站表格；

3、**WR**：用来存储通信信息代码(见下表)；

4、**WR_ID**：用来存储当前主站表格的 Modbus 命令序列编号，表示当前所运行至该表格的哪一条命令序列，编号为 0~n-1，其中 n 为该表格内的命令序列总数；

5、修改配置信息：选择 COM 口通信参数设置，如主站站号、波特率、校验位、超时时间、重传次数等，然后选择 Modbus 主站模式；

注意：

1、要使用该指令的话必须将所设置的 COM 口（232/485）设置为主站模式并勾选配置信息下载到 PLC 中，否则不会执行通信命令，具体设置方式请参照 PLC 参数设置页面内容；

2、该指令的第三个操作数 WR 虽然只需输入 Dn 来表示通信信息代码，但 Dn+1 也是有实际意义的，可用来返回当前主站表格的 Modbus 命令编号，而且 Dn+1 不可在其它地方修改数值；

3、对于 Modbus 主站可设置串口通信的超时时间，默认为 200ms，此外还可设置重传次数，默认为 3。比如：当设置超时时间为 200ms、重传次数为 3 时，表示若 200ms 内从站不返回，PLC 将重新发送一次数据包，若累计超过所设定的 3 次重传次数后从站仍不回应（此时 PLC 共发送了 4 次数据包），则返回通讯超时错误代码 16 即 0x10；

4、超时时间在某些时候可能需要用户根据实际情况而略作调整，比如当波特率较低时，由于数据传输速度较慢，因此需要将超时时间适当增大；另外若每次传输数据包字节数较多（比如每次读写从站长度较大，造成发送或接收该包数据耗时较长）时，也需要将超时时间增大，否则可能会报通讯超时错误；

5、PLC 运行中返回的通信信息代码存放在所设置的 Dn 寄存器中，该数值是一个十进制数，而下面表格

中的均是十六进制数，如：通信返回 Dn 值=16，转换成十六进制=0x10，即表示通讯超时；

6、对于同一个 COM 口，在同一时刻不可能同时与多个从站进行数据通信，PLC 内部访问外部从站设备均是一个一个顺序执行的，用户在编程时应考虑到这点，若其中一个从站设备断开，会导致整个系统在该从站处因多次重传而卡顿一段时间，该卡顿时间即为：（重传次数+1）*超时时间，因此如若需要提高整个系统的快速响应可以考虑减少重传次数；

通信信息代码：

代码	信息
0x01	非法功能码
0x02	非法地址
0x03	非法数据
0x04	从站操作失败
0x05	命令有效，正在处理中
0x06	从机忙
0x07	非法数据类型
0x08	非法命令编号
0x09	非法密码
0x10	通讯超时
0x11	数值错误：数据小于 0 或者大于 255
0x12	主从站设置错误
0x13	实际所接收数据中的站号与所期望的站号不一致
0x14	元件地址溢出：发送的数据量超出元件存储空间
0x15	命令执行失败
0x18	接收信息帧错误，包括接收长度校验错误、接收 CRC 校检错误
0x20	参数不可修改
0x21	参数运行时不可修改
0x22	参数受密码保护

2. 指令的有效操作数

输入输出	类型		操作数	描述
EN	BOOL			**启用输入端接通时，指令执行；否则，不执行**
COM_ID				**选择的通讯口**
Table				**Modbus 主站表标识**
WR	字		**D**	**用来存储通信信息代码**
WR_ID	字		**D**	**当前主站表格的 Modbus 命令序列编号**

3. 基本使用方法

右键点击工程管理窗口“工程”-->“Modbus 表格”文件夹，点击“新建 Modbus 表格”，然后弹出新建对话框。

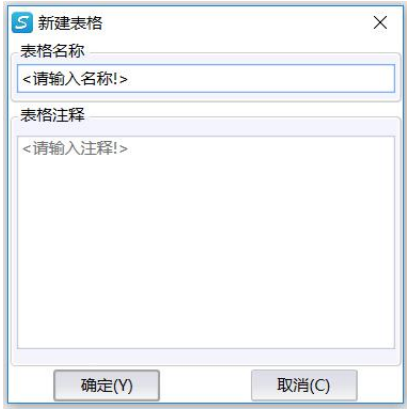


图 1.1 新建表格

输入表格名称（这里是“Modbus 主站表”），然后单击确定按钮。就可以添加并编辑这个表格了。界面的右侧菜单栏提供了对 Modbus 表格的编辑功能。



图 1.2 右侧菜单

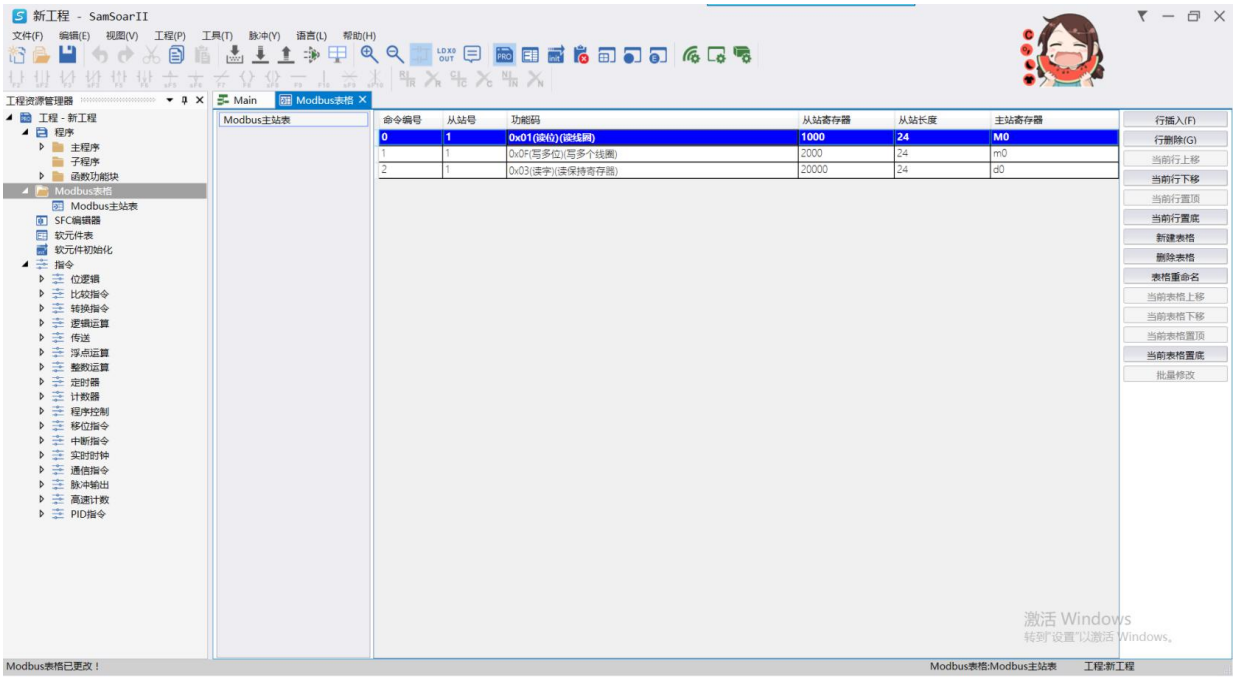


图 1.3 Modbus 表格编辑

小提示：创建 Modbus 表格后，将其从工程管理窗口拖入梯形图，就能创建这个表格的 MBUS 指令了。

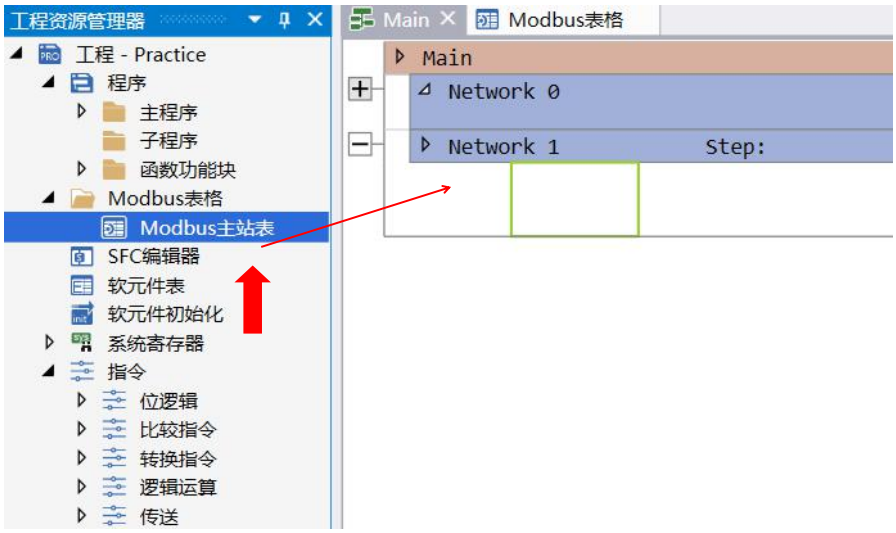
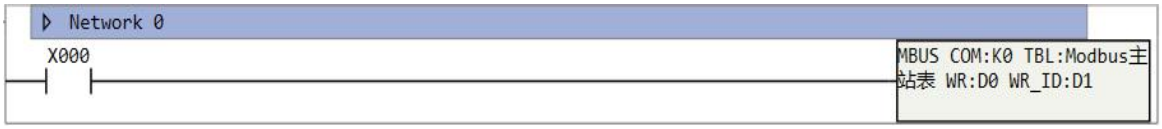


图 1.4 鼠标拖动

指令表：

```
LD      X000
MBUS    K0      Modbus 主站表    D0   D1   // K0 表示 COM 口 232，使用“Modbus 主站表”，D0 存储通信信息，D1 存储当前正在执行的 Modbus 命令编号
```

梯形图：



PLC 梯形图运行流程：

1、先执行从站站号为 1 的表格命令，PLC 向从站站号为 1 的设备发送数据包，并等待该从站返回信息，此时 D1=0；

2、在这段时间内，即便 PLC 扫描到该指令也不会执行相应命令，直到第一个表格命令中的从站设备正确返回或大于超时时间，才会执行该 Modbus 表格中之后的命令；

3、第一个表格命令执行完后，当梯形图再次扫描到该指令，将顺序执行该表格的第二条命令，即 PLC 会向从站站号为 5 的设备发送数据包，并等待该从站返回信息，此时 D1=1；

4、第三个表格命令与前面两条类似，依次循环执行，此时 D1=2；

5、需注意的是，对于同一个 COM 口，不可能同时与多台从站设备进行通信，它们之间必然是有着先后顺序的，只是这个顺序执行的速度很快，甚至可看做是同步执行；

4. 扩展应用

PLC 的 Modbus 主站通信指令在使用上非常灵活，适用于多种不同的额场合及要求，

（1）主站须同时和所有从站一直进行通信

这种情况对于编写梯形图程序来说是最简单的，只需将所有待读写的从站地址信息建立在同一个 Modbus 主站表格中即可，在整个梯形图程序中对于该 COM 口有且只需一条 Modbus 主站通信指令，用户可以用某个使能端来进行控制，用户无需处理多　　个从站之间的转换，也无需理解其如何实现，因为 PLC 内部会自动对其进行切换处理，且是按照该 Modbus 表格内的从站序列号 WR_ID 顺序自动进行切换；

该方法的好处是梯形图处理简单，只需一条 Modbus 指令，以及建立一个 Modbus 表格，用户编写梯形图程序较容易；

该方法的坏处是当所挂载的从站较多或待读写的从站信息序列较多时，由于每次以轮询的方式进行切换每一个从站信息，导致耗时较长；

因此该方法适用于主站必须一直保持与所有从站进行通信的使用场合，或者从站以及读写从站命令较少的场合；

（2）主站根据用户选择和其中某些从站进行通信，而不与其它未选择的从站进行通信

在某些大型使用场合下，更多的情况是由用户选择来选择性地与某些从站进行通信，而未选择到的从站则无需对其进行读写通信操作；

并且该方法可以实现任意形式的通信，当然也包括了（1），下面会给出一个简单示例说明，

该方法的好处是选择性地与所挂载的某些从站进行通信，因此通信速度相比（1）的轮询所有从站要快；

该方法的坏处是梯形图编写较为复杂，需要利用某些串口通讯标志如 M8182/M8183 等由用户手动进行控制不同 Modbus 表格以及 Modbus 指令的切换；

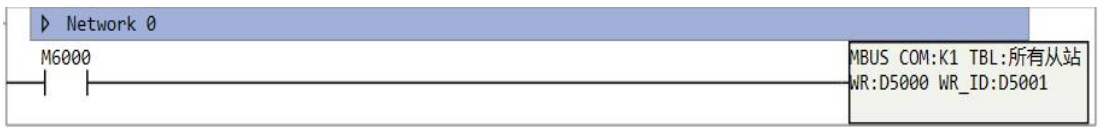
示例 1.

如下图所示，建立一个名为“所有从站”的的 Modbus 表，主站挂载了三个从站，其从站站号分别为 1、2、3，其中每个从站均有 3 条不同的读写序列，

所有从站
站号1
站号2
站号3

从站号	功能码	从站寄存器	从站长度	主站寄存器
1	0x03(读双字)	60200	1	d0
1	0x01(读位)	30000	10	m0
1	0x03(读字)	40000	10	d100
2	0x03(读双字)	60200	1	d200
2	0x01(读位)	30000	10	m100
2	0x03(读字)	40000	10	d300
3	0x03(读双字)	60200	1	d400
3	0x01(读位)	30000	10	m200
3	0x03(读字)	40000	10	d500

表格建好后选择 Modbus 主站通信协议配置，核对好从站串口信息以及站号设置，主站梯形图如下图所示，下载至 PLC 中，通过监视观察是否正常运行，且 WR_ID 一直在从站序列 0~8 之间快速自动切换



示例 2.

接下来给出一种同样可以实现示例 1 的方法，该方法稍显复杂，但理解其原理后会对 Modbus 主站通信指令有更深的了解；

首先将示例 1 中的三个从站分别拆开作为不同的 Modbus 表格，如下图所示分别建立 3 个从站表格；

所有从站
站号1
站号2
站号3

从站号	功能码	从站寄存器	从站长度	主站寄存器
1	0x03(读双字)	60200	1	d0
1	0x01(读位)	30000	10	m0
1	0x03(读字)	40000	10	d100

所有从站
站号1
站号2
站号3

从站号	功能码	从站寄存器	从站长度	主站寄存器
2	0x03(读双字)	60200	1	d200
2	0x01(读位)	30000	10	m100
2	0x03(读字)	40000	10	d300

所有从站
站号1
站号2
站号3

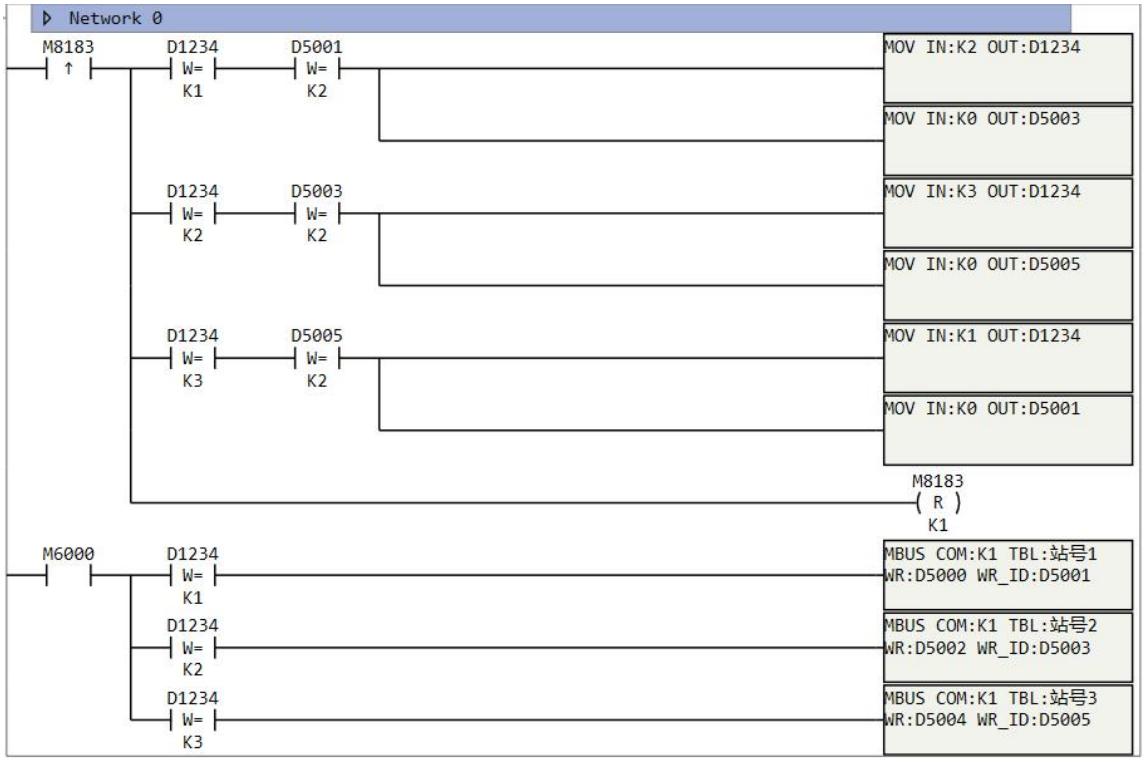
从站号	功能码	从站寄存器	从站长度	主站寄存器
3	0x03(读双字)	60200	1	d400
3	0x01(读位)	30000	10	m200
3	0x03(读字)	40000	10	d500

编写如下所示的梯形图，其中 D1234 用于切换控制不同的 Modbus 表，使能不同的 Modbus 指令，比如当（D1234 == 1）时代表控制 1 号从站，当（D1234 == 3）时代表控制 3 号从站；

M8183 是 485 串口通信接收完成标志，该标志由 PLC 内部在接收到数据后自动置位，需要由用户在梯形图中手动复位，该特殊寄存器是用于手动切换的关键标志；

梯形图第二个网络中的 D5001、D5003、D5005 与 K2 进行比较判断的含义：K2 代表每个 Modbus 表格的从站信息序列，该示例由于三个表格内总共均有 3 个序列，每个表格内序列号依次为：0、1、2，因此这里用 D5001、D5003、D5005 与 K2 进行比较是表示与其相对应的 Modbus 指令运行到该表格的序列末端，实际工程中不同表格内的序列号总数可以不同，但每个表格内的末端序列号一定是该表格的序列总数 n-1，即 WR_ID 分别为：0、1、2 ... n-2、n-1；

将该程序下载至 PLC 中，可以监视调试将 M6000 置位，并将 D1234 置 1，之后可以观察到 PLC 会自动在三个 Modbus 指令之间进行快速切换，且与多个从站之间的读写通信也是正常的。



如果能理解示例 2 中的程序，那么若想实现选择性地手动切换控制与不同的从站进行通信也就非常简单明了了；

试想像一个场景，当 HMI 进入画面 1 时用户需与 1 号从站通信且不与其它从站通信，进入画面 2 时需与 2 号从站通信且不与其它从站通信，... 进入画面 n 时需与 n 号从站通信且不与其它从站通信；该使用场景中的不同画面相当于上面示例 2 中的 D1234，多个不同从站有多个不同的 Modbus 表格以及相对应的 Modbus 指令，示例 2 中实现了自动循环切换，而该使用场景则是由用户给 D1234 寄存器赋不同的值来进行手动切换，其基本思路是一致的，只要灵活使用 M8182/M8183 串口接收完成标志并配合 Modbus 指令中的 WR_ID 即可实现任意方式的通信；注意还有一个关键点在于：梯形图中对于同一个 COM 口允许可以包含多个 Modbus 指令，但是，同一个扫描周期内对于同一个 COM 口有且只能同时导通一个 Modbus 指令，若同时导通了多个 Modbus 指令则会发生数据通信异常。

4.14.2 SEND

1. 指令说明

1、**COM_ID**（通信口选择）:可选择 COM0（232）或 COM1（485）；

2、**Addr**（起始地址）：该参数地址是以 Dn 为起始地址的一片区域，该操作数仅能选择 D 数据寄存器；

3、**Len**（数据长度）：设置待发送数据的长度，该操作数可选择 K/H/D，对于不同型号 PLC 均有其相应的发送最大限制；

4、修改配置信息：选择 COM 口通信参数设置(查看 [标题编号])，如波特率、校验位等，然后选择自由口通信模式，以及数据缓冲位数；

通讯数据缓冲位说明：

8 位：通讯时 D 寄存器的高 8 位字节是无效的，PLC 只利用 D 寄存器的低 8 位字节进行发送和接收数据；

16 位：通讯时在发送过程中，PLC 先发送 D 寄存器中的低 8 位字节数据，然后再发送高 8 位字节数据；

需注意的是，无论选择 8 位还是 16 位数据缓冲位，其发送长度 Len 均为字节长度，当选择 8 位时，一个 D 寄存器就是一个字节长度，而当选择 16 位时，一个 D 寄存器则是 2 个字节长度；

COMn 正在发送标志位(Only Read)：

寄存器	串口 COMn
M_8176	COM0（232）
M_8177	COM1（485）

COMn 发送完成标志位(Read or Write)：

寄存器	串口 COMn
M_8180	COM0（232）
M_8181	COM1（485）

注意：

1、使用该指令的前提是必须将相应的 COM 口（232/485）设置为自由口通讯模式，具体设置方式请参照 PLC 参数设置页面内容；

2、当设置某 COM 口模式为自由口通讯后，该 COM 口则不能再用来进行程序下载和数据监视，也不能用作 Modbus 主、从站间进行通讯；

3、在没有 USB 的情况下，建议只使用 COM1（485）口用作通讯，COM0（232）仅留作编程以及执行数据监视功能；

4、在修改了 PLC 参数设置后，需勾选配置信息，然后下载到 PLC 中，否则参数修改无效；

5、由于受 PLC 数据缓存大小的限制，因此发送数据长度 Len 不能超过其最大限制，对于 16 或 32 点 PLC 最大为 256Byte，而 64 点 PLC 最大为 512Byte；

6、自由口通讯传输属于半双工模式，即发送指令 SEND 和接收指令 REV 不能同时进行；

2. 指令的有效操作数：

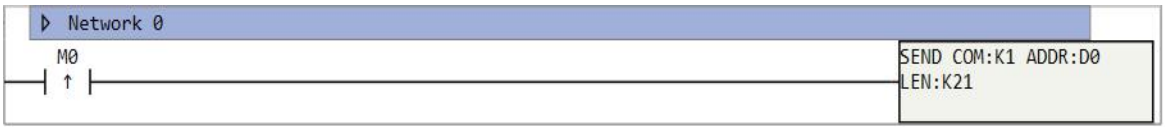
输入输出	类型	操作数	描述
EN	BOOL		启用输入端接通时，指令执行；否则，不执行
COM_ID			选择的通讯口
Addr	字	D	要发送的数据的存储起始地址
Len	字	K/H/D	要发送的数据长度

3. 实例 1

指令表：

```
Network 000
  LDP    M000
  SEND   K1     D0     K21   // 选择 COM1（485），起始地址设为 D0，待发送数据长度为 K21；
                              // 若数据缓冲位为 8，则会将 D0-D20 寄存器中的所有低 8 位数据，共 21Byte 的数据发送至 COM1（485）；
                              // 若数据缓冲位为 16，则会将 D0-D10 寄存器中的高 8 位、低 8 位数据以及 D11 的低 8 位数据，共 21Byte 的数据发送至 COM1（485）；
                              // 此处 SEND 指令的使能端为上升沿，即每来一次上升沿便发送一次，如若选择常闭开关来使能 SEND 指令，则每当梯形图扫描至该指令时都将会执
                            行发送数据功能；
```

梯形图：

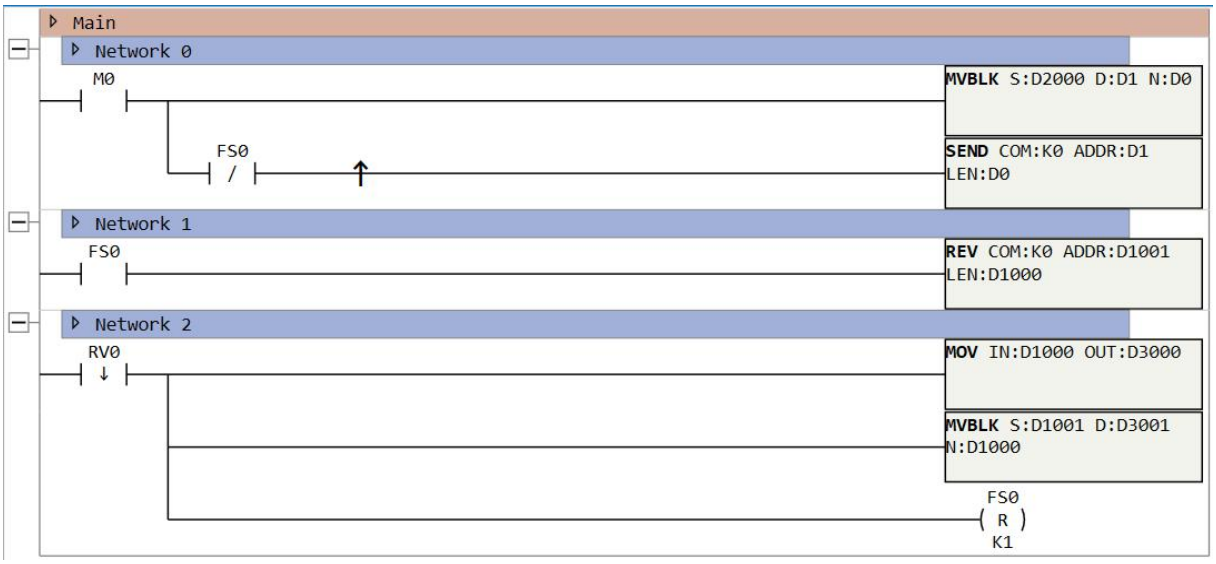


4. 实例 2

指令表：

```
Network 000
  LD     M0                    //首先在参数设置中配置好 COM0 口的相关参数，然后勾选配置信息下载到 PLC 中；
  MPS
  MVBLK  D2000  D1     D0      //将要传输的数据块通过 MVBLK 指令传送至起始地址为 D1，长度为 D0 的一片区域；
  MPP
  ANDN   M8180                 //数据发送完成后将失能 SEND 指令，关闭 COM0 口的数据发送；
  MEP
  SEND   K0     D1     D0      //使能 M0 后，将数据发送至 COM0 口；
Network 001
  LD     M8180                 //M8180 是 COM0 口的发送完成标志，当数据发送完成后该位将自动置 1，需在程序中手动对它进行复位，否则该位一直为 1；
  REV    K0     D1001  D1000   //发送完成后将使能 REV 接收指令，接收到的数据存放在以 D1001 为起始地址的一片区域，并将返回的接收长度存放在 D1000 中；
Network 002
  LDF    M8178                 //M8178 是 COM0 口的正在接收标志位，每当数据接收完成时将产生一个下降沿，该位是只读的，仅由程序控制，对其进行置/复位操作是无效的；
  MOV    D1000  D3000          //当 COM0 口的数据接收完成时，把返回的数据接收长度 D1000 传送至 D3000；
  MVBLK  D1001  D3001  D1000   //把接收到的以起始地址为 D1001 的一片数据块传送至以起始地址为 D3001 的一片区域，传送数据块长度即为本次接收到的数据长度 D1000；
  RST    M8180  K1             //数据接收完成后，清除 M8180 发送完成标志位；
```

梯形图：



4.14.3 REV

1. 指令说明

1、**COM_ID**（通信口选择）:可选择 COM0（232）或 COM1（485）；

2、**Addr**（起始地址）：将接收到的数据存放在以 Dn 为起始地址的一段区域，该操作数仅能选择 D 数据寄存器；

3、**Len**（接收长度）：该参数无需设置，在数据接收过程中会自动返回实际接收到的数据长度，并存放在所设的 D 寄存器中，对于不同型号 PLC 均有其相应的接收最大限制；

4、修改配置信息：选择 COM 口通信参数设置,如波特率、校验位等，并选择自由口通信模式，以及数据缓冲位数；

通讯数据缓冲位说明：

8 位：通讯时 D 寄存器的高 8 位字节是无效的，PLC 只利用 D 寄存器的低 8 位字节进行发送和接收；

16 位：通讯时在接收过程中，PLC 将接收到的数据先存放在 D 寄存器中的低 8 位字节，然后再存放在 D 寄存器中的高 8 位字节；

需注意的是，无论选择 8 位还是 16 位数据缓冲位，通讯时其返回的接收长度 Len 均为字节长度，当选择 8 位时，一个 D 寄存器就是一个字节长度，而当选择 16 位时，一个 D 寄存器是 2 个字节长度；

另外，每次在接收数据前以及重新使能接收 REV 指令后，都会自动清除上一次存放在 D 寄存器中的缓冲数据，因此 REV 指令常和 MVBLK 指令结合使用；

COMn 正在接收标志位(Only Read)：

寄存器	串口 COMn
M_8178	COM0（232）
M_8179	COM1（485）

COMn 接收完成标志位(Read or Write)：

寄存器	串口 COMn
M_8182	COM0（232）
M_8183	COM1（485）

注意：

1、要使用该指令的话必须将所设置的 COM 口（232/485）设置为自由口通讯模式，具体设置方式请参照 PLC 参数设置页面内容；

2、当设置某 COM 口模式为自由口通讯后，则该 COM 口不能用来进行程序下载和监视，也不能用作 Modbus 主站、从站；

3、在修改了 PLC 参数设置后，需勾选配置信息，然后下载到 PLC 中，否则参数修改无效；

4、由于受 PLC 数据缓存大小的限制，因此接收数据长度 Len 不能超过其最大限制，对于 16 或 32 点 PLC 最大为 256Byte，而 64 点 PLC 最大为 512Byte；

5、自由口通讯传输属于半双工模式，即发送指令 SEND 和接收指令 REV 不能同时进行；

2. 指令的有效操作数：

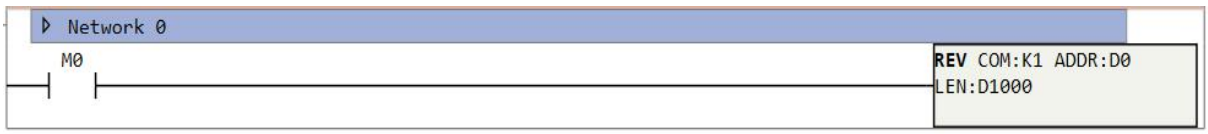
输入输出	类型	操作数	描述
EN	BOOL		启用输入端接通时，指令执行；否则，不执行
COM_ID			选择的通讯口
Addr	字	D	要接收的数据的存储起始地址
Len	字	D	要接收的数据长度

3. 实例 1

指令表：

```
Network 000
  LD    M000
  REV   K1    D0    D1000  //选择COM1（485），起始地址选为D0，当REV指令使能后，将会接收COM1（485）中的数据，假设收到0x10 0x11 0x12 0x13 0x14
                           //若数据缓冲位为8，则返回的接收长度Len为5，D0=16,D1=17,D3=18,D4=19,D5=20;
                           //若数据缓冲位为16，则返回的接收长度Len为5，D0=4368,D1=4882,D2=20;
```

梯形图：



4. 实例 2

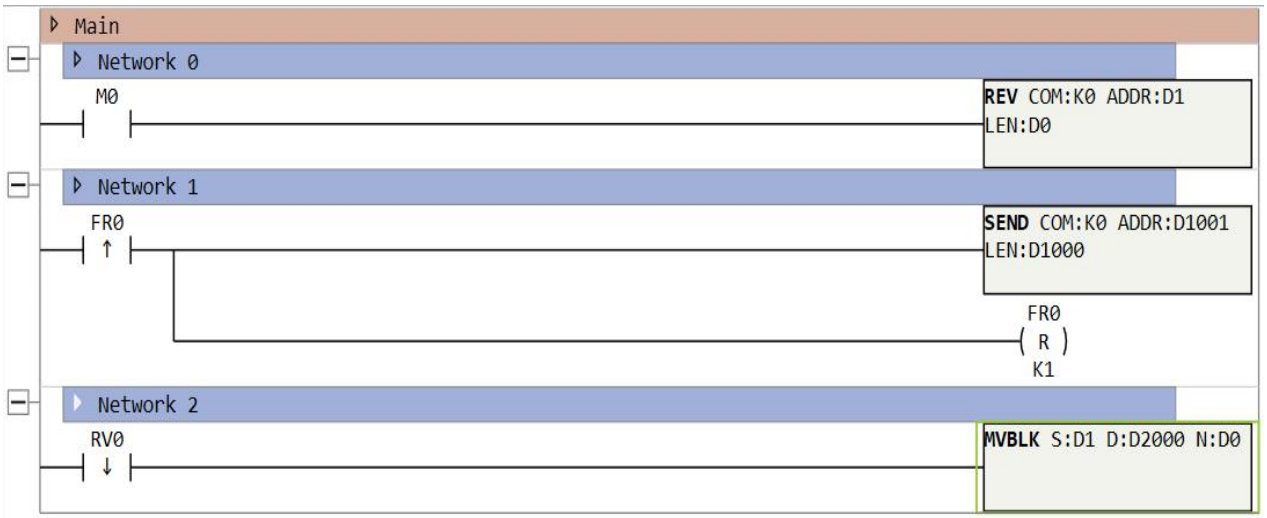
指令表：

```
Network 000
  LD    M0
  REV   K0    D1    D0        //配置好COM0口的相关参数后，使能COM0的REV接收指令，接收到的数据存放在以D1为起始地址的一片区域，并将返回的接收长度存放在D0
中；

Network 001
  LDP   M8182                 //M8182是COM0的接收完成标志位，当数据接收完成时该位将自动置1，需在程序中手动对它进行复位，否则该位一直为1；
  SEND  K0    D1001 D1000     //当COM0口接收到一次数据后将使能SEND发送指令，将长度为D1000且以D1001为起始地址的一片区域发送至COM0口；
  RST   M8182 K1              //清除M8182接收完成标志位；

Network 002
  LDF   M8178                 //M8178是COM0口的正在接收标志位，每当数据接收完成时将产生一个下降沿，该位是只读的，仅由程序控制，对其进行置/复位操作是无效的；
  MVBLK D1    D2000 D0        //将接收到的数据成块地存放在另一片区域，以方便对接收到的数据进行其他指令运算；
```

梯形图：



4.14.4 DESTIP

1. 指令说明

该指令用于建立 PLC 本地 IP 与远端的服务端 IP 间的连接，只有成功建立连接后，才能够进行以太网通信。目前最多能同时建立五个连接，也就是 PLC 作为以太网客户端，最多
能同时与五个以太网服务端建立连接。但是，该指令的远端 IP 参数是可以更改的。当已经建立五个连接时，还需要与新的远端 IP 建立连接，则可以断开指令使能，修改其中一个
的远端 IP 参数，重新接通使能，则 PLC 会与新设置的远端 IP 建立连接，原连接会断开。使用 MBUSTCP 指令与外部服务端进行以太网通信前，要先使用 DESTIP 指令与外部服务端建
立连接。

1) 参数首地址：设置远端 IP 和端口号参数的首地址，其参数格式参考实例；
2) 客户端编号：设置本地 IP 与远端 IP 所建立的连接的编号，编号范围 K0~K4；

2. 注意：

3) 1、PLC 与外部服务端设备建立连接，需要两者处于同一局域网内，即 IP 地址网络号相同；
4) 2、该指令接通使能后，再断开使能条件，之前所建立的连接并不会断开。修改 IP 参数，重新接通使能，才会断开原来的连接，建立新的连接；

3. 实例

如下图所示，D0 为指令参数首地址，其中远端的服务端 IP 参数为 192.168.1.100 端口号为 2000，将该参数分别赋值到 D0~D4，接通 M9，再接通 M10，建立连接，并将该连接编号为 0



4.14.5 MBUSTCP

1. 指令说明

该指令作为以太网客户端与外部的以太网服务端进行以太网 MODBUS TCP 通信。使用 MBUSTCP 指令前，一定要先使用 DESTIP 指令建立运动控制器与外部服务端的连接。具体请
参考 DESTIP 指令帮助文档。

① ID：客户端编号，可设置为 K0~K4，该编号代表使用 DESTIP 绑定的 IP 的连接编号，即选择以太网发送数据的远端 IP；

② Table：Modbus TCP 主站表格 ID，可在工程管理窗口中添加所需的 Modbus 主站表格；

③ WR：用来存储通信信息代码(见下表)；

④ WR_ID：用来存储当前主站表格的 Modbus 命令序列编号，表示当前所运行至该表格的哪一条命令序列，编号为 0~n-1，其中 n 为该表格内的命令序列总数；

⑤ 修改配置信息：进入 PLC 参数设置，选择通信参数选项卡，再选择“网口”选项卡，分别设置其中的通信参数和网口参数，通信参数中，要选择为 Modbus TCP/IP 主站模式；

注意：

1) 要使用该指令的话必须选择 Modbus TCP/IP 主站模式并勾选配置信息下载到 PLC 中，否则不会执行通信命令；
2) 该指令的第三个操作数 WR 虽然只需输入 Dn 来表示通信信息代码，但 Dn+1 也是有实际意义的，可用来返回当前主站表格的 Modbus 命令编号，而且 Dn+1 不可在其它地方修改数值；
3) 通信参数可设置超时时间，默认为 200ms，此外还可设置重传次数，默认为 3。比如：当设置超时时间为 200ms、重传次数为 3 时，表示若 200ms 内以太网服务端不返回, PLC 将重新发送一次数据包，若累计超过所设定的 3 次重传次数后从站仍不回应（此时 PLC 共发送了 4 次数据包），则返回通讯超时错误代码 16 即 0x10；
4) 超时时间在某些时候可能需要用户根据实际情况而略作调整，比如每次传输数据包字节数较多（每次读写从站长度较大，造成发送或接收该包数据耗时较长）时，也需要将超时时间增大，否则可能会报通讯超时错误；
5) PLC 运行中返回的通信信息代码存放在所设置的 Dn 寄存器中，该数值是一个十进制数，而下面表格中的均是十六进制数，如：通信返回 Dn 值=16，转换成十六进制=0x10，即表示通讯超时；
6) 使用 DESTIP 指令可以同时绑定 5 个服务端 IP，MBUSTCP 指令可以同时接通多个，相同客户端编号不同客户端编号均可，PLC 会按接通顺序依次发送数据，但在同一时刻不可能同时发送多组数据，PLC 与以太网服务端均是一个个顺序执行的，用户在编程时应考虑到这点，若其中一个服务端连接异常，会导致整个系统在该连接处因多次重传而卡顿一段时间，该卡顿时间即为：（重传次数+1）*超时时间，因此如若需要提高整个系统的快速响应可以考虑减少重传次数；
7) 帧间隔时间即为两个相邻以太网帧间隔的最短时间，正常通信时，帧间隔时间太长会降低通信效率，太短，则以太网通信将过于繁忙，可能造成通信异常，建议设置 2ms 以上。
8) 其表格创建方法，指令用法与 MBUS 一致。

通信信息代码：

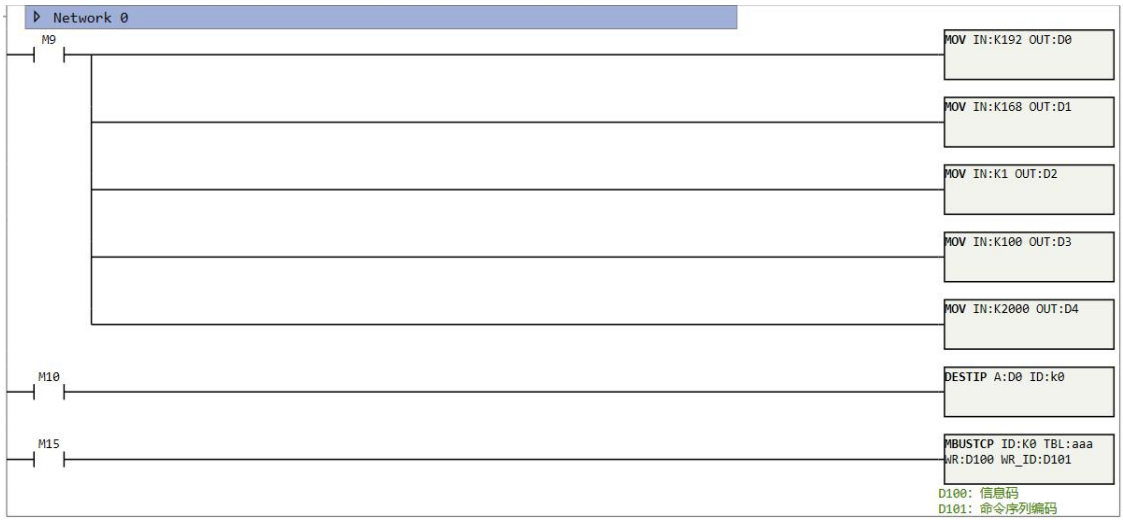
代码	信息
0x01	非法功能码
0x02	非法地址
0x03	非法数据
0x04	从站操作失败
0x05	命令有效，正在处理中
0x06	从机忙
0x07	非法数据类型
0x08	非法命令编号
0x09	非法密码
0x10	通讯超时
0x11	数值错误：数据小于0或者大于255
0x12	主从站设置错误
0x13	实际所接收数据中的站号与所期望的站号不一致
0x14	元件地址溢出：发送的数据量超出元件存储空间
0x15	命令执行失败
0x18	接收信息帧错误，包括接收长度校验错误、接收CRC校检错误
0x20	参数不可修改
0x21	参数运行时不可修改
0x22	参数受密码保护

2. **指令的有效操作数：**

输入输出	类型	操作数	描述
EN	BOOL		**启用输入端接通时，指令执行；否则，不执行**
COM_ID			**选择的通讯口**
Table			**Modbus 主站表标识**
WR	字	**D**	**用来存储通信信息代码**
WR_ID	字	**D**	**当前主站表格的 Modbus 命令序列编号**

3. 实例

先后导通 M9、M10，使用 DESTIP 指令与远端的服务端 IP 192.168.1.100 端口号 2000 建立连接，并编号该连接为 K0，使用 MBUSTCP 指令向 K0 连接发送数据.



4.14.6 CRC

1. 指令说明

1) Addr（起始地址）：将要计算 CRC 校验码的数据存放在以 Dn 为起始地址的一段区域，该操作数仅能选择 D 数据寄存器；
2) Len（数据长度）：以字节为单位，一个 D 寄存器占 2 个字节，低位在前高位在后。
3) Mode（CRC 校验模式）：可以选择不同类型的 CRC 校验。

名称	编号	多项式	校验码长度
CRC-16	K0	X16+X15+X2+1	2 字节
CRC-12	K1	X12+X11+X3+X2+X+1	2 字节
CRC-8	K2	X8+X2+X+1	1 字节
CRC-CCITT	K3	X16+X12+X5+1	2 字节
CRC-32	K4	X32+X26+X23+X22+X16+X12+X11+X10+X8+X7+X5+X4+X2+X+1	4 字节
CRC-32C	K5	X32+X28+X27+X26+X25+X23+X22+X20+X19+X18+X14+X13+X11+X10+X9+X8+X6+1	4 字节

计算的校验码会跟在数据的末尾。

例如，有一段数据为 12 34 56 78（十六进制），存放在以 D0 开始的寄存器中。那么 D0=0x3412，D1=0x78 长度为 4 个字节。

执行 CRC D0 K4 K0 后，CRC 校验码会保存到 D2 中，D2=0x107B。

注意：

- 数据长度为奇数时，校验码会紧跟在最后一个寄存器的高位。
- 注意传入的参数为原始数据的字节长度，最后通信的数据长度要加上后面的校验码长度。

2. 指令的有效操作数：

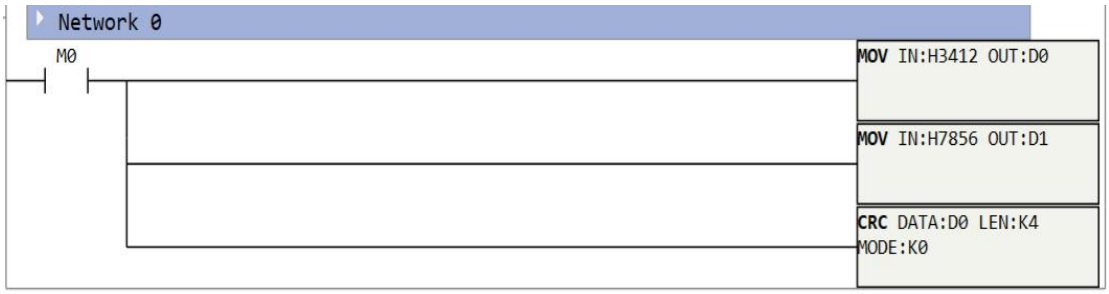
输入输出	类型	操作数	描述
EN	BOOL		启用输入端接通时，指令执行；否则，不执行
Addr	字	**D**	要校验数据的存储起始地址
Len	字	D/K	要校验的数据长度
Mode	字	K	CRC 校验模式

3. 实例

指令表：
Network 000

```
LD      M0
MOV     H3412 D0
MOV     H7856 D1
CRC     D0 K4 K0
```

梯形图：



4.14.7 CRCCHECK

1. 指令说明

1) Addr（起始地址）：将要计算 CRC 校验码的数据存放在以 Dn 为起始地址的一段区域，该操作数仅能选择 D 数据寄存器；
2) Len（数据长度）：以字节为单位，一个 D 寄存器占 2 个字节，低位在前高位在后。
3) Mode（CRC 校验模式）：可以选择不同类型的 CRC 校验。
4) Res（结果）：作为输出结果，校验通过输出 ON，不通过输出 OFF。

名称	编号	多项式	校验码长度
CRC-16	K0	X16+X15+X2+1	2 字节
CRC-12	K1	X12+X11+X3+X2+X+1	2 字节
CRC-8	K2	X8+X2+X+1	1 字节
CRC-CCITT	K3	X16+X12+X5+1	2 字节
CRC-32	K4	X32+X26+X23+X22+X16+X12+X11+X10+X8+X7+X5+X4+X2+X+1	4 字节
CRC-32C	K5	X32+X28+X27+X26+X25+X23+X22+X20+X19+X18+X14+X13+X11+X10+X9+X8+X6+1	4 字节

2. 指令的有效操作数：

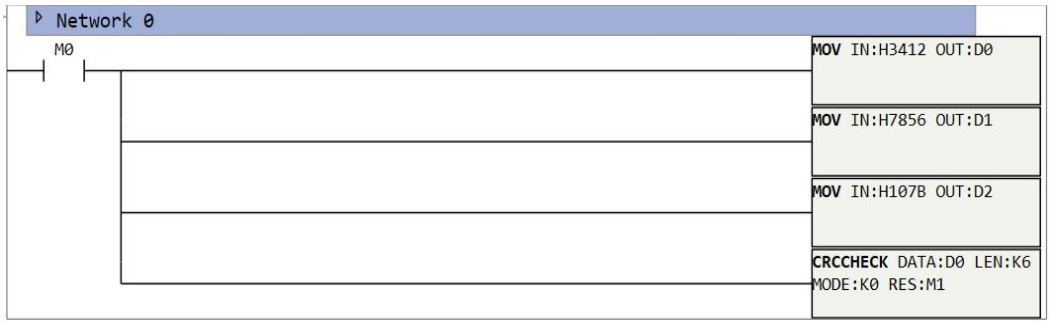
输入输出	类型	操作数	描述
EN	BOOL		**启用输入端接通时，指令执行；否则，不执行**
Addr	字	**D**	**要校验数据的存储起始地址**
Len	字	**D/K**	**要校验的数据长度**
Mode	字	**K**	**CRC 校验模式**
Res	位	**Y/M/S/C/T**	**输出结果**

3. 实例

指令表：
Network 000

```
LD          M0
MOV         H3412 D0
MOV         H7856 D1
MOV         H107B D2
CRCCHECK    D0 K6 K0 M1
```

梯形图：



4.15 脉冲输出指令

内容如下：

1、 可变频率脉冲输出 PLSF/DPLSF

2、 脉宽调制 PWM/DPWM

3、 脉宽调制（周期型）PWMS

4、 单段不带加减速脉冲输出 PLSY/DPLSY

5、 多段脉冲输出 PLSR/DPLSR/PLSRD/DPLSRD/PLSA/DPLSA

6、 脉冲段切换 PLSNEXT

7、 脉冲停止 PLSSTOP

8、 回归原点 ZRN/DZRN/DZRND

9、 相对位置多段脉冲控制 PTO（32 位）

10、 单段脉冲输出 DRVI/DDRVI/DRVA/DDRVA

11、 多段脉冲定位输出 TBL

12、 多段线插补 POLYLINEI/POLYLINEF

13、 直线插补 LINEI/LINEF

14、 圆弧插补 ARCI/ARCF

15、 图形导入指令 BLOCK

16、 HMI 绘图导入指令 HMIBLOCK

17、 带工艺 CAD 图形导入 HMIPLINE

18、 平面暂停 PAUSE

19、 任意点回归原点 ZRNR

20、 点动指令 JOG

21、 跟随指令 Follow

22、 平面系统设置

23、 电子凸轮 ECAM

24、 实时调频、实时调位指令 EDRVI/EDRVA

25、 脉冲密度测定 SPD

26、 增强型电子凸轮 TECAM

4.15.1 PLSF

可变频率脉冲输出 PLSF

对应命令	注释/说明	操作数（寄存器）
PLSF(32 位)	可变频率脉冲输出	输出频率(K/H/D) 输出脉冲 Y

指令说明

1）可变频率脉冲输出 PLSF 指令可用于可变脉冲频率的连续输出，频率跳变时无加减速时间。

2）启用输入端为“接通”时，立即以设定的脉冲频率开始连续的输出脉冲，过程脉冲频率可调，直至脉冲指令的启用输入端断开。PLSF 指令可输出频率范围为 0Hz~200KHz，FGM 系列 PLC 最高频率可达 500KHz；

3）FGs 提供了高速脉冲输出口，16/32 点晶体管型的 PLC，Y0~Y1 为高速脉冲输出口，64 点晶体管型的 PLC，Y0~Y3 为高速脉冲输出口；FGRB 晶体管 PLC，Y0~Y1 为高速脉冲输出口，FGRS 晶体管 PLC，Y0~Y3 为高速脉冲输出口，继电器型的 PLC 不具有高速脉冲功能，输出端口的最大输出频率由继电器的吸合时间决定。FGM 系列 PLC Y0~Y7 Y10~Y11 为高速脉冲输出口。

4）指令不含方向输出口，用户需要时方向信号输出端口号和信号控制可由用户自定义和设置，如实例中接线图，方向信号输出口 Y2，Y3 和 Z 相输入可由用户定义。

脉冲累积计数(读写)：

寄存器	脉冲输出 Yn
D_8140(D_8141)	Y000
D_8142(D_8143)	Y001
D_8144(D_8145)	Y002
D_8146(D_8147)	Y003
对于 FGM 系列 PLC 还具备以下寄存器	
D_8148(D_8149)	Y004
D_8150(D_8151)	Y005
D_8152(D_8153)	Y006
D_8154(D_8155)	Y007
D_8156(D_8157)	Y010
D_8158(D_8159)	Y011

32 位脉冲累积计数溢出标记，溢出为 1(只读)：

寄存器	脉冲输出 Yn
M_8118	Y000
M_8119	Y001
M_8120	Y002
M_8121	Y003
对于 FGM 系列 PLC 还具备以下 M 软元件	
M_8122	Y004
M_8123	Y005
M_8124	Y006
M_8125	Y007
M_8126	Y010
M_8127	Y011

注意：

1. PLSF 输出频率为 32 位正整数，占用两个连续的 D 寄存器；
2. 对于高速脉冲输出，它是针对外部高速器件的，要对脉冲进行计数，只能使用高速脉冲输入计数器，

不能使用内部计数器，也不能使用 Y 边沿变化进行计数。

3. 晶体管的 OFF 时间具有在轻负载时会延长的特性。所以请在需要响应性时设计成让负载在较轻时加负载电阻，增加负载电流；

4. 对于脉冲输出 Y0，寄存器 D8140（D8141）中累计其脉冲个数，D8141 存储高 16 位，D8140 存储低 16 位；

5. 脉冲累积计数寄存器（D8140~D8158）是重要寄存器，可读写，当写入新值时，计数会在新值基础上进行加减；

6. 由于本指令脉冲输出不带方向，所以脉冲累积计数寄存器（D8140~D8158）只能正方向计数；

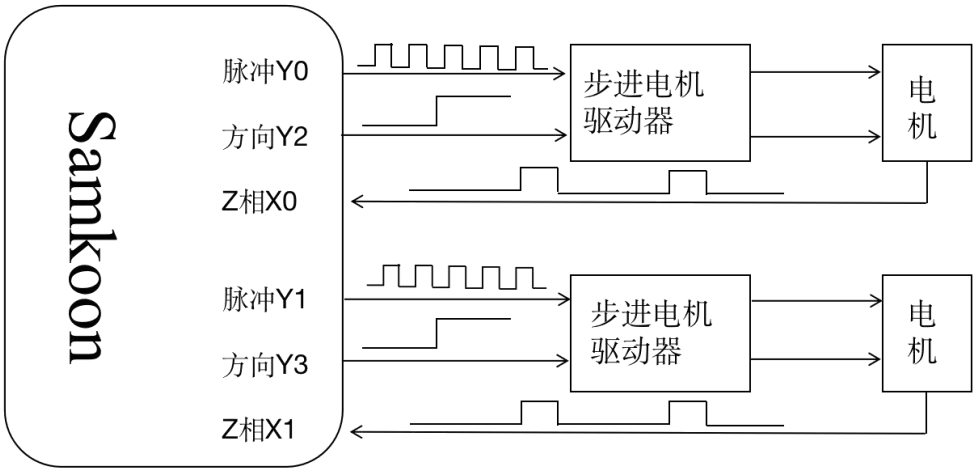
7. 高速脉冲输出时，脉冲累积计数寄存器（D8140~D8158）内的值是不连续的、时刻变化的，用于判断时，请使用大小比较，不要使用等于判断。

8. 当脉冲指令使用到输出口 Y 时，该 Y 口不能用做它用，即普通指令无法再对该 Y 口进行 ON、OFF 操作。

指令的有效操作数

输入输出	类型	操作数	描述
EN	BOOL		**启用输入端接通时，指令执行；否则，不执行**
输出频率（Hz）	双整数	**K/H/D**	**设定的脉冲频率**
输出脉冲 Y	双整数	**Y**	**指定的脉冲输出口**

接线图：



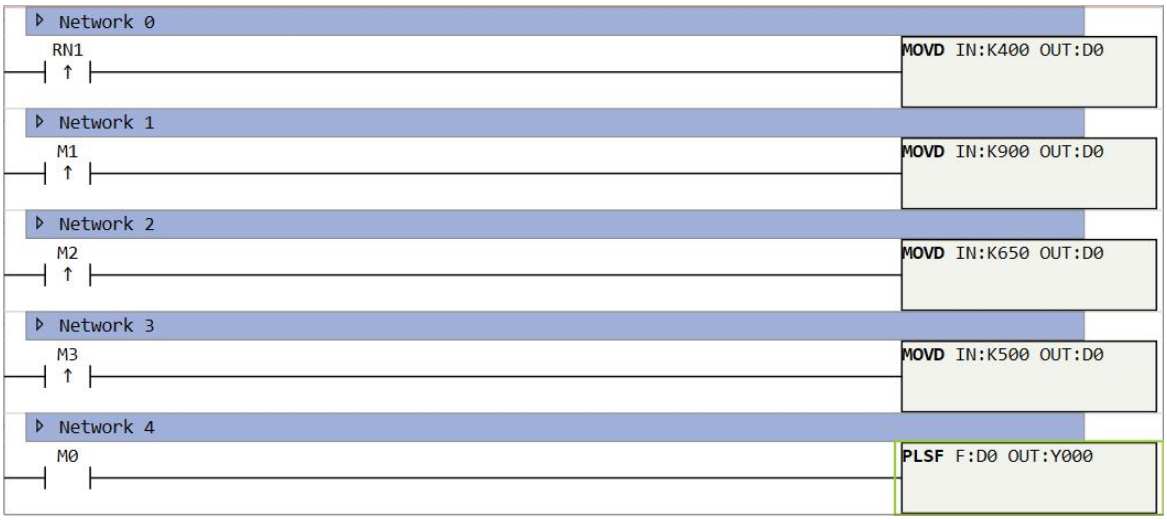
指令表：

```
Network 000
    LDP     M8151              // M8151 上升沿赋值，一定是上升沿，防止 PLC 扫描重复赋值
    MOVD    K400    D0         // 初始频率

Network 001
    LDP     M1
    MOVD    K900    D0         // 改变值 D0D1
```

```
Network 002
    LDP     M2
    MOVD    K650    D0          // 改变值 D0D1
Network 003
    LDP     M3
    MOVD    K500    D0          // 改变值 D0D1
Network 004
    LD      M0
    PLSF    D0      Y000    // 当 M0 为‘1’时，在 Y000 输出频率为 D0D1 的脉冲，否则脉冲输出停止，脉冲频率会随着 D0D1 的数值变化而变化。
```

梯形图：



4.15.2 PWM

脉宽调制 PWM

对应命令	注释/说明	操作数（寄存器）
PWM(32 位)	脉宽调制	输出频率(K/H/D) 输出占空比(K/H/D) 输出脉冲 Y

指令说明

1）脉宽调制 PWM 指令产生一个可变脉冲频率和可变占空比的脉冲输出，频率跳变时无加减速时间。

2）启用输入端为“接通”时，立即以设定的脉冲频率和占空比开始连续的输出脉冲，过程脉冲频率可调，占空比可调，直至脉冲指令的前提条件断开。32 位 PWM 指令可输出频率范围为 0Hz~200KHz，占空比范围 0~100%。对于 FGM 系列 PLC　最大脉冲频率为 500KHz，占空比调节范围为 0~100%。由于高频脉冲存在频率失真情况，所以进行脉宽的精准控制时，频率不要过高。

3）FGs 提供了高速脉冲输出口，16/32 点晶体管型的 PLC，Y0~Y1 为高速脉冲输出口，64 点晶体管型的 PLC，Y0~Y3 为高速脉冲输出口；FGRB 晶体管 PLC，Y0~Y1 为高速脉冲输出口，FGRS 晶体管 PLC，Y0~Y3 为高速脉冲输出口，继电器型的 PLC，不具有高速脉冲功能，输出端口的最大输出频率由继电器的吸合时间决定。FGM 系列 PLC 高速输出口为 Y0~Y7，Y10~Y11。

脉冲累积计数(读写)：

寄存器	脉冲输出 Yn
D_8140(D_8141)	Y000
D_8142(D_8143)	Y001
D_8144(D_8145)	Y002
D_8146(D_8147)	Y003
对于 FGM 系列 PLC 还具备以下寄存器	
D_8148(D_8149)	Y004
D_8150(D_8151)	Y005
D_8152(D_8153)	Y006
D_8154(D_8155)	Y007
D_8156(D_8157)	Y010
D_8158(D_8159)	Y011

32 位脉冲累积计数溢出标记，溢出为 1(只读)：

寄存器	脉冲输出 Yn
M_8118	Y000
M_8119	Y001
M_8120	Y002
M_8121	Y003
对于 FGM 系列 PLC 还具备以下 M 软元件	
M_8122	Y004
M_8123	Y005
M_8124	Y006
M_8125	Y007
M_8126	Y010
M_8127	Y011

注意：

1. PWM 输出频率与占空比为 32 位正整数，占用两个连续的 D 寄存器；
2. 对于高速脉冲输出，它是针对外部高速器件的，要对脉冲进行计数，只能使用高速脉冲输入计数器，不能使用内部计数器，也不能使用 Y 边沿变化进行计数；
3. 晶体管的 OFF 时间具有在轻负载时会延长的特性。所以请在需要响应性时设计成让负载在较轻时加负载电阻，增加负载电流；
4. 对于脉冲输出 Y0，寄存器 D8140（D8141）中累计其脉冲个数，D8141 存储高 16 位，D8140 存储低 16 位；
5. 脉冲累积计数寄存器（D8140~D8158）是重要寄存器，可读写，当写入新值时，计数会在新值基础上进行加减；
6. 由于本指令脉冲输出不带方向，所以脉冲累积计数寄存器（D8140~D8158）只能正方向计数；
7. 高速脉冲输出时，脉冲累积计数寄存器（D8140~D8158）内的值是不连续的、时刻变化的，用于判断时，请使用大小比较，不要使用等于判断；
8. 频率过高时存在波形失真情况，若要进行脉宽精确控制，脉冲频率不要过高。
9. 当脉冲指令使用到输出口 Y 时，该 Y 口不能用做它用，即普通指令无法再对该 Y 口进行 ON、OFF 操作。

指令的有效操作数

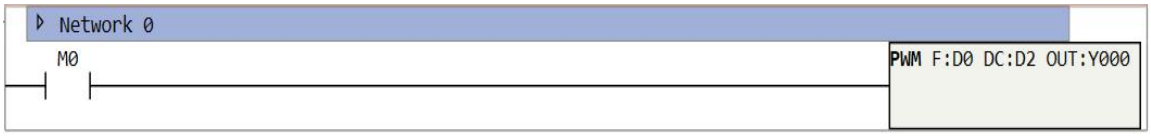
输入输出	类型	操作数	描述
EN	BOOL		**启用输入端接通时，指令执行；否则，不执行**
输出频率（Hz）	整数/双整数	**K/H/D**	**设定的脉冲频率**
占空比(%)	整数/双整数	**K/H/D**	**设定的占空比**
输出脉冲 Y	BOOL	**Y**	**指定的脉冲输出口**

实例

指令表：

```
Network 000
    LD      M0
    PWM     D0   D2   Y000   // 当 M0 为‘1’时，在 Y000 输出频率为 D0D1，占空比为百分之 D2D3 的脉冲
```

梯形图：



4.15.3 PWMS

1.指令说明

1）该指令不支持 FGRS_C8X8T、FGRE_C8X8T、FGs_64MT_A、FGs_64MR_A 及 FGm_64MT_A 机型。

2）脉宽调制（周期型）PWMS 指令产生一个可变周期和可变脉宽时间的脉冲输出。

3）启用输入端为“接通”时，立即以设定的周期和脉宽时间开始连续地输出脉冲，输出过程中周期、脉宽时间可修改并即时生效，直至脉冲指令的前提条件断开。

4）PWMS 指令可选择时间单位为 ms 或 us.当时间单位为 ms 时，周期的设置范围为 2-65535ms，脉宽时间范围为 0-65535ms；当时间单位为 us 时，周期的设置范围为 5-65535us，

脉宽时间范围为 0-65535us。由于高频脉冲存在波形失真情况，所以在时间单位为 us 时，PLC 对脉宽进行了自动限制：若脉宽时间=1us（或周期减去脉宽的时间=1us）时，强制将

脉宽时间（或周期减去脉宽的时间）改为 2us，尽可能保证高低电平的波形。

5）PWMS 指令提供改变电平输出顺序的功能，可以人为地设置在一个脉冲周期内，先输出有效电平还是无效电平。当 M8062 为 OFF 时，无效电平在前，有效电平在后；当 M8062

为 ON 时，有效电平在前，无效电平在后。注意：需要在使能 PWMS 指令之前修改 M8062 设置电平输出顺序。若在 PWMS 输出期间修改 M8062 状态，电平输出顺序不会发生变化。

6）FGs 提供了高速脉冲输出口，16/32 点晶体管型的 PLC，Y0~Y1 为高速脉冲输出口，64 点晶体管型的 PLC，Y0~Y3 为高速脉冲输出口；FGRB 晶体管 PLC，Y0~Y1 为高速脉冲输出

口，FGRS 晶体管 PLC，Y0~Y3 为高速脉冲输出口，继电器型的 PLC，不具有高速脉冲功能，输出端口的最大输出频率由继电器的吸合时间决定。FGM 系列 PLC 高速输出口为 Y0~Y7，

Y10~Y11。

注意：

1. PWMS 周期与脉宽时间为 16 位正整数；

2. 对于高速脉冲输出，它是针对外部高速器件的，要对脉冲进行计数，只能使用高速脉冲输入计数器，不能使用内部计数器，也不能使用 Y 边沿变化进行计数；

3. 晶体管的 OFF 时间具有在轻负载时会延长的特性。所以请在需要响应性时设计成让负载在较轻时加负载电阻，增加负载电流；

4. PWMS 指令无脉冲累积计数功能，输出脉冲时 D8140、D8142 等脉冲计数寄存器的值不会增加；

5. 频率过高时存在波形失真情况，若要进行脉宽精确控制，脉冲频率不要过高。

6. 当脉冲指令使用到输出口 Y 时，该 Y 口不能用做它用，即普通指令无法再对该 Y 口进行 ON、OFF 操作。

7. 如果多脉冲脉冲频率超过 200K，需要在脉冲输出口加入上拉电阻保证脉冲波形不失真，上拉电压 24V,上拉电阻推荐 1K。

2. **指令的有效操作数**

输入输出	类型	操作数	描述
EN	BOOL		**启用输入端接通时，指令执行；否则，不执行**
时间单位	常数	**K/H**	**后续参数的时间单位，只能填入常数。K0：单位为 ms；K1：单位为 US。**
周期	16 位正整数	**K/H/D**	**设定的周期**
脉宽时间	16 位正整数	**K/H/D**	**设定的脉宽时间，不能超过周期的时长。**
输出脉冲 Y	BOOL	**Y**	**指定的脉冲输出口**

实例

指令表：

```
Network 000
LD      M0
PWMS K0 D0 D1 Y000 //  当 M0 为‘1’时，在 Y000 输出周期为 D0(ms)，脉宽时间为 D1(ms)的脉冲。
```

梯形图：



4.15.4 PLSY

单段不带加减速脉冲输出 PLSY

对应命令	注释/说明	操作数（寄存器）
PLSY(32 位)	单段不带加减速脉冲输出	输出频率(K/H/D) 输出脉冲数(K/H/D) 输出脉冲 Y

1.指令说明

1）单段不带加减速脉冲输出 PLSY，指令可用于输出一定频率的指定脉冲数的脉冲。当脉冲正在输出时，相应的脉冲正在输出标志被置位；当输出脉冲数达到设定值时，脉冲输出停止，相应的脉冲正在输出标志被复位。在脉冲输出过程中，脉冲和频率可调，频率跳变时无加减速时间。

2）启用输入端为“接通”时，立即以设定的频率开始连续的输出脉冲，当输出脉冲个数达到设定的输出脉

冲数时，脉冲输出停止，过程脉冲数和频率可调。32 位 PLSY 指令可输出频率范围为 0Hz~200KHz。对于 FGM 系列 PLC 最大脉冲输出频率为 500KHz。

3）FGs 提供了高速脉冲输出口，16/32 点晶体管型的 PLC，Y000~Y001 为高速脉冲输出口，64 点晶体管型的 PLC，Y000~Y003 为高速脉冲输出口；FGRB 晶体管 PLC，Y0~Y1 为高速脉冲输出口，FGRS 晶体管 PLC，Y0~Y3 为高速脉冲输出口，继电器型的 PLC，不具有高速脉冲功能，输出端口的最大输出频率由继电器的吸合时间决定。多脉冲型 PLC 的高速输出口为 Y000~Y007，Y10~Y11。

4）指令不含方向输出口，方向信号输出端口号和信号控制可由用户自定义和设置。

脉冲正在输出标志(只读)：

寄存器	脉冲输出 Yn
M_8134	Y000
M_8135	Y001
M_8136	Y002
M_8137	Y003
对于 FGM 系列 PLC 还具备以下 M 软元件	
M_8138	Y004
M_8139	Y005
M_8140	Y006
M_8141	Y007
M_8142	Y010
M_8143	Y011

脉冲累积计数(读写)：

寄存器	脉冲输出 Yn
D_8140(D_8141)	Y000
D_8142(D_8143)	Y001
D_8144(D_8145)	Y002
D_8146(D_8147)	Y003
对于 FGM 系列 PLC 还具备以下寄存器	
D_8148(D_8149)	Y004
D_8150(D_8151)	Y005
D_8152(D_8153)	Y006
D_8154(D_8155)	Y007
D_8156(D_8157)	Y010
D_8158(D_8159)	Y011

32 位脉冲累积计数溢出标记，溢出为 1(只读)：

寄存器	脉冲输出 Yn
M_8118	Y000
M_8119	Y001
M_8120	Y002
M_8121	Y003
对于 FGM 系列 PLC 还具备以下 M 软元件	
M_8122	Y004
M_8123	Y005
M_8124	Y006
M_8125	Y007
M_8126	Y010
M_8127	Y011

注意：

1. PLSY 输出频率与输出脉冲数为 32 位正整数，占用两个连续的 D 寄存器；
2. 对于高速脉冲输出，它是针对外部高速器件的，要对脉冲进行计数，只能使用高速脉冲输入计数器，不能使用内部计数器，也不能使用 Y 边沿变化进行计数；
3. 晶体管的 OFF 时间具有在轻负载时会延长的特性。所以请在需要响应性时设计成让负载在较轻时加

负载电阻，增加负载电流；

4. 对于脉冲输出 Y0，寄存器 D8140（D8141）中累计其脉冲个数，D8141 存储高 16 位，D8140 存储低 16 位；

5. 脉冲累积计数寄存器（D8140~D8158）是重要寄存器，可读写，当写入新值时，计数会在新值基础上进行加减；

6. 由于本指令脉冲输出不带方向，所以脉冲累积计数寄存器（D8140~D8158）只能正方向计数；

7. 高速脉冲输出时，脉冲累积计数寄存器（D8140~D8158）内的值是不连续的、时刻变化的，用于判断时，请使用大小比较，不要使用等于判断。

8. 当脉冲指令使用到输出口 Y 时，该 Y 口不能用做它用，即普通指令无法再对该 Y 口进行 ON、OFF 操作。

2.指令的有效操作数

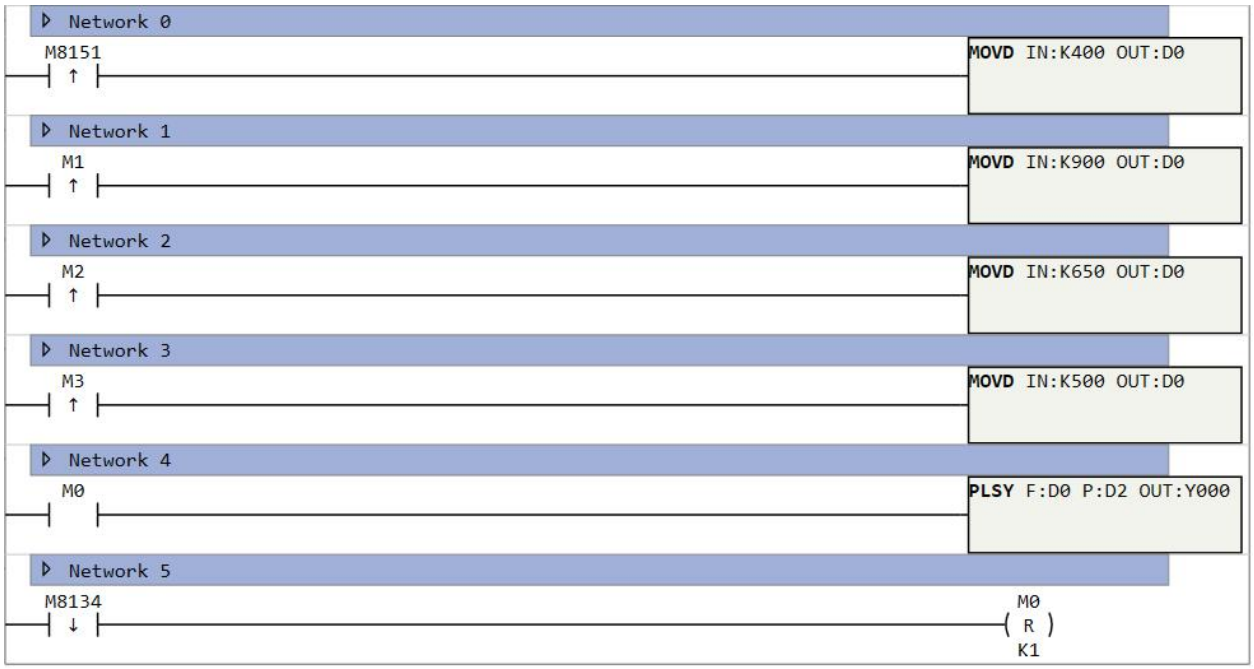
输入输出	类型	操作数	描述
EN	BOOL		**启用输入端接通时，指令执行；否则，不执行**
输出频率（Hz）	双整数	**K/H/D**	**设定的脉冲频率**
输出脉冲数	双整数	**K/H/D**	**设定的输出脉冲数**
输出脉冲 Y	BOOL	**Y**	**指定的脉冲输出口**

3.实例

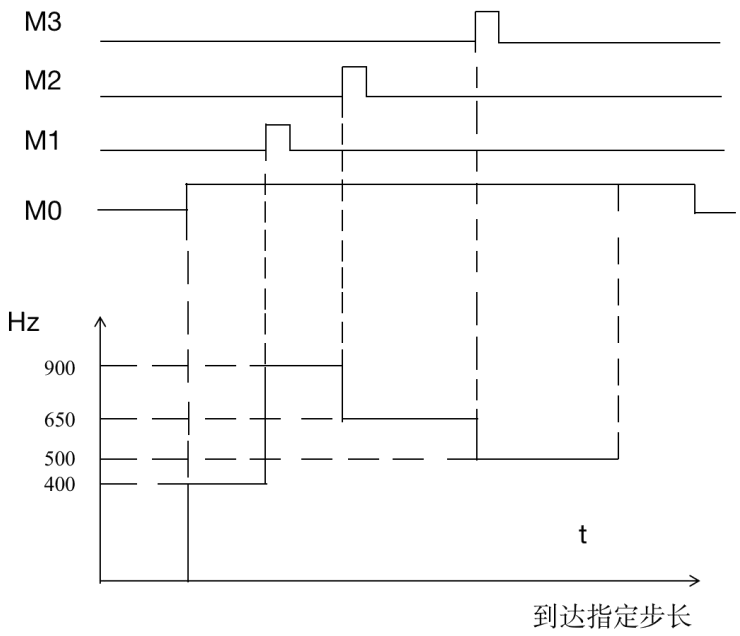
指令表：

```
Network 000
    LDP     M8151          // M8151 上升沿赋值，一定是上升沿，防止 PLC 扫描重复赋值
    MOVD    K400    D0     // 初始频率
Network 001
    LDP     M1
    MOVD    K900    D0     // 改变值 D0
Network 002
    LDP     M2
    MOVD    K650    D0     // 改变值 D0
Network 003
    LDP     M3
    MOVD    K500    D0     // 改变值 D0
Network 004
    LD      M0
    PLSY    D0      D2     Y000   // 当 M0 为‘1’时，在 Y000 端口输出频率为 D0D1 的脉冲，当脉冲数达到 D2D3 个时，输出脉冲停止，并将脉冲正在输出标志 M_8134 复位
Network 005
    LDF     M8134
    RST     M0      K1
```

梯形图：


Network 0
M8151
MOVD IN:K400 OUT:D0
Network 1
M1
MOVD IN:K900 OUT:D0
Network 2
M2
MOVD IN:K650 OUT:D0
Network 3
M3
MOVD IN:K500 OUT:D0
Network 4
M0
PLSY F:D0 P:D2 OUT:Y000
Network 5
M8134
M0
R
K1

时序图：


M3
M2
M1
M0
Hz
900
650
500
400
t
到达指定步长

对应命令	注释/说明	操作数（寄存器）
PLSR(32 位)	单向相对位置多段脉冲输出	指定分段脉冲参数起始软元件地址编号(D) 指定加减速的数值(K/H)或者软元件的地址编号(D) 指定脉冲输出端口编号(Y) 指定脉冲输出方向端口编号(Y)
PLSA(32 位)	绝对位置多段脉冲输出	指定分段脉冲参数起始软元件地址编号(D) 指定加减速的数值(K/H)或者软元件的地址编号(D) 指定脉冲输出端口编号(Y) 指定脉冲输出方向端口编号(Y)
EPLSR(32 位)	双向相对位置多段脉冲输出	指定分段脉冲参数起始软元件地址编号(D) 指定脉冲输出端口编号(Y) 指定脉冲输出方向端口编号(Y)

4.15.5 PLSR/PLSA/EPLSR

指令说明

1）多段脉冲输出 PLSR/PLSA/EPLSR 指令可用于实现多段按预设的频率和脉冲数的脉冲输出，不同段间的输出转换要在预设的加减速时间内完成。启用输入端为“接通”时，立即从第一段的起始频率开始输出脉冲，在预定的加减速时间内加速到第一段预设的频率，当输出脉冲个数达到第一段预设的输出脉冲数时；在预定的加减速时间内加减速到第二段预设的频率，当输出脉冲个数达到第二段预设的输出脉冲数时，按第三段预设值执行，直到段脉冲的个数设定为 0 或频率设置为 0 表示结束分段，脉冲输出停止。

2）多段频率和脉冲数的设定值在脉冲参数地址中设定。以 Dn 为起始地址的一段区域。

32 位指令 PLSR/PLSA：以 D0（双字）为起始地址为例，D0（双字）设定第 1 段脉冲的最高频率、D2（双字）设定第 1 段脉冲的个数(位置)，D4（双字）设定第 2 段脉冲的最高频率、D6（双字）设定第 2 段脉冲的个数（位置），……，以 Dn 设定第(n+4)/4 段脉冲的最高频率、Dn+2 设定第(n+4)/4 段脉冲的个数的设定值为 0 表示分段结束。当某段的脉冲个数设置为-1 时，则表示该段脉冲将一直发送，直到遇到 PLSNEXT 指令进行段落切换。

32 位指令 EPLSR:以 D0（双字）为起始地址为例，D0（双字）设定第 1 段加速时间、D2（双字）设定第 1 段脉冲的最高频率、D4（双字）设定第 1 段脉冲的个数，D6（双字）设定第 1 段加速时间，……，以 Dn 设定第(n+6)/6 段加速时间，以 Dn+2 设定第(n+6)/6 的最高频率，以 Dn+4 设定第(n+6)/6 段的脉冲个数，以 Dn+6 设定第(n+6)/6 段的减速时间，以 Dn+10 设定第(n+6)/6 段的脉冲频率与 Dn+12 设定第(n+6)/6 段的脉冲个数为 0 表示分段结束。当某段的脉冲个数设置为-1 时，则表示该段脉冲将一直发送，直到遇到 PLSNEXT 指令进行段落切换。

3）32 位 PLSR/PLSA/EPLSR 指令可输出频率范围为 0Hz~200KHz，脉冲量范围 K0~K2147483647，对于 FGM 系列 PLC 最大脉冲频率为 500KHz。PLSR/PLSA 各个段落的加减速时间相同，EPLSR 每一段的加减速时间可以单独设置，最后一段可以分别设置加速时间和减速时间。

每段设定脉冲频率，脉冲个数和加减速时间满足关系式：（起始频率 Hz + 结束频率 Hz）×（ 加减速时间 ms÷1000） <= 脉冲个数 × 2。

对于用户设置不合法或者不合理的参数，当是否忽略错误标志 M8070-M8073 被置位时，不忽略错误，程序将会报错，ERR 红灯变亮，出错脉冲指令不执行，其它程序会继续执行。 用户可以读取 D8176 中的错误编号查看错误原因，错误码如下表，出错脉冲段位置可由 D8108-D8111 查看。修改完善后再次下载运行程序，错误状态会取消，并正确执行程序。当 M8070-M8073 被复位时，忽略错误，ERR 灯不会亮，但错误依然会检测到，对应寄存器会显示错误状态，但脉冲会以默认方式调整脉冲后强制发送，此时脉冲频率可能存在跳变情况，但脉冲数量依然准确。

错误码（D8176）	错误码解释	错误原因及修改方式
23	多段脉冲指令的段落数超出最大范围（可能无分段完成标志）	PLSR 指令的频率和脉冲数量参数同时为零代表指令结束，如果报此错可能是未设置结束标志，或设置段数超过默认最大段数。
24	多段脉冲指令第一段最大脉冲频率小于默认起始频率	第一段的默认起始频率是 1000HZ，所以第一段最大频率请不要低于此值。
25	频率小于下限	频率设置得不要太低，如果要使用低频，请选择非加减速脉冲指令。
26	单段脉冲数小于下限	各段落的脉冲数不要设置得太低，否则无法完成加减速。
27	加减速时间小于下限	加减速时间设置太小，频率分段跳变跨度很大，不易起步。
28	参数不满足加减速合法要求	各参数不满足合法关系式，请重新调整参数使其满足上式。
29	最后一段不满足加减速合法要求，请兼顾加速和减速停的脉冲数量	指令最后一段会默认从结束频率减速为零，无需用户设置(除 PTO 指令)，请考虑减速的脉冲数量， 满足：（起始频率 Hz + 结束频率 Hz）×（ 加减速时间 ms÷1000） <= 脉冲个数

4）单向相对位置多段脉冲输出 PLSR 脉冲方向 Y 只由第一段设定的脉冲个数或者脉冲频率的正负决定，当第一段设定的脉冲个数或者频率为正数时，为正方向，Y 输出为 ON；当设定的脉冲个数或者频率为负数时，为负方向，Y 输出为 OFF。对于绝对定位指令 PLSA，脉冲个数代表其绝对位置，各段的正负可不相同，会自动变换方向。

对于双向相对位置多段脉冲输出 EPLSR 脉冲方向由每一段的脉冲数的正负来决定方向，当本段相对上一段的脉冲数的正负发生改变时，上一段会减速到 0，本段会从 0 开始反向加速到本段的频率，根据每一段脉冲数的正负可以自动变换方向，并且对于最后一段可以单独设置加速时间和减速时间。

5）特殊功能寄存器 M8068 为方向极性控制，默认脉冲方向输出口为 ON 时表示正方向，OFF 时表示负方向。若设置 M8068 为 ON 则所有脉冲指令的方向输出极性发生反转，脉冲方向输出为 OFF 表示正方向，为 ON 表示负方向。

6）FGs 提供了高速脉冲输出口，16/32 点晶体管型的 PLC，Y000~Y001 为高速脉冲输出口，64 点晶体管型的 PLC，Y000~Y003 为高速脉冲输出口；FGRB 晶体管 PLC，Y0~Y1 为高速脉冲输出口，FGRS 晶体管 PLC，

Y0~Y3 为高速脉冲输出口，继电器型的 PLC 不具有高速脉冲功能，输出端口的最大输出频率由继电器的吸合时间决定。对于 FGM 系列 PLC 其高速脉冲输出口为 Y0~Y7，Y10~Y11。

7）当特殊功能寄存器 M8069 为 ON 时所有的脉冲指令的加减速为 S 形，为 OFF 时则为梯形。

8）在多段脉冲发送过程中，当脉冲还未发送到该段时，可以任意更改该段的参数，当运行到此段时会以新参数运行。但当脉冲发送到该段脉冲的上一段时，不要将该段的参数改为 0，否则会造成末段的频率突变。同理，可将原本为结束标志的 D 寄存器修改为有参数的脉冲段实现追加脉冲段功能；

脉冲正在输出标志(只读)：

寄存器	脉冲输出 Yn
M_8134	Y000
M_8135	Y001
M_8136	Y002
M_8137	Y003
对于 FGM 系列 PLC 还具备以下 M 软元件	
M_8138	Y004
M_8139	Y005
M_8140	Y006
M_8141	Y007
M_8142	Y010
M_8143	Y011

脉冲方向，正方向为 1(只读)：

寄存器	脉冲输出 Yn
M_8102	Y000
M_8103	Y001
M_8104	Y002
M_8105	Y003
对于 FGM 系列 PLC 还具备以下 M 软元件	
M_8106	Y004
M_8107	Y005
M_8108	Y006
M_8109	Y007
M_8110	Y010
M_8111	Y011

32 位脉冲累积计数溢出标记，溢出为 1(只读)：

寄存器	脉冲输出 Yn
M_8118	Y000
M_8119	Y001
M_8120	Y002
M_8121	Y003
对于 FGM 系列 PLC 还具备以下 M 软元件	
M_8122	Y004
M_8123	Y005
M_8124	Y006
M_8125	Y007
M_8126	Y010
M_8127	Y011

脉冲累积计数(读写)：

寄存器	脉冲输出 Yn
D_8140(D_8141)	Y000
D_8142(D_8143)	Y001
D_8144(D_8145)	Y002
D_8146(D_8147)	Y003
对于 FGM 系列 PLC 还具备以下 M 软元件	
D_8148(D_8149)	Y004
D_8150(D_8151)	Y005
D_8152(D_8153)	Y006
D_8154(D_8155)	Y007
D_8156(D_8157)	Y010
D_8158(D_8159)	Y011

是否忽略错误，继续发送脉冲，默认忽略为 0，若设为 1，出错时停止脉冲输出(读写)：

寄存器	脉冲输出 Yn
M_8070	Y000
M_8071	Y001
M_8072	Y002
M_8073	Y003
对于 FGM 系列 PLC 还具备以下 M 软元件	
M_8074	Y004
M_8075	Y005
M_8076	Y006
M_8077	Y007
M_8078	Y010
M_8079	Y011

检查每段用户设置脉冲个数和频率是否正确 1--发现错误（用于多段脉冲）(只读)：

寄存器	脉冲输出 Yn
M_8086	Y000
M_8087	Y001
M_8088	Y002
M_8089	Y003
对于 FGM 系列 PLC 还具备以下 M 软元件	
M_8090	Y004
M_8091	Y005
M_8092	Y006
M_8093	Y007
M_8094	Y010
M_8095	Y011

当前脉冲段(只读)：

寄存器	脉冲输出 Yn
D_8124	Y000
D_8125	Y001
D_8126	Y002
D_8127	Y003
对于 FGM 系列 PLC 还具备以下 M 软元件	
D_8128	Y004
D_8129	Y005
D_8130	Y006
D_8131	Y007
D_8132	Y010
D_8133	Y011

出错脉冲段位置，0 表示无错误(只读)：

寄存器	脉冲输出 Yn
D_8108	Y000
D_8109	Y001
D_8110	Y002
D_8111	Y003
对于 FGM 系列 PLC 还具备以下 M 软元件	
D_8112	Y004
D_8113	Y005
D_8114	Y006
D_8115	Y007
D_8116	Y010
D_8117	Y011

注意：

1. PLSR/PLSA/EPLSR 每个输出频率为 32 位正整数，占用两个连续的 D 寄存器；

2. 对于高速脉冲输出，它是针对外部高速器件的，要对脉冲进行计数，只能使用高速脉冲输入计数器，不能使用内部计数器，也不能使用 Y 边沿变化进行计数；

3. 晶体管的 OFF 时间具有在轻负载时会延长的特性。所以请在需要响应性时设计成让负载在较轻时加负载电阻，增加负载电流；

4. 对于脉冲输出 Y0，寄存器 D8140（D8141）中累计其脉冲个数，D8141 存储高 16 位，D8140 存储低 16 位；

5. 脉冲累积计数寄存器（D8140~D8158）是重要寄存器，可读写，当写入新值时，计数会在新值基础上进行加减；

6. 由于本指令脉冲输出带方向，所以脉冲累积计数寄存器（D8140~D8158）根据方向计数；

7. 高速脉冲输出时，脉冲累积计数寄存器（D8140~D8158）内的值是不连续的、时刻变化的，用于判断时，请使用大小比较，不要使用等于判断；

8. 当为 n 段脉冲时，第 1 段、第 2 段、第 3 段……第 n 段每段的脉冲频率和脉冲数的地址必须是依次连续的，且第 n+1 段的脉冲频率与脉冲数的地址必须为 0，用来判断脉冲段是否结束；加减速时间的地址不能够紧跟在第 n 段后面；

9. 当脉冲个数为 K-1 或者 HFFFFFFFF(32 位)时，表示该段脉冲会一直输出，直到等到 PLSNEXT 指令才会切换分段；

10. 第一段的起始频率系统默认为 1000HZ，做高速脉冲输出时，第一段的稳定频率不要低于该值；

11. 当前脉冲段（D8124-D8133）只在脉冲发送过程中记录当前脉冲段，指令失能时自动归零。

12. 当脉冲指令使用到输出口 Y 时，该 Y 口不能用做它用，即普通指令无法再对该 Y 口进行 ON、OFF 操作。

指令的有效操作数

输入输出	类型	操作数	描述
EN	BOOL		**启用输入端接通时，指令执行；否则，不执行**
脉冲参数地址	整数/双整数	**D**	**多段频率设定值的起始地址**
加减速时间（ms）	整数/双整数	**K/D**	**多段脉冲设定值的起始地址**
输出脉冲 Y	BOOL	**Y**	**指定的脉冲输出口**
方向 Y	BOOL	**Y**	**PLSRD/DPLSRD 指令的脉冲方向**

