

APLC系列可编程通讯控制器

编程手册

(V1.0.1)



资料版本 V1.1

版权所有，保留一切权利。内容如有改动，恕不另行通知。

版本修改记录

V0.2	修改附录二第 13、14 小节	
V1.0.0	发行版本 V1.0	2023-7
V1.0.1	1.删除 AP40R 型号说明 2.增加网络配置说明	2024-12

目录

前 言	15
目标读者	15
手册内容	15
手册编排	15
阅读指南	15
1.不熟悉 PLC 的读者	15
2.熟悉 PLC 的读者	15
相关文档和参考书目	15
第一章 产品概述	17
1.1 产品简介	18
1.1.1 产品性能规格	18
1.1.2 APLC 系列主模块外形	20
1.1.3 开关量 DI 接线图	21
1.1.4 开关量 DO 接线图	22
1.2 YD-station 编程软件	23
1.2.1 基本配置	23
1.2.2 YD-station 编程软件安装过程	23
1.2.3 YD-station 运行界面	24
1.3 通讯功能	24
1.3.1 Modbus 通讯协议网络	24
1.3.2 自由口协议网络	25
1.4 APLC 系列小型 PLC 的手册资料	25
1.4.1 选型手册	25
1.4.2 编程手册	25
第二章 功能说明	26
2.1 编程资源和原理	27
2.1.1 编程资源	27
2.1.2 PLC 运行机制（扫描周期模型）	28
2.1.3 用户程序运行看门狗功能	29
2.1.4 恒定扫描运行模式	29
2.1.5 用户文件下载和存储	29
2.1.6 元件初始化	29
2.1.7 掉电保存数据功能	30
2.1.8 D 元件数据的永久保存	30
2.1.9 输入点的数字滤波功能	30
2.1.10 无电池模式	30
2.1.11 用户程序保护措施	30
2.2 系统配置	31

2.2.1 系统块	31
元件值保持	36
2.2.2 数据块	37
2.2.3 全局变量表	37
2.3 运行方式和状态控制	38
2.3.1 系统运行停止状态概念	38
2.3.2 运行停止状态转换	38
如何进入运行状态 (STOP→RUN)	38
1. 复位方式	38
2. 输入点开机模式	38
1. 指令控制方式	39
2. 错误停止方式	39
2.3.3 停止状态下输出点状态设定	39
2.4 系统调试	40
2.4.1 程序下载和上载	40
2.4.2 报错机制	40
2.4.3 在线修改	42
2.4.4 清除与格式化	42
2.4.5 PLC 信息在线查询	43
2.4.6 元件值写入和强制、元件监控表	43
2.4.7 从 RAM 中生成数据块	45
第三章 软元件和数据	47
3.1 软元件种类和作用	48
3.1.1 软元件概述	48
3.1.2 软元件一览表	49
3.1.3 输入点和输出点	50
3.1.4 辅助继电器	51
3.1.5 状态继电器	52
3.1.6 定时器	53
3.1.7 计数器	54
3.1.8 数据寄存器	55
3.1.9 特殊辅助继电器	55
3.1.10 特殊数据寄存器	56
3.1.11 变址寻址寄存器	57
3.1.12 局部辅助继电器	58
3.1.13 局部数据寄存器	58
3.2 软元件寻址方式	59
3.2.1 位串组合寻址方式 (Kn 寻址方式)	59
3.2.2 变址寻址方式 (Z 寻址方式)	60
3.2.3 位串组合的变址寻址方式	60
3.2.4 D、V 元件对 32 位数据的存储和寻址	60
D、V 元件地址寻址 32 位数据	61

3.3 数据	61
3.3.1 数据类型	61
3.3.2 元件与数据类型的匹配关系	61
3.3.3 常数	62
第四章 编程概念	63
4.1 编程语言介绍	64
4.1.1 梯形图 (LAD)	64
梯形图基本的编程元素	64
4.1.2 指令列表 (IL)	65
4.1.3 顺序功能图 (SFC)	66
4.2 程序要素	66
4.2.1 用户程序	66
4.2.2 系统块	67
4.2.3 数据块	67
4.3 程序的块注释和变量的注释	67
4.3.1 块注释	67
4.3.2 变量的注释	68
元件注释	69
4.4 子程序	70
4.4.1 子程序概念	70
4.4.2 子程序的使用注意事项	70
4.4.3 子程序变量表定义	70
4.4.4 子程序参数传递	71
4.4.5 子程序的使用示例	71
步骤 3: 编写主程序, 调用子程序	72
4.5 指令通用说明	73
4.5.1 指令的操作数	73
4.5.2 标志位	73
4.5.3 指令的使用限制	73
第五章 基本指令说明	74
5.1 触点逻辑指令	76
5.1.1 LD: 常开触点指令	76
5.1.2 LDI: 常闭触点指令	76
5.1.3 AND: 常开触点与指令	77
5.1.4 ANI: 常闭触点与指令	77
5.1.5 OR: 常开触点或指令	78
5.1.6 ORI: 常闭触点或指令	78
5.1.7 OUT: 线圈输出指令	79
5.1.8 ANB: 能流块与指令	79
5.1.9 ORB: 能流块或指令	80
5.1.10 MPS: 输出能流入栈指令	80

5.1.11 MRD: 读输出能流栈顶值指令	81
5.1.12 MPP: 输出能流栈出栈指令	81
5.1.13 EU: 上升沿检测指令	82
5.1.14 ED: 下降沿检测指令	82
5.1.15 INV: 能流取反指令	83
5.1.16 SET: 线圈置位指令	83
5.1.17 RST: 线圈清除指令	84
5.1.18 NOP: 空操作指令	84
5.2 主控指令	84
5.2.1 MC: 主控指令	84
5.2.2 MCR: 主控清除指令	85
5.3 SFC 指令	86
5.3.1 STL: SFC 状态装载指令	86
5.3.2 SET Sxx: SFC 状态转移	86
5.3.3 OUT Sxx: SFC 状态跳转	87
5.3.4 RST Sxx: SFC 状态清除	87
5.3.5 RET: SFC 程序段结束	87
5.4 计时器指令	88
5.4.1 TON: 接通延时计时指令	88
5.4.2 TONR: 记忆型接通延时计时指令	89
5.4.3 TOF: 断开延时计时器指令	90
5.4.4 TMON: 不重触发单稳计时指令	91
5.5 计数器指令	92
5.5.1 CTU: 16 位增计数器指令	92
5.5.2 CTR: 16 位循环计数指令	93
5.5.3 DCNT: 32 位增减计数指令	94
第六章 应用指令说明	95
6.1 程序流控制指令	100
6.1.1 FOR: 循环指令	100
6.1.2 NEXT: 循环返回	100
6.1.3 LBL: 跳转标号定义指令	102
6.1.4 CJ: 条件跳转指令	103
6.1.5 CFEND: 用户主程序条件返回	104
6.1.6 WDT: 用户程序看门狗清零	104
6.1.7 EI: 中断使能	104
6.1.8 DI: 中断禁止	105
6.1.9 CIRET: 用户中断程序条件返回	105
6.1.10 STOP: 用户程序停止	105
6.1.11 CALL: 用户子程序调用	105
6.1.12 CSRET: 用户子程序条件返回	106
6.2 数据传输指令	107
6.2.1 MOV: 字数据传输指令	107

使用示例	107
6.2.2 DMOV: 双字数据传输指令	108
使用示例	108
6.2.3 RMOV: 浮点数数据传输指令	109
6.2.4 BMOV: 块数据传输指令	109
6.2.5 FMOV: 数据块填充指令	110
6.2.6 DFMov: 数据块双字填充指令	111
6.2.7 SWAP: 高低字节交换指令	112
6.2.8 XCH: 字交换指令	112
6.2.9 DXCH: 双字交换指令	113
6.2.10 PUSH: 数据入栈指令	113
6.2.11 FIFO: 先入先出指令	114
注意事项	115
6.2.12 LIFO: 后入先出指令	115
注意事项	116
6.2.13	116
6.2.14 WSFR: 字串右移动指令	116
6.2.15 WSFL: 字串左移动指令	117
6.3 整数算术运算指令	119
6.3.1 ADD: 整数加法指令	119
使用示例	119
6.3.2 SUB: 整数减法指令	119
使用示例	120
6.3.3 MUL: 整数乘法指令	120
6.3.4 DIV: 整数除法指令	121
6.3.5 SQT: 整数算术平方根指令	121
注意事项	122
6.3.6 INC: 整数加一指令	122
6.3.7 DAT: 整数减一指令	123
6.3.8 VABS: 整数绝对值指令	123
6.3.9 NEG: 整数取负指令	124
6.3.10 DADD: 长整数加法指令	125
使用示例	125
6.3.11 DSUB: 长整数减法指令	125
使用示例	126
6.3.12 DMUL: 长整数乘法指令	126
6.3.13 DDIV: 长整数除法指令	127
6.3.14 DSQT: 长整数算术开方指令	127
注意事项	128
6.3.15 DINC: 长整数增一指令	128
使用示例	128
6.3.16 DDAT: 长整数减一指令	129

6.3.17 DVABS: 长整数绝对值指令	129
6.3.18 DNEG: 长整数取负指令	130
6.3.19 SUM: 整数累加指令	131
使用示例	131
6.3.20 DSUM: 长整数累加指令	131
6.4 浮点算术运算指令	132
6.4.1 RADD: 浮点数加法指令	133
6.4.2 RSUB: 浮点数减法指令	133
6.4.3 RMUL: 浮点数乘法指令	134
6.4.4 RDIV: 浮点数除法指令	135
6.4.5 RSQT: 浮点数算术开方指令	135
6.4.6 RVABS: 浮点数绝对值指令	136
6.4.7 RNEG: 浮点数取负指令	137
6.4.8 SIN: 浮点数 SIN 指令	137
6.4.9 COS: 浮点数 COS 指令	138
6.4.10 TAN: 浮点数 TAN 指令	139
使用示例	139
6.4.11 POWER: 浮点数求幂运算	139
使用示例	140
6.4.12 LN: 浮点数自然对数指令 LN	140
6.4.13 EXP: 浮点数自然数幂指令	141
6.4.14 RSUM: 浮点数累加指令	141
6.5 数值转换指令	143
6.5.1 DTI: 长整数转换整数指令	143
6.5.2 ITD: 整数转换长整数指令	143
6.5.3 FLT: 整数转换浮点数指令	144
6.5.4 DFLT: 长整数转换浮点数指令	145
6.5.5 INT: 浮点数转换整数指令	145
注意事项	146
6.5.6 DINT: 浮点数转换长整数指令	146
注意事项	146
6.5.7 BCD: 字转换 16 位 BCD 码指令	147
6.5.8 DBCD: 双字转换 32 位 BCD 码指令	147
6.5.9 BIN: 16 位 BCD 码转换字指令	148
6.5.10 DBIN: 32 位 BCD 码转换双字指令	149
6.5.11 GRY: 字转换 16 位格雷码指令	149
6.5.12 DGRY: 双字转换 32 位格雷码指令	150
6.5.13 GBIN: 16 位格雷码转换字指令	151
6.5.14 DGBIN: 32 位格雷码转换双字指令	151
6.5.15 SEG: 字转换 7 段码指令	152
6.5.16 ASC: ASCII 码转换指令	152
6.5.17 ITA: 16 位 16 进制数转换 ASCII 码指令	153

使用示例	154
6.5.18 ATI: ASCII 码数转换 16 位 16 进制指令	154
使用示例	154
6.6 字逻辑运算	155
6.6.1 WAND: 字与指令	155
6.6.2 WOR: 字或指令	155
6.6.3 WXOR: 字异或运算	156
6.6.4 WINV: 字取反运算	156
6.6.5 DWAND: 双字与指令	157
6.6.6 DWOR: 双字或指令	158
6.6.7 DWXOR: 双字异或指令	158
6.6.8 DWINV: 双字取反指令	159
操作数说明	159
6.7 位移动旋转指令	160
6.7.1 ROR: 16 位循环右移指令	160
6.7.2 ROL: 16 位循环左移指令	161
6.7.3 RCR: 16 位带进位循环右移指令	162
6.7.4 RCL: 16 位带进位循环左移指令	163
6.7.5 DROR: 32 位循环右移指令	163
6.7.6 DROL: 32 位循环左移指令	164
6.7.7 DRCR: 32 位带进位循环右移指令	165
6.7.8 DRCL: 32 位带进位循环左移指令	166
6.7.9 SHR: 16 位右移指令	166
6.7.10 SHL: 16 位左移指令	167
6.7.11 DSHR: 32 位右移指令	168
6.7.12 DSHL: 32 位左移指令	169
6.7.13 SFTR: 位串右移指令	169
6.7.14 SFTL: 位串左移指令	170
操作数说明	170
6.8 外设指令	172
6.8.1 FROM: 特殊模块缓冲寄存器字读指令	172
6.8.2 DFROM: 特殊模块缓冲寄存器双字读指令	172
6.8.3 TO: 特殊模块缓冲寄存器字写指令	172
6.8.4 DTO: 特殊模块缓冲寄存器双字写指令	172
6.8.5 VRRD: 读模拟电位器值指令	172
6.8.6 REFF: 设置输入滤波常数指令	173
6.8.7 REF: I/O 立即刷新指令	173
使用示例	174
6.8.8 EROMWR: EEPROM 写指令	174
使用示例	175
6.8.9 PR: 打印指令	175
6.9 实时时钟指令	176

6.9.1 TRD: 实时时钟读指令	176
6.9.2 TWR: 实时时钟写指令	177
6.9.3 TADD: 时钟加指令	178
6.9.4 TSUB: 时钟减指令	179
6.9.5 HOUR: 计时表指令	181
使用示例	181
6.9.6 DCMP: (=、<、>、◇、>=、<=) 日期比较指令	182
6.9.7 TCMP: (=、<、>、◇、>=、<=) 时间比较指令	183
操作数说明	183
6.10 高速 IO 指令	185
6.10.1 HCNT: 高速计数器驱动指令	185
6.10.2 DHSCS: 高速计数比较置位指令	186
6.10.3 DHSCI: 高速计数比较中断触发指令	188
6.10.4 DHSCR: 高速计数比较复位指令	189
6.10.5 DHSZ: 高速计数区间比较指令	190
6.10.6 DHST: 高速计数表格比较指令	192
6.10.7 DHSP: 高速计数表格比较脉冲输出指令	193
6.10.8 SPD: 测频指令	196
6.10.9 PLSY: 计数脉冲输出指令	197
使用示例	198
6.10.10 PLSR: 带加减速的计数脉冲输出指令	199
6.10.11 PLS: 包络线脉冲输出指令	201
6.10.12 PLSB: 带基底频率与加减速的计数脉冲输出指令	203
6.10.13 PWM: 脉冲输出指令	205
操作数说明	205
6.11 控制计算指令	207
6.11.1 PID: 功能指令	207
6.11.2 RAMP: 斜坡信号输出指令	211
6.11.3 HACKLE: 锯齿波信号输出指令	213
6.11.4 TRIANGLE: 三角波信号输出指令	215
6.11.5 ABSD: 凸轮绝对控制指令	217
6.11.6 DABSD: 凸轮绝对控制指令	217
6.11.7 ALT: 交替输出指令	217
6.12 通讯指令	218
6.12.1 Modbus: 主站通讯指令	218
6.12.2 EVFWD: 变频器正转指令	220
6.12.3 EVREV: 变频器反转指令	221
6.12.4 EVDFWD: 变频器点动正转指令	221
6.12.5 EVDREV: 变频器点动反转指令	222
6.12.6 EVSTOP: 变频器停止指令	223
6.12.7 EVFRQ: 设置变频器频率指令	223
6.12.8 EVWRT: 写单个寄存器值指令	224

6.12.9 EVRDST: 读取变频器状态指令	225
6.12.10 EVRD: 读取变频器单个寄存器值指令	226
6.12.11 XMT: 自由口发送指令	227
使用示例	227
6.12.12 RCV: 自由口接收指令	228
特殊寄存器	229
6.12.13 MODRW: MODBUS 读写指令	229
6.13 校验指令	230
6.13.1 CCITT: 校验指令	230
注意事项	230
6.13.2 CRC16: 校验指令	231
注意事项	231
6.13.3 LRC: 校验指令	232
操作数说明	232
6.14 增强型位处理指令	233
6.14.1 ZRST: 批量位清零指令	233
6.14.2 ZSET: 批量位置位指令	233
6.14.3 DECO: 解码指令	234
使用示例	234
6.14.4 ENCO: 编码指令	234
6.14.5 BITS: 字中 ON 位统计指令	235
6.14.6 DBITS: 双字中 ON 位统计指令	236
操作数说明	236
6.15 字触点指令	236
6.15.1 BLD: 字位触点 LD 指令	236
6.15.2 BLDI: 字位触点 LDI 指令	237
6.15.3 BAND: 字位触点 AND 指令	237
6.15.4 BANI: 字位触点 ANI 指令	238
6.15.5 BOR: 字位触点 OR 指令	238
6.15.6 BORI: 字位触点 ORI 指令	239
6.15.7 BOUT: 字位线圈输出指令	240
6.15.8 BSET: 字位线圈置位指令	240
6.15.9 BRST: 字位线圈清除指令	241
操作数说明	241
6.16 比较触点指令	241
6.16.1 LD (=, <, >, <>, >=, <=): 整数比较 LD※指令	241
6.16.2 AND (=, <, >, <>, >=, <=): 整数比较 AND※指令	242
6.16.3 OR (=, <, >, <>, >=, <=): 整数比较 OR※指令	243
6.16.4 LDD (=, <, >, <>, >=, <=): 长整数比较 LDD※指令	244
6.16.5 ANDD (=, <, >, <>, >=, <=): 长整数比较 ANDD※指令	245
6.16.6 ORD (=, <, >, <>, >=, <=): 长整数比较 ORD※指令	246
6.16.7 LDR: 浮点数比较指令	247

6.16.8 ANDR: 浮点数比较指令	248
6.16.9 ORR: 浮点数比较指令	249
6.16.10 CMP: 整数比较置位指令	250
6.16.11 LCMP: 长整数比较置位指令	250
6.16.12 RCMP: 浮点数比较置位指令	250
第七章 顺序功能图使用指南	251
7.1 顺序功能图介绍	252
7.1.1 什么是顺序功能图	252
7.1.2 什么是 APLC 系列 PLC 的顺序功能图	252
7.1.3 顺序功能图的基本概念	252
转移	252
7.1.4 编程图元及其连接规则	253
7.1.5 顺序功能图结构	254
7.1.6 顺序功能图程序的执行	258
7.2 顺序功能图与梯形图的对应关系	259
7.2.1 STL 指令与步进状态	259
7.2.2 SFC 状态转移指令	259
7.2.3 RET 指令与 SFC 程序段	259
7.2.4 SFC 状态跳转指令、重置指令	260
7.2.5 SFC 选择分支、并行分支及汇合	260
7.3 顺序功能图编程步骤	260
1. 分析流程, 确定程序流程结构	260
2. 确定主要步骤和主要转移条件, 得出流程草图	260
3. 根据流程草图做出 SFC 顺序功能图	260
4. 做出输入输出点表, 确定各步骤操作对象及实际转移条件	260
5. 步骤和转移条件的输入	260
6. 编写梯形图程序块	260
7.4 顺序功能图编程注意事项	261
7.4.1 常见的编程错误	261
7.4.2 编程技巧	263
7.5 顺序功能图编程实例	265
7.5.1 简单结构流程	265
7.5.2 选择结构	267
7.5.3 并行结构	271
第八章 高速输入输出	276
第九章 中断程序使用指南	277
9.1 理解中断程序	278
9.2 中断事件的处理过程	279
2. 系统处理中断请求:	279
2. 中断程序中禁止调用其他用户子程序。	279
9.3 使用定时中断	279

定时中断的说明	279
9.4 使用外部中断	282
外部中断说明	282
9.5 使用电源失电中断	284
9.6 使用串口中断	284
串口中断说明	284
第十章 通讯功能使用指南	286
10.1 通讯资源	287
10.2 编程口通讯协议	287
10.3 自由口通讯协议	287
10.3.1 简介	287
10.3.2 自由口参数设置	287
10.3.3 自由端口指令	289
注意事项	289
10.4 Modbus 通讯协议	291
10.4.1 简介	291
10.4.2 链路特性	291
10.4.3 RTU 传输模式	291
10.4.4 ASCII 传输模式	291
10.4.5 支持的 Modbus 功能码	292
10.4.6 PLC 元件的寻址方式	292
10.4.7 Modbus 从站	293
10.4.8 元件的读写	293
10.4.9 对双字元件的处理	294
10.4.10 对 LONG INT 的处理	295
10.4.11 诊断功能码	295
10.4.12 异常码	295
10.4.13 Modbus 参数设置	296
10.4.14 Modbus 指令	296
10.5 ModbusTCP 通讯协议	297
10.5.1 简介	297
10.5.2 网络设置	297
附录一 特殊辅助继电器	298
1. PLC 工作状态标志	298
2. 时钟运行位	298
3. 用户程序执行错误	299
4. 中断控制	299
5. 外设指令	299
6. 高速脉冲输出控制	300
7. 脉冲捕捉监视位	300
8. 自由端口 (PORT 0)	300

9. 自由端口 (PORT 1)	301
10. 扩展自由端口 (PORT 2)	301
11. Modbus 通讯	302
12. 集成模拟量通道的使能标志	302
13. 运算标志位	302
14. ASCII 码转换指令标志	303
15. 系统总线错误标志	303
16. 实时时钟错误标志	303
17. EEPROM 标志	303
18. 增/减型计数器计数方向	304
19. 高速计数器的计数方向及监控	305
20. 增强定位	306
附录二 特殊数据寄存器	307
1. PLC 工作状态数据	307
2. 运行错误代码 FIFO 区	307
3. FROM/TO 错误	307
4. 扫描时间	308
5. 输入滤波常数设置	308
6. 高速脉冲输出监控	308
7. 定时中断周期	308
8. 定位指令	309
9. 实时时钟	309
10. 自由端口接收控制及状态 (通讯口 1)	310
11. Modbus (PORT1) 设定	311
12. 扩展自由端口接收控制及状态 (通讯口 2)	312
13. Modbus (PORT2) 设定	313
14. 集成模拟量通道的设定与读取	313
15. DHSP 和 DHST 指令使用	313
16. 发生错误标志	313
17. 增强定位指令	314
附录三 保留元件表	315
附录四 Modbus 通讯错误代码	316
附录五 变频器指令的错误代码	317
附录六 系统错误代码表	318
附录七 Modbus 通讯协议	320
1. Modbus 通讯协议概述	320
2. 支持的 Modbus 功能码及元件寻址方式	320
3. Modbus 功能码叙述	322
3.1 读取线圈状态 (0x01)	322
在 APLC 系列 PLC 中读取位元件, 读取的最大个数是 256 个。	322

3.2 读取离散量输入状态 (0x02)	322
3.3 读取保持寄存器 (0x03)	322
3.4 强置 (写) 单线圈 (0x05)	323
1. 请求帧.....	323
2. 响应帧.....	323
3.5 预置 (写) 单寄存器 (0x06)	323
3.6 回送诊断校验 (0x08)	323
请求帧返回 (0x00) :	324
1. 请求帧.....	324
2. 响应帧.....	324
从站进入 LISTEN ONLY 模式 (0x04) :	324
3.7 强置 (写) 多线圈(0x0F).....	326
3.8 预置 (写) 多寄存器(0x10 Hex).....	327
3.9 故障响应帧 (0x80+功能码)	327
功能码是截获所请求帧的功能码 + 0x80	327
3.10 注意事项.....	327
2. 读取该类软元件的地址和数据范围不能超过协议中规定的范围。举例如下:	327
4. Modbus 通讯控制举例	327
1. 主站发送: 01 01 00 00 01 00 3D 9A.....	328
2. 主站发送: 01 01 00 00 01 01 FC 5A.....	328
3. 主站发送: 01 01 00 64 00 A0 7D AD.....	328
4. 主站发送: 01 01 01 2C 00 0A 7C 38.....	328
5. 主站发送: 01 04 00 02 00 0A D1 CD.....	328
6. 主站发送: 01 02 00 00 00 0A F8 0D.....	328
7. 主站发送: 01 01 04 B0 00 0A BC DA.....	328
2. X 元件不支持写入。	328
5. 对广播的描述	329
附录八 ASCII 码字符编码表	330
附录九 指令排序索引表	331
附录十 指令分类索引表	338

前 言

目标读者

本书适用于自动化技术人员，帮助他们掌握可编程控制器（简称 PLC）的编程、系统设计和调试技术；为初步及深入学习 PLC 编程知识的人员提供了参考。

手册内容

本手册详细地描述了 APLC 系列 PLC 的编程原理、软硬件编程资源、支持的编程语言和详细的指令说明，以及高速输入输出、通讯等技术参考内容；同时介绍了各功能的使用方法并穿插了应用实例。

手册编排

本手册的章节编排由整体到细节，各章节具有独立的内容，可以从头通读逐步掌握 APLC 系列 PLC 的全面内容，也可以随时查阅其中的章节，作为一些技术参考资料。

阅读指南

1. 不熟悉 PLC 的读者

对于初步接触 PLC 的读者，建议先阅读第一章到四章。这些章节讲解了 PLC 基础知识，包括 PLC 功能说明、编程语言、程序要素、数据类型、寻址方式、软元件定义、程序注释功能与编程、主程序及子程序使用等等。

2. 熟悉 PLC 的读者

对于已经熟悉 PLC 基本概念和编程工具的读者，可以直接阅读本书的第五章 基本指令说明和第六章 应用指令说明。这两章提供了完整的 APLC 系列 PLC 指令说明。如果了解顺序功能图、高速 IO、中断以及通讯功能的使用方法，请参阅第七到第十章。同时为了方便读者，本书的附录九 指令排序索引表和附录十 指令分类索引表还分别根据指令分类和指令英文名的字母顺序为读者提供了查找相关指令说明的工具。

相关文档和参考书目

阅读本手册的同时也可以参考阅读以下手册：

- 《APLC 系列可编程控制器用户手册》
- 《YD-station 编程软件用户手册》

目 录

第一章 产品概述	17
第二章 功能说明	26
第三章 软元件和数据	47
第四章 编程概念	63
第五章 基本指令说明	74
第六章 应用指令说明	95
第七章 顺序功能图使用指南	251
第八章 高速输入输出	276
第九章 中断程序使用指南	277
第十章 通讯功能使用指南	286
附录一 特殊辅助继电器	298
附录二 特殊数据寄存器	307
附录三 保留元件表	314
附录四 Modbus 通讯错误代码	316
附录五 变频器指令的错误代码	317
附录六 系统错误代码表	318
附录七 Modbus 通讯协议 (AT100 系列)	320
附录八 ASCII 码字符编码表	330
附录九 指令排序索引表	331
附录十 指令分类索引表	338

第一章 产品概述

本章简要介绍了 APLC 系列 PLC 的产品构成、编程软件平台、网络组态应用。

1.1 产品简介	18
1.1.1 产品性能规格	18
1.1.2 APLC 系列主模块外形	20
1.1.3 开关量 DI 接线图	21
1.1.4 开关量 DO 接线图	22
1.2 YD-station 编程软件	23
1.2.1 基本配置	23
1.2.2 YD-station 编程软件安装过程	23
1.2.3 YD-station 运行界面	24
1.3 通讯功能	24
1.3.1 Modbus 通讯协议网络	24
1.3.2 自由口协议网络	25
1.4 APLC 系列小型 PLC 的手册资料	25
1.4.1 选型手册	25
1.4.2 编程手册	25

1.1 产品简介

APLC 系列小型 PLC 产品包括 APPLC-TCP 小型系列，是适应各种现代工业控制应用的高性能产品。

APLC 系列的产品都是一体式结构的 PLC，拥有内置的高性能微处理器和核心运算控制系统，集成了输入点和输出点；主模块集成了 2 个 RS485 通讯口，1 个 RJ45 通讯口；拥有丰富的内置编程资源，采用三种标准化编程语言，通过性能强大的 YD-station 编程软件可实现调试监控手段；具有完善的用户程序安全保护机制。

1.1.1 产品性能规格

名称		APLC-TCP
I/O	数字 I/O 点	10 输入/6 输出
	逻辑最大 I/O 点	16
	最大特殊模块数	无
	高速脉冲输出	无
	单相计数通道	无
	双相计数通道	无
	高速计数器频率总和最大值	无
	数字滤波功能	X0~X7 采用数字滤波, 输入滤波常数可选 0、8、16、32、64ms, 共 5 档
	晶体管输出点 最大电流	电阻负载 Y0、Y1: 0.3A/1 点 其他: 0.3A/1 点; 0.8A/4 点; 1.6A/8 点 8 点以上每增加 1 点允许总电流增加 0.1A
		感性负载 Y0、Y1: 7.2W/24Vdc; 其他: 12W/24Vdc
		电灯负载 Y0、Y1: 0.9W/24Vdc; 其他: 1.5W/24Vdc
存储器	用户程序	12k 步 (24kByte)
	程序掉电永久保持	是
	元件掉电保持最大数量	位元件 320 个, 字元件 180 个
	硬件支持以及保持时间	EEPROM, 永久保存
软元件资源	定时器	100ms 精度: T0~T209 10ms 精度: T210~T251 1ms 精度: T252~T255
	计数器	16 位增计数器: C0~C199 32 位增减计数器: C200~C235 32 位高速计数器: C236~C255
	数据寄存器	D0~D7999
	局部数据寄存器	V0~V63
	变址寻址寄存器	Z0~Z15
	特殊数据寄存器	SD0~SD255
	辅助继电器	M0~M2047
	局部辅助继电器	LM0~LM63
	特殊辅助继电器	SM0~SM255
	状态继电器	S0~S1023
中断资源	内部定时中断	3 个
	外部中断	10 个
	高速计数器中断	不支持
	串口中断	8 个
	PTO 输出完成中断	不支持
	电源失电中断	不支持
常规	基本指令运行时间	0.3μs
	实时时钟	不支持
	模拟电位器	不支持
通讯	通讯口	PORT0: USB, 程序下载口 PORT1: RS485 PORT2: RS485
	通讯协议	Modbus/自由口/编程口协议
访问控制和用户程序保护措施	设置密码类型	上载密码/下载密码/监控密码/子程序密码/禁止格式化
	禁止上载	支持
应用指令	实时时钟, 时钟指令	不支持
	日期和时钟比较指令	不支持
	浮点数指令	有
	定位指令	不支持
	高速 IO 指令	不支持
	MODBUS 和变频器指令	有
	读写 EEPROM 指令	有
电源特性	控制计算指令	有
	输入电压范围	24VDC

注: 1. APLC 系列 PLC 详细的产品规格、安装说明、运行保养指引, 请参阅《APLC 系列可编程控制器用户手册》。

表 1-1 PLC 主模块性能规格

1.1.2 APLC 系列主模块外形

APLC 系列主模块的外形结构如下图所示（以 APLC-TCP 为例）。



图 1-1 APLC 系列主模块外形结构

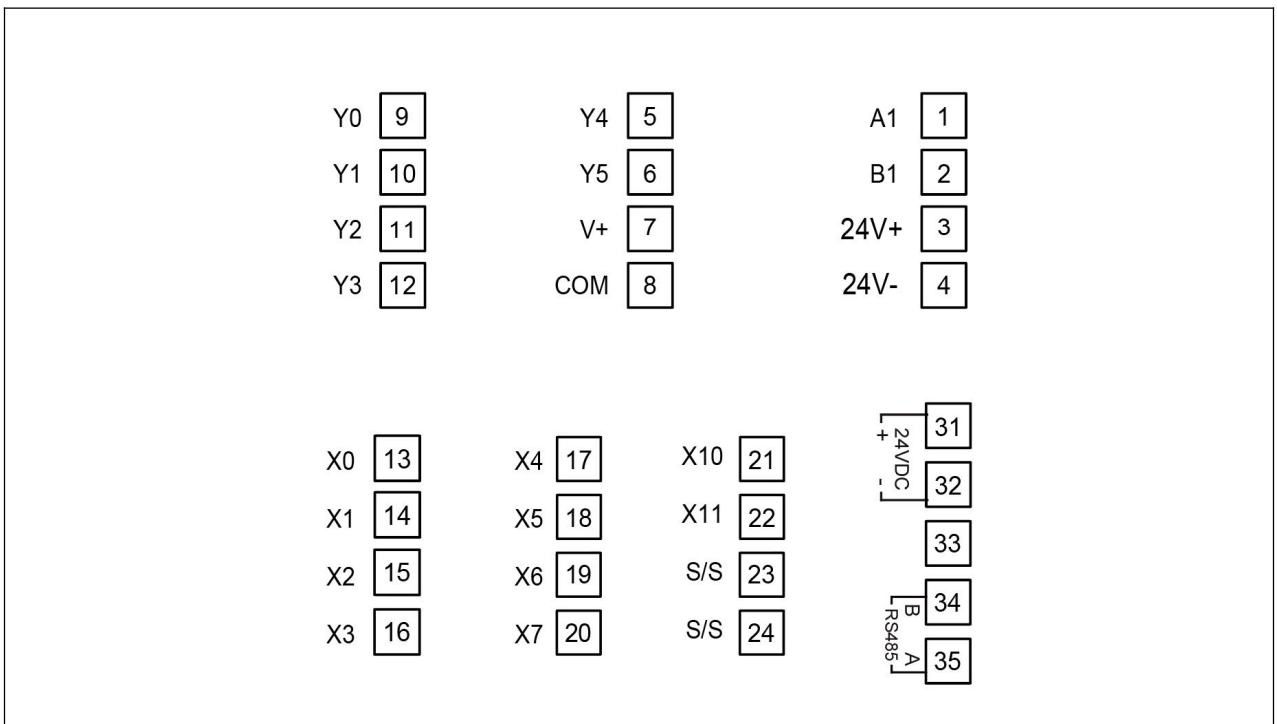


图 1-2 APLC 接线图

PORT1 和 PORT2 为通讯口，提供 RS485 电平。

端口号	名称	说明
1	A1	PORT2 通讯口 A
2	B1	PORT2 通讯口 B
3	24V+	同端口 31
4	24V-	同端口 32
5~6,	Y4,Y5	隔离型 NPN 输出，可外接 5~24VDC 驱动固态继电器或中间继电器
7	V+	输出端口 Y0~Y5 正端，输入范围为 5~24VDC
8	COM	输出端口公共端，接 0V
9~12	Y0~Y3	同端口 5~6
13~22	X0~X11	隔离型光耦输入端口
23	S/S	输入公共端，接 5~24V=，或 0V
24	S/S	同端口 23
31	24V+	输入电源正端
32	24V-	输入电源负端
33	保留	
34	A	PORT1 通讯口 A
35	B	PORT1 通讯口 B