实例

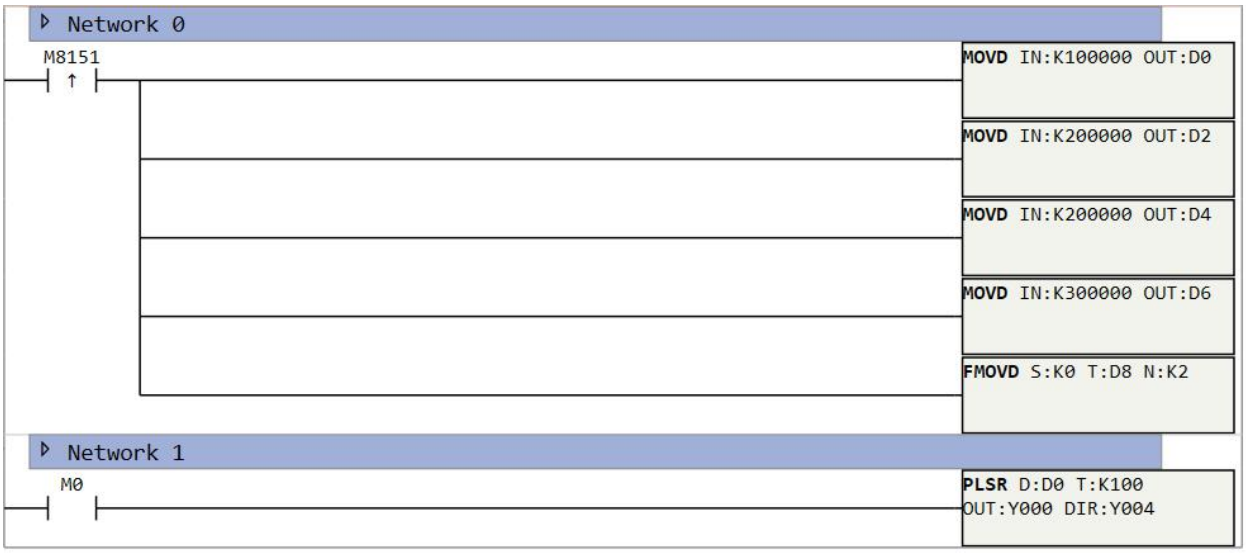
1）使用 MOVD 赋值实现相对位置多段脉冲输

指令表：

```
Network 000
    LDP     M8151
    MOVD    K100000    D0        //第一段频率 100KHz  存入 D0D1
    MOVD    K200000    D2        //第一段脉冲数 2000000  存入 D2D3
    MOVD    K200000    D4        //第二段频率 200KHz  存入 D4D5
```

```
    MOVD      K300000      D6              //第二段脉冲数 300000  存入 D6D7
    FMOVD     K0     D8     K2             //结束标志  频率 0 存入 D8D9，脉冲数 0 存入 D10D11
Network 001
    LD        M0
    PLSR      D0        K100      Y000     Y004// 当 M0 被置位，PLSR 指令开始工作
```

梯形图：



2）使用函数功能块实现相对位置多段脉冲输出

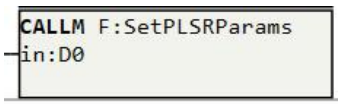
参数设置：

在工程管理窗口“工程”-->“指令”-->“脉冲输出”-->“PLSR”中单击右键，单击“创建参数函数”，会弹出对话框。



图 1.1

输入要新建的函数名称，选择要添加到的函数功能块（也可以选择新建一个函数功能块并输入新的名称，**建议统一放到名叫“脉冲参数设置”的函数功能块中**）。点击确定，就会在梯形图光标处插入了一条新的 CALLM 指令。双击该指令的图元，弹出的参数设置对话窗口和普通的不一样。



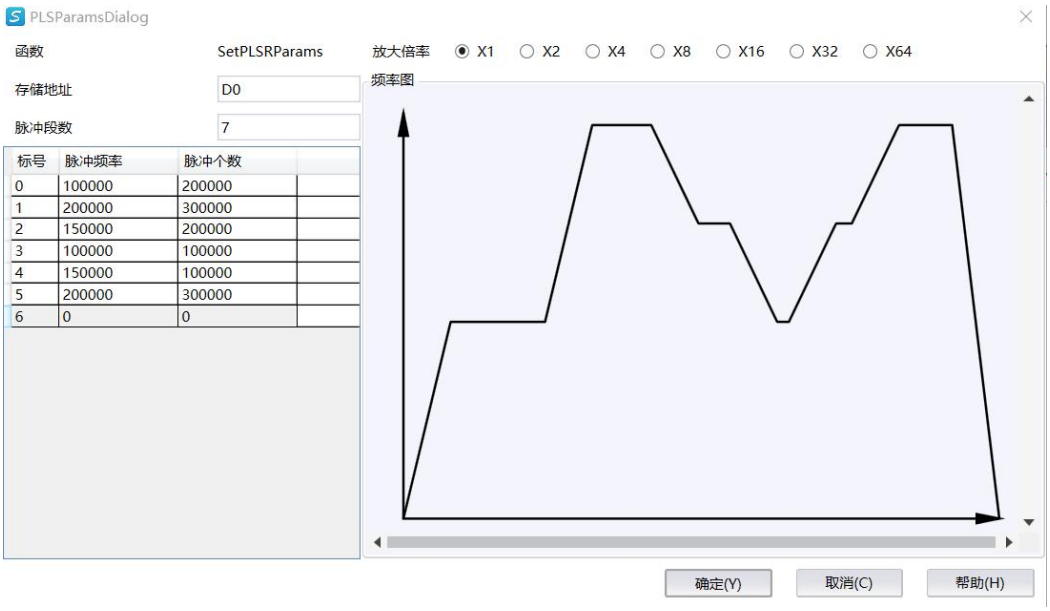


图 1.2

这里可以设置分段脉冲的参数，最后一段脉冲设为(0,0)表示结尾。**请注意，右侧的频率图仅供参考，不代表实际运行时的具体情况。**

点击确定后，除了加减速时间以外，分段脉冲的参数设置就完毕了。可以在对应的函数功能块中看到刚刚添加的函数，函数头部方括号括起来的部分代表函数的属性。**请注意，这个语法为 SAMSOARII 定制，请勿将该语法用在别的 C 语言程序中。**

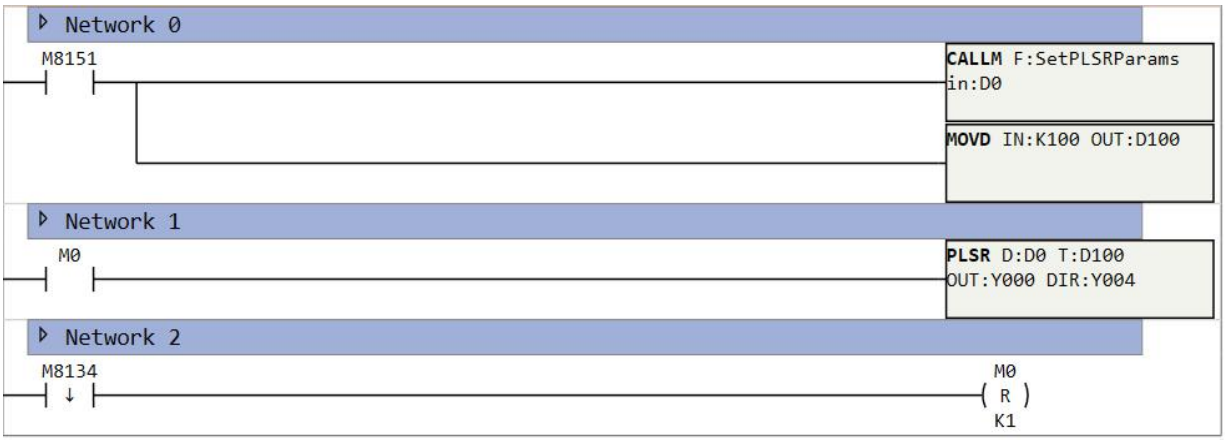
```
[PLSRParams]
void SetPLSRParams(DWORD in)
{
    in[0] = 100000; // 第1段脉冲最高频率
    in[1] = 200000; // 第1段脉冲数
    in[2] = 200000; // 第2段脉冲最高频率
    in[3] = 300000; // 第2段脉冲数
    in[4] = 150000; // 第3段脉冲最高频率
    in[5] = 200000; // 第3段脉冲数
    in[6] = 100000; // 第4段脉冲最高频率
    in[7] = 100000; // 第4段脉冲数
    in[8] = 150000; // 第5段脉冲最高频率
    in[9] = 100000; // 第5段脉冲数
    in[10] = 200000; // 第6段脉冲最高频率
    in[11] = 300000; // 第6段脉冲数
    in[12] = 0; // 第7段脉冲最高频率
    in[13] = 0; // 第7段脉冲数
}
```

图 1.3

指令表：

```
Network 004
    LD      M8151
    CALLM   SetDPLSRParams D0 // 设置分段脉冲的参数
    MOVD    K100    D100      // 加减速时间 100Ms
Network 005
    LD      M0
    PLSR    D0      D100    Y000    Y004// 当 M0 被置位，PLSR 指令开始工作
Network 006
    LDF     M8134             // 脉冲发送完成
    RST     M0      K1        // 复位 M0
```

梯形图：



3）检测到信号后运行指定步长

实现检测到 X0 信号后，再运行 50000 个脉冲停止脉冲发送，程序如下所示。首先以 50KHz 频率发送脉冲，脉冲个数为-1 表示一直发送脉冲。当 X0 检测到上升沿信号时，则执行 PLSNEXT 段落切换，进入下一段，而下一段的脉冲频率为 20KHz，脉冲个数为 50000 个。在第一段运行过程中可以修改第二段脉冲个数 50000 的值，若在第一段启动前不知道第二段的脉冲量，则可随意设置一个参数（不能为 0），在切换段落前修改即可。



4.15.6 PLSNEXT

对应命令	注释/说明	操作数（寄存器）
PLSNEXT	脉冲段切换	输出脉冲端口编号 Y

指令说明

在 PLSR/EPLSR 指令脉冲输出到达当前段最高频率并稳定输出时，脉冲段切换 PLSNEXT 指令可以指向下一段频率，如果 M0 由 OFF—>ON，则立即以加减速时间进入下一段的脉冲输出。

注意：

1. **在脉冲输出的加减速过程中，执行此指令无效；当在最后一段脉冲达到最高频率并稳定输出时，执行 PLSNEXT 指令，则立即以减速时间结束脉冲输出；**
2. **该指令只能对 PLSR/EPLSR 指令进行脉冲段切换；**
3. **PLSNEXT 指令只能使用边沿触发，否则会造成脉冲发送紊乱。**

指令的有效操作数

输入输出	类型	操作数	描述
EN	BOOL		启用输入端接通时，指令执行；否则，不执行
输出脉冲 Y	BOOL	Y	指定的脉冲输出口

实例

指令表：

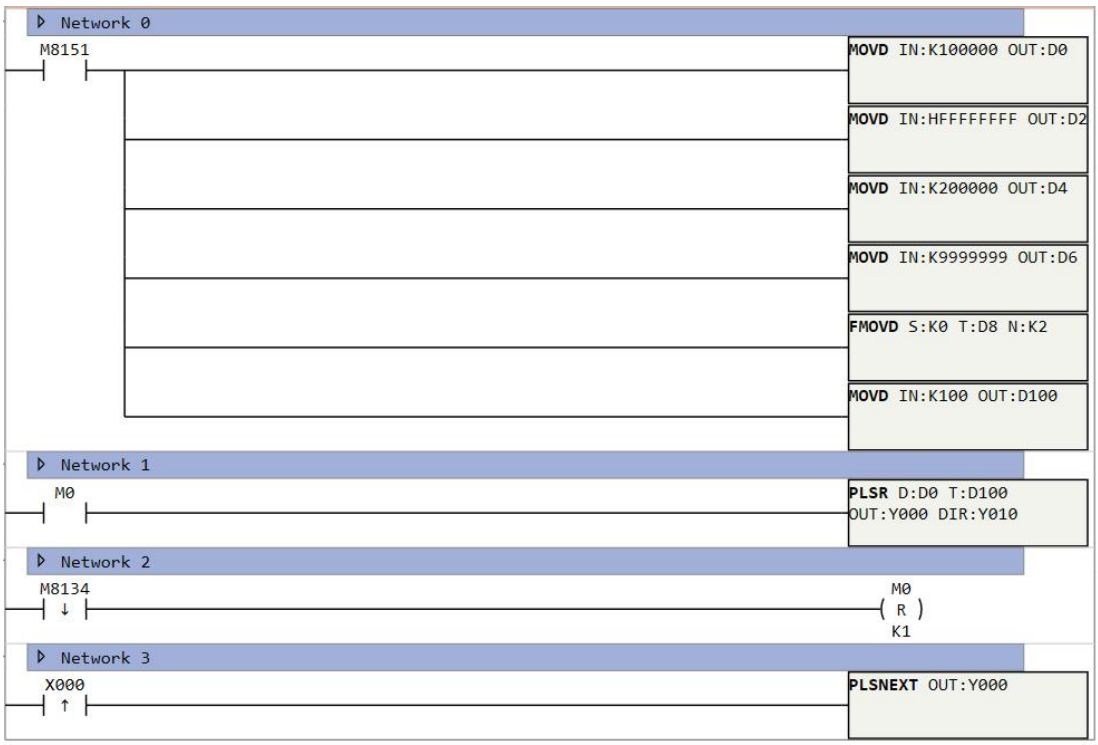
```
Network 000
    LD      M8151
    MOVD    K100000     D0          // 第 1 段脉冲最高频率 100KHz
    MOVD    HFFFFFFFF   D2          // 第 1 段脉冲数为最大值，脉冲在频率稳定时一直输出，等
待 PLSNEXT 指令
    MOVD    K200000     D4          // 第 2 段脉冲最高频率 200KHz
    MOVD    K9999999    D6          // 第 2 段脉冲数 9999999PLS
    FMOVD   K0      D8      K2      // 脉冲分段完成标志
    MOVD    K100    D100            // 加减速时间 100ms


Network 001
    LD      M0
    PLSR    D0      D100    Y000    Y010 // 当 M0 被置位，PLSR 指令开始工作

Network 002
    LDF     M8134               // 脉冲发送完成
    RST     M0      K1          // 复位 M0

Network 003
    LDP     X000                // X000 上升沿触发
    PLSNEXT Y000                // Y000 脉冲段切换
```

梯形图：



4.15.7 PLSSTOP

对应命令	注释/说明	操作数（寄存器）
PLSSTOP	脉冲停止输出	输出脉冲端口编号 Y； 停止模式； 停止时间

1. 指令说明

1）当启用输入端由 OFF 变为 ON 时，Yn 脉冲口根据设置的停止模式执行相应停止动作。当停止模式设置为 K0 时为立即停止脉冲输出，停止时间参数无效。当停止模式设置为 K1 时，则为减速停止，减速时间为停止时间的设置参数。

2）脉冲停止 PLSSTOP 指令为上升沿触发。

2. 指令的有效操作数

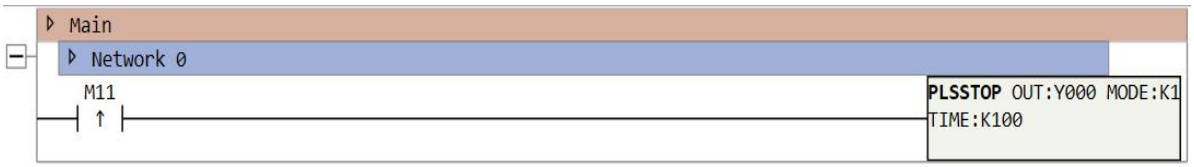
输入输出	类型	操作数	描述
EN	BOOL		**启用输入端接通时，指令执行；否则，不执行**
输出脉冲 Y	BOOL	**Y**	**指定的脉冲输出口**
停止模式	K	**K0/K1**	**K0 为立即停止，K1 为减速停止**
停止时间	D/K/H	**整数**	**减速停止时，设置减速时间参数**

3. 实例

指令表：

```
Network 003
LDP     M11         // M11 上升沿触发
PLSSTOP Y000 K1 K100 // Y000 脉冲减速停止，减速时间为 100ms
```

梯形图：



4.15.8 ZRN

回归原点 ZRN

对应命令	注释/说明	操作数（寄存器）
ZRN（32 位）	回归原点	回归速度（K/H/D） 爬行速度（K/H/D） 近点信号（X/M） 输出脉冲（Y） 方向输出口（Y）

指令说明

启用输入端为“接通”时，回归原点 ZRN 指令先以回归速度设定输出脉冲，当近点信号置位的时候，立即从回归速度在一定时间内减速到爬行速度设定输出脉冲，当近点信号复位时，停止脉冲发送。启用输入端为“断开”时，脉冲发送立即停止。

脉冲正在输出标志(只读)：

寄存器	脉冲输出 Yn
M_8134	Y000
M_8135	Y001
M_8136	Y002
M_8137	Y003
对于 FGM 系列 PLC 还具备以下 M 软元件	
M_8138	Y004
M_8139	Y005
M_8140	Y006
M_8141	Y007
M_8142	Y010
M_8143	Y011

脉冲累积计数(读写)：

寄存器	脉冲输出 Yn
D_8140(D_8141)	Y000
D_8142(D_8143)	Y001
D_8144(D_8145)	Y002
D_8146(D_8147)	Y003
对于 FGM 系列 PLC 还具备以下寄存器	
D_8148(D_8149)	Y004
D_8150(D_8151)	Y005
D_8152(D_8153)	Y006
D_8154(D_8155)	Y007
D_8156(D_8157)	Y010
D_8158(D_8159)	Y011

原点回归加减速时间（读写）：

寄存器	脉冲输出 Yn
D_8076	Y000
D_8077	Y001
D_8078	Y002
D_8079	Y003
对于 FGM 系列 PLC 还具备以下寄存器	
D_8080	Y004
D_8081	Y005
D_8082	Y006
D_8083	Y007
D_8084	Y010
D_8085	Y011

注意：

1. 使用该指令完成回原点动作后，对应输出口的脉冲累积寄存器（D8140~D8158）会自动清零，作为原点；
2. 当脉冲指令使用到输出口 Y 时，该 Y 口不能用做它用，即普通指令无法再对该 Y 口进行 ON、OFF 操作；
3. 对于 ZRN 带方向回原点指令，若需负方向回原点，需将回归速度与爬行速度均设为负值；

指令的有效操作数

输入输出	类型	操作数	描述
EN	BOOL		**启用输入端接通时，指令执行；否则，不执行**
回归速度	双整数	**K/H/D**	**回归速度设定**
爬行速度	双整数	**K/H/D**	**爬行速度设定**
近点信号	双整数	**X/M**	**指定的近点位置信号**
输出脉冲 Y	BOOL	**Y**	**指定的脉冲输出口**
输出方向 Y	BOOL	**Y**	**指定的脉冲方向输出口**

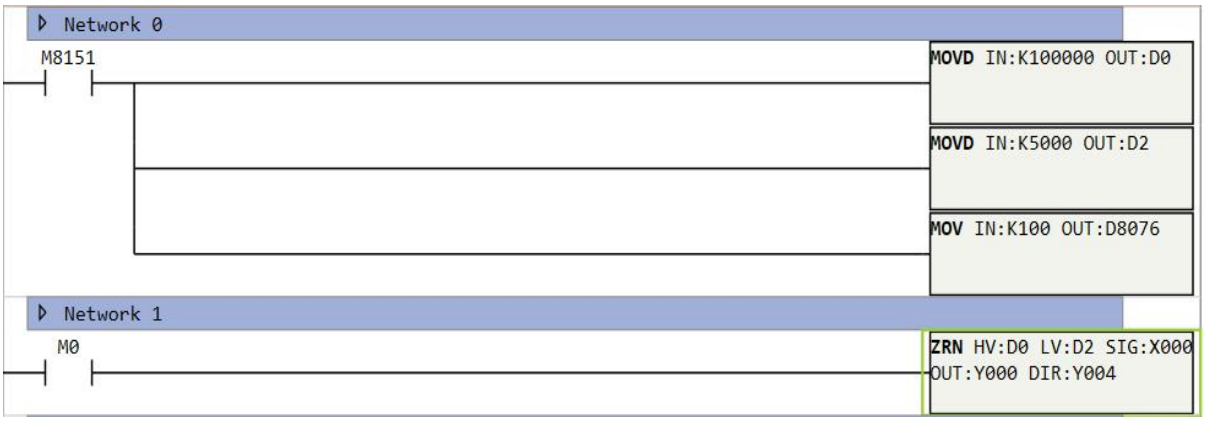
实例

指令表：

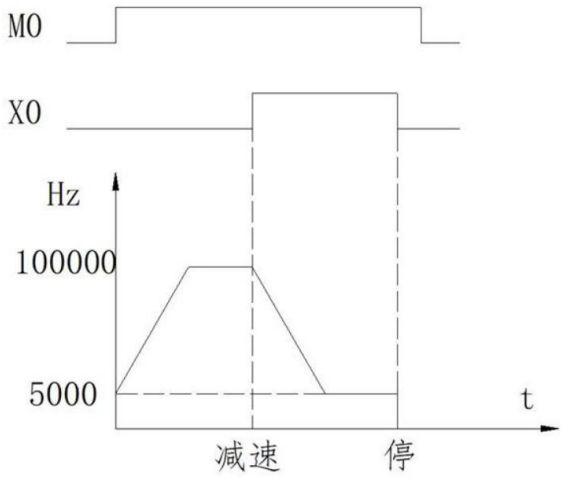
```
Network 000
    LDP     M8151
    MOVD    K100000   D0      // 回归速度，脉冲频率 100KHz
    MOVD    K5000     D2      // 爬行速度，脉冲频率 5KHz
    MOV     K100      D8076   // 减速时间 100ms

Network 001
    LD      M0
    ZRN     D0        D2      X0      Y0      Y4// 当 M0 为‘1’时，在 Y0 先输出 D0 频率的脉冲当 X0 处接收到上升沿时，Y0 输出频率在 D8076 时间内将速度由 D0 转变至 D2，X0 处接收到下降沿则会直接停止脉冲发送
```

梯形图：



时序图：



4.15.9 PTO

相对位置多段脉冲控制 PTO（32 位）

对应命令	注释/说明	操作数（寄存器）
PTO(32 位)	相对位置多段脉冲控制	指定分段脉冲参数起始软元件地址编号(D) 指定脉冲输出端口编号(Y) 指定脉冲输出方向（Y）

指令说明

1）相对位置多段脉冲控制 PTO（32 位）指令可用于实现多段按预设的频率和脉冲数的脉冲输出。启用输入端为“接通”时，立即从第一段的预设起始频率开始输出脉冲，匀加减速达到第一段的预设结束频率，同时，输出脉冲个数达到第一段预设输出脉冲个数；从第二段的预设起始频率开始输出脉冲，匀加减速达到第二段的预设结束频率，同时，输出脉冲个数达到第二段预设输出脉冲个数；依次类推，直到段脉冲的个数设定为 0 值表示结束分段，脉冲输出停止，或者脉冲指令启用输入端断开。

2）多段频率和脉冲数的设定值在脉冲参数地址起始中设定。以 Dn 为起始地址的一段区域。

以 D0（双字）为起始地址为例：D0(D1)设定总的脉冲段数，D2(D3)设定脉冲的方向标志，D4(D5)设定第 1 段脉冲的起始频率、D6(D7)设定第 1 段结束频率，D8(D9)设定第 1 段脉冲的脉冲个数、D10(D11)设定第 2 段脉冲起始频率，……，Dn+4(Dn+5)设定第(n+6)/6 段脉冲的最高频率、Dn+6（Dn+7）设定第(n+6)/6 段脉冲的结束频率、Dn+8（Dn+9）设定第（n+6/6）段脉冲数；

3）可输出频率范围为 0Hz~200KHz，脉冲量范围 K0~K2147483647，对于 FGM 系列 PLC 其最大脉冲频率为 500KHz。加减速时间是指从起始频率到稳定频率的加减速时间，当频率为 0 时停止输出。在各段脉冲输出过程中，无加速度时间设置，频率从起始频率一直加速到结束频率，整个加速过程所走的步长为该段脉冲个数，如时序图所示。

4）FGs 提供了高速脉冲输出口，16/32 点晶体管型的 PLC，Y0~Y1 为高速脉冲输出口，64 点晶体管型的 PLC，Y0~Y3 为高速脉冲输出口；FGRB 晶体管 PLC，Y0~Y1 为高速脉冲输出口，FGRS 晶体管 PLC，Y0~Y3 为高速脉冲输出口，继电器型的 PLC 不具有高速脉冲功能，输出端口的最大输出频率由继电器的吸合时间决定。对于 FGM 系列 PLC 其高速脉冲输出口为 Y0~Y7，Y10~Y11。

脉冲正在输出标志(只读)：

寄存器	脉冲输出 Yn
M_8134	Y000
M_8135	Y001
M_8136	Y002
M_8137	Y003
对于 FGM 系列 PLC 还具备以下 M 软元件	
M_8138	Y004
M_8139	Y005
M_8140	Y006
M_8141	Y007
M_8142	Y010
M_8143	Y011

脉冲方向，正方向为 1(只读)：

寄存器	脉冲输出 Yn
M_8102	Y000
M_8103	Y001
M_8104	Y002
M_8105	Y003
对于 FGM 系列 PLC 还具备以下 M 软元件	
M_8106	Y004
M_8107	Y005
M_8108	Y006
M_8109	Y007
M_8110	Y010
M_8111	Y011

脉冲累积计数(读写)：

寄存器	脉冲输出 Yn
D_8140(D_8141)	Y000
D_8142(D_8143)	Y001
D_8144(D_8145)	Y002
D_8146(D_8147)	Y003
对于 FGM 系列 PLC 还具备以下寄存器	
D_8148(D_8149)	Y004
D_8150(D_8151)	Y005
D_8152(D_8153)	Y006

D_8154(D_8155)	Y007
D_8156(D_8157)	Y010
D_8158(D_8159)	Y011

当前脉冲段(只读)：

寄存器	脉冲输出 Yn
D_8124	Y000
D_8125	Y001
D_8126	Y002
D_8127	Y003
对于 FGM 系列 PLC 还具备以下寄存器	
D_8128	Y004
D_8129	Y005
D_8130	Y006
D_8131	Y007
D_8132	Y010
D_8133	Y011

出错脉冲段位置，0 表示无错误(只读)：

寄存器	脉冲输出 Yn
D_8108	Y000
D_8109	Y001
D_8110	Y002
D_8111	Y003
对于 FGM 系列 PLC 还具备以下寄存器	
D_8112	Y004
D_8113	Y005
D_8114	Y006
D_8115	Y007
D_8116	Y010
D_8117	Y011

注意：

1. 当为 n 段脉冲时，第 1 段、第 2 段、第 3 段……第 n 段每段的脉冲频率和脉冲数的地址必须是依次连续的；

2. 晶体管的 OFF 时间具有在轻负载时会延长的特性。所以请在需要响应性时设计成让负载在较轻时加负载电阻，增加负载电流；

3. 对于脉冲输出 Y0，寄存器 D8140（D8141）中累计其脉冲个数，D8141 存储高 16 位，D8140 存储低 16 位；

4. 脉冲累积计数寄存器（D8140~D8158）是重要寄存器，可读写，当写入新值时，计数会在新值基础上进行加减；

5. 由于本指令脉冲输出带方向，所以脉冲累积计数寄存器（D8140~D8158）根据方向计数；

6. 高速脉冲输出时，脉冲累积计数寄存器（D8140~D8158）内的值是不连续的、时刻变化的，用于判断时，请使用大小比较，不要使用等于判断；

7. 第一段的起始频率系统默认为 100HZ，做高速脉冲输出时，第一段的稳定频率不要低于该值；

8. 当前脉冲段（D8124-D8133）只在脉冲发送过程中记录当前脉冲段，指令失能时自动归零；

9. PTO 指令不能使用 PLSNEXT 指令切换至下个脉冲段落。

10. 当脉冲指令使用到输出口 Y 时，该 Y 口不能用做它用，即普通指令无法再对该 Y 口进行 ON、OFF 操作。

指令的有效操作数

输入输出	类型	操作数	描述
EN	BOOL		**启用输入端接通时，指令执行；否则，不执行**
起始地址	整数/双整数	**D**	**多段参数设定值的起始地址**
输出脉冲 Y	BOOL	**Y**	**指定的脉冲输出口**
方向 Y	BOOL	**Y**	**指定脉冲输出方向**

实例

参数设置：

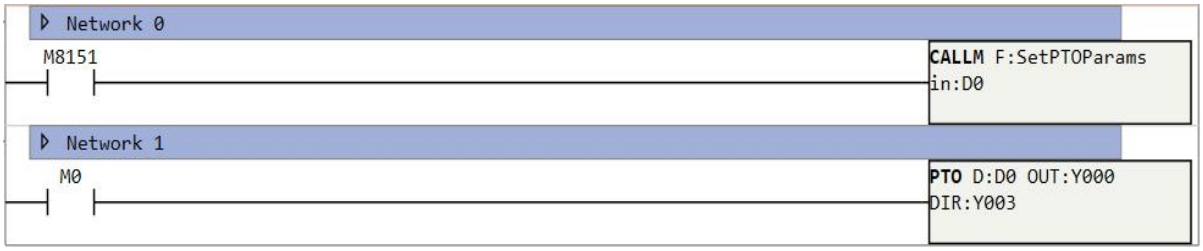
PTO 指令可以通过与 PLSR 类指令同样的方法来设置参数函数。如何创建 PLSR 指令的参数设置函数？(查看 [标题编号])

指令表：

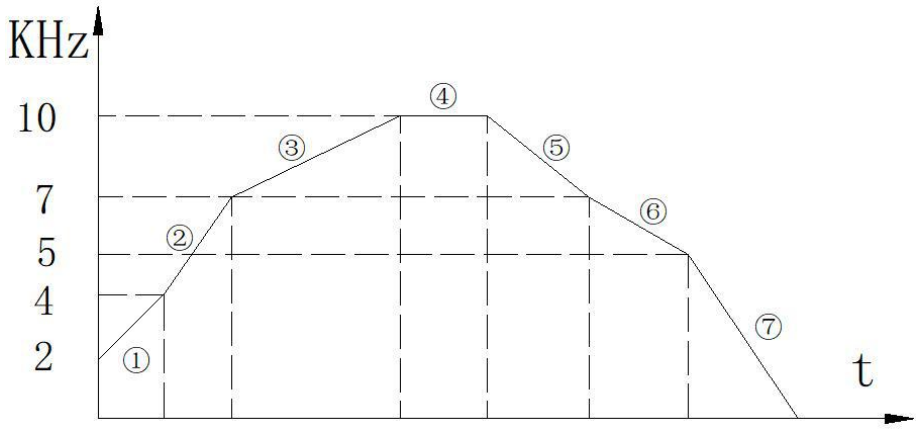
```
Network 000
    LD      M8151
    CALLM   SetPTOParams D0   // 设置多段脉冲参数

Network 001
    LD      M0
    PTO     D0      Y0      Y3   // 数据起始 D0，脉冲输出 Y0，方向 Y3
```

梯形图：



时序图：



4.15.10 DRVI/DRVA

对应命令	注释/说明	操作数（寄存器）
DRVI(32 位)/ DRVA(32 位)	相对(DRVI)/绝对(DRVA) 位置单段脉冲输出	输出频率(K/H/D) 输出脉冲数(K/H/D) 加速时间(K/H/D) 减速时间(K/H/D) 输出脉冲 Y 输出方向 Y

指令说明

1）单段脉冲输出 DRVI/DRVA 是带加减速的可用于实现指定脉冲频率和脉冲数的脉冲输出指令。启用输入端为“接通”时，脉冲从 1000HZ 匀加速到设置频率匀速输出，减速时从设置频率减速到 0，过程中发送完成指定的脉冲个数。当脉冲指令启用输入端断开时，脉冲输出立即停止。

2）DRVI 指令可输出频率范围为 0Hz~200KHz，脉冲量范围 K0~K2147483647。对于 FGM 系列 PLC 其最大脉冲频率为 500KHz。

3）FGs 提供了高速脉冲输出口，16/32 点晶体管型的 PLC，Y0~Y1 为高速脉冲输出口；64 点晶体管型的 PLC，Y0~Y3 为高速脉冲输出口；FGRB 晶体管 PLC，Y0~Y1 为高速脉冲输出口，FGRS 晶体管 PLC，Y0~Y3 为高速脉冲输出口，继电器型的 PLC 不具有高速脉冲功能，输出端口的最大输出频率由继电器的吸合时间决定。对于 FGM 系列 PLC 其高速脉冲输出口为 Y0~Y7，Y10~Y11；

4）指令可用于非对称（加减速时间单独设置）单段梯形、S 形位置控制，当特殊功能寄存器 M8069 为 ON 时所有的脉冲指令的加减速为 S 形，为 OFF 时则为梯形。

5）特殊功能寄存器 M8068 为方向极性控制，默认脉冲方向输出口为 ON 时表示正方向，OFF 时表示负方向。若设置 M8068 为 ON 则所有脉冲指令的方向输出极性发生反转，脉冲方向输出为 OFF 表示正方向，为 ON 表示负方向。

脉冲正在输出标志(只读)：

寄存器	脉冲输出 Yn
M_8134	Y000
M_8135	Y001
M_8136	Y002
M_8137	Y003
对于 FGM 型 PLC 还具备以下 M 软元件	

M_8138	Y004
M_8139	Y005
M_8140	Y006
M_8141	Y007
M_8142	Y010
M_8143	Y011

脉冲累积计数(读写)：

寄存器	脉冲输出 Yn
D_8140(D_8141)	Y000
D_8142(D_8143)	Y001
D_8144(D_8145)	Y002
D_8146(D_8147)	Y003
对于 FGM 型 PLC 还具备以下寄存器	
D_8148(D_8149)	Y004
D_8150(D_8151)	Y005
D_8152(D_8153)	Y006
D_8154(D_8155)	Y007
D_8156(D_8157)	Y010
D_8158(D_8159)	Y011

是否忽略错误，继续发送脉冲，默认忽略为 0，若设为 1，出错时停止脉冲输出(读写)：

寄存器	脉冲输出 Yn
M_8070	Y000
M_8071	Y001
M_8072	Y002
M_8073	Y003
对于 FGM 型 PLC 还具备以下寄存器	
M_8074	Y004
M_8075	Y005
M_8076	Y006
M_8077	Y007
M_8078	Y010
M_8079	Y011

注意：

1. 对于高速脉冲输出，它是针对外部高速器件的，要对脉冲进行计数，只能使用高速脉冲输入计数器，不能使用内部计数器，也不能使用 Y 边沿变化进行计数，当作为高速输出时，不能作为普通输出口用；

2. 晶体管的 OFF 时间具有在轻负载时会延长的特性。所以请在需要响应性时设计成让负载在较轻时加负载电阻，增加负载电流；

3. 对于脉冲输出 Y0，寄存器 D8140（D8141）中累计其脉冲个数，D8141 存储高 16 位，D8140 存储低 16 位；

4. 脉冲累积计数寄存器（D8140~D8158）是重要寄存器，可读写，当写入新值时，计数会在新值基础上进行加减；

5. 由于本指令脉冲输出带方向，所以脉冲累积计数寄存器（D8140~D8158）根据方向计数；

6. 高速脉冲输出时，脉冲累积计数寄存器（D8140~D8158）内的值是不连续的、时刻变化的，用于判断时，请使用大小比较，不要使用等于判断；

7. 当脉冲指令使用到输出口 Y 时，该 Y 口不能用做它用，即普通指令无法再对该 Y 口进行 ON、OFF 操作。

指令的有效操作数

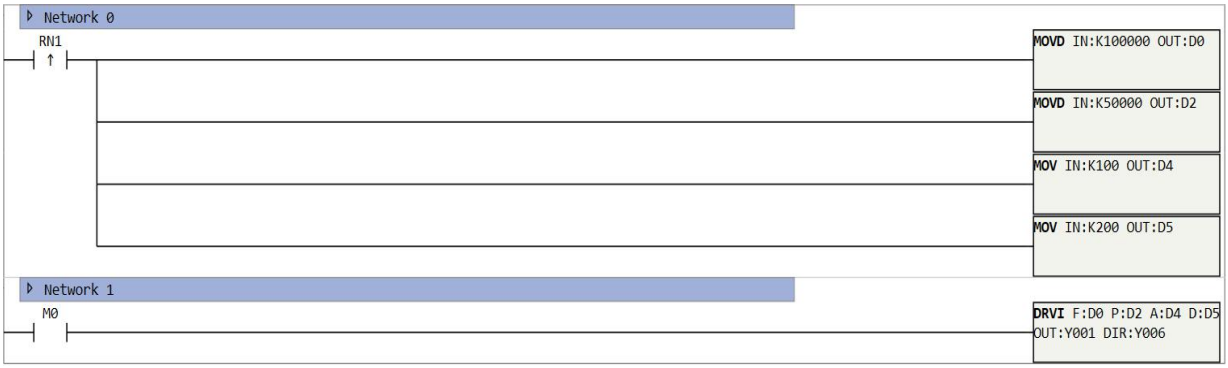
输入输出	类型	操作数	描述
EN	BOOL		**启用输入端接通时，指令执行；否则，不执行**
输出频率	双整数	**K/H/D**	**输出频率的设定值**
输出脉冲数	双整数	**K/H/D**	**输出脉冲数的设定值**
加速时间	整数	**K/H/D**	**加速时间设定值**
减速时间	整数	**K/H/D**	**减速时间设定值**
输出脉冲 Y	BOOL	**Y**	**指定的脉冲输出口**
输出方向 Y	BOOL	**Y**	**指定的脉冲方向输出口**

实例

指令表：

```
Network 000
    LDP     M8151
    MOVD    K100000   D0      // 脉冲输出数 D0D1
    MOVD    K50000    D2      // 脉冲输出频率 D2D3
    MOV     K100      D4      // 加速时间 100ms D4
    MOV     K200      D5      // 减速时间 200ms D5
Network 001
    LD      M0
    DRVI    D0   D2   D4   D5   Y1   Y6   // 当 M0 被置位 DRVI 开始工作
```

梯形图：



4.15.11 TBL

指令说明：

1）对于 FGs 系列 PLC，16/32 点晶体管型的 PLC，Y0~Y1 为高速脉冲输出口，对于 64 点晶体管型的 PLC，Y0~Y3 为高速脉冲输出口，FGRB 晶体管 PLC，Y0~Y1 为高速脉冲输出口，FGRS 晶体管 PLC，Y0~Y3 为高速脉冲输出口，对于 FGM 系列 PLC，64 点 Y0~Y7，Y10~Y11 为高速脉冲输出口；

2）对于 FGs，FGB 系列 PLC，单路脉冲：指令可输出频率范围 0Hz~200KHz，脉冲量范围 K0~K2147483647，当频率为 0 时停止输出，对 FGM 系列 PLC，最高脉冲频率高达 500KHz；

3）对于继电器型的 PLC，不具有高速脉冲功能，所有输出端口的最大输出频率有继电器的吸合时间决定；

4）该指令作用是输出一定频率的指定脉冲数的脉冲，启动的时候从 1000HZ 均加速到固定速度均速输出，结束的时候从固定速度减速到 0，完成一段脉冲输出；

5）脉冲个数和加减速时间满足如下关系式的才是合法的：

（起始频率 Hz + 结束频率 Hz）×（ 加减速时间 ms÷1000） <= 脉冲个数 × 2

6）功能：条件位置控制。

7）使用简述：如下图，数据起始地址用于存放各段频率以及脉冲数值，M200 接通后指令使能，数据将传送至 D 寄存器中，“当前脉冲段”为 1，但不会立刻发送脉冲，需要第一段起始条件 M500 接通才开始发送第一段脉冲。D 寄存器中的数据可以在程序中修改，运行到该段时会自动从 D 寄存器中取值。每段脉冲发送前都有一个等待事件和执行条件，等待事件由 wait 信号、ATC 时间、脉冲发送完成组成。图中第一段，wait 信号为 M500，只有当前要执行的段落为第一段时，M500 接通开始发送第一段，结束后接通结束标识位 M501，**并断开执行条件 M500，**“当前脉冲段”为“跳转至”的段落号。第一段执行完成后跳转至下一段，“跳转至”可填入任何已设置段落，图中为 2。第二段的等待事件为 ATC 时间，等待事件为执行条件中设置的 K2000 单位毫秒，也可设置为 D 寄存器，从第一段运行结束开始计时，2000ms 后开始运行第二段，第二段开始运行就会断开第一段 M501 的结束标志。第二段结束后接通 M503。第三段等待事件为脉冲发送完成，即第二段脉冲发送完成就立即进入第三段脉冲发送，并立刻断开 M503。第三段运行结束后跳转至第四段，第四段与第一段类似。第四段运行结束后跳转至第一段，构成一个循环。



编号	脉冲频率	脉冲个数	等待事件	执行条件	跳转至	结束标识符
1	100000	200000	等待信号	M500	2	M501
2	150000	400000	ATC时间	K2000	3	M503
3	120000	800000	脉冲发送完成	K0	4	M505
4	80000	300000	等待信号	M506	1	M507

梯形图：



脉冲正在输出标志(只读)：

寄存器	脉冲输出 Yn
M_8134	Y000
M_8135	Y001
M_8136	Y002
M_8137	Y003
对于 FGM 系列 PLC 还具备以下 M 软元件	
M_8138	Y004
M_8139	Y005
M_8140	Y006
M_8141	Y007
M_8142	Y010
M_8143	Y011

脉冲累积计数(读写)：

寄存器	脉冲输出 Yn
D_8140(D_8141)	Y000
D_8142(D_8143)	Y001
D_8144(D_8145)	Y002
D_8146(D_8147)	Y003
对于 FGM 系列 PLC 还具备以下 M 软元件	
D_8148(D_8149)	Y004
D_8150(D_8151)	Y005
D_8152(D_8153)	Y006
D_8154(D_8155)	Y007
D_8156(D_8157)	Y010
D_8158(D_8159)	Y011

相对位置的加减速时间（读写）：

寄存器	脉冲输出 Yn
D_8092	Y000
D_8093	Y001
D_8094	Y002
D_8095	Y003
对于 FGM 系列 PLC 还具备以下 M 软元件	
D_8096	Y004
D_8097	Y005
D_8098	Y006
D_8099	Y007
D_8100	Y008
D_8101	Y009

当前脉冲段(只读)：

寄存器	脉冲输出 Yn
D_8124	Y000
D_8125	Y001
D_8126	Y002
D_8127	Y003
对于 FGM 系列 PLC 还具备以下 M 软元件	
D_8128	Y004
D_8129	Y005
D_8130	Y006
D_8131	Y007
D_8132	Y010
D_8133	Y011

注意：

1. 对于高速脉冲输出，它是针对外部高速器件的，要对脉冲进行计数，只能使用高速脉冲输入计数器，不能使用内部计数器，也不能使用 Y 边沿变化进行计数，当作为高速输出时，不能作为普通输出口用；

2. 晶体管的 OFF 时间具有在轻负载时会延长的特性。所以请在需要响应性时设计成让负载在较轻时加负载电阻，增加负载电流；

3. 对于脉冲输出 Y0，寄存器 D8140（D8141）中累计其脉冲个数，D8141 存储高 16 位，D8140 存储低 16 位；

4. 脉冲累积计数寄存器（D8140~D8146）是重要寄存器，可读写，当写入新值时，计数会在新值基础上进行加减；

5. 由于本指令脉冲输出带方向，所以脉冲累积计数寄存器（D8140~D8146）根据方向计数；

6. 高速脉冲输出时，脉冲累积计数寄存器（D8140~D8146）内的值是不连续的、时刻变化的，用于判断时，请使用大小比较，不要使用等于判断；

7. 当等待事件为脉冲发送完成时，执行条件无需填写，默认为 K0；

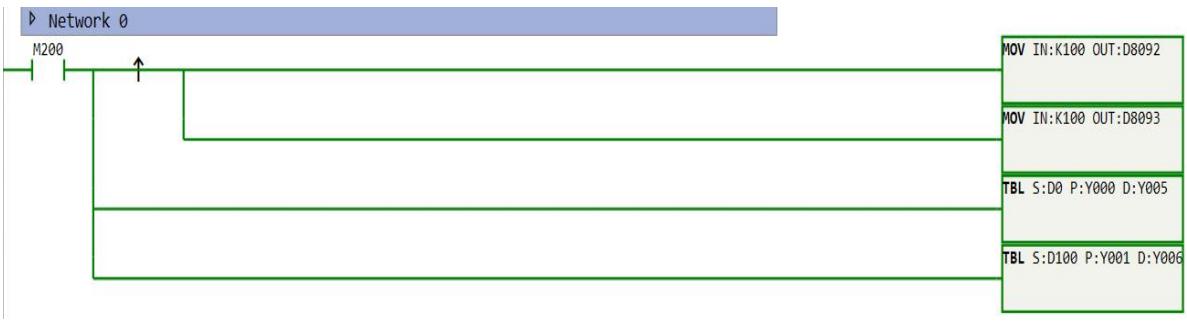
8. “模式”中的“相对”与“绝对”指填入的脉冲数为相对坐标还是绝对坐标；

9. “跳转至”可设置为任意已设置段落，最终都会构成一个循环；

10. 脉冲数和频率不可输入零，如果输入参数不满足加速合法性要求，系统将重新调整速度曲线。

实例：





如图，分别设置 Y0，Y1 脉冲输出参数。M0 接通后运动到第一点坐标（2000，200000），Y0 第一段发送结束 M1 导通，Y1 第一段结束 M1 导通，M0 被自动置 OFF。因为是相对坐标，M0 导通后运动到点（6000，500000），之后依次类推。最后一段 M3 导通后回到原点，运动到点（0,0）。

4.15.12 POLYLINEF/POLYLINEI

多段线插补 POLYLINEI/POLYLINEF

对应命令	注释/说明	操作数（寄存器）
POLYLINEI/POLYLINEF	多段线插补指令	平面编号； 单位； 直线圆弧参数设置； 参数是否映射到 D 寄存器； D 寄存器首地址。

指令说明：

1）其高速脉冲输出口为 Y0～Y7，Y10~Y11，脉冲频率范围 0~500KHz，频率非连续可调；FGRB 和 FGs32 点以下 PLC 高速脉冲输出口为 Y0~Y1，FGRS 和 FGs64 点 PLC 高速脉冲输出口为 Y0~Y3，脉冲频率范围 0~200KHz；

2）POLYLINEI/POLYLINEF 分别为整数参数指令、浮点数参数指令，对应的是以脉冲 pls 为单位和毫米 mm 为单位；

3）该指令作用是完成直线、圆弧组合运动，按照左侧的线段序列框中的线段顺序依次绘制，若相邻的两线段为共线、相切的过渡方式，则在交点处无加减速停顿，平滑过渡，反之，有加减速停顿。其直线圆弧参数设置，请参考 LINEI/LINEF、ARCI/ARCF 指令说明；

4）POLYLINE 参数映射到 D 寄存器中的主要目的是为了动态控制参数，可在 D 寄存器中修改原本参数或者在后面添加参数达到修改位置添加线段的目的。另外，在运动过程中修改速度的 D 寄存器值，可实现实时速度控制。若选择仅 D 控制，则用户无需再 POLYLINE 指令中输入任何位置、速度参数，直接由赋值的 D 寄存器参数序列决定运动。具体参数存放规则参考下段说明。

指令使用说明：



POLYLINEF 指令图

1） 首先选择平面编号，平面设置请参考“平面系统设置”，若平面系统未启用，则无法正常运行；

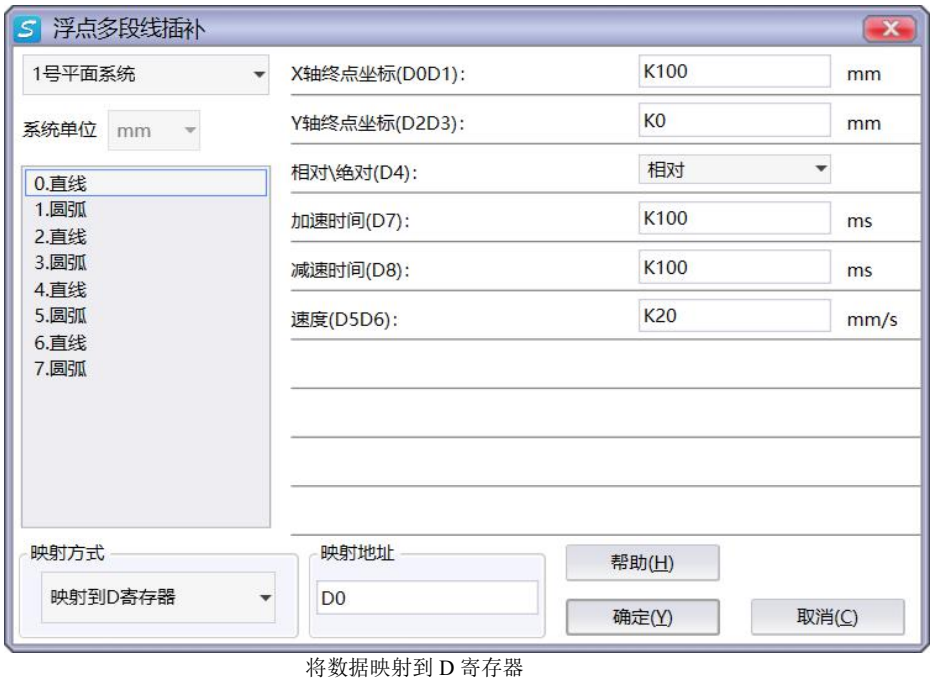
2） 该指令参数的单位，可以为毫米 mm 或者脉冲 pls，分别对应 POLYLINEF 和 POLYLINEI 指令。建议用户使用 mm 作为单位，参数贴合实际，无需换算，不易出错；

3） 在左侧线段序列框中右击鼠标，可选择增加、插入、删除、移动图元；

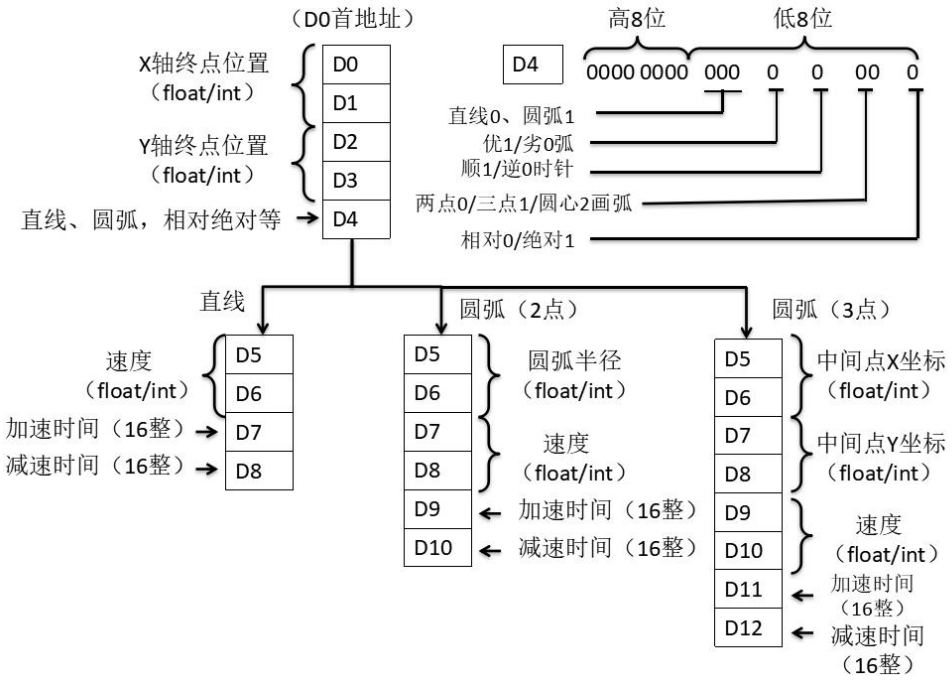
4） 直线圆弧的参数设置请参考 LINEI/LINEF、ARCI/ARCF 指令；

5） 位置和速度参数均可输入 K/D 型数据，分别为不可调和可调。其中，速度参数输入为 D 时，D 寄存器中的值发生改变，速度也相应实时调整，在加减速过程中，调整速度参数，调速不会立即响应，速度的变化会在加减速结束后立刻调速。**强调，使用 MOV 类指令赋值位置、速度参数时，使用 MOVD 还是 MOVF 要根据选择指令的 POLYLINEI 和 POLYLINEF 类型决定。POLYLINEI 为整型指令，对应脉冲单位 pls，其参数需使用 MOVD 赋值，POLYLINEF 为浮点型指令，对应毫米 mm 单位，其参数需要使用 MOVF 赋值。而加减速时间参数单位为 ms，是一个 16 位整数，需使用 MOV 指令，与插补指令类型无关；**

6） 将数据映射到 D 寄存器，如图“映射到 D 寄存器”，是在指令使能时，会将已设置的位置参数、速度参数以特定顺序存放在 D 寄存器中，**并且改变了位置、速度参数的映射路径，若要修改位置或实时调速，必须调整新映射到 D 寄存器中的地址参数才可，**映射的参数存放规则如图“将参数映射到 D 寄存器的参数存放规则”，按照线段的顺序依次循环。选择映射后，参数下方会显示映射到的 D 寄存器地址，用户可在线段未执行前对参数进行更改，调整图形形状。但需要注意的是，线段的执行具有前瞻性，当前正在执行的线段的后一段已被处理，若需改变图元形状，必须从当前正在执行线段的后两段开始改变，否则会出现前瞻性错误，导致报错或运行混乱。该功能仅适用于需要对形状进行小修改时使用。

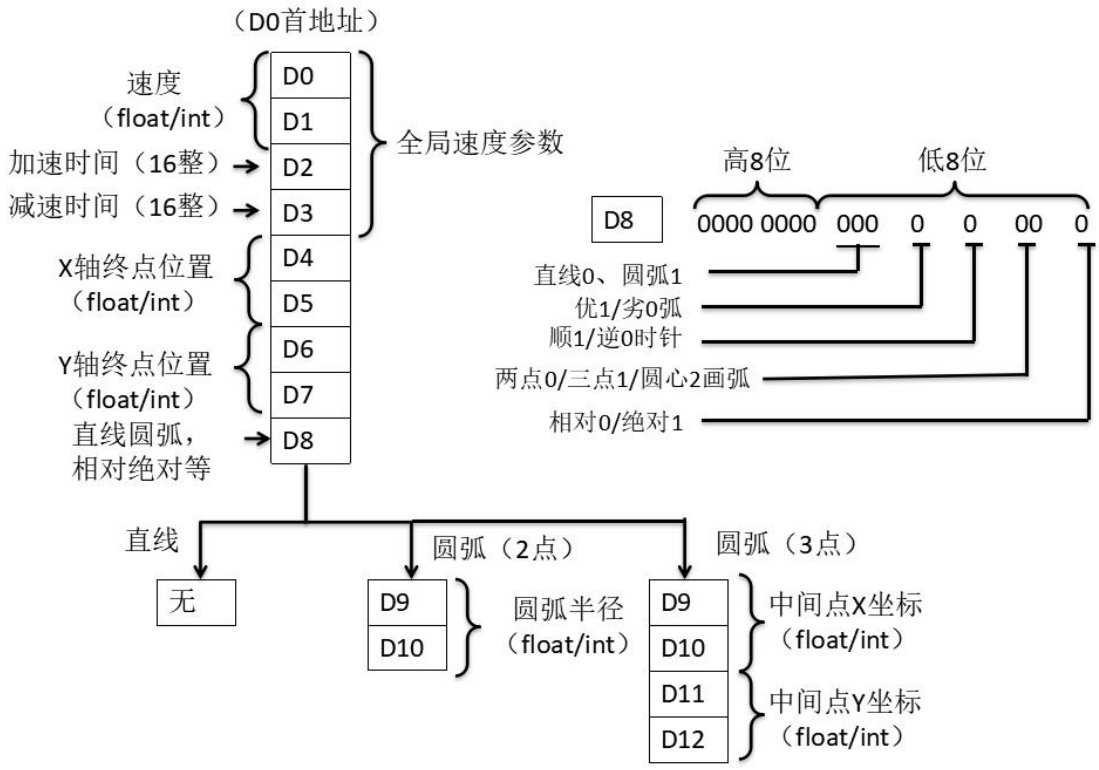


将数据映射到 D 寄存器



将参数映射到 D 寄存器的参数存放规则

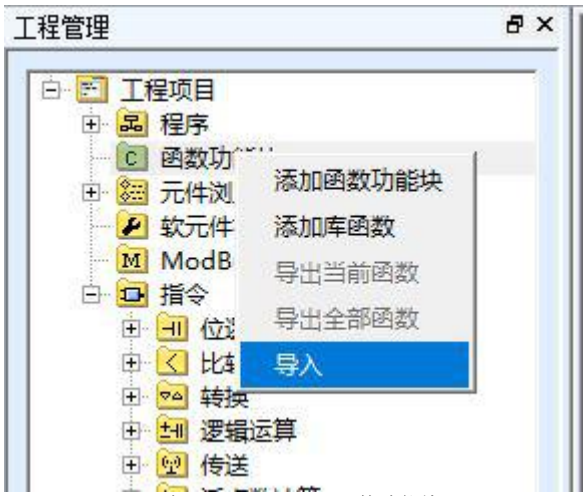
7） 仅 D 寄存器控制，选择此模式时，直线圆弧参数均仅由 D 寄存器控制，指令中无法填入数据，已填入的数据也无效，其数据存储规则如图“仅 D 寄存器控制参数存放规则”，其线段参数依照设置依次循环，需要注意的是，仅 D 控制具有一个全局速度，如果其它线段中的速度与全局速度相同，则可省略，若不相同，则可单独添加在线段数据尾部加入该线段的速度参数，并且将控制位“单独速度”设为 1。实时调速时，调整全局速度不影响具有单独速度的线段，若需要对具有单独速度的线段进行调速，需调整其单独速度地址中的值。该功能适用于需要对图形形状进行批量修改时使用。



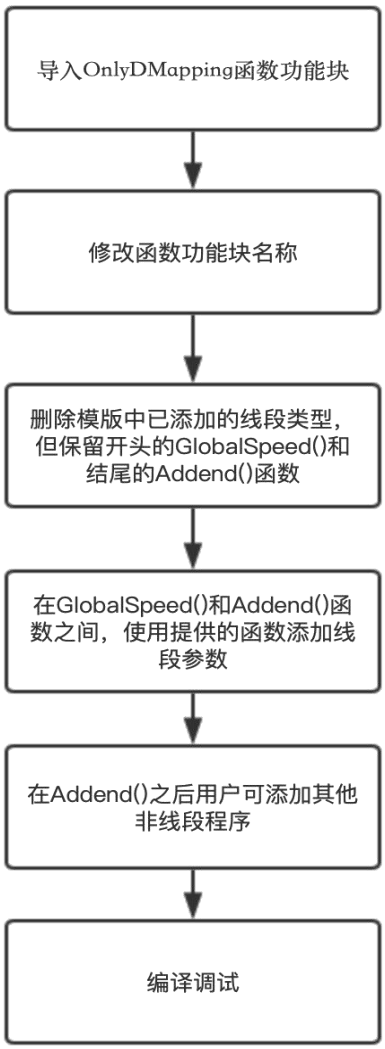
仅 D 寄存器控制参数存放规则

8） 选择仅 D 寄存器控制时，若不采用其他方式，需要使用 MOV 类指令不停的赋值 D 寄存器，易出错且麻烦，为设计方便，开发了针对仅 D 控制的函数功能块模板“OnlyDMappingI”和“OnlyDMappingF”，

该函数的功能块主要功能是通过函数将线段参数赋值到 D 寄存器中。其添加如下图，导入函数功能块，并根据插补指令类型选择文件“OnlyDMappingI.ssc”和“OnlyDMappingF.ssc”，若需要添加多个该函数，需要修改函数功能块名称以及内部子函数名称，避免函数重命名。在函数功能块中有详细的注释，在此仅简述其用法框图如下。



添加 OnlyDMapping 函数功能块



OnlyDMapping 函数功能块使用框图

脉冲正在输出标志(只读)：

寄存器	脉冲输出 Yn
M_8134	Y000
M_8135	Y001
M_8136	Y002
M_8137	Y003
对于 FGM 系列 PLC 还具备以下 M 软元件	
M_8138	Y004
M_8139	Y005
M_8140	Y006
M_8141	Y007
M_8142	Y010
M_8143	Y011

平面系统脉冲正在输出标志(只读)：

对于 FGM 系列 PLC，FGRS_C8X8T 具备以下 M 软元件	
M_8144	平面系统 1
M_8145	平面系统 2
M_8146	平面系统 3
M_8147	平面系统 4
M_8148	平面系统 5

脉冲累积计数(读写)：

寄存器	脉冲输出 Yn
D_8140(D_8141)	Y000
D_8142(D_8143)	Y001
D_8144(D_8145)	Y002
D_8146(D_8147)	Y003
对于 FGM 系列 PLC 还具备以下寄存器	
D_8148(D_8149)	Y004
D_8150 (D_8151)	Y005
D_8152(D_8153)	Y006
D_8154(D_8155)	Y007
D_8156 (D_8157)	Y010
D_8158 (D_8159)	Y011

当前脉冲段(只读)：

寄存器	脉冲输出 Yn
D_8124	Y000
D_8125	Y001
D_8126	Y002
D_8127	Y003
对于 FGM 系列 PLC 还具备以下寄存器	
D_8128	Y004
D_8129	Y005
D_8130	Y006
D_8131	Y007
D_8132	Y010
D_8133	Y011

出错脉冲段位置，0 表示无错误(只读)：

寄存器	脉冲输出 Yn
D_8108	Y000
D_8109	Y001
D_8110	Y002
D_8111	Y003

对于 FGM 系列 PLC 还具备以下寄存器	
D_8112	Y004
D_8113	Y005
D_8114	Y006
D_8115	Y007
D_8116	Y010
D_8117	Y011

注意：

1. 对于高速脉冲输出，它是针对外部高速器件的，要对脉冲进行计数，只能使用高速脉冲输入计数器，不能使用内部计数器，也不能使用 Y 边沿变化进行计数，当作为高速输出时，不能作为普通输出口用；

2. 晶体管的 OFF 时间具有在轻负载时会延长的特性。所以请在需要响应性时设计成让负载在较轻时加负载电阻，增加负载电流；

3. 脉冲累积计数寄存器（D8140~D8158）是重要寄存器，可读写，当写入新值时，计数会在新值基础上进行加减；

4. 由于本指令脉冲输出带方向，所以脉冲累积计数寄存器（D8140~D8158）根据方向计数；

5. 高速脉冲输出时，脉冲累积计数寄存器（D8140~D8158）内的值是不连续的、时刻变化的，用于判断时，请使用大小比较，不要使用等于判断；

6. 加减速时间不要设置太小，否则无加减速效果；

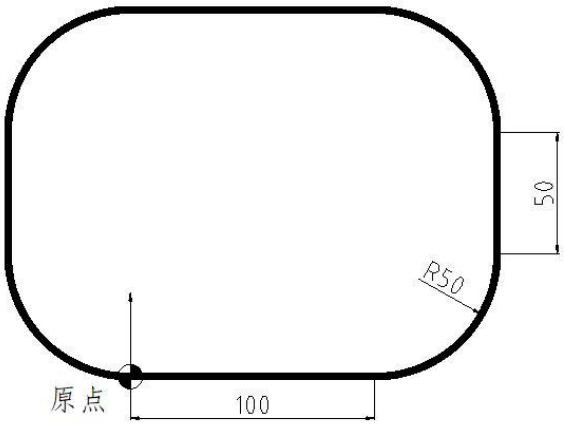
7. 当单位设置为脉冲 pls 时，速度单位为 pls/s，如果平面系统设置的 X 轴 Y 轴脉冲当量不相等，则默认以 X 轴脉冲当量计算其真实速度；

8. 脉冲正在发送标志位，如 M8134 仅代表 Y0 是否正在输出脉冲，如果需要检测平面系统 1 中是否有脉冲正在发送，使用 M8144，运行插补指令时平面系统中任意一轴发送脉冲则该位为 ON；

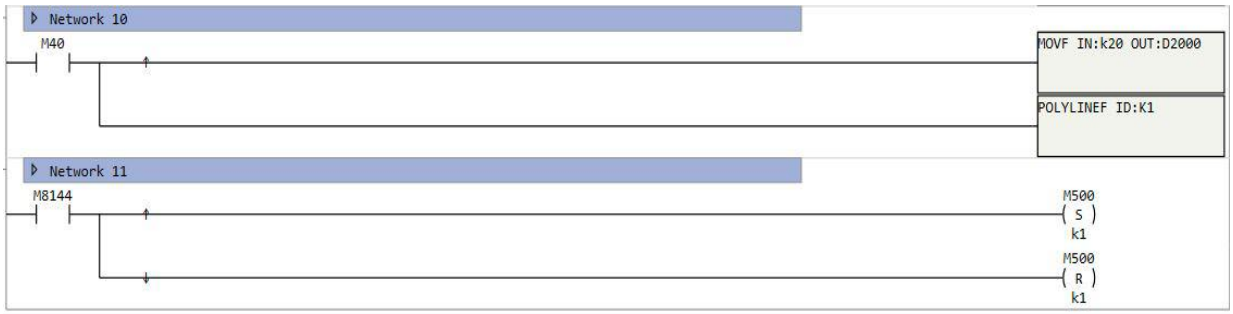
9. 当脉冲指令使用到输出口 Y 时，该 Y 口不能用做它用，即普通指令无法再对该 Y 口进行 ON、OFF 操作。

实例：

绘制如图所示导圆矩形，其直线部分速度为 100mm/s 且直线部分速度可调，圆弧部分速度为 50mm/s，且速度独立不受直线部分影响。开始绘图时置位 M500，完成后复位 M500。



梯形图：



不将数据映射到 D 寄存器：

依次设置添加直线圆弧，并设置其位置参数，相对绝对均可，在调速方面，直线的速度框中需输入 D 寄存器，如 D2000，根据插补指令类型不同，选择 MOVD 和 MOVF 对速度进行赋值。而圆弧速度框中需输入固定的速度 K50。

将数据映射到 D 寄存器：

依次设置添加直线圆弧，并设置其位置参数，可直接在所有速度参数框中输入 K 值，因为即使使用 D 寄存器值，映射到 D 后也会重新建立新 D 寄存器的速度映射关系，调整原来的 D 寄存器值无法调速。根据指令框中提示的速度参数 D 地址，可调整 D 地址实现调速。注意此时每条线段都具有单独的速度参数地址，根据需要逐个修改。

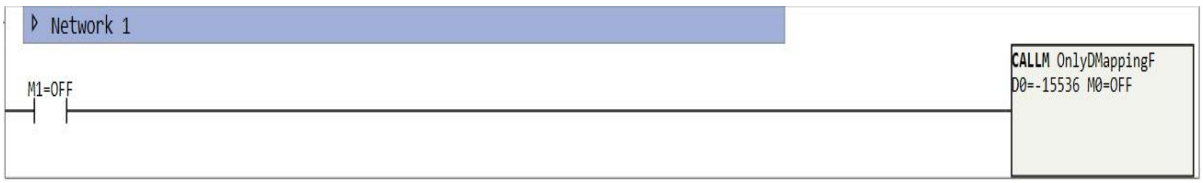
仅 D 寄存器控制：

导入函数功能块模板 OnlyDMappingF，设置如下。另外，使用函数功能块，如下图，需要在使能 POLYLINEF 前，接通函数功能块一次，完成 D 寄存器赋值。

```
//设置全局速度和加减速时间
ParaAddr = GlobalSpeedF(ParaAddr, 100, 200, 200);

ParaAddr = AddLineF(ParaAddr, 100, 0, 0);
ParaAddr = Add2ArcF_S(ParaAddr, 50, 50, 50, 0, 0, 0, 50, 200, 200);
ParaAddr = AddLineF(ParaAddr, 0, 100, 0);
ParaAddr = Add2ArcF_S(ParaAddr, -50, 50, 50, 0, 0, 0, 50, 200, 200);
ParaAddr = AddLineF(ParaAddr, -100, 0, 0);
ParaAddr = Add2ArcF_S(ParaAddr, -50, -50, 50, 0, 0, 0, 50, 200, 200);
ParaAddr = AddLineF(ParaAddr, 0, -100, 0);
ParaAddr = Add2ArcF_S(ParaAddr, 50, -50, 50, 0, 0, 0, 50, 200, 200);

ParaAddr = AddEndF(ParaAddr); //结束
//用户添加其它程序
```



4.15.13 LINEF/LINEI

直线插补 LINEI/LINEF

对应命令	注释/说明	操作数（寄存器）
LINEI/LINEF	直线插补指令	平面编号； 单位； X 轴终点坐标（K/D）； Y 轴终点坐标（K/D）； 相对/绝对坐标； 加速时间（K/D）； 减速时间（K/D）； 速度（K/D）

指令说明：

1）FGm_64MT_A 高速脉冲输出口为 Y0～Y7，Y10~Y11，脉冲频率范围 0~500KHz，频率非连续可调；FGRB 和 FGs32 点以下 PLC 高速脉冲输出口为 Y0~Y1，FGRS 和 FGs64 点 PLC 高速脉冲输出口为 Y0~Y3，脉冲频率范围 0~200KHz；

2）LINEI/LINEF 分别为整数参数指令、浮点数参数指令，对应的是以脉冲 pls 为单位和毫米 mm 为单位；

3）该指令作用是在指定的平面系统中，完成从当前所在位置直线运行到终点位置；

4）运行长度和加减速时间满足如下关系式的才是合法的（实际应用时尽量满足该关系式，若不满足程序会自动调整）：

0.5 * （加速时间 +减速时间） * 速度 <= 直线长度

当用户设置的参数非常不合理，插补指令会在检测到错误时停止脉冲输出，并在 D8176 寄存器中提供错误码，用户可根据错误码排查错误。当用户设置的参数是可调整不合理时，程序会自动修正参数。

指令使用说明：

1） 首先选择平面编号，平面设置请参考“平面系统设置”，若平面系统未启用，则无法正常运行；

2） 选择该指令参数的单位，可以为毫米 mm 或者脉冲 pls，分别对应指令 LINEF 和 LINEI，建议用户使用 mm 作为单位，参数贴合实际，无需换算，不易出错；

3） X 轴终点坐标，Y 轴终点坐标是直线运行的终点位置，可输入 K/D 型参数，使用 D 寄存器输入参数时，终点坐标使用 MOVD 指令还是 MOVF 指令赋值 D 寄存器，取决于插补指令是 LINEI 型还是 LINEF 型；

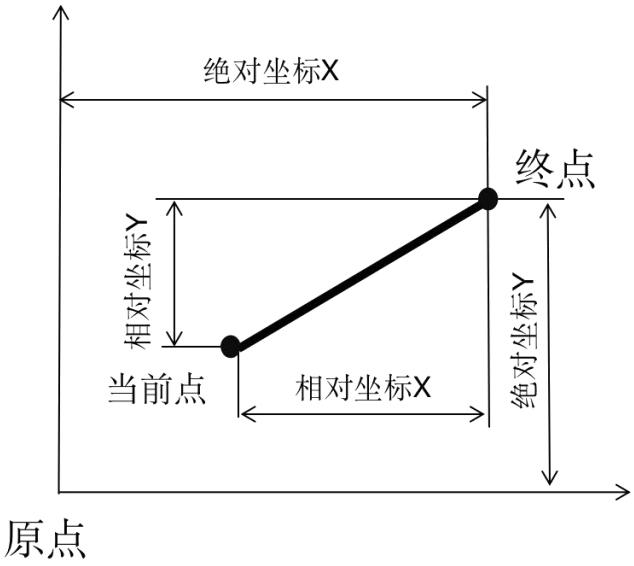
4） 相对/绝对坐标。若选择为相对坐标，则设置的终点坐标为以当前点为基础的偏移量。若为绝对坐标，则终点坐标表示相对原点的绝对位置，如相对/绝对示意图；

5） 加速时间，减速时间，可直接设置为常数 K，也可设置为 D 寄存器，指令初始化时，读取一次 D 寄存器中的值作为加减速时间，加减速时间只能使用 MOV 赋值，其为 16 位整型，单位为 ms。受算法的插补周期影响，如果加减速时间太短，脉冲频率跳变大，无法起到加减速效果。插补加减速为非对称 S 形加减速，加减速段由加减速时长控制，如非对称 S 形加减速图。

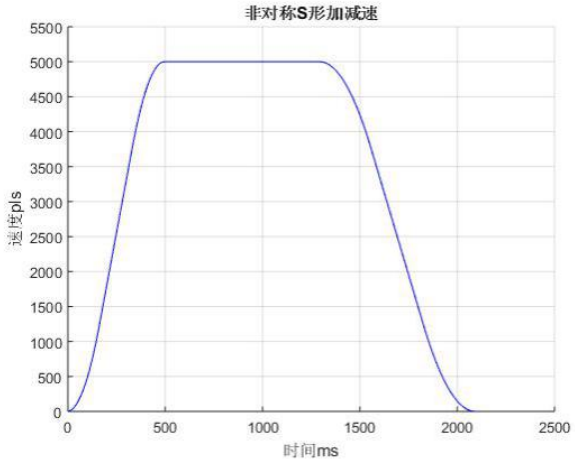
6） 速度代表的是 XY 轴合成速度，速度参数可设置为 K 或 D，若设置为常数 K，则直线速度不可调，如果设置为 D 寄存器，当 D 寄存器值变化时，速度也会相应变化，可实现实时调速，在加减速过程中，调整速度参数，速度的变化会在加减速结束后立刻调速。但需要特别注意，使用 MOVD 还是 MOVF 赋值速度的 D 寄存器取决于插补指令的类型是 LINEI 还是 LINEF。当单位设置为脉冲 pls 时，速度单位为 pls/s，使用 MOVD 赋值速度，反之，使用 MOVF。如果平面系统设置的 X 轴 Y 轴脉冲当量不相等，则默认以 X 轴脉冲当量计算其真实速度。当加减速、速度参数不满足上述合法关系式，系统将自动降低速度参数，使其合法，调整方式参考不合法 S 形调整图。



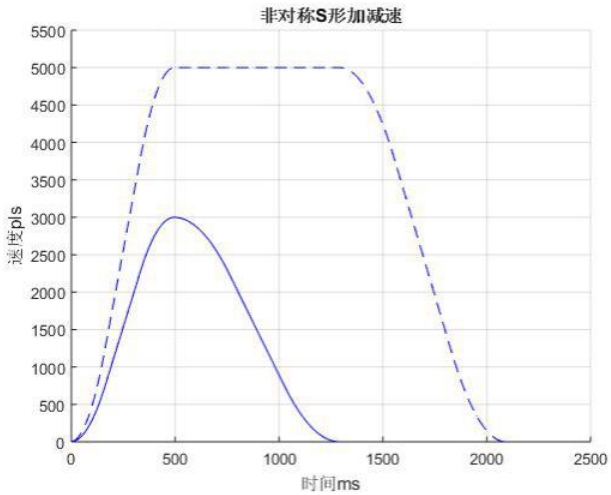
LINE 指令图



相对/绝对示意图



非对称 S 形加减速图



不合法 S 形调整图

脉冲正在输出标志(只读)：

寄存器	脉冲输出 Yn
M_8134	Y000
M_8135	Y001
M_8136	Y002
M_8137	Y003
对于 FGM 系列 PLC	

还具备以下 M 软元件	
M_8138	Y004
M_8139	Y005
M_8140	Y006
M_8141	Y007
M_8142	Y010
M_8143	Y011

平面系统脉冲正在输出标志(只读)：

对于 FGM 系列 PLC， FGRS_C8X8T 具备以下 M 软元件	
M_8144	平面系统 1
M_8145	平面系统 2
M_8146	平面系统 3
M_8147	平面系统 4
M_8148	平面系统 5

脉冲累积计数(读写)：

寄存器	脉冲输出 Yn
D_8140(D_8141)	Y000
D_8142(D_8143)	Y001
D_8144(D_8145)	Y002
D_8146(D_8147)	Y003
对于 FGM 系列 PLC 还具备以下寄存器	
D_8148(D_8149)	Y004
D_8150 (D_8151)	Y005
D_8152(D_8153)	Y006
D_8154(D_8155)	Y007
D_8156 (D_8157)	Y010
D_8158 (D_8159)	Y011

注意：

1. 对于高速脉冲输出，它是针对外部高速器件的，要对脉冲进行计数，只能使用高速脉冲输入计数器，不能使用内部计数器，也不能使用 Y 边沿变化进行计数，当作为高速输出时，不能作为普通输出口用；

2. 晶体管的 OFF 时间具有在轻负载时会延长的特性。所以请在需要响应性时设计成让负载在较轻时加负载电阻，增加负载电流；

3. 脉冲累积计数寄存器（D8140~D8158）是重要寄存器，可读写，当写入新值时，计数会在新值基础上进行加减；

4. 由于本指令脉冲输出带方向，所以脉冲累积计数寄存器（D8140~D8158）根据方向计数；

5. 高速脉冲输出时，脉冲累积计数寄存器（D8140~D8158）内的值是不连续的、时刻变化的，用于判断时，请使用大小比较，不要使用等于判断；

6. 加减速时间不要设置太小，否则无加减速效果；

7. 当单位设置为脉冲 pls 时，速度单位为 pls/s，如果平面系统设置的 X 轴 Y 轴脉冲当量不相等，则默认以 X 轴脉冲当量计算其真实速度；

8. 脉冲正在发送标志位，如 M8134 仅代表 Y0 是否正在输出脉冲，如果需要检测直线是否运行，则需并联两轴的发送标志位，参考示例；

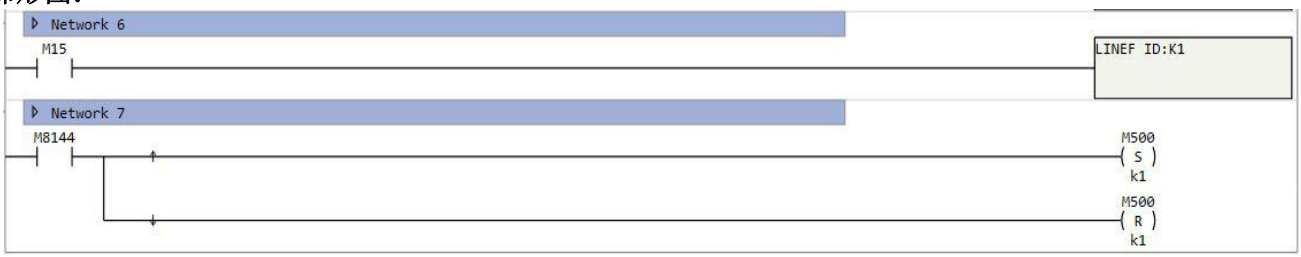
9. 目标点坐标不要与当前点坐标重合；

10. 当脉冲指令使用到输出口 Y 时，该 Y 口不能用做它用，即普通指令无法再对该 Y 口进行 ON、OFF 操作。

实例：

从当前位置，运动到 X、Y 轴各向正方向偏移 100mm 的终点，“直线运动”开始时置位 M500，结束后复位 M500。平面系统设置 X 轴为 Y0 ， Y 轴为 Y1。直线参数设置如“LINE 指令图”，梯形图如下。

梯形图：



4.15.14 ARCF/ARCI

圆弧插补 ARCI/ARCF

对应命令	注释/说明	操作数（寄存器）
ARCI/ARCF	圆弧插补指令	平面编号； 单位； X 轴终点坐标（K/D）； Y 轴终点坐标（K/D）； 相对/绝对坐标； 加速时间（K/D）； 减速时间（K/D）； 速度（K/D）； 圆弧类型（两点，三点）； 两点：半径（K/D）， 顺逆时针， 优劣弧； 三点：中间点 X 坐标（K/D）， 中间点 Y 坐标（K/D）；

指令说明：

1）FGm_64MT_A 高速脉冲输出口为 Y0～Y7，Y10~Y11，脉冲频率范围 0~500KHz，频率非连续可调；FGRB 和 FGs32 点以下 PLC 高速脉冲输出口为 Y0~Y1，FGRS 和 FGs64 点 PLC 高速脉冲输出口为 Y0~Y3，脉冲频率范围 0~200KHz；

2）ARCI/ARCF 分别为整数参数指令、浮点数参数指令，对应的是以脉冲 pls 为单位和毫米 mm 为单位；

3）该指令作用是，在指定的平面系统中，从当前所在位置按设置参数圆弧运行到终点位置；

4）运行长度和加减速时间满足如下关系式的才是合法的（实际应用时尽量满足该关系式，若不满足程序会自动调整速度）：

0.5 * （加速时间 +减速时间） * 速度 <= 圆弧弧长

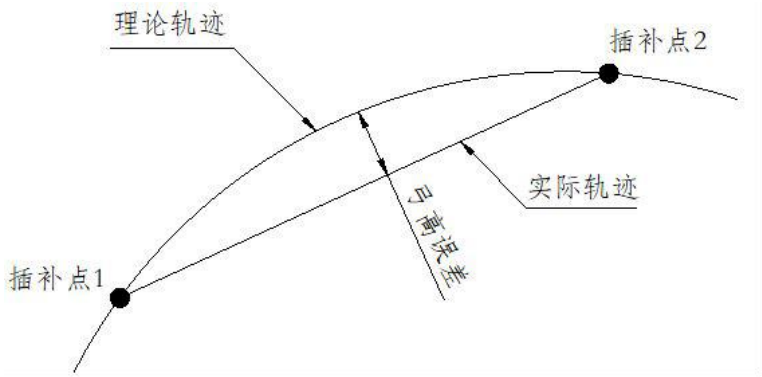
当用户设置的参数非常不合理，插补指令会在检测到错误时停止脉冲输出，并在 D8176 寄存器中提供错误码，用户可根据错误码排查错误。当用户设置的参数是可调整不合理时，程序会自动修正参数。

5）两点圆弧模式时，参数设置时应注意，当前点距离目标点直线长度应小于直径，若不满足该条件，则以该直线距离为直径，重新规划圆弧路径；

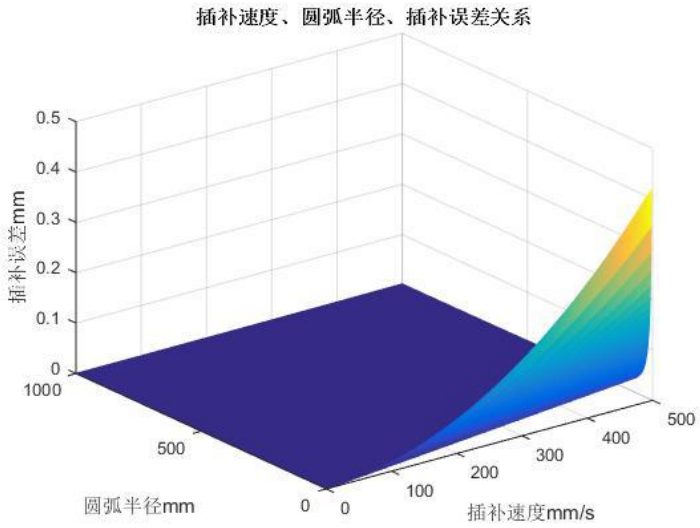
6）三点圆弧模式时，参数设置时应注意，当前点、中间点、目标点不能在一条直线上；

7）圆心画弧模式时，参数设置时应注意，圆心坐标必须在起点、终点连线的垂直平分线上，不能任意制定。

8）当圆弧半径较小，速度较大时，如果其满足加减速合法性要求，还需考虑是否会造成过大的弓高误差，速度、半径、弓高误差的关系如图，当圆弧半径较小，速度较高时容易造成较大的弓高误差。若弓高误差过大，则系统会降速处理（小圆调速特性），实际使用时，不要出现半径过小的圆弧。



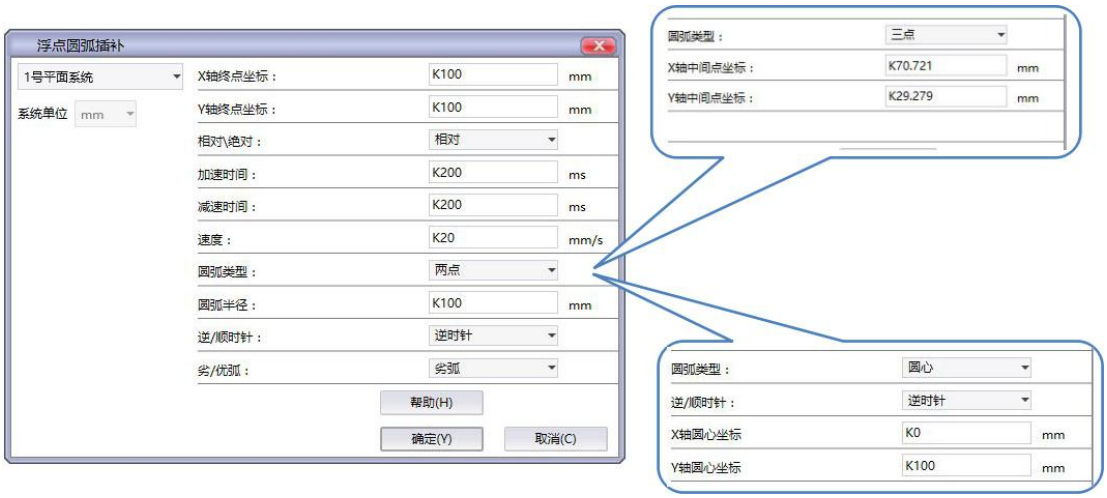
弓高误差



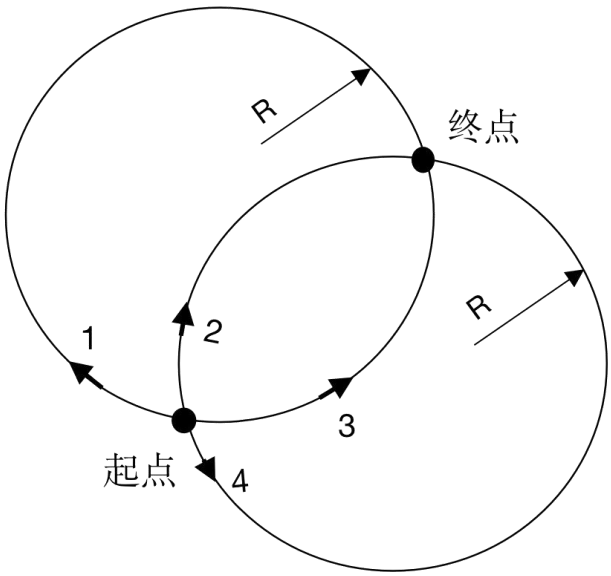
速度、半径、弓高误差关系图

指令使用说明：

1） 首先选择平面编号，平面设置请参考“平面系统设置”，若平面系统未启用，则无法正常运行；
2） 指令单位可以为毫米 mm 或者脉冲 pls，分别对应指令 ARCF 和 ARCI，建议用户使用 mm 作为单位，参数贴合实际，无需换算，不易出错；
3） X 轴终点坐标，Y 轴终点坐标是圆弧运行的终点位置，可输入数据类型 K/D，使用 D 寄存器输入参数时，终点坐标使用 MOVD 指令还是 MOVF 指令赋值 D 寄存器，取决于插补指令是 ARCI 型还是 ARCF 型；
4） 相对/绝对坐标。若选择为相对坐标，则设置的终点坐标为以当前点为基础的偏移量。若为绝对坐标，则终点坐标表示相对世界坐标原点的绝对位置，参考直线指令的相对/绝对示意图；
5） 加速时间，减速时间，可直接设置为常数 K，也可设置为 D 寄存器，指令初始化时，读取一次 D 寄存器中的值作为加减速时间。加减速时间只能使用 MOV 赋值，为 16 位整型，单位为 ms。受算法的插补周期影响，如果加减速时间太短，脉冲频率跳变大，无法起到加减速效果。插补加减速为非对称 S 形加减速，加减速段分别由加减速时间控制，参考直线指令的非对称 S 形加减速图。
6） 速度代表的是 XY 轴合成速度，速度参数可设置为 K 或 D，若设置为常数 K，则直线速度不可调，如果设置为 D 寄存器，当 D 寄存器值变化时，速度也会相应变化，可实现实时调速，在加减速过程中，调整速度参数，速度的变化会在加减速结束后立刻调速。但需要特别注意，使用 MOVD 还是 MOVF 赋值速度的 D 寄存器取决于插补指令的类型是 ARCI 还是 ARCF。当单位设置为脉冲 pls 时，速度单位为 pls/s，使用 MOVD 赋值速度，反之，使用 MOVF。如果平面系统设置的 X 轴 Y 轴脉冲当量不相等，则默认以 X 轴脉冲当量计算其真实速度。当加减速、速度参数不满足上述合法关系式，系统将自动降低速度参数，使其合法，参考直线指令的不合法 S 形调整图。
7） 圆弧类型分为两点画弧、三点画弧、圆心画弧，两者需要设置的参数不同，选择两点画圆时如两点画圆示意图，以确定半径从当前点运动到目标点的圆弧路径有 4 条，根据时针顺序，优劣弧，可唯一确定一条圆弧路径。当选择三点画弧时，只需添加中间点坐标即可，三点确定唯一圆弧，但三点不要在一条直线上。当选择圆心画弧时，已知圆心、起点、终点、顺逆时针可以唯一确定一条圆弧。



ARCF 指令图，绘制半径 100mm 的 1/4 圆弧



两点画圆示意图

脉冲正在输出标志(只读)：

寄存器	脉冲输出 Yn
M_8134	Y000
M_8135	Y001
M_8136	Y002
M_8137	Y003
对于 FGM 系列 PLC 还具备以下 M 软元件	
M_8138	Y004
M_8139	Y005
M_8140	Y006
M_8141	Y007
M_8142	Y010
M_8143	Y011

平面系统脉冲正在输出标志(只读)：

对于 FGM 系列 PLC，FGRS_C8X8T 具备以下 M 软元件	
M_8144	平面系统 1

M_8145	平面系统 2
M_8146	平面系统 3
M_8147	平面系统 4
M_8148	平面系统 5

脉冲累积计数(读写)：

寄存器	脉冲输出 Yn
D_8140(D_8141)	Y000
D_8142(D_8143)	Y001
D_8144(D_8145)	Y002
D_8146(D_8147)	Y003
对于 FGM 系列 PLC 还具备以下寄存器	
D_8148(D_8149)	Y004
D_8150 (D_8151)	Y005
D_8152(D_8153)	Y006
D_8154(D_8155)	Y007
D_8156 (D_8157)	Y010
D_8158 (D_8159)	Y011