表 1-2 端口

1.1.3 开关量 DI 接线图

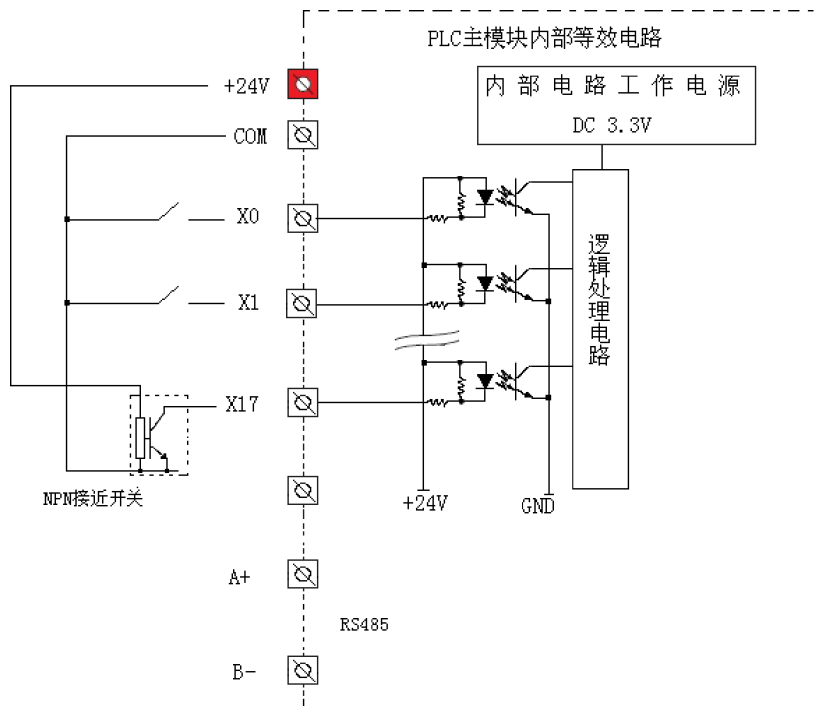


图 1-3 开关量 DI 接线图

1.1.4 开关量 DO 接线图

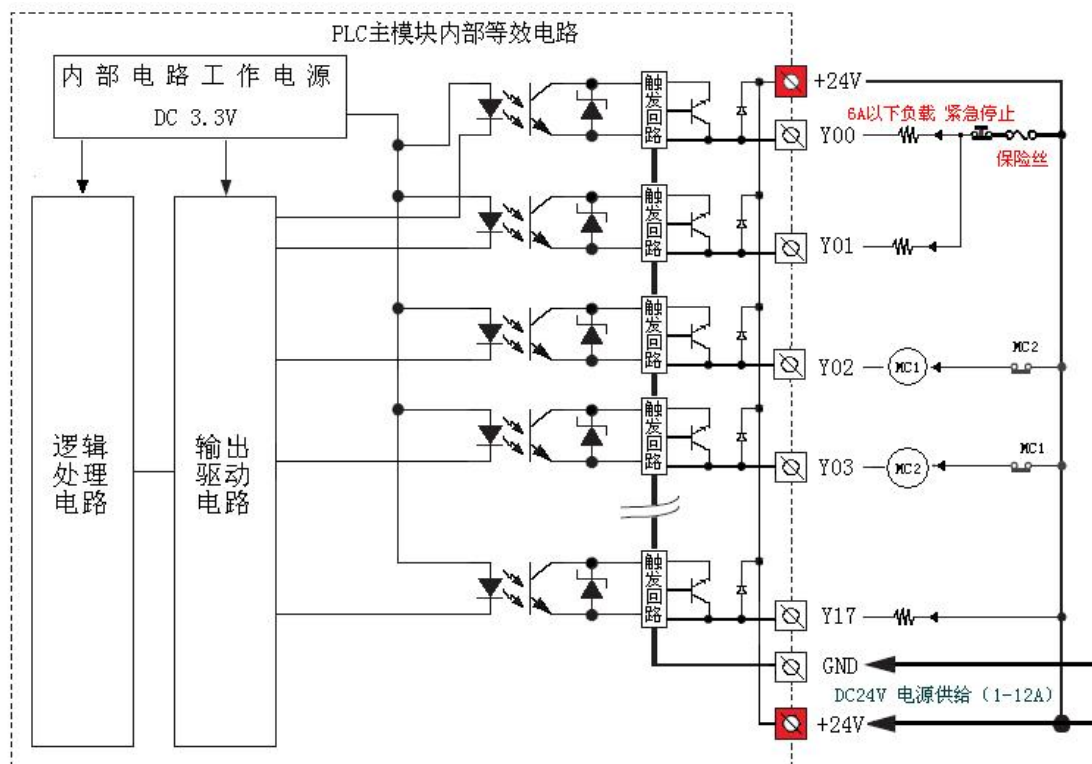


图 1-4 开关量 DO 接线图

1.2 YD-station 编程软件

YD-station 是 APLC 系列与 APLC 系列 PLC 产品的专用编程软件。软件可在公司网站下载。

YD-station 编程软件是标准 Windows 软件、图形化的 PLC 编程工具，采用鼠标键盘操作。编程时可选用 3 种标准语言：梯形图、语句表、SFC 顺序功能图。

YD-station 编程软件与 PLC 的连接采用串口编程电缆，通过串行口转换还可实现 Modbus 网络编程，通过 Modem 还可实现远程编程。有关 Modbus 编程和远程监控的内容请查阅《YD-station 编程软件用户手册》的相关内容。

1.2.1 基本配置

YD-station 编程软件需要安装在 Microsoft Windows 系列操作系统之上，兼容 Windows 7 以上操作系统。

推荐配置如下：

- 内存：4GB 或以上；
- 空间：5GB 可用硬盘空间或以上。

其他设备：

- 连接电缆：USB 电缆（Mini-B 接口）。

1.2.2 YD-station 编程软件安装过程

宇电公司发布的 YD-station 安装包为单独的可执行程序，双击即可启动安装过程，根据安装向导一步步进行安装。用户根据自己的需要可以选择不同的安装路径。

安装完毕后，会把 YD-station 快捷方式图标安装在桌面上，双击快捷方式图标可运行程序。

卸载操作：通过 Windows 控制面板可进行软件卸载。若要升级安装新版本的 YD-station 软件，请先将旧版本 YD-station 软件卸载。

1.2.3 YD-station 运行界面

本程序的主界面基本包括七个部分：菜单、工具栏、工程管理器窗口、指令树窗口、消息窗口、状态栏和工作区。

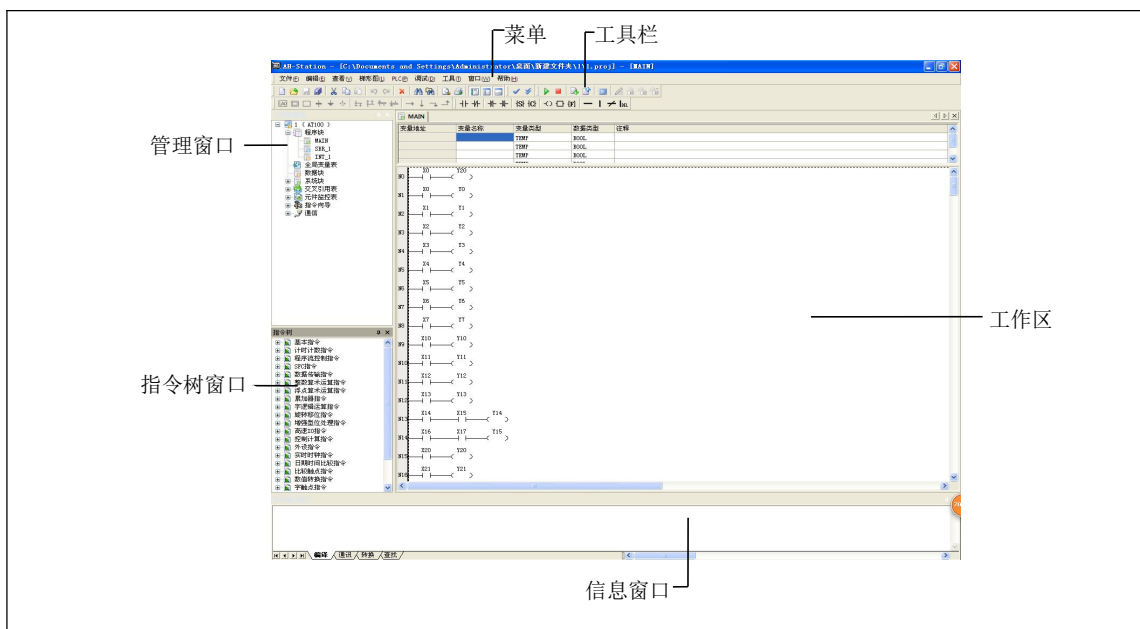


图 1-5 YD-station 主界面

请参阅《YD-station 编程软件用户手册》，该手册详细介绍 YD-station 编程软件使用方法。

1.3 通讯功能

APLC 系列小型 PLC 主模块拥有 2 个集成的串口 PORT 1、PORT 2，1 个以太网口，可以应用于现场总线网络通讯。主模块的 2 个串口具有 Modbus 通讯协议，还可以采用用户定义的自由口协议。主模块的以太网口支持 ModbusTCP 协议（从站）。

1.3.1 Modbus 通讯协议网络

利用 PORT 1 或 PORT 2 提供的 RS485 接口，可与多台变频器、PLC、其它智能仪表设备等组成 RS485 的 Modbus 网络。通讯距离最远达到 1200 米，通讯波特率最高达到 38400bit/s。可选择 RTU 或 ASCII 传输模式。

有关 Modbus 网络通讯详细内容，请参阅第十章 通讯功能使用指南和附录七 Modbus 通讯协议（APLC 系列）。

1.3.2 自由口协议网络

自由口协议是以用户自定义的通讯数据格式来进行通讯的协议模式，支持 ASCII 和二进制两种数据格式。在这种通讯口模式下，PLC 可以用来与各种自定义格式的设备通讯，如采用自由通讯协议的变频器、条码扫描仪、仪表、其他智能设备等。对单台设备，PLC 可通过 RS485 方式通讯；对多台设备也能组成 RS485 网络。

有关自由口协议通讯的详细内容，请参阅第十章 通讯功能使用指南。

1.4 APLC 系列小型 PLC 的手册资料

APLC 系列小型 PLC 的手册资料可以从公司网站下载。如果需要印刷版本请与代理商联系。

1.4.1 选型手册

《PLC 宣传彩页》

1.4.2 编程手册

《APLC 系列小型可编程控制器编程手册》

《YD-station 编程软件用户手册》

第二章 功能说明

本章简要介绍了 APLC 系列 PLC 的编程资源和原理、系统配置方法，以及如何设置 PLC 运行和操作模式，最后还介绍了有关系统调试的各项功能和软件工具。详细内容，请参见《YD-station 编程软件》。

2.1 编程资源和原理	27
2.1.1 编程资源	27
2.1.2 PLC 运行机制（扫描周期模型）	28
2.1.3 用户程序运行看门狗功能	29
2.1.4 恒定扫描运行模式	29
2.1.5 用户文件下载和存储	29
2.1.6 元件初始化	29
2.1.7 掉电保存数据功能	30
2.1.8 D 元件数据的永久保存	30
2.1.9 输入点的数字滤波功能	30
2.1.10 无电池模式	30
2.1.11 用户程序保护措施	30
2.2 系统配置	31
2.2.1 系统块	31
2.2.2 数据块	37
2.2.3 全局变量表	37
2.3 运行方式和状态控制	38
2.3.1 系统运行停止状态概念	38
2.3.2 运行停止状态转换	38
2.3.3 停止状态下输出点状态设定	39
2.4 系统调试	40
2.4.1 程序下载和上载	40
2.4.2 报错机制	40
2.4.3 在线修改	42
2.4.4 清除与格式化	42
2.4.5 PLC 信息在线查询	43
2.4.6 元件值写入和强制、元件监控表	43
2.4.7 从 RAM 中生成数据块	45

2.1 编程资源和原理

2.1.1 编程资源

名称		指标及描述	
I/O 配置	最大 I/O 点数	128 点（理论值）	
	扩展模块数量	不支持	
用户文件容量	用户程序容量	12k 步	
	数据块大小	8000 个 D 元件	
指令速度	基本指令	0.3μs/指令	
	应用指令	几μs~几百μs /指令	
指令个数	基本指令	32	
	应用指令	226	
软元件资源 ^{注 7}	输入输出点	128 入/128 出（输入 X0~X177，输出 Y0~Y177，APLC-TCP 只支持 10 输入 6 输出） ^{注 1}	
	辅助继电器	2048 点（M0~M2047）	
	局部辅助继电器	64 点（LM0~LM63）	
	特殊辅助继电器	256 点（SM0~SM255）	
	状态继电器	1024 点（S0~S1023）	
	定时器	256 个（T0~T255） ^{注 2}	
	计数器	256 个（C0~C255） ^{注 3}	
	数据寄存器	8000 个（D0~D7999）	
	局部数据寄存器	64 个（V0~V63）	
	变址寻址寄存器	16 个（Z0~Z15）	
	特殊数据寄存器	256 个（SD0~SD255）	
	中断资源	外部输入中断	16 个（中断触发边沿可由用户设定，对应于 X0~X7 端子上升和下降沿）
高速计数器中断		不支持	
内部定时中断		3 个	
串口中断		8 个	
PTO 输出完成中断		不支持	
电源失电中断		不支持	
通讯功能	串行通讯口	通讯口 0：编程口；通讯口 1：RS485；通讯口 2：RS485	
	串行通讯协议	通讯口 1、通讯口 2：Modbus 通讯协议、自由口协议	
	以太网通讯口	通讯口 3：RJ45	
	以太网通讯协议	ModbusTCP 通讯协议（从站）	
特殊功能	高速计数器	不支持	
		不支持	
	高速脉冲输出	不支持	
	数字滤波功能	X0~X7 采用数字滤波，其他端口采用硬件滤波	
	模拟电位器 ^{注 4}	不支持	
	子程序调用	最多允许 64 个用户子程序，允许 6 级子程序嵌套。支持局部变量，每个子程序最多可提供 16 个参数传递，支持变量别名	
特殊功能	用户程序保护措施	上载密码	提供 3 种形式的密码，密码不超过 8 个字符，每个字符为字母数字组合，区分大小写
		下载密码	
		监控密码	
		子程序加密	
		其他保护措施	提供禁止格式化、禁止上载的功能
	编程方式 ^{注 5}	YD-station 编程软件 ^{注 6}	需在 PC 中安装运行
	实时时钟	不支持	

表 2-1 APLC 编程资源

注释：

注 1：X、Y 元件的地址编号是采用 8 进制编址，例如：地址 X10 代表第 8 个输入点。

注 2：T 元件地址按计时精度划分三类：

- (1) 100ms 精度 T0~T209
- (2) 10ms 精度 T210~T251
- (3) 1ms 精度 T252~T255

注 3：C 元件地址按计数值的宽度和功能划分为三类：

- (1) 16 位增计数器 C0~C199
- (2) 32 位增减计数器 C200~C235
- (3) 32 位高速计数器 C236~C255

注 4：模拟电位器是提供给用户设定内部软元件值的外部通道，可用于设定 0~255 范围的数值，供用户程序读取。调整设定值时，可采用十字螺丝刀调整，按顺时针方向旋转模拟电位器，设定值从小到大变化，模拟电位器最大旋转角度为 270°，请勿过度用力旋转，以免损坏模拟电位器。

注 5：提供元件强制功能，方便调试和分析用户程序，提高调试效率。最多允许同时强制 128 个位元件和 16 个字元件。

注 6：在 PLC 运行过程中可在线修改用户程序。

注 7：部分 PLC 内部软元件资源已经保留给内部使用，用户程序内应尽量避免使用此类元件，详见附录三 保留元件表。

2.1.2 PLC 运行机制（扫描周期模型）

APLC 系列 PLC 主模块是按照扫描周期模型运行工作的。

系统顺序、循环地执行四种任务：执行用户程序、通讯、内务、刷新 I/O，每一轮任务被称为一个扫描周期。

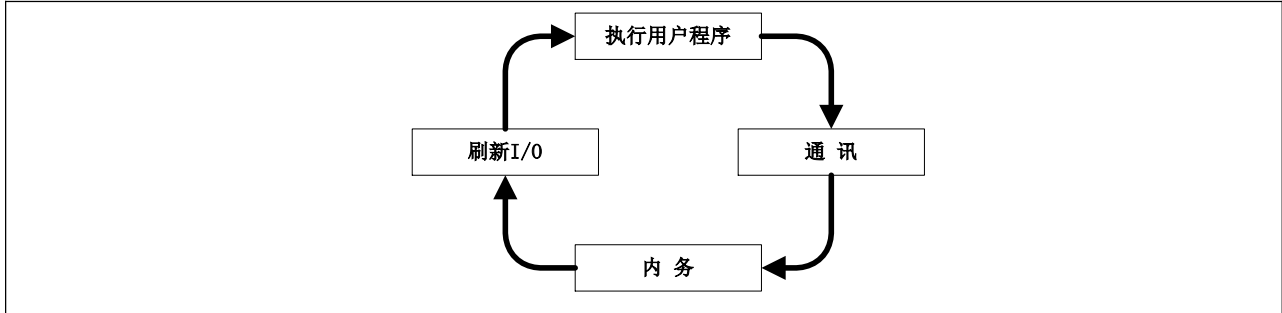


图 2-1 PLC 运行机制

执行用户程序任务

系统顺序执行用户程序的指令序列，从第一条主程序指令开始，逐一执行用户程序中的指令序列，直到执行完主程序结束指令为止。

通讯任务

与编程软件通讯，响应编程软件下达的下载、运行、停止等编程通讯命令。

内务任务

处理各种系统内务，如刷新面板指示灯、更新软件计时器计时值、刷新特殊辅助继电器和特殊数据寄存器。

刷新 I/O 任务

刷新 I/O 包含输出刷新阶段和输入刷新阶段。

输出刷新阶段：根据 Y 元件的值（ON 或 OFF），接通或断开对应的硬件输出点。

输入刷新阶段：将硬件输入点的接通/断开状态，转换为对应的 X 元件值（ON 或 OFF）。

2.1.3 用户程序运行看门狗功能

每个扫描周期系统会对用户程序的运行时间进行监控，一旦发现用户程序运行时间超过设定值，将停止用户程序运行。用户可以在 YD-station 后台软件界面的系统块对话框**设置时间**页中设置看门狗时间。

2.1.4 恒定扫描运行模式

恒定扫描运行模式是指系统在运行状态下，每个扫描周期的时间都相同。用户可以在 YD-station 后台软件界面的系统块对话框的**设置时间**页中激活恒定扫描模式，设定恒定扫描时间。恒定扫描周期默认值为 0，即禁止恒定扫描；当实际扫描周期大于恒定扫描周期时，程序按照实际扫描周期运行。

注意

恒定扫描时间设置不能大于看门狗定时时间

2.1.5 用户文件下载和存储

可以通过下载特定的用户文件到主模块中，对主模块进行编程控制。

用户文件包括四种：用户程序文件、数据块文件、系统块文件、用户辅助信息文件。用户辅助信息文件包括：全局变量表、用户数据源文件。

用户可以选择下载用户程序文件、数据块文件、系统块文件。在选择下载操作时，相应的用户辅助信息文件也会被捆绑下载。

APLC 系列所有用户文件固化到主模块的 FLASH 区中永久存储。

注意

1. 为确保下载文件能正确固化到主模块中，文件下载完毕后一段时间内（大于 30 秒钟）应保证主模块的供电正常。

2.1.6 元件初始化

PLC 在进入运行状态时（STOP→RUN），将根据掉电保存数据、EEPROM 保存数据、数据块及元件值对相关的软元件进行元件初始化，各种数据的优先级顺序如下表。

存储器种类	电源 OFF→ON	STOP→RUN
掉电保存数据	最高	最高
EEPROM 保存数据	高	高
数据块（在系统块的高级设置中选择了“数据块有效”时）	中	中
元件值（在系统块的高级设置中选择了“元件值保持”时）	—	低

表 2-2 PLC 进入运行状态时各种数据初始化的优先级顺序

2.1.7 掉电保存数据功能

APLC 系列不支持掉电保存数据功能。

2.1.8 D 元件数据的永久保存

APLC 系列可以在用户程序中使用 EROMWR 指令将 D 元件的数据 (D6000~D6999) 存储到永久存储器 (EEPROM) 中。永久存储器的操作会使扫描周期增加 2~5ms，通过保存操作所写入的数据会覆盖先前永久存储器中的数据。

注意

由于保存至 EEPROM 的操作次数是有限的 (一般为 100 万次)，所以请注意只在必要时才进行保存操作。如果在程序中持续地进行保存操作，EEPROM 可能会很快失效，从而引起 CPU 故障。

2.1.9 输入点的数字滤波功能

APLC 系列主模块的输入点 (X0~X7) 设有数字滤波功能，可以滤除端口的干扰信号，可以配置系统块中的输入过滤器项改变输入滤波常数的设定。

2.1.10 无电池模式

注意

APLC 系列 PLC 没有电池。

2.1.11 用户程序保护措施

APLC 系列 PLC 设计有多级密码保护等安全策略。

用户程序保护措施	说明
禁止格式化	在系统块中设置禁止格式化并下载系统块入 PLC 后，不能通过格式化删除 PLC 内部的用户程序、系统块和数据块。 要解除禁止格式化，必须通过重新下载新的系统块，同时系统块不应设置禁止格式化
下载密码	用于限制下载功能
禁止上载	在下载操作同时在下载对话框中选择禁止上载选项，则以后用户即使在有上载密码的情况下也不能上载。 要解除禁止上载，必须重新下载用户数据，同时在下载对话框选择允许上载选项
上载密码	用于限制上载功能
监控密码	用于限制监控功能
程序密码	对主程序、子程序和中断子程序，编程人员可以设置密码进行加密。在编程软件中打开项目时，加密后的上述程序其内容不能查看和编辑。只有打开解密对话框输入正确密码后程序方可解密，进行查看和编辑工作。 加密方法：右击要加密的程序，在右键菜单中选中 加密/解密 ，然后输入密码和确认密码，即可实现加密。解密方法：右击要解密的程序，在右键菜单中先选中 加密/解密 ，输入正确的密码即可解密

表 2-3 用户程序保护措施

注意

密码连续输入重试达 10 次错误，APLC 系列小型 PLC 将禁止输入密码功能 5 分钟。

2.2 系统配置

2.2.1 系统块

通过系统块配置的 PLC 配置信息是 PLC 用户文件的一个重要组成部分，称为系统块文件。PLC 在使用前都需要编译和下载系统块文件。

系统块配置包括以下几项：

- 保存范围（元件保存范围）
- 通讯口（通讯口和通讯协议设置）
- 输入过滤器
- 输出表（输出表设置）
- 设置时间（看门狗、恒定扫描时间和失电检测时间设置） 止格式化）
- 输入点（输入点开机模式）
- 中断优先级（中断优先级配置）
- 特殊模块配置
- 高级设置（数据块、内存元件保持、无电池模式、禁

配置好系统块以后，选择 **PLC/全部编译** 菜单，项目的系统块文件被编译，然后可以进行下载操作。

通讯口

打开系统块通讯口页面，对 PLC 的 3 个通讯口进行设置，如图 2-3 所示。设置内容包括使用的通讯协议，以及协议的具体参数等。



图 2-2 设置通讯口

系统对话框中的通讯口 0 默认是编程口协议。通讯口 1、通讯口 2 默认是无协议的。可以分别对通讯口 0、通讯口 1 和通讯口 2 进行设置。

1. 编程口协议

通讯口 0 出厂默认是编程口协议。编程口协议是 APLC 系列 PLC 编程软件通讯的专用协议。在设置为编程口协议后，还可以通过 YD-station 软件的串口配置工具来设置计算机和通讯口 0 的通讯波特率。通讯口 0 只能用于编程口通讯。

2. 自由口协议

自由口协议是自定义数据文件格式的通讯方式。自由端口通讯模式支持 ASCII 和二进制两种数据格式。

可设置内容包括：波特率、数据位、奇偶校验、停止位、起始字符检测、结束字符检测、字符间超时、帧间超时。

3. Modbus 通讯协议

Modbus 通讯设备分为主站和从站两种。主站可以与从站（包括变频器）进行通讯，按照 Modbus 通讯协议功能码，发送对从站的控制帧。从站对主站请求进行响应。

通讯口 1、2 可以设置为主站或从站。

可设置内容包括：波特率、数据位、奇偶校验、停止位、主从方式、站号、传送模式、主模式的超时时间与重试次数。

注意

有关自由口协议、Modbus 通讯协议的详细功能指南，请参阅第十章 通讯功能使用指南。

输入过滤器

单击**输入过滤器**标签，可以为 PLC 输入点设置一个输入滤波常数，通过数字滤波功能过滤掉由输入点外部引入的干扰信号。具有数字滤波功能的开关量输入点是 X0~X17（APLC 系列是 X0~X7）。其它的开关输入点采用的是硬件滤波技术。如图 2-4 所示。

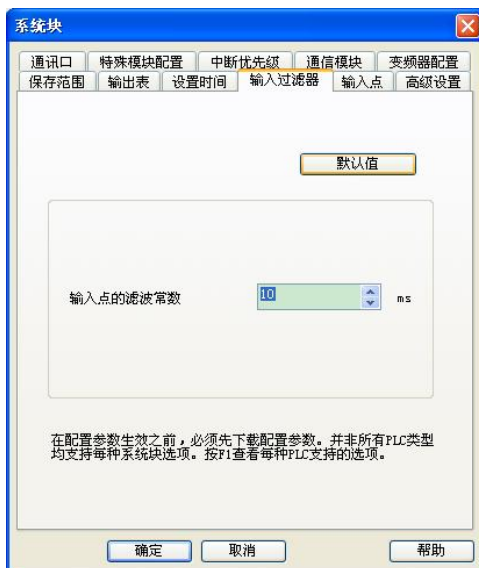


图 2-3 设置输入过滤器

输出表设置

单击**输出表**标签，可以设置 PLC 停机时的输出点状态，如图 2-5 所示。



图 2-4 设置输出表

输出表设置的功能是设置停止状态时的输出点组态，当 CPU 处于停止状态时，输出点组态有以下三种选择：

- (1) 禁止：PLC 将在停机时禁止所有的输出点，在 PLC 由运行状态切换到停机状态的时候生效。
- (2) 冻结：PLC 将在停机状态时将所有输出点冻结在最后的狀態中。
- (3) 组态：PLC 将在停机时设置输出点为已知状态。所有输出点默认状态为关闭（0）的状态。

设置时间

图 2-6 为设置时间页面。

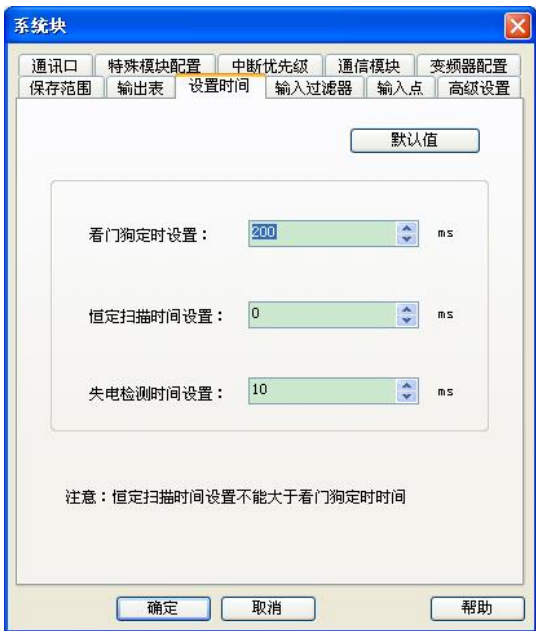


图 2-5 设置时间

1. 看门狗定时设置

设置用户程序运行看门狗时间。看门狗时间是指允许用户程序运行的最大时间，当用户程序的实际执行时间超过看门狗时间时，PLC 会停止用户程序，点亮程序警告灯（红色），输出按系统配置进行输出。看门狗时间可设置范围为：0ms~1000ms，默认值为 200ms。

2. 恒定扫描时间设置

恒定扫描时间是指系统在恒定时间扫描寄存器的时间。读取系统的恒定扫描时间设定寄存器，在恒定时间内对用户程序只作一次扫描。恒定时间可设定范围为：0ms~1000ms。默认值为 0ms。

输入点

如图 2-7 为输入点设置页面。

1. 指定用于开机的输入点

未选中**禁止输入点**复选框时，可指定 X0~X17 中的某个输入点用于强制 PLC 进入 RUN 状态，当系统处于 STOP 状态下，检测该点为 ON 时将系统状态由停止切换到运行状态。

2. 禁止输入点

选中复选框**禁止输入点**，停用输入点开机功能。

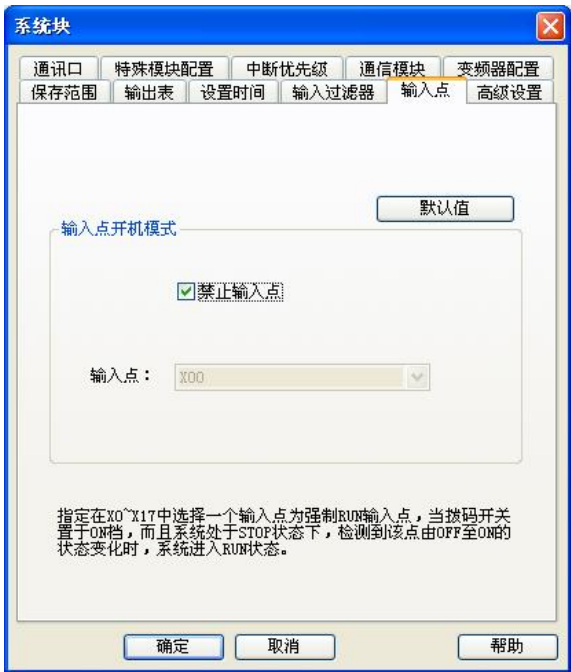


图 2-6 设置输入点

中断优先级

如图 2-8 为中断优先级设置页面。

可以通过对 PLC 内置中断的优先级别进行选择。优先级别有两级，分别是低和高。



图 2-7 设置中断优先级

变频器配置

可以对变频器型号进行选择，并可以对站号进行设置。如下图所示：

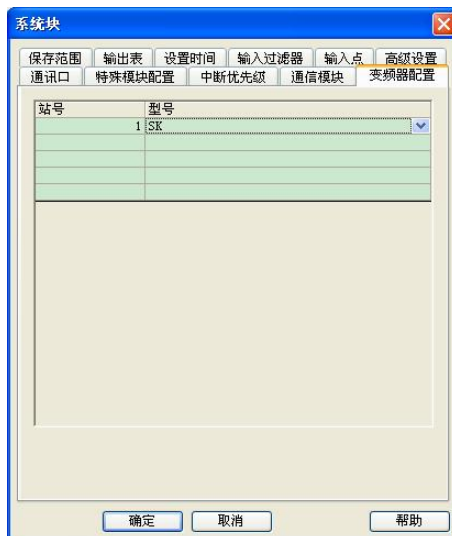


图 2-8 变频器配置

高级设置

功能：配置一些诸如数据块有效、元件值保持、无电池模式等高级设置。

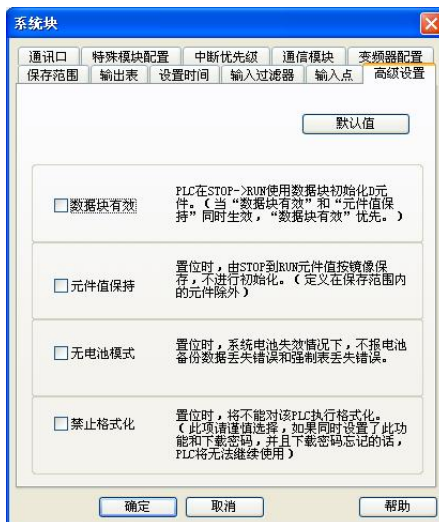


图 2-9 高级设置

1. 数据块有效

选中该项，PLC 在 STOP 状态到 RUN 状态时将使用数据块初始化 D 元件。

元件值保持

选中该项，由 STOP 状态到 RUN 状态时元件值按镜像保存，不进行初始化。

注意

当数据块有效和元件值保持同时生效，数据块有效优先。请参见 2.1.6 元件初始化。

无电池模式

选中该项，在备份电池失效情况下，系统不报电池备份数据丢失错误和强制表丢失错误。

2.2.2 数据块

数据块用于设置 D 元件中的缺省值，设置完成并通过编译后可以下载到 PLC 中。在 PLC 进入运行状态后 PLC 将首先使用数据块来初始化相关的 D 元件。

在数据块编辑器中，可以对 D 寄存器（数据内存）进行初始数据赋值，可以对 D 元件的字或双字赋值，但是不能对字节赋值。也可以在数据块编辑器中编写注释，在字符串前增加双斜线可以将其后的内容设为注释。

对数据块的详细操作说明请参阅《YD-station 编程软件用户手册》。

2.2.3 全局变量表

全局变量是为 PLC 的某个地址定义的有意义的符号名称，该符号名称在整个工程范围内都可被访问到，相当于使用该变量对应的元件，全局变量在全局变量表中定义。全局变量表包含变量名、变量地址、注释三个属性。

全局变量的定义规则为：由 A~Z、a~z、0~9、下划线、汉字混合组合而成，变量名不能以数字开始，也不能是单独的数字。名称不区分大小写，长度不能超过 8 个字节，不能使用元件类型字母加数字作为程序和变量名。变量名中不能包含空格，不能使用和关键字相同的名称，保留的关键字包括：基本数据类型名称、指令名称、指令表语言中的运算符。

对 20/10 系列小型 PLC 而言，全局变量允许下载个数分别不能超过 500/140 个（按照最大注释量），超过这一数目，只能保存在本地，而不能下载。如图 2-11 所示：



	变量名	变量地址	注释
1	停止按钮	X0	停止按钮
2	原点回归	X1	原点回归按钮
3	正向点动	X2	正向点动按钮
4	反向点动	X3	反向点动按钮
5	正向定位控制	X4	正向定位控制按钮
6	反向定位控制	X5	反向定位控制按钮
7	近点信号	X6	近点信号状态
8	系统启动	X7	系统启动读取绝对位
9	Y0输出控制	SM80	Y0输出控制
10	清零有效	SM85	清零有效
11	Y0当前值	SD80	Y0定位指令当前值
12	最高速度	SD85	最高速度
13	加减速时间	SD87	加减速时间
14	伺服故障信号	X13	伺服放大器故障信号

图 2-10 全局变量表

2.3 运行方式和状态控制

可以通过三种方式来控制 PLC 进入或退出运行状态：

1. 通过在系统块设置输入点开机模式及外部端子，由该指定的端子进行控制；
2. 可通过编程软件来控制 PLC 的运行和停止。

2.3.1 系统运行停止状态概念

主模块工作状态分为运行状态与停止状态。

运行状态 (RUN)

当主模块处于运行状态时，用户程序将被系统执行，即一个扫描周期完整地包含四个任务（执行用户程序→通讯→内务→刷新 I/O）。

停止状态 (STOP)

主模块处于停止状态时，系统不执行用户程序，但每个扫描周期中其他三个任务仍然被系统执行（通讯→内务→刷新 I/O）。

2.3.2 运行停止状态转换

如何进入运行状态 (STOP→RUN)

1. 复位方式

复位后（包括系统上电复位）系统自动进入运行状态。

注意

如果主模块中的**输入点控制模式**系统配置项有效，指定输入端的状态应为 ON，否则无法进入运行状态。

2. 输入点开机模式

当系统块**输入点开机模式**系统配置项为有效时，在停止状态下，系统检测到所指定的输入点（X0~X17）发生了 OFF→ON 状态变化则主模块进入运行状态。

注意

选用输入点控制方式时，同时模式选择开关应处于 ON 位置，否则无法进入运行状态。

如何进入停止状态 (RUN→STOP)

1. 指令控制方式

在运行状态下，当用户程序中的 STOP 指令被有效执行后，系统进入停止状态。

2. 错误停止方式

系统检测到有严重错误发生时（如用户程序错误，用户程序超时运行等），自动停止用户程序的执行。

2.3.3 停止状态下输出点状态设定

用户可设定停止状态下输出点（Y）的输出状态，提供三种模式供用户选择：

禁止输出模式—停止状态下所有输出点为 OFF。

冻结输出模式—停止状态所有输出点保持停止前的状态。

组态输出模式—停止状态下用户可以根据需要设置停止状态下输出点的的状态。

用户可在系统块**输出表**中设定停止状态下输出点的状态。参见 2.2.1 系统块的输出表设置。

2.4 系统调试

2.4.1 程序下载和上载

下载

下载功能用于把 YD-station 软件生成的系统块、数据块、用户程序通过串口下载到 PLC 中，下载时要求 PLC 处于停止运行状态。

下载时，如果上次全部编译后程序发生了改变，会提示是否需要重新编译程序

注意

选择**否(N)**不重新编译，软件会使用上次编辑的结果，下载到 PLC 中运行的程序和软件界面显示的程序会不一样。

下载时，如果设有下载密码，并且在启动软件后没有输入过下载密码，软件会弹出密码窗口要求输入下载密码。密码输入校验正确后开始下载，密码错误会提示重新输入密码。点击取消按钮退出下载。

上载

上载功能用于把 PLC 中的系统块、数据块、用户程序等内容通过串口上载到计算机中，并且为其新建工程保存。在电池备份数据有效情况下，选择上载时，相应的用户辅助信息文件也会被捆绑上载。如图 2-12 所示为上载对话框。



图 2-11 上载对话框

上载程序时，没有设上载密码时，可直接上载程序。如果设有上载密码，并且在启动软件后没有输入上载密码，软件会弹出密码窗口要求输入上载密码，密码输入正确则开始上载，密码错误则软件提示并返回上载对话框界面。

如果在下载程序时，选中了**禁止上载**功能，则 PLC 以后将不能上载程序，除非输入正确密码来解除了禁止上载功能。

2.4.2 报错机制

系统可以检测和报出两类错误：系统错误、用户程序运行错误。

系统错误是系统异常运行导致的错误。

用户程序运行错误是用户程序异常执行导致的错误。

所有错误都统一编号，每个错误编号代表一种错误，详见附录六 系统错误代码表。

系统错误的报错机制

当系统检测到有系统错误发生时，系统错误编号将被写入特殊数据寄存器 SD3 中，同时特殊继电器 SM3 会发生置位，可通过读取存储在 SD3 中的错误编号，了解当前发生了何种系统错误。

当多个系统错误同时出现时，系统按错误的严重程度，在 SD3 指示严重程度最高的错误。

严重的系统错误将导致用户程序停止运行，并会引起主模块上的 ERR 指示灯长期点亮。

用户程序运行错误的报错机制

当用户程序运行错误发生时，系统将置位特殊继电器 SM20，同时将当前发生错误的编号写入特殊数据寄存器 SD20。

当正确执行下一条应用指令后，SM20 会被清零，但 SD20 中仍然记录着的上一次错误的编号。

系统以错误记录栈方式记录先后发生的用户程序运行错误，特殊数据寄存器 SD20~SD24 构成了一个大小为 5 的错误记录栈，SD20~SD24 中记录了最近 5 次用户程序运行错误的错误编码。

当发生用户程序运行错误时，当前错误编码和 SD20 中所记录的不一致时，将会发生错误记录的压栈操作。下图演示了用户程序运行错误发生时，错误编码的压栈操作过程：

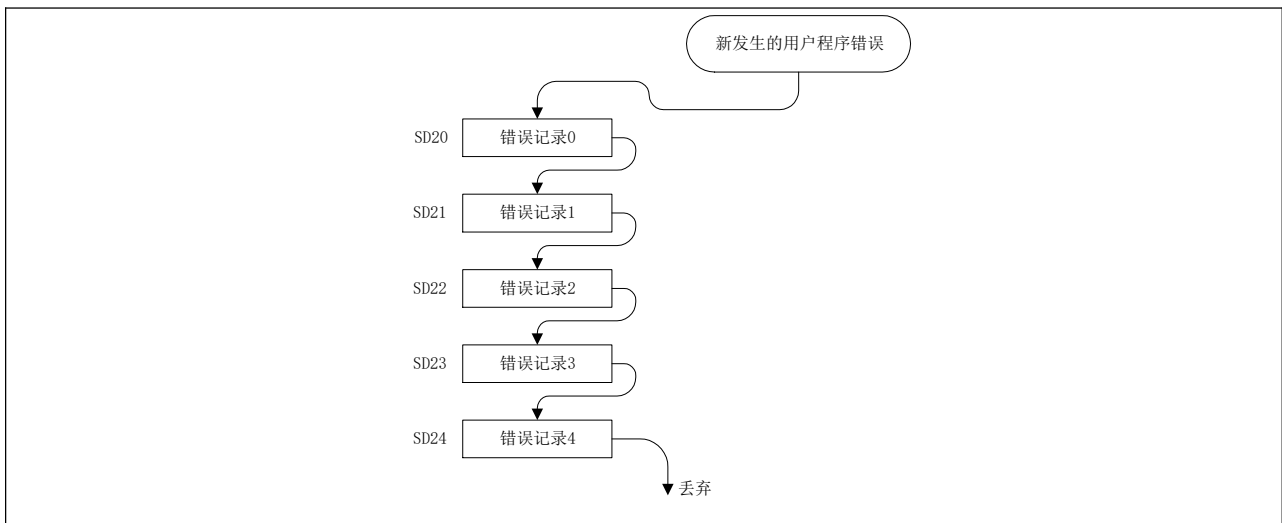


图 2-12 错误编码的压栈操作过程

严重用户程序运行错误将导致用户程序停止运行的，并会引起主模块上的 ERR 指示灯长期点亮，而一般严重的用户程序运行错误，不会点亮主模块上的 ERR 指示灯。

在线检查报错信息

在与 PLC 通过串口联机的情况下，YD-station 编程软件可以读取当前 PLC 的各种状态信息，其中包括了上述系统错误和用户程序运行错误的代码和描述信息。

在 YD-station 软件中，点击 **PLC->PLC 信息** 选项打开如下窗口。



图 2-13 PLC 信息

图中**系统错误编号**就是 SD3 中存放的系统错误的编号，**执行错误编号**就是 SD20 中存放的用户程序运行错误的编号。同时显示的相关错误描述可供用户参考。

2.4.3 在线修改

需要在 PLC 运行状态下修改其中的程序时，可以使用在线修改功能。

警告

在可能导致人员伤亡或财产损失的情况下，应由专业人士，在由相应安全措施保证下使用在线修改用户程序功能。

操作方法

在确保软件已经与 PLC 硬件成功建立了通讯关系，且 PLC 处于运行状态后，点击**调试->在线修改**菜单切换到在线修改状态。在在线修改状态下，可以与正常编辑时一样修改主程序、子程序及中断子程序的内容，修改完之后，点击**PLC->下载**菜单，软件将会编译当前工程的所有程序并自动下载至 PLC 硬件中。下载完成后，PLC 就会按照新下载的程序运行。

限制条件

1. 在线修改状态下，不能修改全局变量表及任何程序的局部变量表，也不能添加或删除任何子程序/中断子程序；
2. 当程序处于在线修改状态下时，如果 PLC 被停止，软件会自动退出在线修改状态。

2.4.4 清除与格式化

清除操作包括了：PLC 元件值清除、PLC 程序清除、PLC 数据块清除。

格式化是将 PLC 内全部数据和程序清除。

PLC 元件值清除

PLC 元件值清除功能清除 PLC 中所有元件值，清除元件值需要 PLC 处于停止运行状态。

清除 PLC 中的元件值可能导致 PLC 运行不正确或者中间工作数据丢失，请谨慎使用此功能。为防止误操作，操作时软件会显示确认窗口。

PLC 程序清除

PLC 程序清除功能清除 PLC 中的用户程序，清除用户程序需要 PLC 处于停止运行状态。

清除 PLC 中的用户程序会导致 PLC 运行后不执行任何用户程序，请谨慎使用此功能。为防止误操作，操作时软件会显示确认窗口。

PLC 数据块清除

PLC 数据块清除功能清除 PLC 中所有数据块设置，清除数据块需要 PLC 处于停止运行状态。

清除 PLC 中的数据块会导致 PLC 运行后不再使用数据块的预设值初始化 D 元件，请谨慎使用此功能。为防止误操作，操作时软件会显示确认窗口。

PLC 格式化

PLC 格式化功能格式化 PLC 中所有数据，包括清除用户程序、恢复默认配置、清除清除数据块，清除数据块需要 PLC 处于停止运行状态。

本操作会丢失 PLC 中已经下载和设置的全部数据，请谨慎使用此功能。为防止误操作，操作时软件会显示确认窗口。

2.4.5 PLC 信息在线查询

PLC 信息

PLC 信息功能从 PLC 中获取并显示各种运行数据和 PLC 重要信息。

在信息显示窗口上可以看到 PLC 当前运行的重要信息。

PLC 时间

APLC 不支持此功能。

2.4.6 元件值写入和强制、元件监控表

元件值写入和强制

在调试过程中，可能为了达成某些条件，而需要手工改变某些软元件的值，元件值写入和强制提供了这个功能。写入与强制的不同之处在于：写入元件值仅一次有效，写入后的值可能会随着程序的运行被改变，但是强制的元件值会一直记录在 PLC 硬件中，直到取消强制为止。

需要执行写入或强制功能时，首先选中要写入或强制的元件，从右键菜单中选择**写入**或者**强制**，此时会弹出相应对话框，列出所选元件引用到的所有软元件地址，可以有选择的写入或强制某些软元件值，确定后这些值就会下发到 PLC 硬件，当这些值在硬件中生效后，在后续的调试过程中就可以看到变化结果。

写入对话框请参见图 2-19：



图 2-14 写入对话框

强制对话框请参见图 2-17:



图 2-15 强制对话框

强制的软元件，在梯形图中会有一个锁标记，如图 2-18 所示：

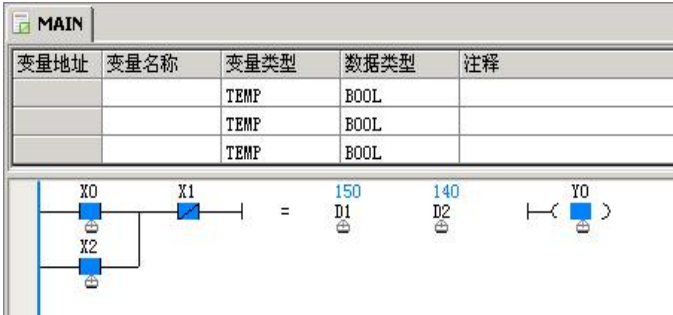


图 2-16 被强制软元件的锁标记

取消强制

对于不再需要强制的元件，可以将它解除强制。需要使用解除强制功能时，先选中要解除的元件，从右键菜单选择**取消强制**，此时会弹出相应对话框，列出所选元件中已经被强制过的软元件，可以有选择的解除某些软元件的强制，点击确定后，这些强制值会从 PLC 硬件中删除，该软元件对应的锁标记也会消失。解除强制对话框请参见图 2-19。



图 2-17 取消强制对话框

元件监控表

元件状态监控表提供了在调试过程中监控元件值的功能，调试程序时允许将程序输入元件、输出元件、寄存器位和字元件置入元件监控表中以便在将程序下载至 PLC 之后追踪其状态。

元件监控表有两种模式：编辑模式和监控模式。编辑模式下可以执行所有的编辑功能，不能执行监控的功能。在监控模式下，可以同时执行监控功能和编辑功能。

元件状态监控表在监控模式下会自动刷新元件值。无论是修改后的元件值还是强制后的元件值都会得到及时更新。

元件状态监控表可提供编辑功能、排序、查找、自动刷新显示指定元件的当前值、写入元件值、强行指定元件/变量的值、解除强制等功能。

元件状态监控表请参见图 2-20：

MAIN	全局变量表	元件监控表			
	元件名称	数据类型	显示格式	当前值	新数值
1	X1	BOOL	二进制		
2	Y1	BOOL	二进制		
3	D1	WORD	十进制		
4	D2	WORD	十进制		
5	开关6	BOOL	二进制		
6		WORD	十进制		
7		WORD	十进制		
8		WORD	十进制		

图 2-18 元件状态监控表示意图

2.4.7 从 RAM 中生成数据块

从 PLC 中连续读取最多 500 个 D 寄存器的数据值显示出来，结果可以合并到数据块中或者覆盖原来的数据块。

打开从 RAM 生成数块窗口，如图 2-21 所示：



图 2-19 RAM 生成数块窗口

输入要读取的数据块的范围，点击**从 RAM 读取**按钮，执行正确后数据被读取到列表中。

可以在**显示类型**中选择以 16、10、8、2 进制来显示数据。

读取成功后**合并到数据块**和**覆盖数据块**按钮变为可用，点击**合并到数据块**按钮将生成结果追加到当前数据块的内容后面；点击**覆盖数据块**按钮将生成结果替代数据块中已经存在的内容。退出寄存器值读取窗口后，软件提示数据块已经发生变化并自动打开数据块窗口。

第三章 软元件和数据

本章详细讲述了 APLC 系列 PLC 各种软元件的定义、种类和作用。

3.1 软元件种类和作用	48
3.1.1 软元件概述	48
3.1.2 软元件一览表	49
3.1.3 输入点和输出点	50
3.1.4 辅助继电器	51
3.1.5 状态继电器	52
3.1.6 定时器	53
3.1.7 计数器	54
3.1.8 数据寄存器	55
3.1.9 特殊辅助继电器	55
3.1.10 特殊数据寄存器	56
3.1.11 变址寻址寄存器	57
3.1.12 局部辅助继电器	58
3.1.13 局部数据寄存器	58
3.2 软元件寻址方式	59
3.2.1 位串组合寻址方式 (Kn 寻址方式)	59
3.2.2 变址寻址方式 (Z 寻址方式)	60
3.2.3 位串组合的变址寻址方式	60
3.2.4 D、V 元件对 32 位数据的存储和寻址	60
3.3 数据	61
3.3.1 数据类型	61
3.3.2 元件与数据类型的匹配关系	61
3.3.3 常数	62

3.1 软元件种类和作用

3.1.1 软元件概述

PLC 在系统设计中配置了多种虚拟的元件，用以代替继电器控制电路中真实的普通继电器、时间继电器等器件。这些虚拟的元件统称为软元件。PLC 采用软元件来进行程序运算和系统功能配置，从而实现所有的运算和控制功能。由于软元件是虚拟的元件，因此可以在程序中反复使用，理论上没有数量的限制（实际上与程序容量有关），而且软元件不存在真实器件的机械电气方面的故障问题，从而使 PLC 的可靠性远远高于继电器控制电路，容易编程，修改逻辑更为方便。

APLC 系列 PLC 的软元件的种类和作用如下图所示。

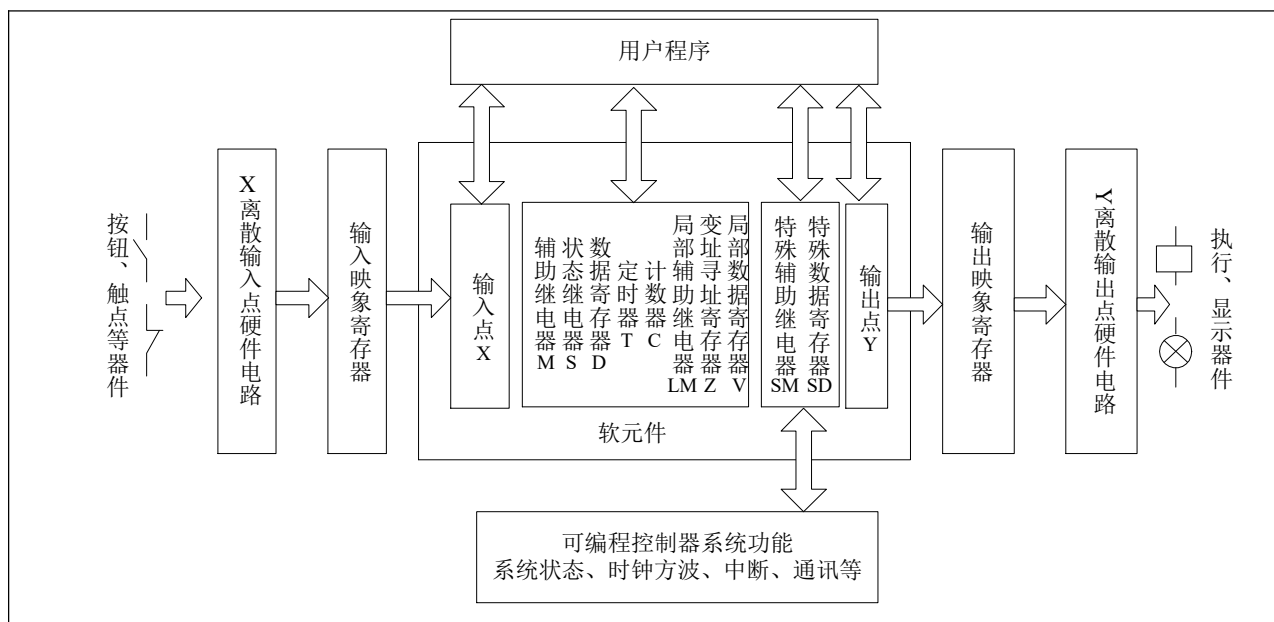


图 3-1 PLC 软元件的种类和作用

在本手册中，软元件按种类名称简称为“某某元件”。例如：

- 输入点 X 简称为“X 元件”
- 输出点 Y 简称为“Y 元件”
- 辅助继电器 M 简称为“M 元件”
- 数据寄存器 D 简称为“D 元件”
- 状态继电器 S 简称为“S 元件”

3.1.2 软元件一览表

APLC 系列小型 PLC 的软元件种类按功能编制和划分，不同元件执行不同的功能，且寻址简便。

APLC 系列小型 PLC 的软元件一览表如下所示。

		APLC 系列	元件编址方式
软元件资源 ^{注4}	输入输出点	128 入/128 出（输入 X0~X177，输出 Y0~Y177，APLC-TCP 只支持 10 输入 6 输出） ^{注1}	8 进制
	辅助继电器	2048 点（M0~M2047）	10 进制
	局部辅助继电器 ^{注5}	64 点（LM0~LM63）	10 进制
	特殊辅助继电器	256 点（SM0~SM255）	10 进制
	状态继电器	1024 点（S0~S1023）	10 进制
	定时器	256 个（T0~T255） ^{注2}	10 进制
	计数器	256 个（C0~C255） ^{注3}	10 进制
	数据寄存器	8000 个（D0~D7999）	10 进制
	局部数据寄存器 ^{注5}	64 个（V0~V63）	10 进制
	变址寻址寄存器	16 个（Z0~Z15）	10 进制
	特殊数据寄存器	256 个（SD0~SD255）	10 进制

注释：

1：X，Y 元件的地址编号采用 8 进制编址，地址 X10 代表第 8 个输入点。这里输入输出点数的最大值是系统容量，实际能扩展的硬件点数需要根据 PLC 系统配置而定（包括可用的扩展模块类型和点数、电源容量限制等）。

2：T 元件地址按计时精度划分三类：

- 100ms 精度 T0~T209
- 10ms 精度 T210~T251
- 1ms 精度 T252~T255

3：C 元件地址按计数值的宽度和功能划分为三类：

- 16 位增计数器 C0~C199
- 32 位增减计数器 C200~C235
- 32 位高速计数器 C236~C255

4：部分 PLC 内部软元件资源已经保留给内部使用，用户程序内应尽量避免使用此类元件，详见附录三 保留元件表。

5：此 2 类软元件为局部变量，不能在全局变量表中定义。在调用子程序和返回主程序时，将会清零，或根据接口参数传递功能来获得参数值或状态

表 3-2 APLC 系列 PLC 软元件一览表

3.1.3 输入点和输出点

简称

- X 元件(离散输入点)
- Y 元件(离散输出点)

作用

它们分别是代表了硬件 X 端子输入状态和硬件 Y 端子输出状态的软元件。

X 元件状态的采集是通过输入映像寄存器进行的。Y 元件状态的输出是通过输出映像寄存器来驱动输出电路实现的。这两个操作都在 PLC 扫描周期模型中的 I/O 刷新阶段进行, 如图 3-2 所示, 详细情况请参见 2.1.2 PLC 运行机制(扫描周期模型)。可见在用户程序运行过程中, PLC 对 I/O 的响应有短暂延迟的特性, 与输入滤波、通讯、内务处理和扫描周期有关。

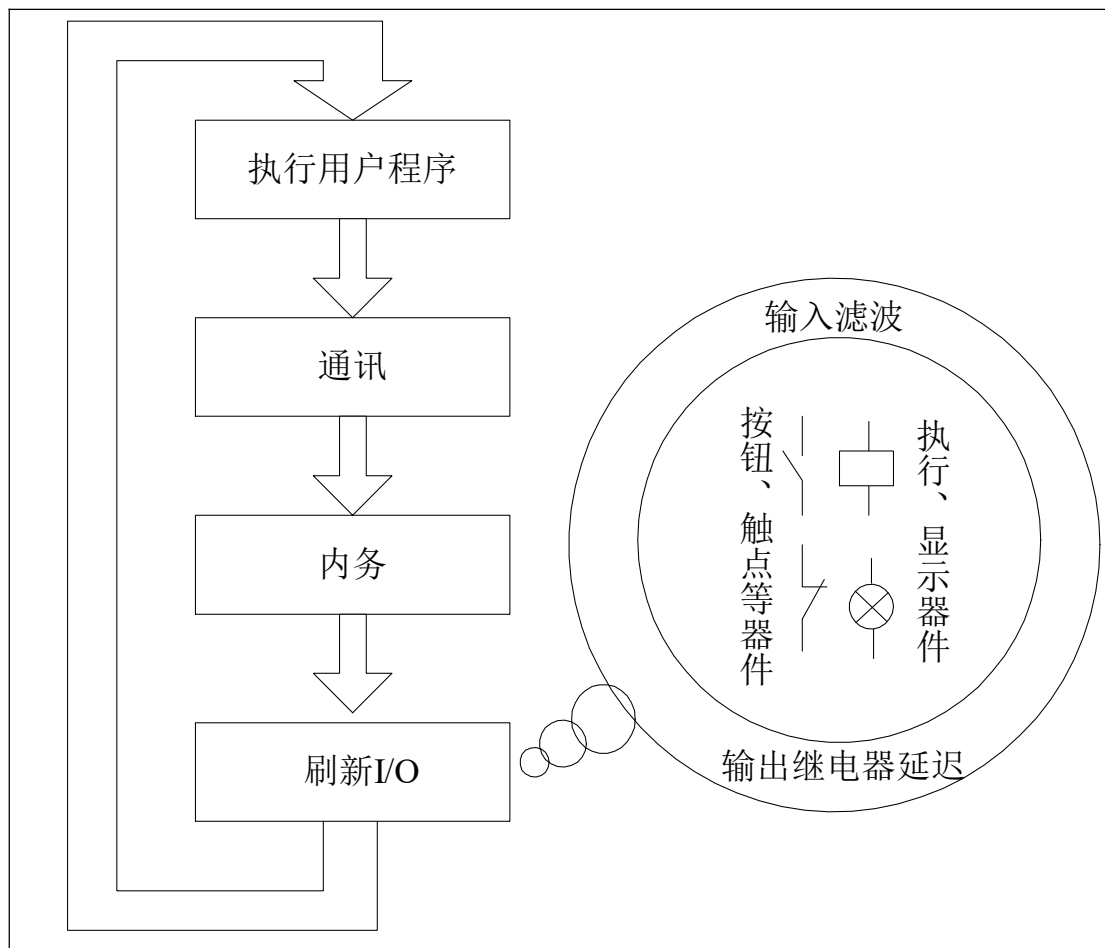


图 3-2 I/O 刷新原理

分类

X 元件对应的输入通道：X0~X17 有数字滤波功能，可通过系统块设置滤波时间；其余 X 输入点均为硬件滤波。X0~X5 可以作高速计数器软元件的计数输入端；X0~X7 还可作为外部中断、脉冲捕捉、SPD 测频指令的输入端。

Y 元件中的 Y0 和 Y1 可以作为高速输出之用，其余为普通输出端。

编址方式

八进制，从地址 0 开始。主模块和 I/O 扩展模块的 X 元件和 Y 元件编址是连续的。对 X 元件来说，连续编址为 X0~X7、X10~X17、X20~X27……对 Y 来说，连续编址为 Y0~Y7、Y10~Y17、Y20~Y27……。

数据类型

X 和 Y 元件均为布尔元件（元件值为 ON 或 OFF）。

可用形式

编程时可以采用 X 元件的常开触点和常闭触点（通过两种的指令引用）。常开和常闭这两种触点状态值相反，有的场合将它们分别称为 a 触点和 b 触点。

编程时也可以采用 Y 元件的常开触点和常闭触点。

赋值方式

1. X 元件只接受硬件输入状态及强制操作状态值，在用户程序中不能通过输出及设置指令修改，也不能在系统调试时接受写入状态值。
2. Y 元件可通过线圈输出指令来赋予其状态值，也可以被设置状态值，还可以在系统调试时接受强制及写入状态值。
3. 通过系统块可以设置在 STOP 状态下 Y 元件的输出状态。

3.1.4 辅助继电器

简称

M 元件

作用

系统提供给用户的一种离散型状态元件，类似于真实电气控制电路中的中间继电器，可用于保存用户程序中的各种中间状态。

编址方式

十进制，从地址 0 开始。

数据类型

布尔（元件值为 ON 或 OFF）。

可用形式

常开触点和常闭触点。

赋值方式

1. 指令操作；
2. 在系统调试时强制及写入状态值。

3.1.5 状态继电器

简称

S 元件

别名

步进状态符

作用

主要用于顺序功能图的编程，作为步进状态的标志。具体参见第七章 顺序功能图使用指南。

分类

S0~S19 为初始步进符，其余均为普通步进符。

编址方式

十进制，从地址 0 开始。

数据类型

布尔（元件值为 ON 或 OFF）。

可用形式

1. 代表步进状态（用于顺序功能图编程 STL 指令时）；
2. 常开触点和常闭触点（不用于顺序功能图编程 STL 指令时）。其特性与 M 元件类似，编程时可以采用 S 元件的常开触点和常闭触点。

赋值方式

1. 指令操作；2. 在系统调试时强制及写入状态值。

3.1.6 定时器

简称

T 元件

作用

T 元件是一个复合型的软元件，它包含了一个字元件（2 字节）和一个位元件。T 字元件记录 16 位的计时值，可以作为数值在程序中使用；T 位元件反映计时器线圈状态，用于逻辑控制。

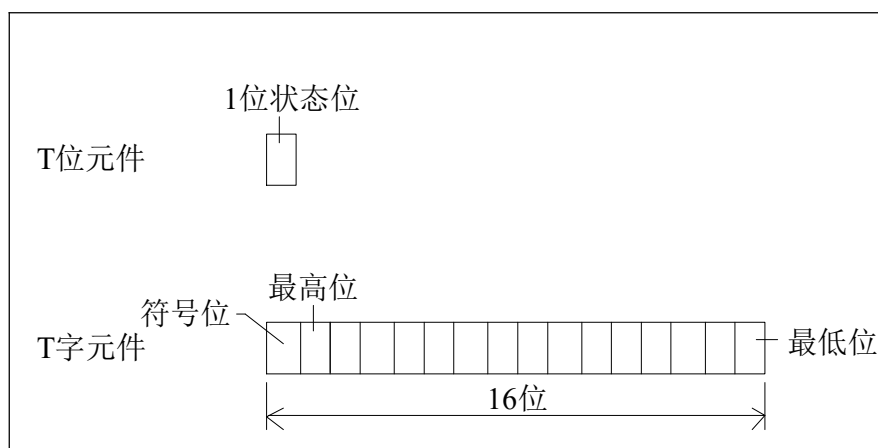


图 3-3 T 元件

分类

T 元件的计时精度有 3 种。如下表所示为不同地址段的 T 元件及其对应的计时精度，使用时需要注意。

T 元件	计时精度
T0~T209	100ms 精度
T210~T251	10ms 精度
T252~T255	1ms 精度

计时精度为 1ms 的 T 元件，其计时为中断触发，与 PLC 扫描周期无关，因此计时的动作时间最准确。计时精度为 10ms 和 100ms 的 T 元件，计时值的刷新和动作时间与 PLC 扫描周期有关。

编址方式

十进制，从地址 0 开始。

数据类型

布尔（元件值为 ON 或 OFF），字。

可用形式

T 元件的计时和动作方式取决于调用它的计时指令。指令共有 4 种：接通延时计时指令、断开延时计时指令、记忆型接通延时计时指令、不重触发单稳计时指令。有关这 4 种指令的说明请参阅第五章 基本指令说明

赋值方式

1. 指令操作；2. 在系统调试时强制及写入状态值。

注意

T 元件最大计时值为 32767，预设值为 -32768~32767。由于 T 元件是以计时值大于或等于预设值为动作条件的，因此将预设值设置为负数没有实际意义。

3.1.7 计数器

简称

C 元件

作用

C 元件是一个复合型的软元件，它包含了一个位元件和一个单字或双字元件（2 字节或 4 字节）。C 字元件记录 16 位或 32 位的计数值，C 位元件反映计数器线圈状态。C 字元件可以作为数值在程序中使用，C 位元件用于逻辑控制。

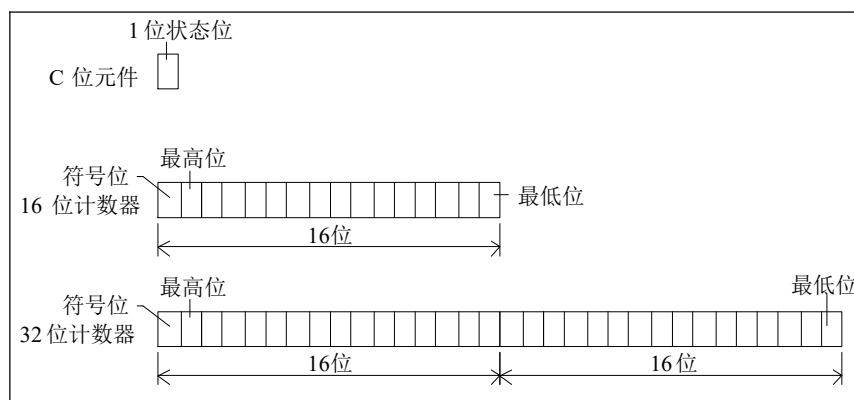


图 3-3C 元件

分类

16 位计数器和 32 位计数器两类。

编址方式

十进制，从地址 0 开始。

数据类型

布尔（元件值为 ON 或 OFF），单字或双字。

可用形式

调用 C 元件的计数指令有 4 类，分别是 16 位增计数器指令、16 位循环计数指令、32 位增减计数指令、高速 I/O 指令。有关这 4 类指令的说明请参阅第五章 基本指令说明和第六章 应用指令说明。C 元件分类如下表所示。

C 元件	计数功能	适用的指令类型
C0~C199	16 位增计数器	16 位增计数指令 16 位循环计数指令
C200~C235	32 位增减计数器	32 位增减计数指令

C236~C255

32 位高速计数器

高速 I/O 指令

赋值方式

1. 指令操作；2. 在系统调试时强制及写入状态值。

3.1.8 数据寄存器

简称

D 元件

作用

作为数据元件，很多运算、控制指令都会用到 D 元件，作为操作数。

编址方式

十进制，从地址 0 开始。

数据类型

每 1 个 D 元件都是 16 位的寄存器，可存储 16 位的数据，例如 16 位整数。

2 个 D 元件可以组合为双字元件，用于存储 32 位数据，例如长整型数据或浮点型数据。

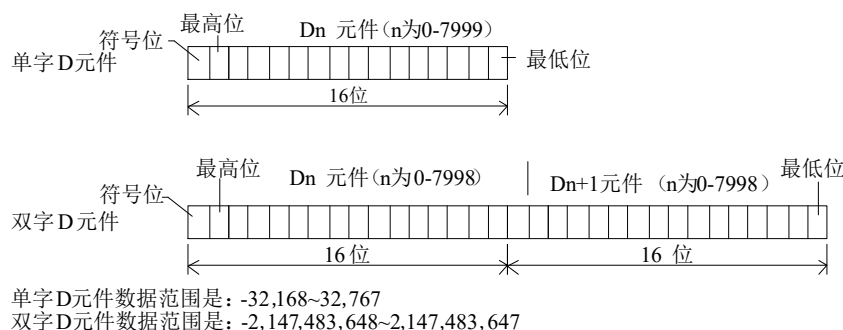


图 3-4 D 元件

注意

双字 D 元件中，高 16 位在第 1 个 D 元件，低 16 位在第 2 个 D 元件。

可用形式

很多运算、控制指令都会用到 D 元件，作为操作数。

赋值方式

1. 数据块初始化；2. 指令操作；3. 在系统调试时强制及写入状态值。

注意

在采用变频器指令功能时，部分 D 元件会被系统调用，用户在编程和修改程序时请留意。

3.1.9 特殊辅助继电器

简称

SM 元件

作用

SM 元件是与 PLC 系统功能密切相关的软元件。SM 元件反映了 PLC 系统功能、状态。有关所有 SM 元件的详细功能说明，请参阅本手册附录一 特殊辅助继电器的相关内容。

分类

常用的此类元件有：

- SM0：监控运行位，在 RUN 状态下保持 ON 状态。
- SM1：初始运行脉冲位，运行第一个扫描周期内为 ON。
- SM3：系统错误，上电后或 STOP 到 RUN 时检测有系统错误发生时为 ON。
- SM10~SM12：分别是以 10ms、100ms、1s 为周期的时钟振荡方波，半个周期翻转一次。

对部分 SM 元件进行状态修改还可以调用、控制、改变 PLC 系统功能。常用的此类元件有：

- SM40~SM68：中断控制标志位，对这些 SM 元件进行置位则可使能相应的中断功能。
- SM80/81：Y0/Y1 高速脉冲输出停止指令。
- SM110~SM114：自由口 0 监控位元件。
- SM135/136：Modbus 通讯标志位元件。

编址方式

十进制，从地址 0 开始。

数据类型

布尔（元件值为 ON 或 OFF）。

可用形式

常开触点和常闭触点。

赋值方式

1. 指令操作；2. 在系统调试时强制及写入状态值。

注意

只读的 SM 元件不能赋值。

3.1.10 特殊数据寄存器

简称

SD 元件

作用

与 PLC 系统功能密切相关的软元件，反映了 PLC 系统功能参数、状态代码值、指令运行数据。有关所有 SD 元件的详细功能说明，请参阅本手册附录二 特殊数据寄存器。

分类

常用的 SD 元件有：

- SD3：系统错误代码。
- SD50~SD57：高速脉冲输出监控。
- SD100~SD106：实时时钟数据。

对部分 SD 元件的数据进行修改还可以改变 PLC 系统功能参数。常用的此类元件有：

- SD66~SD68：定时中断周期设置值；
- SD80~SD89：定位指令参数；

编址方式

十进制，从地址 0 开始。

数据类型

字、双字（整数）元件。

可用形式

整数存放与运算。

赋值方式

1. 指令操作；2. 在系统调试时强制及写入状态值。

注意

只读的 SD 元件不能赋值。

3.1.11 变址寻址寄存器

简称

Z 元件

作用

16 位寄存器元件，可存储符号整数数据。有关变址寻址的内容，请参阅 1.1.1

变址寻址方式（Z 寻址方式）。

编址方式

十进制，从地址 0 开始。

数据类型

字元件。

可用形式

用于变址寻址功能。要使用 Z 元件时，先对 Z 元件写入地址偏移量的数据。

赋值方式

1, 指令操作; 2, 在系统调试时强制及写入状态值。

3.1.12 局部辅助继电器

简称

LM 元件

作用

LM 元件是局部变量。在主程序及子程序中可应用 LM 元件。它们是在各独立程序体内（主程序、子程序和中断程序）局部有效的变量元件，因此，在不同程序体之间是不能直接共用任何 LM 元件的状态的。在用户程序执行中离开了某一个程序体，系统就会重新定义 LM 元件。在返回主程序或调用子程序时，重新定义的 LM 元件的值将会被清零，或者根据接口参数传递功能来获得相应的状态。