注意：

1. 对于高速脉冲输出，它是针对外部高速器件的，要对脉冲进行计数，只能使用高速脉冲输入计数器，不能使用内部计数器，也不能使用 Y 边沿变化进行计数，当作为高速输出时，不能作为普通输出口用；

2. 晶体管的 OFF 时间具有在轻负载时会延长的特性。所以请在需要响应性时设计成让负载在较轻时加负载电阻，增加负载电流；

3. 脉冲累积计数寄存器（D8140~D8158）是重要寄存器，可读写，当写入新值时，计数会在新值基础上进行加减；

4. 由于本指令脉冲输出带方向，所以脉冲累积计数寄存器（D8140~D8158）根据方向计数；

5. 高速脉冲输出时，脉冲累积计数寄存器（D8140~D8158）内的值是不连续的、时刻变化的，用于判断时，请使用大小比较，不要使用等于判断；

6. 加减速时间不要设置太小，否则无加减速效果；

7. 当单位设置为脉冲 pls 时，速度单位为 pls/s，如果平面系统设置的 X 轴 Y 轴脉冲当量不相等，则默认以 X 轴脉冲当量计算其真实速度；

8. 脉冲正在发送标志位，如 M8134 仅代表 Y0 是否正在输出脉冲，如果需要检测平面系统 1 中是否有脉冲正在发送，使用 M8144，运行插补指令时平面系统中任意一轴发送脉冲则该位为 ON；

9. 两点画弧时，当前点与目标点间的直线距离应小于直径；

10. 三点画弧时，三点不要在一条直线上；

11. 圆心画弧时，圆心必须在起点、终点的垂直平分线上；

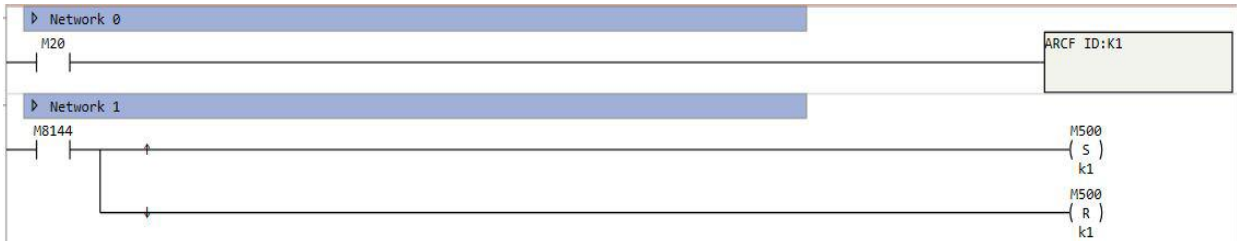
12. 当前点坐标不要与目标点坐标重合；

13. 当脉冲指令使用到输出口 Y 时，该 Y 口不能用做它用，即普通指令无法再对该 Y 口进行 ON、OFF 操作。

实例：

从当前位置，运动到（100mm，100mm）的终点，半径为 100mm，逆时针，劣弧。M20 置位启动圆弧，“圆弧运动”开始时置位 M500，结束后复位 M500。平面系统设置 X 轴为 Y0 ，Y 轴为 Y1，梯形图如下。

梯形图：



4.15.15 BLOCK

对应命令	注释/说明	操作数（寄存器）
BLOCK	图形导入指令	图形名称； 实现/虚线控制位（Y/M）

指令说明：

1）该指令仅 FGM 系列 PLC 与 FGRS_C8X8T 有效，其它型 PLC 该指令无效；
2）FGm_64MT_A 高速脉冲输出口为 Y0～Y7，Y10~Y11，脉冲频率范围 0~500KHz。FGRS_C8X8T 高速脉冲输出口为 Y0~Y3，脉冲频率范围 0~200KHz；
3）该指令作用是在指定的平面系统中，完成导入的图形绘制；
4）目前导入图形仅支持 AutoCAD 中直线和圆弧指令构成的图形，且仅支持另存的 dxf 文件导入。

指令使用说明：

1） 首先在 AutoCAD 软件中，使用直线圆弧指令绘制设计图形，需要注意的是，需要确定平面系统的大小，绘制的图形不可超过平面系统边界。导入图形的绘制方式均为绝对坐标绘制，所以平面系统的原点需与 AutoCAD 中的世界坐标系原点重合，AutoCAD 中的尺寸均为 mm 单位，绘制结束后确认图形中所有线段均为需要绘制的图形，另存为为 2004dxf；
2） 构建平面系统；
3） 图形导入点击路径为：主菜单 –> 脉冲 –> 图形导入；
4） 完成导入后，选择图形所在的平面系统，并设置速度参数；
5） 梯形图中新建 BLOCK 指令，选择需要绘制的图形，并填入控制位，在绘制过程中，有些图形是不能够一笔完成的，所以图元间需要过渡直线，而这种过渡直线是不需要绘制的，称为虚线，需要绘制的称为实线，标志位用于区分实线和虚线，当为虚线时标志位为 0，为实线时标志位为 1；



图 1　图形导入

脉冲正在输出标志(只读)：

寄存器	脉冲输出 Yn
M_8134	Y000
M_8135	Y001
M_8136	Y002
M_8137	Y003
对于 FGM 系列 PLC 还具备以下 M 软元件	
M_8138	Y004
M_8139	Y005
M_8140	Y006
M_8141	Y007
M_8142	Y010
M_8143	Y011

平面系统脉冲正在输出标志(只读)：

对于 FGM 系列 PLC， FGRS_C8X8T 具备以下 M 软元件	
M_8144	平面系统 1
M_8145	平面系统 2
M_8146	平面系统 3
M_8147	平面系统 4
M_8148	平面系统 5

脉冲累积计数(读写)：

寄存器	脉冲输出 Yn
D_8140(D_8141)	Y000
D_8142(D_8143)	Y001
D_8144(D_8145)	Y002
D_8146(D_8147)	Y003
对于 FGM 系列 PLC 还具备以下 M 软元件	
D_8148(D_8149)	Y004
D_8150 (D_8151)	Y005
D_8152(D_8153)	Y006
D_8154(D_8155)	Y007
D_8156 (D_8157)	Y010
D_8158 (D_8159)	Y011

当前脉冲段(只读)：

寄存器	脉冲输出 Yn
D_8124	Y000
D_8125	Y001
D_8126	Y002
D_8127	Y003
对于 FGM 系列 PLC 还具备以下 M 软元件	
D_8128	Y004
D_8129	Y005
D_8130	Y006
D_8131	Y007
D_8132	Y010
D_8133	Y011

出错脉冲段位置，0 表示无错误(只读)：

寄存器	脉冲输出 Yn
D_8108	Y000
D_8109	Y001
D_8110	Y002
D_8111	Y003
对于 FGM 系列 PLC 还具备以下 M 软元件	
D_8112	Y004
D_8113	Y005
D_8114	Y006
D_8115	Y007
D_8116	Y010
D_8117	Y011

注意：

1. 对于高速脉冲输出，它是针对外部高速器件的，要对脉冲进行计数，只能使用高速脉冲输入计数器，不能使用内部计数器，也不能使用 Y 边沿变化进行计数，当作为高速输出时，不能作为普通输出口用；

2. 晶体管的 OFF 时间具有在轻负载时会延长的特性。所以请在需要响应性时设计成让负载在较轻时加

负载电阻，增加负载电流；

3. 脉冲累积计数寄存器（D8140~D8146）是重要寄存器，可读写，当写入新值时，计数会在新值基础上进行加减；

4. 由于本指令脉冲输出带方向，所以脉冲累积计数寄存器（D8140~D8146）根据方向计数；

5. 高速脉冲输出时，脉冲累积计数寄存器（D8140~D8146）内的值是不连续的、时刻变化的，用于判断时，请使用大小比较，不要使用等于判断；

6. 加减速时间不要设置太小，否则无加减速效果；

7. 脉冲正在发送标志位，如 M8134 仅代表 Y0 是否正在输出脉冲，如果需要检测平面系统 1 中是否有脉冲正在发送，使用 M8144，运行插补指令时平面系统中任意一轴发送脉冲则该位为 ON；

8. 速度的数据类型一定是浮点数，与单位设置无关，所以使用 D 寄存器赋值速度进行实时调速时，一定要使用 MOVF 指令赋值速度；

9. 绘制图形时若速度参数不合法，会自动调速，速度根据图形形状有所降低；

10. 导入图形数据大小不要过大，否则无法下载；

11. 当脉冲指令使用到输出口 Y 时，该 Y 口不能用做它用，即普通指令无法再对该 Y 口进行 ON、OFF 操作。

实例：

绘制如下图形，控制位为 Y6。参考图 1 图形导入界面，导入 lufei_A4.dxf 文件，设置平面系统、速度、加减速时间。使用指令 BLOCK 选择该文件，设置实虚控制位 Y6。



梯形图：



4.15.16 HMIBLOCK

对应命令	注释/说明	操作数
HMIBLCOK	HMI 绘图导入指令	参数起始地址

指令说明：

1） 该指令仅 FGM 系列 PLC 与 FGRS_C8X8T 有效，其它型 PLC 该指令无效；

2） 其 FGM 系列 PLC 高速脉冲输出口为 Y0～Y7，Y10~Y11，脉冲频率范围 0~500KHz。FGRS 系列 PLC 脉冲输出口为 Y0~Y3，脉冲频率范围 0~200KHz；

3） 该指令作用是在指定的平面系统中，完成导入的图形绘制；

4） 本指令必须联合 Samkoon 的 SK6.0HMI 一起使用；

5） 目前导入图形仅支持 AutoCAD 中直线和圆弧指令构成的图形，且仅支持另存的 dxf 文件导入。

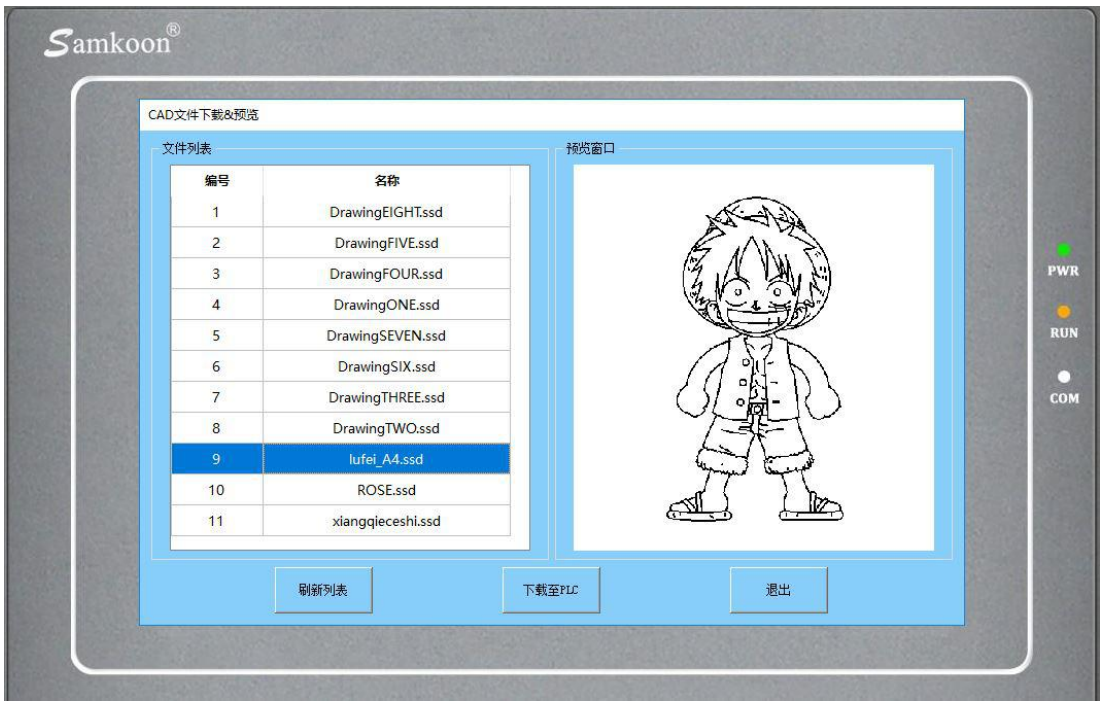
使用说明：

1） 首先在 AutoCAD 软件中，使用直线圆弧指令绘制设计图形，需要注意的是，需要确定平面系统的大小，绘制的图形不可超过平面系统边界。导入图形的绘制方式均为绝对坐标绘制，所以平面系统的原点需与 AutoCAD 中的世界坐标系原点重合，AutoCAD 中的尺寸均为 mm 单位，绘制结束后确认图形中所有线段均为需要绘制的图形，另存为为 2004dxf；

2） 进行图形转换，点击路径为：主菜单 –> 脉冲 –> 图形转换，选择上一步中另存为的.dxf 文件，生成新的.ssd 文件格式，将.ssd 文件考入 U 盘，文件最好在跟目录，在子目录中路径必须为英文路径。转换后的.ssd 单个文件大小不要超过 500KB；

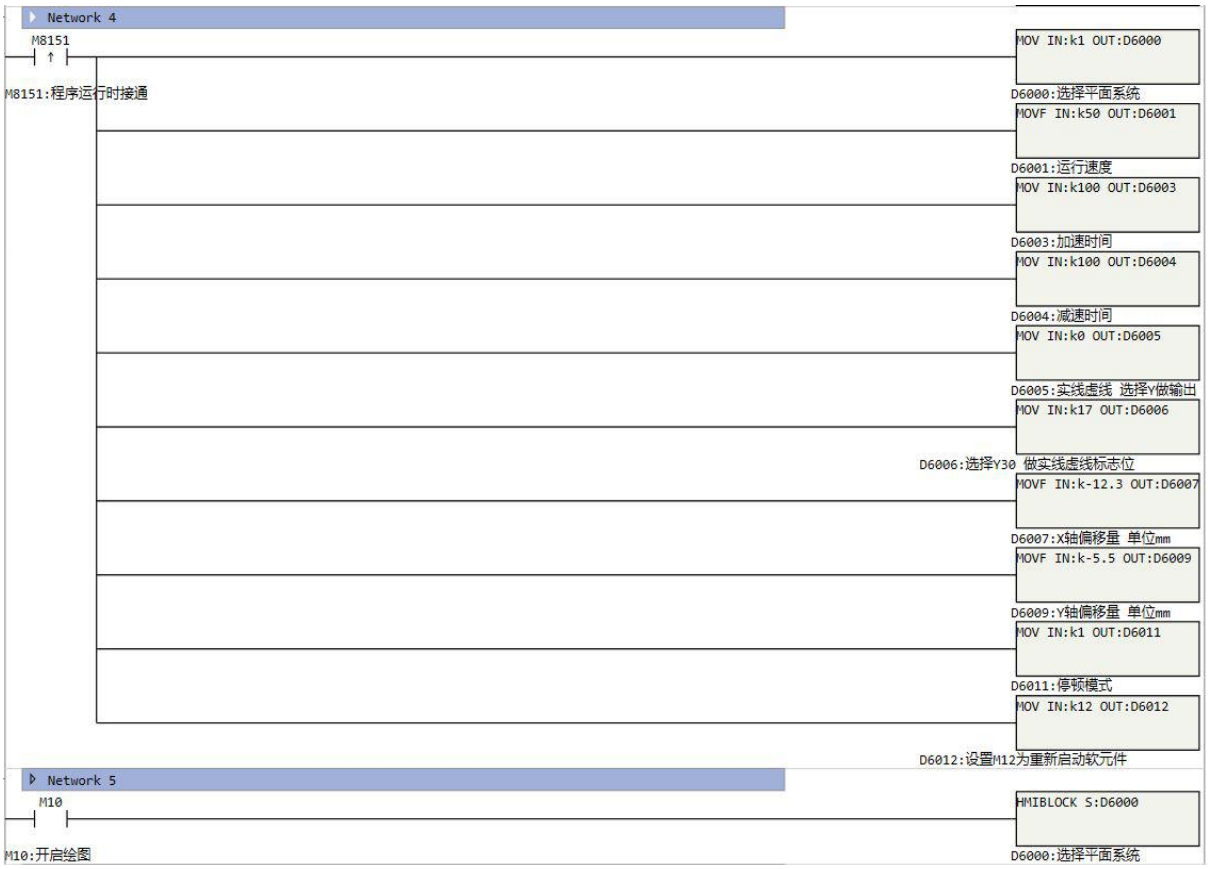
3） 在 SK6.0HMI 中新增‘功能按钮’，选择功能为‘Slave devices IP config’；

4） 将 U 盘插入 HMI 的 U 盘插孔，按下上述功能按钮，进入图形预览界面，如下图。



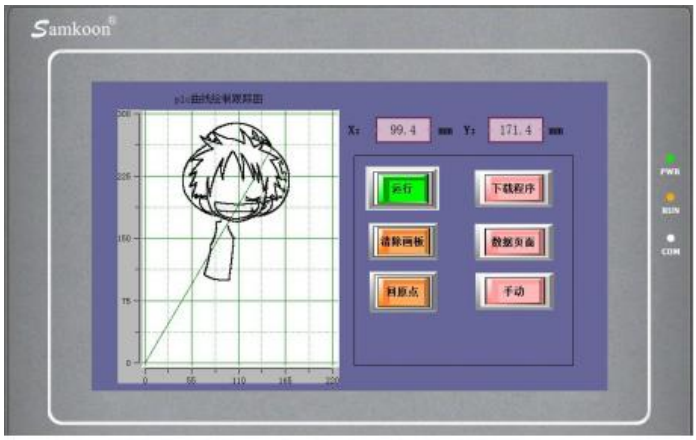
5） 若没有检测到图形，可点击‘刷新列表’按钮，左侧列表中选择需要绘制的图形，点击‘下载至 PLC’，等待进度条下载完成，此时图形已经完成了从 HMI 下载至 PLC；

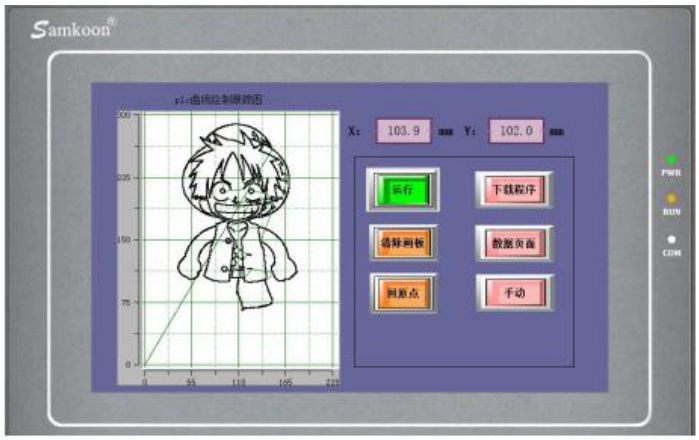
6） PLC 中梯形图如下，其中 D6000 设置参数为平面系统编号，图形会在对应的平面系统中绘图，D6001（D6002）中为绘图速度单位 mm/s，注意 CAD 图形中坐标数据均以 mm 为单位是浮点数，所以速度也为浮点数，**必须使用 MOVF 对速度进行赋值**，D6003 为加速时间，D6004 为减速时间，单位 ms，D6005 为控制位的类型 Y 或 M，写 0 则为 Y，写 1 则为 M，D6006 为控制位序号，若 D6005 写入 0，D6006 写入 10，则表示 Y10 代表控制位状态。控制位含义与 BLOCK 指令中的线条状态相同，代表实线或虚线，实线时控制位 ON，虚线控制位 OFF。D6007（D6008）为整个图形 X 轴偏移量，D6009（D6010）为整个图形 Y 轴偏移量，通过调整该偏移量可以调整图形在坐标系中的位置，参数为浮点数必须使用 MOVF 赋值。D6011 为停顿模式，为零则不停顿，为 1 则在实线虚线交接处停顿。D6012 为停顿后重新启动的软元件，例子中设为实线虚线交接处停顿后，可置位 M12 重新启动绘图；

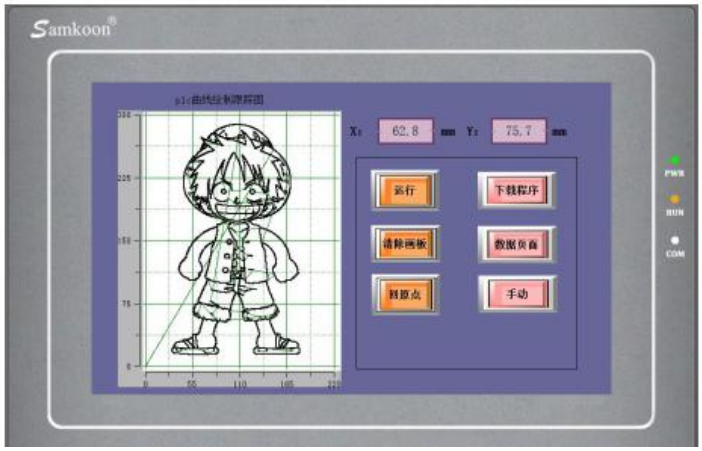


7） 完成上述步骤后，使能 HMIBLOCK 指令，指令便可开始绘图；

8） 若需实现轨迹的实时显示，需要在 PLC 的‘平面系统设置’中，勾选‘启动 HMI 实时显示’并输入数据地址，而 HMI 中需添加‘XY 趋势图’控件，该控件的‘读取地址’必须与 PLC‘平面系统设置’中数据地址设置相同，该控件的其他设置请参考 HMI 帮助文档，如下图是 HMI 实时显示 PLC 绘图轨迹过程；







注意：

1. 对于高速脉冲输出，它是针对外部高速器件的，要对脉冲进行计数，只能使用高速脉冲输入计数器，不能使用内部计数器，也不能使用 Y 边沿变化进行计数，当作为高速输出时，不能作为普通输出口用；

2. 晶体管的 OFF 时间具有在轻负载时会延长的特性。所以请在需要响应性时设计成让负载在较轻时加负载电阻，增加负载电流；

3. 脉冲累积计数寄存器（D8140~D8146）是重要寄存器，可读写，当写入新值时，计数会在新值基础上进行加减；

4. 由于本指令脉冲输出带方向，所以脉冲累积计数寄存器（D8140~D8146）根据方向计数；

5. 高速脉冲输出时，脉冲累积计数寄存器（D8140~D8146）内的值是不连续的、时刻变化的，用于判断时，请使用大小比较，不要使用等于判断；

6. 加减速时间不要设置太小，否则无加减速效果；

7. 脉冲正在发送标志位，如 M8134 仅代表 Y0 是否正在输出脉冲，如果需要检测平面系统 1 中是否有脉冲正在发送，使用 M8144，运行插补指令时平面系统中任意一轴发送脉冲则该位为 ON；

8. HMIBLOCK 指令的速度、XY 轴偏移量必须使用 MOVF 指令赋值；

9. 单个.ssd 文件大小不要超过 500KB；

10. HMIBLOCK 指令不能多个指令同时使用，同一时刻只能使用一个 HMIBLOCK 指令；

11. 使用 HMI 下载一个图形时，会自动覆盖上一个下载的图形。

4.15.17 HMIPLINE

带工艺 CAD 图形导入 HMIPLINE

指令说明：

1） 该指令仅适用于 FGM 系列和 FGRS_C8X8T 型 PLC，其它型 PLC 该指令无效；

2） 其 FGM 系列 PLC 高速脉冲输出口为 Y0～Y7，Y10~Y11，脉冲频率范围 0~500KHz。FGRS 系列 PLC 脉冲输出口为 Y0~Y3，脉冲频率范围 0~200KHz；

3） 该指令作用是在指定的平面系统中，绘制带工艺参数的 CAD 图形，适用于打孔、点胶、点焊等多种场合；

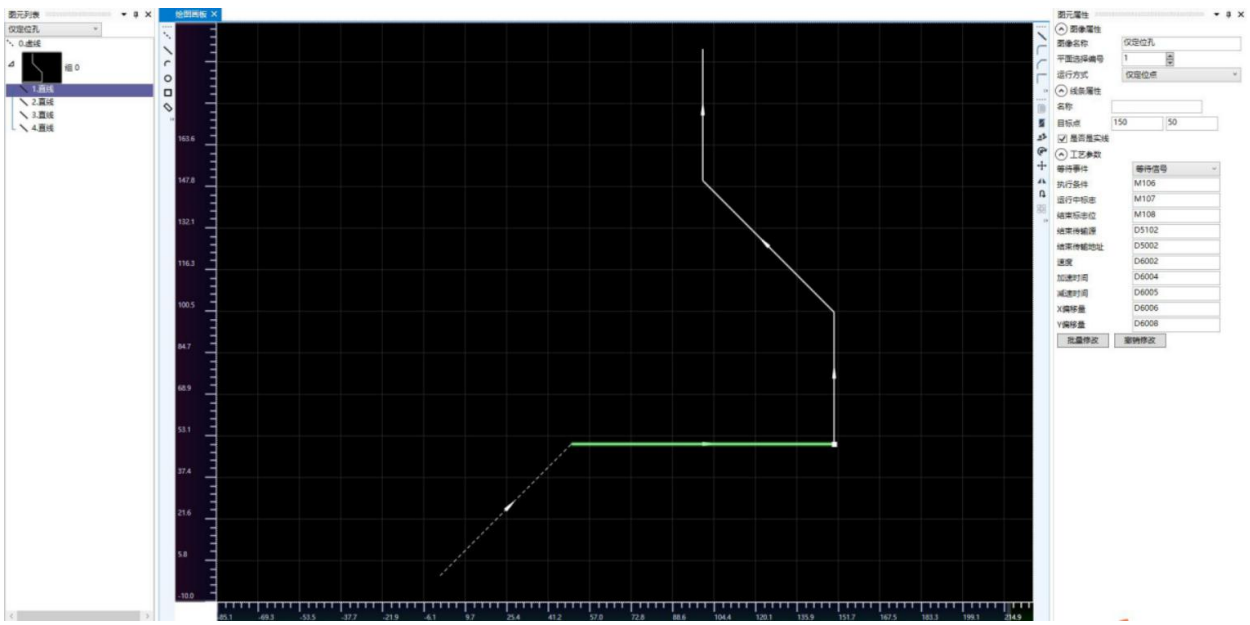
4） 本指令必须联合 Samkoon 的 SK6.0HMI 一起使用；

5） 目前导入图形仅支持 AutoCAD 中直线和圆弧指令构成的图形，且仅支持另存的 dxf 文件导入。

使用说明：

1） 首先在 AutoCAD 软件中，仅能使用直线、圆弧、整圆功能绘制设计图形，并且绘图结束后需要使用‘分解/炸开’功能将图形转化为单一图元。需要注意的是，导入图形的绘制方式均为绝对坐标绘制，平面系统的原点需与 AutoCAD 中的世界坐标系原点重合，AutoCAD 中的尺寸均为 mm 单位，绘图时需要确定平面系统的大小，绘制的图形不可超过实际平面系统边界。当需要实现定位点时，使用直线将所有定位点连接（仅点定位时只能使用直线连接各定位点），然后将连线分解/炸开成单一线段即可，直线段的端点就是需要定位的点。当需要实现特定轨迹的运动时，使用直线、圆弧、整圆功能绘制相应图形，最终分解/炸开成单一图元即可。绘制结束后确认图形中所有线段均为需要绘制的图形，另存为 2004dxf 格式文件；

2）使用显控 CAD 软件 SamCAD 打开另存为的 dxf 文件，具体相关操作可参考 SamCAD 用户手册。如下图所示，为一个仅点位控制的 CAD 图形，图中虚线部分为过渡线，即从原
点运行到第一个点的线段，该线段并非用户绘制，由软件自动连接；



3） 界面右侧图像属性中，图像名称栏可更改图像名称，但需要与最终保存的.ssd 文件名称保持一致。平面选择编号可以选择运行的平面编号，此处设置无用，实际运行的平面系统需在 HMIPLINE 指令中设置。运行方式栏可选择 仅定位点模式与运行轨迹模式，仅定位点模式时，图元仅能是直线，而运行轨迹模式，图元可以是直线、圆弧和整圆。

另外，仅定位点模式可以手动任意切换定位点，在 HMIPLINE 指令参数中有详细说明；

4） 直接在图形中选中相应的线段，或者在左侧列表中选择线段，选中的线段会变成绿色高亮显示，并且右侧图元属性栏中会显示该线段的工艺参数，如下图所示，工艺参数是与 PLC 动作相衔接的参数，每条线段的工艺参数可单独设置，为了控制方便不同图元也可设置为同一个 D 寄存器或 M 元件进行控制，其各参数的含义如下：

等待事件：等待事件分为等待信号、等待时间、脉冲发送完成三种。选择等待信号，当 HMIPLINE 指令处于使能状态时，运行的线段执行条件标志位（图中为 M106）导通时，执行该线段的动作；选择等待时间时，运行的上一个线段完成后，等待相应的时间（图中为 K500，即 500ms），运行该段线段；选择脉冲发送完成，即上段线段运行完成之后，马上运行该线段，图中设置的执行条件栏无效。

执行条件：分别对应不同的等待事件，设置不同的内容，等待事件为等待信号时执行条件仅能为 M 元件，M 导通即为满足执行条件，需要注意的是，执行条件由用户梯形图置位，也需要由用户梯形图复位，若多条线段设置了相同的执行条件，那么避免运行错误需要在下一线段运行前，复位执行条件。等待事件为 ATC 时间时执行条件为时间，可以设置为 K 常数，也可以设置为 D 寄存器动态更改。等待事件为脉冲发送完成时，执行条件无效，可写入 K0。

运行中标志：当 PLC 正在运行该段线段时，运行中标志位会被 PLC 置位，该线段运行结束后，PLC 会复位该标志。需要注意的是，PLC 会自动置位、复位该标志位，用户不需要对其进行梯形图控制，仅作为条件判断即可。

结束标志位：当该线段运行结束时，该标志位会被 PLC 置位。该标志位可作为该点动作的触发位，例如，在该点需要打孔，当结束标志位置位时，则系统运行到指定位置，可由该位触发打孔动作。需要注意的是，该位由 PLC 置位，需要用户使用梯形图对其复位。

结束传输源，结束传输地址：当该线段运行结束时，将结束传输源中的值赋值到结束传输地址中，例如在点焊设备中，不同点位可能对应不同焊接角度，可将结束传输源设置为对应角度，作为重要运行参数。结束传输源可设置为 K 常数，该常数为浮点数，使用时需注意，也可以设置为 D 寄存器。结束传输地址仅能设置为 D 寄存器。

速度：该段线段运行的速度，以 mm 为单位，为浮点数，可以设置为 K 常数，或者 D 寄存器动态更改，为了方便控制，经常将不同的线段设置相同的速度参数。

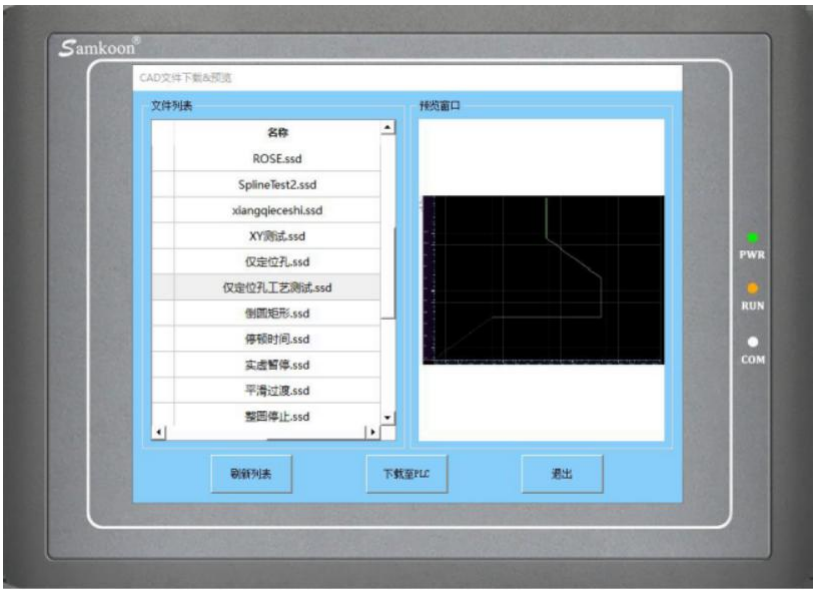
加速时间、减速时间：该段线段运行时的加速时间和减速时间，是单字整数，可设置为 K 常数，或者 D 寄存器。

X、Y 偏移量：由于 CAD 软件绘制的图形均为理论图形，而实际使用中由于设备装配、夹具装夹等因素会存在一定偏差，该偏差可由 X、Y 偏移量来进行修正，该偏移量是修正线段的终点坐标，每条线段均可单独设置偏移量，该参数以 mm 为单位，为浮点数，可以设置为 K 常数，或者 D 寄存器，当所有点设置相同的偏移量时，可视为图形的整体偏移。当运行方式栏选择仅定位点时，不同图元偏移量可以设置为不同值，当运行方式选择运行轨迹时，不同图元偏移量必须设置为相同值，此时仅能整体偏移。



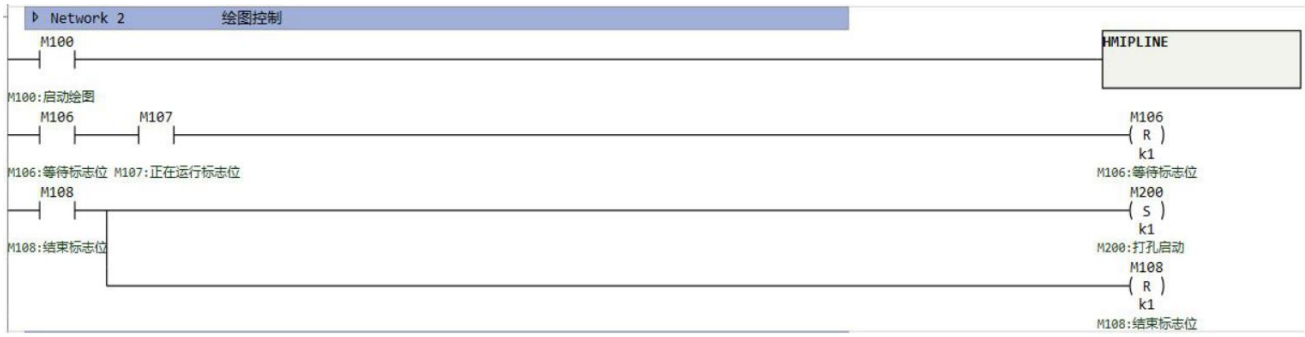
5） 在 SK6.0HMI 中下载制定底层，新增‘功能按钮’，选择功能为‘Slave devices IP config’；

6） 将 U 盘插入 HMI 的 U 盘插孔，按下上述功能按钮，进入图形预览界面，如下图。



7）若没有检测到图形，可点击‘刷新列表’按钮，左侧列表中选择需要绘制的图形，点击‘下载至 PLC’，等待进度条下载完成，此时图形已经完成了从 HMI 下载至 PLC；

8）PLC 中编辑如下指令，双击 HMIPLINE 指令，显示如图参数编辑窗口，其中各参数含义如下：





平面编号：选择图形要运行的平面系统，关于平面系统的设置可参考帮助文档，平面系统设置；

手动、自动模式切换位：在仅点位控制时，可选择自动控制和手动控制，当该切换位为 OFF 时，为自动控制，使能 HMIPLINE 指令，则按照工艺参数设置，运行图形。若该切换位为 ON 时，为手动控制，此时图形的工艺参数均无效，需要通过手动上一点、手动下一点标志位点动切换上一点、下一点。

手动上、下一点位：在仅点位控制时的手动模式下，可手动切换上一点和下一点。

终点坐标显示地址，保存终点坐标文件位、导出选中文件位：暂时不支持相应功能，可任意设置；

运动至点：在仅点位控制时，除了上一点、下一点外，可通过该 D 寄存器，设置指定点位，在手动模式时点击下一点位，即可运行至设置的任意点位。在自动模式时，设置指定点位，指令使能后则直接从设置点位开始运行。

选中文件地址：选中 PLC 中已保存的图形作为要运行的图形，PLC 中可保存 20 个图形文件，用户可根据该寄存器选择要运行文件的编号。

文件信息输出首地址：选择一片 D 寄存器地址，作为 PLC 中所存储的图形文件名和序号的存储地址，其格式如下图所示；

当前文件个数D		
D + 10	编号 D+10	名称 D+11
D + 30	编号 D+30	名称 D+31
D + 50	编号 D+50	名称 D+51
D + 70	编号 D+70	名称 D+71
……		

刷新文件列表，该位被置位时，刷新显示 PLC 中现有文件列表。
清空所有文件位，该位被置位时，删除 PLC 中所有文件。
删除选中文件位，该位被置位时，仅删除选中文件。
结束标志位，当要运行的图形中所有图元都运行完成时，该位会被 PLC 置位，需要用户梯形图程序复位。

9） 若需实现轨迹的实时显示，需要在 PLC 的‘平面系统设置’中，勾选‘启动 HMI 实时显示’并输入数据地址，而 HMI 中需添加‘XY 趋势图’控件，该控件的‘读取
地址’必须与 PLC‘平面系统设置’中数据地址设置相同，该控件的其他设置请参考 HMI 帮助文档；

注意：
1. 对于高速脉冲输出，它是针对外部高速器件的，要对脉冲进行计数，只能使用高速脉冲输入计数器，不能使用内部计数器，也不能使用 Y 边沿变化进行计数，当作为高速
输出时，不能作为普通输出口用；
2. 晶体管的 OFF 时间具有在轻负载时会延长的特性。所以请在需要响应性时设计成让负载在较轻时加负载电阻，增加负载电流；
3. 脉冲累积计数寄存器（D8140~D8146）是重要寄存器，可读写，当写入新值时，计数会在新值基础上进行加减；
4. 由于本指令脉冲输出带方向，所以脉冲累积计数寄存器（D8140~D8146）根据方向计数；
5. 高速脉冲输出时，脉冲累积计数寄存器（D8140~D8146）内的值是不连续的、时刻变化的，用于判断时，请使用大小比较，不要使用等于判断；
6. 加减速时间不要设置太小，否则无加减速效果；
7. 脉冲正在发送标志位，如 M8134 仅代表 Y0 是否正在输出脉冲，如果需要检测平面系统 1 中是否有脉冲正在发送，使用 M8144，运行插补指令时平面系统中任意一轴发送脉
冲则该位为 ON；
8. HMIPLINE 指令的速度、XY 轴偏移量必须使用 MOVF 指令赋值；
9. 单个.ssd 文件大小不要超过 32KB；
10. HMIPLINE 指令不能多个指令同时使用，同一时刻只能使用一个 HMIPLINE 指令；
11. 使用 HMI 下载的图形超过 20 时，会自动覆盖最后一个图形。
12. 如果多脉冲脉冲频率超过 200K，需要在脉冲输出口加入上拉电阻保证脉冲波形不失真，上拉电压 24V, 上拉电阻推荐 1K。

4.15.18 PAUSE

PAUSE 平面暂停

功能说明：

1）该指令仅针对 POLYLINE 和 BLOCK、HMIBLOCK 指令使用；

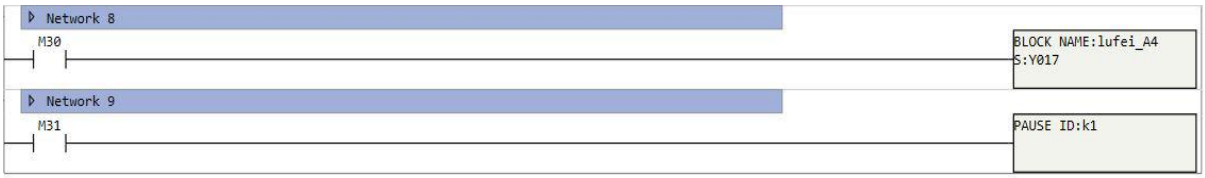
2）该功能是在 POLYLINE 或 BLOCK、HMIBLOCK 指令正在绘图过程中，使能 PAUSE 会暂停对应平面系统的脉冲发送，失能 PAUSE 后会继续绘图，暂停前后不影响绘图轨迹；

使用说明：

1） 首先设置 POLYLINEI/POLYLINEF 指令或 BLOCK、HMIBLOCK 指令的相关参数；

2） 设置暂停指令中的平面系统与 POLYLINE 和 BLOCK、HMIBLOCK 中的平面系统相同；

3） 使能 POLYLINE 和 BLOCK、HMIBLOCK 指令，指令按照设置绘制图形，当需要暂停时，使能 PAUSE 指令，图形绘制会减速停止，减速时间与插补指令设置的减速时间相同。当需要继续绘制时，失能 PAUSE 指令即可，暂停的过程不会影响绘制的轨迹，依然按照原来的设置轨迹绘制。在使用 PAUSE 停止脉冲发送后，不能断开 POLYLINE 和 BLOCK、HMIBLOCK 指令的使能，整个暂停过程都不能断开脉冲指令的使能；



PAUSE 指令使用实例

注意：

1. **暂停有时并不是立刻响应的，POLYLINE 和 BLOCK、HMIBLOCK 绘图本身在加减速过程中，不进行暂停处理，但一达到稳定速度就会暂停（减速停）处理；**
2. **在已经暂停后，不要失能 POLYLINE 或 BLOCK、HMIBLOCK 指令，否则再次使能插补指令时，并非在暂停处开始绘制，而是按照插补指令设置重新绘制；**
3. **暂停后插补指令仍然会占用脉冲和方向端口，暂停后的停止状态是无法移动的，也是不允许移动的；**

4.15.19 ZRNR

任意点回归原点 ZRNR

对应命令	注释/说明	操作数（寄存器）
ZRNR（32 位）	任意点回归原点	回归速度（K/H/D） 爬行速度（K/H/D） 输出脉冲（Y） 方向输出口（Y）

指令说明：

启用输入端接为“接通”时，任意点回零指令 ZRNR 会根据此时工作台所处的位置以及所选择的默认回零方向与回零模式，运行到指定的零点信号范围特定一侧后停止发送脉冲。启用输入端接为“断开”时，脉冲发送停止。

脉冲正在输出标志(只读)：

寄存器	脉冲输出 Yn
M_8134	Y000
M_8135	Y001
M_8136	Y002
M_8137	Y003
对于多脉冲型 PLC FGm_32MT_A FGm_64MT_A 还具备以下 M 软元件	
M_8138	Y004
M_8139	Y005
M_8140	Y006
M_8141	Y007
M_8142	Y010
M_8143	Y011

脉冲累积计数(读写)：

寄存器	脉冲输出 Yn
D_8140(D_8141)	Y000
D_8142(D_8143)	Y001
D_8144(D_8145)	Y002
D_8146(D_8147)	Y003
对于多脉冲型 PLC FGm_32MT_A FGm_64MT_A 还具备以下寄存器	
D_8148(D_8149)	Y004
D_8150 (D_8151)	Y005

D_8152(D_8153)	Y006
D_8154(D_8155)	Y007
D_8156 (D_8157)	Y010
D_8158 (D_8159)	Y011

原点回归加减速时间（读写）：

寄存器	脉冲输出 Yn
D_8076	Y000
D_8077	Y001
D_8078	Y002
D_8079	Y003
对于多脉冲型 PLC FGm_32MT_A FGm_64MT_A 还具备以下寄存器	
D_8080	Y004
D_8081	Y005
D_8082	Y006
D_8083	Y007
D_8084	Y010
D_8085	Y011

指令使用说明：如下图所示。



1. 爬行速度与回归速度参数类型可为 D/K/H，均为 32 位，两个速度同时输入负值，等同于改变默认回零方向，但注意这会改变零点信号位置；
2. 设定脉冲输出端口 Y 与脉冲方向输出端口 Y；
3. 原点信号输入端与正负极限信号输入端接相应的原点传感器、正负极限传感器的信号输出（信号输入端口可选择 X 端口或 M 寄存器），其中正负极限以及 z 相信号是可选项；
4. 原点信号、正负极限信号的极性根据实际使用的传感器信号极性（常开或常闭）来选择；
5. 默认回零方向决定了大部分情况下（位于极限信号内时除外）工作台从初始位置开始回零时的运动方向；
6. 回零模式分为近点信号和原点上升沿两种模式，两种模式回到零点的路径和具体位置不同（具体见回零路径介绍）；
7. 回零模式和默认的回零方向共同决定了零点信号的位置（处于零点信号范围的哪一侧）；
8. 使用 ZRNR 指令之前需要依次确认以下几个方面（以下具体做法参考注意事项）：
   1) 确认正负方向以及正负极限的位置；
   2) 根据实际情况确认默认回零方向；
   3) 选择合适的回零模式；

4) 根据所选的默认回零方向以及回零模式，确定零点信号的位置；

注意：

1. 使用该指令完成回原点动作后，对应输出口的脉冲累积寄存器（D8140~D8147）会自动清零，作为原点；

2. 当脉冲指令使用到输出口 Y 时，该 Y 口不能用做它用，即普通指令无法再对该 Y 口进行 ON、OFF 操作；

3. 各个传感器接入的 X 端口滤波模式应设为不启用，否则终点位置会引起较大的偏差；

4. 正方向、正极限：脉冲方向口为高电平时工作台运动（电机旋转）方向为正方向，相应的极限位置为正极限；反之为负方向，负极限；

5. 回零过程中脉冲方向口输出由 PLC 根据工作台所处的位置自动控制，无需用户控制（确保脉冲方向口不被其他指令所影响）；

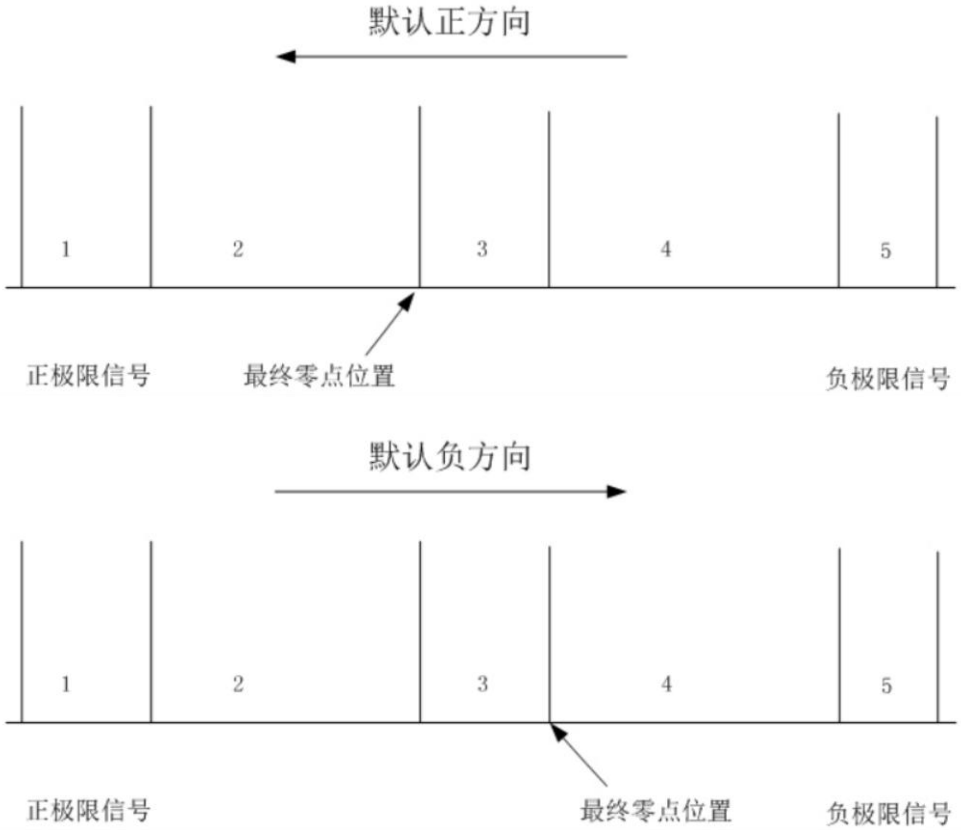
6. 若默认回零方向选择正方向，大部分情况下（初始位置位于极限信号内除外）开始回零时脉冲方向口输出高，工作台正向运动；若选择负方向，情况完全相反。用户可以根据实际情况选择合适的默认回零方向（例如考虑工作台回零开始时初始位置一般所处的区域）；

7. 如下图所示，用户在使用前需要明确零点的位置：

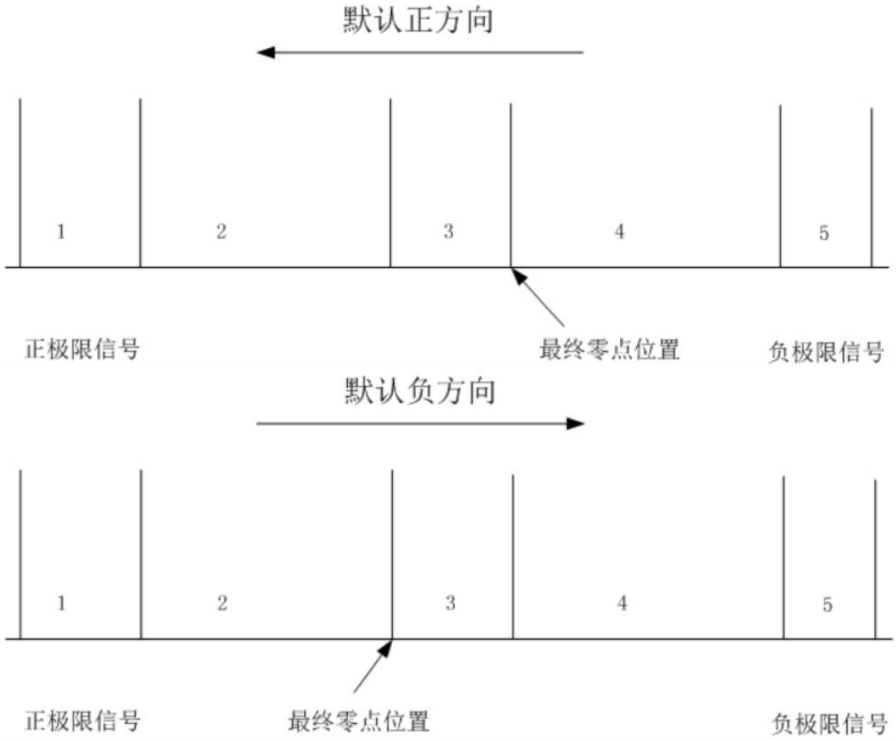
1）近点信号模式：若默认回零方向选择正方向，最终零点位置为原点传感器信号范围中靠近正极限的一侧边沿；若选择负方向，最终零点位置为原点传感器信号范围中靠近负极限的一侧边沿。

2）原点上升沿模式：若默认回零方向选择正方向，最终零点位置为原点传感器信号范围中靠近负极限的一侧边沿；若选择负方向，最终零点位置为原点传感器信号范围中靠近正极限的一侧边沿。

近点信号模式下的最终零点位置：



原点上升沿模式下的最终零点位置：



8. 正负极限信号与 Z 信号均为可选项，如果不选择正负极限信号或其中一个极限信号，则不能实现任意点回零（下述回零路径中涉及到未勾选极限信号的情况均不能实现，其余未涉及到该未勾选信号的路径依旧可以实现）。Z 相信号用户可以根据需要选择（伺服上的 z 相脉冲可接入到 PLC 上，但注意一定要是 24V 的信号，如果不是要经过转换，否则 PLC 无法识别）。
9. 若工作台在回零过程中进入正负极限后停止运动且报错：原点信号丢失，请检查原点传感器信号是否失效。确认排除故障后，重新使能回零指令仍然可以回零。若工作台在回零过程中停止并报错：加减速距离过长，则表明原点上升沿模式减速到 0 的过程中触碰到了极限信号或者工作台完全超出了极限信号，此时请检查工作台位置并且降低加减速时间。
10. 若工作台在原点传感器内无法减速到爬行速度，请适当加宽原点传感器信号范围或者减小加减速时间，使得工作台能够减速到爬行速度。
11. 为确保安全，请保证正负极限有足够的宽度能够使工作台在其信号内部减速到 0，否则超出工作台极限会引起上述加减速距离过长的报错。
12. 为确保安全，建议正负极限信号极性选择为常闭（防止因正负极限失效引起事故）。

指令的有效操作数：

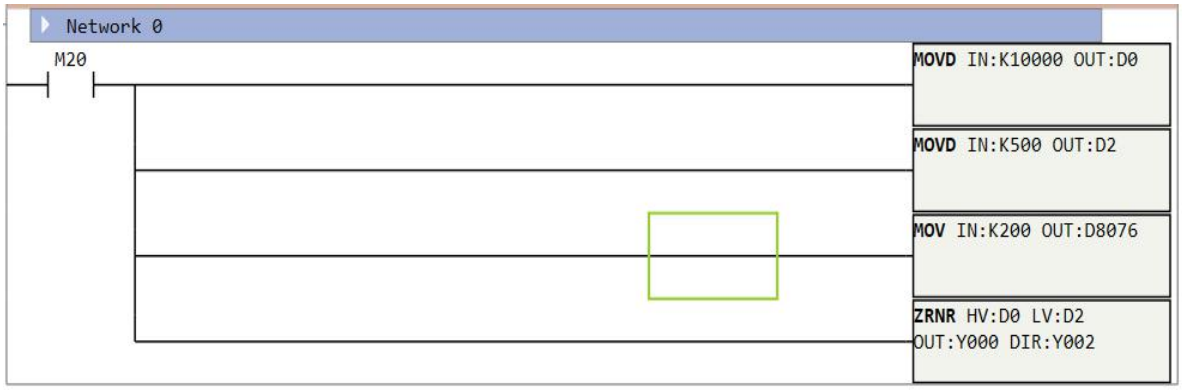
输入输出	类型	操作数	描述
EN	BOOL		启用输入端接通时，指令执行；否则，不执行
回归速度	双整数	K/H/D	回归速度设定
爬行速度	双整数	K/H/D	爬行速度设定
输出脉冲 Y	BOOL	Y	指定的脉冲输出口
输出方向 Y	BOOL	Y	指定的脉冲方向输出口
原点信号输入端	BOOL	X/M	指定的原点输入信号
正极限信号输入端	BOOL	X/M	指定的正极限信号输入端
负极限信号输入端	BOOL	X/M	指定的负极限信号输入端
Z 信号输入端	BOOL	X/M	指定的 Z 相信号输入端

实例：

指令表：

```
Network 000
    LD          M20
    MOVD        K10000    D0          //回归速度，脉冲频率 10000Hz
    MOVD        K5000     D2          //爬行速度，脉冲频率 5000Hz
    MOV         K200      D8076       //减速时间 200ms
    ZRNR        D0   D2   Y0   Y2     //当 M20 为‘1’时，根据初始位置和默认回零方向，自动选择合适的回零模式进行回零。
```

梯形图：



脉冲参数表：

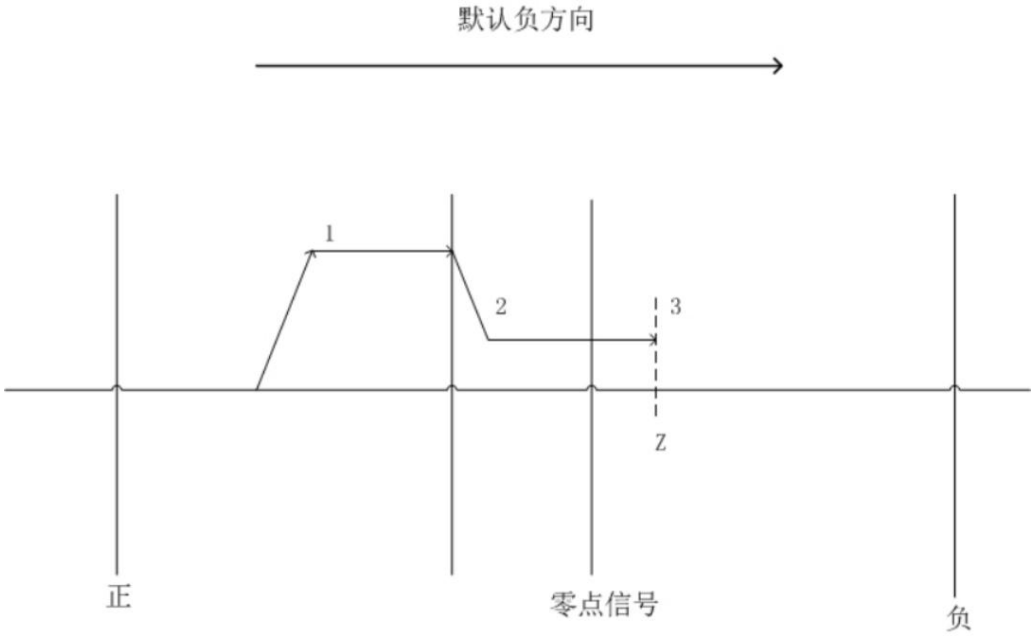


回零示意图：

以默认回零方向为负方向为例，说明工作台处于正负极限间的任意位置时的两种不同模式下的回零路径：

1) 初始位置位于正极限和原点信号之间

a) 近点信号模式



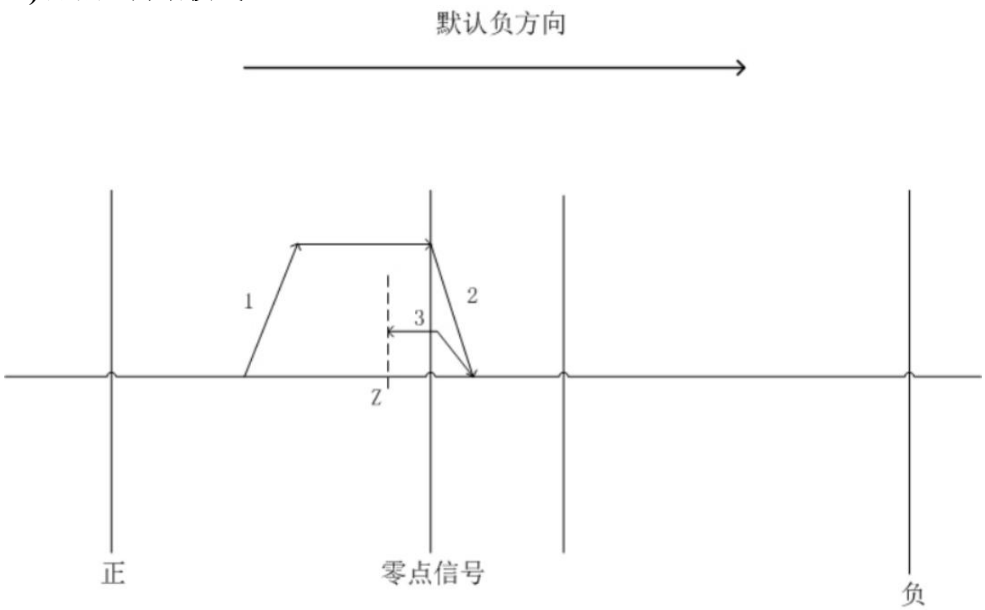
如上图所示：

①工作台负方向加速到回归速度；

②进入原点信号后开始负方向减速达到爬行速度；

③到达零点信号侧时停止，如果选择了 Z 相信号，则在 Z 相信号处停止；

b) 原点上升沿模式



如上图所示：

①工作台负方向加速到回归速度；

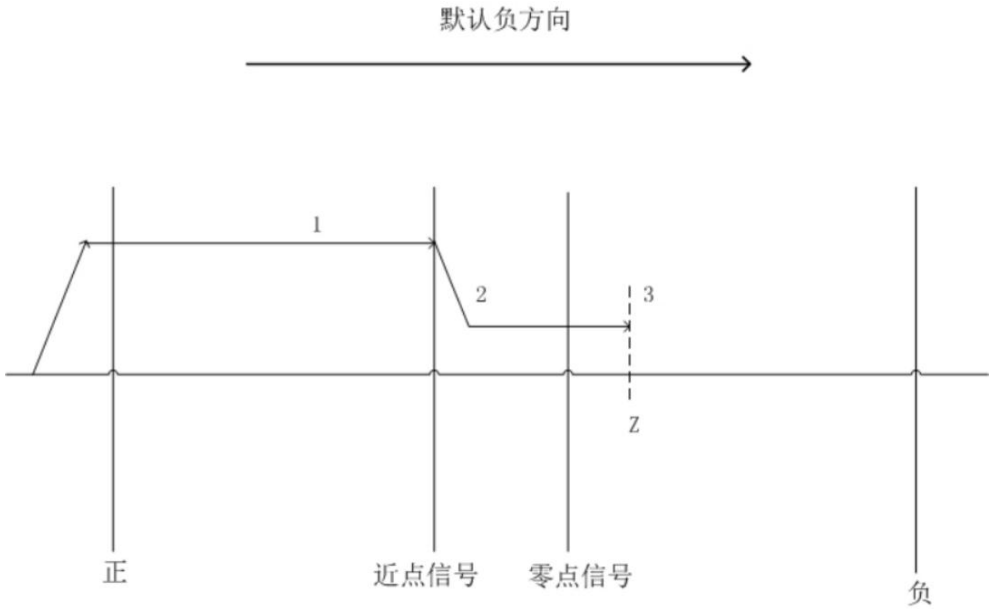
②进入原点信号后开始负方向减速到 0；

③反向加速到爬行速度；

④到达零点信号侧时停止，如果选择了 Z 相信号，则在 Z 相信号处停止；

2) 初始位置位于负极限信号内

a) 近点信号模式



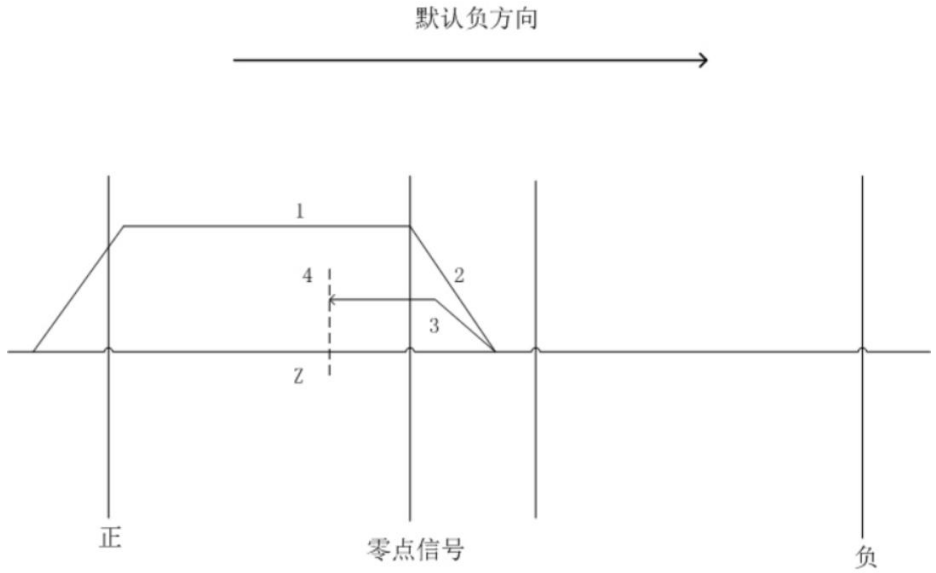
如上图所示：

①工作台负方向加速到回归速度；

②进入原点信号后开始负方向减速达到爬行速度；

③到达零点信号侧时停止，如果选择了 Z 相信号，则在 Z 相信号处停止；

b) 原点上升沿模式



如上图所示：

①工作台负方向加速到回归速度；

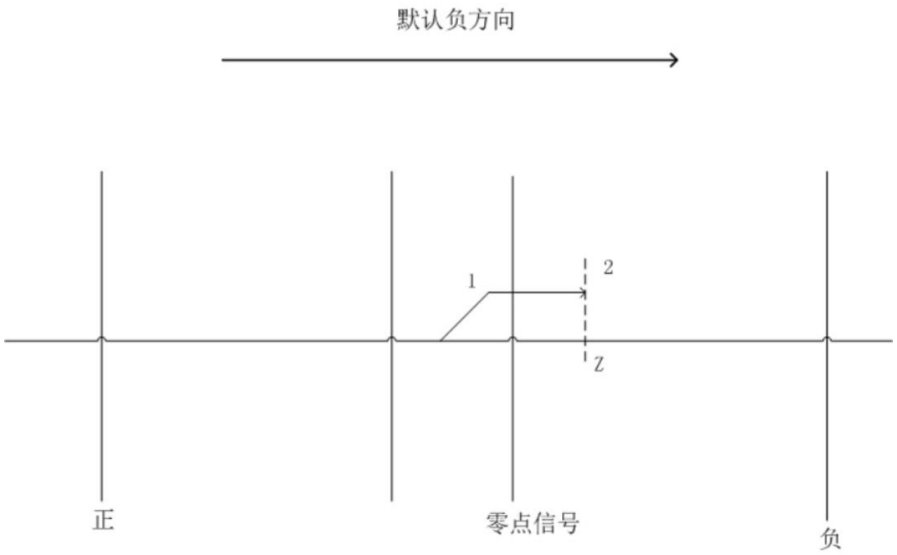
②进入原点信号后开始负方向减速到 0；

③反向加速到爬行速度；

④到达零点信号侧时停止，如果选择了 Z 相信号，则在 Z 相信号处停止；

3) 初始位置位于原点信号内部

a) 近点信号模式

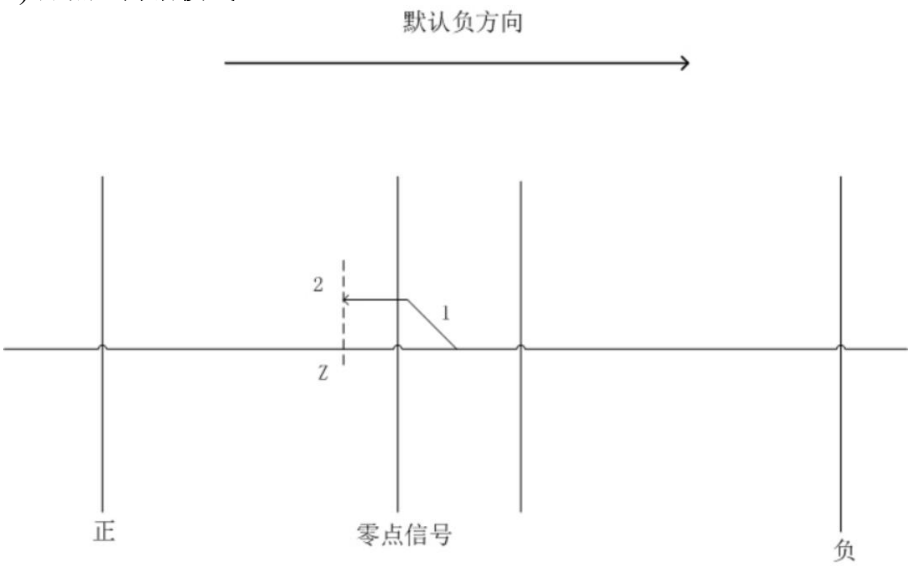


如上图所示：

①工作台负方向加速到回归速度；

②到达零点信号侧时停止，如果选择了 Z 相信号，则在 Z 相信号处停止；

b) 原点上升沿模式



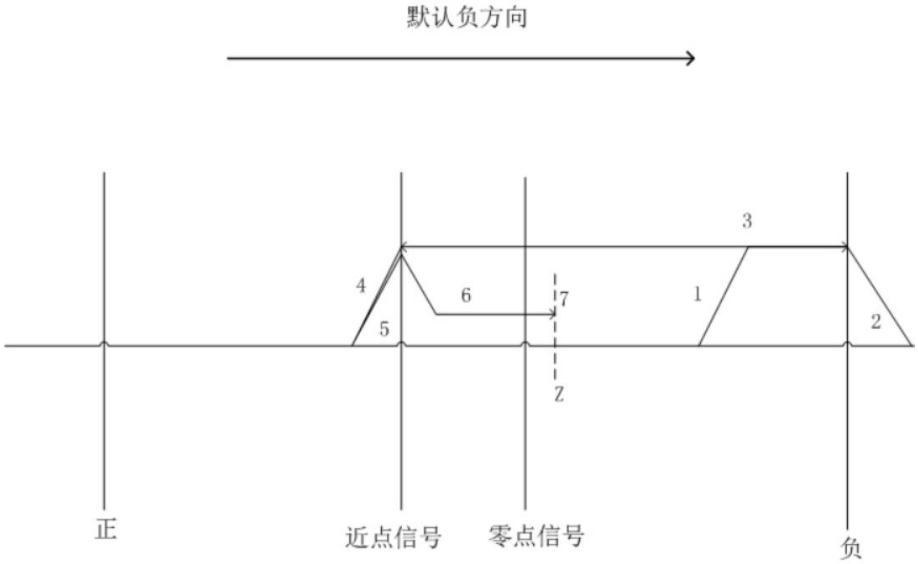
如上图所示：

①工作台正方向加速到回归速度；

②到达零点信号侧时停止，如果选择了 Z 相信号，则在 Z 相信号处停止；

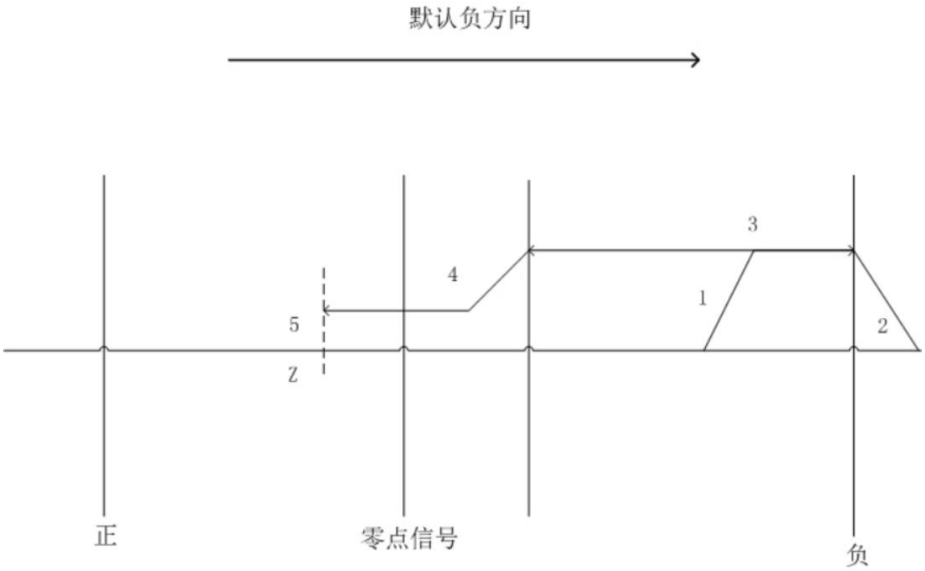
4) 初始位置位于原点信号与负极限之间

a) 近点信号模式



如上图所示：
①工作台负方向加速到回归速度；
②当进入负极限信号时立即减速到 0；
③减速到 0 后立即开始正方向加速到回归速度；
④当离开原点信号范围时，立即开始正方向减速到 0；
⑤减速到 0 后立即开始负方向加速到回归速度；
⑥当进入原点信号范围时，立即开始负方向减速到爬行速度；
⑦到达零点信号侧时停止，如果选择了 Z 相信号，则在 Z 相信号处停止；

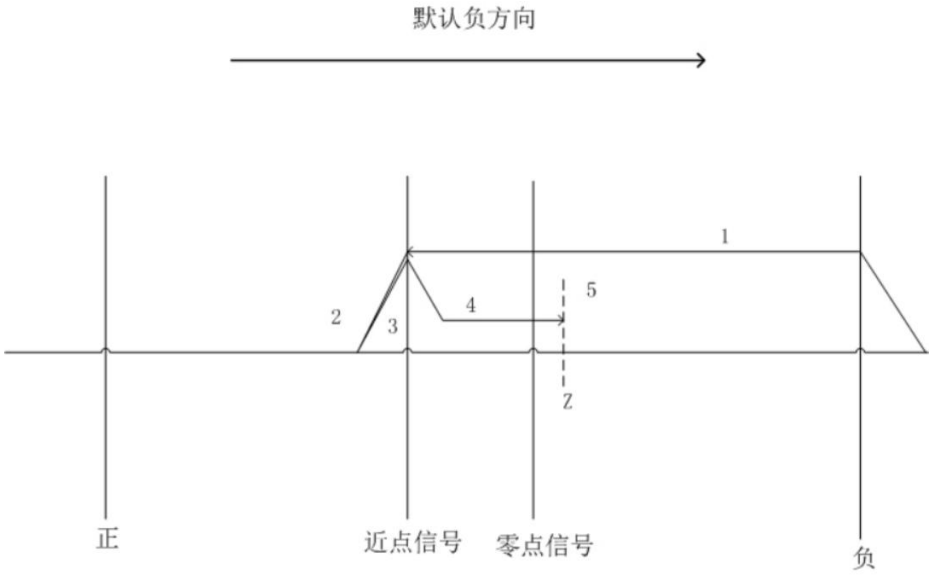
b) 原点上升沿模式



如上图所示：
①工作台负方向加速到回归速度；
②当进入负极限信号时立即减速到 0；
③减速到 0 后立即开始正方向加速到回归速度；
④当进入原点信号范围时，立即开始正方向减速到爬行速度；
⑤到达零点信号侧时停止，如果选择了 Z 相信号，则在 Z 相信号处停止；

5) 初始位置位于负极限信号内

a) 近点信号模式



如上图所示：

①工作台正方向加速到回归速度；

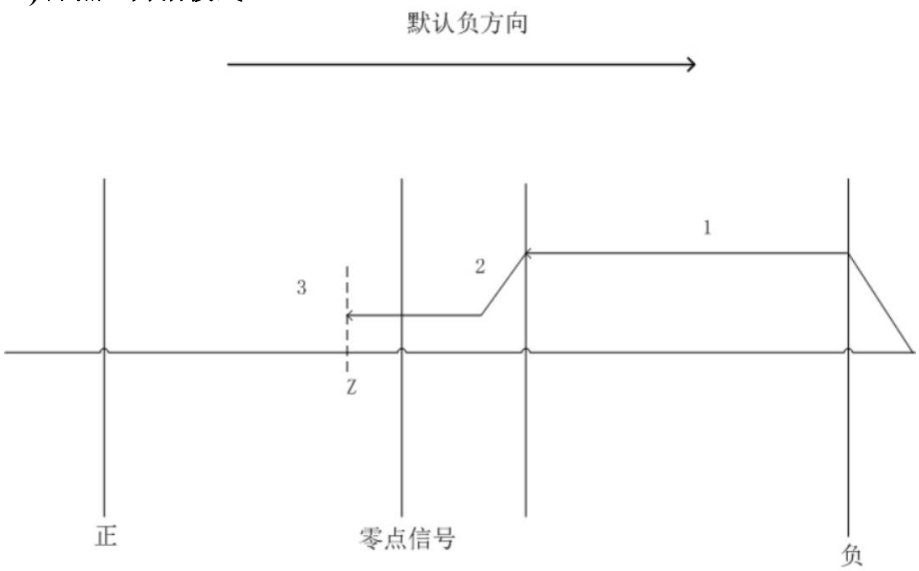
②当离开原点信号范围时，立即开始正方向减速到 0；

③减速到 0 后立即开始负方向加速到回归速度；

④当进入原点信号范围时，立即开始负方向减速到爬行速度；

⑤到达零点信号侧时停止，如果选择了 Z 相信号，则在 Z 相信号处停止；

b) 原点上升沿模式



如上图所示：

①工作台正方向加速到回归速度；

②当进入原点信号范围时，立即开始正方向减速到爬行速度；

③到达零点信号侧时停止，如果选择了 Z 相信号，则在 Z 相信号处停止；；

回零误差分析：

1) 到达最终零点位置时的运行速度（速度越小，偏差越小）；
2) PLC 扫描周期的长短（扫描周期越短，偏差越小）；
3) 运动部分的运动惯量大小（运动惯量越小，偏差越小）；

4.15.20 JOG

点动指令 JOG

对应命令	注释/说明	操作数（寄存器）
JOG (32 位)	点动指令	输出频率(K/H/D) 加减速时间(K/H/D) 输出脉冲 Y 输出方向 Y

指令说明

1）点动 JOG 指令是可以设置频率和加减速时间的带方向点动脉冲指令。启用输入端为“接通”时，脉冲从 1000HZ 匀加速到设置频率匀速输出，输入端“断开”时，减速从设置频率减速到 0。当输入端断开时，脉冲输出并不是立即停止有一个减速过程。

2）JOG 指令可输出频率范围为 0Hz~200KHz。对于 FGM 系列 PLC 其最大脉冲频率为 500KHz。

3）FGs 提供了高速脉冲输出口，16/32 点晶体管型的 PLC，Y0~Y1 为高速脉冲输出口；64 点晶体管型的 PLC，Y0~Y3 为高速脉冲输出口，FGRB 晶体管 PLC，Y0~Y1 为高速脉冲输出口，FGRS 晶体管 PLC，Y0~Y3 为高速脉冲输出口；继电器型的 PLC 不具有高速脉冲功能，输出端口的最大输出频率由继电器的吸合时间决定。对于 FGM 系列 PLC 其高速脉冲输出口为 Y0~Y7，Y10~Y11。

4）指令为对称（加减速时间相同）的梯形、S 形位置控制，当特殊功能寄存器 M8069 为 ON 时所有的脉冲指令的加减速为 S 形，为 OFF 时则为梯形。

5）特殊功能寄存器 M8068 为方向极性控制，默认脉冲方向输出口为 ON 时表示正方向，OFF 时表示负方向。若设置 M8068 为 ON 则所有脉冲指令的方向输出极性发生反转，脉冲方向输出为 OFF 表示正方向，为 ON 表示负方向。

脉冲正在输出标志(只读)：

寄存器	脉冲输出 Yn
M_8134	Y000
M_8135	Y001
M_8136	Y002
M_8137	Y003
对于 FGM 型 PLC 还具备以下 M 软元件	
M_8138	Y004
M_8139	Y005
M_8140	Y006
M_8141	Y007
M_8142	Y010
M_8143	Y011

脉冲累积计数(读写)：

寄存器	脉冲输出 Yn
D_8140(D_8141)	Y000
D_8142(D_8143)	Y001
D_8144(D_8145)	Y002
D_8146(D_8147)	Y003
对于 FGM 型 PLC 还具备以下寄存器	
D_8148(D_8149)	Y004
D_8150(D_8151)	Y005
D_8152(D_8153)	Y006
D_8154(D_8155)	Y007
D_8156(D_8157)	Y010
D_8158(D_8159)	Y011

注意：

1. 对于高速脉冲输出，它是针对外部高速器件的，要对脉冲进行计数，只能使用高速脉冲输入计数器，不能使用内部计数器，也不能使用 Y 边沿变化进行计数，当作为高速输出时，不能作为普通输出口用；
2. 晶体管的 OFF 时间具有在轻负载时会延长的特性。所以请在需要响应性时设计成让负载在较轻时加负载电阻，增加负载电流；
3. 对于脉冲输出 Y0，寄存器 D8140（D8141）中累计其脉冲个数，D8141 存储高 16 位，D8140 存储低 16 位；
4. 脉冲累积计数寄存器（D8140~D8158）是重要寄存器，可读写，当写入新值时，计数会在新值基础上进行加减；
5. 由于本指令脉冲输出带方向，所以脉冲累积计数寄存器（D8140~D8158）根据方向计数；
6. 高速脉冲输出时，脉冲累积计数寄存器（D8140~D8158）内的值是不连续的、时刻变化的，用于判断时，请使用大小比较，不要使用等于判断；
7. 当脉冲指令使用到输出口 Y 时，该 Y 口不能用做它用，即普通指令无法再对该 Y 口进行 ON、OFF 操作。

指令的有效操作数

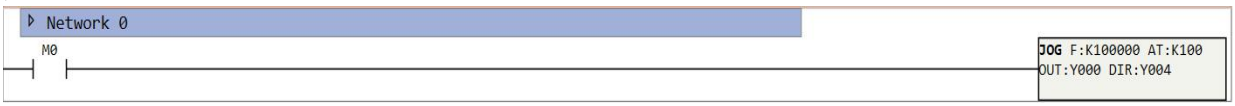
输入输出	类型	操作数	描述
EN	BOOL		启用输入端接通时，指令执行；否则，不执行
输出频率	双整数	K/H/D	输出频率的设定值
加减速时间	整数	K/H/D	加减速时间设定值
输出脉冲 Y	BOOL	Y	指定的脉冲输出口
输出方向 Y	BOOL	Y	指定的脉冲方向输出口

实例

指令表：

```
Network 000
    LD      M0
    JOG     K100000   K100   Y0   Y4   // 当 M0 被置位 JOG 开始工作
```

梯形图：



4.15.21 FOLLOW

对应命令	注释/说明	操作数（寄存器）
FOLLOW	跟随指令	参数首地址； 脉冲发送口； 脉冲方向口。

1. 指令说明：

1） 该指令是脉冲发送指令，其脉冲发送频率根据跟随数值变化的快慢决定，脉冲发送个数根据数值变化的最终大小决定；

2） FGs 系列晶体管型 PLC 高速脉冲输出口为 Y0~Y1，脉冲频率范围为 0~200KHz。FGm 系列 PLC 高速脉冲口为 Y0~Y7，Y10~Y11，脉冲频率范围 0~500KHz；

2. 使用说明：

1） 举例如下图，分别设置 FOLLOW 指令的参数首地址为 D0，脉冲口为 Y0，方向口为 Y12；

2） D0（D1）中为要跟随的目标值，此实例中要跟随的目标值为 CV235 的值，实际也可以为其他值。注意该值的变化速率对应脉冲发送频率，频率不要超过最大频率允许范围。另外，当 FOLLOW 指令使能前，该目标值中有值，则在该值的基础上进行跟随。即跟随是相对的，使能前的初值不影响跟随。

3） D2 参数为采样时间单位 ms，也可认为是跟随刚度，当采样时间较短时，动作严格跟随目标值的变化而变化延时较短，跟随刚度大。当采样时间较长时，动作延时长，且在采样时间内的跟随值的正负变化会相互抵消，实际动作并非严格跟随。建议设置值为 5ms~100ms 之间。

4） D3 参数为前馈补偿比例，设置值只能在 0~100 之间。由于采样时间的滞后，实际运动滞后一个采样时间，若需要在运动过程中保持位置准确性可进行适当的前馈补偿，若对过程中的位置准确性要求不高，建议设置为 0；

5） D4 乘系数，跟随值的**变化量**乘上该系数，对变化值进行放大。D5 除系数，跟随值的**变化量**除上该系数，对变化值进行缩小。乘、除后的最终结果将作为脉冲的发送量。

脉冲正在输出标志(只读)：

寄存器	脉冲输出 Yn
M_8134	Y000
M_8135	Y001
M_8136	Y002
M_8137	Y003
对于多脉冲型 PLC FGm_32MT_A FGm_64MT_A 还具备以下 M 软元件	
M_8138	Y004
M_8139	Y005
M_8140	Y006
M_8141	Y007
M_8142	Y010
M_8143	Y011

脉冲累积计数(读写)：

寄存器	脉冲输出 Yn
D_8140(D_8141)	Y000
D_8142(D_8143)	Y001
D_8144(D_8145)	Y002
D_8146(D_8147)	Y003
对于多脉冲型 PLC FGm_32MT_A FGm_64MT_A 还具备以下寄存器	
D_8148(D_8149)	Y004
D_8150(D_8151)	Y005

D_8152(D_8153)	Y006
D_8154(D_8155)	Y007
D_8156(D_8157)	Y010
D_8158(D_8159)	Y011

注意：

1. 对于高速脉冲输出，它是针对外部高速器件的，要对脉冲进行计数，只能使用高速脉冲输入计数器，不能使用内部计数器，也不能使用 Y 边沿变化进行计数，当作为高速输出时，不能作为普通输出口用；

2. 晶体管的 OFF 时间具有在轻负载时会延长的特性。所以请在需要响应性时设计成让负载在较轻时加负载电阻，增加负载电流；

3. 脉冲累积计数寄存器（D8140~D8158）是重要寄存器，可读写，当写入新值时，计数会在新值基础上进行加减；

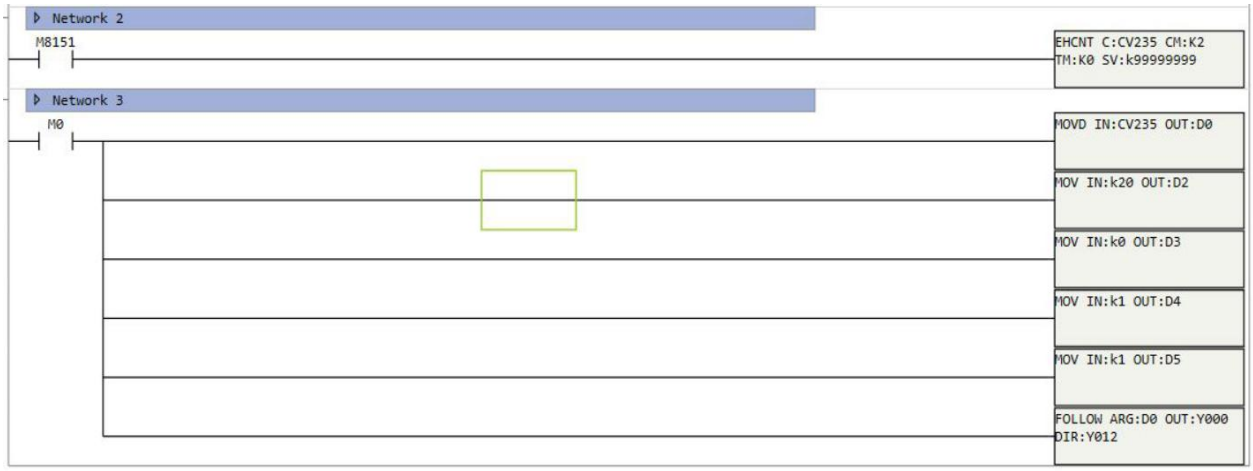
4. 由于本指令脉冲输出带方向，所以脉冲累积计数寄存器（D8140~D8158）根据方向计数；

5. 高速脉冲输出时，脉冲累积计数寄存器（D8140~D8158）内的值是不连续的、时刻变化的，用于判断时，请使用大小比较，不要使用等于判断；

6. 当脉冲指令使用到输出口 Y 时，该 Y 口不能用做它用，即普通指令无法再对该 Y 口进行 ON、OFF 操作；

7. 在跟随过程中跟随目标值不要发生剧烈跳变，否则会脉冲频率会超出最高频率限制。

实例：



4.15.22 平面系统设置

1. 功能说明：

1）在使用任一插补指令（LINEI/LINEF、ARCI/ARCF、POLYLINEI/POLYLINEF、BLOCK 等）前，必须先进行“平面系统设置”；

2）该功能用于创建一个二轴平面系统，插补指令均基于平面实现；

2. 使用说明：

1）“平面系统设置”点击路径为：主菜单 -> 脉冲 -> 脉冲平面 -> 平面系统设置；

2）选择需要创建的平面编号编号，并选择启用；

3）选择参数的单位，可以为毫米 mm 或者脉冲 pls，建议用户使用 mm 作为单位，参数贴合实际，无需换算，不易出错。该单位仅控制该设置界面的参数单位，实际运动指令单位与使用的指令类型相关；

4）越界处理方式，对应设置的平面边界使用，在使用任一插补指令过程中，若下一线段即将越界，则会进行相应的报错与处理。若选择“不处理”，则无视越界，若“减速停”则，停止在前一个不越界线段的终点，即越界线段的起点；

5）定义 X 轴脉冲输出口以及方向口，Y 轴脉冲输出口以及方向口，注意，脉冲输出口必须选择对应 PLC 的高速输出口，不可选用普通输出口，方向输出口可任意选取。另外，脉冲口和方向口都可以使用 D 寄存器在程序中修改对应轴的轴号，但需遵循原则：所有启用的平面系统，脉冲输出口，方向输出口不能完全重复；

6） **脉冲当量是指 PLC 发送一个脉冲，对应的执行机构运行的距离。**省略了中间传动环节，但实际需考虑：伺服（步进）电机细分、电子齿轮比，传动机构传动比等。脉冲当量是设置的坐标数据与实际尺寸衔接的核心参数，必须准确无误，若发现实际尺寸是理论尺寸的缩放，则是脉冲当量设置错误造成；

7） 极限位置是平面系统的理论边界位置，依照实际情况设置，若线段即将超出边界会按照越界处理选择的处理方式处理；

8） 传动间隙（回程间隙）是指传动轴在方向发生改变时，由于机械传动系统磨损、安装等造成的一段空行程。系统会在对应轴的方向发生改变时在若干插补周期内补偿该间隙值，在粗调设备时可不设置该参数，精调时合理的设置该参数。该值不要设置太大，若实际存在大间隙，请调整机械结构，另外在高速运行场合不适合使用该参数。



平面系统设置参数

注意：

1. **X 轴 Y 轴脉冲当量尽量相同，若不相同，尽量选择系统单位为 mm，以避免换算造成参数错误；**
2. **当 X 轴 Y 轴脉冲当量不同时，插补指令中以脉冲为单位而无指定轴的参数（如速度），内部计算均会按 X 轴脉冲当量计算；**
3. **所有启用的平面系统，脉冲输出口，方向输出口不能完全重复；**

4.15.23 ECAM

电子凸轮 ECAM

1.指令说明

在钢材剪切等行业中，传统方法多采用停剪、停切的方法，效率较低，并且未必能满足材料的成型工艺要求。对于包装印刷等对从轴有实时相位跟随需求的行业，传统行业多采用机械凸轮的方式来实现。