可用于定义子程序的接口参数，实现接口参数传递功能。详细内容请参阅 4.4 子程序。

编址方式

十进制，从地址 0 开始。

数据类型

布尔（元件值为 ON 或 OFF）。

可用形式

常开触点和常闭触点。

赋值方式

1. 指令操作; 2. 在系统调试时强制及写入状态值。

3.1.13 局部数据寄存器

简称

V 元件

作用

V 元件是局部变量。在主程序及子程序中可应用 V 元件。它们是在各独立程序体内（主程序和子程序）局部有效的变量元件，因此，在不同程序体之间是不能直接共用任何 V 元件的数据的。在用户程序执行中离开了某一个程序体，系统就会重新定义 V 元件。在返回主程序或调用子程序时，重新定义的 V 元件的值将会被清零，或者根据接口参数传递功能来获得相应的数据。

V 元件可用于定义子程序的接口参数，实现接口参数传递功能。详细内容请参阅 4.4 子程序。

编址方式

十进制，从地址 0 开始。

数据类型

布尔（元件值为 ON 或 OFF）。

可用形式

字元件，可保存数值类型的信息。

赋值方式

1. 指令操作；
2. 在系统调试时强制及写入状态值。

3.2 软元件寻址方式

3.2.1 位串组合寻址方式（Kn 寻址方式）

位串组合寻址方式概念

位串组合寻址方式（Kn 寻址方式）是将位元件串组合成字或长字使用。

位串组合寻址方法

位串组合寻址格式为 K (n) (U)，其中 n 是一个 1~8 的整数，表示元件串长度为 n×4 位。

U 代表元件串的起始位元件地址。

具体示例：

1. K1X0 代表：4 位长的位串（X0，X1，X2，X3）组成的一个字使用。
2. K3Y0 代表：12 位长的位串（Y0，Y01，Y02，Y03），（Y04，Y05，Y06，Y07），（Y10，Y11，Y12，Y13）组成一个字使用。
3. K4M0 代表：16 位长的位串 M0，M1，M2，M3...，M15 组成一个字使用。
4. K8M0 代表：32 位长的位串 M0，M1，M2，M3...，M31 组成一个双字使用。

Kn 寻址方式数据存储格式

举例说明一个具体的数据在 Kn 寻址方式中是如何存储的：

MOV 2#10001001 K2M0（等价于 MOV 16#89 K2M0 或 MOV 137 K2M0）。当该指令执行后，K2M0 具体存储格式见下表：

数据	最高位	中间位						最低位
K2M0	M7	M6	M5	M4	M3	M2	M1	M0
16#89	1	0	0	0	1	0	0	1

位串组合寻址的注意事项

如果指令的目的操作数使用 K_n 寻址方式，而需要存储到目的操作数的数据宽度大于 K_n 寻址所指定的宽度时，系统按保留低位部分，舍去高位部分的规则存储数据。

以下举例说明该种情况：

执行指令“DBITS 16# FFFFFFF0 K1M0”。

该指令执行后，操作数 2 (K1M0) 本应存储的运算结果为 16#1c (28)，但因为 K1M0 能存储的数据的宽度为 4，不能完整地存储运算结果 16#1c，运算的高位部分将被舍去，所以结果操作数 2 实际结果为：K1M0=16#c (12)。

3.2.2 变址寻址方式 (Z 寻址方式)

无

3.2.3 位串组合的变址寻址方式

位串组合寻址方式也可配合变址寻址方式使用，即形如 K1X0Z10。

这种寻址方式首先通过 Z 寻址确定位串组合的起始位元件的地址，再通过 K_n 寻址确定位串的长度。

以下示例说明具体的寻址过程：

```
LD M1
MOV 3 Z10
MOV K1X0Z10 D0
```

以上命令等价于：

```
LD M1
MOV K1X3 D0
```

其寻址过程如下：

```
Z10=3
K1X0Z10=K1X (0+Z10) =K1X3
```

3.2.4 D、V 元件对 32 位数据的存储和寻址

32 位数据在 D、V 元件中的存储方式

DINT、DWORD、REAL 型的数据都是 32 位宽度的，而一个 D 元件或 V 元件只有 16 位宽度，因此需要两个地址连续的 D 元件或 V 元件存储 32 位数据。

D、V 元件地址寻址 32 位数据

一个 D、V 元件地址可寻址一个 16 位数据（如 INT、WORD 型数据），也可寻址到一个 32 位数据（如 DINT、DWORD 型数据）。

如果指令操作数引用了 D 元件地址或 V 元件地址，那么该地址是代表一个 16 位数据，还是代表一个 32 位数据，将由操作数的数据类型决定。

例如：

在指令“MOV 16#34 D0”中，地址 D0 只寻址单独的 D0 元件，这是因为 MOV 指令的操作数 2 的数据类型是 WORD 型。

而在指令“DMOV 16#FEA867DA D0”中，地址 D0 代表了 D0 开始的两个连续字元件 D0，D1，这是因为 DMOV 指令的操作数 2 的数据类型是 DWORD 型。

3.3 数据

3.3.1 数据类型

指令的操作数都带有数据类型属性，共支持六种数据类型，见下表：

+	类型说明	数据宽度	范围
BOOL	位	1	ON、OFF (1、0)
INT	符号整数	16	-32768~32767
DINT	符号长整数	32	-2147483648~2147483647
WORD	字	16	0~65535 (16#0~16#FFFF)
DWORD	双字	32	0~4294967295 (16#0~16#FFFFFFFF)
REAL	浮点数	32	±1.175494E-38~±3.402823E + 38

表 3-3 操作数的数据类型

3.3.2 元件与数据类型的匹配关系

指令操作数所选用的元件类型应与数据类型保持一定的匹配关系，适用元件与数据类型的匹配关系见下表。

数据类型	软元件												
BOOL										C	T		
	X	Y	M	S	LM	SM							
INT	常数	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z
DINT	常数	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C		V	

数据类型	软元件												
WORD	常数	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z
DWORD	常数	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	
REAL	常数							D				V	

表 3-4 元件与数据类型的匹配关系

如果指令的编写不满足匹配关系，该指令将被视非法，例如指令“MOV 10 X0”是非法的，这是因为 MOV 指令操作数 2 的数据类型是符号整数型，而元件 X0 只能存储位型的数据。

说明

1. 当操作数的数据类型为 INT，WORD 型时，其适用软元件为 KnX，KnY，KnM，KnS，KnLM，KnSM， $1 \leq n \leq 4$ 。
2. 当操作数的数据类型为 DINT，DWORD 型时，其适用软元件为 KnX，KnY，KnM，KnS，KnLM，KnSM， $5 \leq n \leq 8$ 。
3. 当操作数的数据类型为 INT，WORD 型时，其适用的 C 元件的编号应为 C0~C199。
4. 当操作数的数据类型为 DINT，DWORD 型时，其适用的 C 元件的编址应为 C200~C255。

3.3.3 常数

用户可使用常数作为指令的操作数，常数的表达形式见下表所示：

常数类型	表达示例	有效范围	说明
常十进制 16 位符号整数	-8949	-32768~32767	十六进制，八进制，二进制常数本身无正负含义。 选用十六进制，八进制或二进制常数作为指令操作数，操作数的正负及大小，要根据其操作数的数据类型决定
常十进制 16 位无符号整数	65326	0~65535	
常十进制 32 位符号整数	-2147483646	-2147483648~2147483647	
常十进制 32 位无符号整数	4294967295	0~4294967295	
十六进制 16 位常数	16#1FE9	16#0~16#FFFF	
十六进制 32 位常数	16#FD1EAFE9	16#0~16#FFFFFFFF	
八进制 16 位常数	8#7173	8#0~8#177777	
八进制 32 位常数	8#71732	8#0~8#3777777777	
二进制 16 位常数	2#10111001	2#0~2#1111111111111111	符合 IEEE-754 标准。 编程软件可以显示、输入 7 位有效精度的浮点常数
二进制 32 位常数	2#101110011111	2#0~2#11111111111111111111111111111111	
单精度浮点常数	-3.1415E-16 3.1415E+3 0.016	$\pm 1.175494E-38 \sim \pm 3.402823E+38$	

表 3-5 常数的表达形式

第四章 编程概念

本章详细介绍了 APLC 系列小型 PLC 编程的相关内容，包括所采用的编程语言、程序要素；介绍子程序的编程和使用；最后介绍了指令的一些通用说明。

4.1 编程语言介绍	64
4.1.1 梯形图 (LAD)	64
4.1.2 指令列表 (IL)	65
4.1.3 顺序功能图 (SFC)	66
4.2 程序要素	66
4.2.1 用户程序	66
4.2.2 系统块	67
4.2.3 数据块	67
4.3 程序的块注释和变量的注释	67
4.3.1 块注释	67
4.3.2 变量的注释	68
4.4 子程序	70
4.4.1 子程序概念	70
4.4.2 子程序的使用注意事项	70
4.4.3 子程序变量表定义	70
4.4.4 子程序参数传递	71
4.4.5 子程序的使用示例	71
4.5 指令通用说明	73
4.5.1 指令的操作数	73
4.5.2 标志位	73
4.5.3 指令的使用限制	73

4.1 编程语言介绍

编程语言有三种：梯形图（LAD）、指令列表（IL）、顺序功能图（SFC）。

4.1.1 梯形图（LAD）

梯形图概念

梯形图是一种与电气（继电器）控制图相类似的图形化 PLC 编程语言，是一种广泛使用的 PLC 编程语言。其主要特点包括：

1. 带有左母线，而右母线省略。
2. 所有控制输出元件（线圈）和功能块（应用指令）只有一个能流输入端。

电气控制图和梯形图有一定的等效关系，如下图所示：

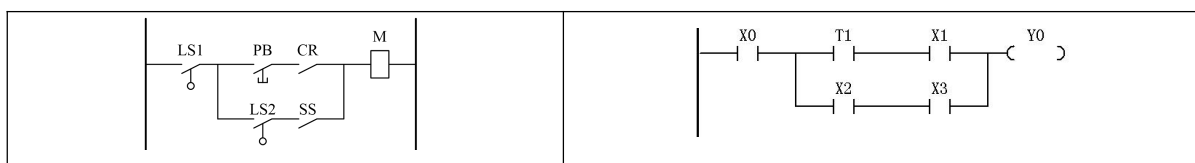


图 4-1 电气控制图和梯形图的等效关系

梯形图基本的编程元素

梯形图根据电气（继电器）控制图的原理，抽象出几种基本编程元素：

1. 左母线：对应于电气控制图中的控制母线，为控制回路提供控制电源。
2. 连接线（—|）：代表电气控制图的电气连接，用于导通彼此相连的其他元件。
3. 触点（|+|）：代表电气控制图中的输入接点，控制着回路中的控制电流的通断，决定着控制电流的方向。触点的并联、串联的连接关系，实质上是代表了控制电路输入逻辑的运算关系，控制着能流的传递。
4. 线圈（(Q)）：代表电气控制图中的继电器输出。
5. 功能块（□）：又称应用指令，对应于电气控制图中连接的完成特殊功能的执行机构或功能器件，功能块可以完成特定的控制功能或控制计算功能（例如数据传输、数据运算、计时器、计数器等）。

能流

能流在梯形图程序中是一个很重要的概念，能流用于驱动线圈元件和应用指令，与电气控制图中驱动线圈输出和机构执行的控制电流相类似。

在梯形图中线圈或应用指令前端必须接入能流，当能流有效时，线圈元件才能输出，应用指令才能被有效执行。

下图演示了梯形图中的能流传递及能流对线圈或功能块的驱动作用。

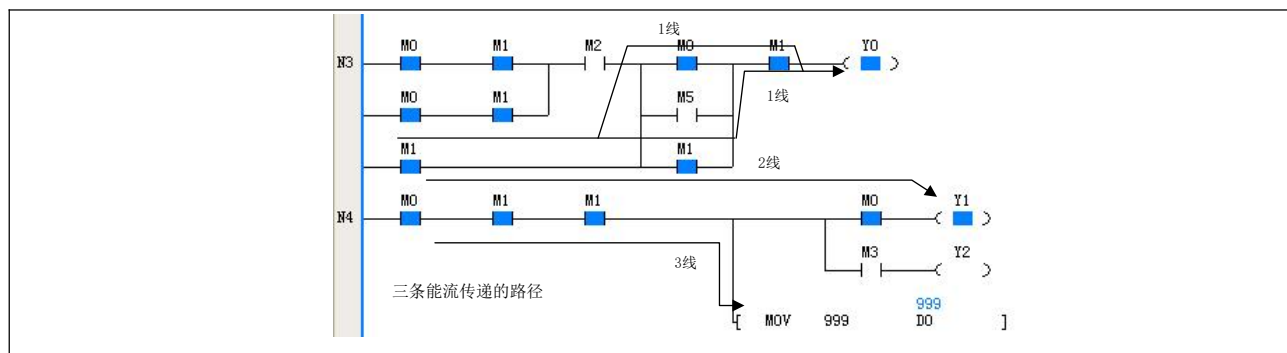


图 4-2 能流传递及能流的驱动作用

4.1.2 指令列表 (IL)

指令列表是文本化的用户程序，是用户编写的指令序列集。

存储在 PLC 主模块中用于执行的用户程序，实际上是主模块可识别的指令序列，系统逐条执行序列中的每一条指令，实现用户程序的控制功能。

下图是梯形图转换成指令列表的示例。

梯形图	指令列表
	LD X0
	OR X1
	AND X14
	MPS
	OUT Y0
	AND X1
	OUT Y1
	MPP
	AND X2
	MPS
	OUT Y2
	AND X3
	AND X4
	OUT Y3
	MRD
	LD X5
	AND X6
	LD X7
	AND X10
	ORB
	ANB
	OUT Y4
	MPP
	OUT Y5

4.1.3 顺序功能图（SFC）

顺序功能图是一种图形化的用户程序框架设计语言，通常用于实现顺序控制功能。

顺序控制是指可划分多个工序（处理步骤），并按一定工作顺序进行处理的控制过程。

按顺序功能图设计出的用户程序，程序结构与实际的顺序控制过程相符合，比较直观清晰。

下图是一个简单顺序功能图的示例。

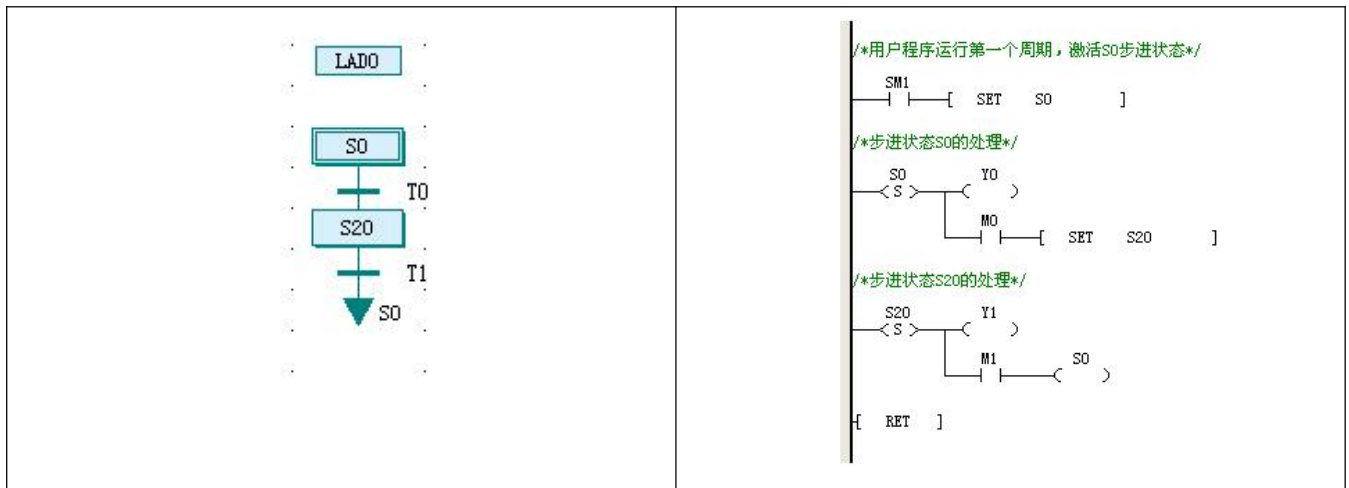


图 4-3 顺序功能图示例

4.2 程序要素

用户程序、系统块和数据块，被称程序要素。用户可以通过编程修改三种程序要素。

4.2.1 用户程序

用户程序是用户所编写的程序代码，经过编译后形成可执行的指令序列，下载到控制器，由控制器执行用户程序的控制功能。

用户程序由主程序、子程序、中断程序三类程序体（POU）构成。

主程序（MAIN）

主程序是用户程序的主体和框架。系统处于运行状态时，主程序被循环执行。

任一用户程序有且只有一个主程序。

子程序（SBR）

子程序是一段结构和功能上独立的、可以被其他程序体调用的用户程序，通常具有调用操作数接口，只在被调用时被执行。

一个用户程序可以没有子程序，也可包含一个或多个子程序。

中断程序（INT）

中断程序是处理特定中断事件的用户程序段。某个特定的中断事件总是对应于特定的中断程序。

只要中断事件发生，一个正常的扫描周期将被打断，用户程序流自动跳转到中断程序执行，直至执行到中断返回指令系统才又恢复到正常的扫描周期流程上。

一个用户程序可以没有中断程序，也可包含一个或多个中断程序。

4.2.2 系统块

系统块中包含多个系统配置选项，用户通过修改、编译、下载系统块，达到配置主模块运行模式的目的。

详细的系统配置项的使用方法请参阅本手册 2.2.1 系统块，或参阅《AH-station 编程软件用户手册》中关于系统块的相关介绍。

4.2.3 数据块

数据块包含 D 元件设置数值，当数据块下载到控制器中，指定 D 元件将被赋予设置值，从而达到批量设定 D 元件值的目的。

如果控制器被配置为在数据块有效工作模式下，在用户程序运行前，数据块中指定的 D 元件将按数据块中内容进行初始化。

4.3 程序的块注释和变量的注释

4.3.1 块注释

在编程时，可以在程序中添加块注释，块注释以文字方式来说明相关的某段程序。每个块注释都会占用一整行的空间。

在程序中右击鼠标，打开右键菜单，选中**行插入**，则可在程序中插入空行。一般应该先以空行作为程序块分界。

需要输入块注释时，首先选择一个空行，然后从右键菜单中选择**插入块注释**，如下图所示：



图 4-4 添加块注释

在弹出的块注释对话框中输入注释文字并确认，如下图所示：

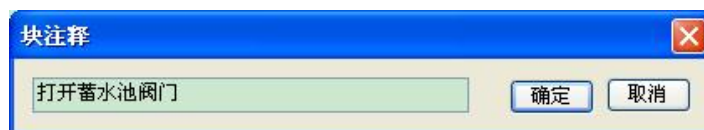


图 4-5 块注释输入对话框

软件会自动在所输入的文字两侧分别加上“/*”和“*/”，并以绿色显示，如下图所示：



图 4-6 程序中的块注释

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

4.3.2 变量的注释

变量可以在全局变量表和局部变量表中定义（具体定义方法请参见 2.2.3 全局变量表和 4.4.3 子程序变量表定义），正确定义的变量可以在梯形图中使用。需要使用某个地址时，可以用代表此地址的变量名来代替，以增强程序的可读性。下图是全局变量表中定义的变量：

MAIN		全局变量表		
	变量名	变量地址	注释	
1	水压	X0	水压标记	
2	计时	T0	计时30分钟	
3	启动1	M0		
4	启动2	M1		
5	停止1	M2		
6	告警灯	Y0	水量告警灯	
7	阀门	Y1	控制阀门	

图 4-7 全局变量表中定义的变量

符号编址

当使用了定义变量后，可以通过选择**符号编址**菜单在变量名和元件地址之间切换，下图分别展示了两显示模式下的同一梯形图程序：

未选中符号编址的状态：

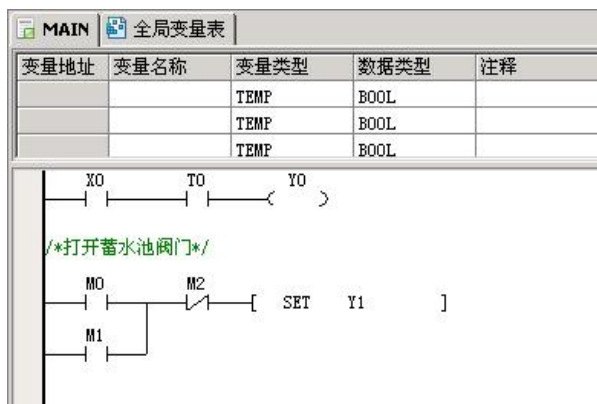


图 4-8 未选中符号编址的状态

采用了符号编址的状态：

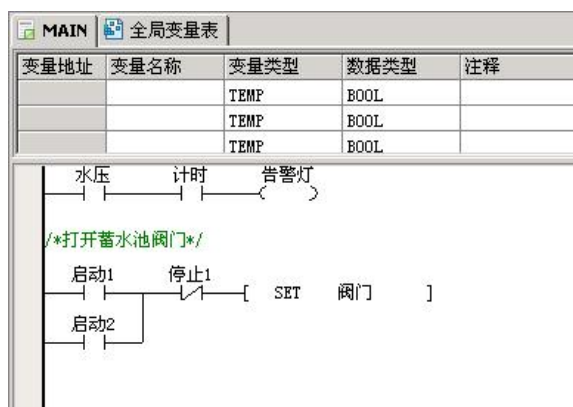


图 4-9 采用了符号编址的状态

元件注释

可以通过选择**元件注释**菜单控制是否在梯形图程序中显示元件注释。以下是在显示元件注释时的梯形图程序：

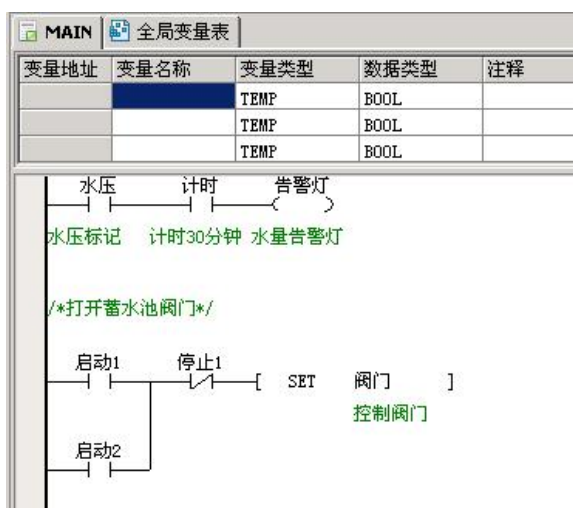


图 4-10 显示元件注释时的梯形图程序

4.4 子程序

4.4.1 子程序概念

子程序是一种独立的程序体，可以被主程序或其他子程序调用。子程序是用户程序的可选构成部分。

使用子程序编写用户程序具有以下优点：

1. 可减少用户程序的尺寸，功能相同的重复用户程序代码段，可以编写成为一个子程序反复调用。
2. 使程序的结构更加清晰，特别是可以简化主程序结构。
3. 提高用户程序的可移植性。

4.4.2 子程序的使用注意事项

在编写或调用子程序时，应注意以下事项：

1. 支持子程序的镶嵌调用，最大镶嵌调用层数为 6 层。

下例演示了一个合法的 6 层镶嵌调用关系：

MAIN→SBR1→SBR2→SBR3→SBR4→SBR5→SBR6。

（→代表使用 CALL 指令调用相应的子程序）

2. 不支持子程序的递归调用和循环调用。

如下两种示例，演示了非法的子程序调用关系：

- MAIN→SBR0→SBR0（递归调用，非法）
- MAIN→SBR0→SBR1→SBR0（循环调用，非法）

3. 一个用户程序中可最多可定义 64 个子程序。
4. 一个子程序的变量表内最多可定义 16 个位型和 16 个字型变量。
5. 子程序调用时，应注意 CALL 指令中填写的操作数的属性应与子程序的变量表中定义的变量属性相匹配，编译器将检查匹配的正确性。
6. 中断程序中不允许调用任何子程序。

4.4.3 子程序变量表定义

子程序变量表

子程序变量表作用是声明子程序的接口参数和局部变量（统称为变量），并规定它们的使用属性。

子程序变量的属性项说明

子程序的接口参数和局部变量（统称为变量）具有以下属性：

1. 变量地址

每个子程序接口参数或局部变量都被分配一个固定的 LM 元件或 V 元件地址。该地址是编程软件根据变量的数据类型，按地址连续的原则自动分配给子程序接口参数或局部变量的。

2. 变量名称

可以为子程序接口参数或局部变量取一个变量名称（别名），在程序中使用变量名称引用的方式使用变量。

3. 变量类型

子程序接口参数或局部变量分为 IN 型、OUT 型、IN_OUT 型、TEMP 型：

- IN 型变量在子程序被调用时，用于传递子程序的输入值。
- OUT 型变量在子程序返回时，用于为传递子程序调用返回值。
- IN_OUT 型变量在子程序被调用时，用于传递输入值。在子程序返回时，用于传递调用返回值。
- TEMP 型变量只作为子程序范围内有效的局部变量使用。

4. 变量数据类型

变量数据类型属性规定了变量的数据宽度及数据范围，下表列出了变量数据类型的种类：

表 4-2 变量数据类型的种类

变量数据类型	数据类型说明	占用 LM/V 元件地址
BOOL	位型变量	占用 1 个 LM 元件地址
INT	符号整数型变量	占用 1 个 V 元件地址
DINT	符号长整数型变量	占用 2 个连续 V 元件地址
WORD	字型变量	占用 1 个 V 元件地址
DWORD	双字型变量	占用 2 个连续 V 元件地址
REAL	浮点型变量	占用 2 个连续 V 元件地址

4.4.4 子程序参数传递

在主程序里调用子程序时，如果子程序里定义了局部输入输出变量，那么子程序的接口参数里需填写相应的数值或者全局/临时变量元件。注意局部变量与接口参数的数据类型应一致。

4.4.5 子程序的使用示例

下面用一个示例说明如何编写及调用子程序

示例功能介绍

在主程序中调用子程序 SBR_1，让子程序 SBR_1 完成两个整型常数的相加运算（3+2），运算结果 5 赋予 D0。

示例操作过程

步骤 1：在工程中创建一个子程序，为子程序命名 SBR_1。

步骤 2：编写子程序 SBR_1

1. 在子程序 SBR_1 的变量表中建立子程序的调用操作数接口。

1) 定义变量 1：取变量名为 IN1，该变量是 IN 型参数，被当成 INT 型数据使用，其被顺序分配一个 V 元件地址 V0。

2) 定义变量 2：取变量名为 IN2，该变量是 IN 型参数，被当成 INT 型数据使用，其被顺序分配一个 V 元件地址 V1。

3) 定义变量 3：取变量名为 OUT1，该变量是 OUT 型参数，被当成 INT 型数据使用，其被顺序分配一个 V 元件地址 V2。

2. 编写子程序 SBR_1 的实现代码：

```
LD SM0
```

```
ADD #IN1 #IN2 #OUT1
```

下图演示了子程序 SBR_1 的编写过程：

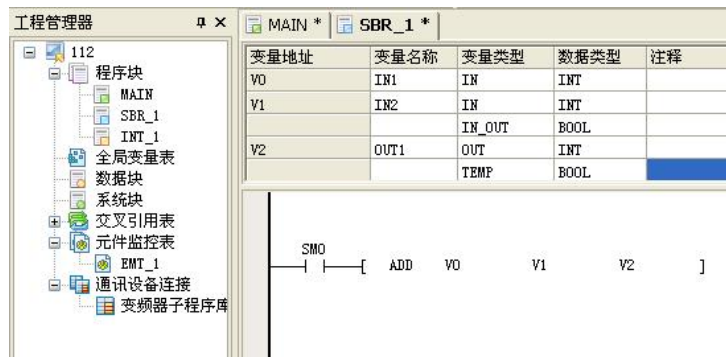


图 4-11 子程序 SBR_1 的编写过程

步骤 3：编写主程序，调用子程序

在主程序中使用 CALL 指令调用子程序 SBR_1。

整个主程序的代码如下：

```
LD M0  
CALL SBR_1 3 2 D0
```

可使用参数传递对应关系表，填写调用子程序时带入或返回的参数。

- 带入参数 IN1 传递了常整数 3
- 带入参数 IN2 传递了常整数 2
- 返回值 OUT1 传递给 D0

见下图：

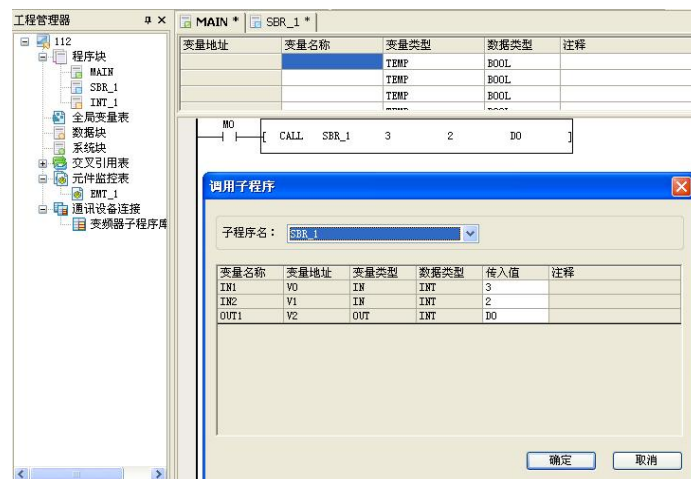


图 4-12 调用子程序

步骤 4：编译、下载、运行用户程序，验证子程序的逻辑正确性。

示例执行结果

当 M0=ON 时，SBR_1 子程序被调用，带入操作数 IN1，IN2 被传递值 3、2 后，完成相加运算返回值 5，最后 D0=5。

4.5 指令通用说明

4.5.1 指令的操作数

指令的操作数可分为以下两类。

- 源操作数：指令对其数据进行读取，用于运算处理。在指令说明中用 **S** 来表示，多于 1 个的，用 **S₁**、**S₂**、**S₃** 等表示。
- 目的操作数：指令对目的操作数进行控制或输出。在指令说明中用 **D** 来表示，多于 1 个的，用 **D₁**、**D₂** 等表示。

操作数有位元件，也有单字元件或双字元件，还有常数。具体请参阅第五章和第六章相关指令的详细说明。

4.5.2 标志位

指令运算可能会对三种标志位产生影响。

零标志 SM180

指令运算产生零结果的，置零标志。

进位标志 SM181

指令运算有进位的，置进位标志。

借位标志 SM182

指令运算有借位的，置借位标志。

4.5.3 指令的使用限制

一些指令的应用有一些限制条件，以下列举了其中的一部分。具体请参阅相关指令详细说明。

硬件资源独占

有的指令执行时，会占用硬件资源，与此硬件资源相关的其它指令就不能同时使用。

例如：高速计数指令、SPD 测频指令等，任意一条这种指令都会占用 X0~X7 的某些输入点。这些指令同时使用都会相互冲突。

时间独占

有些指令执行时会持续一个时间段。因此使用这些指令时，需要保证指令有足够的时间来完成功能，在系统运行某一确定时刻只能执行 1 条。

例如：由于通讯的时间性，对自由口发送指令 XMT，同一时刻只能执行 1 条；同样，自由口接收指令 RCV 也是如此。每次 Modbus 指令在执行时，也会有一段时间的独占情况。对高速输出指令、定位指令、变频器指令等也都有同样的情况。

指令应用范围限制

有些指令的使用范围受限，在某种环境下不能使用。

例如：MC/MCR 指令对不能在 SFC 顺序功能图编程的步进状态中使用。

第五章 基本指令说明

本章详细介绍了 APLC 系列小型 PLC 的基本指令，内容包括指令格式（形式）、操作数、影响标志位、功能、示例、时序图等。

5.1 触点逻辑指令	76
5.1.1 LD：常开触点指令	76
5.1.2 LDI：常闭触点指令	76
5.1.3 AND：常开触点与指令	77
5.1.4 ANI：常闭触点与指令	77
5.1.5 OR：常开触点或指令	78
5.1.6 ORI：常闭触点或指令	78
5.1.7 OUT：线圈输出指令	79
5.1.8 ANB：能流块与指令	79
5.1.9 ORB：能流块或指令	80
5.1.10 MPS：输出能流入栈指令	80
5.1.11 MRD：读输出能流栈顶值指令	81
5.1.12 MPP：输出能流栈出栈指令	81
5.1.13 EU：上升沿检测指令	82
5.1.14 ED：下降沿检测指令	82
5.1.15 INV：能流取反指令	83
5.1.16 SET：线圈置位指令	83
5.1.17 RST：线圈清除指令	84
5.1.18 NOP：空操作指令	84
5.2 主控指令	84
5.2.1 MC：主控指令	84
5.2.2 MCR：主控清除指令	85
5.3 SFC 指令	86
5.3.1 STL：SFC 状态装载指令	86
5.3.2 SET Sxx：SFC 状态转移	86
5.3.3 OUT Sxx：SFC 状态跳转	87
5.3.4 RST Sxx：SFC 状态清除	87
5.3.5 RET：SFC 程序段结束	87
5.4 计时器指令	88
5.4.1 TON：接通延时计时指令	88
5.4.2 TONR：记忆型接通延时计时指令	89
5.4.3 TOF：断开延时计时器指令	90
5.4.4 TMON：不重触发单稳计时指令	91

5.5 计数器指令	92
5.5.1 CTU：16 位增计数器指令	92
5.5.2 CTR：16 位循环计数指令	93
5.5.3 DCNT：32 位增减计数指令	94

5.1 触点逻辑指令

5.1.1 LD：常开触点指令

<div>梯形图：</div> <div></div>								适用机型		APLC-TCP					
								影响标志位							
指令列表：LD (S)								步长		1					
操作数		类型	适用软元件											变址	
S		BOOL	X	Y	M	S	LM	SM				C	T		

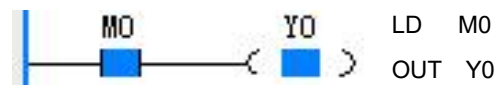
操作数说明

S：源操作数

功能说明

连接左母线，用于接通（状态 ON）或断开（状态 OFF）能流。

使用示例



当 M0 为 ON 时，Y0 输出 ON。

注意事项

对于 AT100 系列的触点逻辑指令，当操作数为 M1536~M2047 时，步长为各指令所指步长加 1。

5.1.2 LDI：常闭触点指令

<div>梯形图：</div> <div></div>										适用机型		APLC-TCP					
										影响标志位							
指令列表：LDI (S)										步长		1					
操作数		类型	适用软元件													变址	
S		BOOL	X	Y	M	S	LM	SM				C	T				

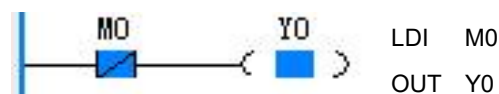
操作数说明

S：源操作数

功能说明

连接左母线，用于接通（状态 OFF）或断开（状态 ON）能流。

使用示例



当 M0 为 OFF 时，Y0 输出 ON。

注意事项

对于 APLC-TCP 系列的触点逻辑指令，当操作数为 M1536~M2047 时，步长为各指令所指步长加 1。

5.1.3 AND：常开触点与指令

梯形图： 								适用机型		APLC-TCP						
								影响标志位								
指令列表：AND (S)								步长		1						
操作数		类型		适用软元件										变址		
S		BOOL		X	Y	M	S	LM	SM				C	T		

操作数说明

S：源操作数

功能说明

将指定触点 (**S**) 的 ON/OFF 状态和当前能流作“与”运算后，赋给当前能流。

使用示例



```
LD  M0
AND M1
OUT Y0
```

当 M0 为 ON 且 M1 为 ON 时，Y0 输出 ON。

5.1.4 ANI：常闭触点与指令

梯形图： <div></div>								适用机型		APLC-TCP						
								影响标志位								
指令列表： ANI (S)								步长		1						
操作数		类型		适用软元件										变址		
S		BOOL		X	Y	M	S	LM	SM				C	T		