1.1 机械凸轮与电子凸轮

由图 1.0 可知，带动凸轮转动的轴为主轴，凸轮推动运动的轴为从轴，主轴处于任意位置都对应从轴一个确定的位置，这个对应关系取决于凸轮的轮廓曲线，本质上反映了主轴位置与从轴位置之间的某种函数关系。实际加工中，用户使用凸轮时并不是整个凸轮曲线都对加工过程起关键作用，往往是其中某些点或者线段对

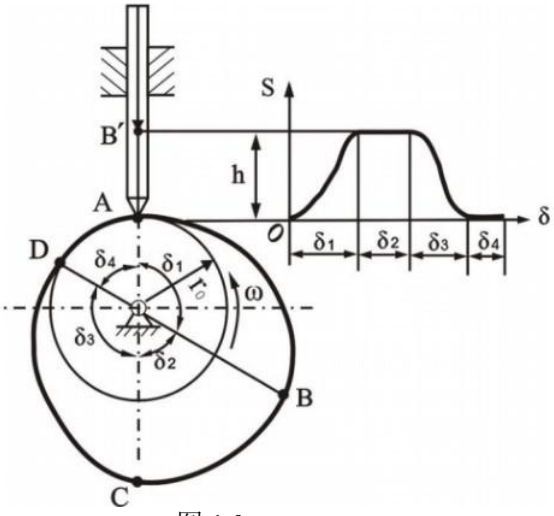


图 1.0

加工起决定性作用，利用这些点和线段，结合合适的数学模型可以建立出主从轴位置间的函数关系。根据当前的主轴位置计算出相应的从轴位置，通过 PLC 发送脉冲，驱动从轴机构运动到对应的位置上即可实现从轴按凸轮轨迹运行的功能，这便是电子凸轮的工作原理。

与机械凸轮相比，电子凸轮更加的灵活，修改曲线更加方便，维护成本也大大降低。目前显控电子凸轮指令有三种模式：追剪、飞剪、通用凸轮。其中追剪和飞剪是通用凸轮的两种特殊的应用场景,应用于型材剪切，定长切纸等领域。通用凸轮可以通过关键点的控制来生成任意非规则的凸轮曲线，使用场景更加灵活。针对这三种模式，显控 ECAM 指令都提供了可交互图形界面来设置参数，还可以预览和调整自动生成的电子凸轮运动曲线。

1.2 追剪概述

在型材、纸管等的送料剪切中，传统工艺多采用停剪的方法，效率较低。电子凸轮追剪功能实现了从轴切台实时跟随主轴运动，在运动过程中利用同步区进行裁剪切割，然后减速到 0，然后迅速返回原点，等待下次跟随运动。这样可以保证主轴送料的不间断，提高生产效率。
追剪的模块由主轴送料驱动、主轴位置反馈、从轴直线跟随，裁切模组这四部分组成。其中主轴送料驱动机构常由伺服或者变频器组成，主轴位置反馈机构主要由编码器组成，编码器反馈主轴当前位置和速度。从轴

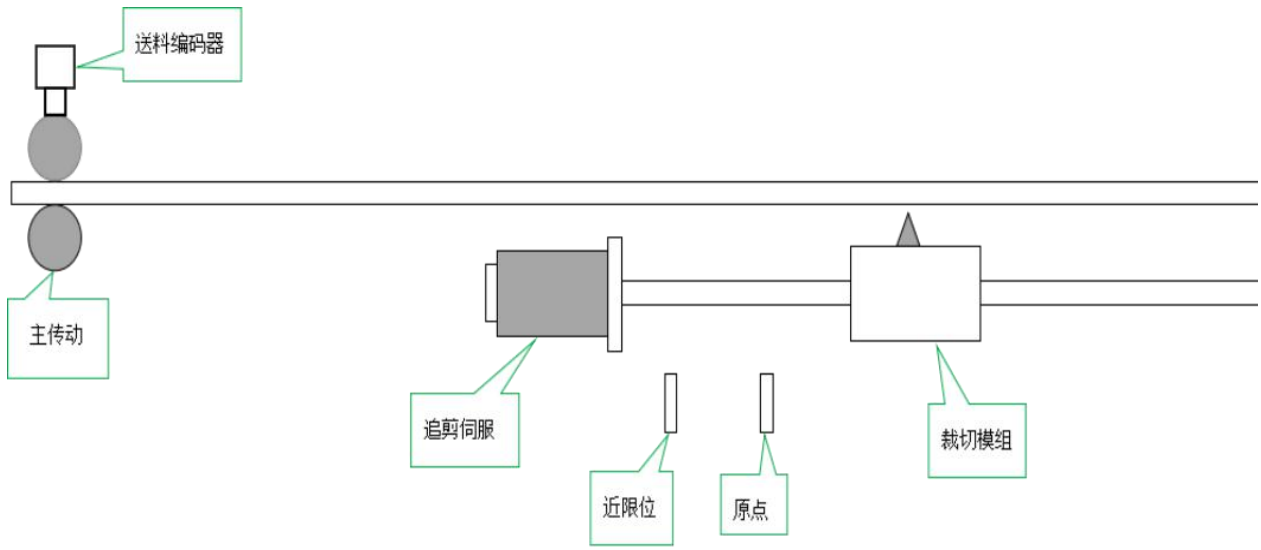


图 1.1 追剪结构示意图

直线跟随机构多由伺服通过丝杠带动裁切模组运动。裁切模组则主要由气压、液压或伺服驱动。相比较于将追剪的程序固化到伺服 中，在 PLC 中实现追剪功能更加便于修改，结合 PLC 应用简便的特点，显控 PLC 的 ECAM 指令使用更加的方便快捷。如图 1.1 所示为追剪系统结构示意图。

主传动部分由伺服或者变频器带动压辊进行送料，送料的位置和速度由编码器接入 PLC 的高速计数器进行反馈，追剪伺服则接收追剪功能中设定的脉冲口发出的脉冲信号， 驱动裁切模组到达对应的从轴所处的凸轮曲线位置，裁切模组则根据同步区标志进行裁切。原点信号用于追剪台的复位，限位用于行程保护限位。
追剪需要配置剪切长度，根据曲线参数以及各种机械参数，自动生成追剪曲线。曲线由 4 段组成，分别为加速区，同步区，减速区，反向区。4 段曲线的示意图如图 1.2 所示。

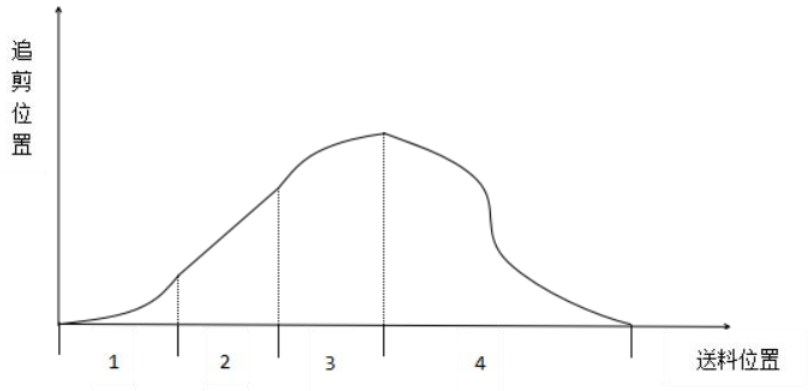


图 1.2　位置曲线

其中横坐标大小代表主轴走过的距离，纵坐标代表从轴实际所处的位置：
1.加速区：本区域长度代表从轴加速过程中主轴走过的距离
2.同步区：本区域长度代表从轴与主轴同步运动过程中主轴走过的距离，此区域内从轴与主轴走过的距离相同
3.减速区：本区域代表从轴减速过程中主轴走过的距离
4.反向区：本区域代表从轴反向回到原点的过程中主轴所走过的距离计算公式：料长 = 加速 + 同步 + 减速 + 反向
　同步区 = 同步速度 * 剪切时间（主轴速度稳定时，同步速度即为主轴速度。剪切时间需大于实际剪切时间）

注意：加速区与减速区越大，从轴加减速越平稳。料长一定时，加减速区域大小会影响同步区的长度大小，同步区长度需要保证能够进行正常的剪切需求。

1.3 飞剪概述

在定长切纸，包装裁剪等行业中，飞剪能够使飞剪轮的转动实时跟随主轴运动，送料编码器接入 PLC 高速计数器反馈主轴位置和速度，同时飞剪伺服接收 ECAM 指令中指定的脉冲口脉冲信号，驱动飞剪轮到达凸轮曲线对应的从轴位置。其中当光电原点信号被检测到时，切刀处于起点位置，如图 1.3 所示为飞剪结构示意图。飞剪的模块组成由主轴送料驱动、主轴位置反馈、飞剪轮这三部分组成。其中主轴送料驱动机构常由伺服或者变频器组成，主轴位置反馈机构主要由编码器组成，编码器反馈主轴当前位置和速度。飞剪轮通常由伺服带动。如图 1.3 所示为飞剪系统结构示意图。

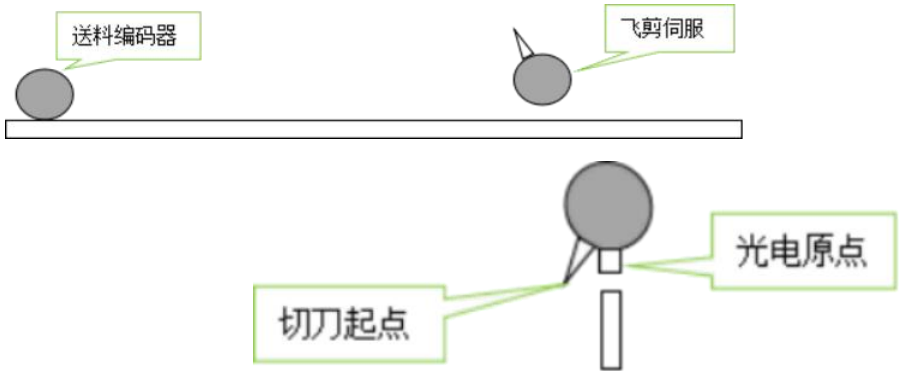


图 1.3 飞剪结构示意图

飞剪轮通过配置剪切长度，加速区，同步区角度，切刀个数等参数自动生成飞剪曲线。飞剪轮运动轨迹划分为加速区，同步区，减速区三个区域。如图 1.4 所示飞剪轮的运行轨迹划分。

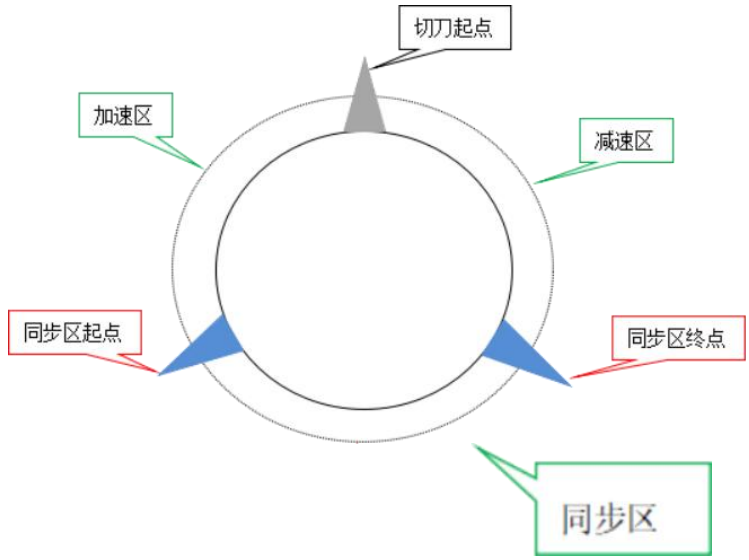


图1.4 飞剪轮运动轨迹划分

飞剪运动曲线如图 1.5 所示：

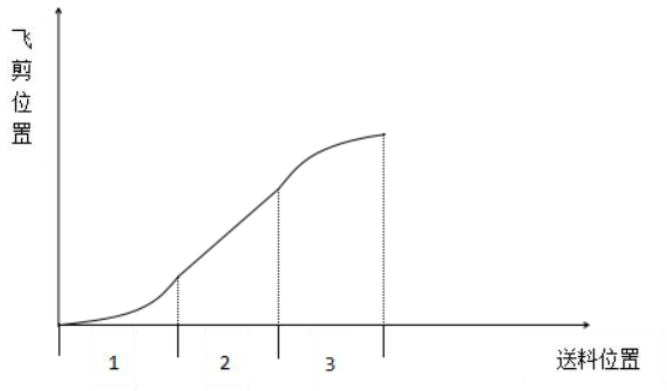


图 1.5 飞剪曲线示意图

其中横坐标大小代表主轴走过的距离，纵坐标代表飞剪轮实际所处的角度（切刀起点处为 0°）
1.加速区：本区域长度代表从轴加速过程中主轴走过的距离
2.同步区：本区域长度代表从轴与主轴同步运动过程中主轴走过的距离，此区域内飞剪轮所转过的线长度与主轴走过的距离相同
3.减速区：本区域代表从轴减速过程中主轴走过的距离

1.4 通用凸轮概述

追剪与飞剪是较为常见的凸轮应用，根据主从轴位置关系的不同还有很多的电子凸轮的应用需求。ECAM 指令提供通用电子凸轮功能，可以通过关键点坐标和线段参数来配置自定义的凸轮曲线。

自定义凸轮需要设定各个关键点参数，每个关键点参数由主轴坐标，从轴坐标，目标斜率，线型四个参数组成。通过关键点坐标可以确定关键点的位置，通过斜率和线型可以确定曲线的形状，从而形成用户自定义的凸轮曲线。
取以下五个关键点【0,0,0,0】【100,50,0.5,1】【300,400,-1.5,0】【500,100,-1.5,1】【600,400,0.5,0】，

可形成图 1.6 所示自定义曲线：

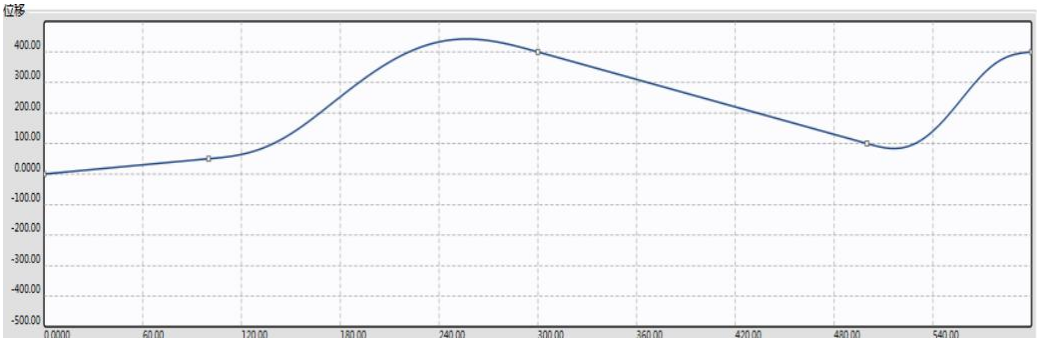


图 1.6　通用凸轮曲线

运行过程说明：

第一段：当主轴从 0 运行到 100 时，从轴从 0 运行到 50，根据终点斜率 0.5 以及本段运行曲线为直线，当主轴匀速运行时，从轴以主轴速度的一半运行到 50

第二段：当主轴从 100 运行到 300 时，从轴从 50 运行到 400，本段曲线为样条曲线，并且终点斜率为-1.5，从轴按照正弦曲线运行到 400 的位置

第三段：当主轴从 300 运行到 500 时，从轴从 400 运行到 100，根据本段为直线并且斜率为-1.5，从轴匀速运行到 100 的位置

第四段：当主轴从 500 运行到 600 时，从轴从 100 运行到 400，本段为样条曲线并且斜率为 0，从轴按照正弦曲线运行到 400 的位置

当主轴运行越过第四段终点时，从轴以终点位置 400 为基准继续按照凸轮曲线开始运行，以此循环。

2.参数配置

ECAM 电子凸轮的参数可以分为两部分，一部分为通用参数，一部分不同模式下的参数。参数设置不合理，凸轮不会运行，并且错误码寄存器 D8176 中会　有相应的错误码 46，参数错误的可能情况在本段注释 2 中详细描述。

2.1 端口配置与通用参数

图 2.1 为参数配置界面，表 2.1 为相应参数设置的详细说明。其中通用参数分为三个类型：补偿系数，运行状态参数，跟随系数。

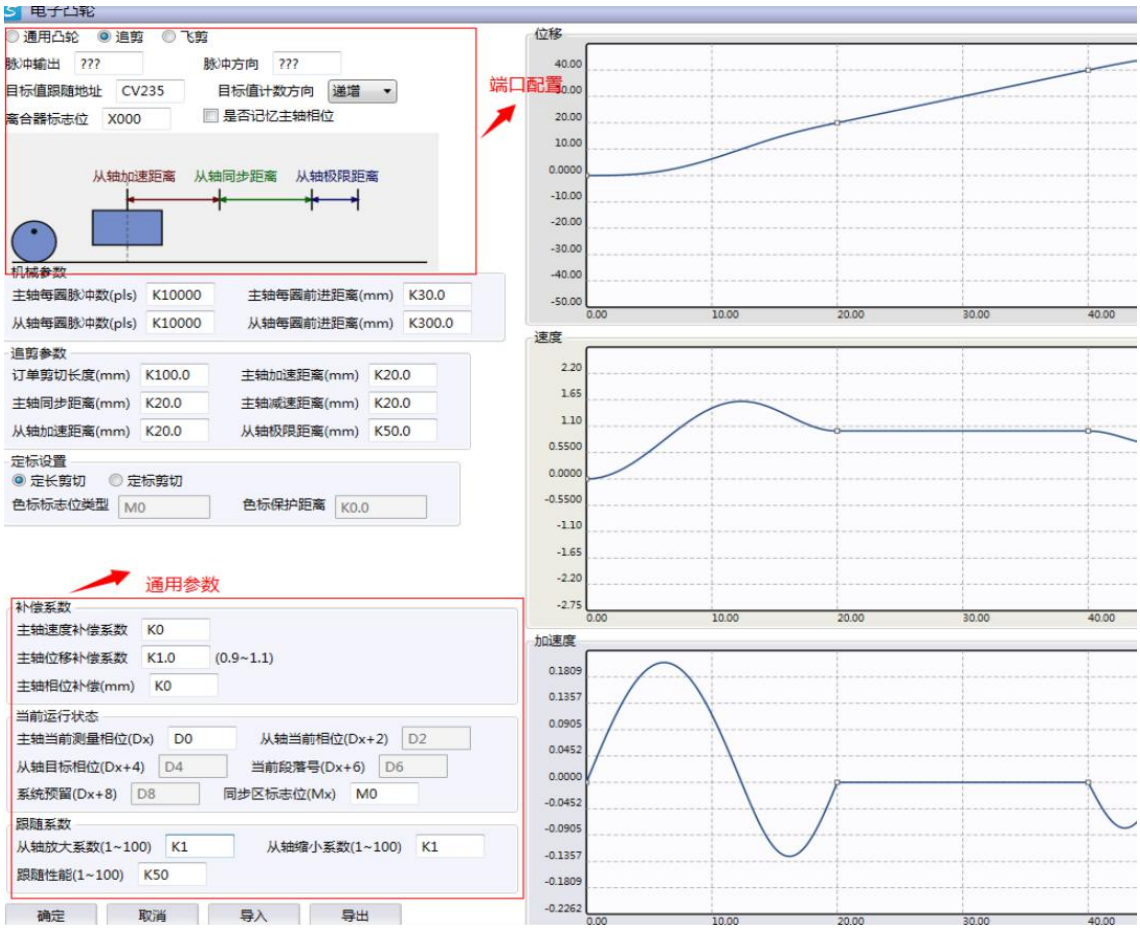


图 2.1

注：数据类型与寄存器类型

- INT32 代表 32 位有符号整数，UINT32 代表 32 位无符号整数，FP32 代表 32 位浮点数，INT16 位 16 位有符号整数，UINT16 为 16 位无符号整数，BOOL 代表标志位
- R 只读，W 只写，R/W 可读可写，单位 pls 代表脉冲数
- 本指令中：寄存器类型 Y 代表 Y 脉冲输出端口位寄存器，D 代表双字 D 寄存器，K 代表常数，CV 代表高速计数器寄存器，X 代表输入端口位寄存器，M 代表内部 M 位寄存器

表 2.1

	参数类型	参数名称	数据类型	寄存器类型	默认值	取值范围	单位	备注
端口配置	脉冲相关	脉冲输出口		Y				PLC 上对从轴伺服输出脉冲
		脉冲方向口		Y				PLC 上对从轴伺服输出方向
	跟随目标值	跟随目标值地址		D/CV				记录主轴位置的相关寄存高速计数器CV
		目标值计数方向			递增			主轴前进中目标值地址的值
	离合器相关	离合器标志位		X/M				目前仅支持X/M,注意选择X X 端口（支持IO 扩展板X 端 X 端口，同时 X 端口需要一（注释5）
	其他	是否记忆主轴相位			否			选择是代表ECAM 指令重新能时的主从轴相位开始运后，将清空之前的主从轴清空），并从头开始运行
通用参数	运行参数（注释6）	主轴当前测量相位（以下为连续的双字D 寄存器）	INT32/W/R（运行中只读，失能后可读可写）	D			PLS	
		从轴当前相位	INT32/W/R	D			pls	从轴在一个相位周期内所的终点后重新。 1 追飞剪：追剪从轴相剪从轴相位周期终点数 1 通用凸轮：相位周期为置对应的脉冲数
		从轴目标相位	INT32/W/R	D			pls	主轴当前位置对应的凸轮 当前位置以此为目标进行跟
		当前段落号	UINT32/W/R	D			pls	当前主轴运行区间号
		系统保留	INT32*3/W/R	D				系统用于记录数据用的寄存器
		同步区标志（追飞剪）（以下为连续的M 标志位）	BOOL/W/R	M				追飞剪中主从轴处于同步同步区内
		主从轴绑定标志（追飞剪色标模式）	BOOL/W/R	M				追飞剪在色标剪切模式下位，on 代表处于绑定状态状态
	跟随系数	从轴放大系数	INT32/W	D/K	1	1-100		成比例的放大凸轮曲线对应
		从轴缩小系数	INT32/W	D/K	1	1-100		成比例的缩小凸轮曲线对应
		跟随性能	INT32/W	D/K	50	1-100		影响从轴跟随目标值的刚从轴跟随目标值越紧密
	补偿系数	主轴速度补偿系数	FP32/W	D/K	0	-100:100		由于算法上的延时，根据相应的补偿量。程序会根个最优的补偿系数，此值量，输入值±100 以内（注
		主轴位移补偿系数	FP32/W	D/K	1	0.9-1.1		定长连续剪切过程中由于距离越长就会累积越多的用来减小这种累积误差， 值
		主轴相位补偿	FP32/W	D/K	0		mm	通过相位补偿可以前移或 平移了主从轴的凸轮位置曲

2.1 追剪参数

图 2.2 为追剪参数配置界面，表 2.2 为相应参数设置的详细说明。

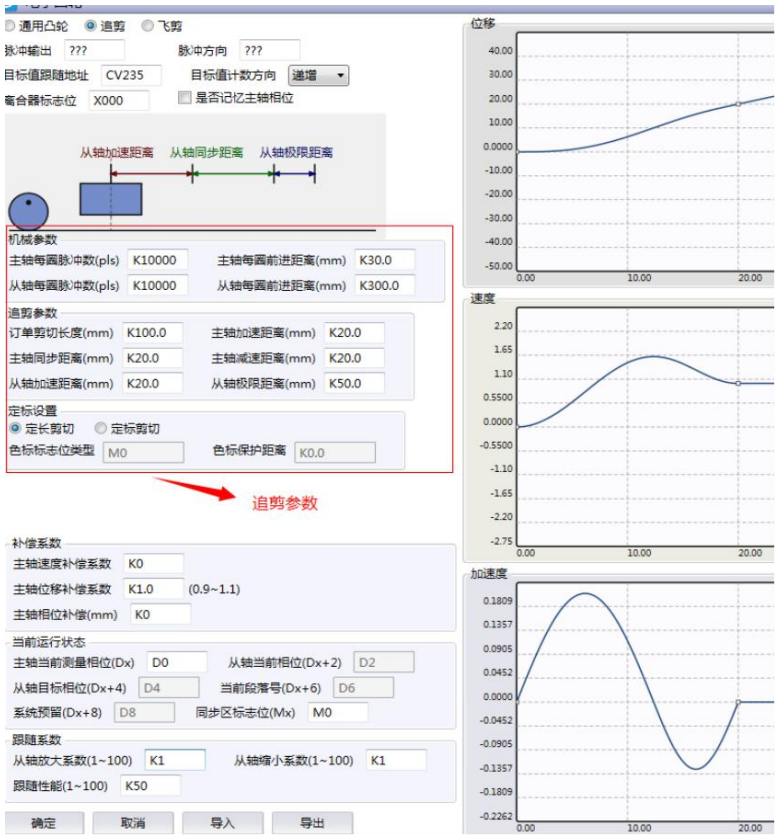


图 2.2

表 2.2（接上表）

	参数类型	参数名称	数据类型	寄存器类型	默认值	取值范围	单位	备注
追剪曲线	机械参数	主轴每圈脉冲数	UINT32/W	D/K		>0	pls	主轴伺服每圈所发脉冲或主轴编码器每圈反馈脉冲数
		主轴每圈距离	FP32/W	D/K		>0	mm	主轴伺服或者编码器转一圈实际送料的距离
		从轴每圈脉冲数	UINT32/W	D/K		>0	pls	从轴伺服每圈所发脉冲数
		从轴每圈距离	FP32/W	D/K		>0	mm	从轴伺服转一圈追剪台前进的距离
	凸轮模式	定标设置						● 定长模式：连续追剪，中间不停顿 ● 定标模式：检测到色标后才开始追剪，追剪完成后等待下个色标到来再开始追剪
		色标标志位		X/M				PLC 上用于检测色标信号的端口（支持IO 扩展板X 端口，不支持扩展模块上的X 端口）
		色标保护距离	FP32/W	D/K	0	>=0	mm	定标模式下，检测到一个色标后，必须再过（订单剪切长度+色标保护距离）才能再次检测到色标
	追剪参数	订单剪切长度	FP32/W	D/K		>0	mm	● 定长模式：实际剪切的料长 ● 定标模式：剪切料长的最小值（注释1）
		主轴加速距离	FP32/W	D/K		>0	mm	从轴加速阶段对应主轴走过的距离（注释2）
		主轴同步距离	FP32/W	D/K		>0	mm	从轴同步阶段对应主轴走过的距离
		主轴减速距离	FP32/W	D/K		>0	mm	从轴减速阶段对应主轴走过的距离
		从轴加速距离	FP32/W	D/K		>0	mm	从轴加速阶段走过的距离
		从轴极限距离	FP32/W	D/K		>0	mm	从轴在运行过程中与原点之间所产生的最大距离（注释2）

2.2 飞剪参数

图 2.3 为追剪参数配置界面，表 2.3 为相应参数设置的详细说明。

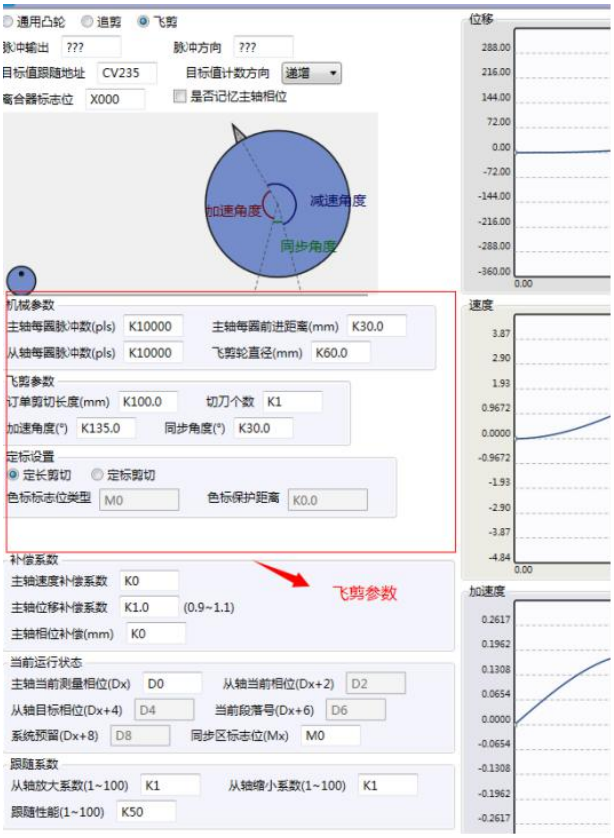


图 2.3

表 2.3（接上表）

	参数类型	参数名称	数据类型	寄存器类型	默认值	取值范围	单位	备注
飞剪曲线	机械参数	主轴每圈脉冲数	UINT32/W	D/K		>0	pls	主轴伺服每圈所发脉冲或主轴编码器每圈反馈脉数
		主轴每圈距离	FP32/W	D/K		>0	mm	主轴伺服转一圈或者编码器转一圈实际送料的距
		从轴每圈脉冲数	UINT32/W	D/K		>0	pls	从轴伺服每圈所发脉冲数
		飞剪轮直径	FP32/W	D/K		>0	mm	
	凸轮模式	定标类型						● 定长模式：连续追剪，中间不停顿 ● 定标模式：检测到色标后才开始飞剪，飞剪成后等待下个色标到来再开始飞剪
		色标标志位		X/M				PLC 上用于检测色标信号的端口（支持IO 扩展板X 端口不支持扩展模块上的X 端口）
		色标保护距离	FP32/W	D/K	0	>=0	mm	定标模式下，检测到一个色标后，必须再过（订剪切长度+色标保护距离）才能再次检测到色标
	飞剪参数	订单剪切长度	FP32/W	D/K		>0	mm	● 定长模式：实际剪切的料长 ● 定标模式：剪切料长的最小值（注释1）
		切刀个数	INT32/W	D/K		>0		
		加速角度	FP32/W	D/K		>0&&<360/ 切刀数	°(度)	飞剪轮的加速区角度
		同步角度	FP32/W	D/K		>0&&<360/ 切刀数	°(度)	飞剪轮的同步区角度（注释2）

2.1 通用凸轮参数

图 2.4 为追剪参数配置界面，表 2.4 为相应参数设置的详细说明。



图 2.4

表 2.4(接上表)

	参数类型	参数名称	数据类型	寄存器类型	默认值	取值范围	单位	备注
通用凸轮	机械参数	毫米/脉冲数						选择通用凸轮里的关键点坐标单位是毫米或脉冲数
		从轴范围						用于标识坐标图中从轴的范围大小
		主轴当量	FP32/W	D/K	0.001	>0	mm/pls	主轴的脉冲当量
		从轴当量	FP32/W	D/K	0.001	>0	mm/pls	从轴的脉冲当量
	关键点参数	主轴坐标		D/K				主从轴坐标单位根据之前选择的毫米或者脉冲数决定（注释4）
		从轴坐标		D/K				
		目标斜率		D/K				本关键点在线段终点处的斜率大小
		线段类型						以本关键点为终点的线段类型：目前支持样条曲线和直线两种
	其他	仅D 控制						选择仅D 控制时，通过设定控制的起始地址，可以将关键点的4 个参数转化为连续的双字D 寄存器地址控制，这样可以通过HMI 来修改关键点参数。注意双字D 寄存器中主从坐标的数据类型要根据单位是mm 还是脉冲数来决定，斜率是浮点数类型，线型是整数类型 **注意：** 1）D 寄存器开始地址第一个双字整型为关键点个数

								2）接下来的D 寄存器中主从坐标的数据类型要根据单位是mm 还是脉冲数来决定，如果是mm 则主从坐标为浮点型，脉冲数则输入双整型 3）斜率是浮点数类型，线型是双整数类型 4）线型：0 代表样条曲线 1 代表直线，其余值代表直线
		控制起始地址						设定连续D 寄存器控制的起始地址

曲线预览区

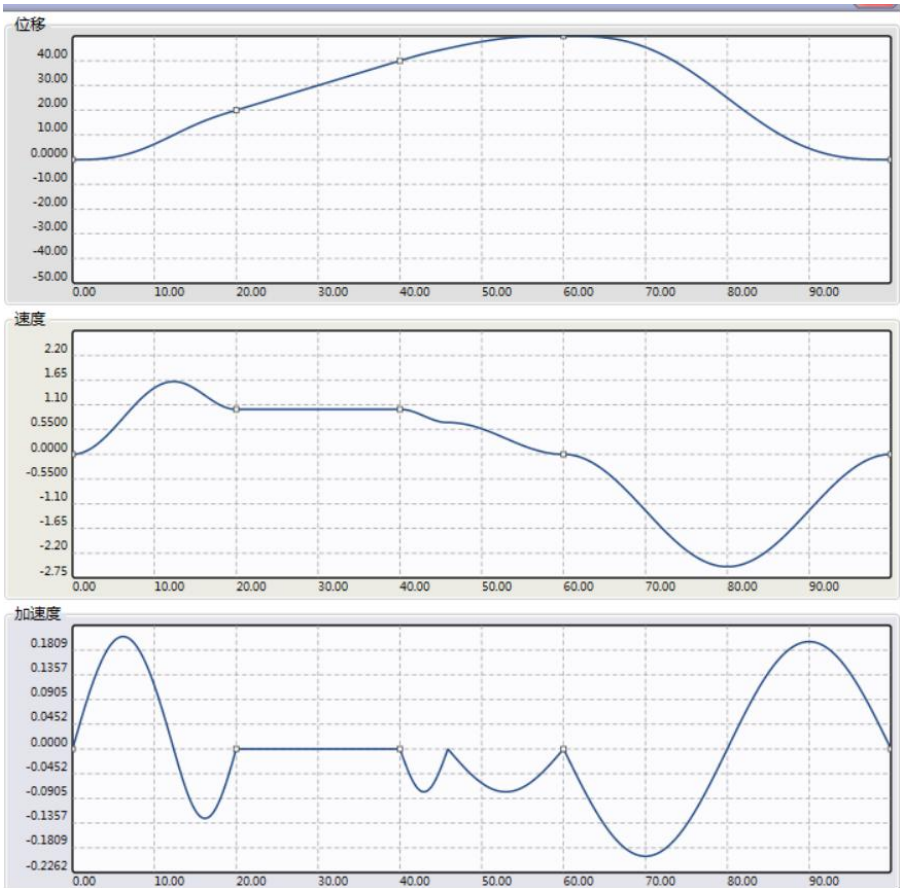


图 2.5

如图 2.5 所示：曲线预览区主轴坐标代表主轴走过的距离，从轴分别代表从轴的位置，速度，加速度，可以通过输入参数来自动生成和调整曲线。根据速度曲线图，参数设置应该尽量保证从轴速度在关键点处的平滑过渡，不宜出现速度太大导致曲线过陡的情况，还可以利用主轴速度大致判断出从轴最高速度是多少，从而判断是否需要调整曲线或者降低主轴的速度以免发生速度超限的情况。

注释：

1.定标模式

如果两个色标检测距离大于剪切订单长度，实际切料长度为色标距离，如果色标检测距离刚好等于订单剪切长度，实际切料距离为剪切订单长度，如果色标检测距离小于订单剪切长度，则第二个色标的检测将不起作用，等待下个色标的检测，并且下个色标距离与起始的色标之间距离大于订单剪切长度该色标才会起作用，这种情况下，有效的两个色标之间可能存在 1 个或多个无效的色标信号

2.参数设置合理性

所有参数设置不能超出取值范围，同时部分参数还需要满足以下关系,否则会引起参数错误报警，D8176 中对应的错误码为 46

- ✓ 追剪加速区 + 同步区 + 减速区 < 剪切长度
- ✓ 追剪从轴加速距离 + 同步区 < 追剪从轴极限距离
- ✓ 飞剪同步区 + 加速区 <（360°/ 切刀个数）

3.三种误差补偿

1) 主轴速度补偿系数

描述：该补偿系数主要是用来补偿与速度成正比的偏差。加速过程中，速度越大，偏差越大；减速过程相反；匀速过程偏差稳定；偏差与主轴关系的示意如图 2.6 所示：

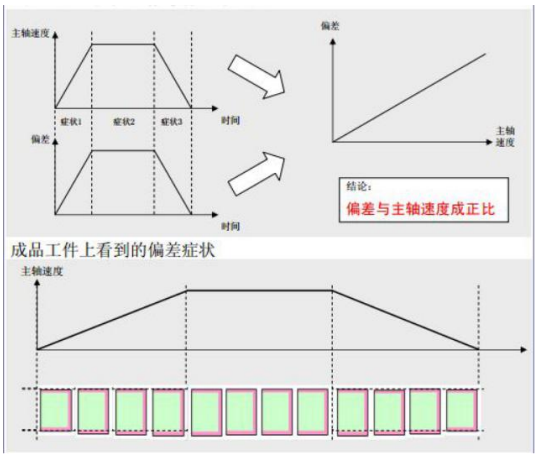


图 2.6

原因：由于电子凸轮的特性，在运算过程中由于处理时间引起的延迟。

解决办法：程序内部会给出一个合适的算法延时时间 T，根据当前的主轴速度 F，即可算出应该补偿的主轴前进量 △=F*T，该补偿量将加入到每次测量到的主轴位置上，从而达到补偿的目的。注意参数中的速度补偿系数的含义是时间 T 的偏移调整量，一般情况下设为 0 即可，需要时可以略微左右调整。

2) 主轴位移补偿系数

描述：主轴移动量的增加会导致同步偏差增加，如图 2.7 所示

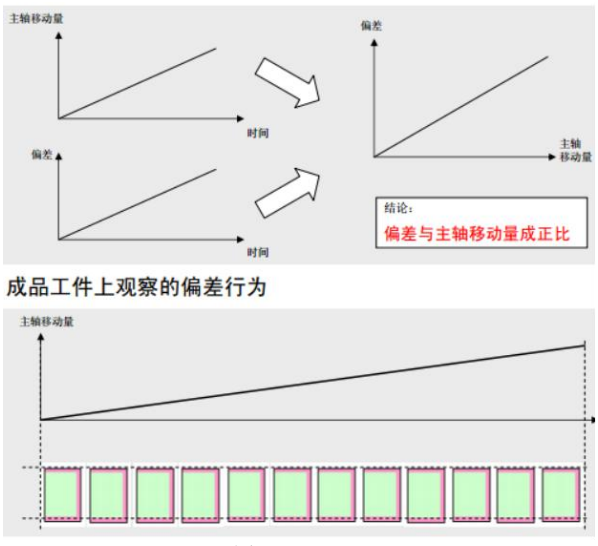


图 2.7

原因：机械结构与给定值之间存在误差。

举例说明：飞剪轮一圈前进举例 L = 飞剪轮直径 R * π。由于π=3.14159265358979…是个无理数。实际计算的时候π会取一定位数，例如 3.14159265358979，当直径为 100m 时，一圈有 0.000265368979 的误差，经过 10000 圈后，就有 2.65368979mm 的误差，这是不能忽略的。这个误差是随着总距离的增大会不断增加。
解决办法：由于该偏差与总长之间呈一个比例关系，所以通过在实际运行过程中的试验可以得到该比例关系，总长 L*比例 K = 总长误差补偿量，在运行过程中的主轴位置上加上该补偿量可以减少由于机械结构误差导致的移动总长误差。
注意：定长剪切中如果机械参数较为准确的情况下，该误差补偿系数非常接近 1。定标剪切中如无必要，可以不考虑，设为默认值 1。

3) 主轴相位剪切补偿

描述：与主轴移动量大小和速度大小没有关系的误差。剪切的第一刀发生了偏差，引起了后续剪切位置的整体偏移，如图 2.8 所示

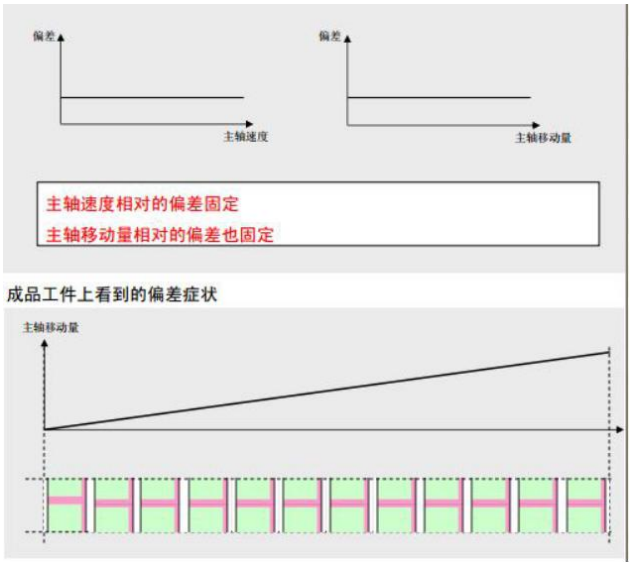


图 2.8

原因：主从轴的位置关系发生了偏差，只有第一次运行的时候会影响加工，但如果是切纸这一类的剪切位置需要对应图样的场景，则完全不允许这种情况发生。

解决办法：在主轴运行的过程中始终偏移一个初始相位即可。对应参数中的主轴相位补偿，补偿值大小为第一刀的偏差量，偏差量为正的时候提前剪切，偏差为负的时候滞后剪切。如图 2.9 所示

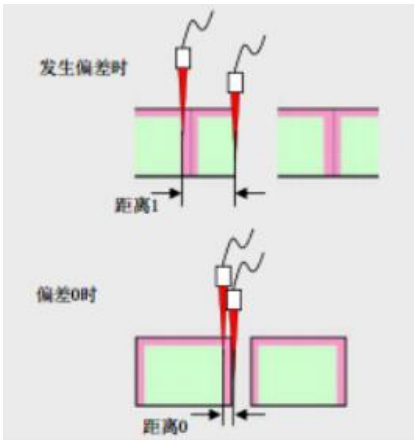


图 2.9

4) 通用凸轮参数设定

通用凸轮的参数设定需要满足一定的规则，并且会做某些合理性的调整

- 如果主轴目标值计数方向为递增，则本段主轴坐标必须大于上一段主轴坐标，否则会引起报错
- 如果本段从轴与上一段从轴的从轴坐标相等，则本关键点的斜率自动调整为 0
- 如果本段关键点线型为直线，则本段的斜率将由从轴坐标与主轴坐标差值之比自动算出，用户输入如果不匹配将自动修改
- 如果本段关键点为直线，上一个关键点的线型为样条，则上一段关键点的斜率将自动调整为本段直线的斜率（保证平滑过渡）

5） 离合器

离合器标志位可以使 X 或 M 位寄存器。置位离合器标志位可以在不断开指令的情况下使得主从轴的绑定关系断开，从而可以利用别的脉冲指令控制从轴运动。复位离合器时， 主从轴将以断开时的主轴位置以及对应的从轴位置开始运动。

注意：置位离合器时，如非特别必要，建议主轴在离合器生效期间不发生位移，否则离合器复位后，第一刀的料长将会产生相应平移。同时从轴如果使用别的脉冲指令后， 必须先断开相应的脉冲指令使能然后才能复位离合器标志位，否则主从轴将不能成功绑定，从轴如果在离合器生效时发生了偏移，那么同步区以及原点位置也将发生偏移，对于追剪而言，改变原点位置导致行程整体平移，这可能会导致超出限位的情况出现。离合器生效期间，如果失能指令，在不记忆主轴相位的情况下重新使能将会从主轴的初始位置开始重新运行，这样很可能导致从轴行程范围的整体偏移。

6） 运行参数

运行参数由连续的双字 D 寄存器和 M 位寄存器组成。如果用户希望指令失能后还能保存当前的运行位置，可以将这些寄存器都设置在掉电保持的范围内，这样重新使能后 ECAM 将自动从这些寄存器记录的位置开始运行，而不是从头开始重新运行。如果用户设勾选了记忆主轴位置，并且又希望清空 ECAM 数据从头开始，可以选择失能 ECAM，梯形图中清空这一系列寄存器，然后重新使能即可重头开始运行。

运行参数在 ECAM 运行过程中只读，失能后可读可写。

3. 应用

本章主要介绍使用 ECAM 指令的步骤和一些关键的注意事项

3.1 追飞剪的应用

调试步骤

1) 检查 PLC 脉冲口是否正常，伺服是否正常工作,接线是否正确
2) 配置脉冲端口，以及目标值地址。确认主轴前进方向对应的目标值是递增还是递减.配置离合器的标志位，可以利用 X 端口外接离合器信号，也可以映射到内部的 M 寄存器上。确认该指令失能时是否需要记录当前主从轴的相位，如果需要记录当前主从轴的相位，运行参数的首地址均需要设在掉电保持区，这样掉电后重新上电使能该指令就能够从上次断电的位置开始重新运行，否则从初始位置开始运行

3) 配置追飞剪的机械参数如每圈脉冲数，每圈前进距离等需要尽可能的准确，否则会对剪切长度或同步区造成影响。

4) 根据实际选择是定标模式还是定长模式，结合实际情况设置追剪或飞剪参数。

（一）追剪

满足参数合法性需求的前提下（见第二部分参数介绍），追剪参数的目的是保证从轴运行中加减速平稳，有足够的同步区距离完成剪切动作，追剪各参数的关系示意图如 3.1 所示。

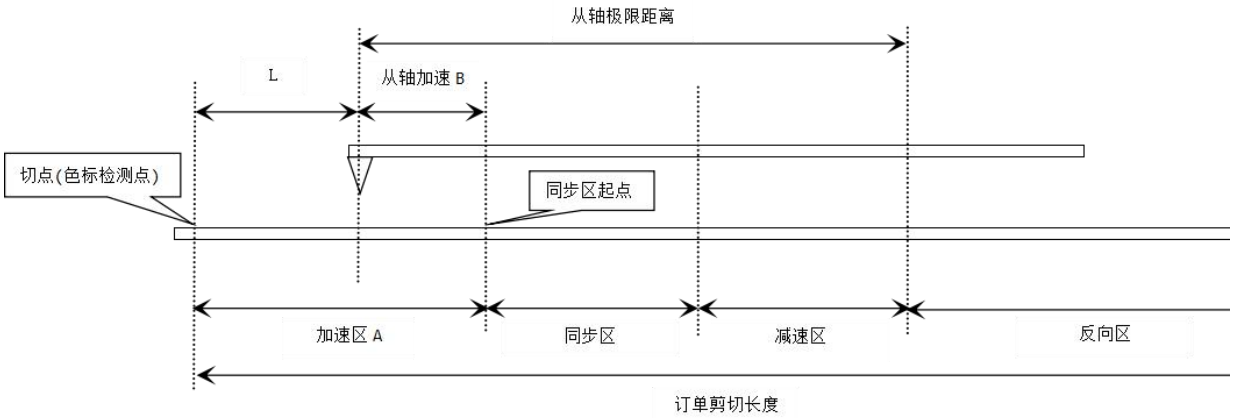


图 3.1

设主轴加速区距离为 A，从轴加速距离 B，当从轴位于起点时，切点距离起点之间的距离为 L，则 L = |A - B|，此时切点位置已知，根据 A，B 的大小关系切点位置可能在起点的不同侧。其余同步区减速区及从轴极限距离在保证速度曲线平稳的前提下根据实际情况设置。
如果选择定长模式，根据切点位置，首料切割时可以保证料长可控。
如果选择定标模式，从轴处于起点位置时，如果色标点位于切点位置，则色标点为剪切点。色标点位置可调的情况下，可以根据实际情况适当偏置色标点。

（二）飞剪
满足参数合法性需求的前提下（见第二部分参数介绍），飞剪参数的目的是保证旋切轴在运行中加减速平稳，有足够的同步区距离完成剪切动作，飞剪各参数的关系示意图如 3.2 所示。

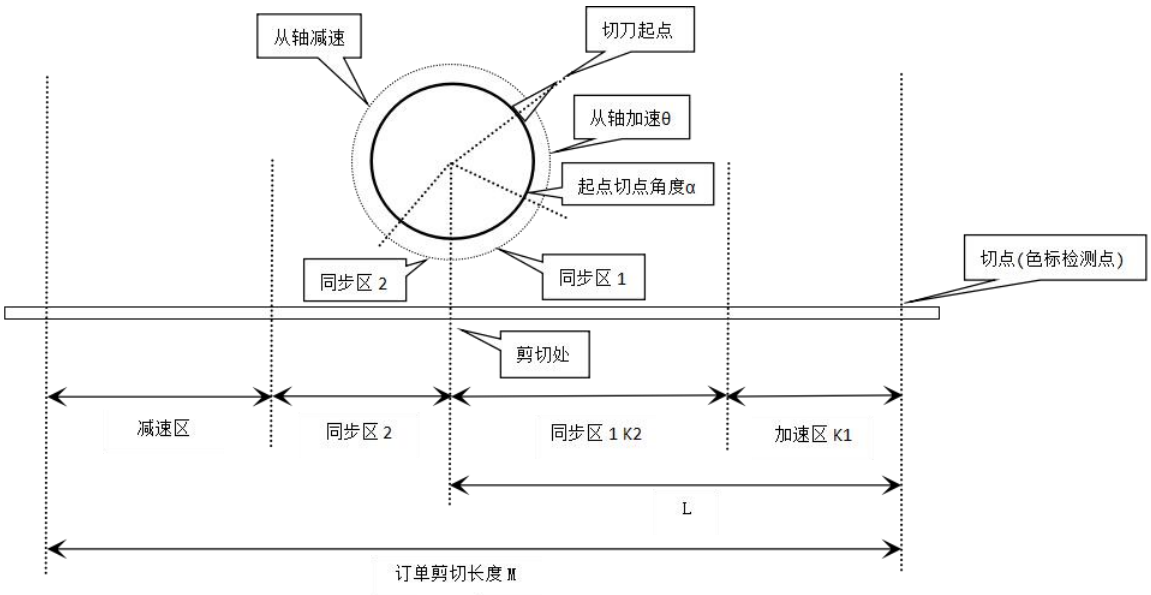


图 3.2

注：α为切刀起点到垂直正下方的剪切处的角度，同步区 1,2 之和为β
设订单剪切长度为 M，从轴初始位置时剪切处与切点距离 L，加速区角度θ，同步区角度β，切刀起点与切点之间角度α，切刀直径 D，切刀个数 i
式①：主轴加速区 K1= M * θ / (360 / i)
式②：主轴同步区 1 K2 = ((α - θ) / 360) * π * D，（α ＞θ，切刀起点与切点之间角度必须大于加速区角度）
式③：起始切点到剪切处的距离 L = K1 + K2
式④：θ + β <（360 / i），0< α - θ <β
其中 M，α，D，i 均为已知参数

如果选择定长剪切，在④的范围内确定合理的角度θ，β，确保加减速过程平稳，结合①②③可以计算得出 L，可以控制首料长度。
如果选择定标剪切，色标即为剪切点时，L 即为色标点到剪切处的距离，并且可以调整。确定一个合适的角度θ，结合①②③，可以计算出 L，结合④，可以得出同步区角度β 的取值范围，可以在这个范围内调整β或调整θ使曲线平稳。当然可以根据实际情况偏置色标点，从而偏置剪切点。
实际运行中如果订单剪切长度远小于飞剪轮周长，旋切轴加减速过程会比较剧烈，反之，旋切轴减速为零后

的等待距离较长。

5. 根据跟随的状况设置合适的跟随系数，跟随系数越大，跟随的越紧密，从轴响应越快

6. 编写主从轴手动运行程序，编写从轴复位到原点的手动程序

7. 设定完成后可以试运行，观察同步区的速度同步情况，如果相差不大可以试切，如果相差过大需要检查参数设定是否正常，确保各个机械参数的准确性，例如由于机械结构的精度影响，主从轴每圈前进距离存在较大误差，飞剪轮包含切刀的直径是否存在较大误差，必要时需要通过手动程序进行实际测量，将修正值重新输入后再次启动 ECAM 指令。

8. 试切后可能出现剪切长度与设定长度不一致的情况，这个时候需要通过手动运行程序，使得主从轴运行固定的圈数，然后根据实际测量的长度计算出实际的主从轴每圈前进距离，并重新输入。对于飞剪，可以借此计算实际的飞剪轮周长和直径

9. 试运行一定数量的周期后，如果定长剪切出现了较为明显的累积误差，可以考虑设置位移补偿系数。在主轴运动方向上，切点滞后（料长长与订单长度），位移补偿系数大于 1，否则小于 1。（建议该值设置时从与 1 相距小于 0.01 开始）

注意事项：

1.失能 ECAM 指令后重新使能才会更新所修改的参数，运行过程中更改参数是不会生效的。

2.注意设置的参数不合理（参数设置章节介绍了参数设置的取值范围和各类限定关系），使能后 PLC 会报凸轮参数错误且不会运行，此时需要检查参数设置是否正确，更改正确后失能重新使能才能继续运行。

3.可以利用同步区标志触发剪切，利用同步区下降沿进行计数。

4.建议加速减速过程尽可能平稳，否则可能会导致机械结构震动过大从而影响编码器的测量，也可能导致从轴速度过高引起伺服超限报警。

5.飞剪的复位和追剪有所区别，飞剪复位的时候回到起点时需要根据切刀当前的位置选择回零路径，注意不能碰到送料轴。

6.注意运行过程中不能清空或者手动修改目标值寄存器，否则很可能会产生因为主轴目标值突变导致从轴脉冲频率超限的错误。

7.离合器标志位置位后，色标检测无效。离合器标志位复位前，必须保证控制该脉冲口的其余脉冲指令均已经失能。

8.如果设置了记忆主轴相位，需要从头开始重新运行凸轮曲线，可以失能 ECAM，清空所有的运行参数，重新使能即可。

3.2 飞剪的实际案例

使用电子凸轮指令前需要对其工作原理有清晰的认识，需要对各个模式下的参数类型和范围有比较清晰的认识，当发生参数配置错误的情况时，参考上一段的注释，修正不合理参数。

案例 1 ：牙签机分切项目

（一）项目背景

牙签塑料包装膜分切机：将用长条形塑料膜包好的若干牙签进行分切，牙签与牙签之间存在一个色标。该项目属于定标飞剪，因此按照调试步骤确定端口配置，机械参数和飞剪参数。

（二）端口配置与飞剪参数确定

1.端口配置

旋切轴（从轴）脉冲输出口和方向口 Y0，Y4，由于客户主轴采用伺服驱动，脉冲端口 Y1，并且主轴压辊上并未安装编码器，目标值寄存器为 D8142（如果有编码器，目标值寄存器为高速计数器相应的寄存器）。

色标传感器 X0，极性常开。旋切轴上有两个切刀原点传感器 X4（一圈内有两次检测），极性常开，使用之前确保传感器信号有效。离合器设置为 M100，接通有效。失能后能够记忆主轴相位。

2.机械参数与飞剪参数：

机械参数：主轴每圈脉冲数 10000pls，主轴每圈前进距离 126mm，从轴每圈脉冲数 10000pls，飞剪轮直径 60mm，两色标点之间距离 92mm，当旋切轴检测到原点信号时，切刀距离剪切处角度 108°，切刀个数 2。要求剪切点即色标点。

飞剪参数：订单剪切长度设 90mm，加速角度θ，色标点距离剪切处的距离 L ，一个周期为 180°

$90 * \theta \ / (360 / 2) + ((108 - \theta) / 360) * \pi \ * 60 = L \ \Rightarrow \ 0.0236 * \theta \ = 56.549 - L$

由于θ 的范围在 0-108° 之间，为了合理起见可以缩小至 30 - 70° 之间，计算出相应的 L 在 55.841 - 54.997mm 之间。设加速角度θ = 50°，得出

L = 55.369 mm，暂取同步区β = 90°（保证同步区角度足够剪切完成）

订单长度与飞剪一个周期走过的弧长距离（周长的一半）相当，所以加减速过程会较为平缓。

（三）创建旋切轴相关的梯形图

1.创建回零指令，如图 3.1 所示



图 3.1

设定电子凸轮从轴脉冲输出口和方向口分别为 Y0、Y4，配置原点信号以及正负极限信号的端口和极性，选择回零的方式和回零方向，根据 zrnr 的说明文档，确定零点的位置。

注意：回零的时候主轴上最好不要有工件，否则就要注意回零路径不能碰到工件。

2.创建手动指令，如图 3.2 所示

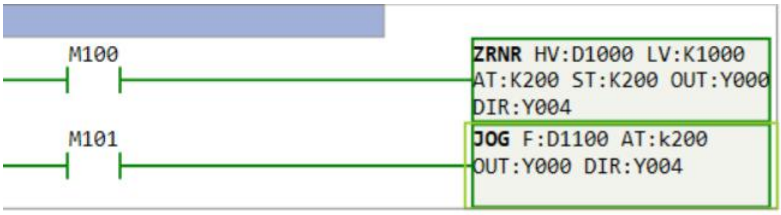


图 3.2

创建 Y0、Y4 的手动指令，用于手动移动电子凸轮从轴。ECAM 离合器接通期间可以使用这些脉冲指令控制旋切轴，离合器失能之前必须保证这些脉冲指令均处在断开状态。

3.梯形图中创建 ECAM 指令，同时设置飞剪的各个参数，如图飞剪 3.3 所示：

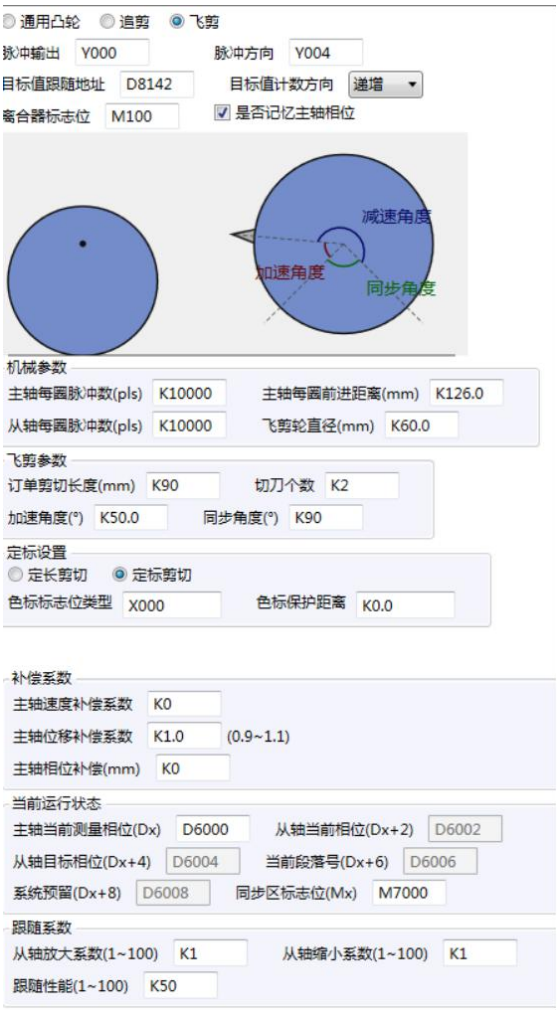


图 3.3

- 端口设置：

脉冲输出：Y0

脉冲方向：Y4

目标值地址：D8142

目标值方向：递增 离

合器标志位：M100

是否记忆主轴相位：是

- 机械参数设置：

主轴每圈脉冲数： 10000pls（主轴伺服旋转一圈所发出的脉冲数）

主轴每圈前进距离： 126mm（主轴伺服旋转一圈主轴实际走过的距离）

从轴每圈脉冲数： 10000pls（追剪伺服电机旋转一圈需要的脉冲数）

飞剪轮直径： 60mm

- 飞剪参数

订单剪切长度： 90mm（剪切料长）

切刀个数： 2

加速角度： 50°

同步角度： 90°

- 定标设置：

选择定标剪切：（检测到色标后进行剪切）

色标标志位：X0

色标保护距离： 0mm（色标检测距离= 订单剪切长度 + 色标保护距离 = 90 + 0 = 90mm）

- 补偿系数：

主轴速度补偿系数： 0（一般设 0 即可）

主轴位移补偿系数： 1（定长剪切需要根据实际运行情况设置，设置范围在 0.9-1.1 之间,定标剪切此处设 1 即可）

主轴相位补偿： 0mm（根据实际需要设置）

- 当前运行状态

D6000 开始的连续地址用于显示运行参数

M7000 开始的连续地址用于显示运行标志位

- 跟随系数：

从轴放大系数： 1

从轴缩小系数： 1

跟随性能： 50（以上参数一般情况下默认即可）

4. 生成的凸轮曲线图

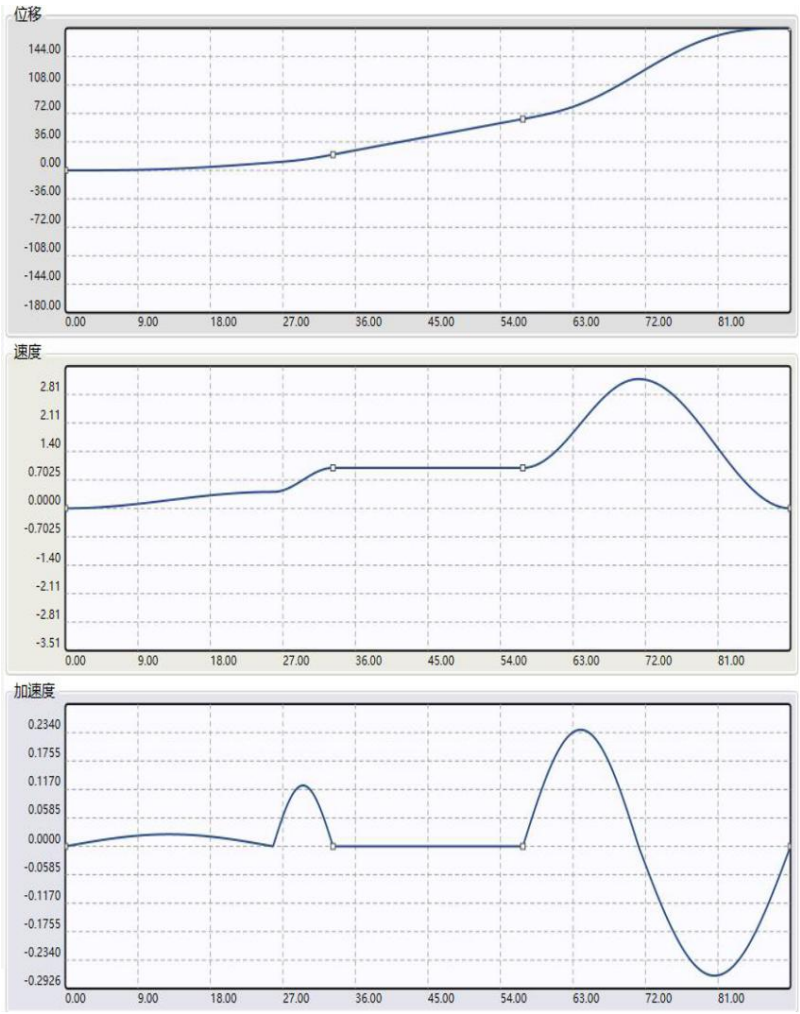


图 3.4

如图 3.4 所示飞剪曲线示意图：其中飞剪在加速区旋切轴速度与主轴速度之比不到 1，减速区速比小于 3，从轴速度在减速区中间较高，本例中可以通过调整同步区或加速区来进一步降低减速区的速度。实际中可能要多次尝试后才能得出一个整体运行平稳的取值。

5. 梯形图逻辑概述

第一次上电时，先调用 ZRNR 指令使得飞剪台回到零点。

确认主轴处于初始位置后，接通 ECAM 指令，进行试运行与试切，观察运行情况，调整机械参数等。

在 ECAM 指令运行中，接通离合器标志位 X1，可以解除主从轴的绑定关系，建议先停止主轴运动再置位离合器标志位。置位离合器标志位后，可以通过 jog 指令移动追剪伺服工作台或者飞剪轮，当 jog 指令失能后，可以复位离合器标志位 X1,此时主从轴重新绑定，并且从离合器标志位置位时的主从轴位置处继续运行。注意接通离合器期间无需失能 ECAM 指令，否则在不记忆主轴位置的情况下会导致主从轴相位关系被清空进而从头开始运行。

由于选择了记忆主轴相位，断电后重新上电会从断电时的位置开始重新运行。

4.15.24 EDRVI/EDRVA

实时调频、实时调位指令

指令说明：

1） 实时调频、实时相对/绝对调位脉冲输出EDRVI/EDRVA 是带加减速的可用于实现指定脉冲频率和脉冲数的脉冲输出指令。启用输入端为“接通”时，脉冲匀加速到设置频率匀速输出，减速时从设置频率减速到0，过程中发送完成指定的脉冲个数。当脉冲指令使能条件断开时，脉冲输出立即停止。在脉冲发送过程中或在脉冲发送结束后但未断开前方使能的情况下，可实时修改脉冲频率和脉冲数，频率调整和脉冲数调整的过程都具有加减速过程。

2） EDRVI/EDRVA 指令可输出频率范围为0Hz~200KHz，脉冲量范围K0~K2147483647。对于FG系列PLC 其最大脉冲频率为500KHz。

3） FGs 提供了高速脉冲输出口，16/32 点晶体管型的PLC，Y0~Y1 为高速脉冲输出口；64 点晶体管型的PLC，Y0~Y3 为高速脉冲输出口；FGRB 晶体管PLC，Y0~Y1 为高速脉冲输出口，FGRS 晶体管PLC，Y0~Y3 为高速脉冲输出口，继电器型的PLC 不具有高速脉冲功能，输出端口的最大输出频率由继电器的吸合时间决定。对于FGM 系列PLC 其高速脉冲输出口为Y0~Y7，Y10~Y11；

4） 指令可用于非对称（加减速时间单独设置）梯形、S 形位置控制，当特殊功能寄存器M8069 ON 时所有的脉冲指令的加减速为S 形，为OFF 时则为梯形。

5） 特殊功能寄存器M8068 为全局方向极性控制，默认脉冲方向输出口为ON 时表示正方向，OFF 时表示负方向。若设置M8068 为ON 则所有脉冲指令的方向输出极性发生反转，脉冲方向输出为OFF 表示正方向，为ON 表示负方向。另外M8192~M8201 分别为Y0~Y7,Y10~Y11 各轴的脉冲方向极性，可对各轴方向极性单独调整。当全局方向极性M8068 为ON 时，则忽略各轴的方向极性设置。

使用说明：

1） 如图1 所示，首先设置频率、脉冲数、加减速时间等参数后接通M0，Y0 开始按照设置参数开始发送脉冲。此时如果不更改频率和脉冲数的值，则指令运行与DRVI 完全相 同。如果在脉冲发送过



图1EDRVI 指令示例

程中更改频率D0(D1)的值，则在扫描周期运行过程检测到D0(D1)变化后，立即根据脉冲数重新计算频率曲线。如果在脉冲发送过程中更改脉冲数D2(D3)， 扫描周期检测到变化后，立即根据新更改的脉冲数位置重新计算频率曲线，脉冲数的更改可能造成电机反转，后续会做详细介绍。另外，如果脉冲发送完成，不断开M0 使能的情况下，更改脉冲数D2(D3)的值，则电机会立即运动到新更改的位置。加减速时间修改无效。

2） 如图2，EDRVI 频率的实时调整曲线图，在t0 时刻开始发送脉冲，经过加速时间(t1- t0)后频率到达设置频率F1，不修改频率情况下，曲线沿着虚线部分继续运行，直到脉冲发送完成，并在t6 时刻结束。如果在t2 时刻，实时修改频率为F2，此时频率会从F1 变速到F2，调速时间为(t3-t2)是设置的加速时间，

并沿着F2 继续运行直到脉冲发送完成或者下次频率更改。调速后t2~t5 所发送的脉冲数，与虚线部分t2~t6 所发脉冲数相同，即调速不影响最终目标脉冲个数。

如图3，如果当前频率未到达稳定频率，或者正处于减速阶段，则曲线会以当前脉冲频率作为起始频率，F2 作为目标频率，剩余发送脉冲数作为总脉冲数进行曲线调整，图2 中是能够到达修改频率F2 的情况，而通常由于剩余脉冲数过少也有无法到达F2 的情况，这时PLC 会自动调整曲线形状并运行。

当只需进行速度调整，且朝一方向一直运行时，当脉冲数设置为H7FFFFFFF 时，脉冲朝正向一直发送，当脉冲设置为H80000000 时，脉冲朝负方向一直发送。若设置为一直发送脉冲，首次导通指令后，再更改脉冲个数无效。

当运行过程中，修改频率为0，则表示暂停脉冲发送，频率曲线会立即从当前频率减速运动到停止。当修改频率为新值时，频率曲线会从零加速运动到新频率，并继续完成脉冲发送。

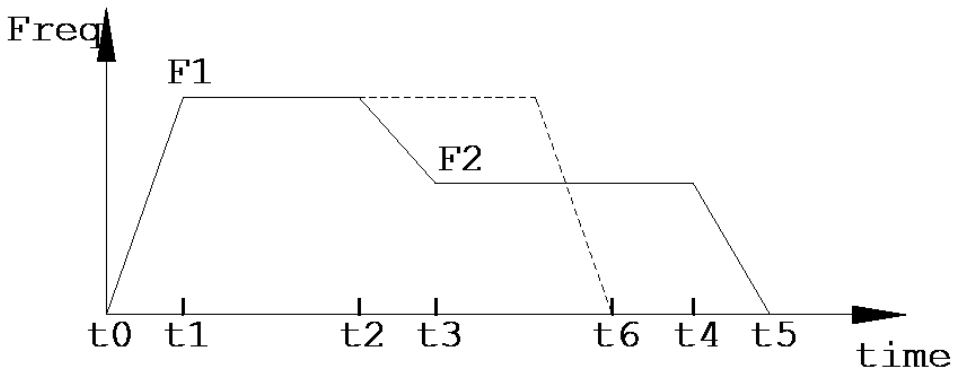


图2频率调整曲线图

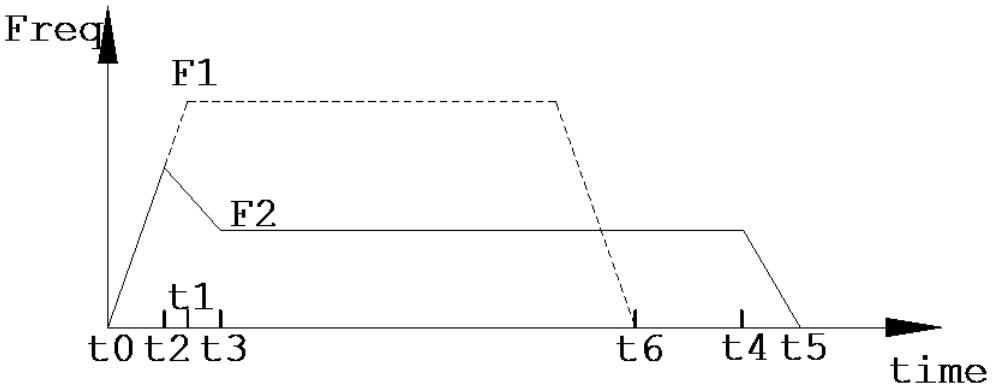


图3加减速阶段频率调整图

3） 实时调整位置有多种用法。如图4，指令导通时工作台在P0 处从静止运动到P1 位置，此时脉冲发送完成。不断开指令前方使能条件，修改目标位置为P2，则指令会马上发送脉冲，从P1 运动到P2，当P2 位于P1 左侧时，也会反向运行，以此类推，当处于静态时可再次调整到其它位置，该调整方式为静态下调整位置。此时需要注意相对指令EDRVI 与绝对指令EDRVA 间的区别。假设图4 中P0、P1、P2 为绝对坐标系下的位置，使用EDRVI 相对指令时，由于起始位置在P0，从P0 运动到P1 脉冲数需要设置为(P1-P0),注意如果要从P1 调位运动到P2，脉冲数需要设置为(P2-P0)，后续所有调整的位置都是基于P0 的相对位置。使用EDRVA 绝对指令时，从P0 运动到P1，脉冲数设置为P1，从P1 调位运动到P2，则脉冲数设置为P2，后续调整的位置都是基于坐标原点的位置。

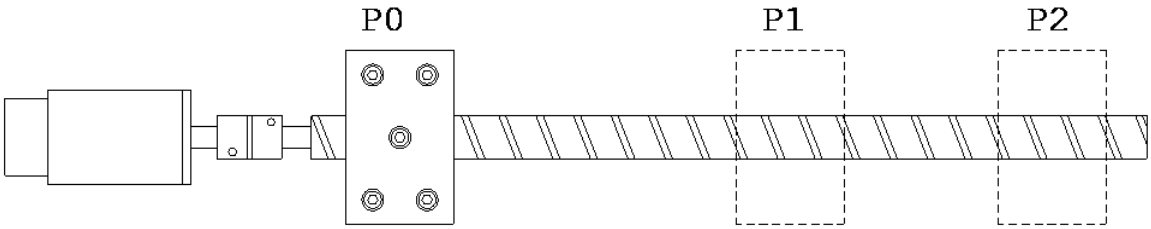


图4 位置调整

4） 如图4，在动态调整位置时，即在P0 运动到P1 的过程中，将目标位置修改为P2，此时有多种情况，对应的运行方式也不同。如图5，不修改目标位置时，从t0 时刻（P0 位置）运动到t3 时刻（P1 位置）。如果在t2 时刻，目标位置被修改为P2 位置，频率不变，则会直接运行到P2 位置，忽略原P1 位置设置。

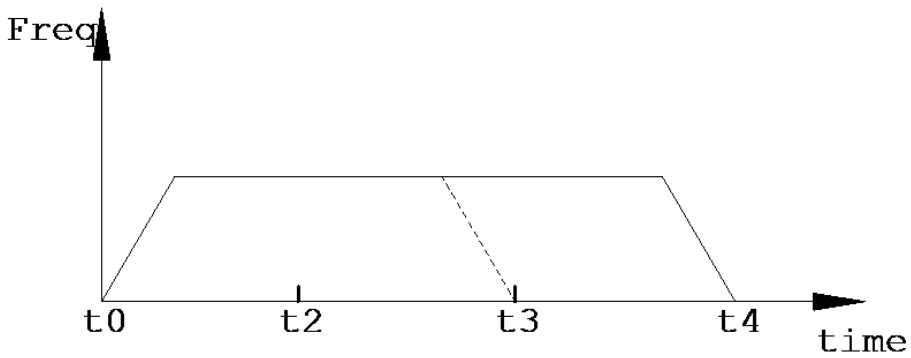


图5动态位置调整

5） 如图6、图7，如果从P0 运动到P1 过程中，当前时刻t2 的位置为P1.5，t2 时刻修改的目标位置为P2，但P2 在P1.5 左侧，则此时脉冲频率会立即沿原方向减速为0，再反向加速启动运动至P2 位置。其速度曲线如图7。除了此情况会造成反向外，当P2 在P1.5 的右侧，但P2 距离P1.5 非常近，不足以按照设置参数正常减速到P2 时，也会先减速到0，再反向运动到P2。其他情况下，如加减速过程中修改位置等，都是以当前频率与目标频率和当前位置与目标位置计算合理的速度曲线运行。

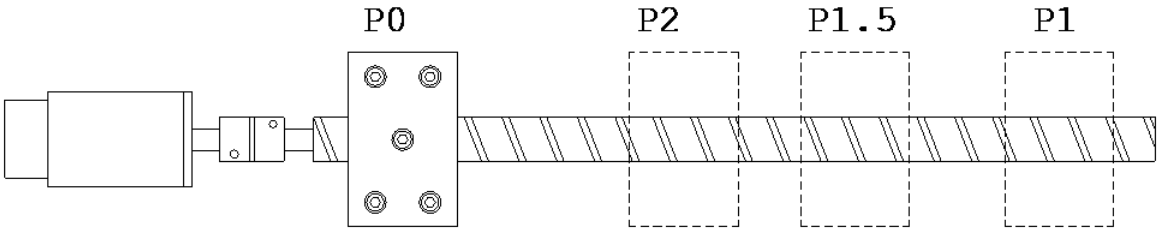


图6 反向位置调整

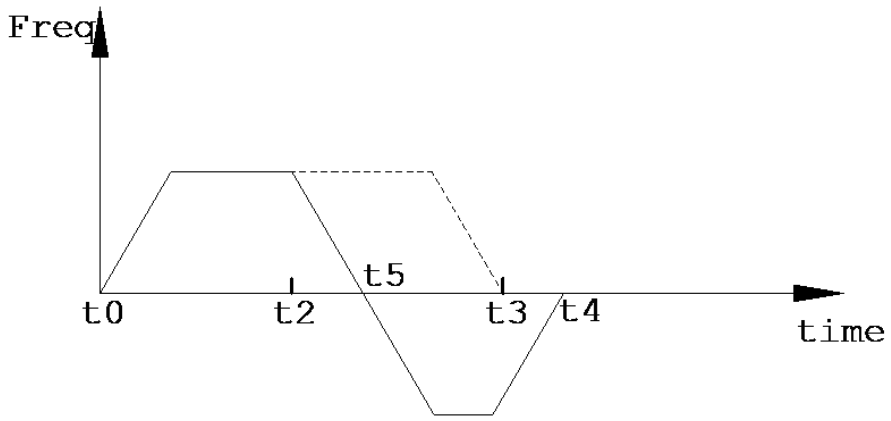


图7反向位置调整速度曲线图

脉冲正在输出标志(只读)：

寄存器	脉冲输出 Yn
M_8134	Y000
M_8135	Y001
M_8136	Y002
M_8137	Y003
对于 FGM 型 PLC 还具备以下 M 软元件	
M_8138	Y004
M_8139	Y005
M_8140	Y006
M_8141	Y007
M_8142	Y010
M_8143	Y011

脉冲累积计数(读写)：

寄存器	脉冲输出 Yn
D_8140(D_8141)	Y000
D_8142(D_8143)	Y001
D_8144(D_8145)	Y002
D_8146(D_8147)	Y003

对于 FGM 型 PLC 还具备以下寄存器	
D_8148(D_8149)	Y004
D_8150(D_8151)	Y005
D_8152(D_8153)	Y006
D_8154(D_8155)	Y007
D_8156(D_8157)	Y010
D_8158(D_8159)	Y011

是否忽略错误，继续发送脉冲，默认忽略为0，若设为1，出错时停止脉冲输出(读写)：

寄存器	脉冲输出 Yn
M_8070	Y000
M_8071	Y001
M_8072	Y002
M_8073	Y003
对于 FGM 型 PLC 还具备以下寄存器	
M_8074	Y004
M_8075	Y005
M_8076	Y006
M_8077	Y007
M_8078	Y010
M_8079	Y011

注意：

1．对于高速脉冲输出，它是针对外部高速器件的，要对脉冲进行计数，只能使用高速脉冲输入计数器，不能使用内部计数器，也不能使用 Y 边沿变化进行计数，当作为高速输出时，不能作为普通输出口用；

2．晶体管的 OFF 时间具有在轻负载时会延长的特性。所以请在需要响应性时设计成让负载在较轻时加负载电阻，增加负载电流；

3．对于脉冲输出 Y0，寄存器 D8140（D8141）中累计其脉冲个数，D8141 存储高 16 位，D8140 存储低 16 位；

4．脉冲累积计数寄存器（D8140~D8158）是重要寄存器，可读写，当写入新值时，计数会在新值基础上进行加减；

5．由于本指令脉冲输出带方向，所以脉冲累积计数寄存器（D8140~D8158）根据方向计数；

6．高速脉冲输出时，脉冲累积计数寄存器（D8140~D8158）内的值是不连续的、时刻变化的，用于判断时，请使用大小比较，不要使用等于判断；

7．当脉冲指令使用到输出口 Y 时，该 Y 口不能用做它用，即普通指令无法再对该 Y 口进行 ON、OFF 操作。

指令的有效操作数

输入输出	类型	操作数	描述
EN	BOOL		**启用输入端接通时，指令执行；否则，不执行**
输出频率	双整数	**K/H/D**	**输出频率的设定值**
输出脉冲数	双整数	**K/H/D**	**输出脉冲数的设定值**
加速时间	整数	**K/H/D**	**加速时间设定值**
减速时间	整数	**K/H/D**	**减速时间设定值**
输出脉冲 Y	BOOL	**Y**	**指定的脉冲输出口**
输出方向 Y	BOOL	**Y**	**指定的脉冲方向输出口**

4.15.25 SPD

脉冲密度测定 SPD

对应命令	注释/说明	操作数（寄存器）
SPD（32 位）	脉冲密度测定	脉冲输入 X， 测定时间 K/H/D，测定结果 D

1. 指令说明

指令对指定时间内的高速输入脉冲进行计数，将结果存储到设定的一片寄存器内，该指令的测定结果将占用连续 6 个 D 寄存器。指令将测定时间内记录的脉冲个数保存到 D 寄存器（32 位）中，将当前测量周期记录的脉冲个数保存到(D+2)寄存器（32 位）中，将当前测定周期剩余时间保存到(D+4)寄存器（32 位）中。

对于型号 FGm_64MT_A，可用的脉冲输入口为 X000-X013，支持的最高脉冲输入频率为 500KHz；对于型号 FGRS_C8X8T，FGRE_C8X8T， FGs-64MT_A，FGs-64MT_R，SK070_32MAA，可用的脉冲输入口为 X000-X003，支持的最高脉冲输入频率为 200KHz；对于其他型号，可用的脉冲输入口为 X000-X001，支持的最高脉冲输入频率为 200KHz。

当某一输入口（例如：X000）被某一脉冲密度测定指令使用并且使能时，不可同时使能该输入口的高速计数器（例如：C235，C241，C244，C246 等），否则高速计数器将不能正常工作。每个输入点只能使用一条脉冲密度测定指令。

测速范围：-2,147,483,648 ~ +2,147,483,647；当设定时间内的输入脉冲数超出此范围时，会产生上溢或下溢现象。测量的精度和测量时间相关，测量时间越长，精度越高，当测量时间是 1 秒时，测量精度可以达到万分之一以内。

注意：

1) **当使用了该指令后，这些端子不能再同时作为其他计数器的输入用，不能作为外部中断使用，也不能作为普通输入方式使用；**
2) **在指令使能后更改测量时间的值，不会立即更新当前测速周期的值，而是按照之前的测量时间完成一次测速，从下一个测量周期开始生效；**
3) **当使用高速计数输入或中断输入时，建议相应输入端口的线缆采用双绞屏蔽线，并将屏蔽层接地(同端子连接或连接信号地)，以提 高抗干扰性；**
4) **晶体管的 OFF 时间具有在轻负载时会延长的特性。所以请在需要响应性时设计成让负载在较轻时加负载电阻，增加负载电流；**

2. 指令的有效操作数

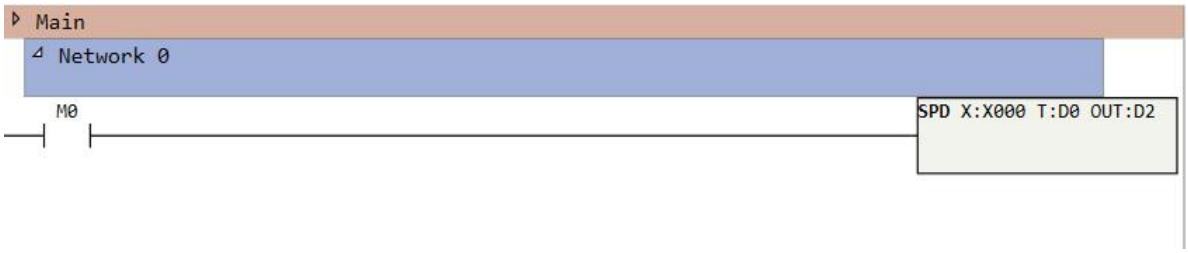
输入输出	类型	操作数	描述
EN	BOOL		**启用输入端接通时，指令执行；否则，不执行**
脉冲输入 X	BOOL	**X**	**待测定的脉冲输入**
测量时间 T	双整数	**K/H/D**	**测定时间（ms）**
结果 OUT	双整数	**D**	**存储测定结果的起始寄存器**

3. 实例

指令表：

```
Network 000
LD      M0
SPD     X000    D0    D2 //测量 X0 端口再 D0 毫秒内的脉冲个数，将结果保存到 D2 中，当前测量周期记录的脉冲个数保存到 D4 中，当前测量周期的剩余时间保存到 D6 中
```

梯形图：



4.15.26 TECAM

增强型电子凸轮 TECAM

适用机型：FGRS_C8X8T、FGRE_C8X8T、FGm_64MT_AC。

该指令由 TECAM_DATASET 和 TECAM 两部分组成，TECAM_DATASET 是电子凸轮表格，是用来设置主从轴的机械参数和凸轮的基本参数，并生成凸轮曲线，目前最多设置两个表格，但支持运行中动态修改（下个周期生效）；TECAM 是电子凸轮指令，是用来设置运行参数的，电子凸轮指令是可以多次使用的，但是要保证运行的地址不相同。从轴只受限于 PLC 的脉冲口数量（但 FGRS_C8X8T/FGRE_C8X8T 最多只能为 3 个从轴提供纠偏信号）。电子凸轮采用凸轮表格的优势是设置好参数后先计算，后面 TECAM 指令直接运行，而不是边走边运算，这种方式一是减少占用的内存，二是提高运行效率，直接查表运行而不用等待运行结果。与 ECAM 指令相比，跟随刚度更高，功能更丰富，使用更便捷，但不支持主轴倒转和主轴速度不连续的凸轮曲线。

1.TECAM_DATASET 指令说明

TECAM_DATASET 有四种模式，分别是飞剪、追剪、通用凸轮（关键点）和通用凸轮（插入点），在初次调用该指令的时候就需要选择模式，不同模式下设置的参数不一样，需要根据实际情况进行设置。TECAM_DATASET 的各地址含义如下表

1.1 数据参数首地址 D，其余地址依次加首地址

偏移地址	描述	数据类型	数据范围	说明
0	电子凸轮表数据建立方式/数据修改起始点索引（中间点模式和关键点模式）	Int	0-9999/ 10000-10049/ 10050/10051	0-9999 为中间插值点输入 10000-10049 为关键点输入 10050　飞剪 10051　追剪
2	数据深度	Int	0-2048/0-50	中间插值点时最多为 2048 关键点输入时最多为 50 数据修改时与索引之和不超过上述值 追飞剪数据输入模式下无效
4	主轴倍率	Float		主轴比例系数，仅关键点模式，默认为 1
6	从轴倍率	Float		从轴比例系数，仅关键点模式，默认为 1
8	数据单位	Int	0,1	0:pls 1:mm 仅关键点模式
10	主轴每圈移动距离	Float		主轴电机转一圈所前进距离

12	主轴每圈脉冲数	Int		主轴电机转一圈所需要的脉冲数
14	从轴每圈移动距离 （飞剪时为飞剪轮直径）	Float		从轴电机旋转一圈所前进的距离： 追剪可能为丝杆的导程， 飞剪为飞剪轮的直径
16	从轴每圈脉冲数	Int		从轴转一圈所需要的脉冲数
18	从轴额定转速	Int		0：默认 190kHz 中间点或插值点输入,单位 Hz 追剪或飞剪输入,单位 r/min
20-28	预留			
30	电子凸轮数据建立错误	Int		(int32_t)1 //凸轮表数据建立方式和数据深度错误 2 //每圈距离或脉冲数错误 3 //切刀个数不合法 4 //剪切角度不合法 5 //同步区超过总剪切长度 6 //从轴额定速度超过了限制 7 //各区长度设置异常 8 //剪切总长不足 9 //同步区超出丝杠行程 10 //主轴数据存在倒转或停滞 11 //数据索引或深度错误 12//电子凸轮表正在占用中
32	从轴额定脉冲频率	Int	只读	由额定转速计算，单位 Hz
34	主轴理论最大速度上限	Int	只读	根据从轴额定频率和最大主从轴速比算出，单位 Hz
36	主从轴同步区倍率	Float	只读	由主轴脉冲当量/从轴脉冲当量所得，插值点输入无效
38	最大主从轴速比	Float	只读	根据曲线算出主从轴最大速比，插值点输入无效

1.2 数据存储首地址 D，其余地址依次加首地址

飞剪、追剪、通用凸轮（关键点）和通用凸轮（插入点）占用的数据存储长度不一样，具体占用寄存器的数量和每个寄存器的含义如下说明。

1.2.1 飞剪输入

偏移地址	描述	数据类型	数据范围	说明
0	主轴剪切长度	float		剪切长度
2	同步区角度	float		单位 °，从轴同步区长度
4	切刀个数	Int		切刀个数:一个主轴周期对应几个从轴周期，默认为 1
6	加速曲线选择	Short		0:恒加速(二次曲线) 1:三次曲线 2:双段

				斜余弦曲线
7	电子凸轮曲线选择	Short		暂未开放
8	同步区起始角度	float		从轴同步区起点，与同步区角度之和要小于从轴总长；恒加速无效
10	主轴同步区起点	float		主轴同步区起点(仅三次曲线可用)，与同步区距离之和要小于主轴总长；恒加速无效
12	原点速比	float		凸轮曲线首尾的斜率

恒加速模式下同步区在整个行程的中间，两侧的速度调整区大小是对应的，自动计算完成；
加速曲线 1 和 2 需要通过同步区起始角度/主轴同步区起点/原点速比来控制线形，目前上位可直接显示三次曲线；
切刀个数:如果填 2，原本的一个从轴周期被拆分成了两个周期。若原本一个周期转一圈，拆分之后时一个周期转半圈。

参数设置要求：
同步区起始角度+同步区角度 ＜ （360°/ 切刀个数）
同步区起点+同步区长度 ＜ 剪切长度

1.2.2 追剪输入

偏移地址	描述	数据类型	数据范围	说明
0	主轴剪切长度	float		剪切长度
2	主轴加速区长度	float		
4	主轴同步区长度	float		
6	主轴减速区长度	float		
8	主轴换向区长度	float		
10	主轴后等待区长度	float		追减末尾的等待距离
12	从轴极限行程	float		从轴能够运动的极限距离
14	剪切时间	Int		从轴用于剪切的时间 ms
16	前等待区长度	float		追减起始的等待距离

参数设置要求：
主轴前等待区+加速区+同步区+减速区+换向区+后等待区< 主轴剪切长度，因为还有返回区，返回区是该不等式右边减去左边的差值。
追剪从轴加速距离+同步区+减速区 ＜ 追剪从轴极限距离，为了保证从轴不越界，保护从轴；当设置的参数不满足该条件时，会自动调整加速区和减速区距离，使总长小于从轴极限行程；如果同步区距离已经超过极限距离，则凸轮曲线生成失败，偏移地址 D30 会报错提示。

1.2.3 通用凸轮（关键点）

点编号	偏移地址	主轴坐标	横轴坐标	斜率（速比）	线型	解析度
0	0	0	0	0		0
1	12					
2	22					
3	32					
...	...					
49	...					

- 第一个关键点只有速比有效，其余皆为 0，数据深度必须>=2
- 每个关键点包含主轴坐标，从轴坐标，该点到前一个点的线型，该点斜率，该点到前一个点的解析度，共占 10 个双字寄存器
- 线型有直线、二次曲线、三次曲线和双段斜余弦曲线（对应 0 1 2 3 4）
- 数据深度最大 50
- 解析度之和不超过 2048
- 只能生成速比连续的凸轮曲线（即每个点只能有一个斜率），起始点斜率与终止点不同时，在周期末会有速度突变

1.2.4 通用凸轮（插入点）

点编号	偏移地址	主轴坐标	横轴坐标
0	0	0	0
1	4		
2	8		
3	12		
...	...		
2047	...		

- 第一个点必须为 0，0，数据深度必须>=2
- 每个关键点包含主轴坐标和从轴坐标（脉冲数），占 2 个双字寄存器
- 数据深度最大 2048

1.3 数据写入标志首地址 M

偏移地址	描述
0	电子凸轮表刷新标志位
1	电子凸轮表写入成功标志位

(首地址 M+0)标志位上升沿有效，写入成功则写入成功标志位(首地址 M+1)置 ON。
当 TECAM_DATASET 使能时，M0 并不被置位，需要手动置位，也可以取一个上升沿进行触发。
当 TECAM_DATASET 失能时，这两个标志位均会复位掉。
修改凸轮曲线时，需要给出刷新标志位的上升沿，才会开始计算新曲线，计算完成后写入成功标志位会置 ON。

2.TECAM 指令说明

TECAM 指令是直接运行 TECAM_DATASET 里算出的数据，运行参数需要在 TECAM 中运行，比如确定主轴和跟随的从轴是在该指令中进行确定的，确定是定长还是定标剪切也是在该指令的运行参数中进行设置的。

2.1 运行参数首地址 D，其余地址依次加首地址

偏移地址	描述	数据类型	寄存器类型	数据范围	说明
0	电子凸轮表序号	Int	D	0，1	目前只能填 0 和 1
2	凸轮错误状态	Int	D		只读。凸轮错误状态提示 (uint32_t)0x00000001 //电子凸轮指令 指定的凸轮表编号不存在 0x00000002 //电子凸轮指令 指定主轴来源不存在 0x00000004 //电子凸轮指令 主轴索引号非法 0x00000008 //电子凸轮指令 纠偏信号不存在 0x00000010 //电子凸轮指令 启动信号不存在 0x00000020 //电子凸轮指令 启动信号索引不存在 0x00000040 //电子凸轮指令 对应的电子凸轮表未初始化 0x00000080 //电子凸轮指令 飞车脉冲限值非法 0x00000100 //电子凸轮指令 相位调整值未设定 0x00000200 //电子凸轮指令 相位调整次数未设定 0x00000400 //电子凸轮指令 原点信号补偿比例异常 0x00000800 //电子凸轮指令 存在启动信号时运行周期数未确定 0x00001000 //电子凸轮指令 单周期纠偏信号触发次数未设置 0x00002000//电子凸轮指令 飞车超限 0x00004000//主轴相位异常突变 0x00008000 //启动信号触发异常 0x00010000 //纠偏信号触发异常 0x00020000 //纠偏累计距离过长 0x00040000 //主轴存在倒转数据

					0x00080000//方向口无效
4	主轴累积脉冲数	Int	D		只读。主轴开始到运行期间累计的相位累积，主轴相位调整量/周期跳转量/纠偏调整量都会被计入
6	主轴当前相位	Int	D		只读。主轴当前走的位置，走一圈后会重新开始计数 目前 10ms 刷一次
8	从轴累积脉冲数	Int	D		只读。从轴启动后累计走的脉冲数，返回原点后归零
10	从轴当前相位	Int	D		只读。从轴当前走的位置，返回后会重新开始计数
12	从轴理论目标累积脉冲	Int	D		只读。与从轴累计脉冲数一致
14	主轴当前速度	Int	D		只读。主轴开启后的运行速度，如果是匀速的话该数值一直不变。单位 Hz
16	主轴理论最大速度	Int	D		只读。对应电子凸轮表
18	从轴当前速度	Int	D		只读。不同的区对应不同的速度，同步区内从轴速度和主轴当前速度应该是一致的。 单位 Hz
20	从轴额定频率	Int	D		只读。对应电子凸轮表
22	从轴当前脉冲发送值	Int	D		只读。乘 100 就是当前从轴的频率
24	从轴当前段落号	Int	D		只读。 追剪： 1.前等待区 2.加速区 3.同步区 4.减速区 5.换向区 6.返回区（匀加速段） 7.返回区（匀减速段） 8.后等待区 飞剪： 长料： 1.等待区 2.加速区 3.同步区 4.减速区 短料： 1.减速区 2.同步区

					3.加速区 关键点模式： 以关键点段落计数 中间点模式： 无效 当某一区长度为 0 时，后面的段落号顺序前移。
26	总计运行周期数	Int	D		只读。 从轴进入周期起始段时计数一次，已运行周期总数
28	纠偏补偿总量	Int	D		只读 纠偏信号处主轴偏移脉冲数，用于速度补偿的理论值为主轴频率×0.01，偏离该值的为机械纠偏脉冲数
30-46	预留				
48	主轴类型	Int	D	1,2,3	0:无效 1:实际主轴 Y 2:高速计数器 CV 3.虚拟主轴 Vy（D0-D8000）
50	主轴位置索引	Int	D		FGRS_C8X8T/FGRE_C8X8T： D48 为 1 时可选 0-3；为 2 时可选 0-21（对应 CV235-CV255）；为 3 时可选 0-8000 FGm_64MT_AC： D48 为 1 时可选 0-9（对应 Y0-Y11）；为 2 时可选 0-11（对应 CV235-CV246）；为 3 时可选 0-8000
52-70	预留				
72	补偿信号	Int	D	0,1,2	0:无 FGRS_C8X8T/FGRE_C8X8T： 1：补偿信号 X6 2：补偿信号 X0 3: 补偿信号 X1 FGm_64MT_AC： 1：补偿信号 X20 2：补偿信号 X21 3：补偿信号 X25 4：补偿信号 X26 或 X30 5：补偿信号 X27 或 X31

					6：补偿信号 X32 7：补偿信号 X33 8：补偿信号 X34 9：补偿信号 X35 10：补偿信号 X36 用于从轴的纠偏 启用后不可再用于其他外部中断功能或高速计数功能
74	凸轮启动信号	Int	D	0,1,2	0:无 定长剪切 1:M 2:X 定标剪切 启动信号发生变化时： 定标剪切会在执行够设定周期数后切换到定长；定长剪切会在下个周期切换到定标
76	凸轮启动信号地址	Int	D		启动地址（十进制，例如 0 代表 M0\X0，9 代表 M9/X11,具体哪个由启动信号类型决定）
78	单周期纠偏信号触发次数	Int	D		主轴单周期会触发补偿信号的次数 飞剪一般是 1， 追减一般是 2
80-96	预留				
98	凸轮执行周期数	Int	D		0:循环执行 >0:执行周期次数，支持动态更改
100	凸轮飞车限值	Int	D		乘 100 就是限定的脉冲频率 防止凸轮脉冲频率过高 限定从轴的速度不能过快
102	主轴调整相位	Int	D	（-主轴剪切长度/2，主轴剪切长度/2）	设置好后使能需要结合下表中（M+4）相位调整使能标志位使用
104	主轴相位调整步数	Int	D	>=2	与主轴相位调整功能共同使用，越小调整速度越快
106	纠偏信号比例系数	float	F		只有开启了纠偏信号才有用， 针对从轴的纠偏信号，越大纠偏幅度越明显 建议设置的值略大于 1
108	主轴相位采样放大倍率	float	F	>0	默认采样倍率为 1，填入值后采用的主轴相位会乘以该倍率作为新的相位，可用于微调剪切长度和主从轴的同步区速比

110-126	预留				

2.2 状态标志位首地址 M，其余地址依次加首地址

偏移地址	描述	说明
0	电子凸轮报警标志位	只读。电子凸轮运行错误标志 0：无错误 1：报错
1	电子凸轮初始化完成	只读。 0：未完成 1：完成
2	主从轴啮合标志	只读。 0：未完成 1：完成
3	周期结束标志位	只读。
4	相位调整使能标志位	只写。需要每次手动触发 0：未使能 1：使能，可在线修改主轴相位，调整的相位值取决于上表中的（运行参数首地址 D+102）主轴调整相位
5	电子凸轮停止方式标志位	只写。 0：立即停止，从轴立刻停下 1：周期结束停止，从轴回到等待区后停止
6	失能后数据是否保持	只写。 0：不保持，运行参数被清掉 1：保持数据，运行参数保持当前数
7	周期跳转标志位	限定速度和相位为 0 时才能跳转，主轴相位会直接跳转至下一周期开头，直接进入下周期
8	纠偏初始化信号	上升沿触发，可在纠偏异常触发后重新读取基准位置
9	错误码清除信号	上升沿触发，可清楚历史报警错误码，刷新错误码状态
10-31	预留	只读

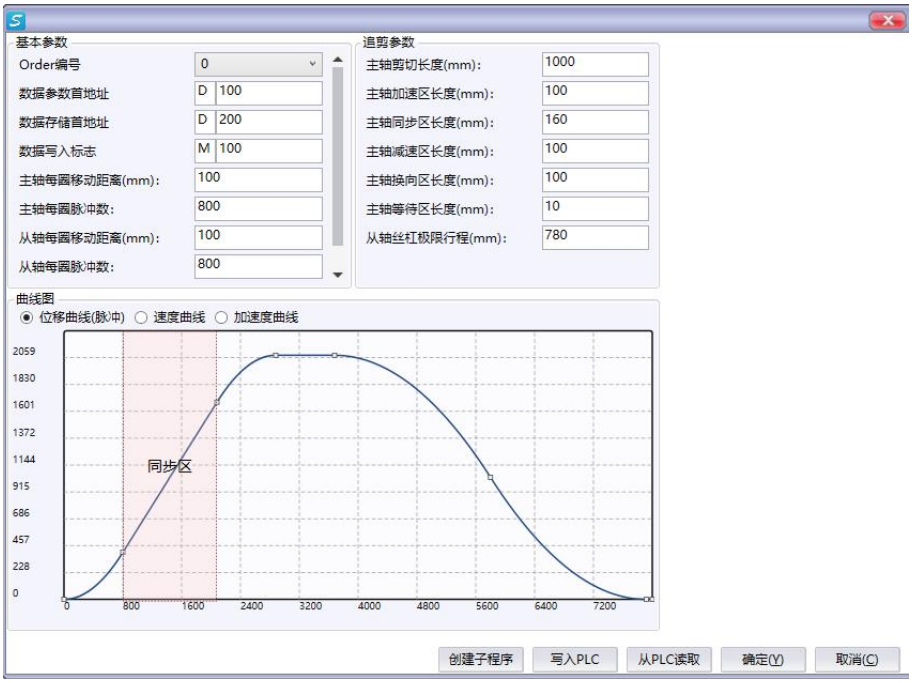
3. 范例

3.1 定长追剪范例

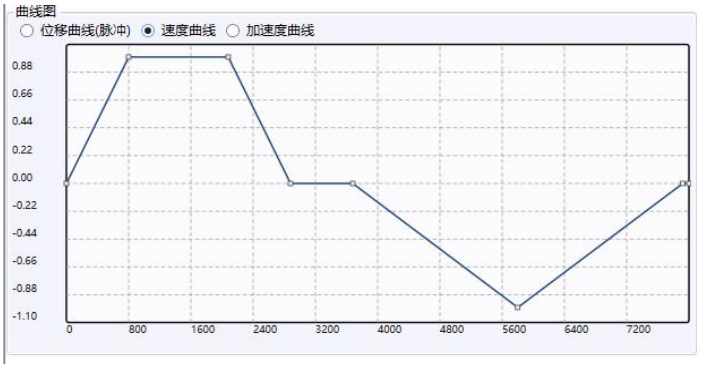
3.1.1 TECAM_DATASET 设置

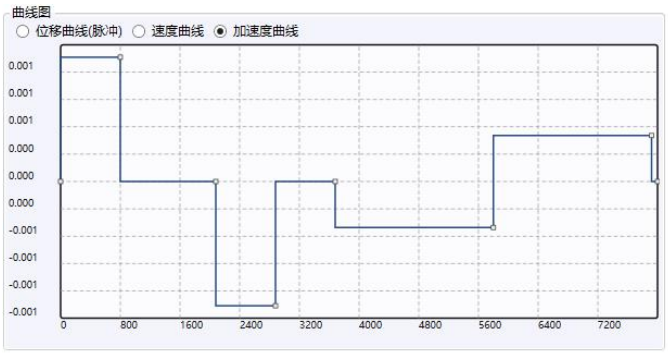
Order 编号可选 0 或 1，表示第一个或第二个凸轮表，范例填 0，此后的 TECAM 可以调用凸轮表里的数据，
数据参数首地址：填写一个 D 寄存器，范例填 D100，该数据会占用掉从 D 到 D+39 共 40 个寄存器
数据存储首地址：填写一个未用过的 D 寄存器，范例填 D200，占用的寄存器数量详情翻上 1.2

数据写入标志：填写一个 M，范例填 M100，占用两个 M 寄存器
主轴每圈移动距离：100，忽略掉中间全部传动机构，实际主轴转一圈，机械走的距离
主轴每圈脉冲数：800，根据电机的细分或者伺服的电子齿轮比得出，最好设置成偶数或者 1000 的倍数
从轴每圈移动距离：100，忽略掉中间全部传动机构，实际从轴转一圈，机械走的距离
从轴每圈脉冲数：800，根据电机的细分或者伺服的电子齿轮比得出，最好设置成偶数或者 1000 的倍数
从轴额定转速：默认 0 就是按照脉冲最高 190kHz 计算；如果是追飞剪，也可以设置成 2000，单位默认 r/min



主轴剪切长度(mm)：1000，float 型，整个追剪过程（从轴启动到从轴返回等待的期间内）主轴走过的长度
主轴加速区长度(mm)：100，float 型，从轴从 0 加速到和主轴同速时主轴走过的距离
主轴同步区长度(mm)：160，float 型，从轴和主轴同速时间内主轴走过的距离
主轴减速区长度(mm)：100，float 型，从轴从和主轴同速减速到 0 时主轴走过的距离
主轴换向区长度(mm)：100，float 型，从轴减速到 0 到再次反向启动前主轴走过的距离
主轴等待区长度(mm)：10，float 型，从轴回来速度降为 0 到下一个周期启动时主轴走过的距离
从轴丝杠极限行程(mm)：780，float 型，从轴在追剪过程中所能走的最大行程，用于从轴的保护
主轴返回区长度(mm)：（530），float 型，主轴返回区长度=主轴剪切长度-主轴加速区长度-主轴同步区长度-主轴减速区长度-主轴换向区长度-主轴等待区长度，返回区有两段，一段是匀加速，另一段是匀减速；如需要从轴快速返回，则将等待区长度设置大一点即可。



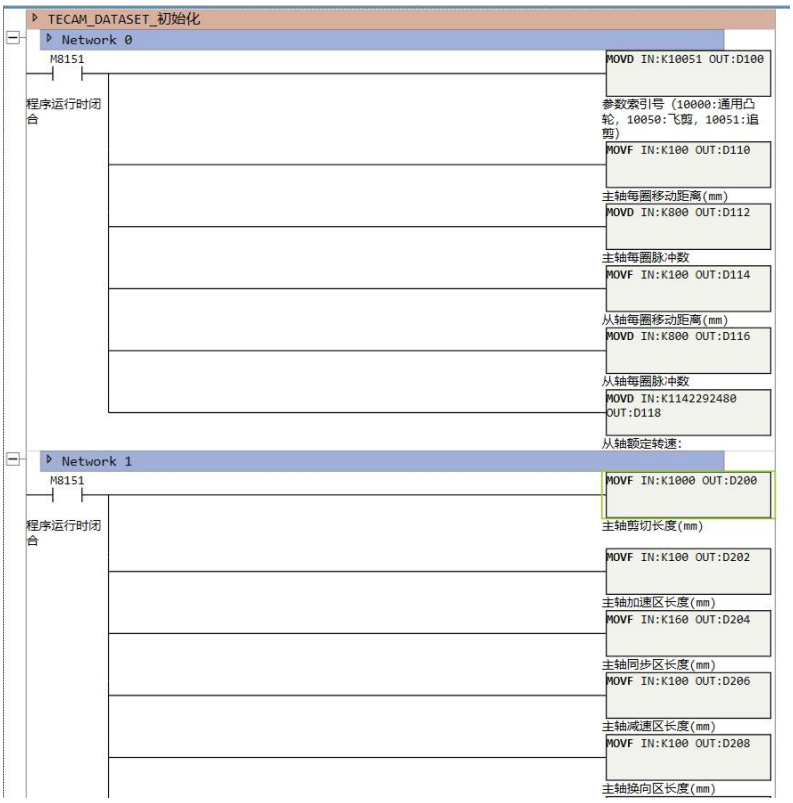


有三个曲线图，位移曲线、速度曲线和加速度曲线，分为加速区、同步区、减速区、换向区、返回区（2 段）和等待区。

点击创建子程序，创建子程序成功后，若需修改原有参数，则要再次创建，第二次创建会将新的数据覆盖下去

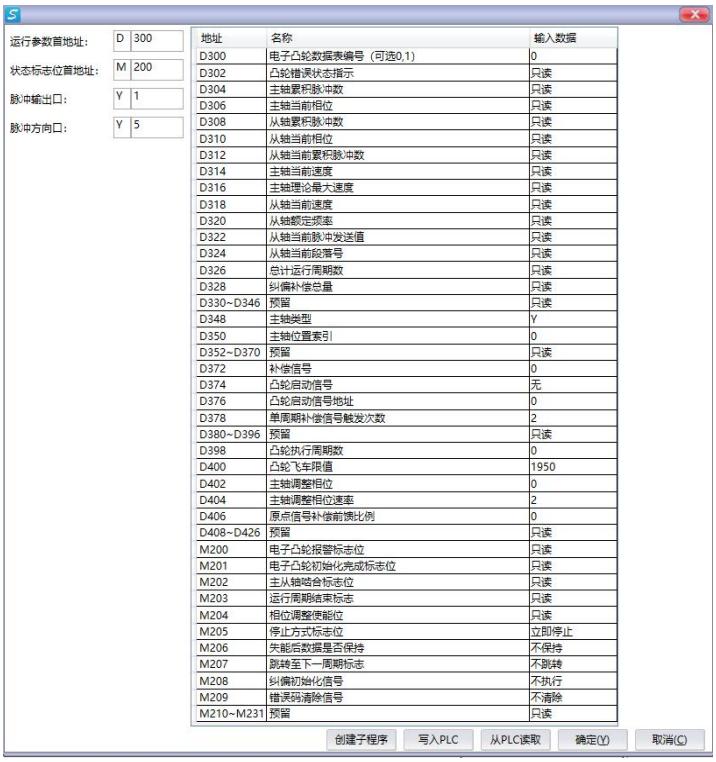


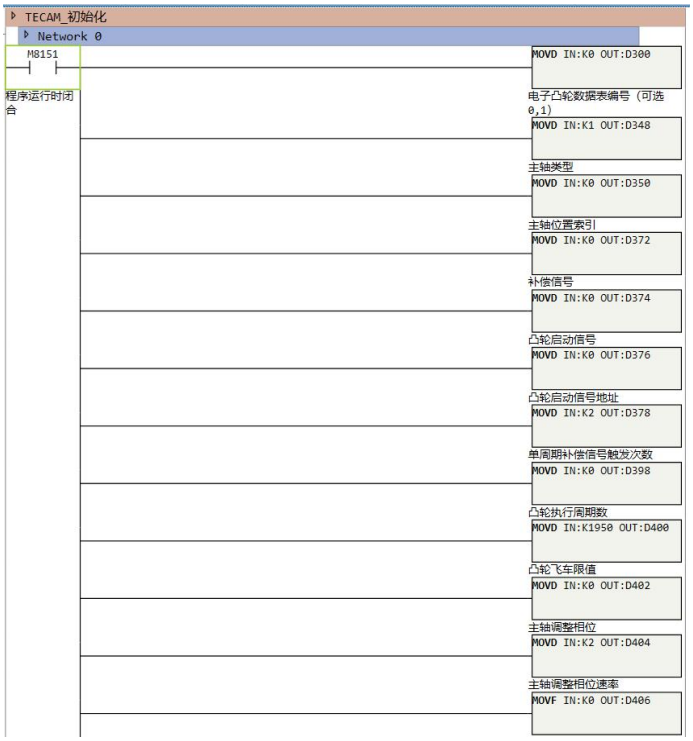
点击确定以后，得到一个新创建的子程序，



至此，TECAM_DATASET 凸轮表格就已经设置完成，后续该子程序需要在 main 中进行调用。

3.1.2 TECAM 指令设置界面





脉冲输出口：追剪跟随的从轴的脉冲输出口
脉冲方向口：追剪跟随的从轴的脉冲方向口

主轴类型：A.跟随的主轴脉冲输出口选 Y（立即刷新），本质是跟随 PLC 内部 Y 口对应的脉冲计数
B.若是跟随伺服的 AB 相返回的脉冲数选 CV（脉冲的 AB 相要差分转集电极后，接到 PLC 的高速计数端口上），这样会占用掉两个高计口，若是还需要启用纠偏信号（X0,X1,X6），可用的纠偏信号可能只剩下一个；如果伺服的响应速度够快且不出现丢脉冲(伺服脉冲频率低于 180kHz)的情况，一般建议选取 Y
C.若是跟随虚拟主轴选 Vy（即刻读取 D 寄存器数据）
附注：差分转集电极方法

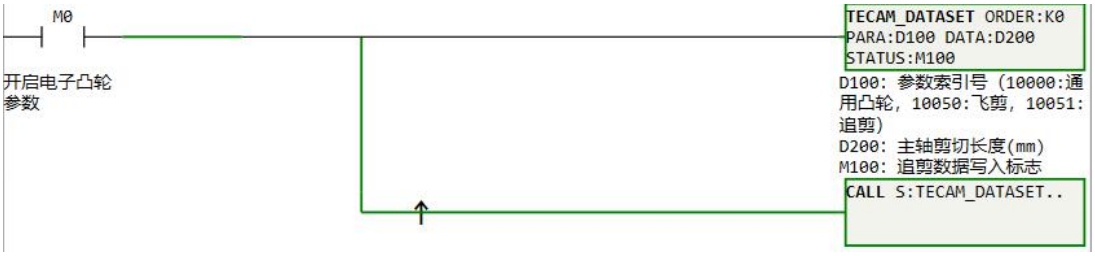
1）伺服的 AB 相差分输出是 5V 的信号，有 A+、A-、B+、B-四根线
2）PLC 的 X 输入端要求 24V 信号，高数计数的双路计数只有两根线接入
3）为了适配伺服的输出，必须加一个差分转集电极的电路板，将 5V 转成 24V，同时四根线转成两根线 A 和 B 输出，此时 A 接 X0，B 接 X1，24V 接 S/S，调用 EHCNT 即可完成对伺服编码器的脉冲计数

输入信号	编码脉冲信号	路数	1 组 A, /A, B, /B
		最大频率	200KHz
		电平	TTL
		电压	5V
		物理接口	凤凰端子
输出信号	编码脉冲信号	路数	1 组 A,B
		频率	与输入相同
		电平	NPN
		电平	24V
		电流	25mA
		物理接口	凤凰端子
供电电源	24 Vdc		

主轴位置索引：确定具体轴，若主轴类型选 Y，主轴位置索引填 0 代表 Y0，填 1 代表 Y1，以此类推
凸轮启动信号：定长追剪选 无
凸轮启动信号地址：定长追剪默认 0
单周期补偿信号触发次数：一般是 2（设置补偿信号后才启用）
凸轮执行周期数：默认 0，循环执行
凸轮飞车限值：设置 1950
主轴相位调整：0，不调整相位
主轴调整相位速率：2
停止方式标志位：断开 TECAM 后从轴的停止方式，立即停止选 OFF，周期结束后停止选 ON
失能后数据是否保持：断开 TECAM 后运行参数是否保持断开前的状态，不保持选 OFF，保持选 ON

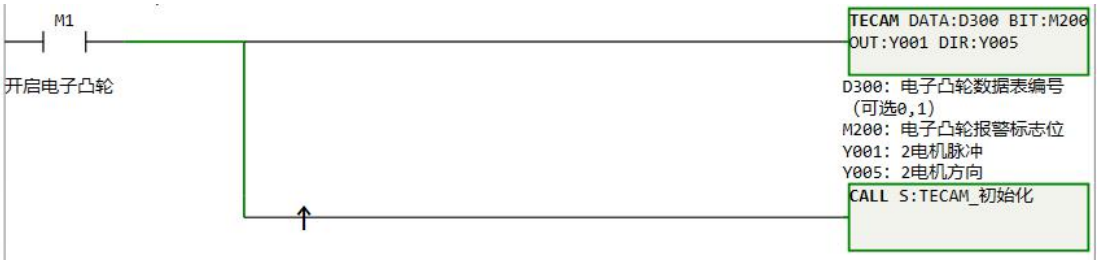
3.1.3 TECAM_DATASET 和 TECAM 的使用流程

A．下载程序成功后，进入监视
B．开启 M0，使能 TECAM_DATASET（如果不想下载程序后还在上位点击写入 PLC，就直接创建一个子程序，并直接调用就可以了；若修改了 TECAM_DATASET 里的数据，子程序的也需要修改，方便的做法是删去原子程序，重新创建一个子程序再进行调用）



C．将数据写入标志位 M100（在 TECAM_DATASET 中）置为 ON，若标志位 M101（在 TECAM_DATASET 中）变为 ON 则写入成功，可以进行下一步；若 M101 仍是 OFF，则说明写入不成功，需要断开 TECAM_DATASET 再使能，并检查追剪参数是否设置合理

D．开启 M1，使能 TECAM 指令（若修改了 TECAM 里的数据，子程序的也需要修改，方便的做法是删去原子程序，重新创建一个子程序再进行调用）



若不使用掉电保持的 D 寄存器区域，需要创建一个 MOV 数据的子程序，因为 TECAM 写入的数据会在上电的时候清除掉，这个子程序的调用和 TECAM 同时或在 TECAM 之前都可以，不能在 TECAM 之后。

F．主轴使能，运行开始。

3.1.4 TECAM 运行参数说明

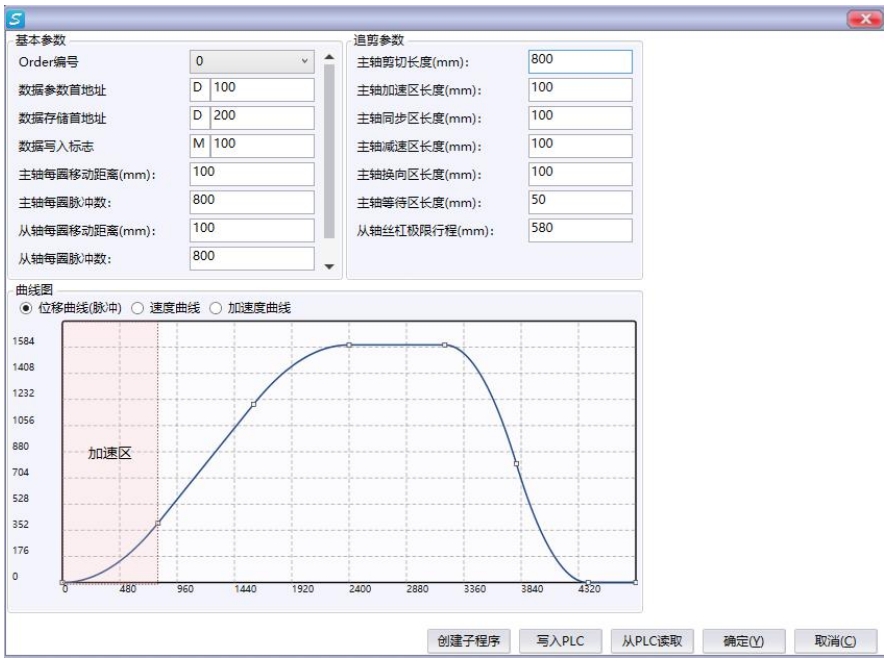
1）运行过程中需要调整相位

（运行参数首地址 D+102）这里是 D402，主轴相位调整：2000，单位为 pls，根据实际需要调整的相位填写

（状态标志位首地址 M+4）这里是 M204，相位调整使能标志位：ON 使能，可在线修改主轴相位，并且在下一个周期立即生效，每次调整需要手动触发，并且需要手动复位

2）**在线修改追剪参数**

A．直接在梯形图监视下或者屏幕端修改寄存器 D 的数值，编辑界面上修改后则需要写入 PLC

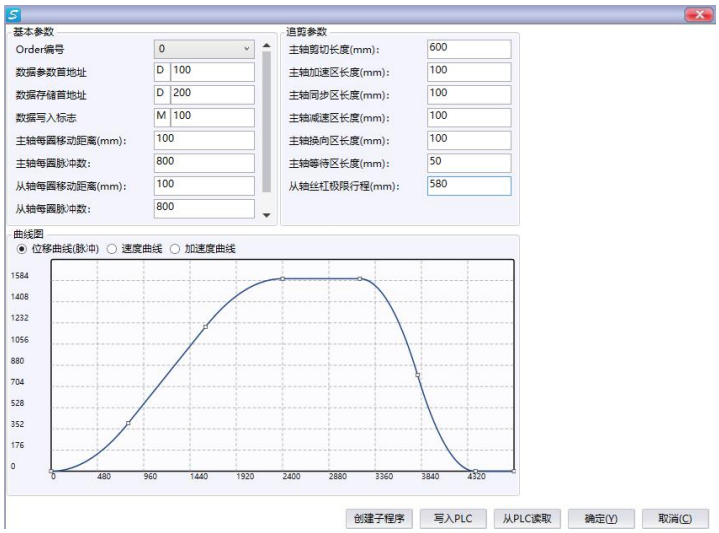


B．置位（数据写入标志首地址）M100，待 M101 变为 ON 后，则修改成功，下一周期就开始执行新的追剪参数

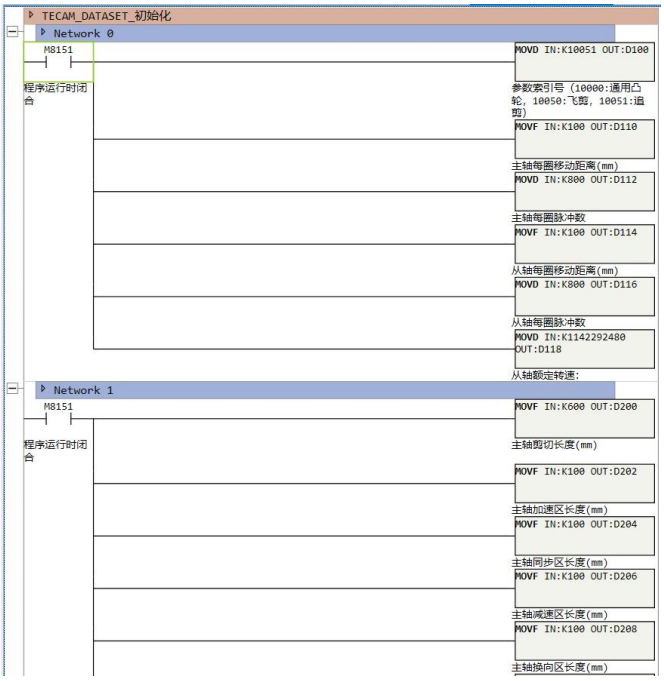
3.2 定标追剪范例

3.2.1 TECAM_DATASET 设置

Order 编号可选 0 或 1，表示第一个或第二个凸轮表，范例填 0，此后的 TECAM 可以调用凸轮表里的数据，
数据参数首地址：填写一个 D 寄存器，范例填 D100，该数据会占用掉从 D 到 D+39 共 40 个寄存器
数据存储首地址：填写一个未用过的 D 寄存器，范例填 D200，占用的寄存器数量详情翻上 1.2
数据写入标志：填写一个 M，范例填 M100，占用两个 M 寄存器
主轴每圈移动距离：100，忽略掉中间全部传动机构，实际主轴转一圈，机械走的距离
主轴每圈脉冲数：800，根据电机的细分或者伺服的电子齿轮比得出，最好设置成偶数或者 1000 的倍数
从轴每圈移动距离：100，忽略掉中间全部传动机构，实际从轴转一圈，机械走的距离
从轴每圈脉冲数：800，根据电机的细分或者伺服的电子齿轮比得出，最好设置成偶数或者 1000 的倍数
从轴额定转速：默认 0 就是按照脉冲最高 190kHz 计算；如果是追飞剪，也可以设置成 2000，单位默认 r/min
追剪参数可以参考范例，具体数值根据实际填写即可。

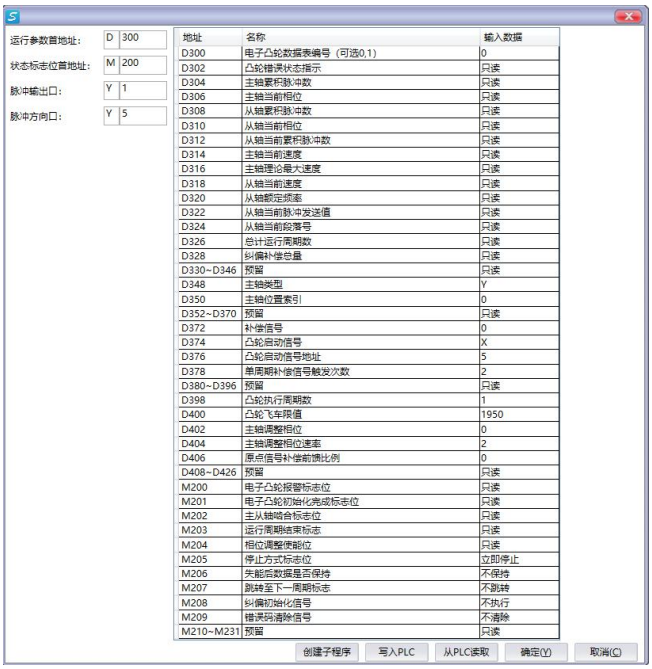


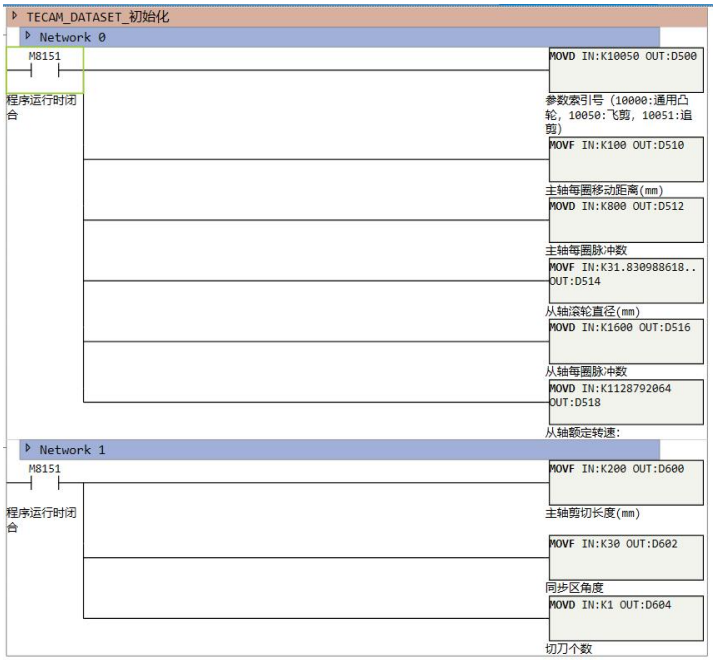
点击创建子程序



至此，TECAM_DATASET 凸轮表格就已经设置完成，后续该子程序需要在 main 中进行调用。

3.2.2 TECAM 指令设置界面





脉冲输出口：追剪跟随的从轴的脉冲输出口

脉冲方向口：追剪跟随的从轴的脉冲方向口

主轴类型：跟随的主轴脉冲输出口选 Y（立即刷新）

主轴位置索引：确定具体轴，若主轴类型选 Y，主轴位置索引填 0 代表 Y0，填 1 代表 Y1，以此类推

凸轮启动信号：定标追剪选 X（也可以选用 M，将 M 作为启动信号），只有初始化过程中生效，运行过程中修改信号无效。启动信号必须是上升沿信号，如果 M 作为启动信号，启动之后 M 要进行复位。

凸轮启动信号地址：5，代表 X5 的信号，在从轴追剪过程中，如果主轴再次触发到 X5 的信号，并不会影响到从轴的运行，也不会打乱从轴的运行状态。因为从轴只有执行完，才去检测主轴的凸轮启动信号，在运行过程中是不会检测凸轮启动信号的。

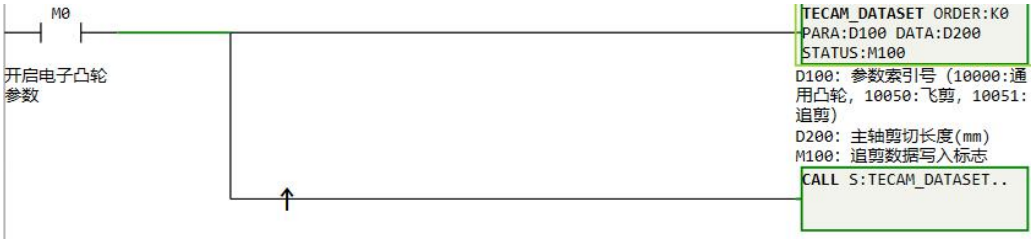
单周期补偿信号触发次数：一般是 2（设置补偿信号后才启用）

凸轮执行周期数：选 1。1 代表检测到凸轮启动信号后从轴执行 1 次追剪，2 代表检测到凸轮启动信号后从轴执

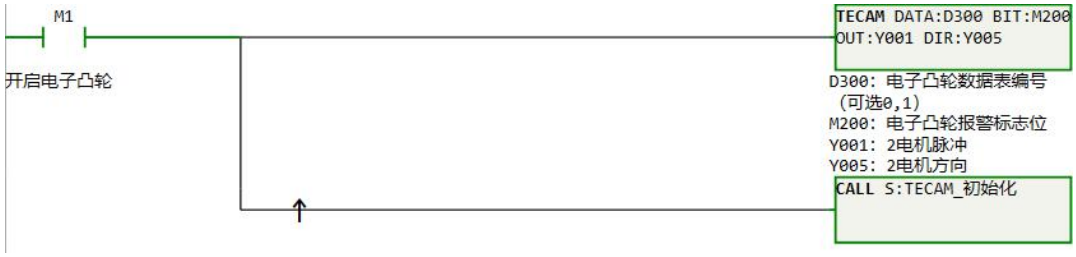
行 2 次追剪，以此类推。0 代表循环执行，一般用在定长中而不用在定标中。运行过程中修改立即生效。
凸轮飞车限值：设置 1950
主轴相位调整：0，不调整相位
主轴调整相位速率：2

3.2.3 TECAM_DATASET 和 TECAM 的使用流程

A．下载程序成功后，进入监视
B．开启 M0，使能 TECAM_DATASET（如果不想下载程序后还在上位点击写入 PLC，就直接创建一个子程序，并直接调用就可以了）



C．将数据写入标志位 M100（在 TECAM_DATASET 中）置为 ON，若标志位 M101（在 TECAM_DATASET 中）变为 ON 则写入成功，可以进行下一步；若 M101 仍是 OFF，则说明写入不成功，需要断开 TECAM_DATASET 再使能，并检查追剪参数是否设置合理
D．开启 M1，使能 TECAM 指令



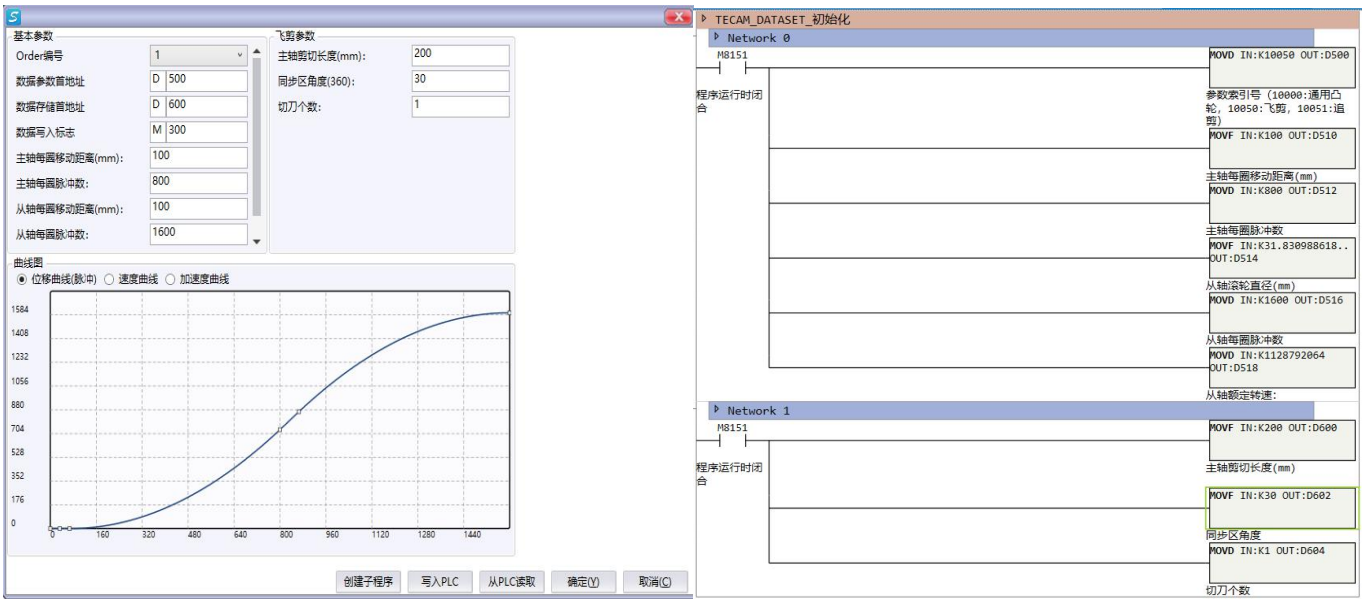
若不使用掉电保持的 D 寄存器区域，需要写一个 MOV 数据的子程序，因为 TECAM 写入的数据会在上电的时候清除掉，这个子程序的调用和 TECAM 同时或在 TECAM 之前都可以，不能在 TECAM 之后。
F．主轴使能，运行开始。

3.3 定长飞剪范例

3.3.1 TECAM_DATASET 设置

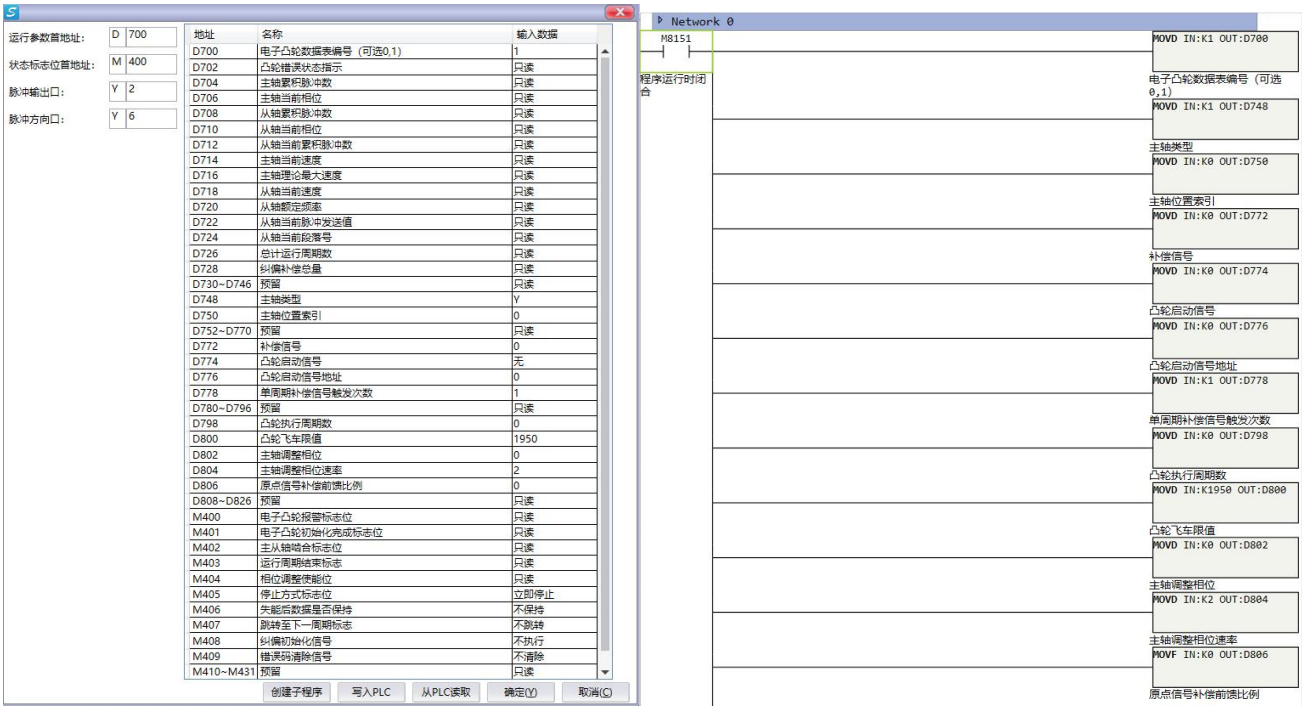
Order 编号可选 0 或 1，表示第一个或第二个凸轮表，范例填 1，此后的 TECAM 可以调用凸轮表里的数据，对应 TECAM 指令中的运行参数首地址需要写入 1，即 3.3.2 中的 D700 要选 1
数据参数首地址：填写一个 D 寄存器，范例填 D500，该数据会占用掉从 D 到 D+39 共 40 个寄存器
数据存储首地址：填写一个未用过的 D 寄存器，范例填 D600，占用的寄存器数量详情翻上 1.2
数据写入标志：填写一个 M，范例填 M300，占用两个 M 寄存器
主轴每圈移动距离：100，忽略掉中间全部传动机构，实际主轴转一圈，机械走的距离
主轴每圈脉冲数：800，根据电机的细分或者伺服的电子齿轮比得出，最好设置成偶数或者 1000 的倍数
从轴每圈移动距离：100，忽略掉中间全部传动机构，实际从轴转一圈，机械走的距离

从轴每圈脉冲数：1600，根据电机的细分或者伺服的电子齿轮比得出，最好设置成偶数或者 1000 的倍数
从轴额定转速：默认 0 就是按照脉冲最高 190kHz 计算；如果是追飞剪，也可以设置成 2000，单位默认 r/min
主轴剪切长度(float)：200，表示整个飞剪过程中，飞剪轮把主轴上的料剪切下来的长度
同步区角度(360°)：30，表示飞剪轮和主轴同速时走过的角度，其走过的长度是同步区角度/360*飞剪轮周长
切刀个数：1，如果填 2，原本的一个从轴周期被拆分成了两个周期。若原本一个从轴周期转一圈，拆分之后一个从轴周期转半圈。



创建子程序如右图，再在 main 程序中调用该子程序。

3.3.2 TECAM 指令设置界面



创建子程序如右图，再在 main 程序中调用该子程序。

脉冲输出口：飞剪跟随的从轴的脉冲输出口

脉冲方向口：飞剪跟随的从轴的脉冲方向口

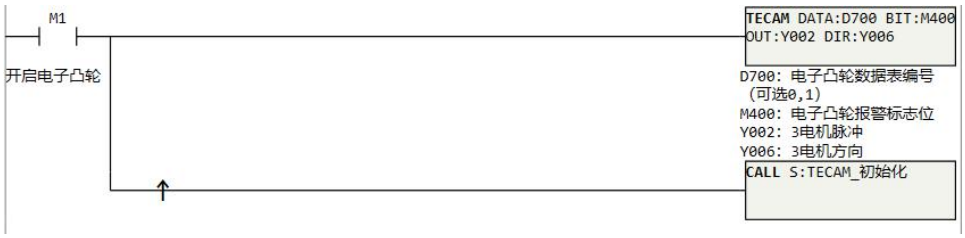
主轴类型：跟随的主轴脉冲输出口选 Y（立即刷新）
主轴位置索引：确定具体轴，若主轴类型选 Y，主轴位置索引填 0 代表 Y0，填 1 代表 Y1，以此类推
单周期补偿信号触发次数：飞剪一般是 1（设置补偿信号后才启用）
凸轮执行周期数：选 0。代表循环执行
凸轮飞车限值：设置 1950
主轴相位调整：0，不调整相位
主轴调整相位速率：2

3.3.3 TECAM_DATASET 和 TECAM 的使用流程

A．下载程序成功后，进入监视
B．开启 M0，使能 TECAM_DATASET（如果不想下载程序后还在上位点击写入 PLC，就直接创建一个子程序，并直接调用就可以了）



C．将数据写入标志位 M300（在 TECAM_DATASET 中）置为 ON，若标志位 M301（在 TECAM_DATASET 中）变为 ON 则写入成功，可以进行下一步；若 M301 仍是 OFF，则说明写入不成功，需要断开 TECAM_DATASET 再使能，并检查追剪参数是否设置合理
D．开启 M1，使能 TECAM 指令



若不使用掉电保持的 D 寄存器区域，需要写一个 MOV 数据的子程序，因为 TECAM 写入的数据会在上电的时候清除掉，这个子程序的调用和 TECAM 同时或在 TECAM 之前都可以，不能在 TECAM 之后。

F．主轴使能，运行开始。

3.4 定标飞剪范例

3.4.1 TECAM_DATASET 设置

Order 编号可选 0 或 1，表示第一个或第二个凸轮表，范例填 1，此后的 TECAM 可以调用凸轮表里的数据，对应 TECAM 指令中的运行参数首地址需要写入 1，即 3.3.2 中的 D700 要选 1

数据参数首地址：填写一个 D 寄存器，范例填 D500，该数据会占用掉从 D 到 D+39 共 40 个寄存器

数据存储首地址：填写一个未用过的 D 寄存器，范例填 D600，占用的寄存器数量详情翻上 1.2

数据写入标志：填写一个 M，范例填 M300，占用两个 M 寄存器

主轴每圈移动距离：100，忽略掉中间全部传动机构，实际主轴转一圈，机械走的距离

主轴每圈脉冲数：800，根据电机的细分或者伺服的电子齿轮比得出，最好设置成偶数或者 1000 的倍数

从轴每圈移动距离：100，忽略掉中间全部传动机构，实际从轴转一圈，机械走的距离

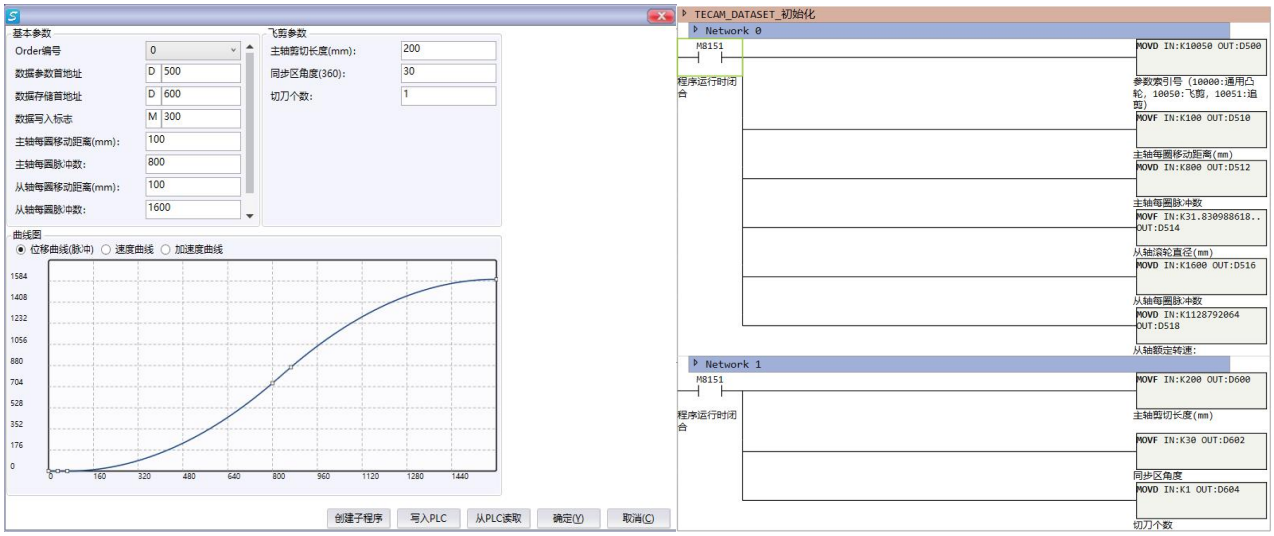
从轴每圈脉冲数：1600，根据电机的细分或者伺服的电子齿轮比得出，最好设置成偶数或者 1000 的倍数

从轴额定转速：默认 0 就是按照脉冲最高 190kHz 计算；如果是追飞剪，也可以设置成 2000，单位默认 r/min

主轴剪切长度(float)：200，表示整个飞剪过程中，飞剪轮把主轴上的料剪切下来的长度

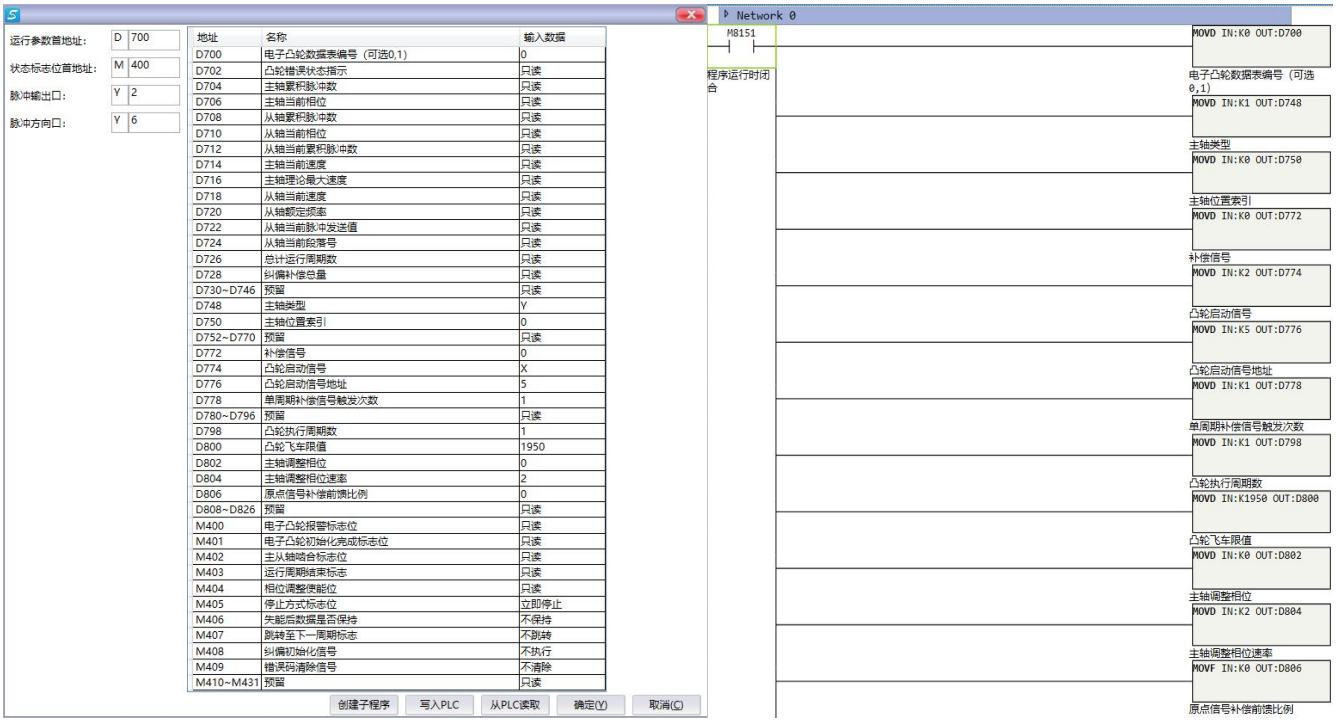
同步区角度(360°)：30，表示飞剪轮和主轴同速时走过的角度，其走过的长度是同步区角度/360*飞剪轮周长

切刀个数：1，如果填 2，原本的一个从轴周期被拆分成了两个周期。若原本一个从轴周期转一圈，拆分之后一个从轴周期转半圈。



创建子程序如右图，再在 main 程序中调用该子程序。

3.4.2 TECAM 指令设置界面



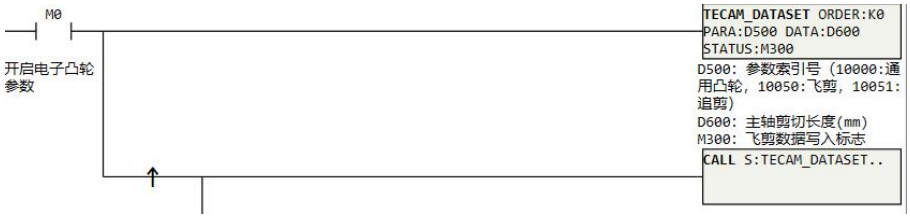
创建子程序如右图，再在 main 程序中调用该子程序。

脉冲输出口：追剪跟随的从轴的脉冲输出口
脉冲方向口：追剪跟随的从轴的脉冲方向口
主轴类型：跟随的主轴脉冲输出口选 Y（立即刷新）
主轴位置索引：确定具体轴，若主轴类型选 Y，主轴位置索引填 0 代表 Y0，填 1 代表 Y1，以此类推
凸轮启动信号：定标飞剪选 X（也可以选用 M，将 M 作为启动信号），只有初始化过程中生效，运行过程中修改信号无效。启动信号必须是上升沿信号，如果 M 作为启动信号，启动之后 M 要进行复位。
凸轮启动信号地址：5，代表 X5 的信号，在从轴飞剪过程中，如果主轴再次触发到 X5 的信号，并不会影响到从轴的运行，也不会打乱从轴的运行状态。因为从轴只有执行完，才去检测主轴的凸轮启动信号，在运行过程中是不会检测凸轮启动信号的。
单周期补偿信号触发次数：飞剪一般是 1（设置补偿信号后才启用）
凸轮执行周期数：选 1。1 代表检测到凸轮启动信号后从轴执行 1 次追剪，2 代表检测到凸轮启动信号后从轴执行 2 次追剪，以此类推。0 代表循环执行，一般用在定长中而不用在定标中。运行过程中修改立即生效。
凸轮飞车限值：设置 1950
主轴相位调整：0，不调整相位
主轴调整相位速率：2

3.4.3 TECAM_DATASET 和 TECAM 的使用流程

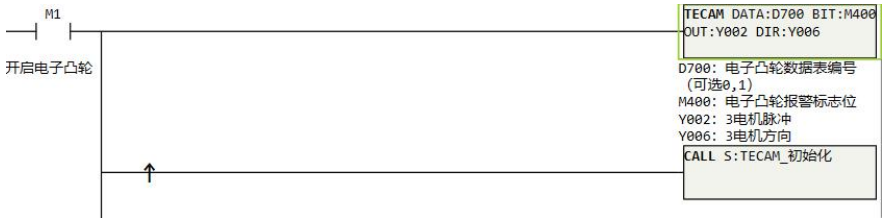
A．下载程序成功后，进入监视

B．开启 M0，使能 TECAM_DATASET（如果不想下载程序后还在上位点击写入 PLC，就直接创建一个子程序，并直接调用就可以了）



C．将数据写入标志位 M300（在 TECAM_DATASET 中）置为 ON，若标志位 M301（在 TECAM_DATASET 中）变为 ON 则写入成功，可以进行下一步；若 M301 仍是 OFF，则说明写入不成功，需要断开 TECAM_DATASET 再使能，并检查追剪参数是否设置合理

D．开启 M1，使能 TECAM 指令



若不使用掉电保持的 D 寄存器区域，需要写一个 MOV 数据的子程序，因为 TECAM 写入的数据会在上电的时候清除掉，这个子程序的调用和 TECAM 同时或在 TECAM 之前都可以，不能在 TECAM 之后。

F．主轴使能，运行开始。

3.5 短料剪切

前面得范例都是长料剪切，整个剪切的长度大于从轴的周长，从轴的运动就是先加速到同步区（最高速），之后减速，这样就实现了剪切长料。

当剪切长度小于从轴周长时，就需要短料切割，从轴的运动就是同步区为最低速，减速到同步区，过了同步区之后加速到调整区。即短料切割时，从轴必须在调整区内先加速，然后再减速到同步速度。

示例飞剪参数设置：

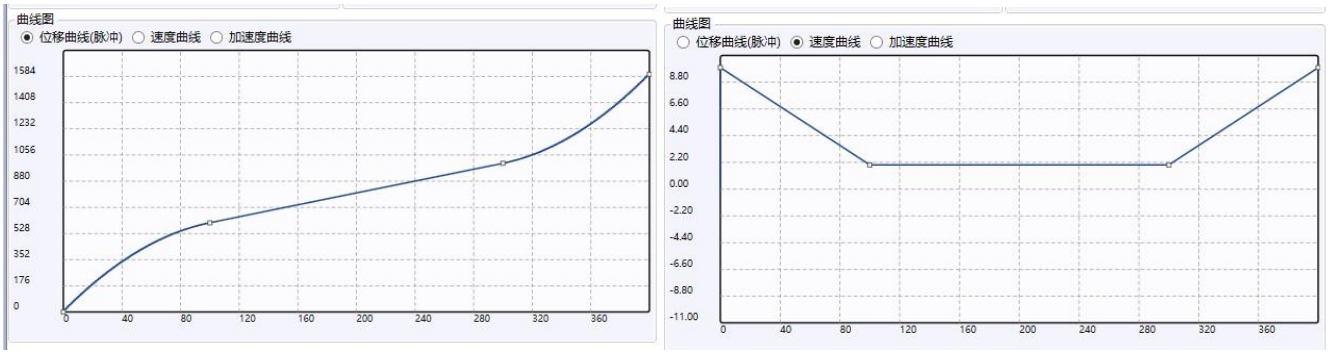
主轴剪切长度：50

同步区角度：90

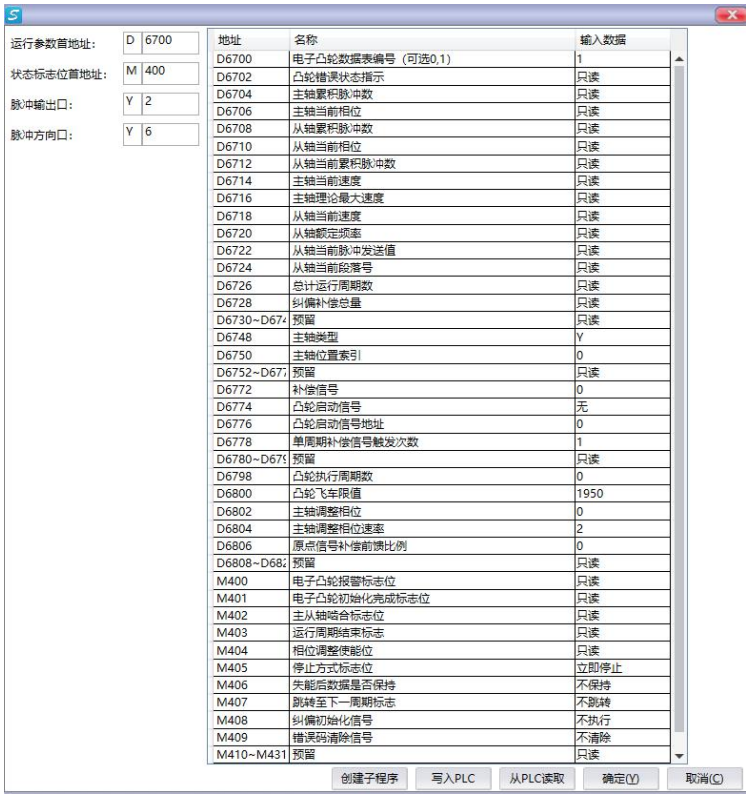
切刀个数：1



短料剪切的位移曲线和速度曲线如下，横坐标为从轴，纵坐标为主轴，速度曲线可明显看到从轴在到同步区段前减速，过了同步区后开始加速，同步区是速度最低的段落。



TECAM 指令的设置和定长飞剪一致。



具体的操作和启动步骤和其他追飞剪流程一致。

3.6 启用补偿信号

纠偏启用是因为要求从轴零点需要非常精确，但这同时也要求原点信号足够稳定，这样才能每次将从轴回零位置校准。

相关参数

运行参数首地址（D700）+偏移地址	描述	数据类型	寄存器类型	数据范围	说明
28	纠偏补偿总量	Int	D		只读 纠偏的脉冲数上限为周期脉冲数(即从轴一圈的脉冲数)

					的 10% 每圈刷新一次
72	补偿信号	Int	D	0,1,2	0:无 1：补偿信号 X6 2：补偿信号 X0 3: 补偿信号 X1 用于从轴的纠偏 启用后不可再用于其他外部中断功能
78	单周期纠偏信号触发次数	Int	D		主轴单周期会触发补偿信号的次数 飞剪一般是 1， 追减一般是 2
106	原点信号补偿前馈比例	float	F	>1	只有开启了纠偏信号才有用， 针对从轴的纠偏信号 建议设置的值大于 1
状态标志位首地址（M400）+偏移地址	描述				说明
8	纠偏初始化信号				0：不执行 1：执行 可以用于运行过程中清除纠偏信号触发异常

在定标飞剪的基础上设置纠偏信号参数

补偿信号：2，启用 X0 的信号输入，此时 X0 不能在用于其他的外部中断

单周期纠偏信号触发次数：1，从轴单周期内碰到从轴原点信号的次数，一般一个周期触发一次，用于矫正从轴回到原点的位置

原点信号补偿前馈比例：1.1，前馈比例设置需要大于 1。

按照正常步骤启用，监视纠偏的相关参数，如下图

D702 的数据类型是 DHEX，若 D702 显示 0x00010000 则说明纠偏信号触发异常，此时置位 M408 可以清掉该异常报错，但是引起这报错的原因是纠偏信号不稳或者从轴走一圈触发的纠偏信号次数大于设定的 1 次。

D728 表示每一圈纠偏的脉冲数，是每圈都会刷新一次，是个累积量，一般其纠偏补偿总量小于整个从轴走一圈脉冲数的十分之一。

该纠偏的信号是为了满足对从轴回原点的要求，保证每次从轴都能回到起始位置，满足对于从轴精度要求高的场合。



3.7 通用凸轮（关键点）

3.7.1 TECAM_DATASET 设置

自定义凸轮需要设定各个关键点参数，每个关键点参数由主轴坐标，从轴坐标，斜率，线型和解析度（解析度之和不超过 2048）五个参数组成。通过关键点坐标可以确定关键点的位置，通过斜率和线型可以确定曲线的形状，从而形成用户自定义的凸轮曲线。

Order 编号选 0，表示第一个凸轮表

数据参数首地址：D100，该数据会占用掉从 D 到 D+39 共 40 个寄存器，D100 的值必须是在 10000-10049 之间
D6102 是在 0-50 之间

数据存储首地址：D200，占用的寄存器数量详情翻上 1.2

数据写入标志：M100，占用两个 M 寄存器

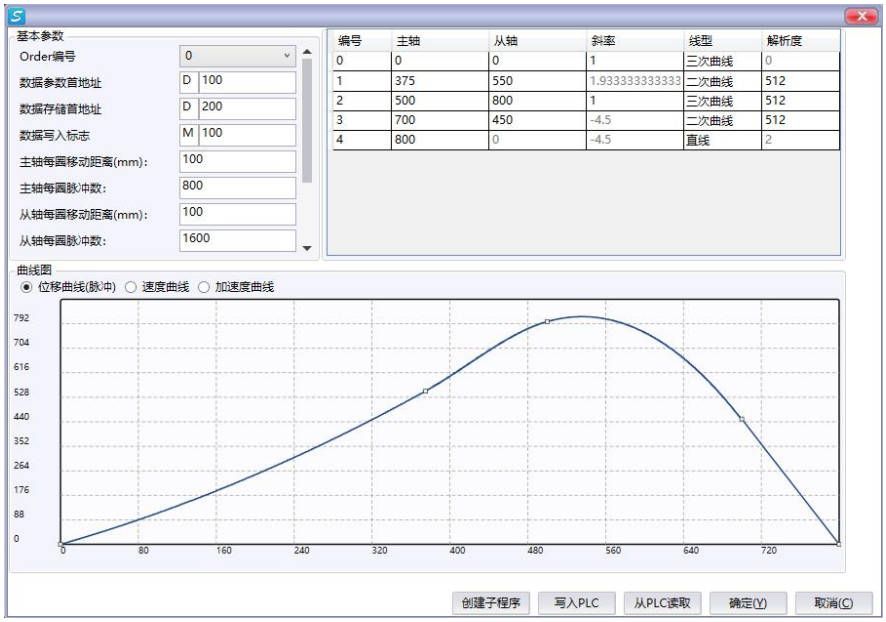
主轴每圈移动距离：100，忽略掉中间全部传动机构，实际主轴转一圈，机械走的距离

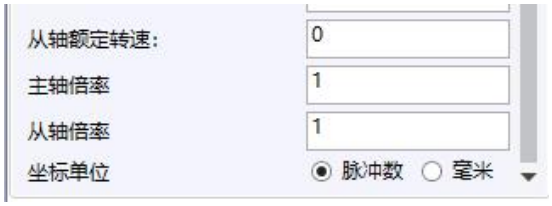
主轴每圈脉冲数：800，根据电机的细分或者伺服的电子齿轮比得出，最好设置成偶数或者 1000 的倍数

从轴每圈移动距离：100，忽略掉中间全部传动机构，实际从轴转一圈，机械走的距离

从轴每圈脉冲数：1600，根据电机的细分或者伺服的电子齿轮比得出，最好设置成偶数或者 1000 的倍数

从轴额定转速：默认 0 就是按照脉冲最高 190kHz 计算





运行过程说明：

第一段：当主轴从 0 运行到 375 时，从轴从 0 运行到 550，本段为二次曲线，终点斜率为 1.93；

第二段：当主轴从 375 运行到 500 时，从轴从 550 运行到 800，本段曲线为三次曲线，斜率为 1；

第三段：当主轴从 500 运行到 700 时，从轴从 800 运行到 450，本段曲线为二次曲线，斜率为-4.5；

第四段：当主轴从 700 运行到 800 时，从轴从 450 运行到 0，本段曲线为直线，斜率为-4.5；

当主轴运行越过第四段终点时，从轴以终点位置 0 为基准继续按照凸轮曲线开始运行，以此循环。

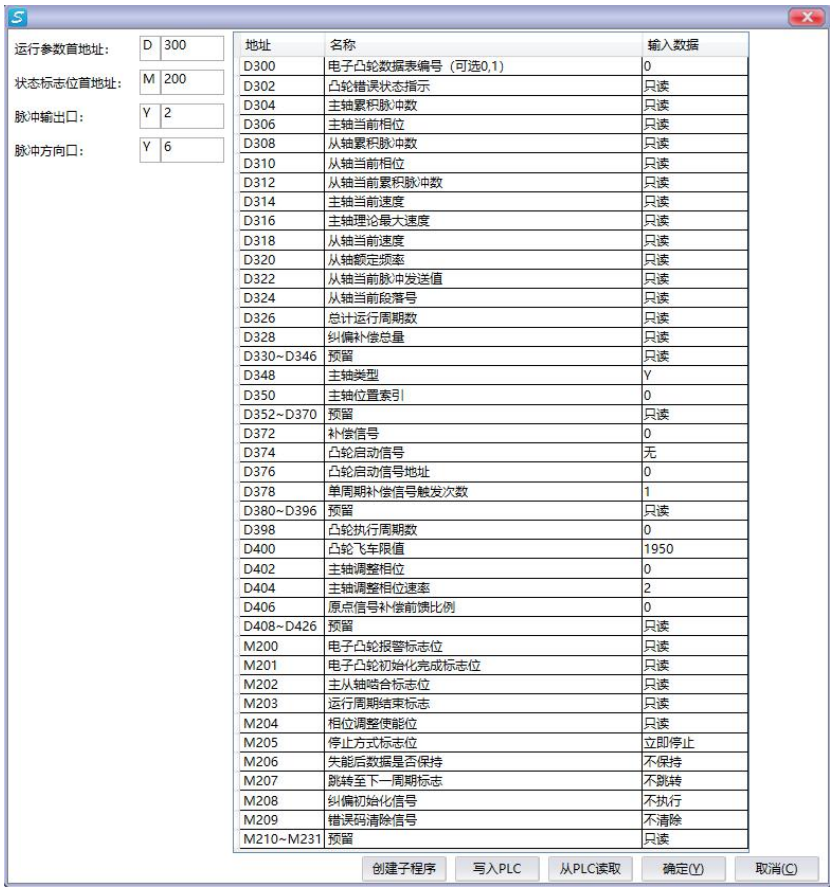
关于终点的说明：

如果曲线的终点回到起始点，下一次的循环就不会出现偏差；若是终点位置不在起始点，那么以后的每一次循环都会有一个位置偏差（终点和起始点之差），且该偏差会随着循环的递增而累加。

3.7.2 TECAM 设置

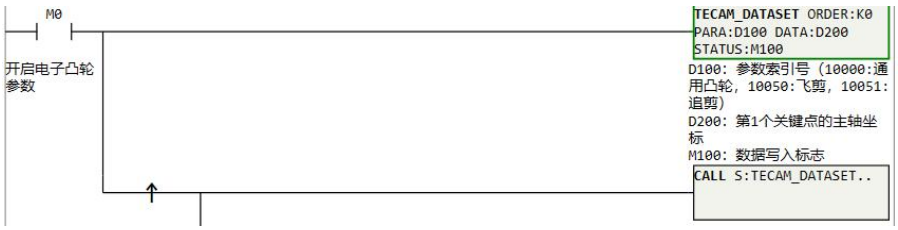
运行参数首地址：D300，占用 D300-D426 共 128 个寄存器

状态标志位首地址：M200，占用 M200-M231 共 32 个寄存器

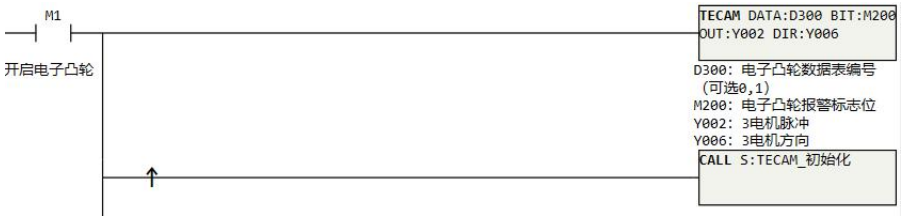


3.7.3 TECAM_DATASET 和 TECAM 的使用流程

A．下载程序成功后，进入监视
B．开启 M0，使能 TECAM_DATASET（如果不想下载程序后还在上位点击写入 PLC，就直接创建一个子程序，并直接调用就可以了）



C．在数据储存地址中写入关键点数据，将数据写入标志位 M100（在 TECAM_DATASET 中）置为 ON，若标志位 M101（在 TECAM_DATASET 中）变为 ON 则写入成功，可以进行下一步；若 M101 仍是 OFF，则说明写入不成功，需要断开 TECAM_DATASET 再使能，并检查追剪参数是否设置合理
D．开启 M1，使能 TECAM 指令

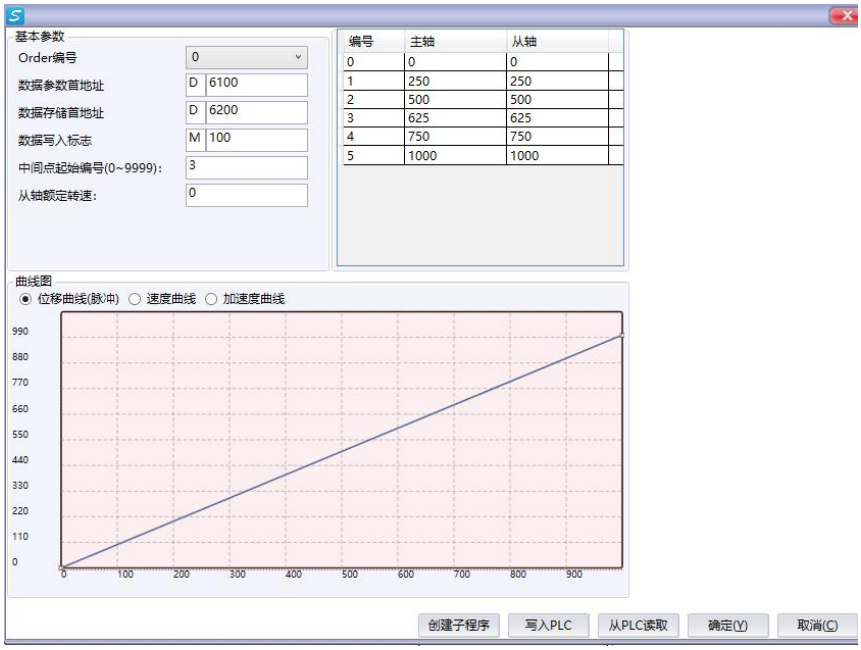


若不使用掉电保持的 D 寄存器区域，需要写一个 MOV 数据的子程序，因为 TECAM 写入的数据会在上电的时候清除掉，这个子程序的调用和 TECAM 同时或在 TECAM 之前都可以，不能在 TECAM 之后。
F．主轴使能，运行开始。

3.8 通用凸轮（插入点）

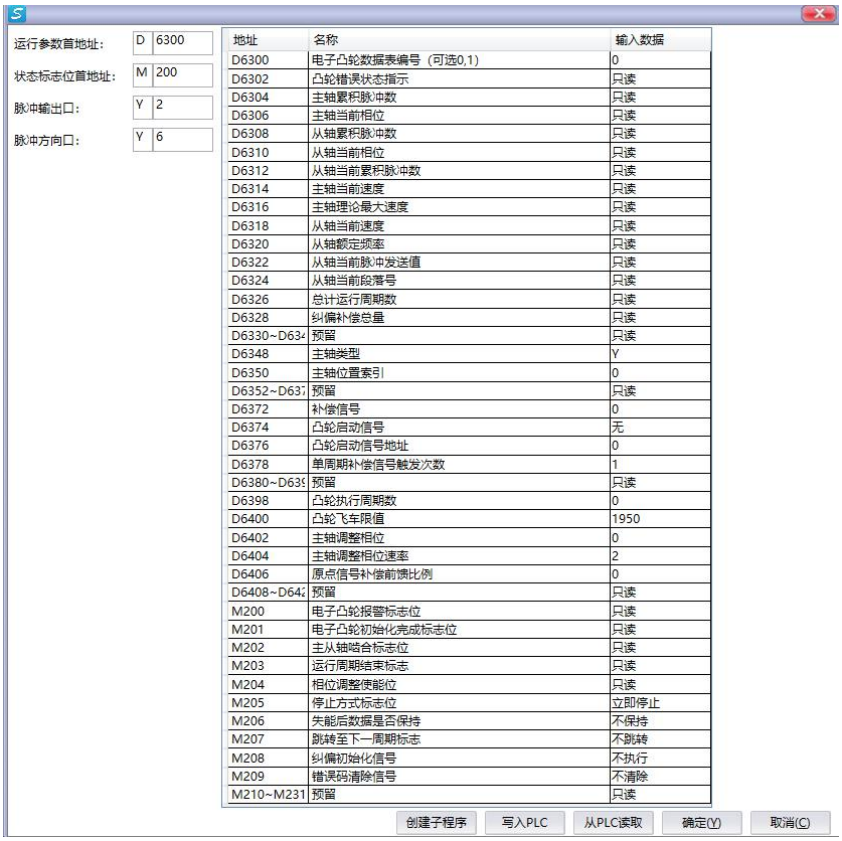
3.8.1 TECAM_DATASET 设置

插入点模式下只需要设置主轴和从轴的脉冲数，不需要主轴和从轴的每转脉冲数和每圈距离，只依据主轴脉冲数对应的从轴脉冲数计算出运动曲线。
Order 编号选 0，表示第一个凸轮表
数据参数首地址：D6100，D6100 的值必须是在 0-9999 之间
D6102 是在 0-2048 之间，数据深度最大是 2048
数据存储首地址：D6200，占用的寄存器数量详情翻上 1.2，一个主轴或从轴坐标占一个双字寄存器
数据写入标志：M100，占用两个 M 寄存器
中间点起始编号：即 D6100 的值，初次写入时为 0，后续修改时为修改部分内容的起始编号



TECAM 指令的设置

定长和定标可具体选择，这里设置的参数是定长的参数



该模式下的启动和关键点模式的一致，也需要手动写入储存区数据；当点数超过 D 寄存器上限时，可修改起始编号和深度，分多次写入。

四种模式都支持凸轮表的动态修改，修改 TECAM_DATASET 对应寄存器区域后，给上升沿到刷新标志位触发，计算完成后下周期改动生效；关键点和中间点模式还支持局部数据修改，指定索引和深度后会修改对应位置和长度的点。

4.16 高速计数指令

内容如下：

1、 高速脉冲计数 HCNT

2、 凸轮 CAM

3、 增强高速脉冲计数 EHCNT

4、 高速计数目标值中断事件 HSCS

5、 闭环控制系统 PID

6、 增强闭环控制系统 EPID

4.16.1 HCNT

1. 指令说明

	高速计数器地址编号	高速计数器输入端接线							
		X000	X001	X002	X003	X004	X005	X006	X007
无启动/复位的单相计数器	C235	U							
	C236		U						
	C237	保留							
	C238	保留							
	C239	保留							
	C240	保留							
指定启动/复位的单相计数器	C241	U	R						
	C242	保留							
	C243	保留							
	C244	U	R					S	
	C245	保留							
单相双计数器	C246	U	D						
	C247	U	D	R					
	C248	保留							
	C249	U	D	R				S	
	C250	保留							
A/B 相计数器	C251	A	B						
	C252	A	B	R					
	C253	保留							
	C254	A	B	R				S	
	C255	保留							

对于型号 FGRs-C8X8T-4T：

	高速计数器地址编号	高速计数器输入端接线							
		X000	X001	X002	X003	X004	X005	X006	X007
单相计数器	C235	U							
	C236		U						
	C237	保留		U					
	C238	保留			U				
	C239	保留							
	C240	保留							
	C241								
	C242	保留							
	C243	保留							
	C244								
	C245	保留							
单相双计数器	C246	U	D						
	C247								
	C248	保留		U	D				
	C249								
	C250	保留							
A/B 相计数器	C251	A	B						
	C252			A	B				
	C253	保留							
	C254								
	C255	保留							

如上表所示，高速计数器编号从 C235 到 C255，脉冲输入口从 X000 到 X0007；其中，U 表示向上计数，D 表示向下计数，R 表示复位清零计数器（当 R 对应的 X 脚，出现上升沿时，计数器会被清零），S 表示启动计数器（带有 S 端的计数器，需要在 S 对应的 X 脚，给予上升沿，才能启动相应的计数器），AB 表示 AB 相。

当某一输入口（例如：X000）被某一高速计数器（例如：C235）使用，那么梯形图中不可以出现使用该输入口的其他高速计数器（例如：C241，C244，C246 等），否则高速计数器将不能正常工作。

计数范围：-2,147,483,648～+2,147,483,647；当计数值超出此范围时，会产生上溢或下溢现象。

注意：

1. 当使用高速计数输入或中断输入时，建议相应输入端口的线缆采用双绞屏蔽线，并将屏蔽层接地(同端子连接或连接信号地)，以提高抗干扰性；

2. 当使用了该类计数器后，这些端子不能再同时作为其他计数器的输入用，也不能作为普通输入方式使用；

3. 晶体管的 OFF 时间具有在轻负载时会延长的特性。所以请在需要响应性时设计成让负载在较轻时加负载电阻，增加负载电流；

4. A/B 相计数的倍频设置要在指令初始化之前才有效。

2. 指令的有效操作数

输入输出	类型	操作数	描述

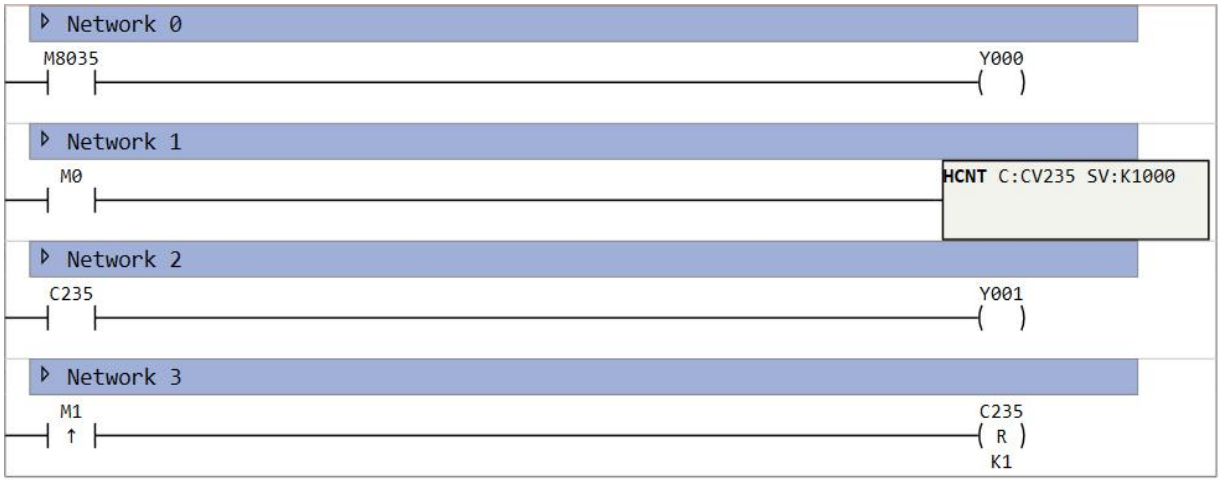
EN	BOOL		**启用输入端接通时，指令执行；否则，不执行**
C	整数/双整数	**CV**	**选定的计数寄存器**
SV	整数/双整数	**K/H/D**	**预设计数值**

3. 实例 1.无启动/复位的单相计数器

指令表：

```
Network 000
    LD      M8035
    OUT     Y000
Network 001
    LD      M0
    HCNT    CV235   K1000          //当 M0 为‘1’，M8035 为‘0’时，CV235 对 X000 的输入脉冲进行正向计数，当计数达到 1000 个时，将置位 C235
Network 002
    LD      C235
    OUT     Y001
Network 003
    LDP     M1
    RST     C235    K1             //当 M1 上升沿来临时，清零 C235 和 CV235
```

梯形图：



4. 实例

指定启动/复位的单相计数器

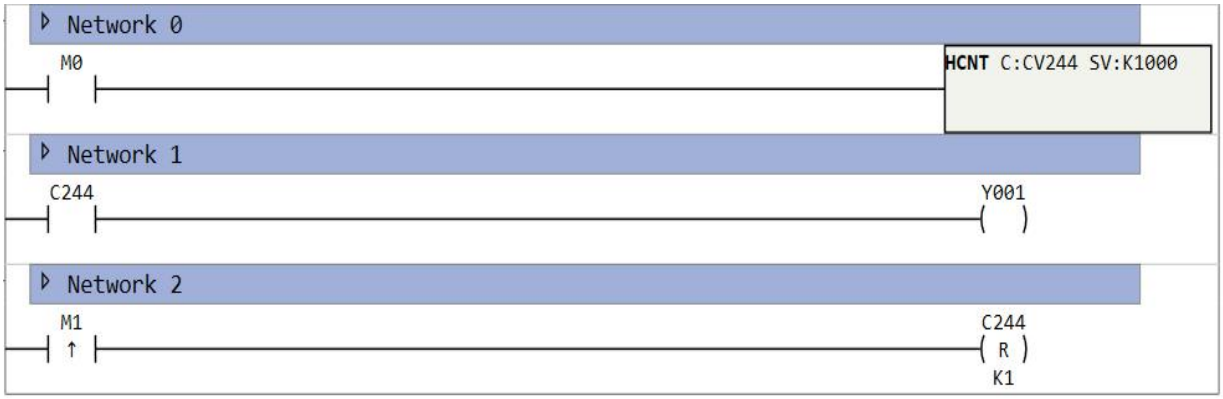
指令表：

```
Network 000
    LD      M0
    HCNT    CV244   K1000          //当 M0 为‘1’时，当 X006 上升沿来临时启动 CV244，之后 CV244 对 X000 的输入脉冲进行正向计数（M8044 为 0 时），当计数达到 1000 个时，将置位 C244
                                   //若 M0 变为‘0’后，再变为‘1’时，需用 X006 的上升沿再次启动计数器
```

```
Network 001
    LD      C244
    OUT     Y001

Network 002
    LDP     M1
    RST     C244    K1          //当 M1 上升沿来临时，清零 C244 和 CV244，也可以通过给 X001 上升沿，来完成此功能
```