操作数说明

S：源操作数

功能说明

将指定的触点 (**S**) 的 ON/OFF 状态取反后，与当前能流值作“与”运算计算后，赋给当前能流。

使用示例



```
LD  M0
ANI M1
OUT Y0
```

当 M0 为 ON 且 M1 为 OFF 时，Y0 输出 ON。

5.1.5 OR：常开触点或指令

<div>梯形图：</div> <div></div>										适用机型		APLC-TCP					
										影响标志位							
指令列表：OR (S)										步长		1					
操作数		类型	适用软元件													变址	
S		BOOL	X	Y	M	S	LM	SM				C	T				

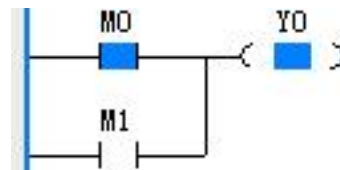
操作数说明

S：源操作数

功能说明

将指定触点 (**S**) 的 ON/OFF 状态和当前能流作“或”运算后，赋给当前能流。

使用示例



```
LD  M0
OR  M1
OUT Y0
```

当 M0 或 M1 为 ON 时，Y0 输出 ON。

5.1.6 ORI：常闭触点或指令

<div>梯形图：</div> <div></div>										适用机型		APLC-TCP					
										影响标志位							
指令列表：ORI (S)										步长		1					
操作数		类型		适用软元件										变址			
S		BOOL		X	Y	M	S	LM	SM				C	T			

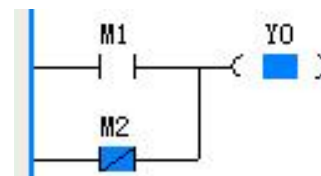
操作数说明

S：源操作数

功能说明

将指定触点 (**S**) 的 ON/OFF 状态取反后和当前能流值作“或”运算后，赋给当前能流。

使用示例



```
LD  M0
ORI M2
OUT Y0
```

当 M1 为 ON 或 M2 为 OFF 时，Y0 输出 ON。

5.1.7 OUT：线圈输出指令

<div>梯形图：</div>										适用机型		APLC-TCP					
										影响标志位							
指令列表： OUT (S)										步长		1					
操作数		类型	适用软元件											变址			
S		BOOL	X	Y	M	S	LM	SM				C	T				

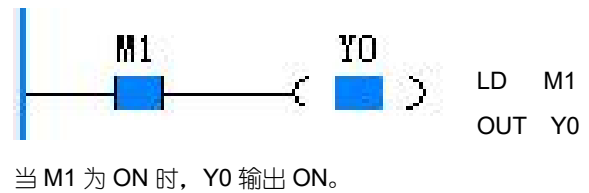
操作数说明

S：源操作数

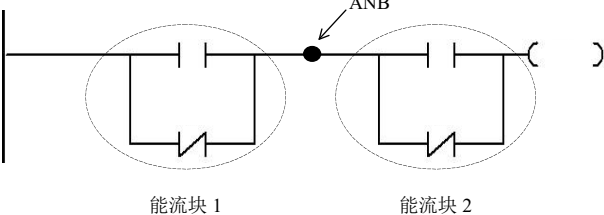
功能说明

将当前能流值赋给指定的线圈 (**D**)。

使用示例



5.1.8 ANB：能流块与指令

梯形图：										适用机型		APLC-TCP				
										影响标志位						
指令列表：ANB（S）										步长		1				
操作数		类型		适用软元件										变址		
S		BOOL		X	Y	M	S	LM	SM				C	T		

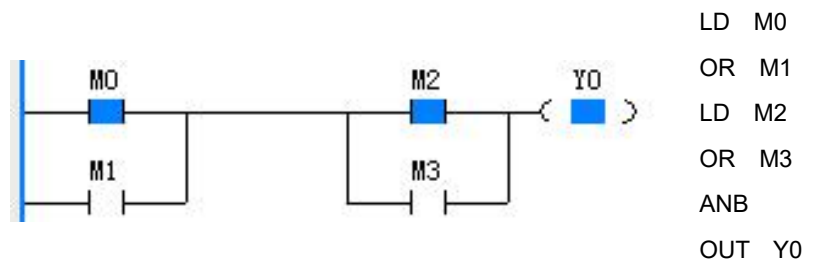
操作数说明

S：源操作数

功能说明

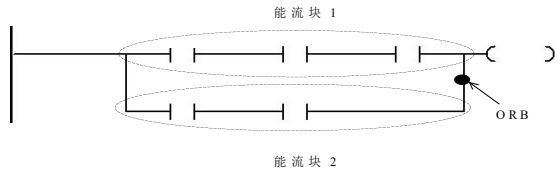
将两个能流块能流值作“与”运算，
赋给当前能流。

使用示例



当 M0 和 M1 中有一个为 ON，并且 M2 和 M3 中有一个为 ON 时，Y0 输出 ON。

5.1.9 ORB：能流块或指令

梯形图：												适用机型	APLC-TCP
												影响标志位	
指令列表：ORB (S)										步长	1		
操作数	类型	适用软元件										变址	
S	BOOL	X	Y	M	S	LM	SM				C	T	

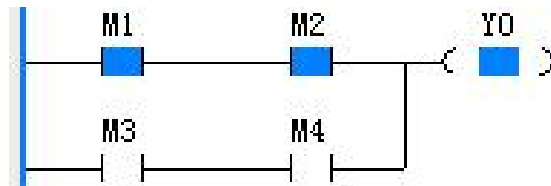
操作数说明

S：源操作数

功能说明

将两个能流块的能流值作“或”运算，
赋给当前能流。

使用示例



```
LD  M1
AND M2
LD  M3
AND M4
ORB
OUT Y0
```

当 M1 和 M2 都为 ON 或者 M3 和 M4 都为 ON 时，Y0 输出 ON。

5.1.10 MPS：输出能流入栈指令

梯形图：		适用机型	APLC-TCP
		影响标志位	
指令列表：MPS		步长	1

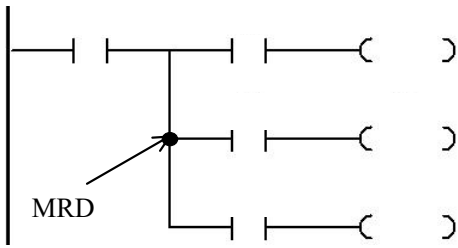
功能说明

将当前能流值压栈保存，供后续的输出分支的能流计算使用。

注意事项

在一个梯形图网络中禁止连续 8 次以上使用 MPS（中间无 MPP 指令），否则将导致能流输出栈的溢出。

5.1.11 MRD：读输出能流栈顶值指令

梯形图： 	适用机型	APLC-TCP
	影响标志位	
指令列表：MRD	步长	1

功能说明

将能流输出栈的栈顶值赋给当前能流。

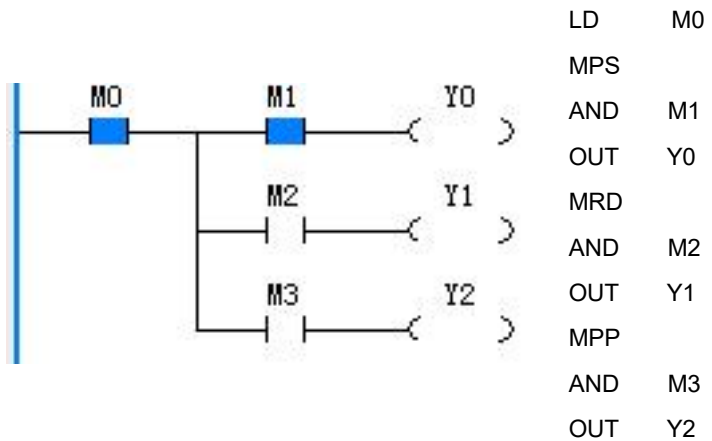
5.1.12 MPP：输出能流栈出栈指令

梯形图： 	适用机型	APLC-TCP
	影响标志位	
指令列表：MPP	步长	1

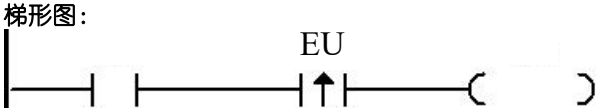
功能说明

对能流输出栈进行弹栈操作，出栈值赋给当前能流。

使用示例



5.1.13 EU：上升沿检测指令

梯形图： 	适用机型	APLC-TCP
	影响标志位	
指令列表：EU	步长	2

功能说明

比较本次扫描与上次扫描输入能流的变化。能流有上升沿变化时（OFF→ON），本扫描周期内输出有效。

使用示例



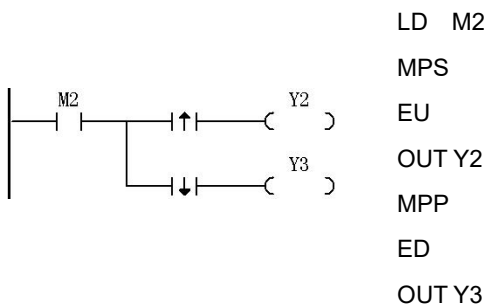
5.1.14 ED：下降沿检测指令

梯形图： 	适用机型	APLC-TCP
	影响标志位	
指令列表：ED	步长	2

功能说明

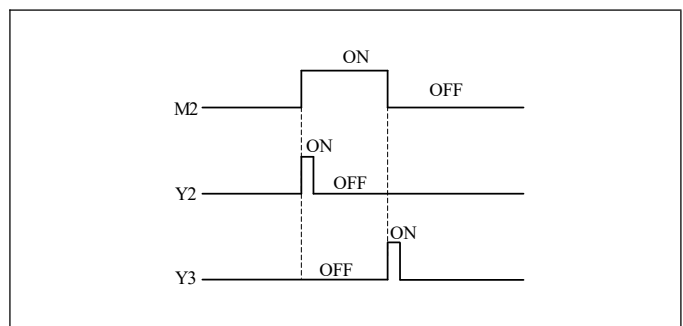
比较本次扫描与上次扫描输入能流的变化。能流有下降沿变化时（ON→OFF），本扫描周期内输出有效。

使用示例



- 在连续两个扫描周期，M2 触点的状态分别是 OFF 和 ON，EU 指令检测到上升沿变化，使得 Y2 输出一个扫描周期宽度的 ON 状态。
- 在连续两个扫描周期，M2 触点的状态分别是 ON 和 OFF，ED 指令检测到下降沿变化，使得 Y3 输出一个扫描周期宽度的 ON 状态。

示例的时序图

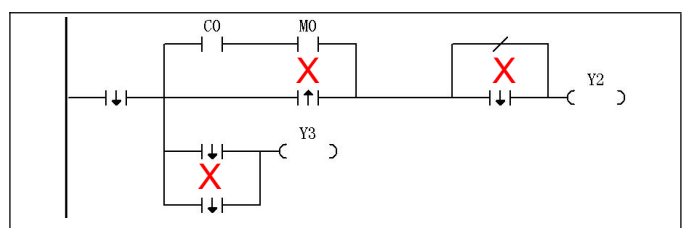


注意事项

在梯形图中，上升沿触点或下降沿触点指令应与其它触点元件串联使用，不能和其它触点元件并联使用。

在梯形图中，上升沿触点或下降沿触点指令不能直接接左能流母线。

以下是 EU/ED 指令在梯形图中错误使用的示例：



5.1.15 INV：能流取反指令

梯形图： 	适用机型	APLC-TCP
	影响标志位	
指令列表：INV	步长	1

功能说明

将当前能流值取反后，再赋给当前能流。

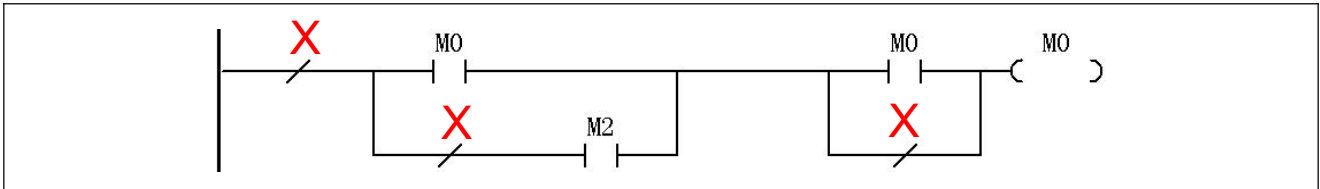
注意事项

在梯形图中能流取反指令应和触点元件串联使用，不能和其它触点元件并联使用。

INV 不可作为输入并联支路第一个指令使用。

在梯形图中能流取反指令不能直接接左能流母线。

以下是 INV 指令在梯形图中错误使用的示例：



5.1.16 SET：线圈置位指令

<div>梯形图：</div> <div></div>										适用机型		APLC-TCP				
										影响标志位						
指令列表：SET (S)										步长		1				
操作数		类型	适用软元件											变址		
S		BOOL		Y	M	S	LM	SM				C	T			

操作数说明

S：源操作数

功能说明

当能流有效时，**D** 指定的位元件将被置位。

使用示例



5.1.17 RST：线圈清除指令

梯形图： 										适用机型		APLC-TCP			
										影响标志位					
指令列表：RST (S)										步长		1			
操作数		类型	适用软元件										变址		
S		BOOL		Y	M	S	LM	SM				C	T		

操作数说明

S：源操作数

功能说明

当能流有效时，指定位元件 (**D**) 将被清零。

使用示例



注意事项

如果 **D** 为 **C** 元件，所对应计数值也将被清零；如果 **D** 为 **T** 元件，所对应计时值也将被清零。

5.1.18 NOP：空操作指令

梯形图： 		适用机型	APLC-TCP									
		影响标志位										
指令列表：NOP		步长	1									

功能说明

该指令不产生任何动作。

注意事项

在梯形图中，该指令不能直接连接左能流母线。

5.2 主控指令

5.2.1 MC：主控指令

梯形图： 		适用机型	APLC-TCP									
		影响标志位										
指令列表：MC (S)		步长	3									
操作数	类型	适用软元件										变址
S	INT	常数										

操作数说明

S：源操作数

5.2.2 MCR：主控清除指令

梯形图： 		适用机型	APLC-TCP											
		影响标志位												
指令列表：MCR (S)		步长	1											
操作数	类型	适用软元件												变址
S	INT	常数												

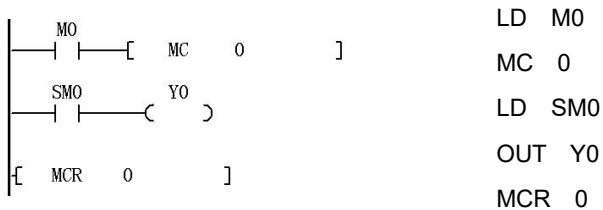
操作数说明

S：源操作数

功能说明

- MC 与 MCR 指令匹配成一个 MC-MCR 结构。MC 指令代表着一个 MC-MCR 结构的开始，其操作数 **S** 为 MC-MCR 结构的标号，其值为 0~7 之间的一个常数。MCR 代表着一个 MC-MCR 结构的结束。
- 当 MC 指令前的能流有效时，执行 MC-MCR 结构中间的指令。
- 当 MC 指令前的能流无效时，MC-MCR 结构中间的指令被跳过不被执行，程序直接跳转到该结构后执行，并且该结构中的 OUT、TON、TOF、PWM、HCNT、PLSY、PLSR、DHSCS、SPD、DHSCI、DHSCR、DHSZ、DHST、DHSP、BOUT 所对应的目的操作数将被清除。

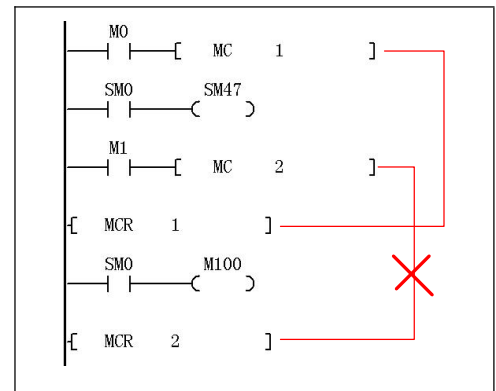
使用示例



当 M0 = ON 时，MC 0 - MCR 0 结构内的指令将被执行，Y0 = ON。当 M0 = OFF 时，MC 0 - MCR 0 结构内的指令将不执行，同时结构内的 OUT 指令的目的操作数所指定的位元件 Y0 被清除 Y0 = OFF。

注意事项

- 在梯形图中，MCR 指令必须直接连接左能流母线。
- 在梯形图中，MCR 指令不能并接或串接其它指令。
- 多个不同编号的 MC-MCR 结构可以镶嵌使用，但镶嵌层数不能超过 7 层。而同一编号的 MC-MCR 结构禁止镶嵌使用。
- 两个 MC-MCR 结构不能交叉使用，如下图所示使用方法是非法的：



注意：在 SFC 编程中不能使用。

5.3 SFC 指令

5.3.1 STL：SFC 状态装载指令

梯形图： 								适用机型	APLC-TCP						
								影响标志位							
指令列表：STL (S)								步长	3						
操作数	类型	适用软元件													变址
S	BOOL						S								

操作数说明

S：源操作数

功能说明

1. 代表一个步进状态 (**S**) 处理的开始。
2. 如果该步进状态有效 (ON)，其内置指令将执行。
3. 如果该步进状态是由有效变为无效 (下降沿变化)，其内置指令序列将不被执行，并且内置的 OUT、TON、TOF、PWM、HCNT、PLSY、PLSR、DHSCS、SPD、DHSCI、DHSCR、DHSZ、DHST、DHSP、BOUT 所对应的目的操作数将被清除。
4. 如果该步进状态无效，其内置的指令序列将不被执行。
5. 连续的 STL 指令 (STL 元件的串联) 代表定义了一个并行汇合结构，STL 指令最大连续使用的次数为 16 次 (并行分支汇合结构的最大分支数为 16)。

5.3.2 SET Sxx：SFC 状态转移

梯形图： 								适用机型	APLC-TCP						
								影响标志位							
指令列表：SET (D)								步长	3						
操作数	类型	适用软元件													变址
D	BOOL						S								

操作数说明

D：目的操作数

功能说明

当能流有效时，将指定步进状态 (**D**) 置为有效，同时使当前有效的步进状态置为失效，完成步进状态转移的动作。

5.3.3 OUT Sxx: SFC 状态跳转

梯形图: 		适用机型	APLC-TCP											
		影响标志位												
指令列表: OUT (D)		步长	3											
操作数	类型	适用软元件												变址
D	BOOL						S							

操作数说明

D: 目的操作数

功能说明

当能流有效时, 将指定步进状态 (D) 置为有效, 同时使当前有效的步进状态置为失效, 完成步进状态跳转动作。

5.3.4 RST Sxx: SFC 状态清除

梯形图: 		适用机型	APLC-TCP											
		影响标志位												
指令列表: RST (D)		步长	3											
操作数	类型	适用软元件												变址
D	BOOL						S							


操作数说明

D: 目的操作数

功能说明

与当能流有效时, 将指定的步进状态 (D) 置为无效。

5.3.5 RET: SFC 程序段结束

梯形图: 		适用机型	APLC-TCP											
		影响标志位												
指令列表: RET		步长	1											

指令功能描述

标志一段顺序功能图程序的结束。

注意事项

只能在主程序中使用。

5.4 计时器指令

5.4.1 TON：接通延时计时指令

梯形图：										适用机型	APLC-TCP				
										影响标志位					
指令列表：TON (D) (S)										步长	5				
操作数	类型	适用软元件													变址
D	INT											T			
S	INT	常数	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	√

操作数说明

D：目的操作数

S：源操作数

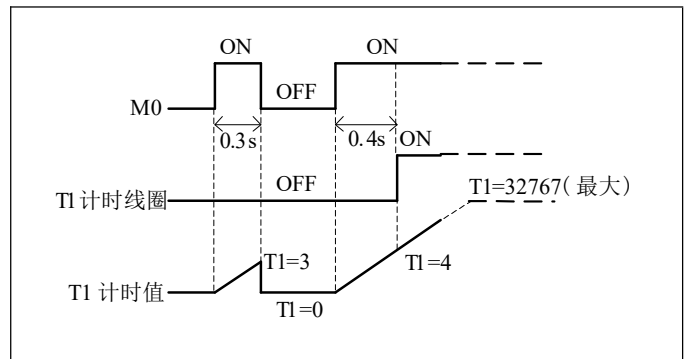
功能说明

1. 能流有效，且计时值 $< 32,767$ 时，所指定的 T 元件 (**D**) 计时（计时值随着走时而累加）。当计时值到达 32,767 后，计时值将保持为 32,767 不变。
2. 当计时值 \geq 预设值 (**S**) 时，所指定的 T 元件的计时线圈输出为 ON。
3. 当能流为 OFF 时，停止计时，计时值清为零，计时线圈输出为 OFF。
4. 系统第一次执行该指令，将把所指定的 T 元件的计时线圈值清为 OFF，计时值清零。

使用示例



示例的时序图



5.4.2 TONR：记忆型接通延时计时指令

梯形图：										适用机型	APLC-TCP				
										影响标志位					
指令列表：TONR (D) (S)										步长	5				
操作数	类型	适用软元件													变址
D	INT											T			
S	INT	常数	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	√

操作数说明

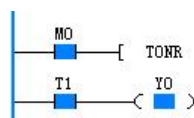
D：目的操作数

S：源操作数

功能说明

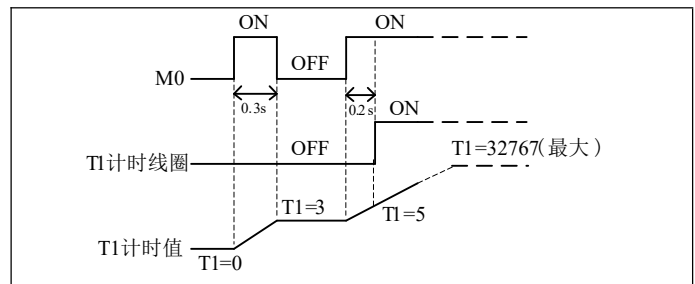
1. 能流有效，且计时值 $< 32,767$ 时，所指定的 T 元件 (**D**) 计时，计时值随着走时递增。当计时值到达 32,767 后，计时值将保持为 32,767 不变。
2. 当计时值 \geq 预设值 (**S**) 时，所指定的 T 元件的计时线圈输出为 ON。
3. 当能流为 OFF 时，停止计时，计时线圈与计时值保持当前计时值不变。

使用示例



```
LD M0
TONR T1 5
LD T1
OUT Y0
```

示例的时序图



5.4.3 TOF：断开延时计时器指令

梯形图： 								适用机型		APLC-TCP					
								影响标志位							
指令列表：TOF (D) (S)								步长		5					
操作数		类型	适用软元件											变址	
D	INT											T			
S	INT	常数	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	√

操作数说明

D：目的操作数

S：源操作数

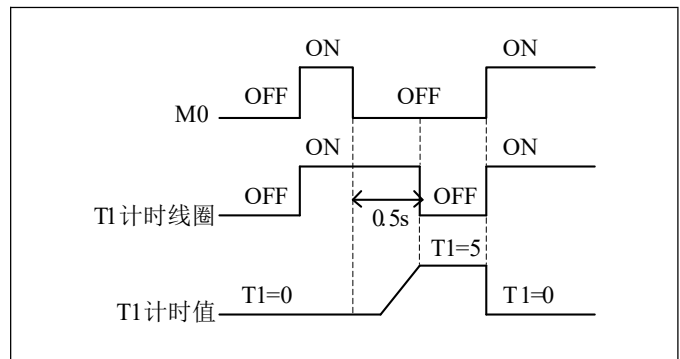
功能说明

1. 当能流有 ON→OFF 变化（下降沿）后，指定计时器 **T (D)** 启动计时。
2. 能流为 OFF，如指定计时器 **T** 已启动计时，继续保持计时。直到计时值等于预设值 (**S**)，所指定的 **T** 元件的计时线圈输出为 OFF，此后计时值将保持为预设值不再变化。
3. 如计时未启动，即使能流输入为 OFF 也不计时。
4. 当能流为 ON 时，停止计时，计时值清为零，计时线圈输出为 ON。

使用示例



示例的时序图



5.4.4 TMON：不重触发单稳计时指令

梯形图： 								适用机型	APLC-TCP						
								影响标志位							
指令列表：TMON (D) (S)								步长	5						
操作数	类型	适用软元件													变址
D	INT											T			
S	INT	常数	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	√

操作数说明

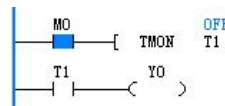
D：目的操作数

S：源操作数

功能说明

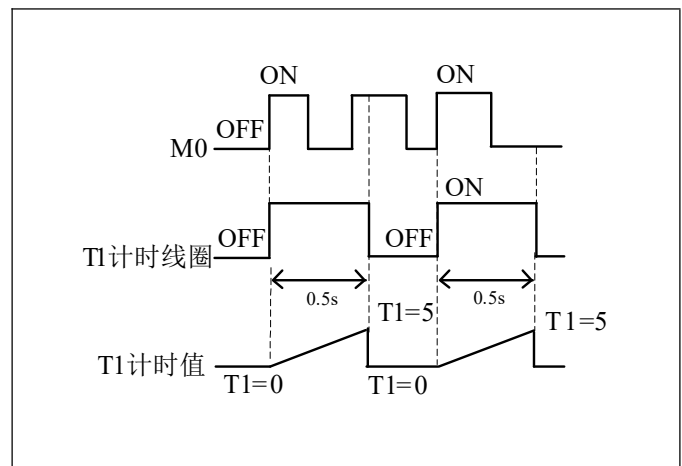
- 当输入能流有 OFF→ON 变化（上升沿）时，且处于未计时状态，启动指定的计时器 **T (D)** 计时（由当前值开始），计时状态（计时状态长度由 **S** 确定）下，保持计时线圈输出为 **ON**。
- 在计时状态（计时长度由 **S** 确定），不论能流如何变化，保持计时，计时线圈输出保持为 **ON**。
- 当计时值到达时，停止计时，计时值清为零值，线圈输出清为 **OFF**。

使用示例



```
LD M0
TMON T1 5
LD T1
OUT Y0
```

示例的时序图



5.5 计数器指令

5.5.1 CTU：16 位增计数器指令

梯形图：										适用机型	APLC-TCP				
										影响标志位					
指令列表：CTU (D) (S)										步长	5				
操作数	类型	适用软元件													变址
D	INT											C			
S	INT	常数	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	√

操作数说明

D：目的操作数

S：源操作数

功能说明

1. 当能流有 OFF→ON 变化（上升沿）时，指定的 16 位计数器 C (**D**) 计数值增一。
2. 当计数值达到 32,767 时，计数值保持不变。
3. 当计数值大于等于计数预设值 (**S**) 时，计数线圈置为 ON。

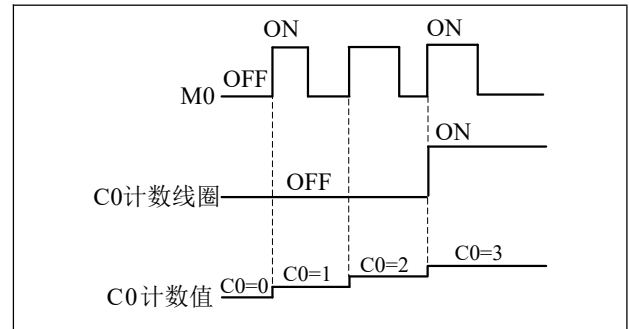
注意事项

(**D**) 所指定的 16 位计数器 C 的地址应在 C0~C199 之内。

使用示例



示例的时序图



5.5.2 CTR: 16 位循环计数指令

梯形图:										适用机型	APLC-TCP				
[CTR (D) (S)]										影响标志位					
指令列表: CTR (D) (S)										步长	5				
操作数	类型	适用软元件													变址
D	INT											C			
S	INT	常数	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	√

操作数说明

D: 目的操作数

S: 源操作数

功能说明

- 当输入能流有 OFF→ON 变化 (上升沿) 时, 指定的 16 位计数器 **C (D)** 计数值增 1。
- 当计数值等于计数预设值 (**S**) 时, 计数线圈置为 ON。
- 当计数值等于计数预设值 (**S**) 后, 如输入能流再有 OFF→ON 变化 (上升沿) 时, 计数值置为 1, 计数线圈清为 OFF。

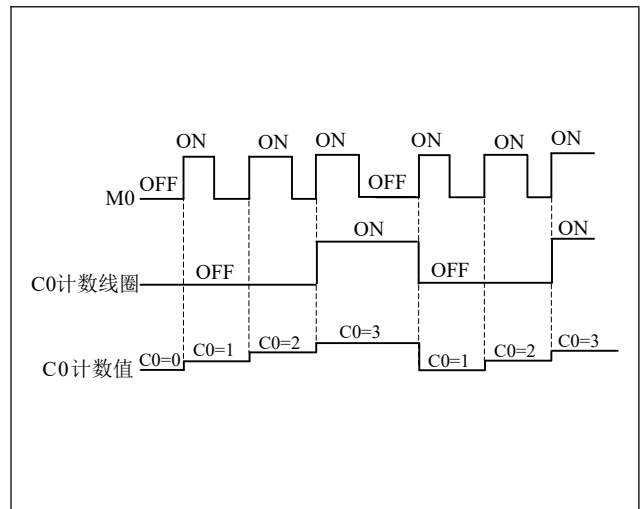
注意事项

- 当计数预设值 (**S**) 小于等于零时, 不产生计数动作。
- (**D**) 所指定的 16 位计数器 **C** 的地址应在 C0~C199 之内。

使用示例



示例的时序图



5.5.3 DCNT：32 位增减计数指令

梯形图：										适用机型	APLC-TCP				
										影响标志位					
指令列表：DCNT (D) (S)										步长	7				
操作数	类型	适用软元件													变址
D	DINT											C			
S	DINT	常数	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	√

操作数说明

D：目的操作数

S：源操作数

功能说明

- 当输入能流有 OFF→ON 变化（上升沿）时，指定的 32 位计数器 **C (D)** 计数值增 1 或减 1（计数增减方向由对应 **SM** 标志位决定）。
- 为增计数器时，当计数值大于等于计数预设值 (**S**) 时，计数线圈置为 ON。
- 为减计数器时，当计数值小于等于计数预设值 (**S**) 时，计数线圈置为 OFF。
- 当计数值 = 2147483647 时，如再次增一计数时，计数值变为 -2147483648。
- 当计数值 = -2147483648 时，如再次减一计数时，计数值变为 2147483647。

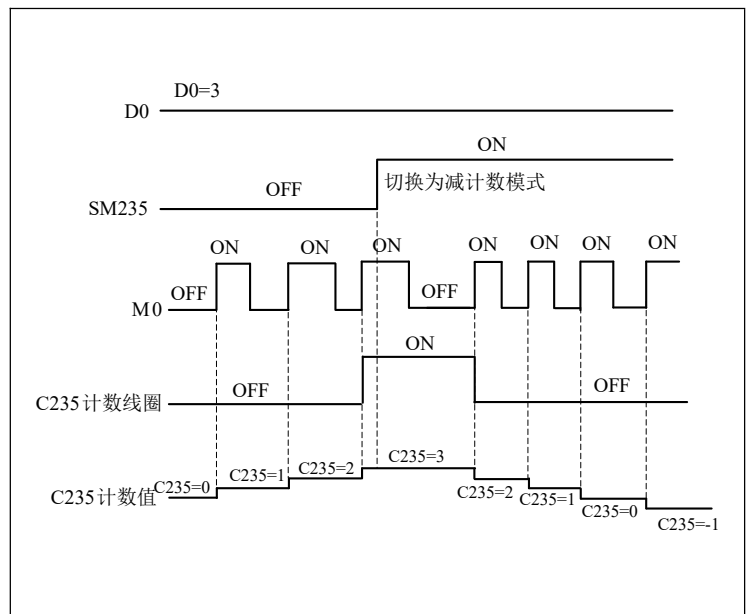
注意事项

D 指定的 **C** 元件的地址应在 C200~C235 之间。

使用示例



示例的时序图



第六章 应用指令说明

本章详细介绍了 APLC 系列小型 PLC 的应用指令，内容包括指令格式（形式）、操作数、影响标志位、功能、示例、时序图等。

6.1 程序流控制指令	100
6.1.1 FOR: 循环指令	100
6.1.2 NEXT: 循环返回	100
6.1.3 LBL: 跳转标号定义指令	102
6.1.4 CJ: 条件跳转指令	103
6.1.5 CFEND: 用户主程序条件返回	104
6.1.6 WDT: 用户程序看门狗清零	104
6.1.7 EI: 中断使能	104
6.1.8 DI: 中断禁止	105
6.1.9 CIRET: 用户中断程序条件返回	105
6.1.10 STOP: 用户程序停止	105
6.1.11 CALL: 用户子程序调用	105
6.1.12 CSRET: 用户子程序条件返回	106
6.2 数据传输指令	107
6.2.1 MOV: 字数据传输指令	107
6.2.2 DMOV: 双字数据传输指令	108
6.2.3 RMOV: 浮点数数据传输指令	109
6.2.4 BMOV: 块数据传输指令	109
6.2.5 FMOV: 数据块填充指令	110
6.2.6 DFMov: 数据块双字填充指令	111
6.2.7 SWAP: 高低字节交换指令	112
6.2.8 XCH: 字交换指令	112
6.2.9 DXCH: 双字交换指令	113
6.2.10 PUSH: 数据入栈指令	113
6.2.11 FIFO: 先入先出指令	114
6.2.12 LIFO: 后入先出指令	115
6.2.13	116
6.2.14 WSFR: 字串右移动指令	116
6.2.15 WSFL: 字串左移动指令	117
6.3 整数算术运算指令	119
6.3.1 ADD: 整数加法指令	119
6.3.2 SUB: 整数减法指令	119
6.3.3 MUL: 整数乘法指令	120
6.3.4 DIV: 整数除法指令	121
6.3.5 SQT: 整数算术平方根指令	121
6.3.6 INC: 整数加一指令	122

6.3.7 DAT: 整数减一指令	123
6.3.8 VABS: 整数绝对值指令	123
6.3.9 NEG: 整数取负指令	124
6.3.10 DADD: 长整数加法指令	125
6.3.11 DSUB: 长整数减法指令	125
6.3.12 DMUL: 长整数乘法指令	126
6.3.13 DDIV: 长整数除法指令	127
6.3.14 DSQT: 长整数算术开方指令	127
6.3.15 DINC: 长整数增一指令	128
6.3.16 DDAT: 长整数减一指令	129
6.3.17 DVABS: 长整数绝对值指令	129
6.3.18 DNEG: 长整数取负指令	130
6.3.19 SUM: 整数累加指令	131
6.3.20 DSUM: 长整数累加指令	131
6.4 浮点算术运算指令	132
6.4.1 RADD: 浮点数加法指令	133
6.4.2 RSUB: 浮点数减法指令	133
6.4.3 RMUL: 浮点数乘法指令	134
6.4.4 RDIV: 浮点数除法指令	135
6.4.5 RSQT: 浮点数算术开方指令	135
6.4.6 RVABS: 浮点数绝对值指令	136
6.4.7 RNEG: 浮点数取负指令	137
6.4.8 SIN: 浮点数 SIN 指令	137
6.4.9 COS: 浮点数 COS 指令	138
6.4.10 TAN: 浮点数 TAN 指令	139
6.4.11 POWER: 浮点数求幂运算	139
6.4.12 LN: 浮点数自然对数指令 LN	140
6.4.13 EXP: 浮点数自然数幂指令	141
6.4.14 RSUM: 浮点数累加指令	141
6.5 数值转换指令	143
6.5.1 DTI: 长整数转换整数指令	143
6.5.2 ITD: 整数转换长整数指令	143
6.5.3 FLT: 整数转换浮点数指令	144
6.5.4 DFLT: 长整数转换浮点数指令	145
6.5.5 INT: 浮点数转换整数指令	145
6.5.6 DINT: 浮点数转换长整数指令	146
6.5.7 BCD: 字转换 16 位 BCD 码指令	147
6.5.8 DBCD: 双字转换 32 位 BCD 码指令	147
6.5.9 BIN: 16 位 BCD 码转换字指令	148
6.5.10 DBIN: 32 位 BCD 码转换双字指令	149
6.5.11 GRY: 字转换 16 位格雷码指令	149
6.5.12 DGRY: 双字转换 32 位格雷码指令	150