梯形图：



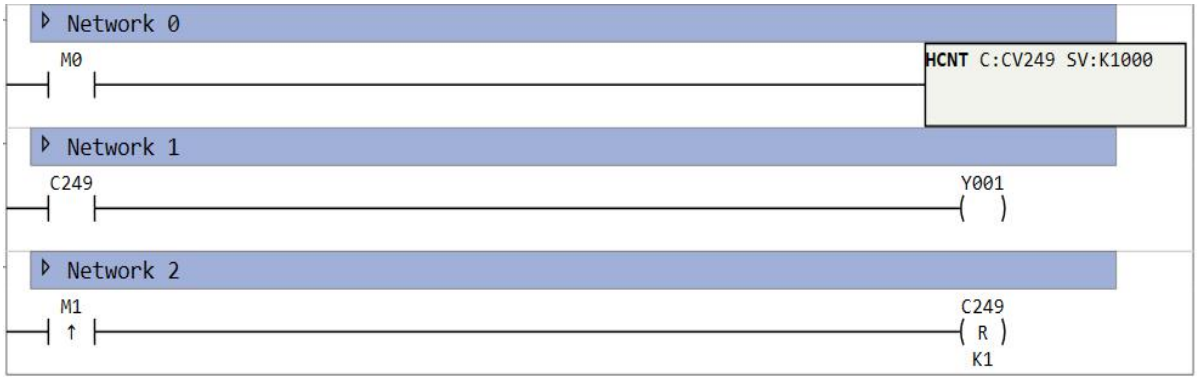
5. 实例

单相双计数器

指令表：

```
Network 000
    LD      M0
    HCNT    CV249   K1000       //当 M0 为‘1’时，当 X006 上升沿来临时启动 CV249，之后 CV249 对 X000 的输入脉冲进行正向计数，对 X001 的输入脉冲进行负向计数，当计数达到 1000 个时，将置位 C249
                                //若 M0 变为‘0’后，再变为‘1’时，需用 X006 的上升沿再次启动计数器
Network 001
    LD      C249
    OUT     Y001
Network 002
    LDP     M1
    RST     C249    K1          //当 M1 上升沿来临时，清零 C249 和 CV249，也可以通过给 X002 上升沿，来完成此功能
```

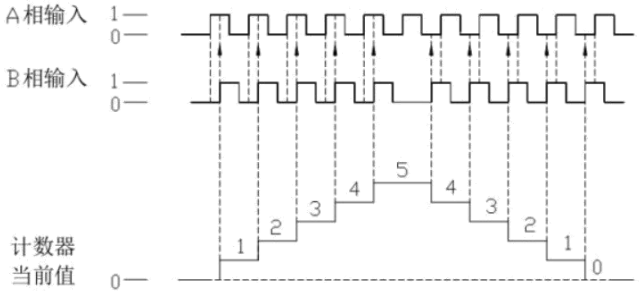
梯形图：



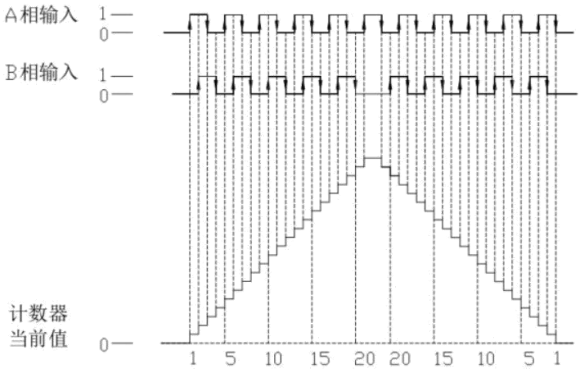
6. AB 相计数器

AB 相计数器分为两种计数模式，通过 M8051~M8055 来控制 AB 相计数器 C251~C255 倍频模式，当相应的 M 的为‘0’时，为一倍频模式。为‘1’时，为四倍频模式，倍频的设置要在计数器初始化之前，如下图：

一倍频模式：



四倍频模式：

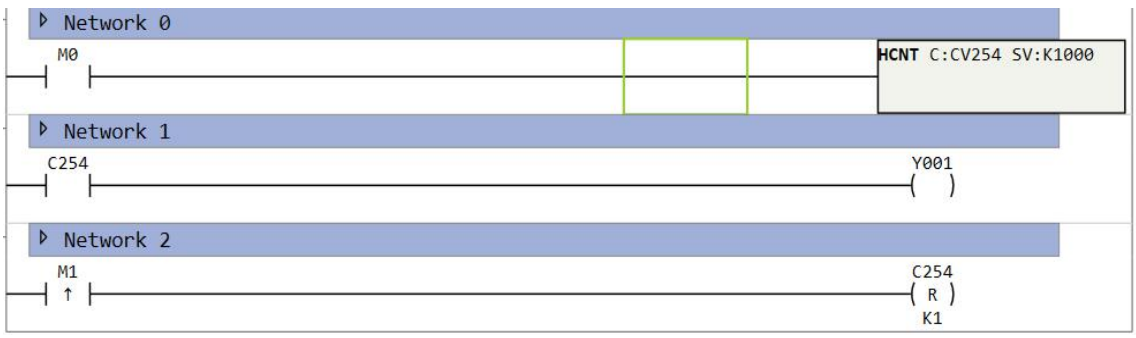


指令表：

```
Network 000
    LD      M0
    HCNT    CV254   K1000       //当 M0 为‘1’时，当 X006 上升沿来临时启动 CV254，之后 CV254 对输入 X000 和 X001 的 AB 相脉冲进行计数，当计数达到 1000 个时，将置位 C254
                                //若 M0 变为‘0’后，再变为‘1’时，需用 X006 的上升沿再次启动计数器
Network 001
    LD      C254
    OUT     Y001

Network 002
    LDP     M1
    RST     C254    K1          //当 M1 上升沿来临时，清零 C254 和 CV254，也可以通过给 X002 上升沿，来完成此功能
```

梯形图：



4.16.2 CAM

凸轮开关

指令说明：

1） 仅 FGm 系列 PLC 具有该功能；

2） 本指令必须与增强高速计数指令 EHCNT 联合使用，单独使用，高速计数器值不会发生变化。

使用说明：

1） 举例如下图，首先 CAM 必须联合 EHCNT 使用；

2） 设置 EHCNT 指令，可选择任一高速计数器任一计数模式，但其触发方式必须选择‘凸轮触发’，且 **EHCNT 的使能必须比 CAM 先导通**；

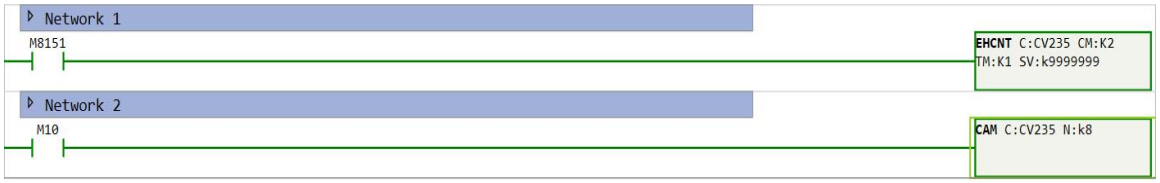
3） CAM 指令中的计数器选择必须与 EHCNT 中设置相同，此例中同为 CV235；

4） 最大目标值根据用户需求设置，CV235 值会在 0 到最大目标值之间循环变化，当 CV235 达到最大目标值时，会自动清零并继续计数到最大目标值；

5） 圈数存放的 D 寄存器中存放的是高速计数器值正向越过最大目标值的次数，用于计算计数器的总计数值，本例中圈数存放在 D100（D101）中；

6） CAM 参数设置窗口中间空白处鼠标右键即可添加中间目标值，以及操作地址和操作方式。中间目标值需设置在 0 和最大目标值之间，可等于 0 和最大目标值，设置超过最大目标值的中间目标值将做为最大目标值处理。操作地址可设置为 Y 输出或 M 寄存器，不同的 Y、M 可设置相同的中间目标值，中间目标值无需按大小顺序排列，其中操作方式置位或复位根据用户需求设置。

7）



CAM 梯形图实例



EHCNT 指令设置窗口



CAM 指令设置窗口

8） 上述均使用‘不将数据映射到 D 寄存器’模式，中间目标值操作地址和方式无法在线修改，当需要在线更改目标值、操作地址方式时，可选‘仅 D 控制’模式并选择控制的映射地址，选择该模式时，设置窗口中设置的中间目标值无效，仅通过 D 寄存器中的参数来进行相应的操作，其中数据在 D 寄存器中的存放格式如下图。其中，中间目标值双字正整数，操作类型写 0 表示操作 M 寄存器、写 1 操作 Y 输出口，例如操作类型写入 1，操作数偏移写入 10，则表示操作 Y10。置位复位操作分别写入 1 或 0 表示。注意，最终结束一定要结束标志，在最后一个操作的末尾加上一个双字-1 表示结尾，无结束标志 PLC 会报错。



CAM‘仅 D 控制’数据存放规则

注意：

1. CAM 指令需要与 EHCNT 配合才能使用，且 EHCNT 需设置为凸轮触发；

2. EHCNT 的使能导通必须在 CAM 之前；

3. 最大目标值和中间目标值不能设置为负数，且在计数过程中负向经过中间目标值不会触发相应动作，即凸轮是单向触发的；

4.　当对 Y 进行操作时，操作是立即执行的，上例中当 CV235 正向经过 1000 时，Y34 被立即置位，该置位动作不受扫描周期影响；

5.　中间目标值设置最大个数不要操作 200 个；

6.　当使用仅 D 寄存器控制时，必须以 32 位的-1 作为结束标志。

4.16.3 **EHCNT**

1.指令说明

高速计数接线端	**X0**	**X1**	**X2**	**X3**	**X4**	**X5**	**X6**	**X7**	**X10**	**X11**	**X12**	**X13**
单路计数	**C235**	**C236**	**C237**	**C238**	**C239**	**C240**	**C241**	**C242**	**C243**	**C244**	**C245**	**C246**
双路计数	**C235**		**C237**		**C239**		**C241**		**C243**		**C245**	

1）仅多脉冲型 PLC(FGm_64MT_A, FGm_32MT_A)支持该功能。

2）如上表所示，高速计数器编号从 C235 到 C246，一共 12 路高速输入，当为单相计数时，有 12 个高速计数器，为双相计数时，共有 6 个高速计数器。

3）脉冲输入口从 X000 到 X0013，最大脉冲输入频率 1MHz，计数范围是-2,147,483,648～+2,147,483,647；当计数值超出此范围时，会产生上溢

或下溢现象；

4）各计数器的计数模式分别有以下几种：

编号	计数模式	单路/双路
K0	单相加计数	单路
K1	A/B 相单倍频计数	双路
K2	A/B 相四倍频计数	双路
K3	单相双计数	双路
K4	脉冲方向	双路

单相加计数模式：可选任一一路计数器；

A/B 相计数：默认低位端口为 A 相、高位端口为 B 相，如 A/B 相 C235，X0 接 A 相，X1 接 B 相；

单相双计数：默认低位端口输入脉冲计数为加计数，高位端口输入脉冲计数为减计数；如单相双计数 C235，X0 输入一个脉冲 CV235 加 1，X1 输入一个脉冲 CV235 减 1；

脉冲方向：默认低位端口为脉冲输入口，高位端口为方向输入口，方向输入口 OFF 时正向计数，ON 时反向

计数；如脉冲方向计数 C235，X0 接脉冲输入，X1 接方向输入；

5）启动复位触发方式，高速计数的触发方式有三种，分别为自触发、外部端口触发和凸轮触发。

编号	触发方式
K0	自触发
K1	凸轮触发
K2	外部触发组 1
K3	外部触发组 2

①自触发是当计数器使能条件导通时即开始计数，达到目标值时置位计数标志位，并通过 RST 指令清空计数器值。

②外部触发方式是通过外部输入信号启动、复位计数器。外部触发组号及端口如下。首先计数器使能必须先导通，再启动端口输入上升沿，启动计数器，复位端口输入上升沿时，复位计数器。当不同的计数器选择相同的外部触发组时，当有触发时会对已经使能的计数器起作用。当端口已用于高速计数外部触发时，则不能再用于其他用途。

组号	启动端口	复位端口
1	X25	X26
2	X27	X32

③凸轮触发是计数器的触发与否决定于对应计数器的凸轮指令(CAM)的使能，首先计数器使能必须先导通，如果对应凸轮指令的使能导通，则计数器开始计数，并根据凸轮指令的设置复位计数器值。

6）M8050~M8055 分别为 C235~C246 的方向信号，为 OFF 时为正方向，为 ON 时为负方向。

2. 注意事项

- 当使用高速计数输入或中断输入时，建议相应输入端口的线缆采用双绞屏蔽线，并将屏蔽层接地(同端子连接或连接信号地)，以提高抗干扰性；
- 当使用了该类计数器后，这些端子不能再同时作为其他计数器的输入用，也不能作为普通输入方式使用；
- 晶体管的 OFF 时间具有在轻负载时会延长的特性。所以请在需要响应性时设计成让负载在较轻时加负载电阻，增加负载电流；

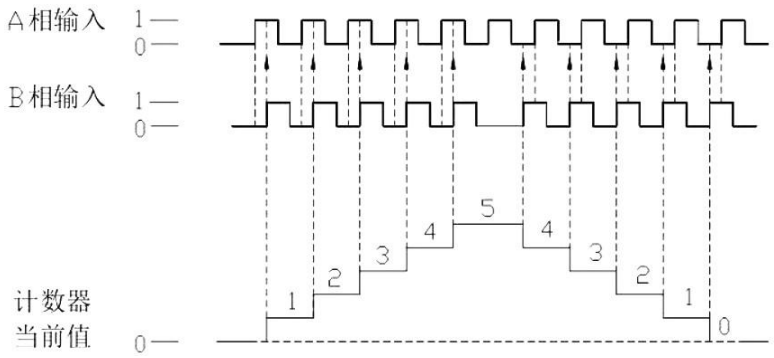
- 达到目标值后置位标志位为实时响应，不受扫描周期影响。

3. 指令的有效操作数

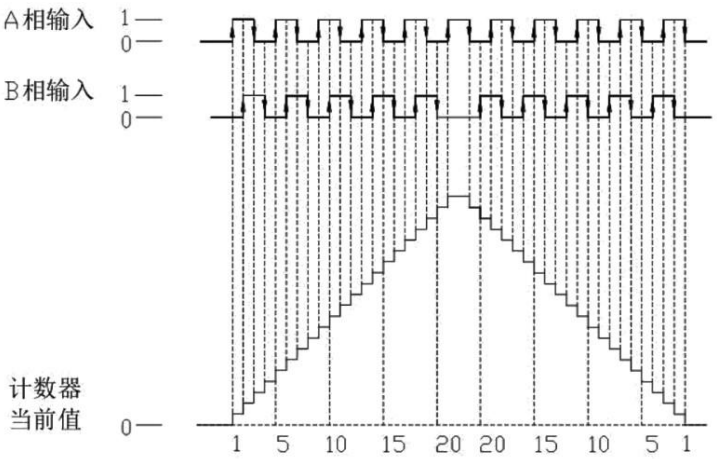
输入输出	类型	操作数	描述
EN	BOOL		**启用输入端接通时，指令执行；否则，不执行**
高速计数器	CV	**CV**	**选择高速计数器通道**
计数模式	整数	**K**	**选择高速计数器的计数模式**
启动复位触发方式	BOOL	**K**	**选择高速计数器启动复位触发方式**
目标值	BOOL	**K/D**	**高速计数器标志位置位的目标值**

4. 实例

AB 相计数器（一倍频模式）：



AB 相计数器（四倍频模式）：



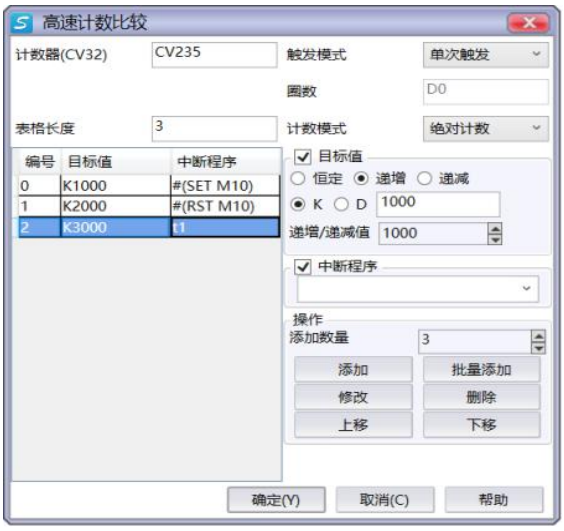
4.16.4 HSCS

1. 指令说明

- 目前该指令支持除了 FGm_64MT_A 以外的所有机型
- HSCS 指令与单相高速计数器（如 CV235、CV236 等）或者 AB 相高速计数器（CV251）联动使用，必须先接通对应的 HCNT 高速计数再接通 HSCS 指令才有效
- 失能该指令后不会触发中断程序。如果高速计数值没有越过下一个中断目标值，重新接通后可以继续触发。如果越过了下一个中断目标值，则后续中断操作均不能触发（AB 相反向时经过下一个目标值，依然可以依次触发后续所有中断）
- 利用 RST 指令复位对应高速计数器的标志位如 C235，C251 时，清空高速计数器计数值的同时会初始化目标值中断功能，从第一段开始重新触发预设的目标值中断事件
- 中断事件均在中断中即时实现，与扫描周期无关

2. 参数设置

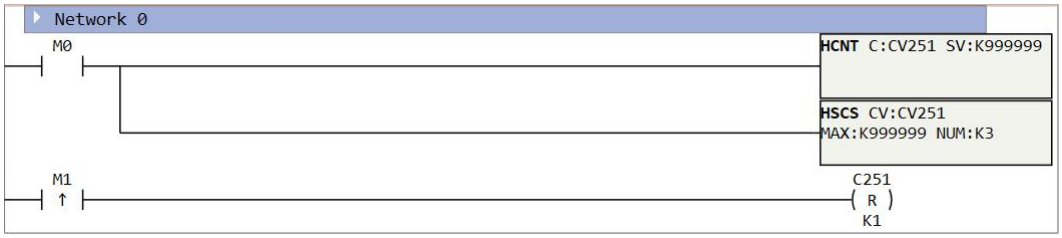
参数设置界面如下：



- 计数器（CV32）：输入高速计数器的编号，单相高速计数器 CV235-CV238，AB 相高速计数器 CV251-CV252。（目前不支持双相计数器）
- 触发模式与圈数：触发模式中分为循环触发和单次触发两种，循环触发模式下可以指定显示循环圈数的双字 D 寄存器。

✧ 循环触发：当计数值到达最后一个目标值并且触发对应的中断程序后，当前计数器的值会清零后重新开始计数，同时圈数会加 1，此时从第一段开始按照预设目标值触发中断事件。

✧ 单次触发：当计数值到达最后一个目标值并且触发对应的中断程序后，当前计数器的值会继续计数，如果不复位计数器将不会再次触发中断



Tips:

- 中断触发都是按照预设好的中断目标值的顺序来依次执行的，在触发下一段目标值对应的中断事件前都不会触发其余的目标值中断

- 计数模式：计数模式分为绝对模式和相对模式

✧ 绝对模式：目标值代表实际计数到该目标值便触发中断。如参数设置界面所示，当高速计数器 CV235 分别到达 1000,2000,3000 时进入触发对应中断事件。

✧ 相对模式：目标值代表计数到达上一段目标值加上本段增加的计数值处时触发中断事件，如参数设置界面所示，当高速计数器 CV235 分别到达 1000,1000+2000=3000,3000+3000=6000 时触发相应的中断事件。

- 目标值：目标值可以是 K（双字常数），D（双字寄存器）两种类型，在参数设置界面可以通过设定递增量批量增加目标值，在操作中可以单条、批量增加或修改目标值，还

可以上下移某条目标值，但需要注意目标值之间需要满足一定条件才能生效。

Tips:

- 绝对模式中，单相高速计数器的目标值必须大于上一段的目标值，AB 相高速计数器的目标值不等于上一段的目标值。
- 相对模式中，单相高速计数器的目标值必须大于 0，AB 相高速计数器的目标值不等于 0。
- 无论是哪种模式，单相高速计数器目标值不能设置负值，AB 相可以设置负值。
- 第一个目标值不能为 0。
- 如果本段目标值不满足以上条件，则不会触发本段目标值对应的中断事件，后续的事件仍然按照顺序依次触发。
- 目标值之间的间隔建议大于等于 50，否则在高频计数时可能无法正常进入中断。

- 中断程序：中断事件分为置位某个线圈，复位某个线圈，执行某个中断子程序，可在参数界面目标值的中断程序这一列下进行设置

✓ 置位线圈：输入 #(SET M10)代表置位线圈 M10，输入#(SET Y003)代表置位输出 Y003。
✓ 复位线圈：输入 #(RST M10)代表复位线圈 M10，输入#(RST Y003)代表复位输出 Y003。
✓ 中断子程序：在梯形图中创建好子程序 t1，在对应位置输入 t1 即可
✓ 空：无输入，该处计数到达该处目标值不会有任何操作

Tips:

- 目前仅支持输入 M,Y 类型的寄存器，并且 Y 不包括扩展模块的范围。
- 需要先在梯形图中创建好中断子程序，然后在目标值中输入该中断子程序的名称。
- 空操作对应的目标值可以作为循环模式下的计数最大值，计数到达该值后会自动循环。

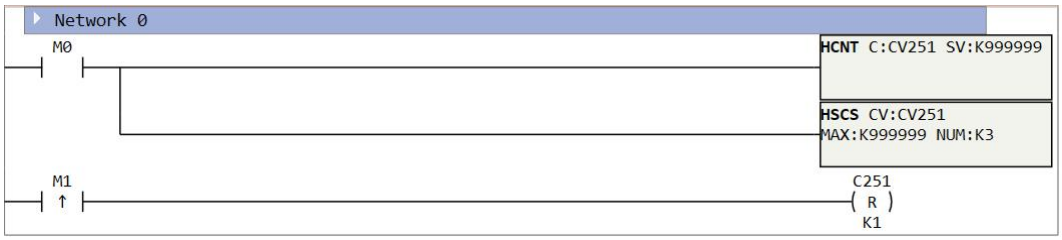
- 表格长度：该值会自动显示当前总共创建了多少个目标值，目前限制最大目标值段数为 100 段。

3. 梯形图实例

指令表：

```
Network 0
0    LD      M0
1    HCNT    CV251    K999999
2    HSCS    CV251    K999999    K3
3    POP
4    LDP     M1
5    RST     C251     K1
6    POP
```

梯形图：



实例说明：

当置位 M0 时，打开高速计数器的同时打开目标值中断指令 HCSC，该例中打开了 AB 相高速计数器 CV251，同时当 M1 上升沿时，高速计数器以及目标值中断指令 HCSC 复位，计数从零开始，并且重新从第一段开始触发目标值中断。

Tips:

- 循环模式圈数记录寄存器不会自动清零，复位计数器时需要用户手动清零

4.17.1 PID 指令

闭环控制系统 PID

对应命令	注释/说明	操作数（寄存器）
PID	闭环控制	PID 回路(K) 模式(K0-手动模式 K1-模糊 PID) 过程变量（D） 输入（D/AI） 输出（Y/M/D/AO） 目标值（D/K/H/LW）

FGs 能够进行 PID 控制，支持模糊 PID 和手动 PID，同时支持 PID 自整定。最多支持 10 路 PID 控制(其中温控路数为 8 路)

PID 是闭环控制系统的比例－积分－微分控制算法。

PID 控制器根据设定值（给定）与被控对象的实际值（反馈）的差值，按照 PID 算法计算出控制器的输出量，控制执行机构去影响被控对象的变化。

PID 控制是负反馈闭环控制，能够抑制系统闭环内的各种因素所引起的扰动，使反馈跟随给定变化。

根据具体项目的控制要求，在实际应用中有可能用到其中的一部分，比如常用的是 PI（比例－积分）控制，这时没有微分控制部分。

1. 指令说明

FGS 的 PID 功能是通过 PID 指令函数功能块实现。通过定时（按照采样时间）执行 PID 功能块，按照 PID 运算规律，根据当时的给定、反馈、比例－积分－微分数据，计算出控制量。

PID 功能块通过一个 PID 回路表交换数据，这个表是在 D 数据存储区中的开辟，长度为 17 字节。因此每个 PID 功能块在调用时需要指定 5 点：PID 控制回路号，控制回路表的起始地址（以 D 表示），PID 设定值,PID 输出(此地址与回路表输出类型相关,类型为 0 代表输出为位元件，1 代表字元件)，以及 PID 自整定采样点数设置。



图 1.1

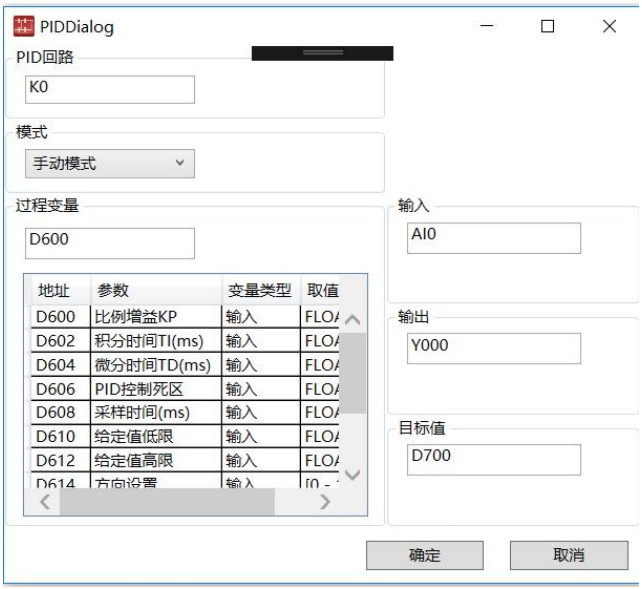


图 1.2

2.PID 各个参数解释

回路号:PID 功能块总共支持 10 个 PID 回路(其中温控支持 8 路)

过程变量：

比例参数 KP：控制器的输出与输入偏差值成比例关系。系统一旦出现偏差，比例调节立即产生调节作用以减少偏差。特点：过程简单快速、比例作用大，可以加快调节，减小误差；但是使系统稳定性下降，造成不稳定，有余差。数据类型为 32 位浮点数。

积分时间 TI(ms)：积分环节主要是用来消除静差，所谓静差，就是系统稳定后输出值和设定值之间的差值，积分环节实际上就是偏差累计的过程，把累计的误差加到原有系统上以抵消系统造成的静差。数据类型为 32 位浮点数

微分时间 TD(ms)：微分信号则反应了偏差信号的变化规律，或者说是变化趋势，根据偏差信号的变化趋势来进行超前调节，从而增加了系统的快速性。数据类型为 32 位浮点数

PID 控制死区：在 PID 产生控制作用时,当误差小于死区值时,PID 将停止计算,维持前一状态输出，数据类型为 32 位浮点数。

采样时间(ms)：此时间为数据输入和 PID 输出刷新时间,需要根据系统不同设置不同值,不能小于扫描周期，数据类型为 32 位浮点数。

给定值底限/给定值高限：PID 输出限幅于此区间。

方向：方向为 1 表示 PID 正向输出(加热),为 0 则表示反向输出。

自整定状态：0 表示整定完成,1 表示正在整定,2 表示整定失败。

整定采样点数：当采用 PID 自整定(即自整定状态为 1),此值代表 PID 在设定点数内未完成整定则判失败(若设置为 K30,则代表若在 30 个采样周期结束未完成 PID 自整定则失败,自整定状态位置 2)

输出量：当输出类型为 0 时，此值表示位元件开启时间,输出类型为 1 此值无效。

PID 输入：采样值存放地址

PID 输出：当此地址为位元件时(确保输出类型为 0)，PID 控制量为此元件正脉冲脉宽，当此地址为字元件时，PID 计算结果输出于此

PID 设定值:设定 PID 最终目标值

3.PID 模式

GC 系列 PID 分为手动模式和模糊 PID,用户可根据不同系统采用不同模式 PID

（1）手动 PID

根据用户输入参数执行 PID 运算输出，也可以先采用自整定完成 PID 参数设置

（2）PID 自整定

先决条件

（1） 进行自整定的回路必须处于自动模式(自整定状态为 1)
（2） PID 控制量能使系统在设定值附近产生振荡
（3） 在进行自整定之前，系统需要确定以下几个参数(采样时间，给定值高低限，PID 方向，整定状态置 1，输出类型，设定值)
（4） 如果在自整定期间(整定状态为 1)，系统在 N(N=自整定采样点数)个采样周期内没有在设定值附近产生振荡，则整定状态置 2，表示整定失败

自整定过程

自整定在整定状态位为 1 且 PID 指令使能时开始执行。系统首先会比较设定值和测量值大小，当测量值大于设定值时,PID 将负向输出，产生第一个零相交事件，当测量值小于设定值,PID 正向输出，再次产生一个零相交事件，在完成 10 次零相交事件后，整个 PID 整定序列采样完成，系统将会自动更新 PID 参数,从而系统开始 PID 控制，PID 整定状态将会清 0，表示整定完成。自整定过程变量的峰--峰值和零相交事件的产生速度都与控制过程的动态特性直接相关。

下图为 PID 自整定过程，系统首先会捕获 10 次零相交事件，10 次零相交事件满足 PID 整定序列则输出 PID 参数从而转为 PID 控制

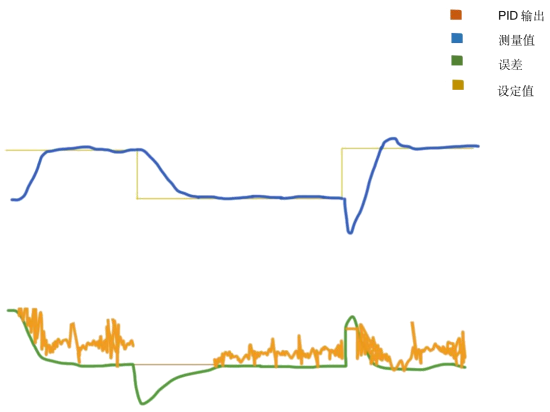


图 1.3

下图为 PID 控制过程,GC 系列 PID 指令加入了抗积分饱和，加快了系统反应速度

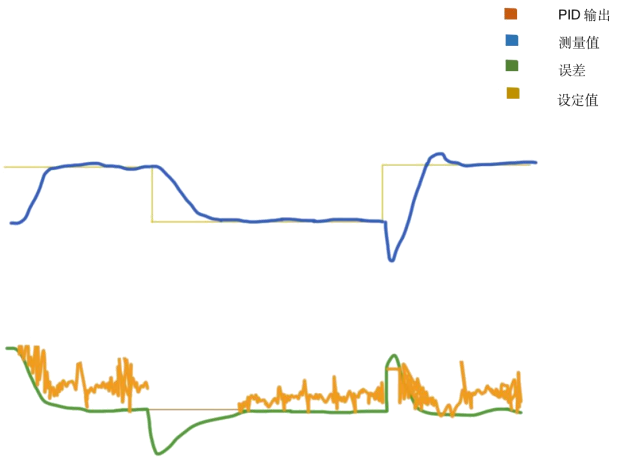


图 1.4

PID 自整定失败的原因

（1） 系统无法产生 10 次零相交事件,比如加热系统中加热器功率不足无法升温或者系统无法降温等

（2） 输入信号即测量值采样紊乱，系统无法获取整定序列

（3） 参数设置比如采样时间过长过短，高低限阈值不合理，方向，自整定状态设置错误以及整定采样次数过小从而失败

4.17.2 EPID 指令

1. 指令说明

显控 FGs 与 FGr 系列的 PLC 本体均支持 PID 闭环控制，具有以下特点：

1）输出可以选择 D 寄存器模拟量，也可以选择 Y/M 开关量。

2）可以通过自整定得到一组较为合适的 PID 参数，可以手动调整这些参数达到更好的控制效果。

3）自整定采用临界震荡法，可以从任意温度开始进行自整定。建议用户自整定开始前通过手动控制模式使得系统接近目标值，这样可以缩短自整定的时间。自整定期间根

据情况可能会自动调整某些参数。

4）可以设定 EPID 向导的正逆动作：温控场合下，正动作一般用于冷却控制，随着测定值变大，输出变大；

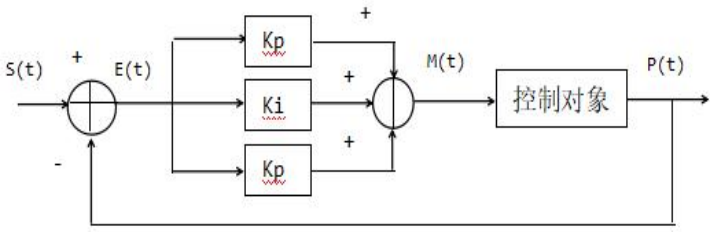
逆动作一般用于升温控制，随着测定值变大，输出变小。

5）PID 模式可选三种运行模式，运行期间可以互相切换。常规控制模式或自适应模糊 PID 控制模式由 EPID 向导控制输出，手动模式下由用户指定输出。手动切换到自动模式

时，EPID 向导会根据切换时的输出大小做平稳过渡处理。

6）参数输入非法会置位相应的错误码标志位，维持之前的参数值不变进行运算输出。

PID 控制规律：



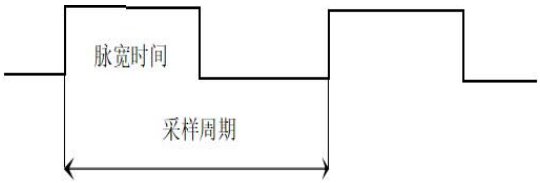
PID 控制系统原理图

$E(t) = S(t) - P(t)$ （1）

$M(t) = Kp[E(t) + 1/Ti\int E(t)dt + Td\ dE(t)/dt]$ （2）

（2）中 Kp、Ti、Td 分别为比例系数，积分时间，微分时间。

开关量：运算结果为 Y/M 端口在一个采样周期中接通所占用的时间，相当于端点输出的脉宽时间。占空比 = M(t)/T * 100%,其中 T 为采样时间。



2. 参数设置

用户可以通过 SamSoarII 软件上的 EPID 向导来创建一个 PID 控制回路。

EPID 向导的设置界面中可以输入相关参数，在监视模式下点击写入 PLC 可以将参数写入对应的 D 寄存器中。下载程序时，也会将界面上设置好的数据写入对应 D 寄存器中。当

然，用户也可以在使能 EPID 之前，利用 movf 等指令将参数写入对应的寄存器中，使用过程中用户可以随时修改相关的参数寄存器。如果用户修改了寄存器的内容，可以在监视模

式下通过读取 PLC 将参数读取到向导之中。

EPID 向导参数设置分为几个步骤，以下是具体的设置界面：

（1）回路设置



- 回路：目前除了 FGRS_C8X8T 和 FGm_64MT_AC 支持创建 8 个 PID 回路以外，其它的型号 EPID 向导支持创建 2 个 PID 回路，此处选择要创建的回路号。

（2）基本设置



- 回路号与回路名称：与之前的回路设置相对应，回路名称可以自己定义。
- 目标值：数据类型为 DWord（双字）。PID 控制的设定值 SV。
- 测量值：数据类型为 DWord（双字）。PID 控制的测量值 PV。
- 输出模式：分为开关量 Y/M，模拟量 D。
- 输出地址：根据输出模式，选择相应的输出地址。

（3）参数设置



- 采样时间：单位 ms。采样时间决定了 PID 采样和计算的周期。如果选择开关量输出，采样时间同时还是端口输出的周期。
- 比例增益：比例控制。
- 积分时间：单位 ms。积分时间越大，积分作用越小。
- 微分时间：单位 ms。微分时间越大，微分作用越大。
- 控制死区：当测量值与目标值偏差小于控制死区时，PID 控制不起作用，防止因为稳态时测量值 PV 的抖动导致不必要的计算
- 输出上下限值

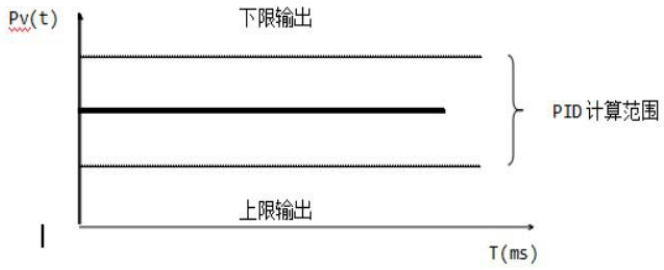
开关量输出：输出上限小于等于采样时间，输出上限与采样时间之比即为最大输出占空比，相当于加热设备的输出上限。同理，下限值决定了最小输出占空比。

- 模拟量输出：PID 输出结果处于上下限之间。
- 输出方向

逆方向：随着测量值 PV 减小，输出 MV 变大。温控场合用于升温控制。

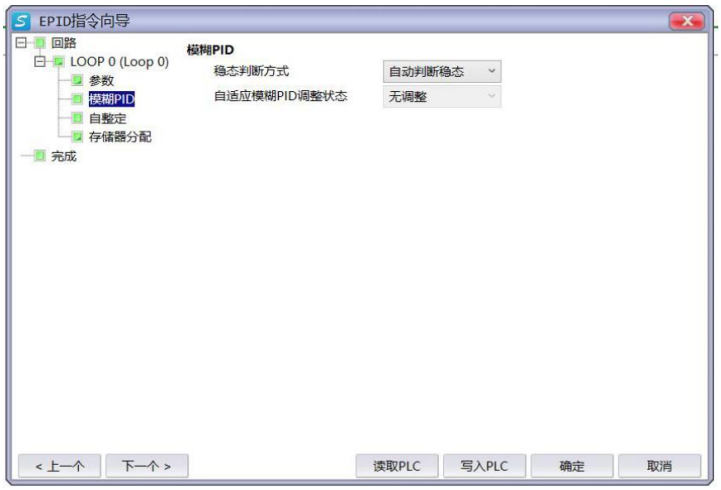
正方向：随着测量值 PV 减小，输出 MV 变小。温控场合用于冷却控制。

- PID 计算范围：当目标值 SV 与测量值 PV 的偏差小于 PID 计算范围时，PID 控制生效。否则根据偏差以及动作方向，按照最大或者最小限值输出。



- PID 控制方式：决定控制方式是 PID 或 PI 或 P 控制。不同的控制模式适用于不同的场景，PID 控制方式一般用于温控场合，PI 控制方式一般用于压力流量控制。
- PID 模式：确定 PID 运行的不同模式，具体信息见注意事项说明。

（4）自适应模糊 PID



- 稳态判断方式：默认选择自动判断，EPID 会自动判断系统是否达到稳态，达到稳态后将会进行自适应模糊 PID 参数调整计算。如果不勾选，需要用户手动向自适应 PID

调整状态寄存器输入 1 才能进行调整计算。

自动判断的稳态条件：偏差保持在 20°以内，并且来回震荡达到两个周期以上，如果偏差超过 20°，将会重新开始稳态判断，直至达到稳态。如果用户选择手动输入

开启自适应调整，也需要尽量使得系统先处于稳态条件下，再手动开启调整计算。稳态震荡幅度越小，调节时间越短，否则调节时间变长甚至可能调节失败。

- 自适应模糊 PID 调整状态：当自动判断稳态时，对应偏移 26 字的寄存器只读，1 代表处于调整状态，0 代表没有进行自适应调整并且以用户输入的初始参数进行 PID 控

制。当用户手动判断时，该寄存器只写，写入 1 代表进行自适应调整，清 0 代表按照用户输入的初始参数进行控制。

（5）自整定



- 开启自整定：在非监视模式下，可以通过勾选是否启用，点击写入 PLC 对应的寄存器。监视模式下可以显示当前自整定的状态。
- 自整定超时时间：单位 min。自整定最大持续时间，防止自整定时间过长。如果自整定成功，对应寄存器置 0，如果自整定失败，对应寄存器置 2。

Ps:先将自整定状态寄存器置 1，然后使能 PID 会自动进行自整定，整定完成后自动进行 PID 常规控制。自整定期间会根据升温的情况自动调整 PID 计算范围以及输出上下限这
几个参数。

（6）存储器分配



- 起始地址：指定用于存放 PID 参数的 D 寄存器起始地址。目前 PID 需要占用 50 个 D 寄存器，总计 100 个字节，用户可以设置在掉电保持区。

Ps:注意这些地址是连续的，确定后工程中其他地方不能再使用，否则会导致 PID 无法正常运行。

3.寄存器分配表

W 代表用户输入的参数；R 代表指令计算得出，用户只读的参数；R/W 代表可以由用户输入，也可能由程序自动写入。

地址	参数名称	参数类型	R/W	备注
0	采样时间	DWord	W	单位：ms。开关量默认值1000ms，模拟量默认值为0（必须由用户指定）
2	比例增益	float	R/W	设置为0，代表没有比例作用
4	积分时间	float	R/W	单位：ms。设置为0，代表没有积分作用。
6	积分系数	float	R	积分系数 = 比例增益X采样时间÷积分时间。实际参与运算的系数，只读。
8	微分时间	float	R/W	单位：ms。设置为0，代表没有微分作用。
10	微分系数	float	R	微分系数 = 比例增益X微分时间÷采样时间。实际参与运算的系数，只读。
12	控制死区	float	W	当测定值与目标值的偏差在死区之内时，PID不计算。
14	输出上限值	float	R/W	开关量对应最大输出占空比，同时也是自整定时的输出值。
16	输出下限值	float	R/W	开关量对应最低输出占空比，同时也是自整定时的输出值。
18	PID配置	hex	W	PID控制过程中用户可配置选项，具体信息见注意事项说明。
20	PID计算量	DWord	R	仅开关量有效，存储一个采样周期中端口接通时间。模拟量输出时无效。
22	PID计算范围	float	R/W	PID计算生效的偏差范围。
24	参数检测错误码	hex	R	错误码用于指示向导接通后输入参数合法性，具体信息见注意事项说明。
25	PID模式	word	W	用于切换PID的运行模式。0:PID自动控制模式 1：用户手动控制模式 2：自适应模糊 其余整数：暂停PID控制（输出维持不变）
26	自适应模糊PID调整状态	word	R/W	用于控制或者指示自适应调整状态。0:无调整状态 1：自适应调整状态。输入其他值无效
27-33	保留区域			程序保留区域。
34	自整定状态	word	R/W	0：自整定完成 1：自整定中 2：自整定失败。输入其他值无效。
36	自整定超时时间	dword	W	单位min。
40-49	保留区域			程序保留区域。

说明：
1）第一次创建 EPID 向导后，开关量输出时配置界面自动填入默认值，模拟量输出时默认值为 0，需要用户手动指定（否则会置位相应的 PID 参数检测错误码）。
2）所有的参数可以在监视模式下，通过写入 PLC 来写入相应的 D 寄存器中。如果用户手动更改了 D 寄存器中的参数，可以通过读取 PLC 来上传至向导中。下载时会自动将当前向导
中的参数写入到 PLC 中。如果用户在梯形图中手动修改了 D 寄存器中的参数，需要先读取再下载，防止梯形图中修改信息丢失。
3）梯形图中手动写入参数时请注意各个参数的数据类型。
4）用户必须在梯形图中给偏移地址 25 的寄存器赋值才能切换到手动/自动模式/自适应模糊 PID 模式。可以在运行过程中赋值从而切换模式。手动模式切换到自动模式时，如果
存在 PID 参数，会自动计算出的积分项，使得过渡平稳。

4.EPID 使用流程

A.PID 常规控制模式（无自整定）
1）参数设置：参考注意事项说明
- 设置采样时间
- 设置 PID 参数
- 设置输出上下限制
- 设置 PID 计算范围
- 设置 PID 控制方式和方向
- 分配存储首地址

2）PID 模式选择：PID 模式寄存器清 0
3）使能 EPID 指令
4）过程中可以直接失能 EPID 指令从而关闭 PID 控制，也可以向 PID 模式寄存器中写入大于 2 的整数，暂停 PID 控制，维持输出不变。

B.PID 常规控制模式（自整定）
1）参数设置：参考注意事项说明
- 设置采样时间
- 设置输出上下限制
- 设置 PID 计算范围
- 设置 PID 控制方式和方向
- 设置自整定超时时间
- 设置自整定偏移幅值
- 自整定状态寄存器写入 1
- 分配存储首地址

2）PID 模式选择：PID 模式寄存器清 0
3）使能 EPID 指令
4）过程中可以直接失能 EPID 指令从而关闭 PID 控制，也可以向 PID 模式寄存器中写入大于 2 的整数，暂停 PID 控制，维持输出不变。
5）自整定成功状态寄存器清 0，否则置 2。

C.自适应模糊 PID 控制模式

1）参数设置：参考注意事项说明
- 设置采样时间
- 设置输出上下限制
- 设置 PID 计算范围
- 设置 PID 控制方式和方向
- 设置是否自动稳态判断
- 设置初始 PID 参数
- 分配存储首地址

2）PID 模式选择：PID 模式寄存器写 2
3）使能 EPID 指令
4）如果自动判断稳态，调整状态寄存器置 1 代表进入稳态调整阶段，非自动判断用户需要手动置 1 进入调整阶段。
5）过程中可以直接失能 EPID 指令从而关闭 PID 控制，也可以向 PID 模式寄存器中写入大于 2 的整数，暂停 PID 控制，维持输出不变。

D.PID 手动控制模式
1）参数设置：参考注意事项说明
- 设置采样时间
- 设置 PID 参数（可选）
- 开关量向偏移 20 字的 PID 计算量中写入期望的输出量。模拟量向基本设置中的输出地址写入期望的输出量。
- 分配存储首地址

2）PID 模式选择：PID 模式寄存器写 1
3）使能 EPID 指令
4）过程中可以直接失能 EPID 指令从而关闭 PID 控制，也可以向 PID 模式寄存器中写入大于 2 的整数，暂停 PID 控制，维持输出不变。
5）如果设置了 PID 参数，切换到常规控制模式时会根据设置的 PID 参数进行平滑过渡

E.PID 运行中模式切换
- 切换到任意模式时，需要同时设置好该模式下需要设置的相关参数以及有关的配置。
- 可以在运行过程中从常规模式或者自适应模式切换到手动模式。
- 可以从手动模式切换到自动模式或者自适应模式，并且有平滑过渡处理。
- 常规控制和自适应控制模式之间也可以互相切换，切换到自适应模式时，会重新进行稳态判断。

5.注意事项

1.采样时间
1）合法性判断：采样时间必须大于扫描周期。开关量的采样时间必须大于等于输出上限。如果不满足合法性判断，参数错误检测码中相应的位会置 1，输入合法自动清零。
2）温控等变化较为缓慢的场合一般建议采样时间在 1000ms 左右；压力、流量等变化较为迅速的场合一般建议在 100ms 左右。
3）运行过程中尽量不要更改采样时间大小，尤其是在稳定阶段，这会引起 PID 参数的变化，从而导致 PID 控制波动。

2.输出上下限值

1）合法性判断:输出上限要大于输出下限。开关量输出时，输出上限要小于等于采样时间，并且上下限都要大于等于 0。如果不满足合法性判断，参数错误检测码中相应的位会
置 1，输入合法自动清零。
2）如果不清楚设备的运行功率大小，试运行时上限可以适当降低。开关量时向导界面会根据采样时间和上下限值计算出输出占空比的大小供参考。3）合适的上下限输出值会使得自整定进程缩短。上限值过大或者下限值过小，会导致交替输出时，系统震荡幅度过大，导致自整定时间较长。上限值过小或者下限值过大，可
能会导致系统无法在目标值附近来回震荡从而导致自整定失败。

3.PID 计算范围

1）合法性判断：PID 计算范围必须要大于控制死区。如果不满足合法性判断，参数错误码中相应的位会置 1，输入合法自动清零。
2）PID 范围越小，全功率输出的范围就越大，达到目标值的速度就越快。对于惯性较大的滞后系统，可能也会引起较大的超调，初始可将 PID 范围调大，后续根据调节曲线可以
适当调小 PID 范围，加快调节速度。

4.PID 配置

Bit	15-4	3	2	1	0
Value	保留	SD	TYPE		Dir

1）Dir: 0：逆动作 1：正动作。逆动作时 E = Sv -Pv，温控场合常用于加热控制。正动作时 E=Pv - Sv，温控场合常用于冷却控制。
2）TYPE: 00:PID 控制 01：PI 控制 10：P 控制。PID 控制一般适用于大滞后场合例如温控，PI 控制适用于相应速度较快的场合比如压力流量等。
3）SD：0：开启稳态自动判断 1：关闭稳态自动判断。自适应模糊 PID 控制首先需要用户根据一组经验参数达到稳态，该位默认开启自动判断稳态。如果置位，则关闭自动判断
稳态，此时需要用户自行开启自适应 PID 调整。

5.PID 模式

1）PID 常规控制模式：当处于该模式时，PID 按照用户输入的 PID 参数进行调节控制。该模式下可以在运行过程中将自整定状态寄存器置 1 从而启动自整定。
2）PID 手动控制模式：当处于该模式时，PID 按照用户输入输出量进行输出。如果此时存在 PID 参数，会根据 PID 参数在切换到常规模式时或自适应模糊 PID 模式时做平稳过渡处
理。
3）自适应模糊 PID 模式：当处于该模式时，PID 按照用户给出的 PID 初始参数进行稳态判断，进入稳态后自动或者用户手动进入自适应调整状态。
4）输入大于 2 的正整数，PID 将会进入暂停计算状态，此时输出不变，也不会进行任何 PID 计算。

6.自整定

1）自整定超时时间不低于 60min，该时间仅用于超时判断，不会影响实际自整定效果。

2）自整定开启前确认输出上限和下限值是否大致合理，过大的上限值可能导致超调过大，过小的上限值可能导致系统无法在目标值附近来回震荡。

3）建议开启自整定之前先利用手动模式输出，使得测定值接近目标值，此时开启自整定会大大缩短自整定的时间。

7.自适应模糊 PID 控制

1）概述：自适应模糊 PID 控制根据当前偏差 e,以及本次偏差与前次偏差差值 ec，通过一定的推理过程，不断调整 Kp,Ki,kd 三个参数。调整后的参数利用增量式的 PID 计算公式计算输出量。

2）使用流程：

✧ 参数设置：采样时间，PID 计算范围，输出上下限。当偏差大于 PID 计算范围时，按照最大或最小功率输出。合适的输出上下限以及 PID 计算范围可以使得系统尽快进入稳态。

✧ 初始 PID 参数：用户给出一组初始 PID 参数，通过调整这一组初始参数，使得系统达到稳态，需要进行稳态判断。

✧ 自适应 PID 调整：自动或者手动进入该阶段后，将自动调整 PID 参数达到理想的控制效果。

注意：未到达稳态要求的情况下开启自适应调整的时间会增加并且存在调节不成功的更可能性。

3）初始 PID 参数稳态调节

✧ 温控场合 PID 参数参考：

注：根据实际的输入输出量纲的对比确定 PID 参数的量级。例如 p 的大小范围是根据输出范围和输入范围比较得出的，温度场合输出量为在上下限之间，对比温度变化的大致范围不难得出 P 的大小范围。

✧ 经验法调整参数，以下提供温控场景的调节方法，仅供参考，实际可能需要反复调节多次才能达到理想效果。

● 比例系数 Kp 从 0.1 开始向上调整，使得系统能够在目标值附近来回震荡且不发散，记录下此时的 Kp 值。调节 Kp 的值可以改变震荡幅度大小，但是过大会使得系统震荡次
数和幅度变大，甚至出现发散的情况。

● 积分时间从一个较大的值开始慢慢减小（积分时间越小积分作用越强），直到系统能够在一定震荡次数内趋于稳定。调节 TI 的值可以改变趋于稳定过程中震荡的次数，
但过大也会导致系统出现大幅度震荡。

● 一般微分时间设置为 0。调节微分时间逐渐变大，减小系统的震荡幅度。调节 Td 可以使得变化平稳，减小峰值和震荡次数，但会使得调节时间变长，系统响应速度下降。

4）稳态判断：偏差在 20°以内来回震荡两个周期以上即视为达到稳态，会自动在自适应 PID 调节状态寄存器中填入 1。如果用户选择手动判断，需要自己在该寄存器中填入 1 来
启动调整环节。自动稳态判断期间温度偏差超过 20°将会重新开始稳态判断，直至稳态判断成立。未达到稳态之前，可以随时调节 PID 参数来使得进入稳态的速度加快。

5）自适应 PID 调节阶段：EPID 会在这个阶段自动微调 PID 参数达到理想状态。

注意：1.重新调整初始参数进行稳态判断的方法：用户手动判断时直接清空调整状态寄存器，调整初始参数。自动判断时需要先改成手动判断，然后清空状态寄存器，调整初始参数再改成自动稳态判断。

2.改变目标值：改变目标值后与当前采样值偏差超过 20°，自动判断模式下会清空状态寄存器，同时以初始 PID 参数进行控制并重新进行稳态判断，所以请注意改变目标
值后可能需要根据实际情况调整输出上下限和 PID 参数使得系统更快的进入稳态。手动判断的情况下需要用户自己清空状态寄存器，达到稳态后再次手动写入 1 启动调整，否则
可能调节效果不理想。

8.参数错误码检测

Bit	15-3	2	1	0
Value	保留	RA	HL	ST

ST：1 表示采样时间不合法 0 代表合法
HL：1 代表输入高限或低限不合法 0 代表合法
RA：1 代表 PID 计算范围不合法 0 代表合法
注意：第一次输入参数不合法时，PID 不进行运算，也没有输出。运行过程中参数更改不合法，EPID 向导将会保持上一次的合法值进行运算输出同时会有相应错误码提示。具体的不合法信息请见各自参数的注意事项说明。

6.PID 控制案例

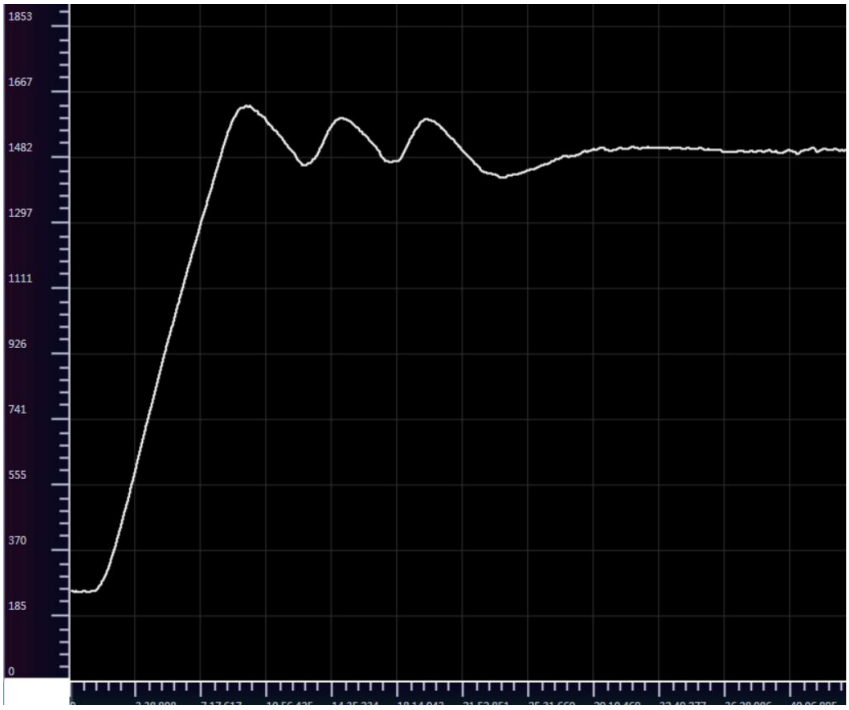
一、电热铁温控案例

场景：利用开关量 Y 控制一个电热铁温度维持在一个恒定温度。室温 25℃，目标温度设定为 150℃。
1）参数设置：

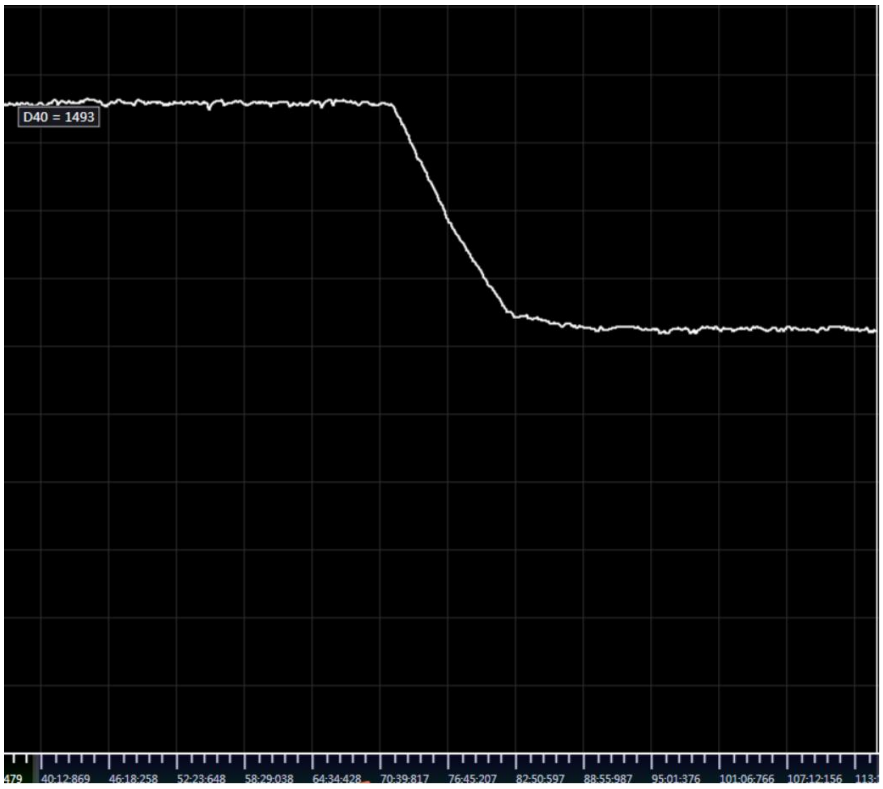
① 采样时间 1000ms；
② 比例，积分，微分时间均设置为 0，由自整定得出参数；
③ 输出上限设置 500，下限设 0；
④ 逆方向；
⑤ PID 计算范围 300，实际计算范围在 1200-1800；
⑥ 开启自整定；
⑦ 自整定超时时间 80min；
⑧ 自整定赋值偏移 150，即自整定时最大峰值温度超过 165℃后会自动降低输出上限，减小峰值偏移；
⑨ 存储起始地址 D1000

2）自整定以及 PID 调节序列

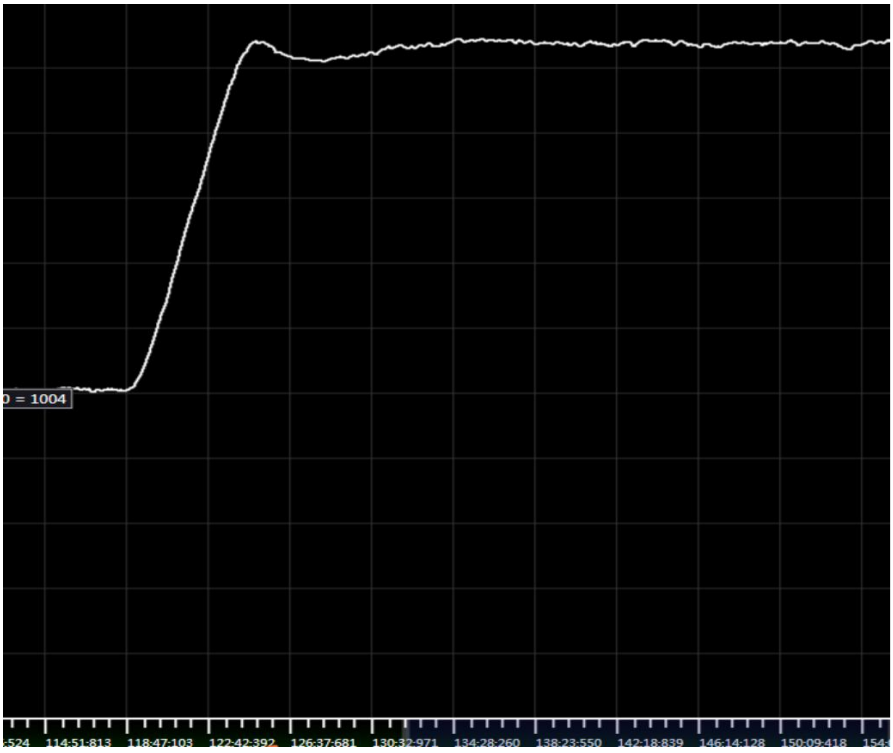
◆ 从室温 25℃开始加热，18min 左右完成自整定，最终 28min 左右温度稳定在了 150℃±1 以内。自整定完成后会有一个周期的震荡，然后趋于稳定。



◆ 2.改变目标值为 100℃，大约经过 20min 后趋于稳定，新的目标值处稳定迅速。



◆ 3.将目标值改为 180℃，大约经历 20min 后温度趋于稳定，达到新的目标值处大约有±4℃的波动，然后趋于稳定



分析：

- 本例中目标温度 150℃自整定完成后会产生一个周期的较大震荡，本例中大概有±7℃的偏差，观察本例中自整定时震荡幅度比较小，所以可以调大输出上限重新进行自整定。
- 如果后续调节出现震荡幅度较大的情况，可以适当减小比例增益，同时适当增大微分时间来降低震荡速度。如果出现趋向目标值时大幅震荡次数较多的情况，请适当增大微分时间。
- 如果出现自整定失败，表明自整定的时间大于超时时间，可能是系统由于加热功率等原因出现无法升温或者降温的情况，否则请加大超时时间。

二、管道加热压缩空气

场景一：利用模拟量 4-20ma 输出，通过信号放大模块，控制加热设备加热管道内流动的压缩空气温度维持恒定。目标温度在 500°。

1）参数设置：扩展模块 4-20ma 输出，数字量范围 0-32767

1.采样时间 100ms；

2.比例，积分，微分时间均设置为 0；

3.输出上限设置 21000，下限设 0；（输出上限功率约为 65%）

4.逆方向，死区控制 2，控制方式 PID；

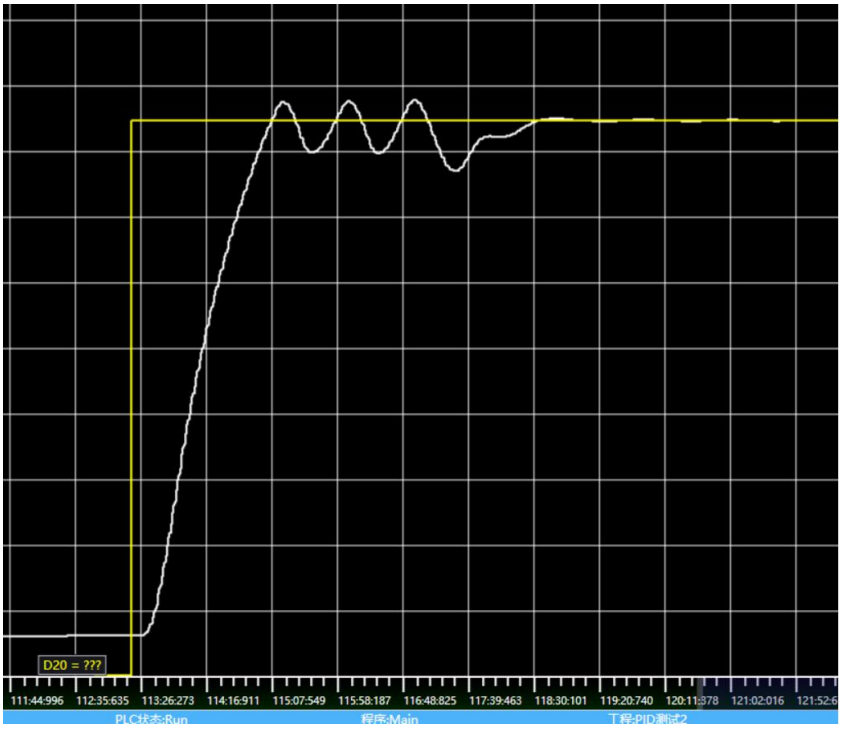
5.PID 计算范围 400，实际计算范围在 4600-5400；

6.开启自整定；

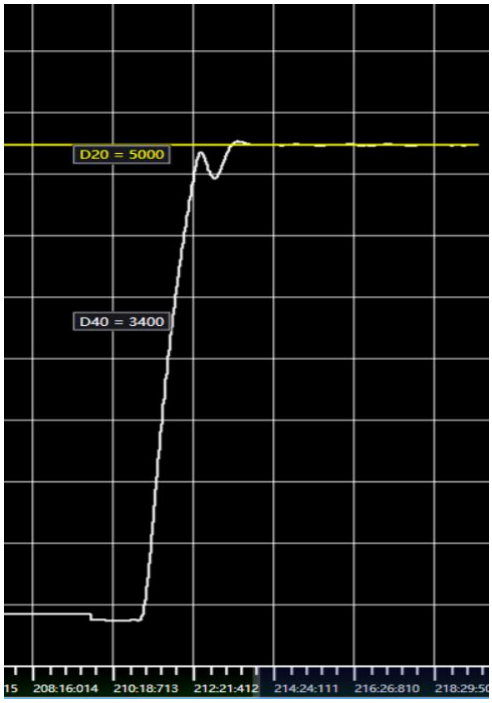
7.自整定超时时间 80min；

8.存储起始地址 D1000

2）从室温开始自整定，持续时间 4min，调节稳定时间 1min，总计 5min 趋于 500°稳定。
整定参数： P:34.27956 I:25350 D:6084。趋于稳定后温度稳定在±1°以内。

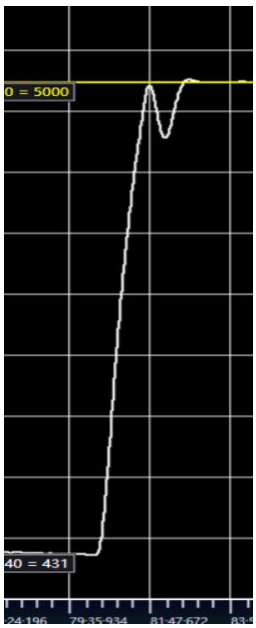


3）在自整定得出的参数下，从 50°加热到 500°。调节稳定时间在 2min 左右，控制效果良好。
本例中，由于加热的是流动的压缩空气，所以相比较一般的温控场合，系统响应较快，自整定的速度也快很多。
观察曲线后，可以通过缩小 PID 计算范围可以进一步加快趋于稳定的速度。

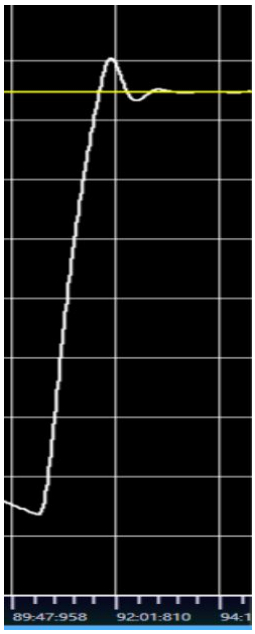


4）在之前自整定的参数下，先用手动模式控制系统到达某个测定值，然后再切换自动模式。手动模式切换到自动模式时，向导会自动做平滑过渡处理

- 手动模式以 22000（65%功率）加热到 460，切换自动模式。切换到自动后，系统温度经过波动后趋于稳定。可以通过加大手动控制的功率来进一步减小波动。

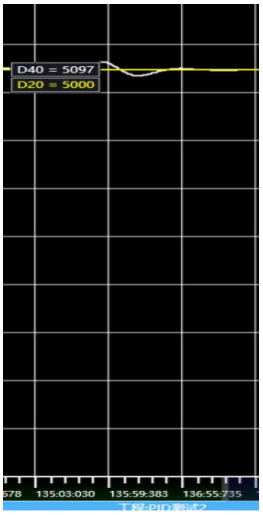


- 手动模式以 22000（65%功率）加热到 490°，再切换到自动模式，波动依然存在，但是趋于稳定的速度加快了。



先手动控制，再切换自动控制，调节稳定的效果可以更好。可以适当调整手动控制的功率使得过渡更加平稳。

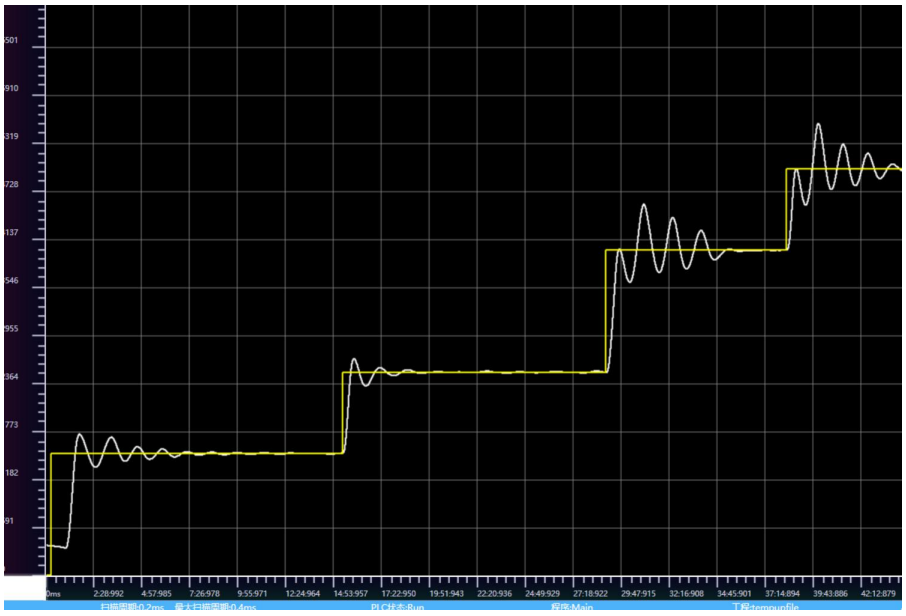
5）在这个配置的基础上，稳定阶段调节气阀开度使得系统发生震荡，可以观察 PID 的调节情况。
突然减小气阀，使得系统温度升高了 10°，1min 的时间内经过一轮震荡后系统重新趋于稳定。PID 调节在稳态时具有抗扰动的能力，可以通过调节微分时间进一步加强抗干扰能力。



场景二：利用开关量输出，自适应模糊 PID 控制加热管道内流动的压缩空气温度维持恒定。目标温度 150°-250°-400°-500°。

1）参数设置：
1.采样时间 500ms；
2.比例，积分，微分时间均设置为 0；
3.输出上限分别设置 200-300-400-400，下限设 0；
4.逆方向，死区控制 2，控制方式 PID；
5.PID 计算范围 400-500-600-700；
6.比例增益设置为 0.2，积分时间设置为 20000ms，微分时间设置为 0，初始参数使得系统在各个目标值处能够进入稳态；
7.PID 模式设置为 2，选择自适应模糊 PID 控制方式，自动判断稳态；
8.存储起始地址 D1000；

2）以下为实际的控制曲线图



分析总结
1）目标值 200°，300°时，分别在初始 PID 参数控制下,经过两次 20°以内的震荡后进入稳态，随后进入参数自适应调整阶段。
2）当目标温度切换到 400°和 500°时，由于输出上限低较低，第一次升温幅度较低，导致震荡次数较多，进入稳态的速度变慢了。可以通过调节输出上限值和 PID 计算范围使
得第一次升温产生震荡，同时可以适当调整比例增益来减小震荡幅度。
3）当目标温度切换到较高的 500°时，此时应该加大输出上限，同时根据情况调整比例增益，使得系统的震荡幅度达到稳态要求。

5. 函数功能块

SAMSOARII 第二代编辑软件是在第一代编辑软件 SAMSOAR Developer 的基础上，针对显控推出的新一代编译型可编程控制器（PLC）的组态程序的图形界面编辑工具。与之前的版本相比，在保留了其原有功能的基础上，对运行性能和用户体验等方面上做了大幅度的改进。

本公司推出的 PLC 产品简单易学，应用范围广泛，其帮助系统能够帮助您掌握 SAMSOAR II 软件的基本操作与用法。

帮助文档包括：

10、 快速入门(查看 [标题编号])
11、 编辑界面
12、 指令系统
13、 寄存器
14、 仿真界面
15、 通信协议
16、 硬件手册
17、 系统设置
18、 联系方式

5.1 C 语言入门

5.1.1 C 语言程序类型

1. 整型量

整型量包括整型常量、整型变量。整型常量就是整常数。在 C 语言中，使用的整常数有八进制、十六进制和十进制三种。

2. 整型常量

1）.八进制整常数八进制整常数必须以 0 开头，即以 0 作为八进制数的前缀。数码取值为 0～7。八进制数通常是无符号数。

以下各数是合法的八进制数：

015(十进制为 13) 0101(十进制为 65) 0177777(十进制为 65535)

以下各数不是合法的八进制数：

256(无前缀 0) 03A2(包含了非八进制数码) -0127(出现了负号)

2）.十六进制整常数的前缀为 0X 或 0x。其数码取值为 0~9，A~F 或 a~f。

以下各数是合法的十六进制整常数：

0X2A(十进制为 42)　0XA0 (十进制为 160)　0XFFFF (十进制为 65535)

以下各数不是合法的十六进制整常数：

5A (无前缀 0X)　0X3H (含有非十六进制数码)

3）.十进制整常数没有前缀。其数码为 0～9。

以下各数是合法的十进制整常数：

237 -568 65535 1627

以下各数不是合法的十进制整常数：

023 (不能有前导 0) 23D (含有非十进制数码)。

在程序中是根据前缀来区分各种进制数的。因此在书写常数时不要把前缀弄错造成结果不正确。

4）.整型常数的后缀在 16 位字长的机器上，基本整型的长度也为 16 位，因此表示的数的范围也是有限定的。十进制无符号整常数的范围为 0～65535，有符号数为-32768～+32767。八进制无符号数的表示范围为 0～0177777。十六进制无符号数的表示范围为 0X0～0XFFFF 或 0x0～0xFFFF。如果使用的数超过了上述范围，就必须用长整型数来表示。长整型数是用后缀“L”或“l”来表示的。例如：

十进制长整常数　158L (十进制为 158) 358000L (十进制为-358000)

八进制长整常数　012L (十进制为 10) 077L (十进制为 63) 0200000L (十进制为 65536)。

十六进制长整常数　0X15L (十进制为 21) 0XA5L (十进制为 165) 0X10000L (十进制为 65536)。

长整数 158L 和基本整常数 158 在数值上并无区别。但对 158L，因为是长整型量，C 编译系统将为它分配 4 个字节存储空间。而对 158，因为是基本整型，只分配 2 个字节的存储空间。因此在运算和输出格式上要予以注意，避免出错。无符号数也可用后缀表示，整型常数的无符号数的后缀为“U”或“u”。例如： 358u,0x38Au,235Lu 均为无符号数。前缀，后缀可同时使用以表示各种类型的数。如 0XA5Lu 表示十六进制无符号长整数 A5，其十进制为 165。

3. 整型变量

整型变量可分为以下几类：

1.基本型，类型说明符为 int，在内存中占 4 个字节，其取值为基本整常数。

2.短整量，类型说明符为 short。在内存中占 2 个字节，其取值为短整常数。

3.长整型，类型说明符为 long ，在内存中占 8 个字节，其取值为长整常数。

4.无符号型，类型说明符为 unsigned。

无符号型又可与上述三种类型匹配而构成：

(1) 无符号基本型类型说明符为 unsigned int 或 unsigned。

(2) 无符号短整型类型说明符为 unsigned short

(3) 无符号长整型类型说明符为 unsigned long

各种无符号类型量所占的内存空间字节数与相应的有符号类型量相同。但由于省去了符号位，故不能表示负数。 下表列出了 ARM 中各类整型量所分配的内存字节数及数的表示范围。

类型说明符	数的范围	分配字节数
int	-2147483648~2147483647	■■■■
unsigned int	0~4294967295	■■■■
short	-32768~32767	■■
unsigned short	0~65536	■■
long	-922337203685477808~922337203685477807	■■■■■■■■
unsigned long	0~18446744073709551615	■■■■■■■■

4. 整型变量的说明

变量说明的一般形式为： 类型说明符 变量名标识符，变量名标识符，...；例如：

int a,b,c； (a,b,c 为整型变量)

long x,y； (x,y 为长整型变量)

unsigned p,q； (p,q 为无符号整型变量)

在书写变量说明时，应注意以下几点：

1）. 允许在一个类型说明符后，说明多个相同类型的变量。各变量名之间用逗号间隔。类型说明符与变量名之间至少用一个空格间隔。

2）. 最后一个变量名之后必须以“；”号结尾。

3）. 变量说明必须放在变量使用之前。一般放在函数体的开头部分。

```
int a, b;
short c;
short d = 100;
a = d - 20;
b = a + d;
c = a + b + d;
d = d - a + c - b;
```

5. 实型常量

实型也称为浮点型。实型常量也称为实数或者浮点数。在C语言中，实数只采用十进制。它有二种形式：

1）十进制数形式，由数码 0~ 9 和小数点组成。例如：0.0，.25，5.789，0.13，5.0，300.，-267.8230 等均为合法的实数。

2）指数形式，由十进制数，加阶码标志“e”或“E”以及阶码(只能为整数，可以带符号)组成。其一般形式为 a E n (a 为十进制数，n 为十进制整数)其值为 a*10,n 如： 2.1E5 (等于 2.1*10,5), 3.7E-2 (等于 3.7*10,)-2*) 0.5E7 (等于 0.5*10,7), -2.8E-2 (等于-2.8*10,)-2*)以下不是合法的实数 345 (无小数点) E7 (阶码标志 E 之前无数字) -5 (无阶码标志) 53.-E3 (负号位置不对) 2.7E (无阶码)。

标准C允许浮点数使用后缀。后缀为“f”或“F”即表示该数为浮点数。如 356f 和 356.是等价的。

6. 实型变量

实型变量分为两类：单精度型和双精度型，其类型说明符为 float 单精度说明符，double 双精度说明符。在 Turbo C 中单精度型占 4 个字节(32 位)内存空间，其数值范围为 3.4E-38～3.4E+38，只能提供七位有效数字。双精度型占 8 个字节(64 位)内存空间，其数值范围为 1.7E-308～1.7E+308，可提供 16 位有效数字。

实型变量说明的格式和书写规则与整型相同。

例如： float x,y； (x,y 为单精度实型量)

double a,b,c； (a,b,c 为双精度实型量)

实型常数不分单、双精度，都按双精度 double 型处理。

```
float a = 32;
float b;
double d;
b = 12345678;
```

```
d = b * 100;
d = d + a;
d = d + 58.123456;
```

7. 字符型量

字符型量包括字符常量和字符变量。

字符常量是用单引号括起来的一个字符。例如'a','b','=','+','?'都是合法字符常量。在C语言中，字符常量有以下特点：

1. 字符常量只能用单引号括起来，不能用双引号或其它括号。
2. 字符常量只能是单个字符，不能是字符串。
3. 字符可以是字符集中任意字符。但数字被定义为字符型之后就不能参与数值运算。如'5'和 5 是不同的。'5'是字符常量，不能参与运算。

8. 字符变量

字符变量的取值是字符常量，即单个字符。字符变量的类型说明符是 char。字符变量类型说明的格式和书写规则都与整型变量相同。

例如：

char a,b； 每个字符变量被分配一个字节的内存空间，因此只能存放一个字符。字符值是以 ASCII 码的形式存放在变量的内存单元之中的。如 x 的

十进制 ASCII 码是 120，y 的十进制 ASCII 码是 121。对字符变量 a,b 赋予'x'和'y'值： a='x'； b='y'；实际上是在 a,b 两个单元内存放 120 和 121 的二进制代码：

a 0 1 1 1 1 0 0 0

b 0 1 1 1 1 0 0 1

所以也可以把它们看成是整型量。 C语言允许对整型变量赋以字符值，也允许对字符变量赋以整型值。在输出时，允许把字符变量按整型量输出，也允许把整型量按字符量输出。整型量为二字节量，字符量为单字节量，当整型量按字符型量处理时，只有低八位字节参与处理。

```
char a = 49;
char b;
char d;
b = a + 10;
d = a + b;

char c1, c2;
c1 = 'a';
c2 = 'b';
c1 = c1 - 32;
c2 = c2 - 32;
```

9. 字符串常量

字符串常量是由一对双引号括起的字符序列。例如： "CHINA" ，"C program:" ， "$12.5" 等都是合法的字符串常量。字符串常量和字符常量是不同的量。它们之间主要有以下区别：

1. 字符常量由单引号括起来，字符串常量由双引号括起来。
2. 字符常量只能是单个字符，字符串常量则可以含一个或多个字符。
3. 可以把一个字符常量赋予一个字符变量，但不能把一个字符串常量赋予一个字符变量。在C语言中没有相应的字符串变量。
4. 字符常量占一个字节的内存空间。字符串常量占的内存字节数等于字符串中字节数加 1。增加的一个字节

中存放字符"\0"(ASCII 码为 0)。这是字符串结束的标志。例如，字符串 "C program"在内存中所占的字节为：C program\0。字符常量'a'和字符串常量"a"虽然都只有一个字符，但在内存中的情况是不同的。

'a'在内存中占一个字节，可表示为：a

"a"在内存中占二个字节，可表示为：a\0 符号常量

10. 符号常量

在 C 语言中，可以用一个标识符来表示一个常量，称之为符号常量。符号常量在使用之前必须先定义，其一般形式为：

#define 标识符 常量

其中#define 也是一条预处理命令，称为宏定义命令，其功能是把该标识符定义为其后的常量值。一经定义，以后在程序中所有出现该标识符的地方均代之以该常量值。习惯上符号常量的标识符用大写字母，变量标识符用小写字母，以示区别。

```
#define PI 3.14159
void main()
{
    float s, r;
    r = 5;
    s = PI * r * r;
    printf("s=%f\n", s);
}
```

由宏定义命令定义 PI 为 3.14159 s,r 定义为实数 5->r PI*r*r->s

显示程序结果 float s,r; r=5; s=PI*r*r；本程序在主函数之前由宏定义命令定义 PI 为 3.14159，在程序中即以该值代替 PI 。s=PI*r*r 等效于 s=3.14159*r*r。应该注意的是，符号常量不是变量，它所代表的值在整个作用域内不能再改变。也就是说，在程序中，不能再用赋值语句对它重新赋值。

5.1.2 变量初值和类型转换

1. 变量赋初值

在程序中常常需要对变量赋初值，以便使用变量。语言程序中可有多种方法，在定义时赋以初值的方法，这种方法称为初始化。在变量说明中赋初值的一般形式为：

类型说明符 变量 1= 值 1，变量 2= 值 2，……； 例如：

```
int a = b = c = 5;
float x = 32, y = 3f, z = 0.75;
char ch1 = 'K', ch2 = 'P';
```

应注意，在说明中不允许连续赋值，如 a=b=c=5 是不合法的。

2. 变量类型的转换

变量的数据类型是可以转换的。转换的方法有两种， 一种是自动转换，一种是强制转换。

自动转换发生在不同数据类型的量混合运算时，由编译系统自动完成。自动转换遵循以下规则：

1）.若参与运算量的类型不同，则先转换成同一类型，然后进行运算。

2）.转换按数据长度增加的方向进行，以保证精度不降低。如 int 型和 long 型运算时，先 把 int 量转成 long 型后再进行运算。

3）.所有的浮点运算都是以双精度进行的，即使仅含 float 单精度量运算的表达式，也要先转换成 double 型，

再作运算。

4）.char 型和 short 型参与运算时，必须先转换成 int 型。

5）.在赋值运算中，赋值号两边量的数据类型不同时，赋值号右边量的类型将转换为左边量的类型。如果右边量的数据类型长度左边长时，将丢失一部分数据，这样会降低精度，丢失的部分按四舍五入向前舍入。

3. 强制类型转换

强制类型转换是通过类型转换运算来实现的。其一般形式为： (类型说明符) (表达式) 其功能是把表达式的运算结果强制转换成类型说明符所表示的类型。例如： (float) a 把 a 转换为实型(int)(x+y) 把 x+y 的结果转换为整型在使用强制转换时应注意以下问题：

1）.类型说明符和表达式都必须加括号(单个变量可以不加括号)，如把(int)(x+y)写成(int)x+y 则成了把 x 转换成 int 型之后再与 y 相加了。

2）.无论是强制转换或是自动转换，都只是为了本次运算的需要而对变量的数据长度进行的临时性转换，而不改变数据说明时对该变量定义的类型。

5.1.3 基本运算符和表达式

C 语言中运算符和表达式数量之多， 在高级语言中是少见的。正是丰富的运算符和表达式使 C 语言功能十分完善。 这也是 C 语言的主要特点之一。

C 语言的运算符不仅具有不同的优先级， 而且还有一个特点，就是它的结合性。在表达式中， 各运算量参与运算的先后顺序不仅要遵守运算符优先级别的规定，还要受运算符结合性的制约，以便确定是自左向右进行运算还是自右向左进行运算。这种结合性是其它高级语言的运算符所没有的，因此也增加了 C 语言的复杂性。

运算符的种类 C 语言的运算符可分为以下几类：

1.算术运算符，用于各类数值运算。包括加(+)、减(-)、乘(*)、除(/)、求余(或称模运算，%)、自增(++)、自减(--)共七种。

2.关系运算符，用于比较运算。包括大于(>)、小于(<)、等于(==)、大于等于(>=)、小于等于(<=)和不等于(!=)六种。

3.逻辑运算符，用于逻辑运算。包括与(&&)、或(||)、非(!)三种。

4.位操作运算符，参与运算的量，按二进制位进行运算。包括位与(&)、位或(|)、位非(~)、位异或(^)、左移(<<)、右移(>>)六种。

5.赋值运算符，用于赋值运算，分为简单赋值(=)、复合算术赋值(+=,-=,*=,/=,%=)和复合位运算赋值(&=,|=,^=,>>=,<<=)三类共十一种。

6.条件运算符，这是一个三目运算符，用于条件求值(?:)。

7.逗号运算符，用于把若干表达式组合成一个表达式(，)。

8.指针运算符，用于取内容(*)和取地址(&)二种运算。

9.求字节数运算符，用于计算数据类型所占的字节数(sizeof)。

10.特殊运算符，有括号()，下标[],等几种。

5.1.4 一维数值组

数组在程序设计中，为了处理方便， 把具有相同类型的若干变量按有序的形式组织起来。这些按序排列的同类数据元素的集合称为数组。在C语言中，数组属于构造数据类型。一个数组可以分解为多个数组元素，这些数组元素可以是基本数据类型或是构造类型。因此按数组元素的类型不同，数组又可分为数值数组、字符数组、指针数组、结构数组等各种类别。

本章介绍数值数组和字符数组，其余的在以后各章陆续介绍。数组类型说明在C语言中使用数组必须先进行类型说明。数组说明的一般形式为： 类型说明符数组名 [常量表达式]，……； 其中，类型说明符是任一种基本数据类型或构造数据类型。数组名是用户定义的数组标识符。方括号中的常量表达式表示数据元素的个数，也称为数组的长度。例如：

```
int   a[10]; //  说明整型数组a，有 10 个元素。
float b[10], c[20]; //  说明实型数组b，有 10 个元素，实型数组c，有 20 个元素。
char ch[20]; //  说明字符数组ch，有 20 个元素。
```

对于数组类型说明应注意以下几点：

1.数组的类型实际上是指数组元素的取值类型。对于同一个数组，其所有元素的数据类型都是相同的。

2.数组名的书写规则应符合标识符的书写规定。

3.数组名不能与其它变量名相同，例如：

```
void main()
{
    int a;
    float a[10];
    //……
}
```

是错误的。

4.方括号中常量表达式表示数组元素的个数，如a[5]表示数组a有5个元素。但是其下标从0开始计算。因此5个元素分别为a[0],a[1],a[2],a[3],a[4]。

5.不能在方括号中用变量来表示元素的个数， 但是可以是符号常数或常量表达式。例如：

```
#define FD 5
void main()
{
    int a[3+2],b[7+FD];
    //……
}
```

是合法的。但是下述说明方式是错误的。

```
void main()
{
    int n=5;
    int a[n];
    //……
}
```

6.允许在同一个类型说明中，说明多个数组和多个变量。

例如： int a,b,c,d,k1[10],k2[20];

1. 数组元素的表示方法

数组元素是组成数组的基本单元。数组元素也是一种变量，其标识方法为数组名后跟一个下标。下标表示了元素在数组中的顺序号。数组元素的一般形式为：数组名[下标] 其中的下标只能为整型常量或整型表达式。如为小数时，C编译将自动取整。例如，a[5],a[i+j],a[i++]都是合法的数组元素。数组元素通常也称为下标变量。必须先定义数组，才能使用下标变量。在C语言中只能逐个地使用下标变量，而不能一次引用整个数组。