6.5.13 GBIN: 16 位格雷码转换字指令	151
6.5.14 DGBIN: 32 位格雷码转换双字指令	151
6.5.15 SEG: 字转换 7 段码指令	152
6.5.16 ASC: ASCII 码转换指令	152
6.5.17 ITA: 16 位 16 进制数转换 ASCII 码指令	153
6.5.18 ATI: ASCII 码数转换 16 位 16 进制指令	154
6.6 字逻辑运算	155
6.6.1 WAND: 字与指令	155
6.6.2 WOR: 字或指令	155
6.6.3 WXOR: 字异或运算	156
6.6.4 WINV: 字取反运算	156
6.6.5 DWAND: 双字与指令	157
6.6.6 DWOR: 双字或指令	158
6.6.7 DWXOR: 双字异或指令	158
6.6.8 DWINV: 双字取反指令	159
6.7 位移动旋转指令	160
6.7.1 ROR: 16 位循环右移指令	160
6.7.2 ROL: 16 位循环左移指令	161
6.7.3 RCR: 16 位带进位循环右移指令	162
6.7.4 RCL: 16 位带进位循环左移指令	163
6.7.5 DROR: 32 位循环右移指令	163
6.7.6 DROL: 32 位循环左移指令	164
6.7.7 DRCR: 32 位带进位循环右移指令	165
6.7.8 DRCL: 32 位带进位循环左移指令	166
6.7.9 SHR: 16 位右移指令	166
6.7.10 SHL: 16 位左移指令	167
6.7.11 DSHR: 32 位右移指令	168
6.7.12 DSHL: 32 位左移指令	169
6.7.13 SFTR: 位串右移指令	169
6.7.14 SFTL: 位串左移指令	170
6.8 外设指令	172
6.8.1 FROM: 特殊模块缓冲寄存器字读指令	172
6.8.2 DFROM: 特殊模块缓冲寄存器双字读指令	172
6.8.3 TO: 特殊模块缓冲寄存器字写指令	172
6.8.4 DTO: 特殊模块缓冲寄存器双字写指令	172
6.8.5 VRRD: 读模拟电位器值指令	172
6.8.6 REFF: 设置输入滤波常数指令	173
6.8.7 REF: I/O 立即刷新指令	173
6.8.8 EROMWR: EEPROM 写指令	174
6.8.9 PR: 打印指令	175
6.9 实时时钟指令	176
6.9.1 TRD: 实时时钟读指令	176

6.9.2 TWR: 实时时钟写指令	177
6.9.3 TADD: 时钟加指令	178
6.9.4 TSUB: 时钟减指令	179
6.9.5 HOUR: 计时表指令	181
6.9.6 DCMP: (=、<、>、◇、>=、<=) 日期比较指令	182
6.9.7 TCMP: (=、<、>、◇、>=、<=) 时间比较指令	183
6.10 高速 IO 指令	185
6.10.1 HCNT: 高速计数器驱动指令	185
6.10.2 DHSCS: 高速计数比较置位指令	186
6.10.3 DHSCI: 高速计数比较中断触发指令	188
6.10.4 DHSCR: 高速计数比较复位指令	189
6.10.5 DHSZ: 高速计数区间比较指令	190
6.10.6 DHST: 高速计数表格比较指令	192
6.10.7 DHSP: 高速计数表格比较脉冲输出指令	193
6.10.8 SPD: 测频指令	196
6.10.9 PLSY: 计数脉冲输出指令	197
6.10.10 PLSR: 带加减速的计数脉冲输出指令	199
6.10.11 PLS: 包络线脉冲输出指令	201
6.10.12 PLSB: 带基底频率与加减速的计数脉冲输出指令	203
6.10.13 PWM: 脉冲输出指令	205
6.11 控制计算指令	207
6.11.1 PID: 功能指令	207
6.11.2 RAMP: 斜坡信号输出指令	211
6.11.3 HACKLE: 锯齿波信号输出指令	213
6.11.4 TRIANGLE: 三角波信号输出指令	215
6.11.5 ABSD: 凸轮绝对控制指令	217
6.11.6 DABSD: 凸轮绝对控制指令	217
6.11.7 ALT: 交替输出指令	217
6.12 通讯指令	218
6.12.1 Modbus: 主站通讯指令	218
6.12.2 EVFWD: 变频器正转指令	220
6.12.3 EVREV: 变频器反转指令	221
6.12.4 EVDFWD: 变频器点动正转指令	221
6.12.5 EVDREV: 变频器点动反转指令	222
6.12.6 EVSTOP: 变频器停止指令	223
6.12.7 EVFRQ: 设置变频器频率指令	223
6.12.8 EVWRT: 写单个寄存器值指令	224
6.12.9 EVRDST: 读取变频器状态指令	225
6.12.10 EVRD: 读取变频器单个寄存器值指令	226
6.12.11 XMT: 自由口发送指令	227
6.12.12 RCV: 自由口接收指令	228
6.12.13 MODRW: MODBUS 读写指令	229

6.13 校验指令	230
6.13.1 CCITT: 校验指令	230
6.13.2 CRC16: 校验指令	231
6.13.3 LRC: 校验指令	232
6.14 增强型位处理指令	233
6.14.1 ZRST: 批量位清零指令	233
6.14.2 ZSET: 批量位置位指令	233
6.14.3 DECO: 解码指令	234
6.14.4 ENCO: 编码指令	234
6.14.5 BITS: 字中 ON 位统计指令	235
6.14.6 DBITS: 双字中 ON 位统计指令	236
6.15 字触点指令	236
6.15.1 BLD: 字位触点 LD 指令	236
6.15.2 BLDI: 字位触点 LDI 指令	237
6.15.3 BAND: 字位触点 AND 指令	237
6.15.4 BANI: 字位触点 ANI 指令	238
6.15.5 BOR: 字位触点 OR 指令	238
6.15.6 BORI: 字位触点 ORI 指令	239
6.15.7 BOUT: 字位线圈输出指令	240
6.15.8 BSET: 字位线圈置位指令	240
6.15.9 BRST: 字位线圈清除指令	241
6.16 比较触点指令	241
6.16.1 LD (=, <, >, <>, >=, <=): 整数比较 LD※指令	241
6.16.2 AND (=, <, >, <>, >=, <=): 整数比较 AND※指令	242
6.16.3 OR (=, <, >, <>, >=, <=): 整数比较 OR※指令	243
6.16.4 LDD (=, <, >, <>, >=, <=): 长整数比较 LDD※指令	244
6.16.5 ANDD (=, <, >, <>, >=, <=): 长整数比较 ANDD※指令	245
6.16.6 ORD (=, <, >, <>, >=, <=): 长整数比较 ORD※指令	246
6.16.7 LDR: 浮点数比较指令	247
6.16.8 ANDR: 浮点数比较指令	248
6.16.9 ORR: 浮点数比较指令	249
6.16.10 CMP: 整数比较置位指令	250
6.16.11 LCMP: 长整数比较置位指令	250
6.16.12 RCMP: 浮点数比较置位指令	250

6.1 程序流控制指令

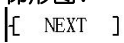
6.1.1 FOR：循环指令

梯形图： 		适用机型	APLC-TCP												
		影响标志位													
指令列表：FOR (S)		步长	3												
操作数	类型	适用软元件													变址
S	INT	常数	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	√

操作数说明

S：源操作数

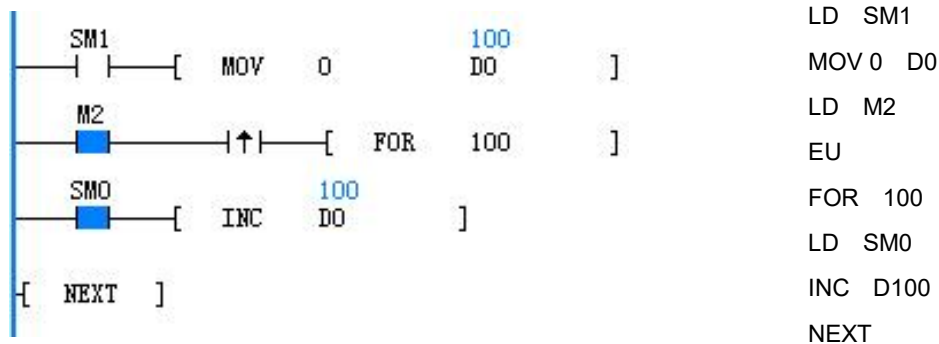
6.1.2 NEXT：循环返回

梯形图： 		适用机型	APLC-TCP												
		影响标志位													
指令列表：NEXT		步长	1												

功能说明

- FOR 指令与 NEXT 匹配成一个 FOR-NEXT 结构。
- 当 FOR 前的能流有效，且循环次数（S）大于零时，FOR-NEXT 结构中间的指令被连续循环执行 S 次。当循环执行完 S 次后，继续执行 FOR-NEXT 结构后的指令。
- 如果 FOR 前的能流无效，或循环次数（S）小于等于零时，FOR-NEXT 结构中间的指令不被执行，程序直接跳转到该 FOR-NEXT 结构后继续执行。

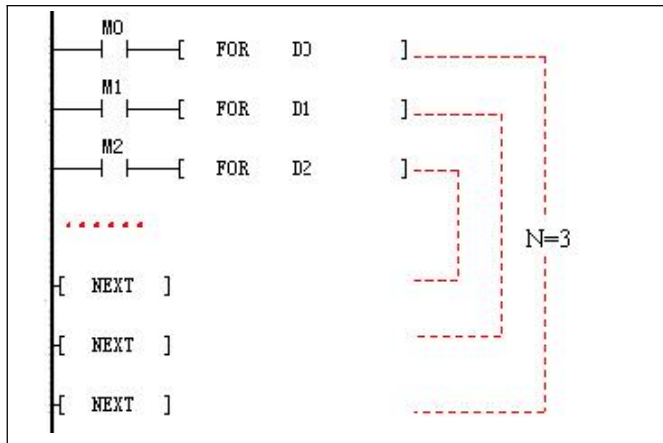
使用示例



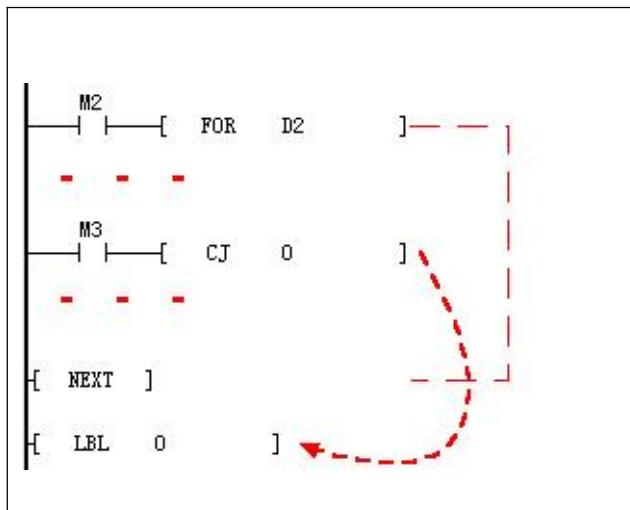
运行的初始条件 D0=0，M2=OFF。当 M2 有 OFF→ON 变化时，FOR-NEXT 结构内指令被连续执行 100 次，D0 被 100 次增一操作，循环结束后，D0=100。

注意事项

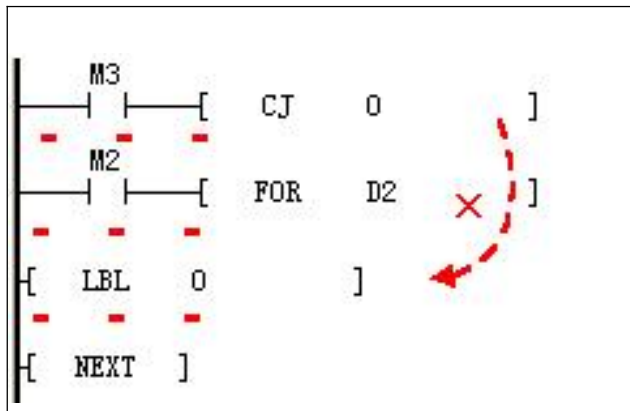
1. FOR-NEXT 指令在一个程序体（POU）中必须成对使用，否则用户程序不能正确编译通过。
2. 支持多个 FOR-NEXT 结构嵌套。（下图示例了一个 3 层 FOR-NEXT 结构嵌套）



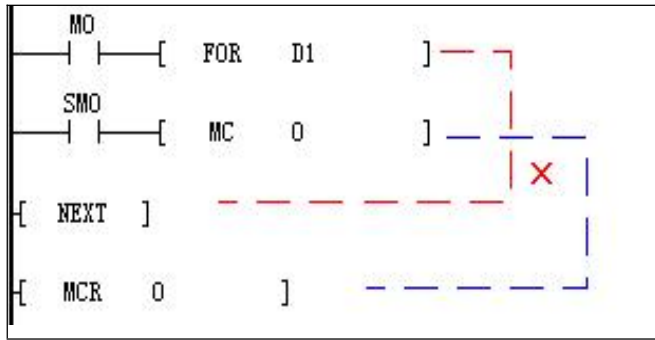
3. 可以在循环体内使用条件跳转指令（CJ）跳出循环体，从而达到提前终止循环体执行的目的，如下梯形图所示：



4. 禁止用户使用跳转语句（CJ）跳入一个循环体，如下梯形图将不能正确编译通过：



5. 禁止 MC-MCR 结构体和 FOR-NEXT 结构体的交叉，如下梯形图将不能通过正确编译：



注意

FOR-NEXT 循环体执行较为耗时，循环次数越多，或循环体内所包含的指令越多，执行耗时也就越长。为防止运行超时错误发生，请注意在耗时的循环体内使用 WDT 指令。

6.1.3 LBL：跳转标号定义指令

梯形图： [LBL (S)]		适用机型	APLC-TCP											
		影响标志位												
指令列表：LBL (S)		步长	3											
操作数	类型	适用软元件												变址
S	INT	常数												

操作数说明

S：标号值

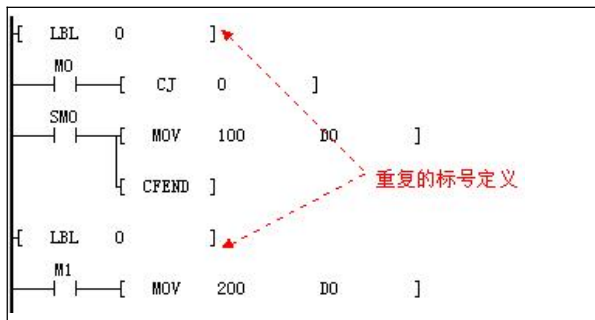
功能说明

- 定义了一个标号值为 **S** 的标号。
- 不产生实质性操作，只是为条件跳转指令（CJ）标明了跳转的具体位置。

注意事项

- 标号值 **S** 的范围： $0 \leq S \leq 127$ 。
- 在一个用户程序中，不允许在同一个程序体中出现两个重复定义的标号，否则用户程序将不能通过编译。但允许不同程序体（如不同的子程序）中出现重复标号定义。

错误的程序示例



6.1.4 CJ：条件跳转指令

梯形图：				适用机型		APLC-TCP							
				影响标志位									
指令列表：CJ (S)				步长		3							
操作数	类型	适用软元件										变址	
S	INT	常数											

操作数说明

S：标号值

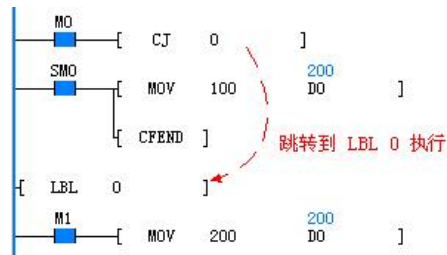
功能说明

1. 当能流有效时，用户程序跳转到编号为 **S** 的合法标号指令处执行。
2. 如能流无效时，不发生跳转操作，顺序执行 CJ 后一条指令。

注意事项

1. CJ 指令所要跳转的标号 **S** ($0 \leq S \leq 127$) 应是一个合法的、已定义的标号，否则用户程序将不能正确通过编译。
2. 不允许使用 CJ 指令跳转到一个 FOR-NEXT 结构中。
3. 可以使用 CJ 指令跳出或跳入 MC-MCR 结构和 SFC 状态，但这样将破坏 MC-MCR 和 SFC 状态的逻辑，使程序复杂化，建议不要这样使用。

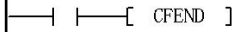
使用示例



```
LD M0
CJ 0
LD SMO
MOV 100 D0
CFEND
LBL 0
LD M1
MOV 200 D0
```

1. 初始条件 M0=OFF，M1=ON，CJ 0 不跳转，D0=100。执行 CFEND 后，程序流提前退出主程序，指令 LD M1 和 MOV 200 D0 不执行。
2. 当 M0=ON，M1=ON 时，指令 CJ 0 被执行，MOV 100 D0 和 CFEND 指令被跨越。跳转至 LBL 0 后，执行 MOV 200 D0 指令，此时 D0=200。

6.1.5 CFEND：用户主程序条件返回

梯形图： 	适用机型	APLC-TCP
	影响标志位	
指令列表：CFEND	步长	1

功能说明

- 当指令的能流有效时，主程序从当前扫描周期（用户程序的主程序是按扫描周期，被系统周而复始地调用执行）中返回系统，其后的主程序中的指令不被执行。
- 当指令的能流无效时，该指令不产生任何动作，其后的指令被顺序执行。

注意事项

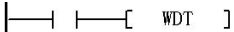
CFEND 指令必须在用户主程序中出现，否则不能通过编译。

使用示例



程序运行时，当 M0=OFF 时，CFEND 指令不产生动作，其后的 LD SM12，OUT Y0 被执行，可观测到 Y0 周期闪烁输出。当 M0=ON 时，CFEND 指令产生动作，程序流提前从主程序返回系统，其后的 LD SM12，OUT Y0 不被执行，Y0 周期闪烁现象消失。

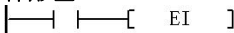
6.1.6 WDT：用户程序看门狗清零

梯形图： 	适用机型	AT100
	影响标志位	
指令列表：WDT	步长	1

功能说明

当能流有效时，该指令会将用户程序看门狗的计时值归零，系统用户程序看门狗重新开始计时。

6.1.7 EI：中断使能

梯形图： 	适用机型	APLC-TCP
	影响标志位	
指令列表：EI	步长	1

功能说明

- 当能流有效时，中断使能。
- 当 EI 指令有效时，中断请求将被允许加入到中断请求队列中，等待系统响应。

6.1.8 DI：中断禁止

梯形图： 	适用机型	APLC-TCP
	影响标志位	
指令列表：DI	步长	1

功能说明

1. 当能流有效时，中断全局使能标志失效，即关全局中断。
2. 当全局中断使能标志失效时，各类中断事件不能产生中断请求。

注意事项

关中断请求指令生效时，如中断请求队列中仍有中断请求未处理完时，剩余的中断请求仍然要被响应，但新中断事件将不能产生中断请求。

6.1.9 CIRET：用户中断程序条件返回

梯形图： 	适用机型	APLC-TCP
	影响标志位	
指令列表：CIRET	步长	1

功能说明

当能流有效时，提前退出正在执行的中断程序。

6.1.10 STOP：用户程序停止

梯形图： 	适用机型	APLC-TCP
	影响标志位	
指令列表：STOP	步长	1

功能说明

当能流有效时，系统将立即停止用户程序的执行。

6.1.11 CALL：用户子程序调用

梯形图： 	适用机型	APLC-TCP
	影响标志位	
指令列表：CALL (子程序名) (子程序参数1) (子程序参数2) ...	步长	由子程序所带参数决定

功能说明

当能流有效时，调用指定名称的子程序执行，子程序执行完成后，返回至 CALL 之后的指令继续执行。

注意事项

1. CALL 指令中所调用的子程序，必须要在用户程序中事先定义好。当 CALL 中出现一个从未定义的子程序，将不能通过程序编译。

2. CALL 指令中所带操作数的元件类型，应与子程序的局部变量表中定义的数据类型相匹配，否则不能通过编译。

以下示例说明非法的匹配使用：

例一：SBR1 子程序的局部变量表中，操作数一的数据类型为 DINT/DWORD 型。

以下使用都是非法的：

- CALL SBR1 Z0 (Z 元件不能用作数据类型为 DINT/DWORD 型)
- CALL SBR1 C199 (C0~C199 元件不能用作数据类型为 DINT/DWORD 型)
- CALL SBR1 K2X0 (Kn 寻址 $1 \leq n \leq 3$ ，不能用作数据类型为 DINT/DWORD 型)

例二：SBR1 子程序的局部变量表中，操作数一的数据类型为 INT/WORD 型。

以下使用都是非法的：

- CALL SBR1 C200 (C200~C255 元件不能用作数据类型为 INT/WORD 型)
- CALL SBR1 K2X0 (Kn 寻址 $4 \leq n \leq 8$ ，不能用作数据类型为 INT/WORD 型)

3. CALL 指令中所带的操作数的元件类型，应与子程序局部变量表中定义的变量类型相匹配，否则不能通过编译。

以下示例说明非法的匹配使用：

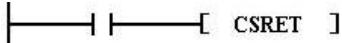
例：SBR1 子程序的局部变量表中，操作数一的操作数类型为 OUT 或 IN_OUT 型。

以下使用都是非法的：

- CALL SBR1 321 (常数不可能改变，所以与 OUT 或 IN_OUT 型操作数不匹配)
- CALL SBR1 K4X0 (K4X0 只具有只读性质，所以与 OUT 或 IN_OUT 型操作数不匹配)
- CALL SBR1 SD0 (SD0 只具有只读性质，所以与 OUT 或 IN_OUT 型操作数不匹配)

4. CALL 指令中所带的操作数个数，应与子程序局部变量表相配，否则不能通过编译。

6.1.12 CSRET：用户子程序条件返回

梯形图： 	适用机型	APLC-TCP
	影响标志位	
指令列表：CSRET	步长	1

功能说明

当能流有效时，退出当前执行的子程序，返回上一级子程序。

6.2 数据传输指令

6.2.1 MOV：字数据传输指令

梯形图： 										适用机型	APLC-TCP				
指令列表：MOV (S) (D)										影响标志位					
										步长	5				
操作数	类型	适用软元件													变址
S	INT	常数	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	√
D	INT			KnY	KnM	KnS	KnLM		D	SD	C	T	V	Z	√

操作数说明

S：源操作数

D：目的操作数

功能说明

当能流有效时，**S** 的内容赋给 **D**，**S** 的值不变。

注意事项

- MOV 指令支持有符号和无符号两种整数。如果指令两个操作数都是软元件，则数据类型都是有符号整数。如果指令的源操作数是有符号长整数，如（-10，+100），则目的操作数也是有符号整数。如果源操作数是无符号的长整数，如（100，45535），则目的操作数也是无符号整数。
- 对应的软元件 C 只支持 C0~C199。

使用示例



当 X0=ON 时，D0 的内容赋给 D10，D10=500。

6.2.2 DMOV：双字数据传输指令

梯形图： 		适用机型	APLC-TCP												
		影响标志位													
指令列表：DMOV (S) (D)		步长	7												
操作数	类型	适用软元件													变址
S	DINT	常数	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	√
D	DINT			KnY	KnM	KnS	KnLM		D	SD	C		V		√

操作数说明

S：源操作数

D：目的操作数

功能说明

当能流有效时，**S** 的内容赋给 **D**，**S** 的值不变。

注意事项

- DMOV 指令支持有符号和无符号两种长整数。如果指令两个操作数都是软元件，则数据类型都是有符号整数。如果指令的源操作数是有符号长整数，如（-10，+100），则目的操作数也是有符号整数。如果源操作数是无符号的长整数，如（100，45535），则目的操作数也是无符号整数。
- 对应的软元件 C 只支持 C200~C255。

使用示例



当 X0=ON 时，（D0, D1）的内容赋给（D10, D11），（D10, D11）=50000。

6.2.3 RMOV: 浮点数数据传输指令

梯形图:									适用机型	APLC-TCP					
指令列表: RMOV (S) (D)									影响标志位						
									步长	7					
操作数	类型	适用软元件													变址
S	REAL	常数							D				V		√
D	REAL								D				V		√

操作数说明

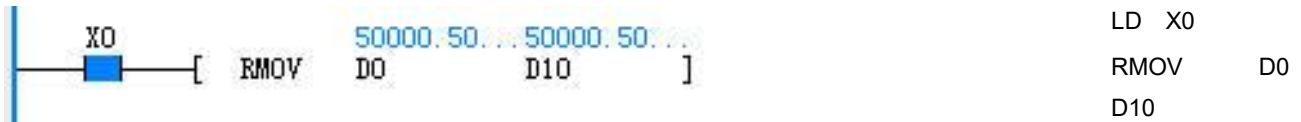
S: 源操作数

D: 目的操作数

功能说明

当能流有效时, **S** 的内容赋给 **D**, **S** 的值不变。

使用示例



当 X0=ON 时, (D0,D1) 的内容赋给 (D10,D11), (D10,D11) =50000.5。

6.2.4 BMOV: 块数据传输指令

梯形图:									适用机型	APLC-TCP					
指令列表: BMOV (S1) (D) (S2)									影响标志位						
									步长	7					
操作数	类型	适用软元件													变址
S1	INT		KnX	KnY	KnM	KnS	KnLM		D	SD	C	T	V		√
D	INT			KnY	KnM	KnS	KnLM		D		C	T	V		√
S2	INT	常数	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	√

操作数说明

S: 源操作数, 数据块起始单元

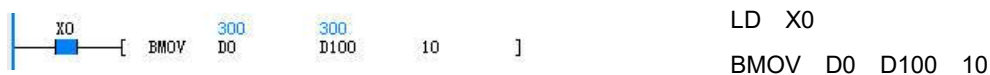
D: 目的操作数, 数据块起始单元

S2: 数据块大小

功能说明

当能流有效时, **S1** 单元开始的 **S2** 个单元的内容赋给 **D** 单元开始的 **S2** 个单元, **S1** 单元开始的 **S2** 个单元的内容不变。

使用示例



当 X0=ON 时, D0 开始的 10 个单元的内容赋给 D100 开始的 10 个单元。D100=D0, D101=D1,, D109=D9。

6.2.5 FMOV: 数据块填充指令

梯形图： 									适用机型		APLC-TCP					
									影响标志位							
指令列表： FMOV (S1) (D) (S2)									步长		7					
操作数		类型	适用软元件												变址	
S1		INT	常数	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	√
D		INT			KnY	KnM	KnS	KnLM		D		C	T	V		√
S2		INT	常数	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	√

操作数说明

S1: 源操作数, 数据块起始单元

D: 目的操作数, 数据块起始单元

S2: 数据块大小

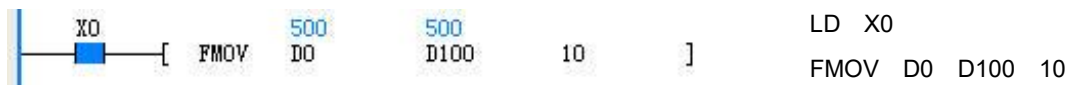
功能说明

当能流有效时, **S1** 单元的内容填充到 **D** 单元开始的 **S2** 个单元中, **S1** 单元的内容不变。

注意事项

1. **S1**、**D**、**S2** 使用 C 元件时, 合法范围为 C0~C199。
2. **S2** 大于等于 0。
3. **S1**、**D** 同时为 Kn 寻址时, Kn 应相等。

使用示例



当 X0=ON 时, D0 的内容填充到 D100 开始的 10 个单元。D100=D101=.....=D109=D0=500。

6.2.6 DFMOV：数据块双字填充指令

梯形图： 									适用机型	APLC-TCP					
指令列表：DFMOV (S1) (D) (S2)									影响标志位						
									步长	9					
操作数	类型	适用软元件													变址
S1	DINT	常数	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C		V		√
D	DINT			KnY	KnM	KnS	KnLM		D		C		V		√
S2	INT	常数	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	√

操作数说明

S1：源操作数起始

D：目的操作数，数据块起始单元

S2：数据块大小

功能说明

当能流有效时，**S1**单元的内容填充到**D**单元开始的**S2**个单元中，**S1**单元的内容不变。

注意事项

1. **S1**、**D**、**S2**使用C元件时，合法范围为C200~C255。
2. **S2**大于等于0。
3. **S1**、**D**同时为Kn寻址时，Kn应相等。

使用示例



当X0=ON时，(D0,D1)的内容填充到D10开始的10×2个单元。(D10,D11)=(D12,D13)=.....=(D28,D29)=(D0,D1)=100000。

6.2.7 SWAP：高低字节交换指令

梯形图：										适用机型	APLC-TCP				
[SWAP (D)]										影响标志位					
指令列表：SWAP (D)										步长	3				
操作数	类型	适用软元件													变址
D	INT			KnY	KnM	KnS	KnLM		D		C	T	V	Z	√

操作数说明

D：目的操作数，指被交换高低字节的字元件

功能说明

当能流有效时，**D** 的内容高低字节交换后的值保存到 **D** 单元。

使用示例



```
LD X0
SWAP D0
```

当 X0=ON 时，D0=0x1027 (4135) 的内高低字节交换后的值保存到 D0，D0=0x2710 (10000)。

6.2.8 XCH：字交换指令

梯形图：										适用机型	APLC-TCP				
[XCH (D1) (D2)]										影响标志位					
指令列表：XCH (D1) (D2)										步长	5				
操作数	类型	适用软元件													变址
D1	INT			KnY	KnM	KnS	KnLM		D		C	T	V	Z	√
D2	INT			KnY	KnM	KnS	KnLM		D		C	T	V	Z	√

操作数说明

D1：目的操作数 1

D2：目的操作数 2

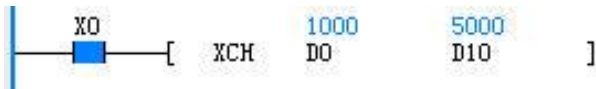
功能说明

当能流有效时，**D1** 的内容与 **D2** 的内容互换后的值保存到 **D1**、**D2** 单元当中。

注意事项

在使用 Kn 寻址方式时，**D1** 和 **D2** 中的 Kn 应相同。

使用示例



```
LD X0
XCH D0 D10
```

当 X0=ON 时，D0 与 D10 的内容互换。执行前：D0=5000，D10=1000。

执行后：D0=1000，D10=5000。

6.2.9 DXCH: 双字交换指令

梯形图: 									适用机型	APLC-TCP					
指令列表: DXCH (D1) (D2)									影响标志位						
									步长	7					
操作数	类型	适用软元件													变址
D1	DINT			KnY	KnM	KnS	KnLM		D		C	T	V	Z	√
D2	DINT			KnY	KnM	KnS	KnLM		D		C	T	V	Z	√

操作数说明

D1: 目的操作数 1;

D2: 目的操作数 2

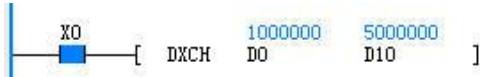
功能说明

当能流有效时, **D1** 的内容与 **D2** 的内容互换后的值保存到 **D1**、**D2** 单元当中。

注意事项

在使用 Kn 寻址方式时, **D1**、**D2** 中的 Kn 应相同。

使用示例



LD X0
DXCH D0 D10

当 X0=ON 时, (D0,D1) 与 (D10,D11) 的内容互换。执行前: (D0,D1) =5000000, (D10,D11) =1000000。执行后: (D0,D1) =1000000, (D10,D11) =5000000。

6.2.10 PUSH: 数据入栈指令

梯形图: 									适用机型	APLC-TCP					
指令列表: PUSH (S1) (D) (S2)									影响标志位						
									步长	7					
操作数	类型	适用软元件													变址
S1	INT	常数	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	√
D	INT								D				V		√
S2	INT	常数	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	√

操作数说明

S1: 入栈值

D: 存储栈中元素个数, 同时其元件标号为栈底位置

S2: 栈的大小

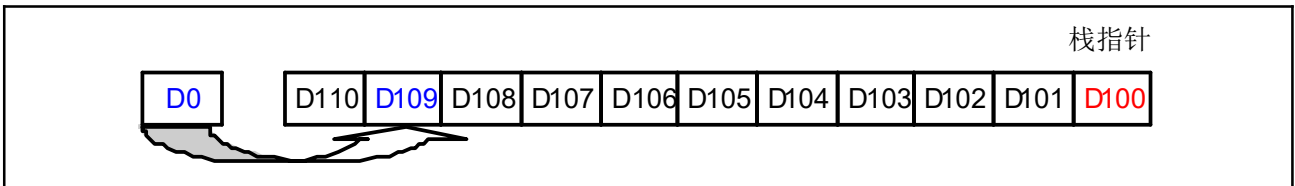
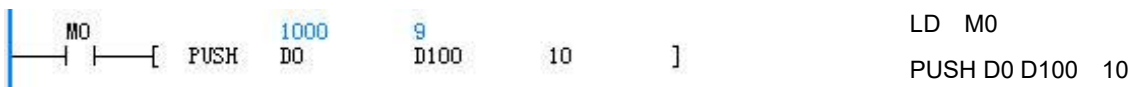
功能说明

1. 当能流有效时，将 **S1** 的值压入以 **D** 单元为栈底的栈顶中，同时 **D** 的值增加 1。此时栈顶单元的编号为：**D** 的编号 + **D** 值。
2. 当 **D** 值等于 **S2** 值时，仍有入栈指令执行，运算进位标志位 (**SM181**) 置 1，不执行压栈操作。

注意事项

1. 当所操作栈定义非法时，（当栈尺寸小于等于零，栈中元素个数小于零。栈中元素个数大于栈尺寸限制）报操作栈定义非法错误。
2. 栈的大小不包括包括栈底元素 (**D** 指定的元件)。
3. **S2** 指明栈的大小，范围大于等于 0。

使用示例



1. 当 **M0=ON** 时，将 **D0** 内容压入 **D100** 为栈底的栈中。
2. 执行前：**D0=1000**，**D100=8**，**D109=0**。
3. 执行后：**D0=1000**，**D100=9**，**D109=1000**。

6.2.11 FIFO：先入先出指令

梯形图：									适用机型	APLC-TCP					
[FIFO (D1) (D2) (S)]									影响标志位						
指令列表：FIFO (D1) (D2) (S)									步长	7					
操作数	类型	适用软元件												变址	
D1	INT								D				V		√
D2	INT			KnY	KnM	KnS	KnLM		D		C	T	V	Z	√
S	INT	常数	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	√

操作数说明

D1：栈中元素个数，同时其元件编号 + 1 元件为栈首元件

D2：出栈值存储单元

S：队列大小

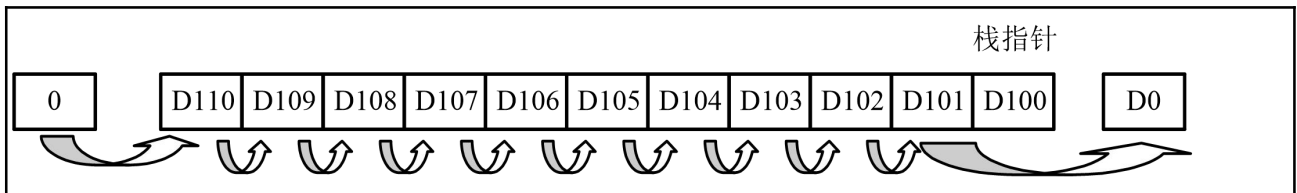
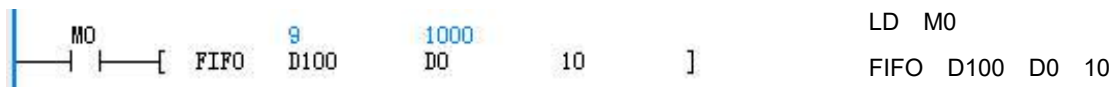
功能说明

1. 当能流有效时，将以 **D1** 为队首的字栈首值 (**D1** 后一单元的内容) 赋给 **D2** 单元中，同时 **D1** 的值减 1，**D1** 后的 **S** 个单元内容由后向前移动，最后的单元填 0。
2. 当 **D1** 值等于 0 时表明栈空出，零标志 (SM180) 位置 1。

注意事项

1. 当所操作栈定义非法时，（当栈尺寸小于等于零，栈中元素个数小于零。栈中元素个数大于栈尺寸限制）报操作栈定义非法错误
2. 栈的大小不包括包括栈底元素 (**D1** 指定的元件)
3. **S** 指明栈的大小，范围大于等于 0。

使用示例



1. 当 **M0=ON** 时，将 **D101** 内容填入 **D0** 中，同时 **D101~D110** 单元内容由后向前移动，**D110** 内容填 0。
2. 执行前：**D0=0**，**D100=10**，**D101=1000**，**D102=2000**，.....，**D109=9000**，**D110=10000**。
3. 执行后：**D0=1000**，**D100=9**，**D101=2000**，**D102=3000**，.....，**D109=10000**，**D110=0**。

6.2.12 LIFO：后入先出指令

<div>梯形图：</div> <div><div><div></div><div></div></div><div><div></div><div></div></div><div>[</div><div>LIFO</div><div>(D1)</div><div>(D2)</div><div>(S)</div><div>]</div></div>									适用机型		APLC-TCP					
									影响标志位							
指令列表：LIFO (D1) (D2) (S)									步长		7					
操作数		类型	适用软元件												变址	
D1		INT							D				V		√	
D2		INT			KnY	KnM	KnS	KnLM		D		C	T	V	Z	√
S		INT	常数	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	√

操作数说明

D1：队列中元素个数，同时其元件编号 + 1 元件为队首元素

D2：出栈值存储单元

S：队列大小

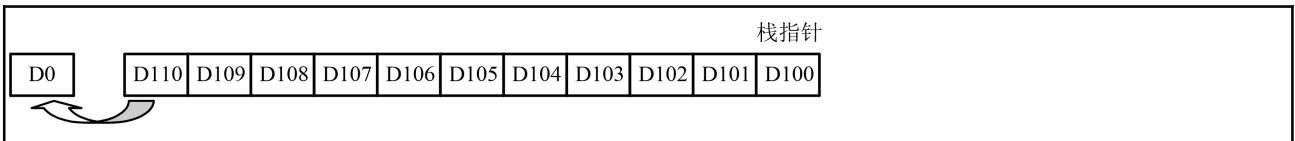
功能说明

1. 当能流有效时，将以 **D1** 为栈底的栈顶单元的内容赋给 **D2** 单元，同时 **D1** 的值减 1。
2. 当 **D1** 值等于 0 时表明栈空出，零标志（SM180）位置 1。

注意事项

1. 当所操作栈定义非法时，（当栈尺寸小于等于零，栈中元素个数小于零。栈中元素个数大于栈尺寸限制）报操作栈定义非法错误。
2. 栈的大小不包括包括栈底元素（**D1** 指定的元件）。
3. **S** 指明栈的大小，范围大于等于 0。

使用示例



1. 当 M0=ON 时，将 D110 的内容赋给 D0，D101~D110 单元的内容不变。
2. 执行前：D0=0，D100=10，D101=1000，D102=2000，……，D109=9000，D110=10000。
3. 执行后：D0=10000，D100=9，D101=1000，D102=2000，……，D109=9000，D110=10000。

6.2.13

6.2.14 WSFR：字串右移动指令

<div>梯形图：</div> <div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div>[WSFR (S1) (D) (S2) (S3)]</div></div>									适用机型		APLC-TCP					
									影响标志位		零标志 进位标志 借位标志					
指令列表：WSFR (S1) (D) (S2) (S3)									步长		9					
操作数		类型	适用软元件												变址	
S1		INT		KnX	KnY	KnM	KnS	KnLM		D	SD	C	T	V		√
D		INT			KnY	KnM	KnS	KnLM		D		C	T	V		√
S2		INT	常数	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	√
S3		INT	常数	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	√

操作数说明

S1：源操作数

D：目的操作数，字串起始元件

S2：目的字队列的大小

S3：右移填入字个数

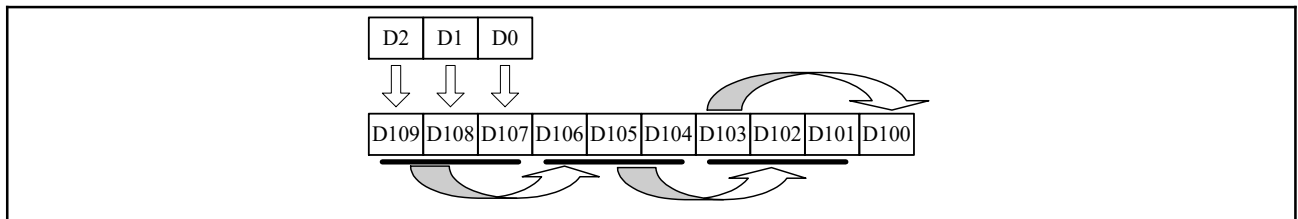
功能说明

当能流有效时，将 **D** 单元开始的 **S2** 个单元的内容以字为单位右移 **S3** 个单元，最右端 **S3** 个数据将被丢弃，同时，以 **S1** 单元开始的 **S3** 个单元的内容将被移入字串的左端。

注意事项

1. 左右顺序，以小元件编号元件为右，大元件编号为左。
2. **S2** 大于等于 0，**S3** 大于等于 0。
3. **S2** 大于等于 **S3**。
4. 当 **S1**、**D** 同时为 Kn 寻址时，Kn 应相等

使用示例



1. 当 M0=ON 时，将 D100 单元开始的 10 个单元的内容，以字为单位右移 3 个单元，最右端 D102~D100 单元的数据被丢弃。同时，D0 单元开始的 3 个单元的内容被移入字串的左端。
2. 执行前：D2=300，D1=200，D0=100。D109=10000，D108=9000，D107=8000，D106=7000，D105=6000，D104=5000，D103=4000，D102=3000，D101=2000，D100=1000。
3. 执行后：D0~D2 内容不变。D2=300，D1=200，D0=100。D109=300，D108=200，D107=100，D106=10000，D105=9000，D104=8000，D103=7000，D102=6000，D101=5000，D100=4000。

6.2.15 WSFL：字串左移动指令

<div>梯形图：</div> <div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div>[WSFL (S1) (D) (S2) (S3)]</div></div>									适用机型		APLC-TCP				
									影响标志位		零标志 进位标志 借位标志				
指令列表： WSFL (S1) (D) (S2) (S3)									步长		9				
操作数	类型	适用软元件													变址
S1	INT		KnX	KnY	KnM	KnS	KnLM		D	SD	C	T	V		√
D	INT			KnY	KnM	KnS	KnLM		D		C	T	V		√
S2	INT	常数	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	√
S3	INT	常数	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	√

操作数说明

S1: 源操作数

D: 目的操作数, 字串起始元件

S2: 目的字队列的大小

S3: 右移填入字数

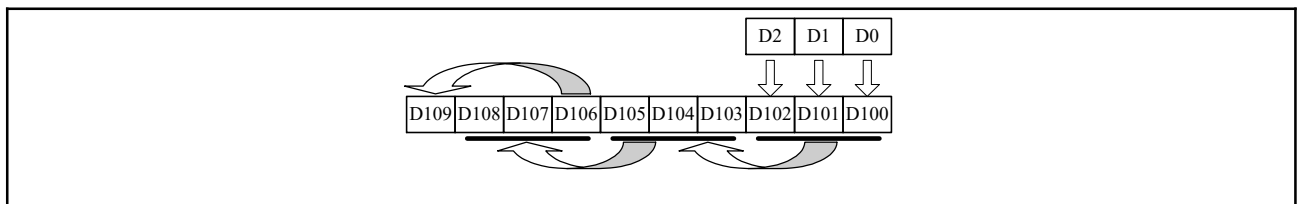
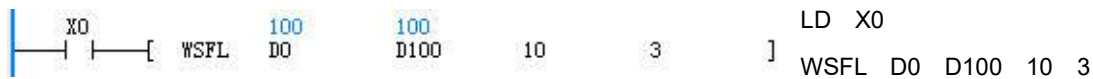
功能说明

当能流有效时, 将 **D** 单元开始的 **S2** 个单元的内容以字为单位左移 **S3** 个单元, 最左端 **S3** 个数据将被丢弃, 同时, 以 **S1** 单元开始的 **S3** 个单元的内容将被移入字串的右端。

注意事项

1. 左右顺序, 以小元件编号元件为右, 大元件编号为左。
2. **S2** 大于等于 0; **S3** 大于等于 0。
3. **S2** 大于等于 **S3**。
4. 当 **S1**、**D** 同时为 Kn 寻址时, Kn 应相等。

使用示例



1. 当 **X0=ON** 时, 将 **D100** 单元开始的 10 个单元的内容以字为单位左移 3 个单元, 最左端 **D109~D107** 单元的数据将被丢弃, 同时, **D0** 单元开始的 3 个单元的内容被移入字串的右端。
2. 执行前: **D0=100, D1=200, D2=300. D109=10000, D108=9000, D107=8000, D106=7000, D105=6000, D104=5000, D103=4000, D102=3000, D101=2000, D100=1000.**
3. 执行后: **D0~D2** 内容不变。 **D2=300, D1=200, D0=100. D109=7000, D108=6000, D107=5000, D106=4000, D105=3000, D104=2000, D103=1000, D102=300, D101=200, D100=100.**

6.3 整数算术运算指令

6.3.1 ADD：整数加法指令

梯形图： 										适用机型	APLC-TCP				
										影响标志位	零标志 进位标志 借位标志				
指令列表：ADD (S1) (S2) (D)										步长	7				
操作数	类型	适用软元件													变址
S1	INT	常数	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	√
S2	INT	常数	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	√
D	INT			KnY	KnM	KnS	KnLM		D		C	T	V	Z	√

操作数说明

S1：源操作数 1

S2：源操作数 2

D：目的操作数

功能说明

- 当能流有效时，**S1** 加 **S2**，运算结果赋予 **D**。
- 运算结果 (**D**) 大于 32767 时，置进位标志位 (SM181)。运算结果等于 0 时，置零标志位 (SM180)。运算结果小于 -32768 时，置借位标志位 (SM182)。

使用示例



当 X0=ON 时，D0 (1000) 加上 D1 (2000) 结果赋给 D10，D10=3000。

6.3.2 SUB：整数减法指令

梯形图： 										适用机型	APLC-TCP				
										影响标志位	零标志 进位标志 借位标志				
指令列表：SUB (S1) (S2) (D)										步长	7				
操作数	类型	适用软元件													变址
S1	INT	常数	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	√
S2	INT	常数	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	√
D	INT			KnY	KnM	KnS	KnLM		D		C	T	V	Z	√

操作数说明

S1: 源操作数 1

S2: 源操作数 2

D: 目的操作数

功能说明

1. 当能流有效时, **S1** 减 **S2**, 运算结果赋予 **D**。
2. 当运算结果 (**D**) 大于 32767 时, 置进位标志位 (SM181)。运算结果等于 0 时, 置零标志位 (SM180)。运算结果小于 -32768 时, 置借位标志位 (SM182)。

使用示例



当 X0=ON 时, D0 (1000) 减去 D1 (2000) 结果赋给 D10, D10=-1000。

6.3.3 MUL: 整数乘法指令

<div>梯形图：</div> <div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></</div></div>								
--	--	--	--	--	--	--	--	--

操作数说明

S1: 源操作数 1

S2: 源操作数 2

D: 目的操作数

功能说明

当能流有效时, **S1** 乘 **S2**, 运算结果赋予 **D**。

注意事项

MUL 指令的运算结果是 32 位数据。

使用示例



当 X0=ON 时, D0 (1000) 乘以 D1 (2000) 结果赋给 (D10,D11), (D10,D11)=2000000。

6.3.4 DIV：整数除法指令

梯形图： 										适用机型	APLC-TCP				
指令列表：DIV (S1) (S2) (D)										影响标志位	零标志 进位标志 借位标志				
										步长	7				
操作数	类型	适用软元件													变址
S1	INT	常数	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	√
S2	INT	常数	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	√
D	INT			KnY	KnM	KnS	KnLM		D		C	T	V	Z	√

操作数说明

S1：源操作数 1

S2：源操作数 2

D：目的操作数

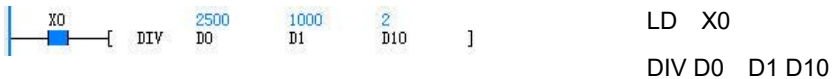
功能说明

当能流有效时，**S1** 除以 **S2**，运算结果赋予 **D**（D 包括两个单元，第一个单元存储商值，第二个单元存储余值）。

注意事项

S2≠0，否则报 0 除错误，不执行除法运算。

使用示例



当 X0=ON 时，D0（2500）除以 D1（1000）结果赋给（D10,D11）。D10=2，D11=500。

6.3.5 SQT：整数算术平方根指令

梯形图： 										适用机型	APLC-TCP				
指令列表：SQT (S) (D)										影响标志位	零标志 进位标志 借位标志				
										步长	5				
操作数	类型	适用软元件													变址
S	INT	常数	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	√
D	INT			KnY	KnM	KnS	KnLM		D		C	T	V	Z	√

操作数说明

S：源操作数

D：目的操作数

功能说明

1. 当能流有效时，**S** 开方，运算结果赋予 **D**。
2. 当运算结果 (**D**) 等于 0 时，置零标志位 (SM180)。运算结果舍去小数时，置借位标志位 (SM182)。

注意事项

S≥0，否则报操作数错误，不执行开方运算。

使用示例



当 X0=ON 时，D0 (1000) 开方结果赋给 D10，D10=31。

6.3.6 INC：整数加一指令

梯形图： 									适用机型		APLC-TCP					
									影响标志位		零标志 进位标志 借位标志					
指令列表： INC (D)									步长		3					
操作数		类型	适用软元件											变址		
D		INT			KnY	KnM	KnS	KnLM		D		C	T	V	Z	√

操作数说明

D：目的操作数

功能说明

当能流有效时，**D** 自增 1。

注意事项

本指令为循环加指令，范围为-32768~32767；C 元件的支持范围为：C0~C199。

使用示例



当 X0=ON 时，D0 (1000) 自增 1，执行后 D0=1001。

6.3.7 DAT：整数减一指令

梯形图： 										适用机型	APLC-TCP				
指令列表：DAT (D)										影响标志位	零标志 进位标志 借位标志				
										步长	3				
操作数	类型	适用软元件													变址
D	INT			KnY	KnM	KnS	KnLM		D		C	T	V	Z	√

操作数说明

D：目的操作数

功能说明

当能流有效时，**D** 自减 1。

注意事项

本指令为循环减，范围为-32768~32767。

使用示例



当 X0=ON 时，D0（1000）自减 1，执行后 D0=999

6.3.8 VABS：整数绝对值指令

梯形图： 										适用机型	APLC-TCP				
指令列表：VABS (S) (D)										影响标志位	零标志 进位标志 借位标志				
										步长	5				
操作数	类型	适用软元件													变址
S	INT	常数	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	√
D	INT			KnY	KnM	KnS	KnLM		D		C	T	V	Z	√

操作数说明

S：源操作数

D：目的操作数

功能说明

当能流有效时，**S** 取绝对值，结果赋予 **D**。

注意事项

S 的范围应为-32767~32767；**S** 值为-32768 时，报操作数非法错误，指令不产生动作。

使用示例



当 X0=ON 时，D0 (-1000) 取绝对值，结果赋给 D10，D10=1000。

6.3.9 NEG：整数取负指令

梯形图：										适用机型	AT100				
[NEG (S) (D)]										影响标志位	零标志 进位标志 借位标志				
指令列表：NEG (S) (D)										步长	5				
操作数	类型	适用软元件													变址
S	INT	常数	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	√
D	INT			KnY	KnM	KnS	KnLM		D		C	T	V	Z	√

操作数说明

S：源操作数

D：目的操作数

功能说明

当能流有效时，**S** 取相反数，结果赋予 **D**。

注意事项

S 的范围应为-32767~32767；当 **S** 值为-32768 时，报操作数非法错误，指令不产生动作。

使用示例



当 X0=ON 时，D0 (1000) 取相反数，结果赋给 D10，D10=-1000。

6.3.10 DADD：长整数加法指令

梯形图： 										适用机型	APLC-TCP				
指令列表：DADD (S1) (S2) (D)										影响标志位	零标志 进位标志 借位标志				
										步长	10				
操作数	类型	适用软元件													变址
S1	DINT	常数	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C		V		√
S2	DINT	常数	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C		V		√
D	DINT			KnY	KnM	KnS	KnLM		D		C		V		√

操作数说明

S1：源操作数 1

S2：源操作数 2

D：目的操作数

功能说明

1. 当能流有效时，**S1** 加 **S2**，运算结果赋予 **D**。
2. 当运算结果 (**D**) 大于 2147483647 时，置进位标志位 (SM181)。运算结果等于 0 时，置零标志位 (SM180)。运算结果小于 -2147483648 时，置借位标志位 (SM182)。

使用示例



当 X0=ON 时，(D0,D1) 的值 (100000) 加上 (D2,D3) 的值 (200000)，结果赋给 (D10,D11)，(D10,D11) =300000。

6.3.11 DSUB：长整数减法指令

梯形图： 										适用机型	APLC-TCP				
指令列表：DSUB (S1) (S2) (D)										影响标志位	零标志 进位标志 借位标志				
										步长	10				
操作数	类型	适用软元件													变址
S1	DINT	常数	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C		V		√
S2	DINT	常数	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C		V		√
D	DINT			KnY	KnM	KnS	KnLM		D		C		V		√

操作数说明

S1：源操作数 1

S2: 源操作数 2

D: 目的操作数

功能说明

- 当能流有效时，**S1** 减 **S2**，运算结果赋予 **D**。
- 当运算结果 (**D**) 大于 2147483647 时，置进位标志位 (**SM181**)。运算结果等于 0 时，置零标志位 (**SM180**)。运算结果小于 -2147483648 时，置借位标志位 (**SM182**)

使用示例



当 X0=ON 时，(D0,D1) 的值 (100000) 减去 (D2,D3) 的值 (200000)，结果赋给 (D10,D11)，(D10,D11) = -100000。

6.3.12 DMUL：长整数乘法指令

<div>梯形图：</div> <div><div><div></div><div></div></div><div><div></div><div></div></div><div>[</div><div>DMUL</div><div>(S1)</div><div>(S2)</div><div>(D)</div><div>]</div></div>									适用机型		APLC-TCP				
									影响标志位		零标志 进位标志 借位标志				
指令列表： DMUL (S1) (S2) (D)									步长		10				
操作数		类型	适用软元件												变址
S1		DINT	常数	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C		V	√
S2		DINT	常数	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C		V	√
D		DINT			KnY	KnM	KnS	KnLM		D		C		V	√

操作数说明

S1: 源操作数 1

S2: 源操作数 2

D: 目的操作数

功能说明

当能流有效时，**S1** 乘 **S2**，运算结果赋予 **D**。

注意事项

DMUL 指令的运算结果是 32 位数据，可能会产生溢出，请用户注意。

使用示例



当 X0=ON 时，(D0,D1) 的值 (83000) 乘以 (D2,D3) 的值 (2000) 结果赋给 (D10,D11)，(D10,D11) = 166000000。

6.3.13 DDIV：长整数除法指令

梯形图： 										适用机型	APLC-TCP				
										影响标志位	零标志 进位标志 借位标志				
指令列表：DDIV (S1) (S2) (D)										步长	10				
操作数	类型	适用软元件													变址
S1	DINT	常数	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C		V		√
S2	DINT	常数	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C		V		√
D	DINT			KnY	KnM	KnS	KnLM		D		C		V		√

操作数说明

S1：源操作数 1

S2：源操作数 2

D：目的操作数

功能说明

当能流有效时，**S1** 除以 **S2**，运算结果赋予 **D**（D 包括 4 个单元，前两个单元存储商值，后两个单元存储余值）

注意事项

S2≠0，否则报 0 除错误，不执行除法运算。

使用示例



当 X0=ON 时，（D0,D1）的值（83000）除以（D2,D3）（2000）结果赋予（D10,D11）、（D12,D13）。（D10,D11）=41，（D12,D13）=1000。

6.3.14 DSQT：长整数算术开方指令

梯形图： 										适用机型	APLC-TCP				
										影响标志位	零标志 进位标志 借位标志				
指令列表：DSQT (S) (D)										步长	7				
操作数	类型	适用软元件													变址
S	DINT	常数	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C		V		√
D	DINT			KnY	KnM	KnS	KnLM		D		C		V		√

操作数说明

S: 源操作数

D: 目的操作数

功能说明

1. 当能流有效时，**S** 开方，运算结果赋予 **D**。
2. 当运算结果 (**D**) 等于 0 时，置零标志位 (SM180)。运算结果舍去小数时，置借位标志位 (SM182)

注意事项

S≥0，否则报操作数错误，不执行开方运算。

使用示例



当 X0=ON 时，(D0,D1) 的值 (83000) 开方，结果赋给 (D10,D11)，(D10,D11) =288。

6.3.15 DINC：长整数增一指令

<div>梯形图：</div> <div><div><div></div><div></div></div><div><div></div><div></div></div><div>[DINC (D)]</div></div>										适用机型		APLC-TCP					
										影响标志位		零标志 进位标志 借位标志					
指令列表：DINC (D)										步长		4					
操作数		类型		适用软元件												变址	
D		DINT				KnY	KnM	KnS	KnLM		D		C		V		√

操作数说明

D: 目的操作数

功能说明

当能流有效时，**D** 自增 1。

注意事项

1. 本指令为循环加指令，范围为-2147483648~2147483647。
2. C 元件的支持范围为：C200~C255。

使用示例



当 X0=ON 时，(D0,D1) 的值 (100000) 自增 1，执行后 (D0,D1) =100001。

6.3.16 DDEC：长整数减一指令

梯形图： 										适用机型		APLC-TCP					
										影响标志位		零标志 进位标志 借位标志					
指令列表：DDAT (D)										步长		4					
操作数		类型		适用软元件												变址	
D		DINT				KnY	KnM	KnS	KnLM		D		C		V		√

操作数说明

D：目的操作数

功能说明

当能流有效时，**D** 自减 1。

注意事项

本指令为循环减，范围为-2147483648~2147483647。

使用示例



当 X0=ON 时，(D0,D1) 的值 (100000) 自减 1，执行后 (D0,D1) =99999。

6.3.17 DVABS：长整数绝对值指令

梯形图： 									适用机型		APLC-TCP				
									影响标志位		零标志 进位标志 借位标志				
指令列表： DVABS (S) (D)									步长		7				
操作数		类型	适用软元件											变址	
S	DINT	常数	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C		V		√
D	DINT			KnY	KnM	KnS	KnLM		D		C		V		√

操作数说明

S：源操作数

D：目的操作数

功能说明

当能流有效时，**S** 取绝对值，结果赋予 **D**。

注意事项

S 的范围应为-2147483647~2147483647；当 **S** 值为-2147483648 时，报操作数非法错误，指令不产生动作。

使用示例



当 X0=ON 时，（D0,D1）的值（-100000）取绝对值，结果赋给（D10,D11），（D10,D11）=100000。

6.3.18 DNEG：长整数取负指令

梯形图：										适用机型	APLC-TCP					
[DNEG (S) (D)]										影响标志位	零标志 进位标志 借位标志					
指令列表：DNEG (S) (D)										步长	7					
操作数	类型	适用软元件														变址
S	DINT	常数	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C		V			√
D	DINT			KnY	KnM	KnS	KnLM		D		C		V			√

操作数说明

S：源操作数

D：目的操作数

功能说明

当能流有效时，**S** 取相反数，结果赋予 **D**。

注意事项

S 的范围应为 -2147483647~2147483647；当 **S** 值为 -2147483648 时，报操作数非法错误，指令不产生动作。

使用示例



当 X0=ON 时，（D0,D1）（100000）取相反数，结果赋给（D10,D11），（D10,D11）= -100000。

6.3.19 SUM：整数累加指令

<div>梯形图：</div> <div><div><div></div><div></div></div><div>[</div><div>SUM</div><div>(S1)</div><div>(S2)</div><div>(D)</div><div>]</div></div>									适用机型		APLC-TCP				
									影响标志位		零标志 进位标志 借位标志				
指令列表：SUM (S1) (S2) (D)									步长		8				
操作数	类型	适用软元件													变址
S1	INT		KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	√
S2	INT	常数	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	√
D	DINT			KnY	KnM	KnS	KnLM		D		C		V		√

操作数说明

S1：源操作数，累加起始单元 **S2**：源操作数，累加数据个数 **D**：目的操作数，累加结果

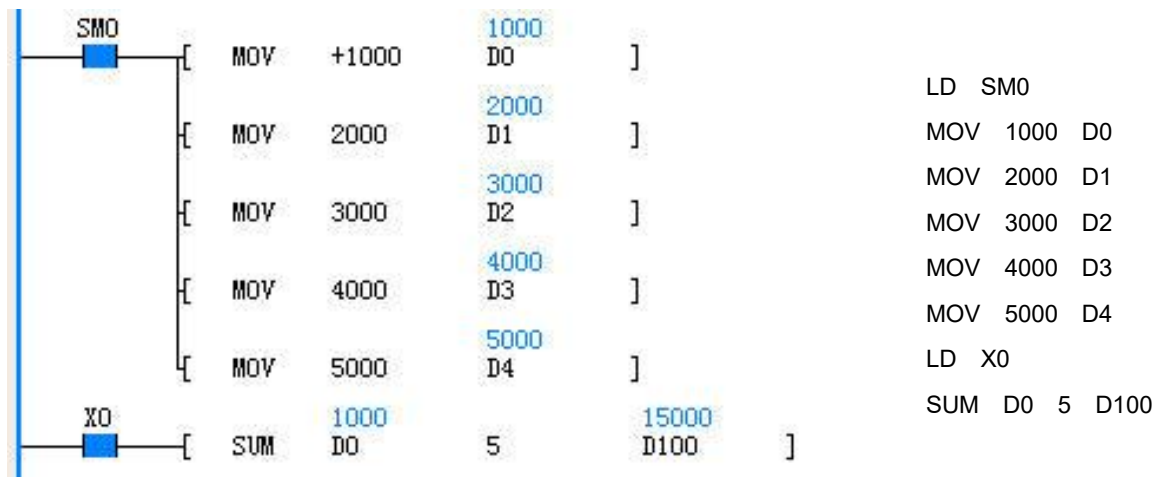
功能说明

当能流有效时，将累加起始单元 (**S1**) 开始的 **S2** 个单元的内容，累加运算后的结果赋给 **D** 单元。

注意事项

1. SUM 指令的运算结果是 32 位数据。
2. $0 \leq S2 \leq 255$ ，否则报操作数错误。
3. 由于 **D** 为 32 位数据，所以进位、借位标志恒为 0。零标志根据最终累加结果确定。

使用示例



当 X0=ON 时，将 D0 起始的 5 个单元的数据累加，结果赋给 (D100,D101)。(D100,D101)=D0+.....+D4=15000。

LD X0

DSUB D0 D2 D10

6.3.20 DSUM：长整数累加指令

梯形图： 										适用机型	APLC-TCP				
										影响标志位	零标志 进位标志 借位标志				
指令列表：DSUM (S1) (S2) (D)										步长	9				
操作数	类型	适用软元件													变址
S1	DINT		KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C		V		√
S2	INT	常数	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	√
D	DINT			KnY	KnM	KnS	KnLM		D		C		V		√

操作数说明

S1：源操作数，累加起始单元

S2：源操作数，累加数据个数

D：目的操作数，累加结果

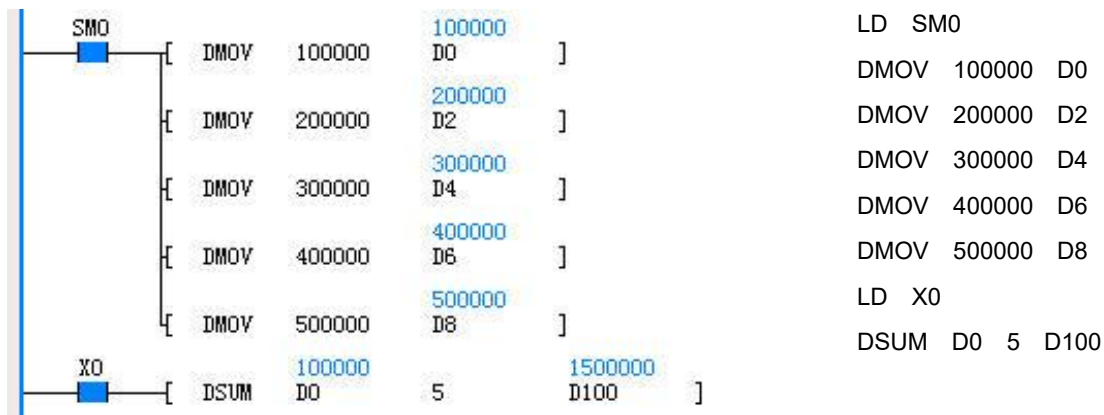
功能说明

当能流有效时，将累加起始单元（**S1**）开始的 **S2**×2 个单元的内容，按长整型数据累加运算后的结果赋给 **D** 单元。

注意事项

0≤**S2**≤255，否则报操作数错误。

使用示例



当 X0=ON 时，将 D0 起始的 5×2 个单元的长整数累加，结果赋给 (D100,D101)。

(D100,D101) = (D0,D1) + + (D8,D9) =1500000。

6.4 浮点算术运算指令

6.4.1 RADD: 浮点数加法指令

梯形图: 										适用机型	APLC-TCP				
指令列表: RADD (S1) (S2) (D)										影响标志位	零标志 进位标志 借位标志				
										步长	10				
操作数	类型	适用软元件													变址
S1	REAL	常数							D				V		√
S2	REAL	常数							D				V		√
D	REAL								D				V		√

操作数说明

S1: 源操作数 1

S2: 源操作数 2

D: 目的操作数

功能说明

- 当能流有效时, **S1** 加 **S2**, 运算结果赋予 **D**。
- 当运算结果 (**D**) 大于 $1.701412e+038$ 或小于 $-1.701412e+038$ 时, 置进位标志位 (SM181)。运算结果等于 0 时, 置零标志位 (SM180)。

使用示例



当 X0=ON 时, (D0,D1) 的值 (-10000.2) 加上 (D2,D3) 的值 (2000.5), 结果赋给 (D10,D11), (D10,D11) = -7999.7。

6.4.2 RSUB: 浮点数减法指令

梯形图: 										适用机型	APLC-TCP				
指令列表: RSUB (S1) (S2) (D)										影响标志位	零标志 进位标志 借位标志				
										步长	10				
操作数	类型	适用软元件													变址
S1	REAL	常数							D				V		√
S2	REAL	常数							D				V		√
D	REAL								D				V		√

操作数说明

S1: 源操作数 1

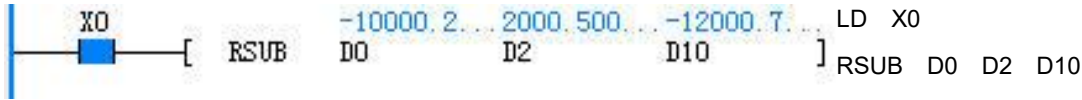
S2: 源操作数 2

D: 目的操作数

功能说明

1. 当能流有效时，**S1** 减 **S2**，运算结果赋予 **D**。
2. 当运算结果 (**D**) 大于 $1.701412e+038$ 或小于 $-1.701412e+038$ 时，置进位标志位 (SM181)。运算结果等于 0 时，置零标志位 (SM180)。

使用示例



当 X0=ON 时，(D0,D1) 的值 (-10000.2) 减去 (D2,D3) 的值 (2000.5)，结果赋给 (D10,D11)，(D10,D11) = -12000.7。

6.4.3 RMUL: 浮点数乘法指令

梯形图: 										适用机型	APLC-TCP				
										影响标志位	零标志 进位标志 借位标志				
指令列表: RMUL (S1) (S2) (D)										步长	10				
操作数	类型	适用软元件												变址	
S1	REAL	常数							D				V		√
S2	REAL	常数							D				V		√
D	REAL								D				V		√

操作数说明

S1: 源操作数 1

S2: 源操作数 2

D: 目的操作数

功能说明

1. 当能流有效时，**S1** 乘 **S2**，运算结果赋予 **D**。
2. 当运算结果 (**D**) 大于 $1.701412e+038$ 或小于 $-1.701412e+038$ 时，置进位标志位 (SM181)。运算结果等于 0 时，置零标志位 (SM180)。

使用示例



当 X0=ON 时，(D0, D1) 的值 (-10000.2) 乘以 (D2, D3) 的值 (2000.5) 结果赋给 (D10,D11)，(D10,D11) = -20005400.0 (实际上乘积应为 -20005400.1，由于计量精度问题，0.1 被舍去了)。

6.4.4 RDIV：浮点数除法指令

梯形图： — —[RDIV (S1) (S2) (D)]									适用机型	APLC-TCP					
									影响标志位	零标志 进位标志 借位标志					
指令列表：RDIV (S1) (S2) (D)									步长	10					
操作数	类型	适用软元件												变址	
S1	REAL	常数							D				V		√
S2	REAL	常数							D				V		√
D	REAL								D				V		√

操作数说明

S1：源操作数 1

S2：源操作数 2

D：目的操作数

功能说明

1. 当能流有效时，**S1** 除以 **S2**，运算结果赋予 **D**（**D** 包括 4 个单元，前两个单元存储商值，后两个单元存储余值）
2. 当运算结果（**D**）大于 $1.701412\text{e}+038$ 或小于 $-1.701412\text{e}+038$ 时，置进位标志位（SM181）。运算结果等于 0 时，置零标志位（SM180）

注意事项

S2≠0，否则报 0 除错误，不执行除法运算。

使用示例



当 X0=ON 时，（D0,D1）=-10000.2 除以（D2,D3）=2000.5 结果赋给（D10,D11）。（D10,D11）=-4.998850。

6.4.5 RSQT：浮点数算术开方指令

梯形图： — —[RSQT (S) (D)]									适用机型	APLC-TCP					
									影响标志位	零标志 进位标志 借位标志					
指令列表：RSQT (S) (D)									步长	7					
操作数	类型	适用软元件												变址	

S	REAL	常数							D				V		√
D	REAL								D				V		√

操作数说明**S**: 源操作数**D**: 目的操作数**功能说明**

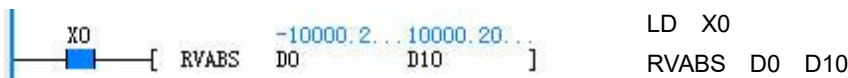
1. 当能流有效时，**S** 开方，运算结果赋予 **D**。
2. 当运算结果 (**D**) 等于 0 时，置零标志位 (SM180)。

注意事项**S**≥0，否则报操作数错误，不执行开方运算。**使用示例**

当 X0=ON 时，(D0,D1) 的值 (10000.2) 开方，结果赋给 (D10,D11)，(D10,D11) =100.000999。

6.4.6 RVABS: 浮点数绝对值指令

梯形图： 										适用机型		APLC-TCP				
										影响标志位		零标志 进位标志 借位标志				
指令列表：RVABS (S) (D)										步长		7				
操作数	类型	适用软元件												变址		
S	REAL	常数							D				V		√	
D	REAL								D				V		√	