初始化赋值的一般形式为： static 类型说明符数组名[常量表达式]={值，值……值}； 其中 static 表示是静态存储类型，C 语言规定只有静态存储数组和外部存储数组才可作初始化赋值(有关静态存储，外部存储的概念在第五章中介绍)。在{ }中的各数据值即为各元素的初值， 各值之间用逗号间隔。例如： static int a[10]={ 0,1,2,3,4,5,6,7,8,9 }； 相当于 a[0]=0；a[1]=1...a[9]=9；

2. C 语言对数组的初始赋值

1.可以只给部分元素赋初值。当{ }中值的个数少于元素个数时，只给前面部分元素赋值。例如： static int a[10]={0,1,2,3,4}；表示只给 a[0]～a[4]5 个元素赋值，而后 5 个元素自动赋 0 值。

2.只能给元素逐个赋值，不能给数组整体赋值。 例如给十个元素全部赋 1 值，只能写为：static int a[10]={1,1,1,1,1,1,1,1,1,1}；而不能写为： static int a[10]=1；

3.如不给可初始化的数组赋初值，则全部元素均为 0 值。

4. 如给全部元素赋值，则在数组说明中， 可以不给出数组元素的个数。例如： static int a[5]={1,2,3,4,5}；可写为： static int a[]={1,2,3,4,5}；动态赋值可以在程序执行过程中，对数组作动态赋值。 这时可用循环语句配合 scanf 函数逐个对数组元素赋值。

3. 字符数组

用来存放字符量的数组称为字符数组。 字符数组类型说明的形式与前面介绍的数值数组相同。例如： char c[10]；由于字符型和整型通用，也可以定义为 int c[10]但这时每个数组元素占 2 个字节的内存单元。字符数组也可以是二维或多维数组，例如： char c[5][10]；即为二维字符数组。字符数组也允许在类型说明时作初始化赋值。例如： static char c[10]={`c`,` `,`p`,`r`,o`,g`,r`,`a`,`m`}；赋值后各元素的值为：数组 C c[0]c[1]c[2]c[3]c[4]c[5]c[6]c[7]c[8]c[9]其中 c[9]未赋值，由系统自动赋予 0 值。 当对全体元素赋初值时也可以省去长度说明。例如：static char c[]={`c`,` `,`p`,`r`,`o`,`g`,`r`,`a`,`m`}；这时 C 数组的长度自动定为 9。

C 语言允许用字符串的方式对数组作初始化赋值。例如：

```
static char c[]={'c',' ','p','r','o','g','r','a','m'};
```

可写为：

```
static char c[]={"C program"};
```

或去掉{}写为：

```
static char c[]="C program";
```

用字符串方式赋值比用字符逐个赋值要多占一个字节， 用于存放字符串结束标志'\0'。上面的数组 c 在内存中的实际存放情况为：C program\0`\0'是由 C 编译系统自动加上的。由于采用了`\0'标志，所以在用字符串赋初值时一般无须指定数组的长度，而由系统自行处理。在采用字符串方式后，字符数组的输入输出将变得简单方便。除了上述用字符串赋初值的办法外，还可用 printf 函数和 scanf 函数一次性输出输入一个字符数组中的字符串，而不必使用循环语句逐个地输入输出每个字符。

5.1.5 小结

1. C 的数据类型

基本类型，构造类型，指针类型，空类型

2.基本类型的分类及特点

类型说明符	字节	数值范围
字符型 char	1	0~255
基本整型 int	4	-214783648～214783647
短整型 short	2	-32768～32767
长整型 long	8	-922337203685477808～922337203685477807
无符号型 unsigned	4	0～4294967295
无符号长整型 unsigned long	8	0～1844744073709551615
单精度实型 float	4	3/4E-38～3/4E+38
双精度实型 double	8	1/7E-308～1/7E+308

3. 常量后缀

L 或 l　长整型
U 或 u　无符号数
F 或 f　浮点数

4. 常量类型

整数，长整数，无符号数，浮点数，字符，字符串，符号常数，转义字符。

5. 数据类型转换

·自动转换

在不同类型数据的混合运算中，由系统自动实现转换， 由少字节类型向多字节类型转换。不同类型的量相互赋值时也由系统自动进行转换，把赋值号右边的类型转换为左边的类型。

·强制转换

由强制转换运算符完成转换。

6. 运算符优先级和结合性

一般而言，单目运算符优先级较高，赋值运算符优先级低。 算术运算符优先级较高，关系和逻辑运算符优先级较低。 多数运算符具有左结合性，单目运算符、三目运算符、赋值。

7. 表达式

表达式是由运算符连接常量、变量、函数所组成的式子。 每个表达式都有一个值和类型。 表达式求值按运算符的优先级和结合性所规定的顺序进行。

8. 数组

1. 数组是程序设计中最常用的数据结构。数组可分为数值数组(整数组，实数组)，字符数组以及后面将要介绍的指针数组，结构数组等。

2. 数组可以是一维的，二维的或多维的。

3. 数组类型说明由类型说明符、数组名、数组长度 (数组元素个数)三部分组成。数组元素又称为下标变量。数组的类型是指下标变量取值的类型。

4. 对数组的赋值可以用数组初始化赋值，输入函数动态赋值和赋值语句赋值三种方法实现。对数值数组不能用赋值语句整体赋值、输入或输出，而必须用循环语句逐个对数组元素进行操作。

5.2 C 语言程序设计初步

5.2.1 C 程序的语句

C 程序的执行部分是由语句组成的。 程序的功能也是由执行语句实现的。C 语句可分为以下五类：

1. 表达式语句

表达式语句由表达式加上分号“；”组成。其一般形式为： 表达式； 执行表达式语句就是计算表达式的值。例如： x=y+z；赋值语句 y+z； 加法运算语句，但计算结果不能保留，无实际意义 i++；自增 1 语句，i 值增 1

2. 控制语句

控制语句用于控制程序的流程， 以实现程序的各种结构方式。

它们由特定的语句定义符组成。C 语言有九种控制语句。可分成以下三类：

(1) 条件判断语句，if 语句，switch 语句

(2) 循环执行语句，do while 语句，while 语句，for 语句

(3) 转向语句，break 语句，goto 语句，continue 语句，return 语句

3. 空语句

只有分号“；”组成的语句称为空语句。 空语句是什么也不执行的语句。在程序中空语句可用来作空循环体。例如 while(getchar()!='\n'); 本语句的功能是，只要从键盘输入的字符不是回车则重新输入。这里的循环体为空语句。

4. 赋值语句

赋值语句是由赋值表达式再加上分号构成的表达式语句。 其一般形式为： 变量=表达式； 赋值语句的功能和特点都与赋值表达式相同。它是程序中使用最多的语句之一。在赋值语句的使用中需要注意以下几点：

1.由于在赋值符“=”右边的表达式也可以又是一个赋值表达式，因此，下述形式变量=(变量=表达式)；是成立的，从而形成嵌套的情形。其展开之后的一般形式为：变量=变量=…=表达式； 例如：

a=b=c=d=e=5；

按照赋值运算符的右接合性，因此实际上等效于：

e=5；d=e；c=d；b=c；a=b；

2.注意在变量说明中给变量赋初值和赋值语句的区别。给变量赋初值是变量说明的一部分，赋初值后的变量与其后的其它同类变量之间仍必须用逗号间隔，而赋值语句则必须用分号结尾。

3.在变量说明中，不允许连续给多个变量赋初值。 如下述说明是错误的： int a=b=c=5 必须写为 int a=5,b=5,c=5； 而赋值语句允许连续赋值

4.注意赋值表达式和赋值语句的区别。赋值表达式是一种表达式，它可以出现在任何允许表达式出现的地方，而赋值语句则不能。

下述语句是合法的：

if((x=y+5)>0) z=x；

语句的功能是，若表达式 x=y+5 大于 0 则 z=x。下述语句是非法的：

if((x=y+5;)>0) z=x；

因为 x=y+5；是语句，不能出现在表达式中。

5.2.2 分支结构程序

1. 关系运算符和表达式

在程序中经常需要比较两个量的大小关系， 以决定程序下一步的工作。比较两个量的运算符称为关系运算符。在C语言中有以下关系运算符：

< 小于
<= 小于或等于
> 大于
>= 大于或等于
== 等于
!= 不等于

关系运算符都是双目运算符，其结合性均为左结合。关系运算符的优先级低于算术运算符，高于赋值运算符。在六个关系运算符中，<,<=,>,>=的优先级相同，高于==和!=，==和!=的优先级相同。

关系表达式的一般形式为：

表达式 关系运算符 表达式

例如：

a+b>c-d, x>3/2, 'a'+1<c, -i-5*j==k+1；

都是合法的关系表达式。由于表达式也可以又是关系表达式。因此也允许出现嵌套的情况，例如：a>(b>c),a!=(c==d)等。关系表达式的值是“真”和“假”，用“1”和“0”表示。

2. 逻辑运算符和表达

逻辑运算符C语言中提供了三种逻辑运算符 && 与运算 || 或运算 ！非运算 与运算符&&和或运算符||均为双目运算符，具有左结合性。非运算符!为单目运算符，具有右结合性。逻辑运算符和其它运算符优先级的关系可表示如下：

按照运算符的优先顺序可以得出：

a>b && c>d 等价于 (a>b) && (c>d)

!b==c||d<a 等价于 ((!b)==c)||(d<a)

a+b>c && x+y<b 等价于 ((a+b)>c) && ((x+y)<b)

逻辑运算的值也为“真”和“假”两种，用“1”和“0 ”来表示。其求值规则如下：

1.与运算&&参与运算的两个量都为真时，结果才为真，否则为假。例如，5>0 && 4>2,由于 5>0 为真，4>2 也为真，相与的结果也为真。

2.或运算||参与运算的两个量只要有一个为真，结果就为真。两个量都为假时，结果为假。例如：5>0||5>8, 由于 5>0 为真，相或的结果也就为真

3.非运算!参与运算量为真时，结果为假；参与运算量为假时，结果为真。例如：!(5>0)的结果为假。

虽然 C 编译在给出逻辑运算值时，以“1”代表“真”，“0 ”代表“假”。 但反过来在判断一个量是为“真”还是为“假”时，以“0”代表“假”，以非“0”的数值作为“真”。例如：由于 5 和 3 均为非“0”因此 5&&3 的值为“真”，即为 1。

又如：5||0 的值为“真”，即为 1。

逻辑表达式逻辑表达式的一般形式为：表达式 逻辑运算符 表达式 其中的表达式可以又是逻辑表达式，从而组成了嵌套的情形。例如：(a&&b)&&c 根据逻辑运算符的左结合性，上式也可写为： a&&b&&c 逻辑表达式的值是式中各种逻辑运算的最后值，以“1”和“0”分别代表“真”和“假”。

3. if 语句

用 if 语句可以构成分支结构。它根据给定的条件进行判断，以决定执行某个分支程序段。C 语言的 if 语句有三种基本形式。

1.第一种形式为基本形式

```
if(表达式) 语句；
```

其语义是：如果表达式的值为真，则执行其后的语句，否则不执行该语句。

2.第二种形式为 if-else 形式

```
if(表达式)
    语句 1；
else
    语句 2；
```

其语义是：如果表达式的值为真，则执行语句 1，否则执行语句 2 。

输入两个整数，输出其中的大数。改用 if-else 语句判别 a,b 的大小，若 a 大，则输出 a，否则输出 b。

3.第三种形式为 if-else-if 形式

前二种形式的 if 语句一般都用于两个分支的情况。当有多个分支选择时，可采用 if-else-if 语句，其一般形式为：

```
if(表达式 1)
    语句 1；
else if(表达式 2)
    语句 2；
else if(表达式 3)
    语句 3；
…
else if(表达式 m)
    语句 m；
else
    语句 n；
```

其语义是：依次判断表达式的值，当出现某个值为真时，则执行其对应的语句。然后跳到整个 if 语句之外继续执行程序。 如果所有的表达式均为假，则执行语句 n 。 然后继续执行后续程序。

在使用 if 语句中还应注意以下问题

(1) 在三种形式的 if 语句中，在 if 关键字之后均为表达式。 该表达式通常是逻辑表达式或关系表达式， 但也可以是其它表达式，如赋值表达式等，甚至也可以是一个变量。例如： if(a=5) 语句；if(b) 语句； 都是允许的。只要表达式的值为非 0，即为“真”。如在 if(a=5)…；中表达式的值永远为非 0，所以其后的语句总是要执行的，当然这种情况在程序中不一定会出现，但在语法上是合法的。

又如，有程序段：

```
if (a = b)
    printf("%d", a);
else
    printf("a=0");
```

本语句的语义是，把 b 值赋予 a，如为非 0 则输出该值，否则输出“a=0”字符串。这种用法在程序中是经常出现的。

(2) 在 if 语句中，条件判断表达式必须用括号括起来，在语句之后必须加分号。

(3) 在 if 语句的三种形式中，所有的语句应为单个语句，如果要想在满足条件时执行一组(多个)语句，则必须把这一组语句用{} 括起来组成一个复合语句。但要注意的是在}之后不能再加分号。例如：

```
if(a>b)
{
    a++;
    b++;
}
else
{
    a=0;
    b=10;
}
```

4. 条件运算符和条件表达式

如果在条件语句中，只执行单个的赋值语句时， 常可使用条件表达式来实现。不但使程序简洁，也提高了运行效率。

条件运算符为?和：，它是一个三目运算符，即有三个参与运算的量。由条件运算符组成条件表达式的一般形式为：

表达式 1 ? 表达式 2 : 表达式 3

其求值规则为：如果表达式 1 的值为真，则以表达式 2 的值作为条件表达式的值，否则以表达式 2 的值作为整个条件表达式的值。 条件表达式通常用于赋值语句之中。

例如条件语句：if(a>b) max=a；else max=b；可用条件表达式写为 max=(a>b)?a:b；执行该语句的语义是：如 a>b 为真，则把 a 赋予 max，否则把 b 赋予 max。

使用条件表达式时，还应注意以下几点：

1. 条件运算符的运算优先级低于关系运算符和算术运算符，但高于赋值符。因此 max=(a>b)?a:b 可以去掉括号而写为 max=a>b?a:b

2. 条件运算符?和：是一对运算符，不能分开单独使用。

3. 条件运算符的结合方向是自右至左。

5.2.3 Switch 语句

C 语言还提供了另一种用于多分支选择的 switch 语句，其一般形式为：

switch (表达式) {

case 常量表达式 1: 语句 1；
case 常量表达式 2: 语句 2；
…
case 常量表达式 n: 语句 n；
default : 语句 n+1；
}

其语义是：计算表达式的值。 并逐个与其后的常量表达式值相比较，当表达式的值与某个常量表达式的值相等时，即执行其后的语句，然后不再进行判断，继续执行后面所有 case 后的语句。 如表达式的值与所有 case 后的常量表达式均不相同时，则执行 default 后的语句。

在使用 switch 语句时还应注意以下几点：

1.在 case 后的各常量表达式的值不能相同，否则会出现错误。

2.在 case 后，允许有多个语句，可以不用{}括起来。

3.各 case 和 default 子句的先后顺序可以变动，而不会影响程序执行结果。

4.default 子句可以省略不用。

5.2.4 循环结构顺序

循环结构是程序中一种很重要的结构。其特点是，在给定条件成立时，反复执行某程序段，直到条件不成立为止。给定的条件称为循环条件，反复执行的程序段称为循环体。C 语言提供了多种循环语句，可以组成各种不同形式的循环结构。

1. while 语句

while 语句的一般形式为： **while**(表达式)语句； 其中表达式是循环条件，语句为循环体。

while 语句的语义是：计算表达式的值，当值为真(非 0)时， 执行循环体语句。

使用 while 语句应注意以下几点：

1.while 语句中的表达式一般是关系表达或逻辑表达式，只要表达式的值为真(非 0)即可继续循环。

2.循环体如包括有一个以上的语句，则必须用{}括起来，组成复合语句。

3.应注意循环条件的选择以避免死循环。

2. do-while 语句

do-while 语句的一般形式为：

do 语句；
while(表达式)；

其中语句是循环体，表达式是循环条件。

do-while 语句的语义是：

先执行循环体语句一次， 再判别表达式的值，若为真(非 0)则继续循环，否则终止循环。

do-while 语句和 while 语句的区别在于 do-while 是先执行后判断，因此 do-while 至少要执行一次循环体。而 while 是先判断后执行，如果条件不满足，则一次循环体语句也不执行。

while 语句和 do-while 语句一般都可以相互改写。

在本例中，循环条件改为--n，否则将多执行一次循环。这是由于先执行后判断而造成的。

对于 do-while 语句还应注意以下几点：

1.在 if 语句，while 语句中，表达式后面都不能加分号， 而在 do-while 语句的表达式后面则必须加分号。

2.do-while 语句也可以组成多重循环，而且也可以和 while 语句相互嵌套。

3.在 do 和 while 之间的循环体由多个语句组成时，也必须用{}括起来组成一个复合语句。

4.do-while 和 while 语句相互替换时，要注意修改循环控制条件。

3. For 语句

for 语句是 C 语言所提供的功能更强，使用更广泛的一种循环语句。其一般形式为：

for(表达式 1；表达式 2；表达 3) 语句；

表达式 1　　通常用来给循环变量赋初值，一般是赋值表达式。也允许在 for 语句外给循环变量赋初值，此时可以省略该表达式。

表达式 2　　通常是循环条件，一般为关系表达式或逻辑表达式。

表达式 3　　通常可用来修改循环变量的值，一般是赋值语句。

这三个表达式都可以是逗号表达式， 即每个表达式都可由多个表达式组成。三个表达式都是任选项，都可以省略。

一般形式中的“语句”即为循环体语句。for 语句的语义是：

1.首先计算表达式 1 的值。

2.再计算表达式 2 的值，若值为真(非 0)则执行循环体一次，否则跳出循环。

3.然后再计算表达式 3 的值，转回第 2 步重复执行。在整个 for 循环过程中，表达式 1 只计算一次，表达式 2 和表达式，3 则可能计算多次。循环体可能多次执行，也可能一次都不执行。

在使用 for 语句中要注意以下几点：

1.for 语句中的各表达式都可省略，但分号间隔符不能少。如：for(；表达式；表达式)省去了表达式

<1>for(表达式；；表达式)省去了表达式

<2>for(表达式；表达式；)省去了表达式

<3>for(；；)省去了全部表达式。

2.在循环变量已赋初值时，可省去表达式 1，如例 3.27 即属于这种情形。如省去表达式 2 或表达式 3 则将造成无限循环， 这时应在循环体内设法结束循环。

3.循环体可以是空语句。

```
#include <stdio.h>
void main()
{
    int n = 0;
    printf("input a string:\n");
    for (; getchar() != '\n'; n++);
    printf("%d", n);
}
```

本例中，省去了 for 语句的表达式 1，表达式 3 也不是用来修改循环变量，而是用作输入字符的计数。这样，就把本应在循环体中完成的计数放在表达式中完成了。因此循环体是空语句。应注意的是，空语句后的分号不可少，如缺少此分号，则把后面的 printf 语句当成循环体来执行。反过来说，如循环体不为空语句时， 决不能在表达式的括号后加分号， 这样又会认为循环体是空语句而不能反复执行。这些都是编程中常见的错误，要十分注意。

5.2.5 break 语句

break 语句只能用在 switch 语句或循环语句中， 其作用是跳出 switch 语句或跳出本层循环，转去执行后面的程序。由于 break 语句的转移方向是明确的，所以不需要语句标号与之配合。break 语句的一般形式为： break；

上面例题中分别在 switch 语句和 for 语句中使用了 break 语句作为跳转。使用 break 语句可以使循环语句有多个出口，在一些场合下使编程更加灵活、方便。

5.2.6 continue 语句

continue 语句只能用在循环体中，其一般格式是：

continue；

其语义是：结束本次循环，即不再执行循环体中 continue 语句之后的语句，转入下一次循环条件的判断与执行。应注意的是，本语句只结束本层本次的循环，并不跳出循环。

5.2.7 小结

1.从程序执行的流程来看， 程序可分为三种最基本的结构： 顺序结构，分支结构以及循环结构

2.程序中执行部分最基本的单位是语句。C 语言的语句可分为五类：

(1)表达式语句　　任何表达式末尾加上分号即可构成表达式语句，常用的表达式语句为赋值语句。

(2)函数调用语句　由函数调用加上分号即组成函数调用语句。

(3)控制语句　　用于控制程序流程，由专门的语句定义符及所需的表达式组成。主要有条件判断执行语句，循环执行语句，转向语句等。

(4)复合语句　　由{}把多个语句括起来组成一个语句。复合语句被认为是单条语句，它可出现在所有允许出现语句的地方，如循环体等。

(5)空语句　　仅由分号组成，无实际功能。

3.关系表达式和逻辑表达式是两种重要的表达式， 主要用于条件执行的判断和循环执行的判断。

4. C 语言提供了多种形式的条件语句以构成分支结构。

(1)if 语句主要用于单向选择。

(2)if-else 语句主要用于双向选择。

(3)if-else-if 语和 switch 语句用于多向选择。

这几种形式的条件语句一般来说是可以互相替代的。

5. C 语言提供了三种循环语句。

(1)for 语句主要用于给定循环变量初值， 步长增量以及循环次数的循环结构。

(2)循环次数及控制条件要在循环过程中才能确定的循环可用 while 或 do-while 语句。

(3)三种循环语句可以相互嵌套组成多重循环。循环之间可以并列但不能交叉。

(4)可用转移语句把流程转出循环体外，但不能从外面转向循环体内。

(5)在循环程序中应避免出现死循环，即应保证循环变量的值在运行过程中可以得到修改，并使循环条件逐步变为假，从而结束循环。

6. C 语言语句小结

名称	一般形式
简单语句	表达式语句表达式；
空语句	；

条件语句　　if(表达式)语句；
if(表达式)语句 1；　else 语句 2；
if(表达式 1)语句 1；　else if(表达式 2)　语句 2…else 语句　n；
开关语句　　switch(表达式){ case 常量表达式:　语句…default:　语句；　}
循环语句　　while 语句
while(表达式)语句；
for 语句　for(表达式 1；表达式 2；表达式 3)语句；
break 语句　break；
continue 语句　continue；
return　语句　return(表达式)；

5.3　函数功能块的设计

5.3.1 创建一个新的函数功能块

1.右键点击工程管理器“工程”-->“程序”-->“函数功能块”文件夹（或者其中的子文件夹），弹出的菜单中点击“新建函数块”。

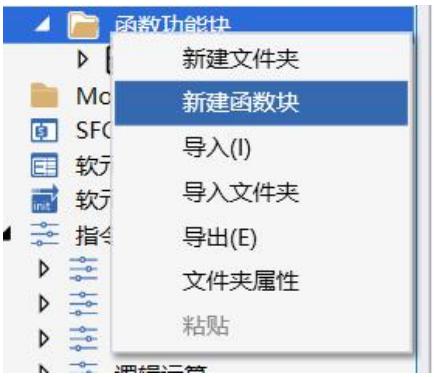


图 1.1 新建函数功能块

2.弹出新建函数块的向导窗口，窗口顶部是函数块名称，作者和版本的填写文本框，左侧是内部函数的列表，右侧是左侧选中函数的参数。点击【新建函数】可以在左侧列表新建一个函数，点击【删除函数】可以删除选中的函数，点击【确定】生成新的函数功能块，点击【取消】取消生成并返回。



图 1.2 创建向导

3.双击新建的函数功能块的项，主界面会显示代码编程的文本界面。

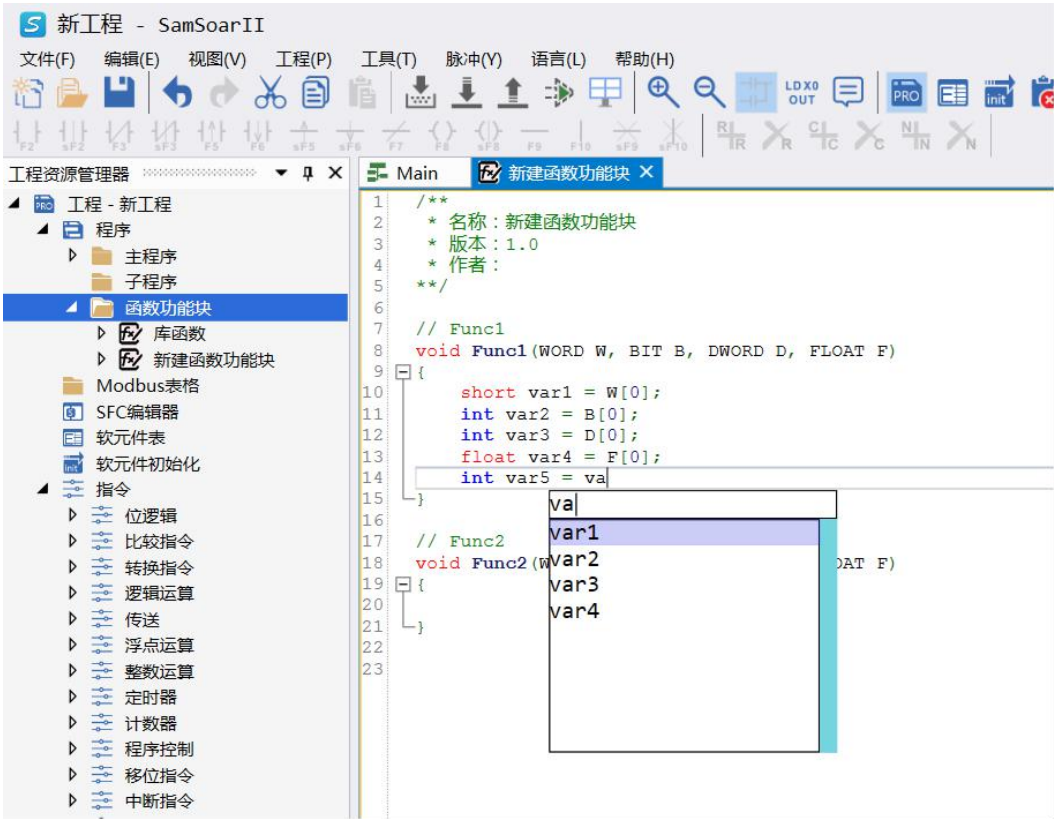
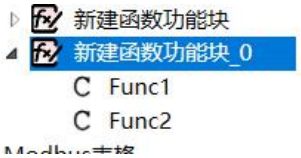


图 1.3

3.编写函数功能块的代码。在编程的过程中，SAMSOARII 支持自动缩进，括号补全以及代码补全。代码补全包括 C 语言关键字，函数参数，局部变量以及全局常量（变量）。弹出的代码补全选择框中的词，都包含现在正在输入的词作为子词。例如输入单词 va，那么 var1,var2,var3,var4 都包含单词 va，所以都显示在表中。可以通过**鼠标点击**或者方向键控制光标回车确定的**键盘控制**来选择要补全的词。

4.编写完毕后，可以在工程管理器中函数块的子项中查看实现的所有函数。



可以通过 F4 编译快捷键将梯形图和函数块一起编译，**SAMSOARII 会优先编译梯形图然后再编译函数块，所以需要保证之前梯形图完全正确。**当然也可以通过顶部菜单栏“工具”-->“程序检测”-->“函数功能块检测”来跳过之前的步骤直接检查函数功能块。

5.刚完成的 C 语言程序，进行第一次编译会出现一些编译错误，**这属于正常现象**。

```
void Func1(BIT arg1, WORD arg2, DWORD arg3, FLOAT arg4)
{
    if (arg5[0]) *arg3 = *arg2;
    if (arg6[1]) *arg4 = *arg3;
    if (arg7[2]) *arg2 = *arg4;
    break;
}

void Func2(BIT arg1, WORD arg2, DWORD arg3, FLOAT arg4)
{
    if (arg8[0]) *arg2 = *arg3;
    if (arg9[1]) *arg3 = *arg4;
    if (arg1[2]) *arg4 = *arg2;
    int i;
    for ( i = 0 ; j < 100 ; ii++)
        aarg1 = argg2[i];
    continue;
}
```

错误列表

错误 11　警告 0

	说明	程序	坐标
■	'arg6' undeclared (first use in this function)	新建函数功能块_0	(4, 6)
■	'arg7' undeclared (first use in this function)	新建函数功能块_0	(5, 6)
■	break statement not within loop or switch	新建函数功能块_0	(6, 2)
■	'arg8' undeclared (first use in this function)	新建函数功能块_0	(11, 6)
■	'arg9' undeclared (first use in this function)	新建函数功能块_0	(12, 6)
■	'j' undeclared (first use in this function)	新建函数功能块_0	(15, 16)
■	'ii' undeclared (first use in this function)	新建函数功能块_0	(15, 26)
■	'aarg1' undeclared (first use in this function)	新建函数功能块_0	(16, 3)
■	'argg2' undeclared (first use in this function)	新建函数功能块_0	(16, 11)
■	continue statement not within a loop	新建函数功能块_0	(17, 2)

图 1.4 错误信息

当 SAMSOARII 检查到所有的错误后，会报告给错误列表并弹出。双击错误列表中的一项，主界面会自动导航到相应位置。修改完毕后再编译，错误列表中的所有之前发生的错误项均会消失，并且不会再自动弹出，而会弹出提示成功的对话框。

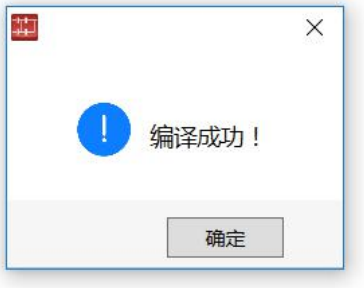


图 1.5 编译成功提示

注意，虽然错误信息不存在，但是警告信息可能依然存在，虽然也视为编译通过，但是这个时候谨慎起见还是要查看错误列表并确认（通过顶部菜单栏“视图”-->“错误列表”来打开窗口）。

```
void Func1(BIT arg1, WORD arg2, DWORD arg3, FLOAT arg4)
{
    if (arg1[0]) *arg3 = arg2;
    if (arg1[1]) *arg4 = *arg3;
    if (arg1[2]) *arg2 = arg4;
}

void Func2(BIT arg1, WORD arg2, DWORD arg3, FLOAT arg4)
{
    if (arg1[0]) *arg2 = arg3;
    if (arg1[1]) *arg3 = arg4;
    if (arg1[2]) *arg4 = *arg2;
}
```

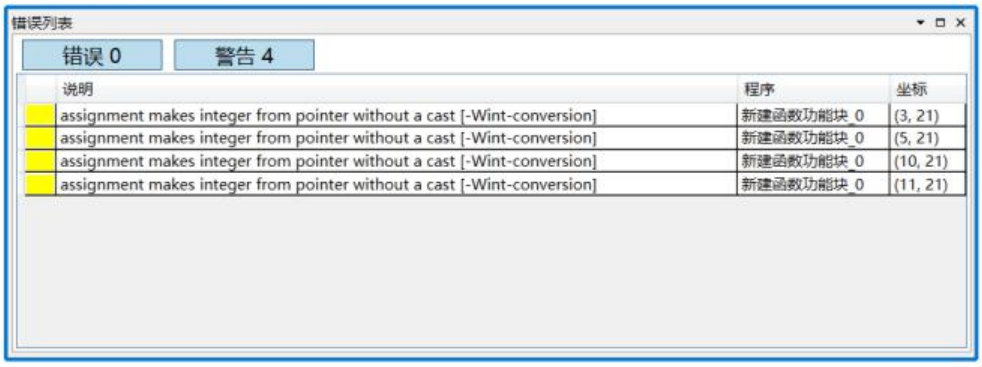


图 1.6 警告信息

注意，有时候编译通过，并且警告也不存在，但是还有可能出现其他常见的问题。所以设计使用函数功能块时需仔细慎重，以免造成不必要的麻烦和损失。

```
int     i, j;
for     ( i=0 ; i < n ; i++ )

        for ( j=0 ; j<m ;i++ )    !
```

图 1.7 变量名称错误

5.3.2 能被 CALLM 指令调用的函数的条件

函数块中的函数即使能通过编译，并不代表能被梯形图指令 CALLM 来调用。实际上，能被 CALLM 调用的函数需符合严格的约束条件。

1. 参数不能大于 4 个。例如 func1(WORD i, WORD j, WORD k, WORD l, WORD m)参数为 5 个，不符合条件。
2. 参数必须为 SAMSOARII 所定制的全大写的类型。实际上，这些类型均为基本类型的指针。

名称	对应类型	含义
BIT	uint32_t*	位寄存器(M/Y/S)的指针
WORD	int16_t*	带符号整数(D/TV/CV/AO)的指针
UWORD	uint16_t*	不带符号整数的指针

DWORD	int32_t*	带符号双整数的指针
UDWORD	uint32_t*	不带符号双整数的指针
FLOAT	float*	浮点数的指针

表 1.1 参数类型表

其中 BIT 对应的类型为 32 位，但是每个 32 位整数只映射了一个位寄存器的值，并不是通过位组合生成的，使用时要注意。

5.3.3 兼容一代 SAMSOAR Developer 的设计

1.之前的 SAMSOAR Developer 中每个函数功能块只包含一个可被 CALLM 调用的函数，并且必须符合 void Func0(WORD W, BIT B)的格式。这里同样也可以写成如上格式。并且 BIT B 不需要时可以去掉，变为 void Func0(WORD W)，同理 WORD W 也可以。

2.一代版本中，FW[1]代表 W[2]和 W[3]的合并并组成浮点数，DW[1]代表组合后的双整数，这一代也兼容这个模式，并且可以定义更多更自由的参数来代替，使用起来更方便。

3.一代版本_PI 代表圆周率，二代改为 PI，前面的下划线去掉。

5.3.4 通过构造块来编辑指令表

创建构造块来编写指令表

简介

在编写梯形图程序时，除了直接在梯形图上编辑以外，SamSoarII 还提供了另外一种方式来编写指令表。指令表是梯形图经过初步编译后得到的代表程序逻辑的指令队列，可以通过“工具栏”-->“指令模式” 

来显示并查看。（默认情况下，梯形图和指令表会共同显示在界面上，可以通过点击“梯形图模式”  使之弹开来单独显示指令表。）

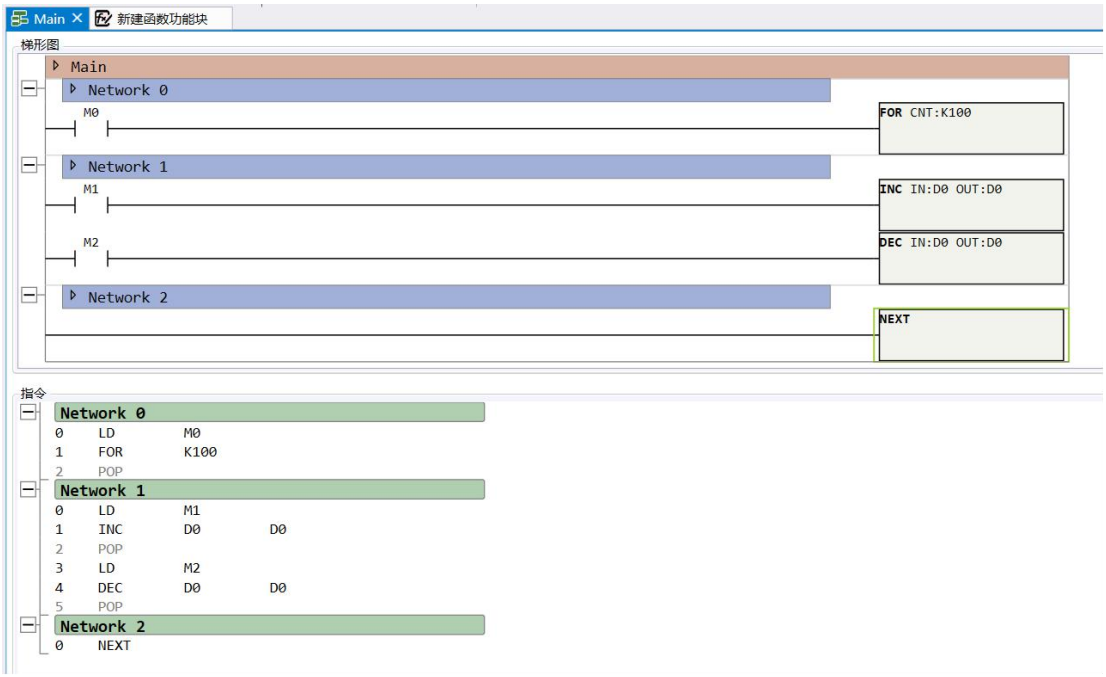


图 1.1 指令表和梯形图共存模式

点击指令表中的一条指令，梯形图会自动导航到对应位置。双击这个指令，会弹出参数的设置窗口。但是与梯形图不同的是，指令表是不能直接通过**文本输入指令**的方式来进行编辑的。如果要直接编辑指令表的话，可以用函数功能块以托管编辑的方式来实现编辑。

操作步骤

1）在“工程资源管理器”窗口中，右键点击要编辑的梯形图程序，弹出右键菜单。点击“创建构造函数”。

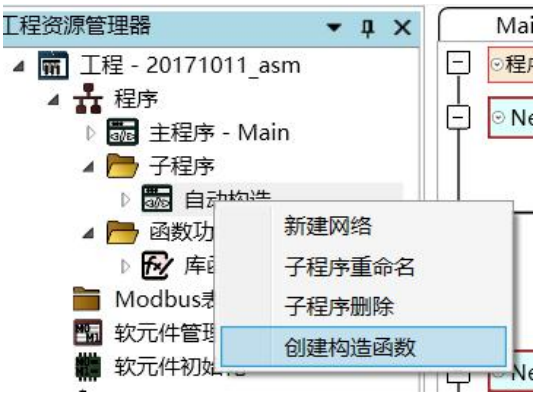


图 1.2

2）之后会弹出创建构造函数的窗口。这里的操作方式与 PLSR/PTO 参数函数的添加(查看 [标题编号])基本一致。



图 1.3

3）点击“确定”后，界面会自动导航到新添加的构造函数。这样构造函数的创建过程就完成了。（也可以直接编写以下格式的代码，上位软件会自动识别并生成对应的梯形图程序）。

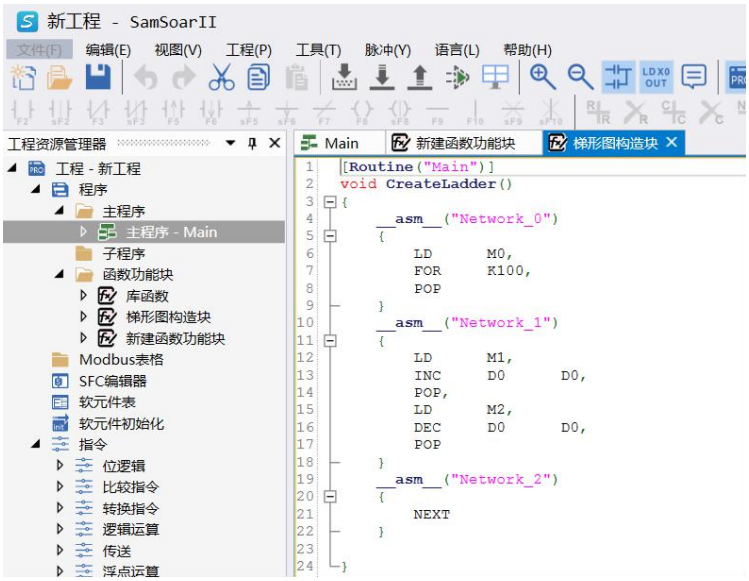


图 1.4

构造函数与一般的 C 语言函数不同，有几点自带的特性：

1. 函数头部会声明函数特性[Routine("程序名称")]，代表该函数是构造这个梯形图程序的构造函数。

2. 函数内部由__asm__结构构成，字符串参数代表要编辑的网络，内部为直接输入的指令表。可以添加这个结构来新建新的网络。

3. 指令表之间用','符号来分割，在保证分割的前提下，多条指令可以放在同一行中。**但是两个','之间不能为空，开头和结尾不允许多余的','。**

编写完毕后，通过编译（快捷键 F4）来生成梯形图，如果指令表有误，会以梯形图错误列表的形式显示出来。

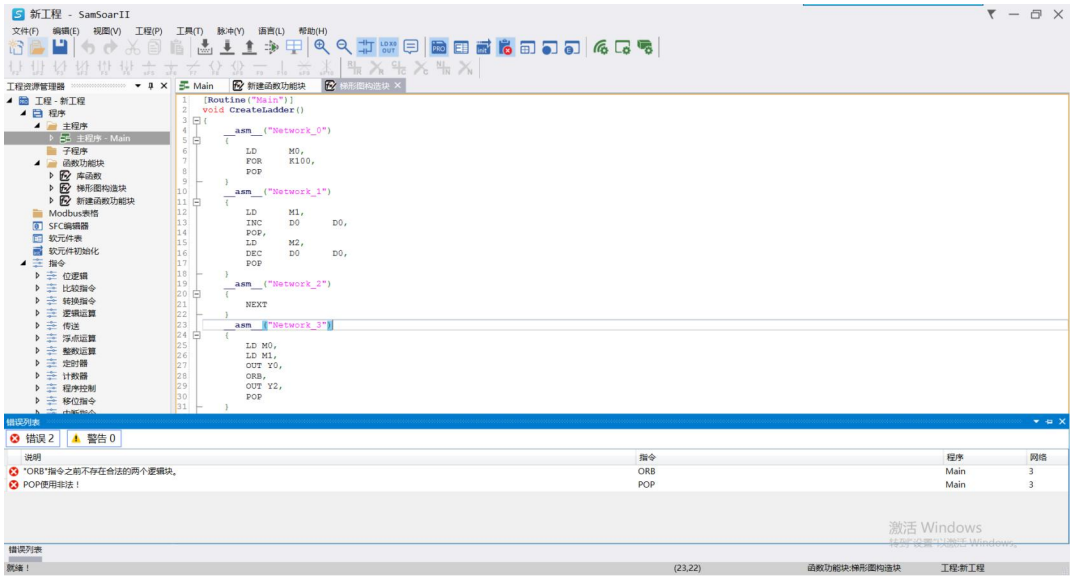


图 1.5 指令表错误

指令表合法并编译通过后，SamSoarII 会自动绘制对应的梯形图。

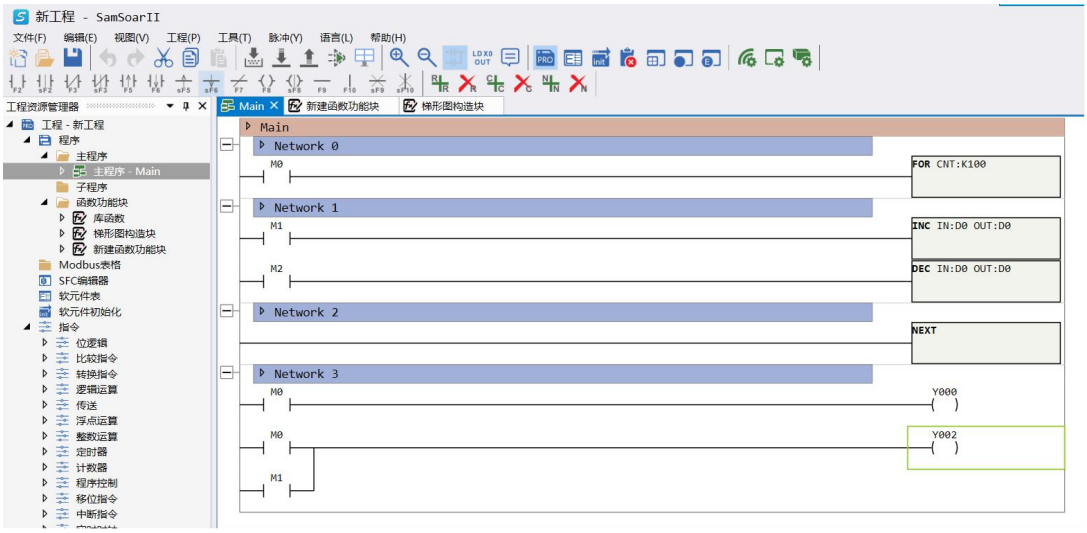


图 1.6 绘制的梯形图

注意以下几点：

1. **构造函数要严格遵守上面的程序结构，否则绘制出来的梯形图可能与之对应的指令表不符合。**
2. **用 CALLM 调用或者函数功能块中调用构造函数，并不会执行任何操作，请不要用以上两种方式来调用。**
3. **可以在函数内部和__asm__内部添加 C 语言注释，但是请注意不要注释到指令和','分割符。**

5.3.5 软元件的使用

在用户实现的函数功能块中，除了能调用函数参数和局部变量以外，还可以像梯形图一样直接使用软元件寄存器。

1. 软元件的变量

每个软元件基地址在函数块中都有一个对应的全局变量名称。比如说，基地址为 M 的软元件的全局变量为 MBit，基地址为 D 的软元件的全局变量为 DWord，并且这些变量均视为指针。M0 可以用 MBit[0]表示，D0 可以用 DWord[0]表示，等等。

以下为所有软元件的基地址和对应的变量名称。

基地址	变量名	类型
X,Y,M,S,C,T	XBit,YBit,MBit,SBit,CBit,TBit	BIT（uint32_t*）
D,CV,TV,AI,AO,V,Z	DWord,CVWord,TVWord,AIWord,AOWord,VWord,ZWord	WORD(int16_t*)
CV32	CVDoubleWord	DWORD(int32_t*)

表 1.1 软元件变量

2. 实例

对于以下梯形图程序：

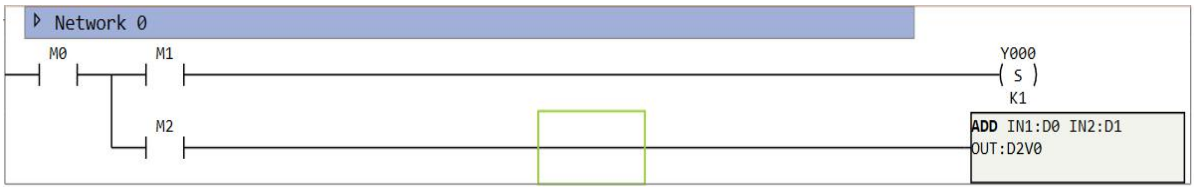


图 1.1 梯形图程序

同样也可以用函数块来实现：

```
void Func()
{
    if (MBit[0])
    {
        if (MBit[1]) YBit[0] = 1;
        if (MBit[2]) DWord[2+VWord[0]] = DWord[0] + DWord[1];
    }
}
```

图 1.2 C 语言程序

以上两个程序在逻辑上是等价的。

注意，这里只是一个示范用例，并不能代表所有的梯形图程序都能用函数功能块代替。

简化使用

可以在函数头部加入特性[Simplify("Register")]，来开启软元件输入简化的功能。

```
[Simplify("Register")]
void Func()
{
    if (M0)
    {
        if (M1) Y0 = 1;
        if (M2) D2V0 = D0 + D1;
    }
}
```

图 1.3 简化后的效果

与图 1.2 所示程序是等价的。

这个特性可以按照一定的语法规则对代码进行大幅简化，包括变值寻址，类型转换等操作都可以用简洁的语法来实现。

1. **寄存器变值访问。**例如 D0V0，如果在 C 语言代码里写成 DWord[0+VWord[0]]，若加入简化特性可以直接写作 D0Z0。

2. **寄存器特殊类型的使用。**C 语言里要经过取地址->转换地址指针->取值的复杂步骤来实现，例如若将 D0 当做双字使用，本来需要写成*((DWORD)(DWord+0))，加入了简化特性后可以写成 DD0，大大提高了编程效率。

下表为原代码和简化后的代码的对比。

简化前的代码	简化后的代码
DWord[0]	D0
DWord[0+VWord[0]]	D0V0
*((DWORD)(DWord+10))	DD10
*((FLOAT)(DWord+20))	FD20
*((UWORD)(CVWord+5))	UCV5
*((UDWORD)(DWord+20+ZWord[2]))	UDD20Z2

图 1.4 简化前后的对比

6. 软元件的使用

SAMSOARII 第二代编辑软件是在第一代编辑软件 SAMSOAR Developer 的基础上，针对显控推出的新一代编译型可编程控制器（PLC）的组态程序的图形界面编辑工具。与之前的版本相比，在保留了其原有功能的基础上，对运行性能和用户体验等方面上做了大幅度的改进。

本公司推出的 PLC 产品简单易学，应用范围广泛，其帮助系统能够帮助您掌握 SAMSOAR II 软件的基本操作与用法。

帮助文档包括：

19、 快速入门(查看 [标题编号])

20、 编辑界面

21、 指令系统

22、 寄存器

23、 仿真界面

24、 通信协议

25、 硬件手册

26、 系统设置

27、 联系方式

6.1 字取位

字软元件的位

1. 功能

指定 16 位字软元件的位，可以将其作为位软元件使用

2. 书写格式

Dm.n 表示 Dm 字寄存器的第 n 位

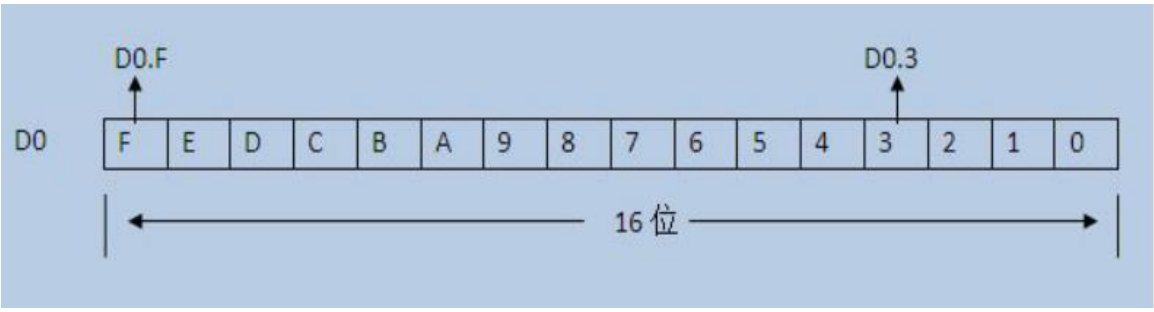
m：字软元件编号

n：位编号（16 进制 0—F）

m，n 不能使用变址修饰

D，V，Z 类型寄存器支持此方式输入。

字地址位对应图，如下图所示：



3. 实例

指令表：

```
LD      D1.1
OUT     D2.5   // D1 的第一位为 1(0)时导通(断开)，D2 的第五位被写入 1(0)。
```

梯形图：



6.2 位合成字

位软元件合成字

1. 功能

将如：X，Y，M，S 等位软元件通过组合之后作为数值处理软元件使用

2. 书写格式

以位数 Kn 和位软元件的编号来组合，使用至 16 位指令时，n 可以取（1—16），32 位指令时，n 可以取（1—32）。

Kn 与位软元件不可用变址修饰

例如：K5M3 则是由 M3—M7 组合成的一个字软元件，该软元件可应用于 16 位指令或 32 位指令中。

K20M10 则是由 M10—M29 组合成的一个字软元件，由于该软元件位数大于 16 位所以只能应用于 32 位指令中。

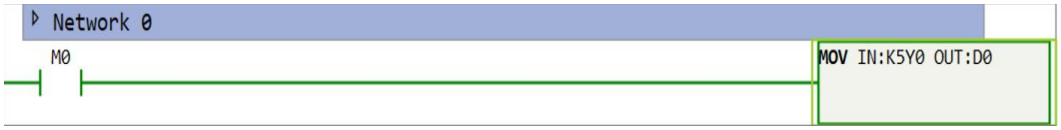
3. 实例

1）. 合成字作为读取地址

指令表：

```
LD      M0
MOV     K5Y0    D0   // 将合成字写入 16 位或 32 位数时,数据长度不足的高位都会被视为 0
```

梯形图：

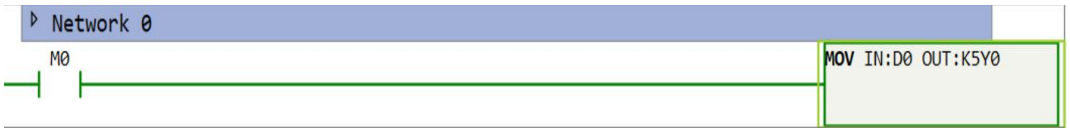


2）. 合成字作为写入地址

指令表：

```
LD      M0
MOV     D0      K5Y0   // 合成字作为写地址时可能出现位数不足，无法写入所有值的情况(如 D0 为 32767，则只能对 Y0—Y4 写入 1，多余位则被舍弃不会自动填充到之后的 Y5,6,7…)
```

梯形图：



6.3 局部变量

1. 功能

提供特定的程序内部使用的，程序外部无法访问修改的变量。

格式

局部变量类型	局部变量书写格式	初始值
位	LBxxx，xxx 部分可以是任意长度的数字/字符/下划线	OFF
字	LWxxx，xxx 部分可以是任意长度的数字/字符/下划线	0
双字	LDxxx，xxx 部分可以是任意长度的数字/字符/下划线	0
浮点	LFxxx，xxx 部分可以是任意长度的数字/字符/下划线	0.0

局部变量可以任意起名，但要保证相同的变量用的是相同的名称。**局部变量区分大小写，例如 LWa 和 LWA 视为不同的变量，这点请格外注意。**

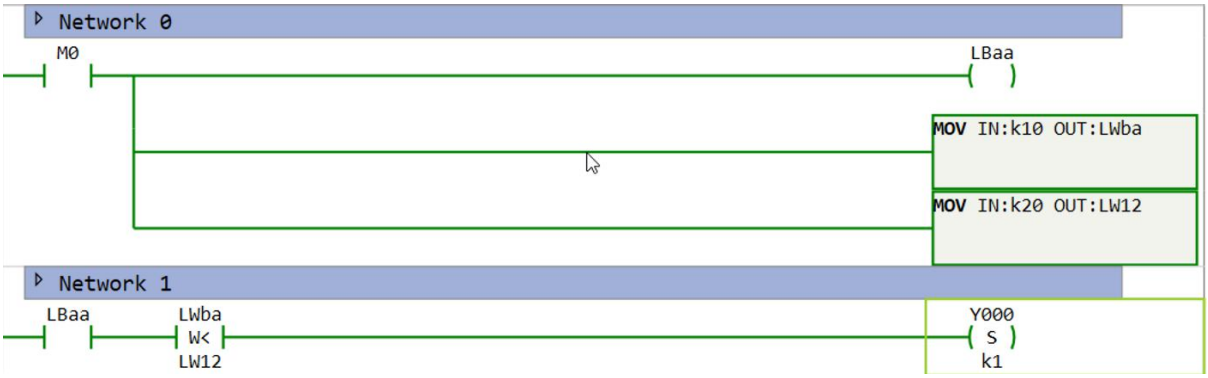
实例

指令表：

```
Network 1
    LD M0
    OUT LBaa          // 以 M0 为输入，局部变量 LBaa 作为线圈输出，相当于将 M0 的值赋给 LBaa
    MOV K10 LWba      // 当 M0 为 ON 时，对局部变量 LWba 赋值为 10
    MOV K20 LW12      // 当 M0 为 ON 时，对局部变量 LW12 赋值为 20
    POP

Network 2
    LD LBaa
    AWL LWba LW12
    SET Y0 K1     // 当 LBaa 为 ON，并且 LWba<LW12 时，将 Y0 置位。只有 M0 为 ON 时条件才会成立，否则 LBaa 的初始值为 OFF，LWba 和 LW12 的初始值均为 0.
    POP
```

梯形图：



7. 仿真界面

SamSoarII 仿真功能

与一代上位软件相比，SamSoarII 新增了强大的仿真功能。仿真可以脱离 PLC 设备，直接在电脑上模拟运行 PLC 工程程序。

1. 开始 SamSoarII 仿真
2. 仿真控制
3. 断点的使用
4. 单步调试
5. 软元件条件断点

7.1 快速开始

SamSoarII 仿真开始

保证工程程序完整合法后，就可以点击工具栏的仿真按钮，或者点击菜单栏中“工具”-->“仿真”（快捷键为 Shift+F12）来开启仿真。这时会进入仿真界面，编辑栏会消失，取而代之的是仿真控制栏。



图 1.1 仿真按钮

和开始监视时一样，会弹出监视窗口。上位软件会模拟 PLC 的运行，用户可以当做监视 PLC 来进行操作。

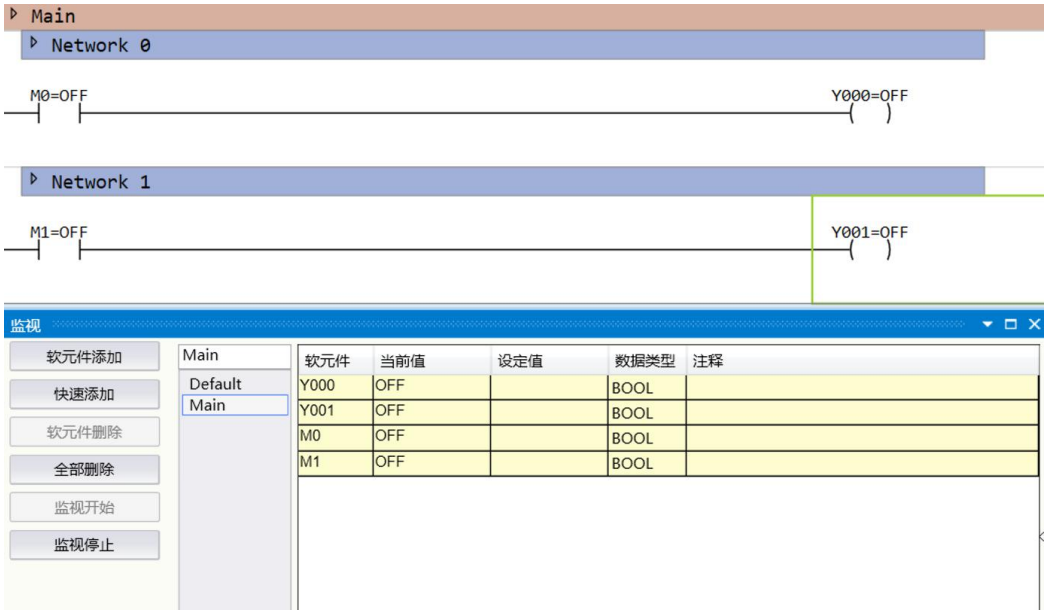


图 1.2

7.2 仿真控制

SamSoarII 仿真控制

仿真开始运行时和 PLC 的运行状态一样，是处于 RUN 状态。可以切换到 STOP 状态来停止运行，还可以切换到仿真特有的 PAUSE 状态。通过仿真控制栏最左侧的控制按钮来操控仿真的运行状态。



图 1.1 仿真控制栏

仿真处于 RUN 状态时，仿真正在运行，可以查看寄存器的当前值，并且可以修改值。



图 1.2 仿真运行

仿真处于 STOP 状态时，所有的寄存器都无法查看当前值，并且无法进行修改操作。

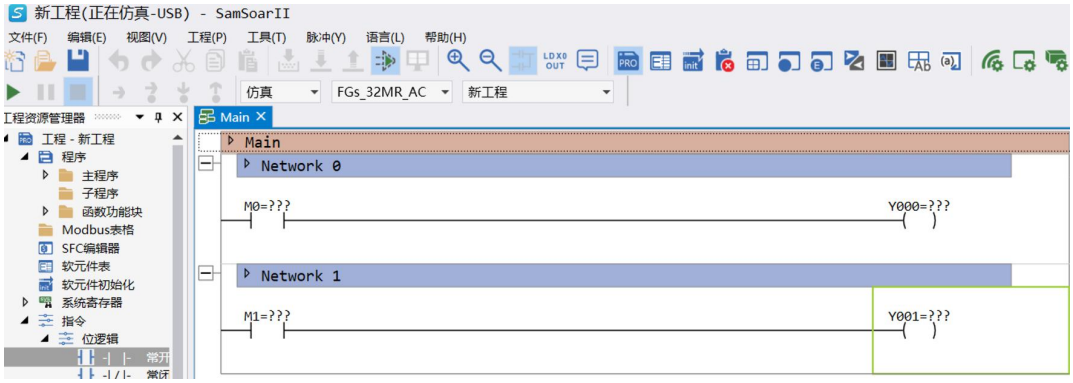


图 1.3 仿真停止

仿真处于 PAUSE 状态时，此时会停在程序运行时的某一个位置，停靠的位置用黄色光标来指示。与 STOP 不同，用户依然可以查看和修改寄存器。

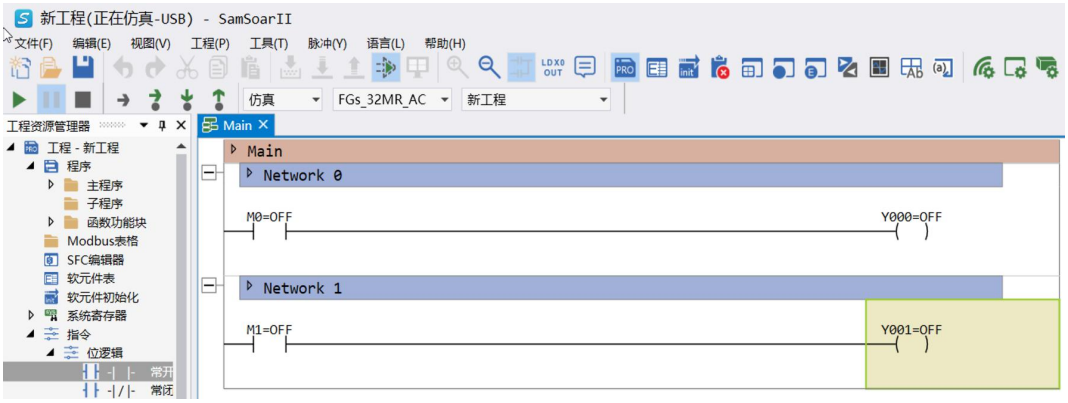


图 1.4 仿真暂停

7.3 单步调试

SamSoarII 单步调试

单步系统（Step System）是 SamSoarII 强大的仿真功能的一部分，可以一步一步跟踪程序执行的流程，将细节放大更容易找到程序的问题所在。

单步操作只有在 PAUSE 状态下才能使用，通过仿真控制栏来进行操作。



图 1.1

（1） 逐过程（快捷键 F9）

在暂停状态下执行单步操作“逐过程”后，会跳到下一条指令并继续暂停，期间只运行了一条指令。**要注意，“逐过程”会跳过子程序和函数功能块的调用。**

（2） **逐语句（快捷键 F10）**

在暂停状态下执行单步操作“逐过程”后，会跳到下一条指令并继续暂停，若要调用子程序和函数功能块时，会跳进调用的地方。

（3） **跳出（快捷键 F11）**

在调用子程序或者函数功能块时，执行单步操作“跳出”后，程序会将该子程序或者函数功能块的剩余部分运行完毕后，跳到调用位置的下一条指令并继续暂停。若在主程序执行该操作，会脱离 PAUSE 状态继续执行。

（4） **查看当前暂停位置**

界面会导航到当前暂停的位置，以便于查看。

7.4 程序断点

SamSoarII 仿真断点

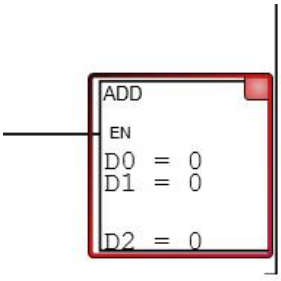
断点系统（Breakpoint System）是 SamSoarII 强大的仿真功能的一部分，可以让程序在需要的地方进行暂停（进入 PAUSE 状态），从而方便分析与调试。

1. 添加一个断点

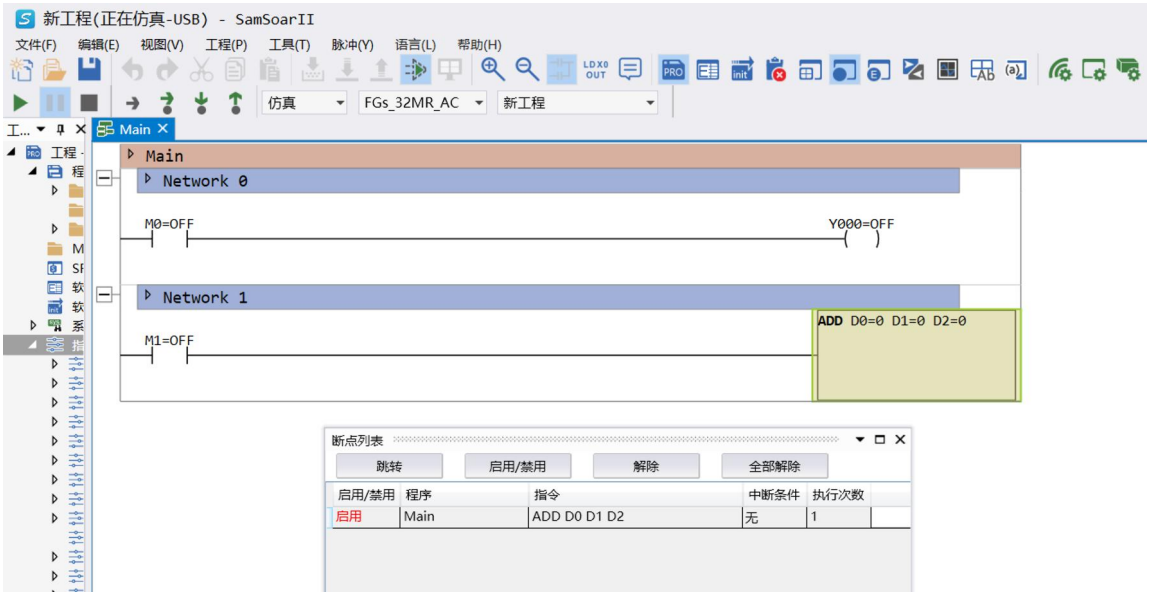
断点包括梯形图断点和函数块断点。以梯形图断点为例，可以在要添加断点的位置点击右键弹出右键菜单，点击“添加断点”按钮来完成操作。



添加断点后，梯形图相应元件的周围会有一个红色矩形图元包围，表示这里存在一个断点。



当程序运行到这个位置时，会自动停留在这里并进入 PAUSE 状态。



断点设置

在添加了一个断点后，在该断点处点击右键弹出右键菜单，点击“断点设置”按钮，会弹出断点列表的窗口。这里可以设置断点的属性或者进行其他操作。



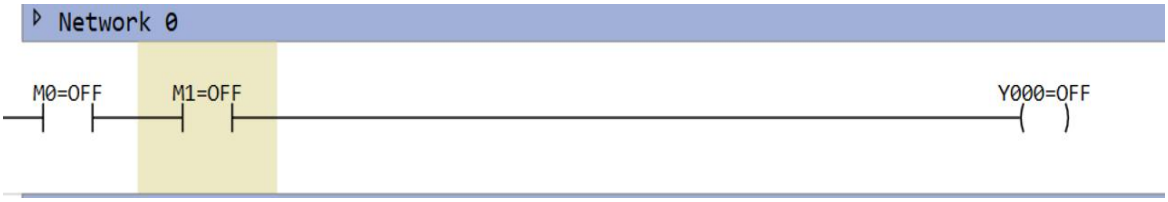
在选定列表中的断点后，点击 跳转 程序会运行到断点处并暂停，点击 启用/禁用 断点会在启用和禁用状态中切换，点击 解除 会立即删除掉这个断点，点击 全部解除 会删除掉所有的断点。

2.1 设置中断条件

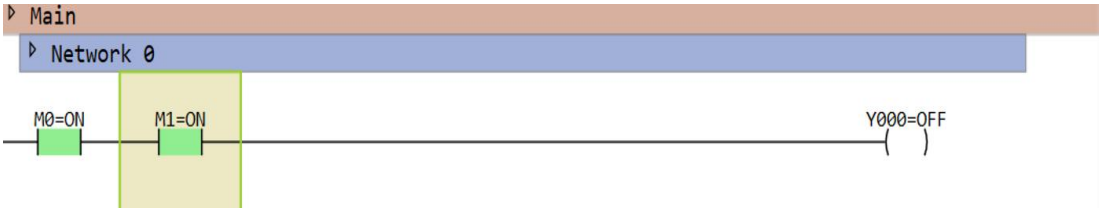
断点设置中断条件后，这个断点在符合这个条件下才会触发中断。断点的中断条件分为五种，分别为 ON，OFF，上升沿，下降沿和边沿。

（1）中断条件“ON”

中断条件设置 ON 后，当前断点只有在输出 ON 时才会触发。以下为设置中断条件 ON 的例子。



这里设置了中断条件后，断点图元的右上方会设置标记，代表这个中断条件的类型。这里 ON 用 1 来表示。设置了条件后，若条件不合法就不会触发断点中断，在这里需要两条指令都导通的情况下才会中断。



注意，触发了断点中断后，程序就会停留在这个位置等待用户的下一个操作，后面的指令并没有执行，所以 Y0 没有被导通。

（2）中断条件“OFF”

与 ON 相反，OFF 只有在输入 OFF 时才会触发。

（3）中断条件“上升沿”

在当前断点的输出从 0 到 1 变化时才会触发。

（4）中断条件“下降沿”

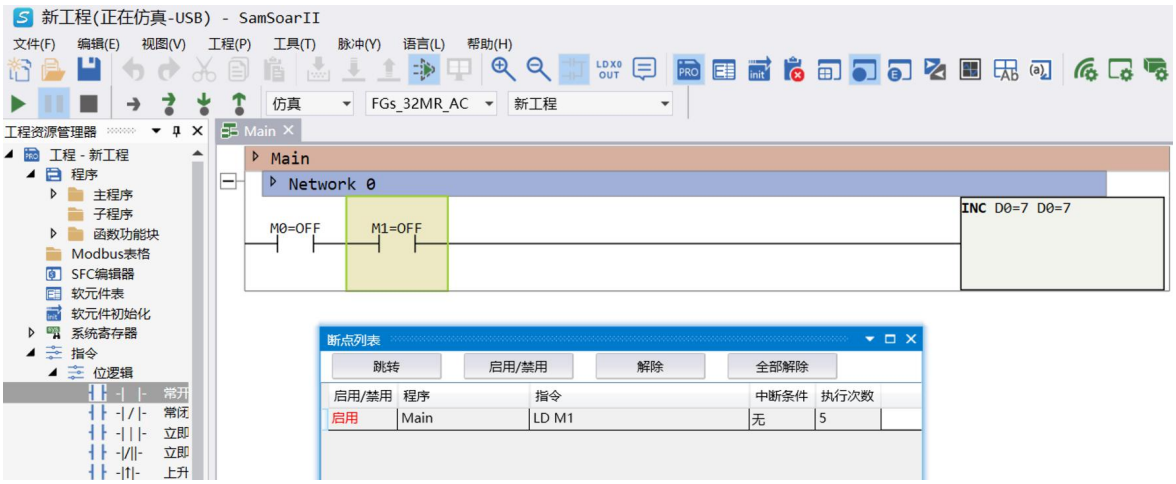
在当前断点的输出从 1 到 0 变化时才会触发。

（5）中断条件“边沿”

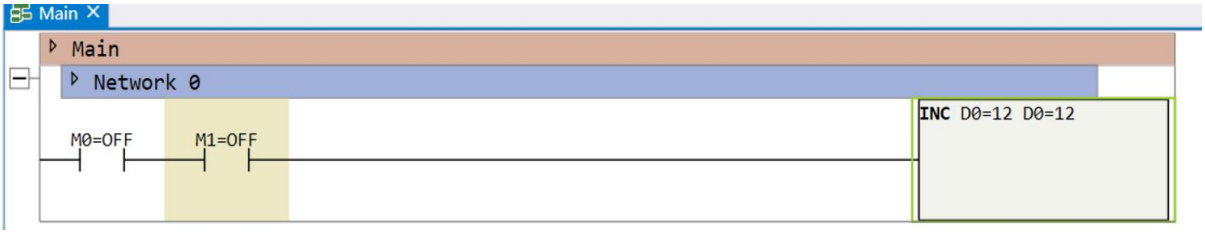
在当前断点的输出从 0 到 1 变化或者从 1 到 0 变化时才会触发，是以上两个条件的或集。

2.2 设置执行次数

设置了执行次数后，断点本来要被触发的情况也可能不会被触发，要累积一定次数后才会触发，触发后累积次数清 0。这里有一个例子。



断点在这里触发之后，当前 D0 的值为 7，由于要累计执行 5 次之后再触发这个断点，所以在此期间 D0 也会被累积增加 5 次。



断点删除

可以通过断点列表窗口中的解除按钮，或者右键菜单中的“删除断点”，来删除这个断点。

7.5 软元件断点

软元件系统（Register Breakpoint System）是 SamSoarII 强大的仿真功能的一部分，和一般断点设置中断条件类似，满足了一定的条件后才会触发，不过软元件断点可以在满足条件下任意位置触发。

设置软元件断点

点击菜单栏“视图”-->“软元件断点”，打开软元件断点列表来进行编辑。



1.1 添加软元件断点

点击 条件添加 按钮，可以新建一个软元件断点。



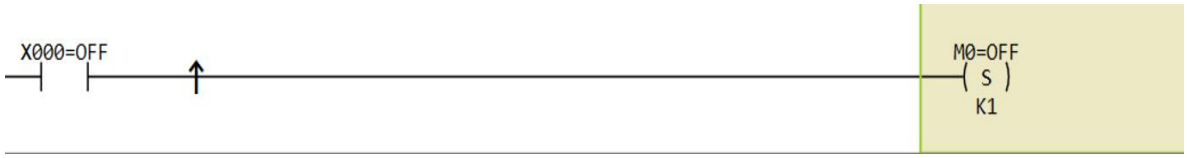
一开始表项为空，必然是非法的。可以在里面填写以实现各样功能。以下为软元件断点的具体用例。

1. 当 M0 为 ON 时断点暂停。

可以这样填写：



进行仿真时，满足条件会触发断点并进入暂停状态。



2. 当 M0 发生变化时断点暂停。

可以这样填写：



3. 当单字 D0 大于单字 D1 时断点暂停。

可以这样填写：



4. 当浮点 D0 小于浮点 D1 时断点暂停。

可以这样填写：



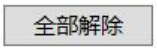
2.2 启用/禁用软元件断点

通过 

按钮来对下表选中的断点进行启用/禁用的切换。

2.3 删除软元件断点

通过 

按钮来删除选中的软元件断点，通过 

按钮来对删除所有的软元件断点。

8. SFC 编程

与一代上位软件相比，SamSoarII 支持了 SFC（步进梯形图）的编程。以下几个章节主要介绍 SamSoarII 的 SFC 编辑功能。

1. 介绍 SamSoarII 的 SFC 模块
2. 在编辑模式下对 SFC 进行编辑
3. SFC 的检查和编译
4. 在查看模式下通过 SFC 元件来导航

8.1 SFC 简介

循序功能图（Sequential Function Chart，SFC）是 IEC 61131-3 所规范的 PLC 编程语言之一，其最大的特色在于以类似流程图的观念来规划程序，适用于着重流程控制及状态转移的控制程序。

SamSoarII 的 SFC 大幅度简化了传统的 SFC 结构，保留了基本的逻辑框架，使得更为清晰易懂和操作简洁。

SamSoarII 的 SFC 程序结构

整个 SFC 程序主要由状态节点和转移元件交替构成，状态节点包括普通状态节点和跳转节点，而转移元件分为一对一的单次转移，一对多的分支转移以及多对一的合并转移。如下为一个 SamSoarII 的 SFC 程序案例，最顶端为状态 S0 的状态节点，下面依次为单次转移，S1 状态节点，单次转移，S2 跳转节点，保证状态和转移交替进行。默认情况下，所有最顶行的状态节点被设为起始点。

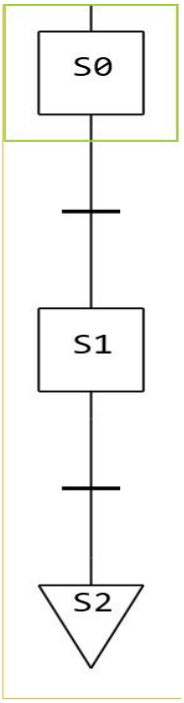


图 1.1 一个简单的 SFC 程序

2. SamSoarII 的 SFC 运行原理

SamSoarII 的 SFC 并不直接参与程序的运行，而是先通过编译生成 STL 顺序控制梯形图，然后编辑 STL 中的状态内的动作和状态之间的转移条件，最后才能生成一个完整的程序。所以，SFC 在这里的定位是辅助生成 STL 程序的一个功能模块。以上面提到的 SFC 程序为例，初步转换后会生成以下的 STL 程序。基本逻辑保持一致，状态转移的条件用位置参数的 LD 指令来替代，来代表待编辑。

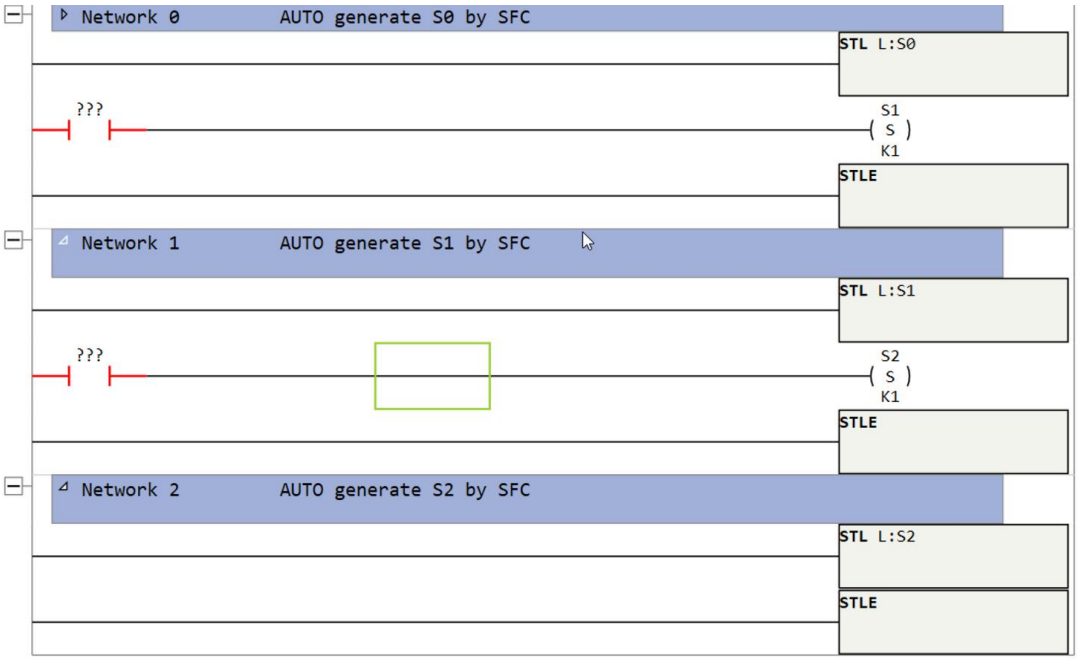


图 1.2 转化后的 STL 程序

上步操作完成后，可以在梯形图下编辑完成 STL 程序。一般情况下，在 STL 指令下方来实现状态内的动作，将 LD ???指令替换为状态之间的转移条件。以下为“在状态 S0 下输出 Y0，并等待一秒后转移到 S1”的具体案例。

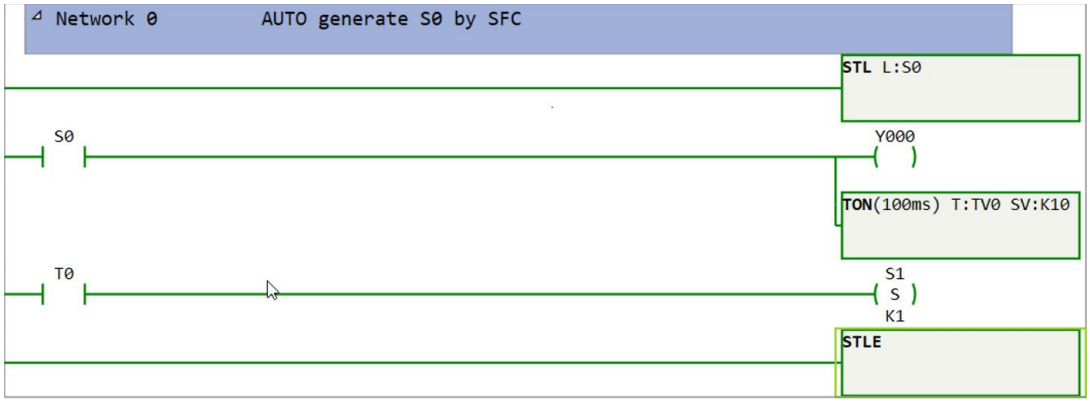


图 1.3 状态 S0 的动作和转移

SamSoarII 的 SFC 元件

上面提到。SFC 元件主要分为状态节点元件和转移元件，而状态节点又分为以下几类。

（1）普通状态节点

代表一个状态，到达这个状态后可以继续转移，所以下面可以和转移元件相连。

（2）跳转节点

同样也代表一个状态，但与普通状态节点不同，是以跳转的概念到达这个状态的，所以下面不能连接转移元件。

转移元件的种类更多，分为以下几类。

（1）单次转移

只带一个转移条件的一对一转移，当在上一个状态下符合转移条件后，会转移到下一个状态，上一个状态会被关闭。

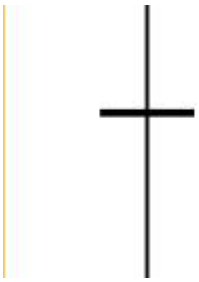


图 1.4 单次转移

其对应的梯形图为：



图 1.5

（2）并行分支

在 SFC 程序中可以有并行处理的程序，这样的结构我们便称之为分支，而分歧结构依执行的机制可分为并行分支与选择分支。

下图为并行分支的图例，其中分支的步进点会刻意用双线来进行连结；而并行分支的特征便是所有分支的步进点会共享同一个转换点，当该转换点的条件成立之后，下方的所有步进点就会同时被启动，亦即所有的平行路径都会执行。

如下图，当转换点成立后，S0 便会关闭，而 S1 及 S2 则会一起启动。

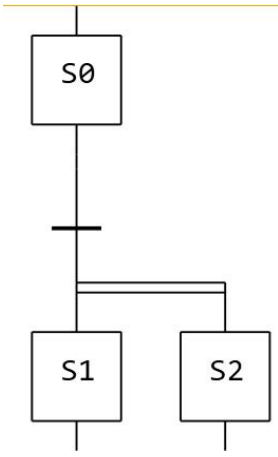


图 1.6 并行分支

其对应的梯形图为：

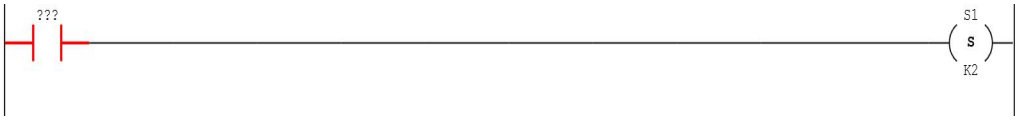


图 1.7

（3）选择分支

下图为选择分歧的图例，其特征为所有分歧的步进点都会有各自的转换点，当其中某一个转换点的条件成立之后，执行权就会移转至该转换点下方的步进点，而其他的转换点则不再继续判断，因此在选择分歧的平行路径中，同时间只有一条路径的步进点程序会被执行。此外，如果有多个转换点条件同时成立时，执行权会同时转向这多个转换点。

如下图，当转换点 1 成立后，S0 便会关闭，而 S1 则会启动。同理转换点 2 成立后，S0 便会关闭，而 S2 则会启动。当转换点 1 和转换点 2 同时成立，S0 便会关闭，而 S1 及 S2 则会一起启动。

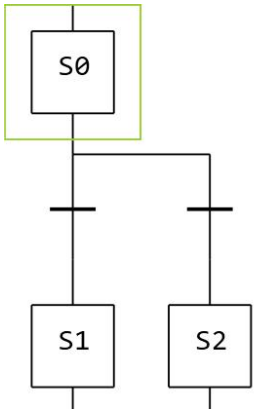


图 1.8 选择分支

其对应的梯形图为：

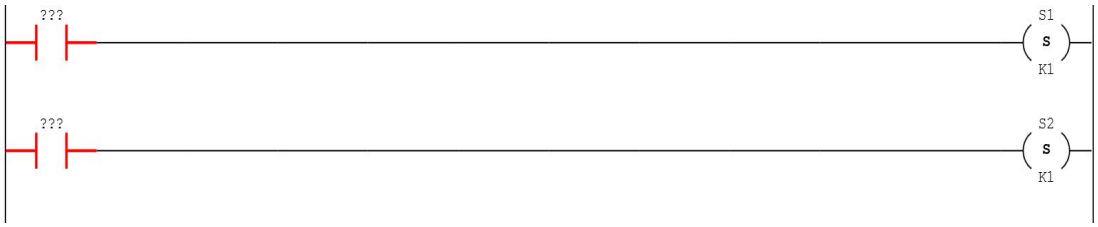


图 1.9

（4）选择并行分支

下图为选择并行分支的图例，选择并行分支实际上是选择分支和并行分支的结合，选择分支的每个转换点后面都套了一个并行分支。当其中某一个转换点的条件成立之后，执行权就会移转至下方并行分支的所有步进点。

如下图，当转换点 1 成立后，S0 便会关闭，而 S1,S2 则会启动。同理转换点 2 成立后，S0 便会关闭，而 S3,S5 则会启动。当转换点 1 和转换点 2 同时成立，S0 便会关闭，下面的所有状态全部同时启动。

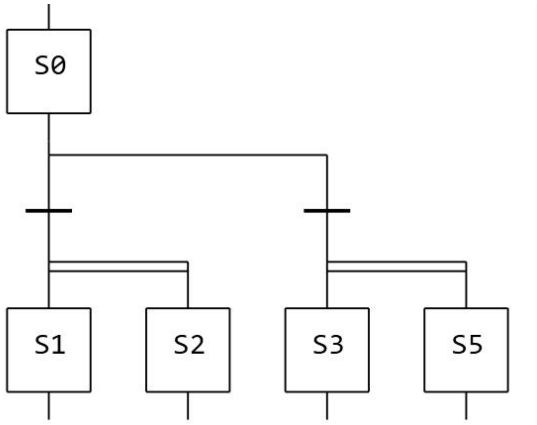


图 1.10 选择并行分支

其对应的梯形图为：

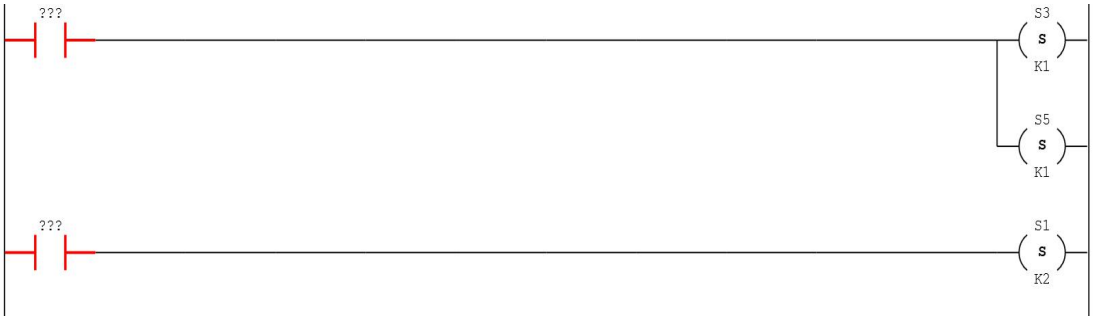


图 1.11

（5）并行合并

在 SFC 的分支结构中，最后仍须将所有分支的路径再重新汇整在一起，而这样的汇整程序便称之为合并；如同分支结构一般，合并依执行的机制亦可分为并行合并与选择合并。

在图示中，并行合并的步进点之间亦刻意用双线来连结，而其特征则为所有分歧的步进点都会联结至同一个转换点，因此必须等待连接至该转换点上方的所有步进点都被启动后，转换点才可进行判断。如下图的范例，须先等待 S1 及 S2 都被启动后，转换点才会开始判断，而当转换点的条件成立时，S1 及 S2 便会一起被关闭，并启动 S3。

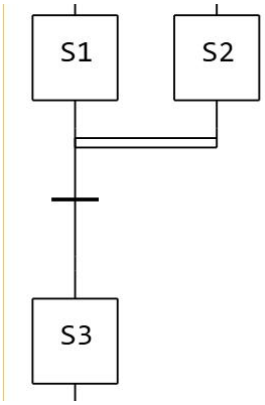


图 1.12 并行合并

其对应的梯形图为：

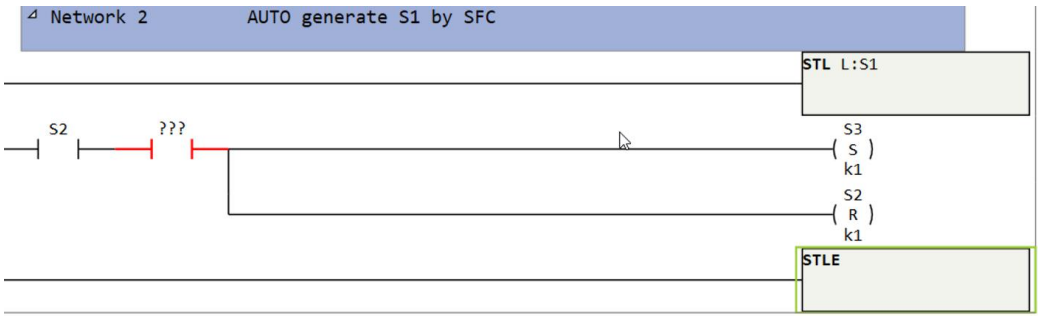


图 1.13

（6）选择合并

下图为选择合并的图例，其特征为所有合流的步进点都会有各自的转换点，而只要其中某一个转换点的条件成立之后，执行权就会先行移转至下一个的步进点；但若平行路径中仍有其他被启动的步进点时，则所有被启动的步进点仍会继续执行，一直到其合流的转换点成立之后，便会再次启动合流下方的步进点，因此选择合流有可能造成合流点下方的循序流程同时有多个步进点被启动，使用上请务必确认是否会造成程序上的错乱。

如下图，当转换点 1 成立时，S1 便会关闭并直接将 S3 启动，而后续当转换点 2 的条件也成立时，S2 便一样会被关闭，且不论 S3 的状态为何，系统仍会再次启动 S3。转换点 2 先成立，转换点 1 后成立的情况也一样。

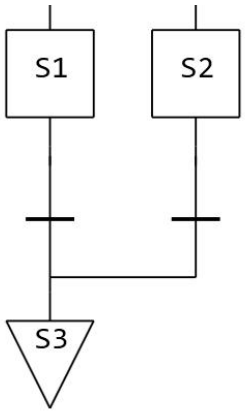


图 1.14 选择合并

其对应的梯形图为：

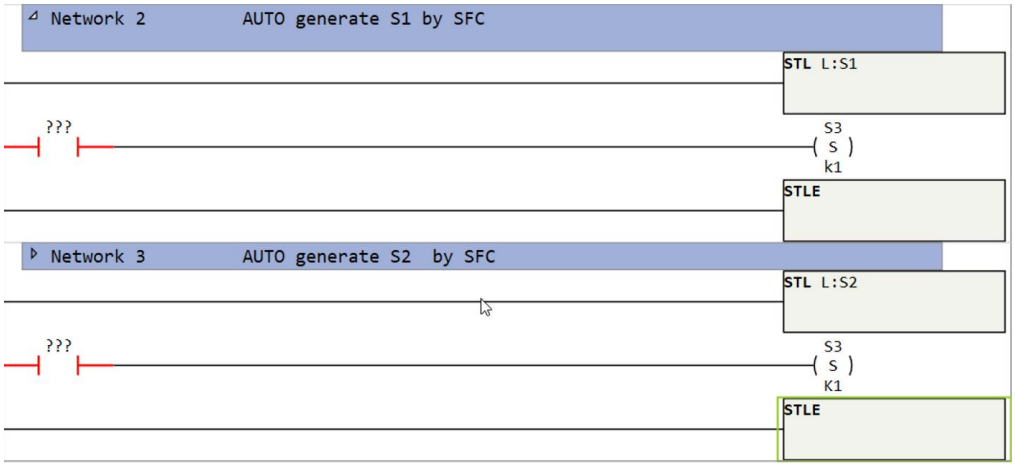


图 1.15

（7）选择并行合并

下图为选择并行合并的图例，其特征是合并后的步进点前面为多个转换点，每个转换点前面又有多个并行连接到上一行的步进点。当转换点前面连接的步进点均在启动状态，并且转换条件成立后，执行权就会先行移转至下一个的步进点，同时前面并行连接的多个步进点均被关闭。

如下图，当 S1 和 S2 均为启动状态，并且转换点 1 成立时，S1,S2 便会关闭并直接将 S5 启动。同理，当 S3 和 S4 均为启动状态，并且转换点 2 成立时，S3,S4 便会关闭并直接将 S5 启动。所以 S5 可能因为上面两个转换点而被启动两次，这两次启动只和对应的转换点有关，之间是不相关的。

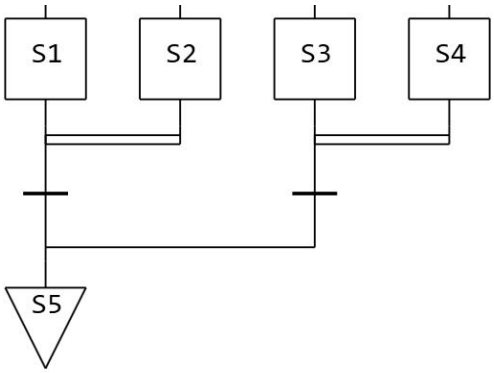
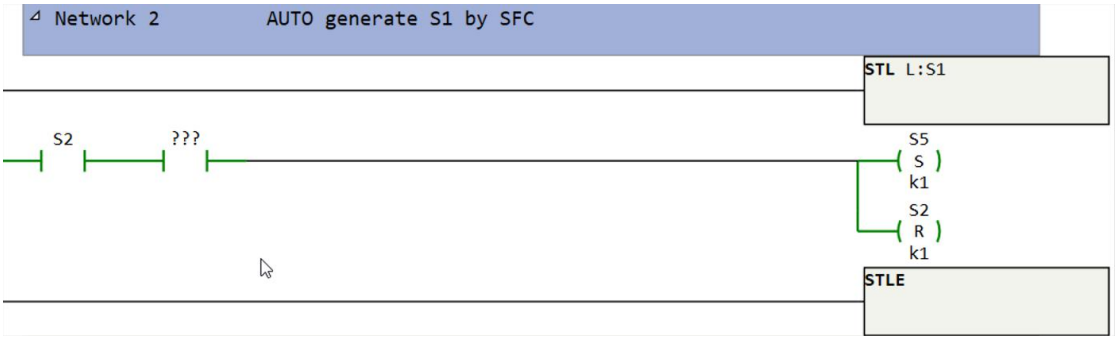
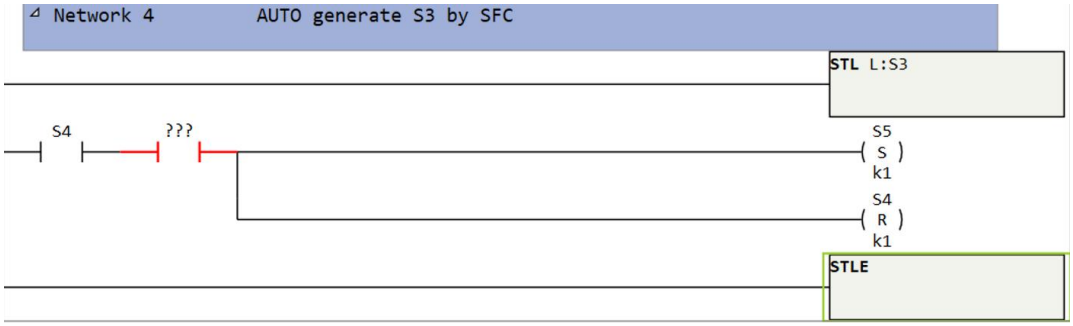


图 1.16 选择并行合并

其对应的梯形图为：


Network 2 AUTO generate S1 by SFC
STL L:S1
S2
???
S5
S
k1
S2
R
k1
STLE


Network 4 AUTO generate S3 by SFC
STL L:S3
S4
???
S5
S
k1
S4
R
k1
STLE

8.2 SFC 编辑模式

1. 使用 SFC 编辑器创建 SFC 梯形图

双击“工程资源管理器”-->“SFC 编辑器”，即可打开 SFC 编辑器。

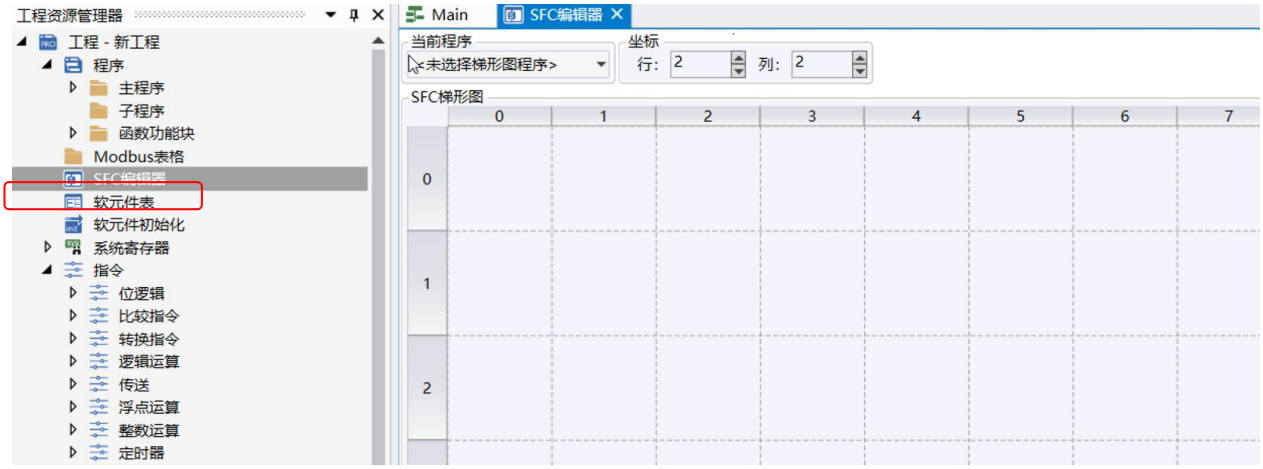


图 1.1 SFC 编辑器

一开始未选中对应的梯形图程序，所以 SFC 编辑界面是空的。可以在“当前程序”里选择要编辑的梯形图程序，如果这个程序之前未编辑过 SFC，会生成一个简单的 SFC 样例程序。

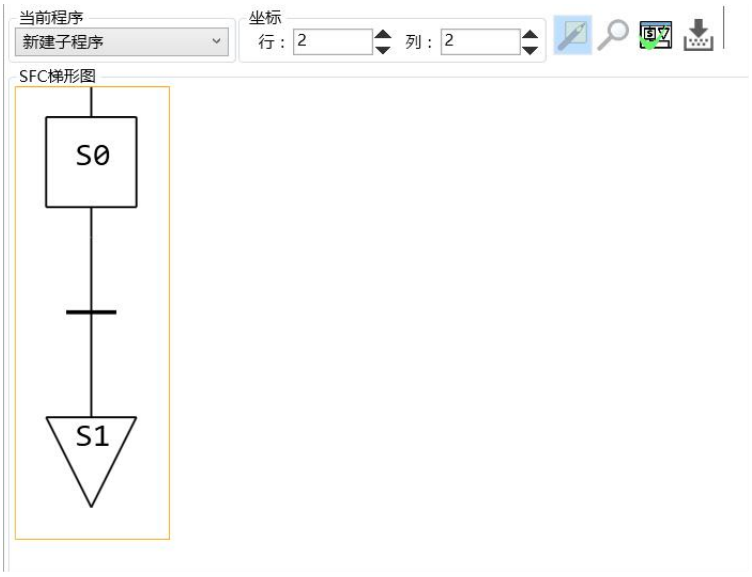


图 1.2 SFC 样例程序

2. SFC 基本编辑操作

SamSoarII 支持一系列对 SFC 程序的修改操作，包括对元件的修改替换，对行和列的添加删除等。

（1）元件添加

通过双击 SFC 梯形图中要添加元件的位置，或者绿色光标移动到该位置并按下回车键，会出现元件编辑对话框。

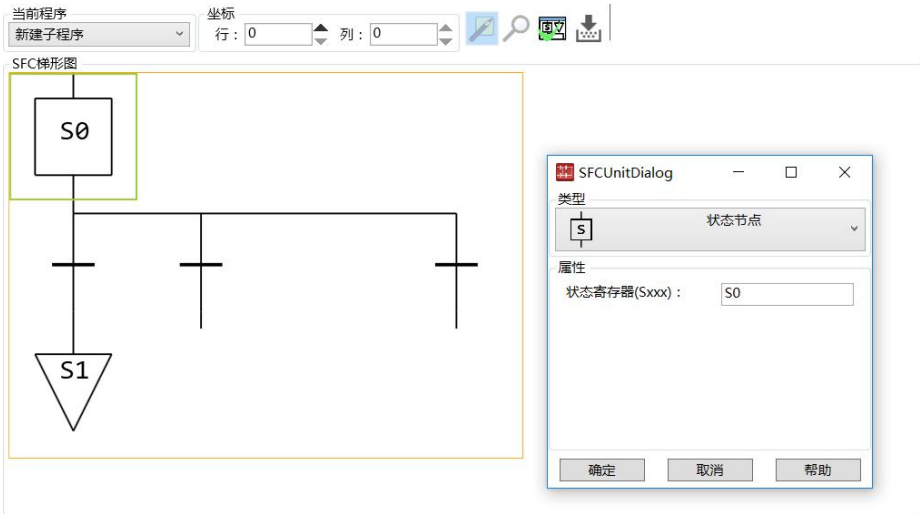


图 1.3 编辑 SFC 元件

对话框的顶部可以选择元件的类型，不同的行能够选择的元件类型也不相同。一般来说，状态类的元件（状态节点，跳转节点）只能在偶数行（0,2,4,...,2n）的位置添加，转移类的元件（单次转移，分支，合并）只能在奇数行（1,3,5,...,2n+1)的位置添加，竖线连线可以在任意位置添加，这样做确保状态和转移之间能够合理的交替放置。

添加状态类的元件，需要输入对应的 S 类寄存器的名称，来制定这个节点对应的状态。

添加竖线和单次转移时，不用输入其他额外的东西，直接点击“确定”按钮即可添加。

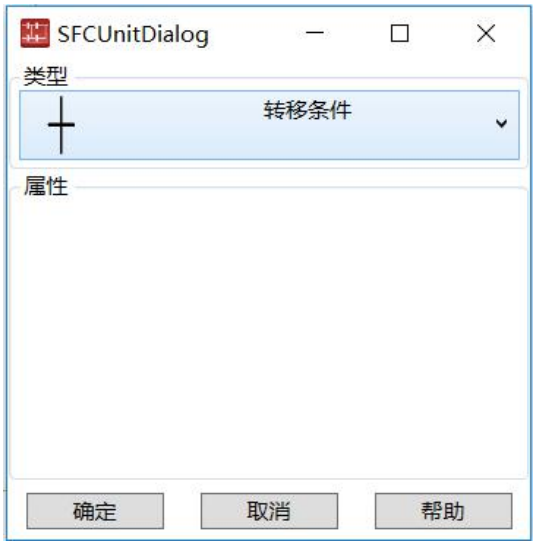


图 1.4

添加并行分支，选择分支，并行合并和选择合并时，需要在属性区域出现的文本框内输入一定规则的参数。如果这些元件添加到 SFC 梯形图中，需要制定每个转移点的列坐标，坐标用绝对位置第几列来表示（第 0 列用 0 表示，第 1 列用 1 表示，...，第 N 列用 N 表示），坐标之间用逗号’,’来分割区分。比如说，要添加一个转移点分别在第 0 列，第 1 列和第 3 列的条件分支，需要在属性里输入“0,1,3”，然后点击“确定”，就可以将对应元件添加进去了。

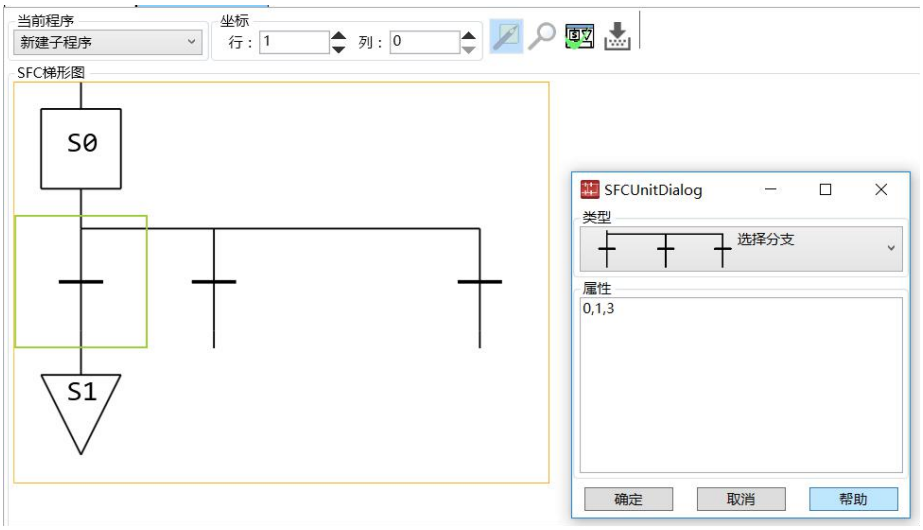


图 1.5

添加选择并行分支，选择并行合并时，默认规定每个选择转移点需要和对应的第一个并行转移点的坐标一致，只需要输入所有并行转移点的列坐标即可，每个**并行**转移点的坐标值用**’,’**分隔，不同的**选择**转移点之间用**’;’**来分隔。比方说,在选择并行合并元件的属性框内输入“1,2;3,4,6”并点击确定，会添加列的起始位置为 1，并行转移点坐标和所设置的一致的选择并行合并元件。

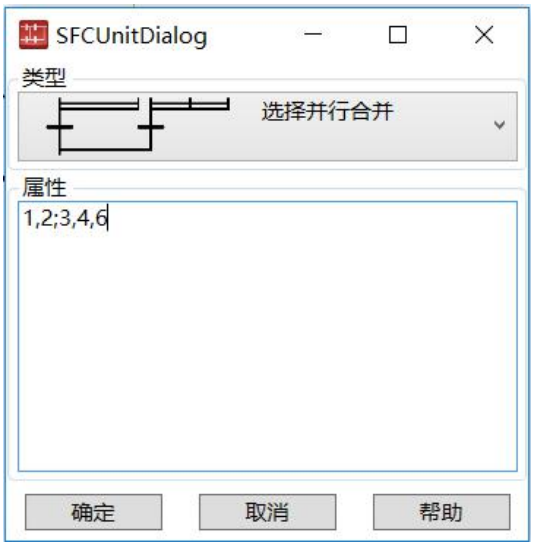


图 1.6 添加前

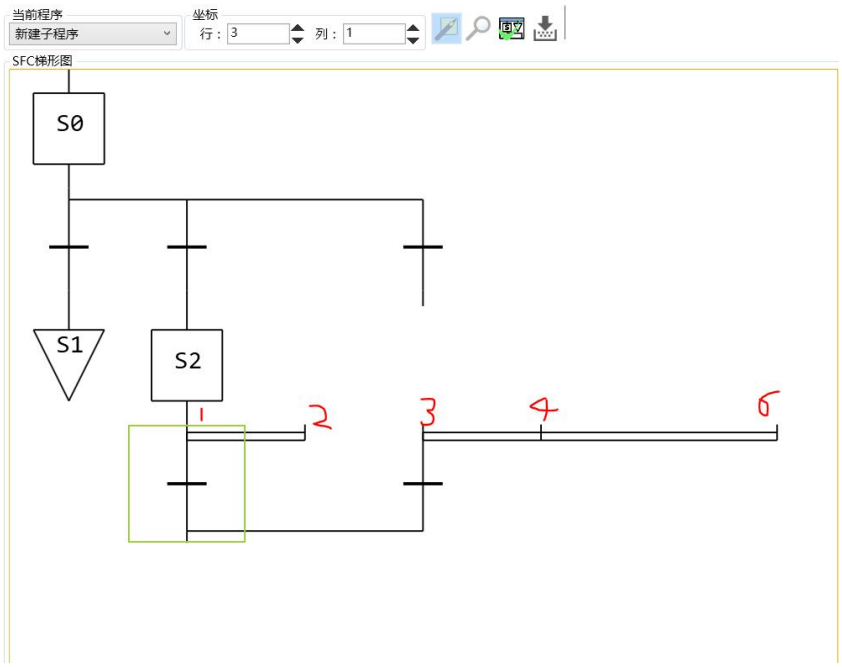


图 1.7 添加后

（2）行添加

通过右键菜单中的行添加，或者工具栏中的按钮，或者快捷键 Ctrl+Down，来添加一个新行。默认的行添加是在当前选择范围的下一行添加新行，右键菜单中可以选择添加行的位置（向前添加，向后添加，末尾添加）。

执行了行添加操作后，**会强制添加两行**，保证添加的行前后的状态类元件和转移类元件的交替性。如果添加的行前面有连线相连，会在新添加的行中添加对应的竖线。

（3）行删除

通过右键菜单中的行删除，或者工具栏中的按钮，或者快捷键 Ctrl+Shift+R，来删除选择范围内的行。删除的行后面的部分会向上移动偶数步，以保证交替性。

（4）列添加

通过右键菜单中的行添加，或者工具栏中的按钮，或者快捷键 Ctrl+Right，来添加一个新列。默认的列添加是在当前选择范围的下一列添加新列，右键菜单中可以选择添加列的位置（向前添加，向后添加，末尾添加）。

执行了列添加之后，所有的分支和合并元件可能会包含添加的列，其每个转移点的位置也会做相应的调整。

（5）列删除

通过右键菜单中的列删除，或者工具栏中的按钮，或者快捷键 Ctrl+Shift+C，来删除选择范围内的列。删除的列后面的部分会向左移动。

8.3 SFC 检查编译

对 SFC 程序进行检查

SFC 程序完成之后，可以点击 SFC 编辑器顶部工具栏的  按钮，来对 SFC 进行检查操作。检查出来的错误会与梯形图错误相同的方式来显示出来。

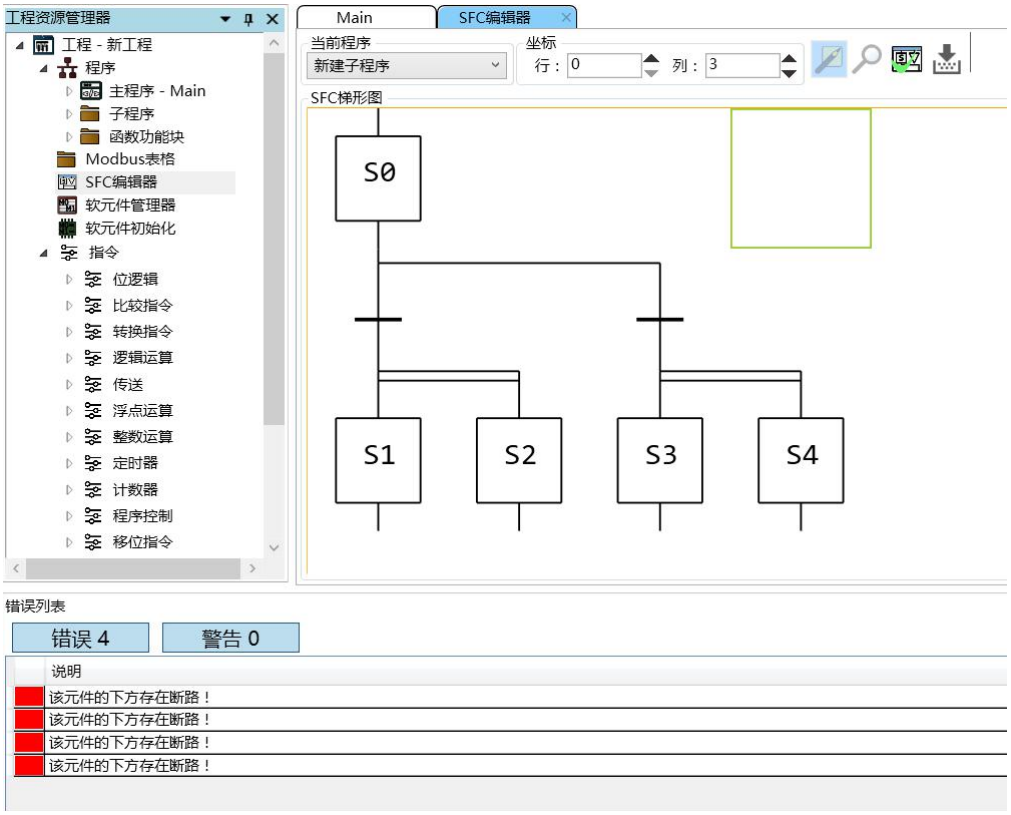


图 1.1

修改完所有错误后，再进行检查，会提示检查成功，并对 SFC 梯形图进行重新排版。

编译 SFC 程序

可以点击 SFC 编辑器顶部工具栏的  按钮，来对 SFC 进行编辑并在梯形图程序中生成对应的 STL 结构。以下面的 SFC 程序为例。

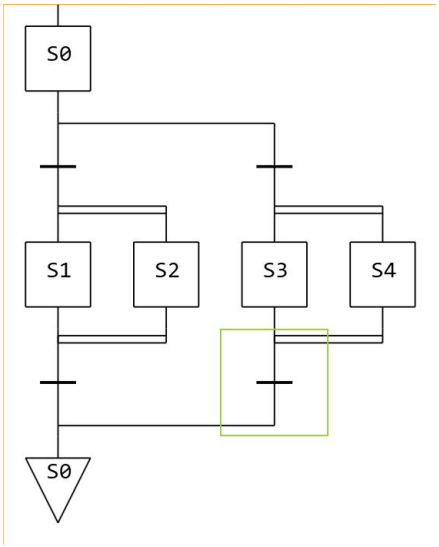


图 1.2

编译成功之后，会在对应的梯形图程序生成对应的 STL 程序。

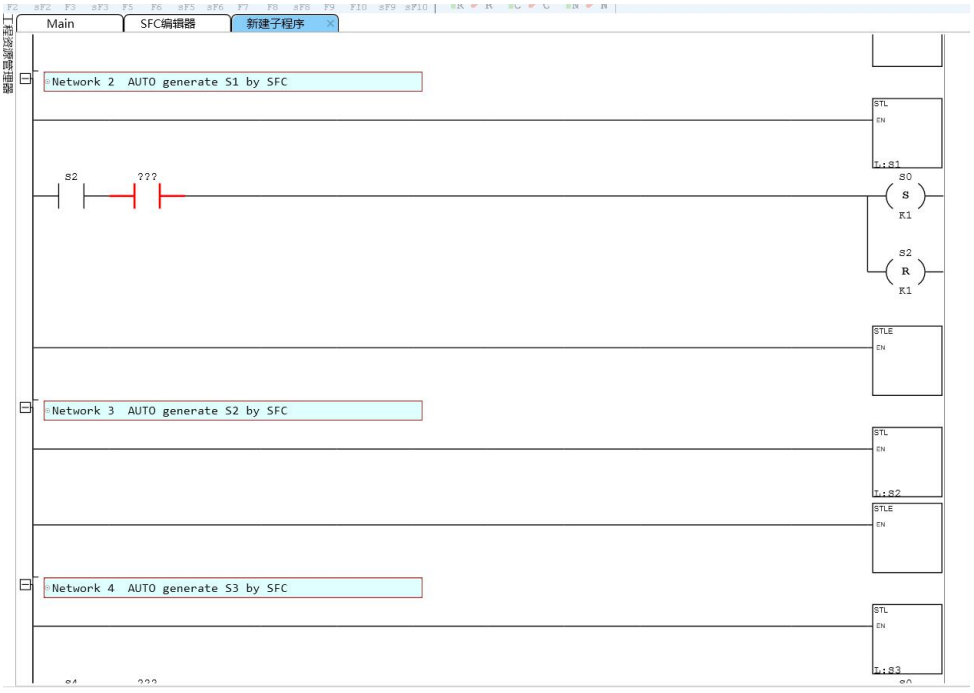


图 1.3

编译成功后，可以切换到查看模式，通过从 SFC 导航到梯形图的形式，来完成整个程序。

要注意，SFC 编辑器只能在梯形图中添加 STL 结构，没有权限来删除梯形图的部分。所以若要连续编译，请手动删除之前编译生成的 STL 结构，确保生成的程序的正确性。

8.4 SFC 查看模式

对 SFC 的编译成功后，工具栏中的  会被选中，表示可以进入查看模式了。点击这个按钮来进入查看模式。

在查看模式下，双击 SFC 元件，界面就会自动导航到对应的梯形图位置。以上面的 SFC 程序为例，双击行 2 的第一个条件。

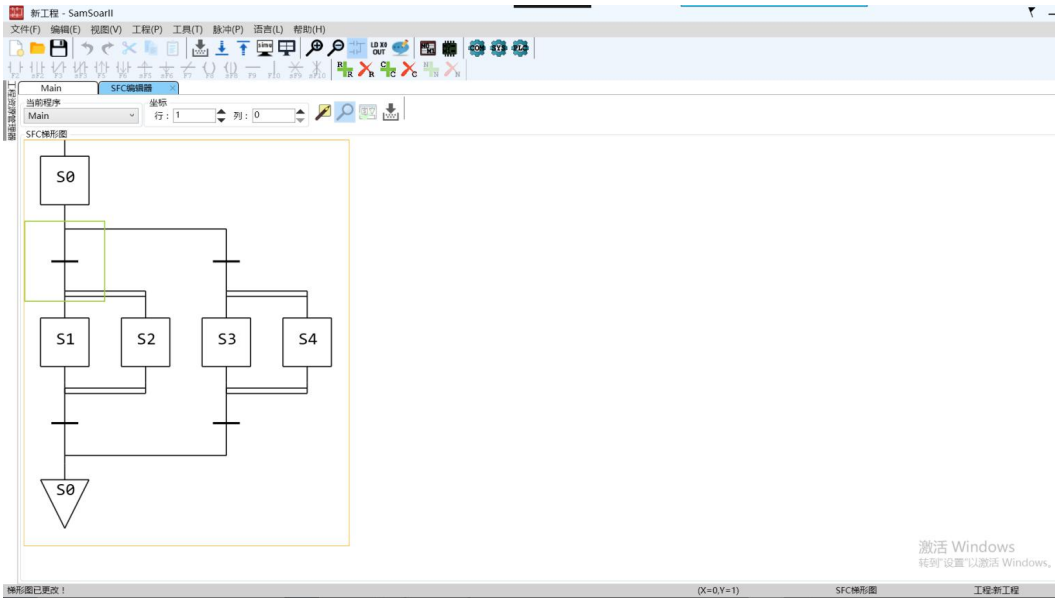


图 1.1

双击过后，主界面会自动跳转到对应的梯形图的位置。

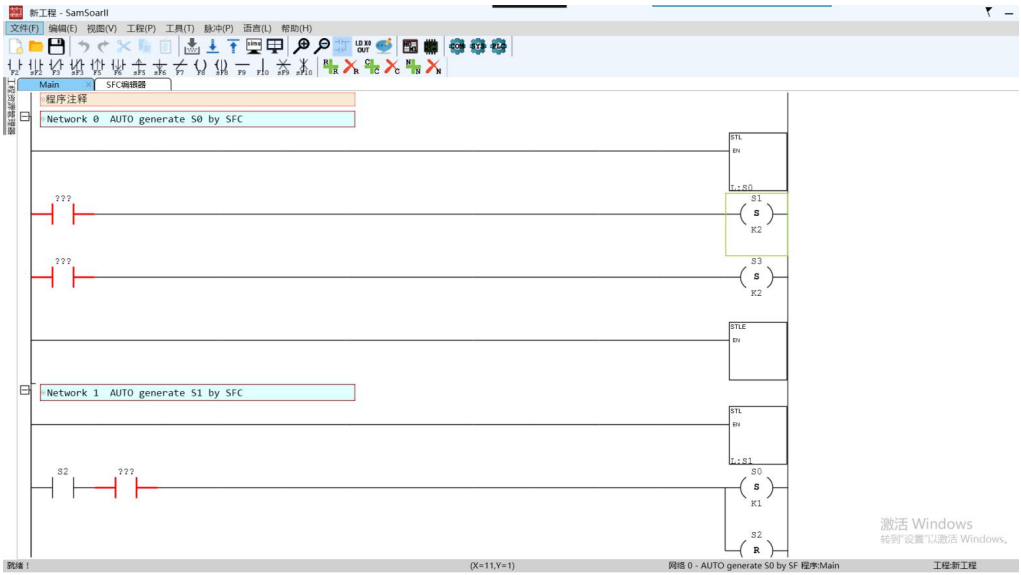


图 1.2

在这个位置更改完毕后，可以返回 SFC 编辑器，通过这样的查看-导航模式来完成剩余的部分。如果对 STL 顺序控制较为熟悉，也可以停留在梯形图界面内，直接修改 STL 结构。

9. 寄存器

SamSoarII 系统寄存器

系统寄存器			
存器名称	范围	存储单位	掉电保持范围
输入触点 X	[0,128)	BOOL	无
输出线圈 Y	[0,128)	BOOL	无
模拟输入寄存器 AI	[0,32)	WORD	无
模拟输出寄存器 AO	[0,32)	WORD	无
位寄存器 M	[0,8192)	BOOL	[0,7999]
数据寄存器 D	[0,8192)	WORD	[0,5999]
变址寄存器 V	[0,8)	WORD	无
变址寄存器 Z	[0,8)	WORD	无
系统状态寄存器 S	[0,1000)	BOOL	[600,1000)
计数器 CV	[0, 235) 低速，[235, 256)高速	[0,200)WORD, [200, 256)DWORD	[100,200), [220,235),[235,256)
定时器 TV	[0,256)	WORD	无
计数器开关量 C	[0,256)	BOOL	无
定时器开关量 T	[0,256)	BOOL	无

掉电保持范围:
位寄存器 M　　最大长度 8000
数据寄存器 D　　最大长度 6000

系统特殊功能寄存器		
存器名称	功能	读写权限
数据寄存器 D		
D_8066	COM0（232）站号，范围 1~127　（附：D8066~D8075 需掉电重启后才生效）	Read & Write
D_8067	COM1（485）站号，范围 1~127	Read & Write
D_8068	COM0（232）波特率，范围 0~7，分别对应配置信息中的 1200~115200 波特率	Read & Write
D_8069	COM1（485）波特率，范围 0~7，分别对应配置信息中的 1200~115200 波特率	Read & Write
D_8070	COM0（232）数据位,范围 0，对应数据位为 8	Read & Write
D_8071	COM1（485）数据位,范围 0，对应数据位为 8	Read & Write
D_8072	COM0（232）停止位,范围 0~1，对应停止位为 1、2	Read & Write
D_8073	COM1（485）停止位,范围 0~1，对应停止位为 1、2	Read & Write
D_8074	COM0（232）校验码，范围 0~2，对应校验码为 NONE、EVEN、ODD	Read & Write

D_8075	COM1（485）校验码，范围 0~2，对应校验码为 NONE、EVEN、ODD	Read & Write
D_8076	Y000 原点回归加减速时间	Read & Write
D_8077	Y001 原点回归加减速时间	Read & Write
D_8078	Y002 原点回归加减速时间	Read & Write
D_8079	Y003 原点回归加减速时间	Read & Write
D_8092	Y000 绝对，相对定位指令的加减速时间	Read & Write
D_8093	Y001 绝对，相对定位指令的加减速时间	Read & Write
D_8094	Y002 绝对，相对定位指令的加减速时间	Read & Write
D_8095	Y003 绝对，相对定位指令的加减速时间	Read & Write
D_8108	Y000 出错脉冲段编号（用于多段脉冲）	Read
D_8109	Y001 出错脉冲段编号（用于多段脉冲）	Read
D_8110	Y002 出错脉冲段编号（用于多段脉冲）	Read
D_8111	Y003 出错脉冲段编号（用于多段脉冲）	Read
D_8124	Y000 当前脉冲段（用于多段脉冲）	Read
D_8125	Y001 当前脉冲段（用于多段脉冲）	Read
D_8126	Y002 当前脉冲段（用于多段脉冲）	Read
D_8127	Y003 当前脉冲段（用于多段脉冲）	Read
D_8140(D_8141)	Y000 的 32 位脉冲累积计数	Read & Write
D_8142(D_8143)	Y001 的 32 位脉冲累积计数	Read & Write
D_8144(D_8145)	Y002 的 32 位脉冲累积计数	Read & Write
D_8146(D_8147)	Y003 的 32 位脉冲累积计数	Read & Write
D_8173	中断定时器 0 设定值，设定值范围 1~32767ms	Read & Write
D_8174	中断定时器 1 设定值，设定值范围 1~32767ms	Read & Write
D_8175	当前扫描周期（0.1ms）	Read
D_8176	ERROR 状态错误编号	Read & Write
D_8177	最大扫描周期（0.1ms）	Read & Write
线圈寄存器 M		
M_8035~M_8045	预留	Read & Write
M_8046	扩展模块初始化错误标志位	Read
M_8051	高速计数器 C251 倍频设置（1—四倍频 0—一倍频）	Read & Write
M_8052	高速计数器 C252 倍频设置（1—四倍频 0—一倍频）	Read & Write
M_8053	预留	Read & Write
M_8054	高速计数器 C254 倍频设置（1—四倍频 0—一倍频）	Read & Write
M_8055	预留	Read & Write
M_8070	Y000 是否忽略错误，继续发送脉冲，默认忽略为 0；停止脉冲输出为 1	Read & Write
M_8071	Y001 是否忽略错误，继续发送脉冲，默认忽略为 0；停止脉冲输出为 1	Read & Write
M_8072	Y002 是否忽略错误，继续发送脉冲，默认忽略为 0；停止脉冲输出为 1	Read & Write

M_8073	Y003 是否忽略错误，继续发送脉冲，默认忽略为 0；停止脉冲输出为 1	Read & Write
M_8086	Y000 检查每段用户设置脉冲个数和频率是否正确(用于多段脉冲)	Read
M_8087	Y001 检查每段用户设置脉冲个数和频率是否正确(用于多段脉冲)	Read
M_8088	Y002 检查每段用户设置脉冲个数和频率是否正确(用于多段脉冲)	Read
M_8089	Y003 检查每段用户设置脉冲个数和频率是否正确(用于多段脉冲)	Read
M_8102	Y000 脉冲方向，正方向为 1	Read
M_8103	Y001 脉冲方向，正方向为 1	Read
M_8104	Y002 脉冲方向，正方向为 1	Read
M_8105	Y003 脉冲方向，正方向为 1	Read
M_8118	Y000 脉冲累积计数溢出标记，溢出为 1	Read
M_8119	Y001 脉冲累积计数溢出标记，溢出为 1	Read
M_8120	Y002 脉冲累积计数溢出标记，溢出为 1	Read
M_8121	Y003 脉冲累积计数溢出标记，溢出为 1	Read
M_8134	Y000 脉冲正在输出	Read
M_8135	Y001 脉冲正在输出	Read
M_8136	Y002 脉冲正在输出	Read
M_8137	Y003 脉冲正在输出	Read
M_8150	第一个扫描周期接通，接通一个扫描周期	Read
M_8151	程序运行时闭合	Read
M_8152	程序运行时断开	Read
M_8153	RTC 电池电压低（注:1-ERROR 0-OK ）	Read
M_8154	DC24V 故障（注:1-ERROR 0-OK ）	Read
M_8155	IO 配置错误（注:1-ERROR 0-OK ）	Read
M_8156	PLC 硬件错误（注:1-ERROR 0-OK ）	Read
M_8157	输出刷新错误（注:1-ERROR 0-OK ）	Read
M_8158	10ms 时钟脉冲输出（注:1-HIGH 0-LOW ）	Read
M_8159	100ms 时钟脉冲输出（注:1-HIGH 0-LOW ）	Read
M_8160	1sec 时钟脉冲输出（注:1-HIGH 0-LOW ）	Read
M_8161	1min 时钟脉冲输出（注:1-HIGH 0-LOW ）	Read
M_8162	非掉电保持区内存全部清除（ 注:1-CLEAR 0-NONE ）	Read & Write
M_8163	掉电保持区内存全部清除（ 注:1-CLEAR 0-NONE ）	Read & Write
M_8164	所有输出禁止（注:1-DISABLE 0-ENABLE ）	Read & Write
M_8165	位移动指令方向指示（注:1-LEFT 0-RIGHT ）	Read
M_8166	移位指令溢出位（注:1-ERROR 0-OK ）	Read
M_8167	移位指令零标志（注:1-ERROR 0-OK ）	Read
M_8168	BCD 转换错误标记（注:1-ERROR 0-OK ）	Read
M_8169	运算结果溢出（注:1-ERROR 0-OK ）	Read
M_8170	运算负结果（注:1-ERROR 0-OK ）	Read
M_8171	运算结果为零（注:1-ERROR 0-OK ）	Read
M_8172	除数为零（注:1-ERROR 0-OK ）	Read
M_8173	偏移地址越界（注:1-ERROR 0-OK ）	Read
M_8174	错误状态（注:1-ERROR 0-OK ）	Read
M_8175	运行状态（注:1-RUN 0-STOP ）	Read

M_8176	COM0（232）正在发送标志	Read
M_8177	COM1（485）正在发送标志	Read
M_8178	COM0（232）正在接收标志	Read
M_8179	COM1（485）正在接收标志	Read
M_8180	COM0（232）发送完成标志	Read & Write
M_8181	COM1（485）发送完成标志	Read & Write
M_8182	COM0（232）接收完成标志	Read & Write
M_8183	COM1（485）接收完成标志	Read & Write
M_8184	扩展模块 1 传输数据错误标志位	Read
M_8185	扩展模块 2 传输数据错误标志位	Read
M_8186	扩展模块 3 传输数据错误标志位	Read
M_8187	扩展模块 4 传输数据错误标志位	Read
M_8188	扩展模块 5 传输数据错误标志位	Read
M_8189	扩展模块 6 传输数据错误标志位	Read
M_8190	扩展模块 7 传输数据错误标志位	Read
M_8191	扩展模块 8 传输数据错误标志位	Read

10. 通信协议

1. MODBUS 简介

FGs 系列 PLC 采用的是 MODBUS 通信协议。Modbus 协议是应用于控制器上的一种通用语言。通过此协议，控制器相互之间、控制器经由网络和其它设备之间可以通信。通讯采用应答方式，由主机发起请求，从机执行请求并且应答。PLC 最大从站数为 127 个，设置的范围为 1~127。

2. 接口方式

目前 PLC 支持 MODBUS 通信的接口为 RS232、RS485 以及 USB 接口。

串口通信默认的数据格式为：9600bps、8 位数据位、1 位停止位，无校验。波特率的可设置范围是：1200bps、2400bps、4800bps、9600bps、19200bps、38400bps、57600bps、115200bps

3. 协议格式

协议支持的是 MODBUS 的 RTU 模式

从机地址	功能码	数据	CRC 高字节	CRC 低字节

4. 功能码描述

本系列 PLC 支持的功能码如下：

功能码	名称	作用	支持的寄存器
0x01	读取线圈状态	取得一组逻辑线圈的当前状态（ON/OFF）	Y、X、M、S、T、C
0x02	读取输入状态	取得一组开关输入的当前状态（ON/OFF）	X
0x03	读取保持寄存器	在一个或多个保持寄存器中取得当前的二进制值	D、TV、CV、CV32、AI、AO
0x04	读取输入寄存器	在一个或多个输入寄存器中取得当前的二进制值	无
0x05	强置（写）单线圈	写一个逻辑线圈的通断状态	Y、M、S、T、C
0x06	预置（写）单寄存器	把具体二进值装入一个数据寄存器	D、TV、CV、CV32、AI、AO
0x08	回送诊断校验	把诊断校验报文送从机，以对通信处理进行评鉴	无
0x0f	强置（写）多线圈	强置（写）一串连续逻辑线圈的通断	Y、M、S、T、C
0x10	预置（写）多寄存器	把具体的二进制值装入一串连续的保持寄存器	D、TV、CV、CV32、AI、AO

5.地址偏移表

寄存器	地址范围	数据类型	读功能码	写功能码	对应 modbus 地址加上偏移量
X	0~127	bit	0x01/0x02	0x05	0
EX	0~511	bit	0x01/0x02	0x05	512
Y	0~127	bit	0x01	0x05/0x0f	10000
EY	0~511	bit	0x01/0x02	0x05	10512
M	0~8191	bit	0x01	0x05/0x0f	30000
S	0~999	bit	0x01	0x05/0x0f	50000
T	0~255	bit	0x01	0x05/0x0f	60768
C	0~255	bit	0x01	0x05/0x0f	60512
D	0~8191	word	0x03/0x04	0x06/0x10	40000
TV	0~255	word	0x03	0x06/0x10	60256
CV	0~199	word	0x03	0x06/0x10	60000
CV32	200~255	dword	0x03	0x06/0x10	60200
AI	0~31	word	0x03	0x06/0x10	20000
EAI	0~79	word	0x03	0x06/0x10	20100
AO	0~31	word	0x03	0x06/0x10	20512
EAO	0~79	word	0x03	0x06/0x10	20612

1. CV32 用标准的 MODBUS 通信协议访问 1 长度将返回/写入双字。
2. EX,EY,EAI,EAO 分别为 X,Y,AI,AO 的扩展模块部分。

扩展模块的 EX 和 EY 是八进制，modbus 是以十进制来计算的，带 A 的附件是属于本体的

寄存器	地址范围	对应 modbus 地址
EX	1000-1017	512-527
	1100-1117	576-591
	1200-1217	640-655
	1300-1317	704-719
	1400-1417	768-783
	1500-1517	832-847
	1600-1617	896-911
	1700-1717	960-975

1000	512+0=512
1001	512+1=513
1002	512+2=514
1003	512+3=515
1004	512+4=516
1005	512+5=517
1006	512+6=518
1007	512+7=519
1010	512+8=520
1011	512+9=521
1012	512+10=522
1013	512+11=523
1014	512+12=524
1015	512+13=525
1016	512+14=526
1017	512+15=527

1000	512
1100	512+1*64=576
1200	512+2*64=640
1300	512+3*64=704
1400	512+4*64=768
1500	512+5*64=832
1600	512+6*64=896
1700	512+7*64=960

11. 扩展模块

扩展模块是在 PLC 基本单元功能的扩展和延伸，直观上达到了输入输出点数的增加，更重要的是增加了用户项目开发的灵活性和便利性。

目前所支持的模块包括：

数字量模块：FGs-E8X、FGs-E8T、FGs-E8R、FGs-E8X8T、FGs-E8X8R、FGs-E16X、FGs-E16T、FGs-E16R、FGR-E8X8R、FGR-E8X8T、FGR-E16X、FGR-E16T、FGR-12R；

模拟量模块：FGs-E4AI、FGs-E2AO、FGs-E4AI2AO、FGs-E4TC、FGR-E4TC、FGR-E4AI2AO。

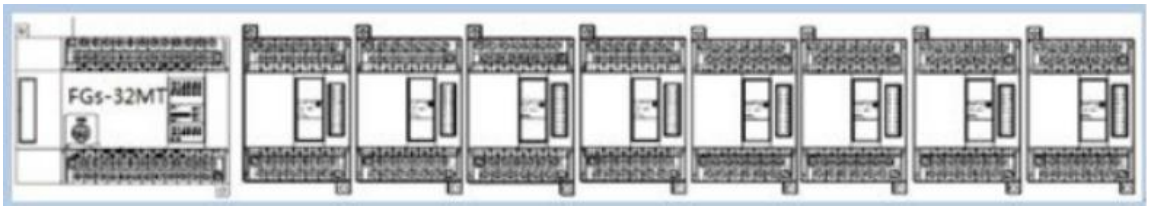
FGs-E 系列与 FGR-E 系列 PLC 扩展模块不仅具有强大的逻辑处理、数据运算、高速处理等功能，而且具有基本的输入输出、 A/D、 D/A 转换等功能， 通过使用输入输出模块、模拟量模块、 温度控制模块等等，使 FGs-E 系列与 FGR-E 系列的 PLC 在温度、流量、 液位、压力等过程控制系统中得到了广泛的应用。

1. 模块型号及功能

型号	功能
FGs-E8X	8 通道数字量输入模块
FGs-E8T	8 通道数字量晶体管输出模块
FGs-E8R	8 通道数字量继电器输出模块
FGs-E8X8T	8 通道数字量输入，8 通道数字量晶体管输出模块
FGs-E8X8R	8 通道数字量输入，8 通道数字量继电器输出模块
FGs-E16X	16 通道数字量输入模块
FGs-E16T	16 通道数字量晶体管输出模块
FGs-E16R	16 通道数字量继电器输出模块
FGs-E4AI	4 通道模拟量输入（12-bit）模块，输入电压电流均可选
FGs-E2AO	2 通道模拟量输出（16-bit）模块，输出电压电流均可选
FGs-E4AI2AO	4 通道模拟量输入（12-bit），2 通道模拟量输出（16-bit）模块，输入输出电压电流均可选
FGs-E4TC	4 通道模拟量输入（16-bit）模块，输入 PT100、K 型热电偶均可选
FGR-E8X8R	MINI 型 8 通道数字量输入，8 通道数字量继电器输出模块
FGR-E8X8T	MINI 型 8 通道数字量输入，8 通道数字量晶体管输出模块
FGR-E16X	MINI 型 16 通道数字量输入模块
FGR-E16T	MINI 型 16 通道数字量晶体管输出模块
FGR-E12R	MINI 型 12 通道数字量继电器输出模块

图 1.1

FGs-E 系列扩展模块可以安装在 FGs 系列 PLC 的主单元一侧：



输入输出开关量序号为八进制

输入输出模拟量序号为十进制，每个模块固定占四个寄存器，如有缺失通道则不作使用

PLC 本体最多可以外接 8 个扩展模块，种类不受限制，可以为输入输出开关量，也可以是模拟量、温度控制模块等。

FGR-E 系列 MINI 型扩展模块可以安装在 FGR 系列 MINIPLC 的主单元一侧：

输入输出开关量序号为八进制

输入输出模拟量序号为十进制，每个模块固定占四个寄存器，如有缺失通道则不作使用

扩展 IO 板相当于 PLC 本体上的 IO 点，XY 编号以 8 进制紧跟 PLC 本体编号（如 X0010-X0017，Y0010-Y0017）

PLC 本体最多可以外接 3 个扩展模块与 1 个扩展 IO 板。扩展模块种类不受限制，可以为输入输出开关量，也可以是模拟量、温度控制模块等，扩展 IO 板只能是开关量模块。

SAMSOARII 中的配置

模块在使用时，首先需要在 PLC 的上位机编程软件 SamSoII 中进行相应的配置，方可正常使用模块。下面分别以模块 FGs-E 系列与 FGR-E 系列为例，说明如何在 SAMSOARII 中进行配置，步骤如下所示：

FGs-E 系列： A、 打开编程软件，点击菜单栏：工程 ＞PLC 参数设置；或直接点击图标 ，将弹出系统参数设置对话框，如下图所示



B、 然后选择“扩展模块设置”，将切换到“扩展模块设置”页面，如下图所示：



C、 勾选“使用扩展功能”，启动扩展模块功能，如下图所示：



然后选择目标模块类型，并勾选“启用”来激活此模块，此时模块类型列表中会显示出对应序号的模块类型（FGs-E4AI2AO 为例），

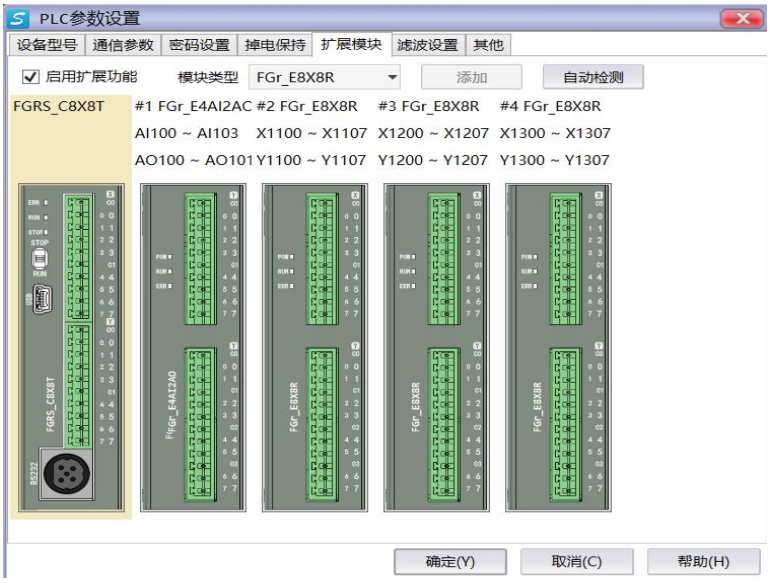
点击输入部分的“通道”右侧的下拉框进行通道切换，勾选“启用”后，相应的通道才可以使用，不同的通道有不同的通道参数，参数主要包括：模式、平均采样次数、数字量范围；输出部分与输入雷同。需要注意的是每个模拟量模块都会固定占用 4 个 AI 或 4 个 AO 寄存器，即使没有使用。另外系统根据模块的序列和相应模块的类型自动在对话框页面右下方提示模块寄存器的地址。

FGR-E 系列：

A、 打开编程软件，点击菜单栏：工程 > PLC 参数设置；或直接点击图标，将弹出系统参数设置对话框，如下图所示



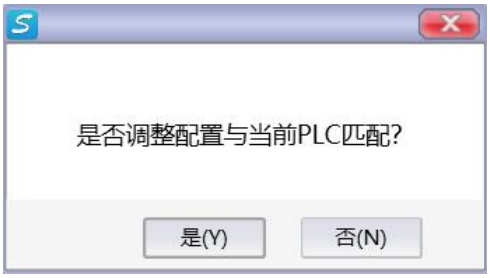
B、 然后选择“扩展模块设置”，切换到“扩展模块设置”页面，勾选启用扩展模块，然后可以选择模块类型手动添加，点击模块图片可以选择是否启用该扩展模块，如下图所示：





C、 如果用户点击自动检测，PLC 将短暂的进入 stop 状态并且检测实际连接的扩展模块类型，并提示用户是否该需要调整配置与实际一致，如下图所示：





如果用户均选择是，则会自动调整配置与实际连接情况一致，否则将维持用户的配置不变。

D、 如果用户选择启用扩展模块，则在下载的初始阶段会进行扩展模块类型检测，用户配置（包含未启用的模块）与实际的连接模块如果有数量和型号不一致的情况会进行报错，如果用户选择继续下载则继续，否则退回配置界面：



注：如果用户选择未启用某个扩展模块，也需要在实际对应位置连接相应的扩展模块，否则会在下载时提示数量不对应，用户也可以选择直接删除该扩展模块的配置。未启用的扩展模块可以位于任意位置，并且在下载的时候会提示。

3. 模拟量输入输出模块 FGs-E4AI2AO

本章主要介绍 FGs-E4AI2AO 模块的规格、端子说明、输入定义号的分配、 工作模式设定、 外部连接以及模数转换图。

3.1 模块特点及规格

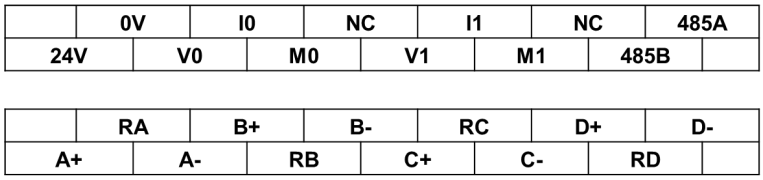
FGs-E4AI2AO 模拟量输入输出模块将 4 点模拟输入数值（电压输入， 电流输入）转换成数字值，并且把他们传输到 PLC 主单元，同时还可以将 2 点模拟量输出的数值（电压输出，电流输出）转换成模拟量的输出。

* 4 通道模拟量输入：电压输入或电流输入自由选择，其中电压输入有 0～5V、 0～10V 两种模式；电流输入有 4～20mA 模式。
* 12 位的高精度模拟量输入和 16 位的高精度模拟量输出。
* 作为 FGs-E 系列的特殊功能模块，最多可在 PLC 主单元右边连接 6 台 FGs-E4AI2AO 模块。
* FGs-E4AI2AO 模拟、数字部分电源隔离处理。

模块规格:

项目	电压输入	电流输入	电压输出	电流输出
模拟量输入范围	0～5V、 0～10V	4～20mA	0～5V、 0～10V	4～20mA
最大输入输出范围	±15V	0～40mA	±15V	0～40mA
数字输入输出范围	12 位二进制数		16 位二进制数	
分辨率	1/4095（ 12Bit）		1/65535（16Bit）	
综合精确度	1‰		1‰	
转换速度	20ms/通道		20ms/通道	
模拟量用电源	DC24V±10%， 100mA			

3.2 端子说明



端子信号

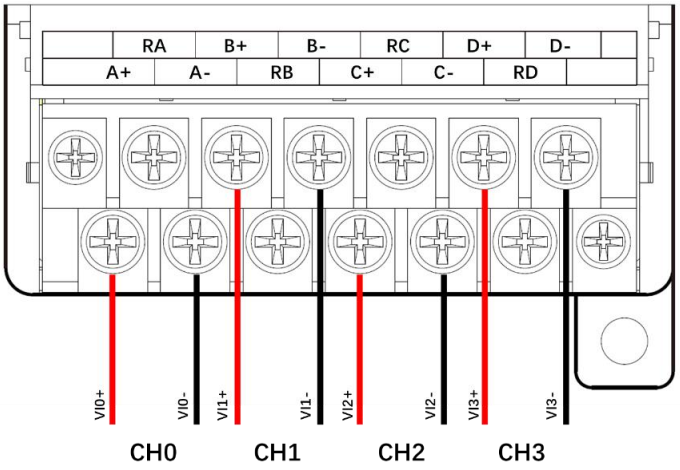
	通道	端子名	信号名
模拟量输入	输入 CH0	A+	A 模拟量+
		RA	如果是电流输入需与 A+短接
		A-	A 模拟量-
	输入 CH1	B+	B 模拟量+
		RB	如果是电流输入需与 B+短接
		B-	B 模拟量-
	输入 CH2	C+	C 模拟量+
		RC	如果是电流输入需与 C+短接
		C-	C 模拟量-
	输入 CH3	D+	D 模拟量+
		RD	如果是电流输入需与 D+短接
		D-	D 模拟量-
模拟量输出	输出 CH0	V0	A 输出+（电压）

		I0	A 输出+（电流）
		M0	A 输出-
	输出 CH1	V1	B 输出+（电压）
		I1	B 输出+（电流）
		M1	B 输出-
485	-	485A	485 通讯+
	-	485B	485 通讯-

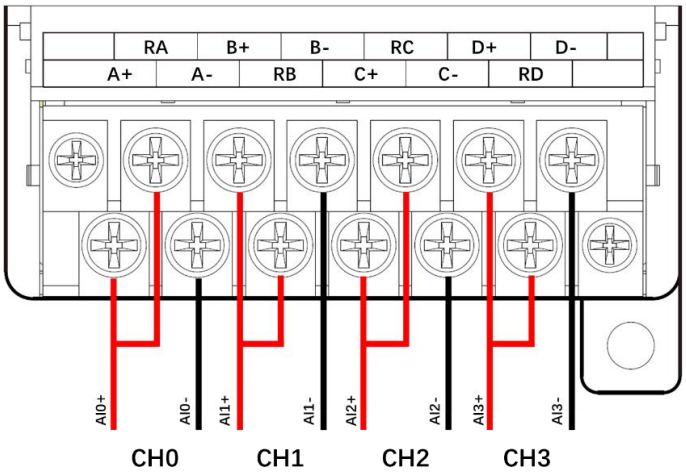
注：485 只作从机使用

3.3 输 FGs-E4AI2AO 接线示意图

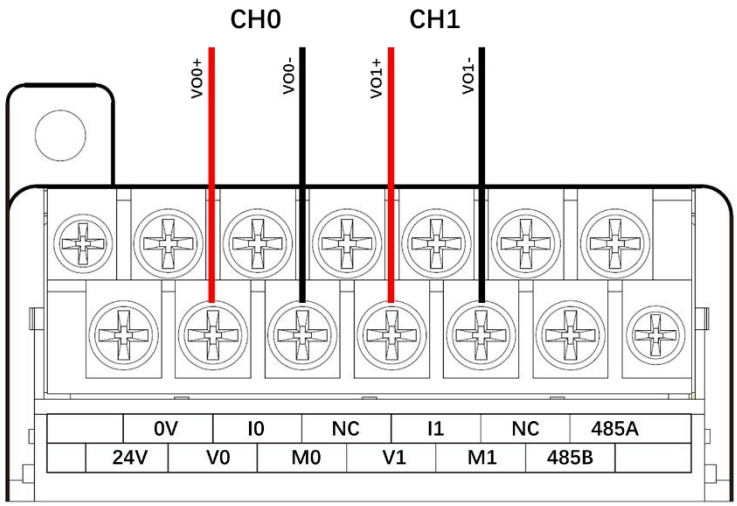
3.3.1 模拟量电压输入



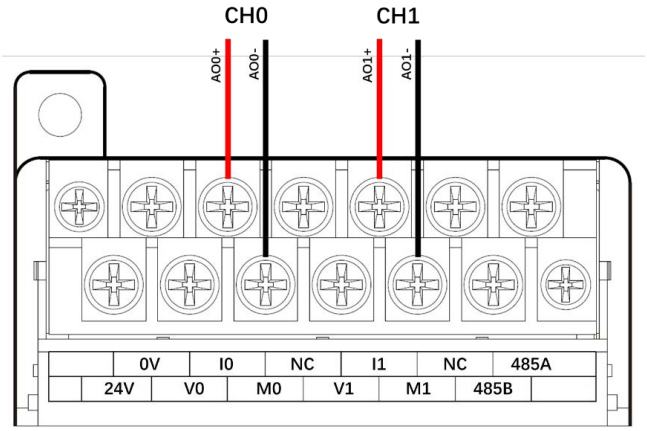
3.3.2 模拟量电流输入



3.3.3 模拟量电压输出



3.3.4 模拟量电流输出



3.4 输入输出定义号分配

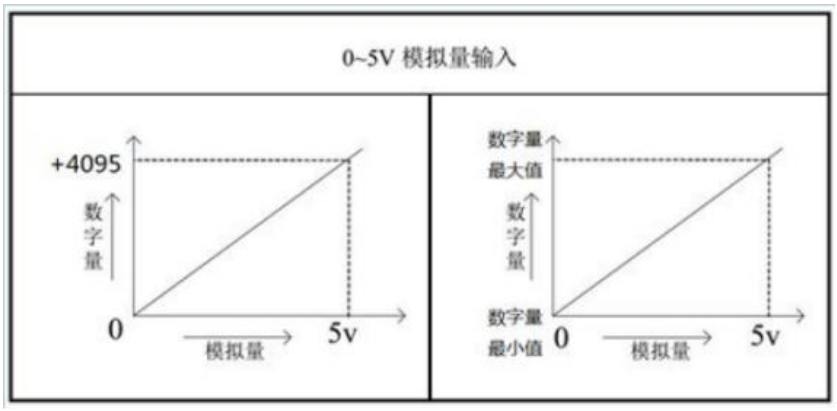
连续 8 个（MINI 型 3 个）FGs-E4AI2AO 扩展模块寄存器定义号：

	扩展模块 1	扩展模块 2	扩展模块 3	扩展模块 4	扩展模块 5	扩展模块 6	扩展模块 7	扩展模块 8
输入通道	AD 信号	AD 信号	AD 信号	AD 信号	AD 信号	AD 信号	AD 信号	AD 信号
输入 CH0	AI100	AI110	AI120	AI130	AI140	AI150	AI160	AI170
输入 CH1	AI101	AI111	AI121	AI131	AI141	AI151	AI161	AI171
输入 CH2	AI102	AI112	AI122	AI132	AI142	AI152	AI162	AI172
输入 CH3	AI103	AI113	AI123	AI133	AI143	AI153	AI163	AI173
输出通道	DA 信号	DA 信号	DA 信号	DA 信号	DA 信号	DA 信号	DA 信号	DA 信号

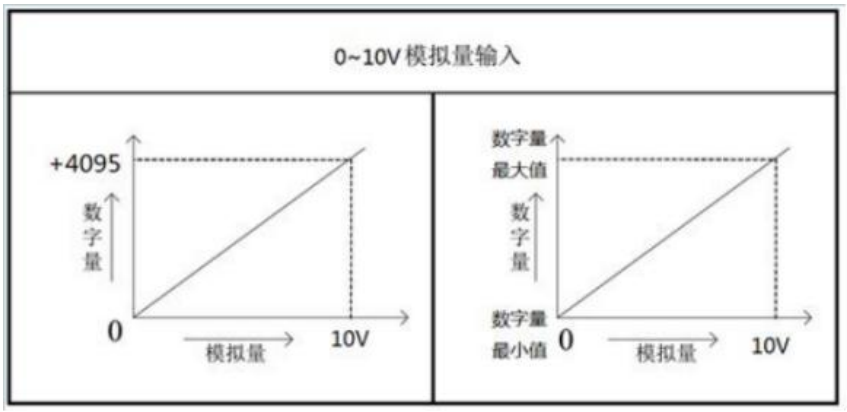
输出 CH0	AO100	AO110	AO120	AO130	AO140	AO150	AO160	AO170
输出 CH1	AO101	AO111	AO121	AO131	AO141	AO151	AO161	AO171

3.5 模数转换图

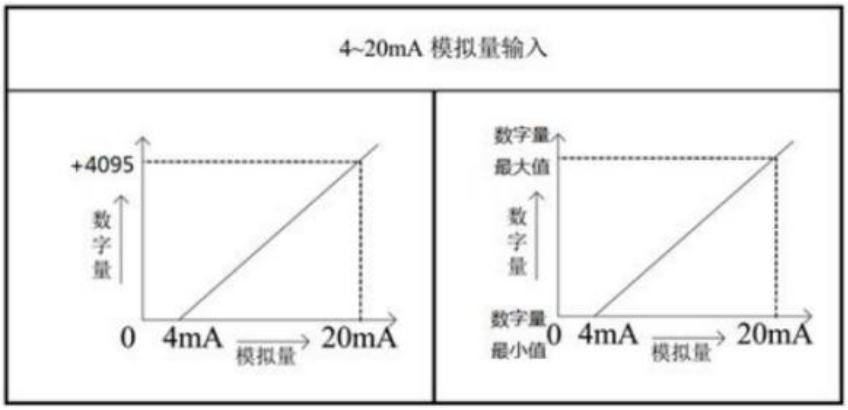
0~5V 模拟量输入数字量对应关系



0~10V 模拟量输入数字量对应关系



4~20mA 模拟量输入数字量对应关系



同样的道理也适用于模拟量输出，不过输出的精度是 16bit，即 1/65535。

4. 模拟量温度模块 FGs-E4TC

本章主要介绍 FGs-E4TC 模块的规格、端子说明、输入定义号的分配、 工作模式设定、 外部连接以及模数转换图。

4.1 FGs-E4TC 模块特点及规格

FGs-E4TC 模拟量输入输出模块将 4 点温度的模拟量转换成数字值，并且把他们传输到 PLC 主单元。

* 4 通道温度模拟量输入：支持 K 型热电偶或 PT100 热电阻两种传感器。
* 16 位的高精度模拟量输入。
* 作为 FGs-E 系列的特殊功能模块，最多可在 PLC 主单元右边连接 6 台 FGs-E4TC 模块。
* FGs-E4TC 模拟、数字部分电源隔离处理。

模块规格

项目	PT100	K 型热电偶
数字输入输出范围	16 位二进制数	
分辨率	1/65535（ 16 Bit）	
综合精确度	1‰	
转换速度	20ms/通道	
模拟量用电源	DC24V±10%， 100mA	

4.2 FGs-E4TC 端子说明

	0V	NC	NC	NC	NC	485A
24V	NC	NC	NC	NC	485B	

	A1	B0	B3	C1	D0	D3
A0	A3	B1	C0	C3	D1	

图 5.1

FGs-E4TC 端子信号

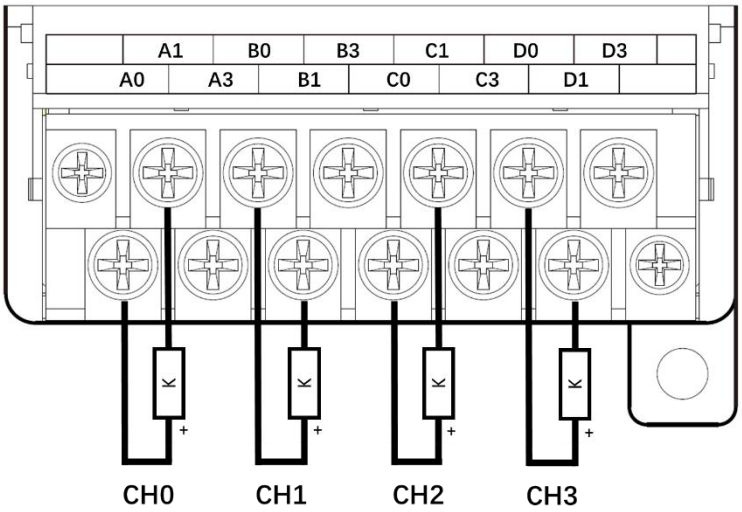
	通道	端子名	信号名
温度输入	输入 CH0	A0	CH0 热电阻、热电偶输入端
		A1	CH0 热电阻、热电偶输入公共端
		A3	CHO 热电阻 A0 同色端（三线制）
	输入 CH1	B0	CH1 热电阻、热电偶输入端
		B1	CH1 热电阻、热电偶输入公共端
		B3	CH1 热电阻 B0 同色端（三线制）
	输入 CH2	C0	CH2 热电阻、热电偶输入端
		C1	CH2 热电阻、热电偶输入公共端
		C3	CH2 热电阻 C0 同色端（三线制）
	输入 CH3	D0	CH3 热电阻、热电偶输入端
		D1	CH3 热电阻、热电偶输入公共端
		D3	CH3 热电阻 D0 同色端（三线制）
485	-	485A	485 通讯+
	-	485B	485 通讯-

图 5.2

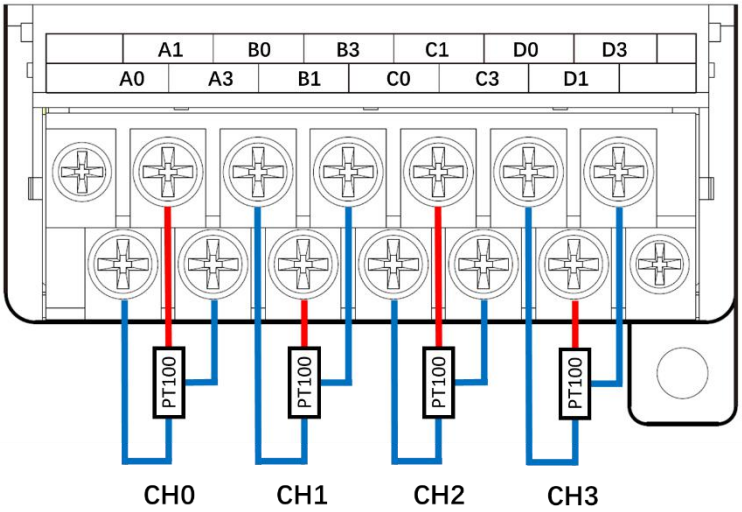
注：485 只作从机使用

4.3 输 FGs-E4TC 接线示意图

4.3.1 K 型热电偶



4.3.2 PT100 型热电阻



4.4 FGs-E4TC 输入输出定义号分配

连续 8 个（MINI 型为 3 个）FGs-E4TC 模块寄存器定义号

	扩展模块 1	扩展模块 2	扩展模块 3	扩展模块 4	扩展模块 5	扩展模块 6	扩展模块 7	扩展模块 8
通道	AD 信号	AD 信号	AD 信号	AD 信号	AD 信号	AD 信号	AD 信号	AD 信号
输入 CH0	AI100	AI110	AI120	AI130	AI140	AI150	AI160	AI170
输入 CH1	AI101	AI111	AI121	AI131	AI141	AI151	AI161	AI171
输入 CH2	AI102	AI112	AI122	AI132	AI142	AI152	AI162	AI172
输入 CH3	AI103	AI113	AI123	AI133	AI143	AI153	AI163	AI173

5. 数字量模块

FGs-E8X\FGs-E8T\FGs-E8R\FGs-E8X8T\FGs-E8X8R\FGs-E16X\FGs-E16T\FGs-E16R
FGR-E8X8R\FGR-E8X8T\FGR-E16X\FGR-16T\FGR-12R

5.1 模块特点及规格

数字量输入输出模块,基本型 PLC 可扩展 8 个模块，MINI 型 PLC 可扩展 1 个 IO 板加 3 个模块，模块种类丰富，外形小巧，为更多的输入和输出点提供了可能，满足了实际生产需要。

* 晶体管输出的频率最大为 200Hz，继电器则建议不要超过 10Hz
* 数字量输入采用兼容设计，可以根据 SS 的状态来选择 NPN 输入或是 PNP 输入，接收的动作间隔最快不能超过 200Hz

5.2 数字量模块端子说明

FGs-E8X、FGs-E8T、FGs-E8R、FGs-E8X8T、FGs-E8X8R、FGR-E8X8R、FGR-E8X8T（FGs-E8T、FGs-E8X8T、FGR-E8X8R、FGR-E8X8T 的 COM0\COM1\COM2 内部短接）

	NC	S/S	X1	X3	X5	X7
NC	NC	X0	X2	X4	X6	

	Y0	Y1	Y2	COM3	Y5	Y7
COM0	COM1	COM2	Y3	Y4	Y6	

FGs-E16X、FGR-E16X

	NC	S/S	X11	X13	X15	X17
NC	NC	X10	X12	X14	X16	

	NC	S/S	X1	X3	X5	X7
NC	NC	X0	X2	X4	X6	

FGs-E16T 、 FGs-E16R 、 FGR-E16T 、 FGR-E12R （ FGs-E16T 、 FGs-E16R 、 FGR-E16T 、 FGR-E12R 的 COM0\COM1\COM2 内部短接，COM10\COM11\COM12 内部短接）

	Y10	Y11	Y12	COM13	Y15	Y17
COM10	COM11	COM12	Y13	Y14	Y16	

	Y0	Y1	Y2	COM3	Y5	Y7
COM0	COM1	COM2	Y3	Y4	Y6	

5.3 数字量模块输入输出定义号分配

连续 8 个（MINI 型为 3 个）数字量扩展模块寄存器定义号

	扩展模块 1		扩展模块 2		扩展模块 3		扩展模块 4	
类型	X 信号	Y 信号	X 信号	Y 信号	X 信号	Y 信号	X 信号	Y 信号
FGs-E8X	X1000-X1007	-	X1100-X1107	-	X1200-X1207	-	X1300-X1307	-
FGs-E8T\FGs-E8R	-	Y1000-Y1007	-	Y1100-Y1107	-	Y1200-Y1207	-	Y1300-Y1307
FGs-E8X8T\FGs-E8X8R\GR-E8X8T\FGR-E8X8R	X1000-X1007	Y1000-Y1007	X1100-X1107	Y1100-Y1107	X1200-X1207	Y1200-Y1207	X1300-X1307	Y1300-Y1307
FGs-E16X\FGR-E16X	X1000-X1017	-	X1100-X1117	-	X1200-X1217	-	X1300-X1317	-
FGs-E16T\FGs-E16R\FGR-E16T\FGR-12R	-	Y1000-Y1017	-	Y1100-Y1117	-	Y1200-Y1217	-	Y1300-Y1317

	扩展模块 5		扩展模块 6		扩展模块 7		扩展模块 8	
类型	X 信号	Y 信号	X 信号	Y 信号	X 信号	Y 信号	X 信号	Y 信号
FGs-E8X	X1400-X1407	-	X1500-X1507	-	X1600-X1607	-	X1700-X1707	-
FGs-E8T\FGs-E8R	-	Y1400-Y1407	-	Y1500-Y1507	-	Y1600-Y1607	-	Y1700-Y1707
FGs-E8X8T\FGs-E8X8R	X1400-X1407	Y1400-Y1407	X1500-X1507	Y1500-Y1507	X1600-X1607	Y1600-Y1607	X1700-X1707	Y1700-Y1707
FGs-E16X	X1400-X1417	-	X1500-X1517	-	X1600-X1617	-	X1700-X1717	-
FGs-E16T\FGs-E16R	-	Y1400-Y1417	-	Y1500-Y1517	-	Y1600-Y1617	-	Y1700-Y1717

6. 异常情况说明

6.1 异常时的指示灯情况

异常情况	**数字量模块**	**FGs_E4AI2AO**	**FGs_E4TC**
RUN 灯闪烁	情况 1：PLC 拨码处于 stop 状态，此时 ERR 亮，检查配置后重新上电或切换 PLC 拨码		
	情况 2：数字量输入输出不工作，请重新上电或切换拨码	情况 2：模拟量输出异常，请检查外部 24V 电源然后重新上电或切换拨码	

POW 灯闪烁	MINI 型扩展模块内部 IO 板与核心板不匹配（IO 板匹配错误或损坏）
ERR 灯亮	PLC 未正确识别模块（干扰引起或 PLC 拨码 stop），检查配置后重新上电或切换拨码开关
ERR 灯快闪	输入/输出数据处于异常状态（通信过程中存在干扰）

扩展模块正常状态下：ERR 灯(红灯)常灭，PWR 灯常亮，RUN 灯常亮。

6.2 异常时的标志位寄存器

	特殊寄存器	功能	
初始化过程	M[8046]	所有扩展模块中任一初始化失败时都会置 1	
数据传输过程	M[8184]	扩展模块 1 传输数据错误时置 1	
	M[8185]	扩展模块 2 传输数据错误时置 1	
	M[8186]	扩展模块 3 传输数据错误时置 1	
	M[8187]	扩展模块 4 传输数据错误时置 1	
	M[8188]	扩展模块 5 传输数据错误时置 1	
	M[8189]	扩展模块 6 传输数据错误时置 1	
	M[8190]	扩展模块 7 传输数据错误时置 1	
	M[8191]	扩展模块 8 传输数据错误时置 1	

12. 硬件手册

一、硬件参数表

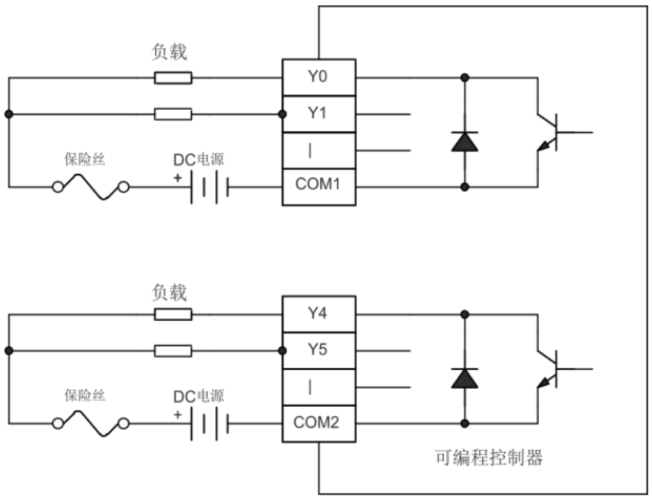
项目	FGs-64MR-A	FGs-32MR-A	FGs-16MR-A	FGs-64MT-A	FGs-32MT-A	FGs-16MT-A
尺寸(mm)	216*92*80	147.6*92*80	109.5*92*80	216*92*80	147.6*92*80	109.5*92*80
重量(g)	831	590	465	769	563	464
功耗	6.8W	5.2W	3.2W	3.6W	3W	2.6W
SINK/SRCE 输入接线	借由内部共点端子 S/S 及外部共线的接线来变换					
输入信号电压	24VDC±10%					
输入滤波	0~50ms 可调					
数字量输入	32	16	8	32	16	8
输入信号逻辑 0 限值	X000~X003 最大 10.6VDC，X004~X037 最大 12.2VDC	X000~X001 最大 10.6VDC，X002~X017 最大 12.2VDC	X000~X001 最大 10.6VDC，X002~X007 最大 12.2VDC	X000~X003 最大 10.6VDC，X004~X037 最大 12.2VDC	X000~X001 最大 10.6VDC，X002~X017 最大 12.2VDC	X000~X001 最大 10.6VDC，X002~X007 最大 12.2VDC
输入信号逻辑 1 限值	X000~X003 最小 11.5VDC，X004~X037 最小 15.6VDC	X000~X001 最小 11.5VDC，X002~X017 最小 15.6VDC	X000~X001 最小 11.5VDC，X002~X007 最小 15.6VDC	X000~X003 最小 11.5VDC，X004~X037 最小 15.6VDC	X000~X001 最小 11.5VDC，X002~X017 最小 15.6VDC	X000~X001 最小 11.5VDC，X002~X007 最小 15.6VDC
模拟输入	无					

高速计数	2 路分别最高 150KHz					
AB 相计数	1 路最高 150KHz					
输出类型	继电器			晶体管		
输出电压	＜250VAC，30VDC			6~30VDC		
最大感应负载	80VA			12W		
最大电阻负载	1.2A/1 点			0.5A/高速分别最高 150KHz（Y0/Y1/Y2/Y3）；0.2A/低速 200Hz (Y4~Y37)，共点不能超过 3A	0.5A/高速分别最高 150KHz（Y0/Y1）；0.2A/低速 200Hz (Y2~Y17)，共点不能超过 3A	0.5A/高速分别最高 150KHz（Y0/Y1）；0.2A/低速 200Hz (Y2~Y7)，共点不能超过 3A
数字输出	32	16	8	32	16	8
脉冲输出	ON/OFF 用，不适合作频繁交换用途			4 路分别最高 150KHz	2 路分别最高 150KHz	2 路分别最高 150KHz
模拟量输出	无					
扩展模块	8					
X 寄存器数量	128					
Y 寄存器数量	128					
电源功率	30W					
扩展模块负载	24VDC，200mA					
工作电压（外部供电电压）	AC 100V ~ 240V					
隔离保护	隔离 3600V					
容许瞬间断电时间	50ms					
24V 输出	隔离					
工作环境温度	密闭：5℃～40℃，开放：5℃～55℃					
保存环境温度	-25℃~70℃					
工作环境湿度	5～95%RH					
保存环境湿度	5～95%RH					
使用环境	无腐蚀性、可燃性气体，导电性尘埃不严重的地点					
抗电磁干扰	由电磁干扰电压 1000Vp-p、电磁干扰幅度 1μS、上升沿触发 1ns、周期 30-100Hz 的电磁干扰模拟器					
防震度	10-25HZ(XYZ 方向 2G/30 分钟）					
输入软件防抖功能	有					
输入设备连接错误检测功能	有					
掉电保存软件	有					

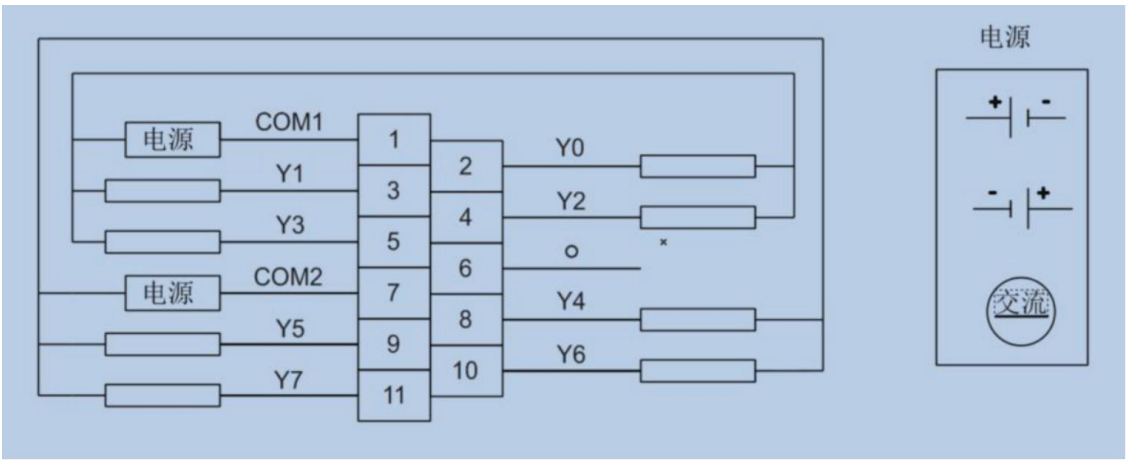
防抖功能	
RS232	1 个 mini232 接口 支持下载，监视，串口通信 波特率 1200Bps~115200 Bps（FGRs 系列波特率 2400Bps~115200 Bps）
RS485	1 个 支持串口通信 波特率 1200 Bps ~115200 Bps
RJ45	无
USB 设备口	支持下载，在线监视
通讯协议	支持 FGs 通信协议，Modbus RTU 主站，Modbus RTU 从站， 自由协议通信 Modbus 主站支持连接 128 个从站
电磁相容性(EMC) EN61000-6-4:2007+A1：2011 EN61000-6-2:2005 EN61000-3-2:2006+A1：2009+A2：2009 EN61000-3-3:2008	辐射噪音测试 传导噪音测试 抗静电能力测试
FCC Part 15 Subpart B(FCC) FCC:ANSI C63.4:2003	PASS
抗静电	+/-8KV

二、PLC 输出端信号接线示意图

晶体管输出（漏型）连接示例：

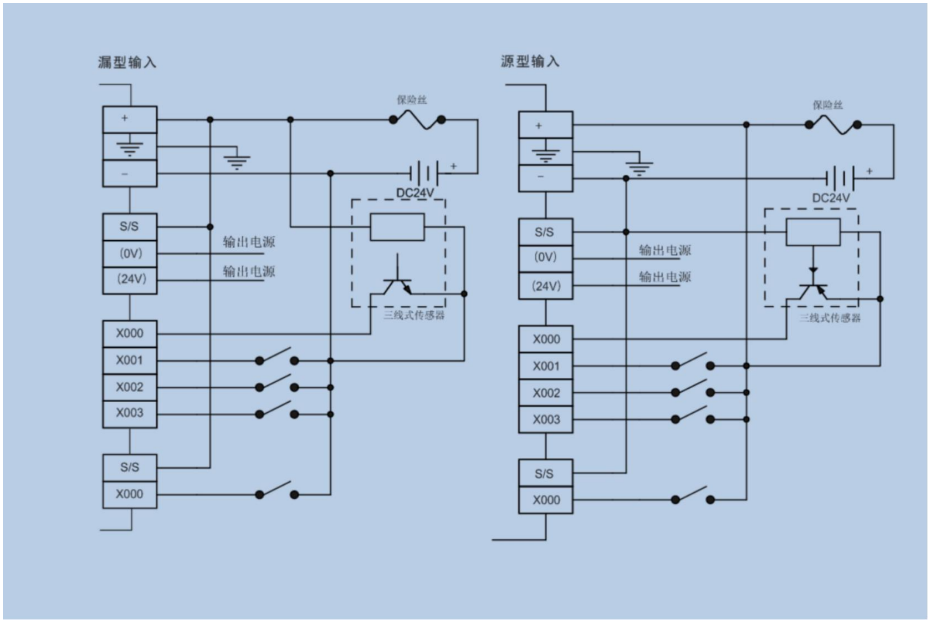


继电器输出连接示例：

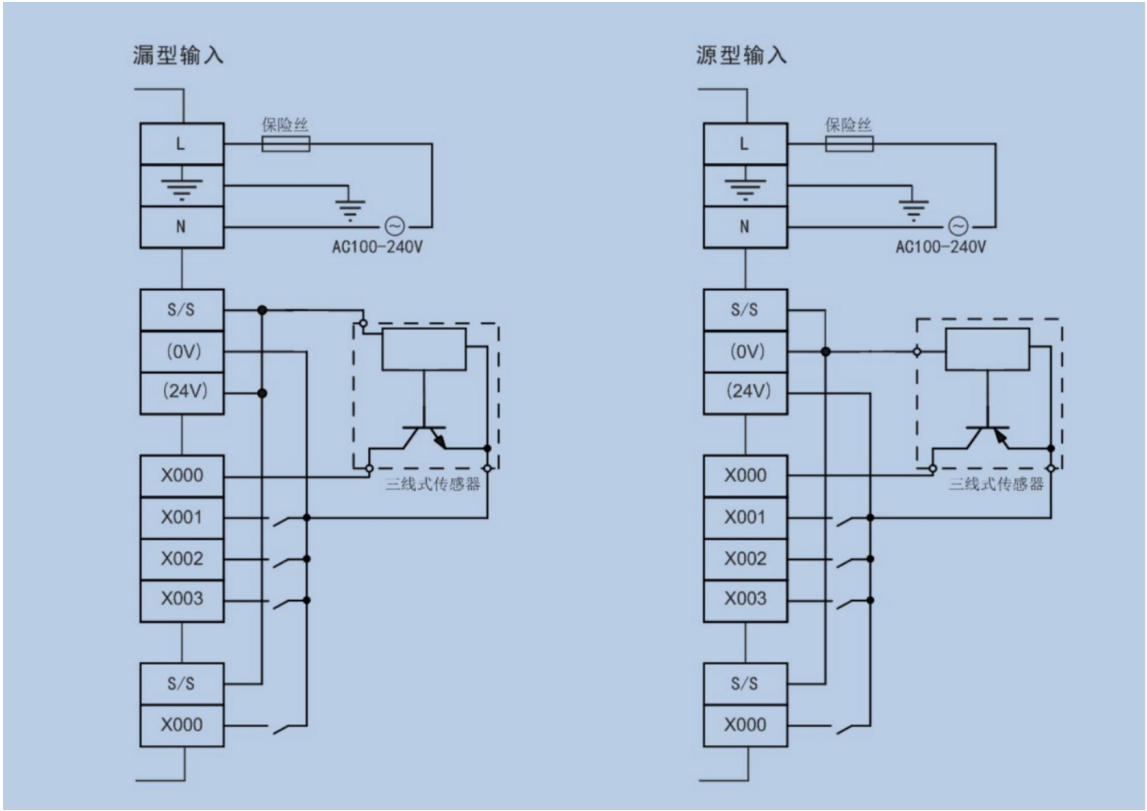


三、电源及输入端连接方式

DC24V 输入的连接示意图（DC 电源供电）



DC24V 输入的连接示意图（AC 电源供电）



13. 联系方式

深圳市显控科技股份有限公司

Shenzhen Samkoon Technology Corporation Ltd.

地址：深圳市南山区高新中区深圳软件园一期 2 栋 4 楼

Address：ShenZhen Software Park Building NO.2, 4th floor, the middle region of High-tech zone，NanShan district, ShenZhen

电话(TEL)：0755—29419028　　0755—29419038

0755—29419068　　0755—29419098

服务热线：400-606-9669

传真（FAX）：0755-29455559

E-Mail：samkoon@samkoon.com.cn

公司网址：Http://www.samkoon.com.cn