操作数说明**S**: 源操作数**D**: 目的操作数**功能说明**当能流有效时，**S** 取绝对值，结果赋予 **D**。**使用示例**

当 X0=ON 时，(D0,D1) 的值 (-10000.2) 取绝对值，结果赋给 (D10,D11)，(D10,D11) =10000.2。

6.4.7 RNEG: 浮点数取负指令

梯形图:									适用机型	APLC-TCP					
[RNEG (S) (D)]									影响标志位	零标志 进位标志 借位标志					
指令列表: RNEG (S) (D)									步长	7					
操作数	类型	适用软元件												变址	
S	REAL	常数							D				V		√
D	REAL								D				V		√

操作数说明

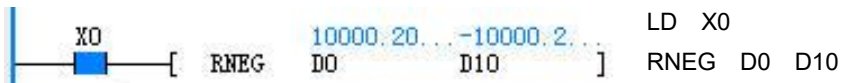
S: 源操作数

D: 目的操作数

功能说明

当能流有效时，**S** 取相反数，结果赋予 **D**。

使用示例



当 X0=ON 时，(D0,D1) =10000.2 取相反数，结果赋给 (D10,D11)，(D10,D11) =-10000.2

6.4.8 SIN: 浮点数 SIN 指令

梯形图:									适用机型	APLC-TCP					
[SIN (S) (D)]									影响标志位	零标志 进位标志 借位标志					
指令列表: SIN (S) (D)									步长	7					
操作数	类型	适用软元件												变址	
S	REAL	常数							D				V		√
D	REAL								D				V		√

操作数说明

S: 源操作数

D: 目的操作数

功能说明

1. 当能流有效时，求 **S**（单位是弧度）的 SIN 值，结果赋予 **D**。
2. 当运算结果（**D**）等于 0 时，置零标志位（SM180）

使用示例



当 X0=ON 时，（D0,D1）=1.57 取 SIN 值，结果赋给（D10,D11），（D10,D11）=1。

6.4.9 COS: 浮点数 COS 指令

梯形图:										适用机型	APA40R			
— — [COS (S) (D)]										影响标志位	零标志 进位标志 借位标志			
指令列表: COS (S) (D)										步长	7			
操作数	类型	适用软元件										变址		
S	REAL	常数								D			V	√
D	REAL									D			V	√

操作数说明

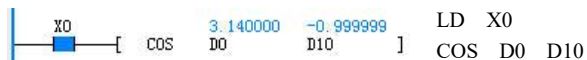
S: 源操作数

D: 目的操作数

功能说明

1. 当能流有效时，求 **S**（单位是弧度）的 COS 值，结果赋予 **D**。
2. 当运算结果（**D**）等于 0 时，置零标志位（SM180）。

使用示例



当 X0=ON 时，（D0,D1）=3.14 求 COS 值，结果赋给（D10,D11），（D10,D11）=-0.999999。

6.4.10 TAN：浮点数 TAN 指令

梯形图： 									适用机型	APLC-TCP					
									影响标志位	零标志 进位标志 借位标志					
指令列表：TAN (S) (D)									步长	7					
操作数	类型	适用软元件												变址	
S	REAL	常数							D				V		√
D	REAL								D				V		√

操作数说明

S：源操作数

D：目的操作数

功能说明

1. 当能流有效时，求 **S**（单位是弧度）的 TAN 值，结果赋予 **D**。
2. 当运算结果（**D**）大于 $1.701412e+038$ 或小于 $-1.701412e+038$ 时，置进位标志位（SM181）。运算结果等于 0 时，置零标志位（SM180）

使用示例



当 X0=ON 时，(D0,D1) =1.57 求 TAN 值，结果赋给 (D10,D11)，(D10,D11) =1255.848398。

6.4.11 POWER：浮点数求幂运算

梯形图： 									适用机型	APLC-TCP					
									影响标志位	零标志 进位标志 借位标志					
指令列表：POWER (S1) (S2) (D)									步长	10					
操作数	类型	适用软元件												变址	
S1	REAL	常数							D				V		√
S2	REAL	常数							D				V		√
D	REAL								D				V		√

操作数说明

S1：源操作数 1

S2：源操作数 2

D：目的操作数

功能说明

1. 当能流有效时，求 **S1** 的 **S2** 次幂，运算结果赋予 **D**。
2. 当运算结果 (**D**) 大于 $1.701412e+038$ 或小于 $-1.701412e+038$ 时，置进位标志位 (SM181)。运算结果等于 0 时，置零标志位 (SM180)。

注意事项

1. 当 **S1**=0 且 **S2**≤0 时，报操作数值错误，不执行运算。
2. 当 **S1**<0 且 **S2** 的尾数部分不为 0 时，报操作数值错误，不执行运算。

使用示例



当 X0=ON 时，求 (D0,D1) 的 (D2,D3) 次幂 (即 55.0 的 3.0 次幂)，结果赋给 (D10,D11)，(D10,D11)=166375.0。

6.4.12 LN：浮点数自然对数指令 LN

梯形图：										适用机型	APLC-TCP				
[LN (S) (D)]										影响标志位	零标志 进位标志 借位标志				
指令列表：LN (S) (D)										步长	7				
操作数	类型	适用软元件												变址	
S	REAL	常数								D				V	√
D	REAL									D				V	√

操作数说明

S：源操作数

D：目的操作数

功能说明

1. 当能流有效时，求 **S** 的 LN 值，结果赋予 **D**。
2. 当运算结果 (**D**) 大于 $1.701412e+038$ 或小于 $-1.701412e+038$ 时，置进位标志位 (SM181)。运算结果等于 0 时，置零标志位 (SM180)。

使用示例



当 X0=ON 时，(D0,D1)=1000.0 求 LN 值，结果赋给 (D10,D11)，(D10,D11)=6.907755。

6.4.13 EXP：浮点数自然数幂指令

梯形图： 										适用机型	APLC-TCP				
										影响标志位	零标志 进位标志 借位标志				
指令列表：EXP (S) (D)										步长	7				
操作数	类型	适用软元件												变址	
S	REAL	常数							D				V		√
D	REAL								D				V		√

操作数说明

S：源操作数

D：目的操作数

功能说明

1. 当能流有效时，求 **S** 的 EXP 值，结果赋予 **D**。
2. 当运算结果 (**D**) 大于 $1.701412e+038$ 或小于 $-1.701412e+038$ 时，置进位标志位 (SM181)。运算结果等于 0 时，置零标志位 (SM180)

使用示例



当 X0=ON 时，(D0,D1) =10.0 求 EXP 值，结果赋给 (D10,D11)，(D10,D11) =22026.464844。

6.4.14 RSUM：浮点数累加指令

梯形图： 										适用机型	APLC-TCP				
										影响标志位	零标志 进位标志 借位标志				
指令列表：RSUM (S1) (S2) (D)										步长	9				
操作数	类型	适用软元件												变址	
S1	REAL								D				V		√
S2	INT	常数	KnX	KnY	KnM	KnS	KnLM	KnSM	D				V		√
D	REAL								D				V		√

操作数说明

S1：源操作数，累加起始单元

S2：源操作数，累加数据个数

D：目的操作数，累加结果

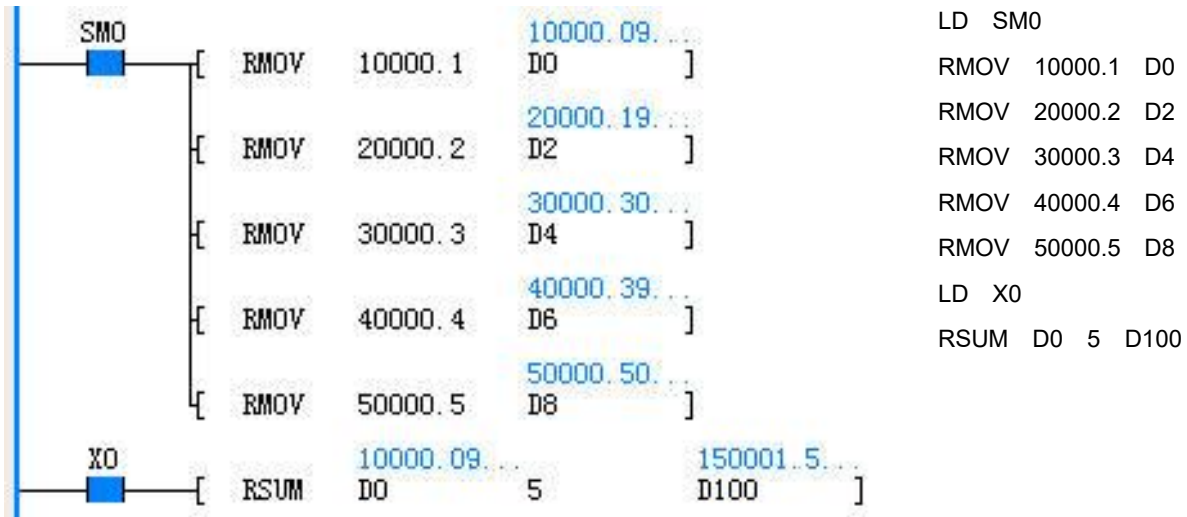
功能说明

当能流有效时，将累加起始单元（**S1**）开始的 **S2**×2 个单元的内容，按浮点型数据累加，运算后的结果赋给 **D** 单元。

注意事项

1. $0 \leq S2 \leq 255$ ，否则报操作数错误。
2. 出现溢出情况，将不再执行累加运算。

使用示例



当 X0=ON 时，将 D0 起始的 5×2 个单元的浮点数累加，结果赋给 (D100,D101)。

$(D100,D101) = (D0,D1) + \dots + (D8,D9) = 150001.5$ 。

6.5 数值转换指令

6.5.1 DTI：长整数转换整数指令

梯形图： 										适用机型	APLC-TCP				
										影响标志位	零标志 进位标志 借位标志				
指令列表：DTI (S) (D)										步长	6				
操作数	类型	适用软元件													变址
S	DINT	常数	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C		V		√
D	INT			KnY	KnM	KnS	KnLM		D		C	T	V	Z	√

操作数说明

S：源操作数

D：目的操作数

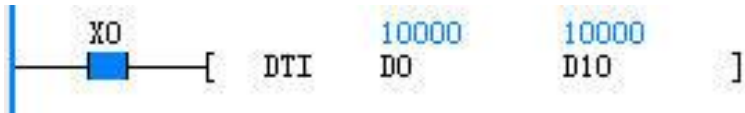
功能说明

当能流有效时，**S** 由长整数转换成整数，结果赋予 **D**。

注意事项

当 **S** > 32767 或 **S** < -32768 时，系统报操作数错误，不执行转换，**D** 内容不变。

使用示例



```
LD X0
DTI D0 D10
```

当 X0=ON 时，(D0, D1) = 10000 由长整数转换成整数，赋给 D10。D10=10000。

6.5.2 ITD：整数转换长整数指令

梯形图： 										适用机型	APLC-TCP				
										影响标志位	零标志 进位标志 借位标志				
指令列表：ITD (S) (D)										步长	6				
操作数	类型	适用软元件													变址
S	INT	常数	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	√
D	DINT			KnY	KnM	KnS	KnLM		D		C		V		√

操作数说明

S：源操作数

D：目的操作数

功能说明

当能流有效时，**S** 由整数转换成长整数，结果赋予 **D**。

使用示例



当 X0=ON 时，D0=1000 由整数转换成长整数，赋给 D10，(D10,D11) =1000。

6.5.3 FLT：整数转换浮点数指令

梯形图：									适用机型	APLC-TCP					
[FLT (S) (D)]									影响标志位	零标志 进位标志 借位标志					
指令列表：FLT (S) (D)									步长	6					
操作数	类型	适用软元件													变址
S	INT	常数	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	√
D	REAL								D				V		√

操作数说明

S：源操作数

D：目的操作数

功能说明

当能流有效时，**S** 由整数转换成浮点数，结果赋予 **D**。

使用示例



当 X0=ON 时，D0=10005 由整数转换成浮点数，赋给 (D10,D11)，(D10,D11) =10005.0。

6.5.4 DFLT：长整数转换浮点数指令

梯形图： — —[DFLT (S) (D)]										适用机型	APLC-TCP				
										影响标志位	零标志 进位标志 借位标志				
指令列表：DFLT (S) (D)										步长	7				
操作数	类型	适用软元件													变址
S	DINT	常数	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C		V		√
D	REAL								D				V		√

操作数说明

S：源操作数

D：目的操作数

功能说明

当能流有效时，**S** 由长整数转换成浮点数，结果赋予 **D**。

使用示例



当 X0=ON 时，(D0,D1) =100000，由长整数转换成浮点数，赋给 (D10,D11)，(D10,D11) =100000.0。

6.5.5 INT：浮点数转换整数指令

梯形图： — —[INT (S) (D)]										适用机型	APLC-TCP				
										影响标志位	零标志 进位标志 借位标志				
指令列表：INT (S) (D)										步长	6				
操作数	类型	适用软元件													变址
S	REAL	常数							D				V		√
D	INT			KnY	KnM	KnS	KnLM		D		C	T	V	Z	√

操作数说明

S：源操作数

D：目的操作数

功能说明

- 当能流有效时，**S** 由浮点数转换成整数，结果赋予 **D**。
- 该指令影响零标志、借位标志。当转换结果为零时，置零标志。结果舍去小数时，置借位标志。当结果超出长整形数据范围时，置进位（溢出）标志。

注意事项

当 $S > 32767$ 时, $D=32767$ 。 $S < -32768$ 时, $D=-32768$, 同时置进位（溢出）标志位。

使用示例



当 $X0=ON$ 时, $(D0,D1) = 10000.5$, 由浮点数转换成整数, 赋给 $D10$, $D10=10000$ 。

6.5.6 DINT：浮点数转换长整数指令

梯形图： [DINT (S) (D)]									适用机型	APLC-TCP					
指令列表：DINT (S) (D)									影响标志位	零标志 进位标志 借位标志					
									步长	7					
操作数	类型	适用软元件												变址	
S	REAL	常数							D				V		√
D	DINT			KnY	KnM	KnS	KnLM		D		C		V		√

操作数说明

S：源操作数

D：目的操作数

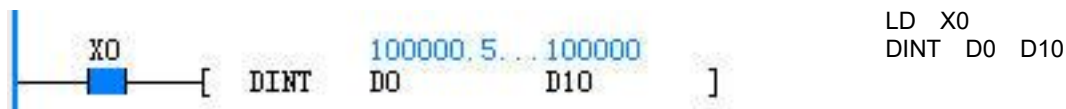
功能说明

1. 当能流有效时, S 由浮点数转换成长整数, 结果赋予 D 。
2. 当转换结果为零时, 置零标志。结果舍去小数时, 置借位标志。当结果超出长整型数据范围时, 置进位（溢出）标志。

注意事项

当 $S > 2147483647$ 时, $D=2147483647$ 。 $S < -2147483648$ 时, $D=-2147483648$, 同时置进位（溢出）标志位。

使用示例



当 $X0=ON$ 时, $(D0,D1) = 100000.5$ 由浮点数转换成长整数, 赋给 $(D10,D11)$, $(D10,D11) = 100000$ 。

6.5.7 BCD：字转换 16 位 BCD 码指令

梯形图： 										适用机型	APLC-TCP				
指令列表：BCD (S) (D)										影响标志位	零标志 进位标志 借位标志				
										步长	5				
操作数	类型	适用软元件													变址
S	WORD	常数	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	√
D	WORD			KnY	KnM	KnS	KnLM		D		C	T	V	Z	√

操作数说明

S：源操作数，≤9999

D：目的操作数

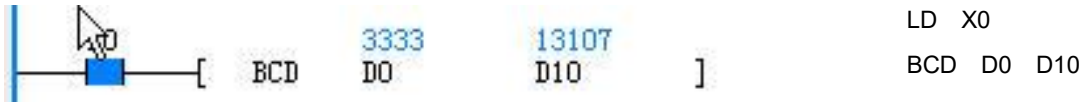
功能说明

当能流有效时，**S** 由整数转换成 16 位 BCD 码，结果赋予 **D**。

注意事项

当 **S** > 9999 时，系统报操作数错误，不执行转换，**D** 内容不变。

使用示例



当 X0=ON 时，D0=0x0D05 (3333) 由整数转换成 16 位 BCD 码，赋给 D10，D10=0x3333 (13107)。

6.5.8 DBCD：双字转换 32 位 BCD 码指令

梯形图： 										适用机型	APLC-TCP				
指令列表：DBCD (S) (D)										影响标志位	零标志 进位标志 借位标志				
										步长	7				
操作数	类型	适用软元件													变址
S	DWORD	常数	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C		V		√
D	DWORD			KnY	KnM	KnS	KnLM		D		C		V		√

操作数说明

S：源操作数，≤99999999

D：目的操作数

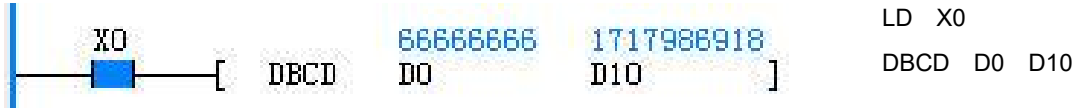
功能说明

当能流有效时，**S** 由长整数转换成 32 位 BCD 码，结果赋予 **D**。

注意事项

当 $S > 99999999$ 时，系统报操作数错误，不执行转换， D 内容不变。

使用示例



当 $X0=ON$ 时， $(D0,D1)=0x3F940AA(66666666)$ 由长整数转换成 32 位 BCD 码，赋给 $(D10,D11)$ ， $(D10,D11)=0x66666666(1717986918)$ 。

6.5.9 BIN：16 位 BCD 码转换字指令

梯形图： [BIN (S) (D)]									适用机型		APLC-TCP				
									影响标志位		零标志 进位标志 借位标志				
指令列表： BIN (S) (D)									步长		5				
操作数	类型	适用软元件													变址
S	WORD	常数	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	√
D	WORD			KnY	KnM	KnS	KnLM		D		C	T	V	Z	√

操作数说明

S：源操作数，数据格式必须符合 BCD 码格式

D：目的操作数。

功能说明

当能流有效时， S 由 16 位 BCD 码转换成整数，结果赋予 D 。

注意事项

当 S 的数据格式不符合 BCD 码格式时，系统报操作数错误，不执行转换， D 内容不变。

使用示例



当 $X0=ON$ 时， $D0=0x5555(21845)$ 由 16 位 BCD 码转换成整数，赋给 $D10$ ， $D10=0x15B3(5555)$ 。

6.5.10 DBIN：32 位 BCD 码转换双字指令

梯形图： — — [DBIN (S) (D)]										适用机型	APLC-TCP				
										影响标志位	零标志 进位标志 借位标志				
指令列表：DBIN (S) (D)										步长	7				
操作数	类型	适用软元件													变址
S	DWORD	常数	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C		V		√
D	DWORD			KnY	KnM	KnS	KnLM		D		C		V		√

操作数说明

S：源操作数

D：目的操作数

功能说明

1. 能流有效时，**S** 由 32 位 BCD 码转换成长整数，结果赋予 **D**。
2. **S** 的数据格式必须符合 BCD 码格式

注意事项

当 **S** 的数据格式不符合 BCD 码格式时，系统报操作数错误，不执行转换，**D** 内容不变。

使用示例



当 X0=ON 时，(D0,D1) = 0x99999999 (2576980377) 由 32 位 BCD 码转换成长整数，赋给 (D10,D11)，(D10,D11) = 0x5F5E0FF (99999999)。

6.5.11 GRY：字转换 16 位格雷码指令

梯形图： [GRY (S) (D)]										适用机型	APLC-TCP				
										影响标志位	零标志 进位标志 借位标志				
指令列表：GRY (S) (D)										步长	5				
操作数	类型	适用软元件													变址
S	WORD	常数	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	√
D	WORD			KnY	KnM	KnS	KnLM		D		C	T	V	Z	√

操作数说明

S：源操作数

D：目的操作数

功能说明

当能流有效时，**S** 由整数转换成 16 位格雷码，结果赋予 **D**。

使用示例



当 X0=ON 时，D0=0xAAAA (43690) 由整数转换成 16 位格雷码，赋给 D10，D10=0xFFFF (65535)。

6.5.12 DGRY：双字转换 32 位格雷码指令

梯形图：										适用机型	AT100				
[DGRY (S) (D)]										影响标志位	零标志 进位标志 借位标志				
指令列表：DGRY (S) (D)										步长	7				
操作数	类型	适用软元件													变址
S	DWORD	常数	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C		V		√
D	DWORD			KnY	KnM	KnS	KnLM		D		C		V		√

操作数说明

S：源操作数

D：目的操作数

功能说明

当能流有效时，**S** 由长整数转换成 32 位格雷码，结果赋予 **D**。

使用示例



当 X0=ON 时，(D0,D1) =0x88888888 (2290649224) 由长整数转换成 32 位格雷码，赋给 (D10,D11)，(D10,D11) =0xCCCCCCCC (3435973836)。

6.5.13 GBIN：16 位格雷码转换字指令

梯形图： 										适用机型	APLC-TCP				
指令列表： GBIN (S) (D)										影响标志位	零标志 进位标志 借位标志				
										步长	5				
操作数	类型	适用软元件													变址
S	WORD	常数	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	√
D	WORD			KnY	KnM	KnS	KnLM		D		C	T	V	Z	√

操作数说明

S：源操作数

D：目的操作数

功能说明

当能流有效时，**S** 由 16 位格雷码转换成整数，结果赋予 **D**。

使用示例



当 X0=ON 时，D0=0xFFFF (65535) 由 16 位格雷码转换成整数，赋给 D10，D10=0xAAAA (43690)。

6.5.14 DGBIN：32 位格雷码转换双字指令

梯形图： 										适用机型	APLC-TCP				
指令列表： DGBIN (S) (D)										影响标志位	零标志 进位标志 借位标志				
										步长	7				
操作数	类型	适用软元件													变址
S	DWORD	常数	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C		V		√
D	DWORD			KnY	KnM	KnS	KnLM		D		C		V		√

操作数说明

S：源操作数

D：目的操作数

功能说明

当能流有效时，**S** 由 32 位格雷码转换成长整数，结果赋予 **D**。

使用示例



当 X0=ON 时，(D0,D1)=0xCCCCCCCC (3435973836) 由 32 位格雷码转换成长整数，赋给 (D10,D11)，(D10,D11)=0x88888888 (2290649224)。

6.5.15 SEG：字转换 7 段码指令

梯形图： 										适用机型	APLC-TCP				
										影响标志位	零标志 进位标志 借位标志				
指令列表：SEG (S) (D)										步长	5				
操作数	类型	适用软元件													变址
S	WORD	常数	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	√
D	WORD			KnY	KnM	KnS	KnLM		D		C	T	V	Z	√

操作数说明

S：源操作数， $S \leq 15$

D：目的操作数

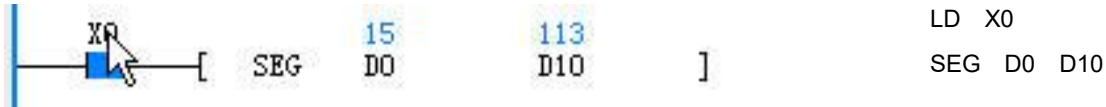
功能说明

当能流有效时，**S** 由整数转换成 7 段码，结果赋予 **D**。

注意事项

当 $S > 15$ 时，系统报操作数错误，不执行转换，**D** 内容不变。

使用示例



当 $X0=ON$ 时， $D0=0x0F$ (15) 由整数转换成 7 段码，赋给 $D10$ ， $D10=0x71$ (113)。

6.5.16 ASC：ASCII 码转换指令

梯形图： 										适用机型	APLC-TCP				
										影响标志位	零标志 进位标志 借位标志				
指令列表：ASC (S1~S8) (D)										步长	19				
操作数	类型	适用软元件													变址
S1	WORD	常数													
S2	WORD	常数													
S3	WORD	常数													
S4	WORD	常数													
S5	WORD	常数													
S6	WORD	常数													
S7	WORD	常数													
S8	WORD	常数													
D	WORD								D		C	T	V	Z	√

操作数说明

S1~S8: 源操作数 (不足 8 个, 剩余字符填 0)

仅支持 ASCII 码为 0x21~0x7E 的字符 (键盘输入, 不足 8 个字符用 0x00 补充)

D: 目的操作数

功能说明

当能流有效时, 将字符串 **S1~S8** 转换成 ASCII 码, 结果赋予 **D** 开始元件中。当 SM186=OFF, 每个 **D** 元件高低字节存两个 ASCII 码数据, 当 SM186=ON, 每个 **D** 元件低字节存 1 个 ASCII 码数据。

使用示例



当 M0=ON, 执行 ASCII 转换, 数据存放分两种方式:

- 若 SM186=OFF, 执行结果是: D0=0x3231, D1=0x3433, D2=0x3635, D3=0x3837。
- 若 SM186=ON, 执行结果是: D0=0x31, D1=0x32, D2=0x33, D3=0x34, D4=0x35, D5=0x36, D6=0x37, D7=0x38。

6.5.17 ITA: 16 位 16 进制数转换 ASCII 码指令

梯形图： 									适用机型		APLC-TCP				
									影响标志位		零标志 进位标志 借位标志				
指令列表：ITA (S1) (D) (S2)									步长		7				
操作数	类型	适用软元件													变址
S1	WORD	常数	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	√
D	WORD			KnY	KnM	KnS	KnLM		D		C	T	V	Z	√
S2	WORD	常数	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	√

操作数说明

S1: 待转换源 16 进制数据; **D**: 目的操作数

S2: ASCII 码数量 ($1 \leq S2 \leq 256$)

功能说明

当能流有效时, 将 **S1** 元件开始的 16 进制数转换成 **S2** 个 ASCII 码, 结果赋予 **D** 开始元件中。当 SM186=OFF, 每个 **D** 元件高低字节存两个 ASCII 码数据, 当 SM186=ON, 每个 **D** 元件低字节存 1 个 ASCII 码数据。

注意事项

1. **S1**、**D** 使用 Kn 寻址时, Kn=4。
2. 当 **S2** 不在 1~256 之间时, 系统报操作数错误, 不执行转换, **D** 内容不变。
3. 若 **S1** 为常数, **S2**≥4 时, 默认 **S2**=4 处理。按默认处理不再报操作数错误。

使用示例

	源数据：0x9876。 LD M0 ITA 16#9876 D20 6
--	---

当 M0=ON，执行 ITA 转换，数据存放分两种方式：

- 若 SM186=OFF，执行结果是：D20=0x3839，D21=0x3637。
- 若 SM186=ON，执行结果是：D20=0x39，D21=0x38，D22=0x37，D23=0x36。

6.5.18 ATI：ASCII 码数转换 16 位 16 进制指令

梯形图： 										适用机型	APLC-TCP				
指令列表：ATI (S1) (D) (S2)										影响标志位	零标志 进位标志 借位标志				
										步长	7				
操作数	类型	适用软元件													变址
S1	WORD	常数	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	√
D	WORD			KnY	KnM	KnS	KnLM		D		C	T	V	Z	√
S2	WORD	常数	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	√

操作数说明

S1：待转换源 ASCII 码数据

0x30≤**S1**≤0x39 或 0x41≤**S1**≤0x46（当 SM186=OFF，需要 **S1** 高低字节都符合此范围）

D：目的操作数

S2：ASCII 码数量（1≤**S2**≤256）

功能说明

当能流有效时，将 **S1** 元件开始的 **S2** 个 ASCII 码数据转换成 16 进制数据，结果每 4 位保存在 **D** 开始元件中。当 SM186=OFF，每个 **D** 元件高低字节存的两个 ASCII 码数据，当 SM186=ON，每个 **D** 元件低字节存 1 个 ASCII 码数据。

注意事项

1. **S1**、**D** 使用 Kn 寻址时，Kn=4。
2. 当 **S1** 不在 0x30~0x39 或 0x41~0x46，或者 **S2** 不在 1~256 之间时，系统报操作数错误，不执行转换，**D** 内容不变。
3. 若 **S1** 为常数，当 SM186=OFF 且 **S2**≥2 时，默认 **S2**=2 处理。当 SM186=ON 且 **S2**≥1 时，默认 **S2**=1 处理。按默认处理不再报操作数错误。

使用示例

	LD M0 ATI D10 D30 4
--	------------------------

源数据：D10=0x3938，D11=0x3736，D12=0x3534，D13=0x3332。

当 M0=ON，执行 ATI 转换，根据数据存放分方式，产生结果如下：

- 若 SM186=OFF，执行结果是：D30=0x8967。
- 若 SM186=ON，执行结果是：D30=0x8642。

6.6 字逻辑运算

6.6.1 WAND：字与指令

梯形图： 										适用机型	APLC-TCP				
指令列表： WAND (S1) (S2) (D)										影响标志位					
										步长	7				
操作数	类型	适用软元件													变址
S1	WORD	常数	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	√
S2	WORD	常数	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	√
D	WORD			KnY	KnM	KnS	KnLM		D		C	T	V	Z	√

操作数说明

S1：源操作数 1

S2：源操作数 2

D：目的操作数

功能说明

当能流有效时，**S1** 与 **S2** 按位逻辑与，结果赋予 **D**。

使用示例



当 X0=ON 时，D0=2#1011011010010011 (46739) 与 D1=2#1001001100101110 (37678) 位逻辑与，结果赋给 D10，D10=2#1001001000000010 (37378)。

6.6.2 WOR：字或指令

梯形图： 										适用机型	APLC-TCP				
指令列表： WOR (S1) (S2) (D)										影响标志位					
										步长	7				
操作数	类型	适用软元件													变址
S1	WORD	常数	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	√
S2	WORD	常数	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	√
D	WORD			KnY	KnM	KnS	KnLM		D		C	T	V	Z	√

操作数说明

S1：源操作数 1

S2：源操作数 2

D：目的操作数

功能说明

当能流有效时，**S1** 与 **S2** 按位逻辑或，结果赋予 **D**。

使用示例



当 X0=ON 时，D0=2#1011011010010011 (46739) 与 D1=2#1001001100101110 (37678) 位逻辑或，结果赋给 D10，D10=2#101101110111111 (47039)。

6.6.3 WXOR：字异或运算

梯形图： [WXOR (S1) (S2) (D)]										适用机型	APLC-TCP				
指令列表：WXOR (S1) (S2) (D)										步长	7				
操作数	类型	适用软元件													变址
S1	WORD	常数	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	√
S2	WORD	常数	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	√
D	WORD			KnY	KnM	KnS	KnLM		D		C	T	V	Z	√

操作数说明

S1：源操作数 1

S2：源操作数 2

D：目的操作数

功能说明

当能流有效时，**S1** 与 **S2** 按位逻辑异或，结果赋予 **D**。

使用示例



当 X0=ON 时，D0=2#1011011010010011 (46739) 与 D1=2#1001001100101110 (37678) 位逻辑异或，结果赋给 D10，D10=2#0010010110111101 (9661)。

6.6.4 WINV：字取反运算

梯形图： [WINV (S) (D)]										适用机型	APLC-TCP				
指令列表：WINV (S) (D)										步长	5				
操作数	类型	适用软元件													变址
S	WORD	常数	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	√
D	WORD			KnY	KnM	KnS	KnLM		D		C	T	V	Z	√

操作数说明

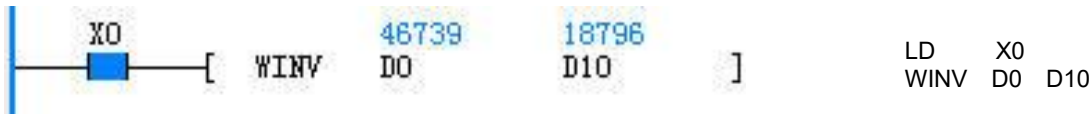
S: 源操作数

D: 目的操作数

功能说明

当能流有效时，对 **S** 按位逻辑取反，结果赋予 **D**。

使用示例



当 X0=ON 时，对 D0= (46739) 按位逻辑取反，结果赋给 D10, D10= (18796)。

6.6.5 DWAND: 双字与指令

梯形图: 										适用机型	APLC-TCP				
指令列表: DWAND (S1) (S1) (D)										影响标志位					
										步长	10				
操作数	类型	适用软元件													变址
S1	DWORD	常数	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C		V		√
S2	DWORD	常数	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C		V		√
D	DWORD			KnY	KnM	KnS	KnLM		D		C		V		√

操作数说明

S1: 源操作数 1

S2: 源操作数 2

D: 目的操作数

功能说明

当能流有效时，**S1** 与 **S2** 按位逻辑与，结果赋予 **D**。

使用示例



当 X0=ON 时，(D0,D1) =2#10110010101001101110011001010010 (2997282386) 与 (D2, D3) =2#00111010001110110011000100110011 (976957747) 位逻辑与，结果赋给 (D10, D11)，(D10, D11) =2#00110010001000100010000000010010 (841097234)。

6.6.6 DWOR：双字或指令

梯形图： 										适用机型	APLC-TCP				
指令列表： DWOR (S1) (S2) (D)										影响标志位					
										步长	10				
操作数	类型	适用软元件													变址
S1	DWORD	常数	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C		V		√
S2	DWORD	常数	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C		V		√
D	DWORD			KnY	KnM	KnS	KnLM		D		C		V		√

操作数说明

S1：源操作数 1

S2：源操作数 2

D：目的操作数

功能说明

当能流有效时，**S1** 与 **S2** 按位逻辑或，结果赋予 **D**。

使用示例



当 X0=ON 时，（D0, D1）=2#10110010101001101110011001010010（2997282386）与（D2, D3）=2#00111010001110110011000100110011（976957747）位逻辑或，结果赋给（D10, D11），（D10, D11）=2#1011101010111111111011101110011（3133142899）。

6.6.7 DWXOR：双字异或指令

梯形图： 										适用机型	APLC-TCP				
指令列表： DWXOR (S1) (S2) (D)										影响标志位					
										步长	10				
操作数	类型	适用软元件													变址
S1	DWORD	常数	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C		V		√
S2	DWORD	常数	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C		V		√
D	DWORD			KnY	KnM	KnS	KnLM		D		C		V		√

操作数说明

S1：源操作数 1

S2：源操作数 2

D：目的操作数

功能说明

当能流有效时，**S1** 与 **S2** 按位逻辑异或，结果赋予 **D**。

使用示例



当 $X0=ON$ 时， $(D0,D1) = 2\#10110010101001101110011001010010$ (2997282386) 与 $(D2,D3) = 2\#00111010001110110011000100110011$ (976957747) 按位逻辑异或，结果赋给 $(D10, D11)$ ， $(D10, D11) = 2\#10001000100111011101011101100001$ (2292045665)。

6.6.8 DWINV：双字取反指令

梯形图：										适用机型	APLC-TCP				
— — [DWINV (S) (D)]										影响标志位					
指令列表：DWINV (S) (D)										步长	7				
操作数	类型	适用软元件												变址	
S	DWORD	常数	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C		V		√
D	DWORD			KnY	KnM	KnS	KnLM		D		C		V		√

操作数说明

S：源操作数

D：目的操作数

功能说明

当能流有效时，对 **S** 按位逻辑取反，结果赋予 **D**。

使用示例



当 $X0=ON$ 时，对 $(D0,D1) = 2\#10110010101001101110011001010010$ (2997282386) 按位逻辑取反，结果赋给 $(D10,D11)$ ， $(D10,D11) = 2\#01001101010110010001100110101101$ (1297684909)。

6.7 位移动旋转指令

6.7.1 ROR: 16 位循环右移指令

梯形图: 										适用机型	APLC-TCP				
指令列表: ROR (S1) (D) (S2)										影响标志位	进位标志 SM181				
										步长	7				
操作数	类型	适用软元件													变址
S1	WORD	常数	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	√
D	WORD			KnY	KnM	KnS	KnLM		D		C	T	V	Z	√
S2	INT	常数	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	√

操作数说明

S1: 源操作数 1

D: 目的操作数

S2: 源操作数 2

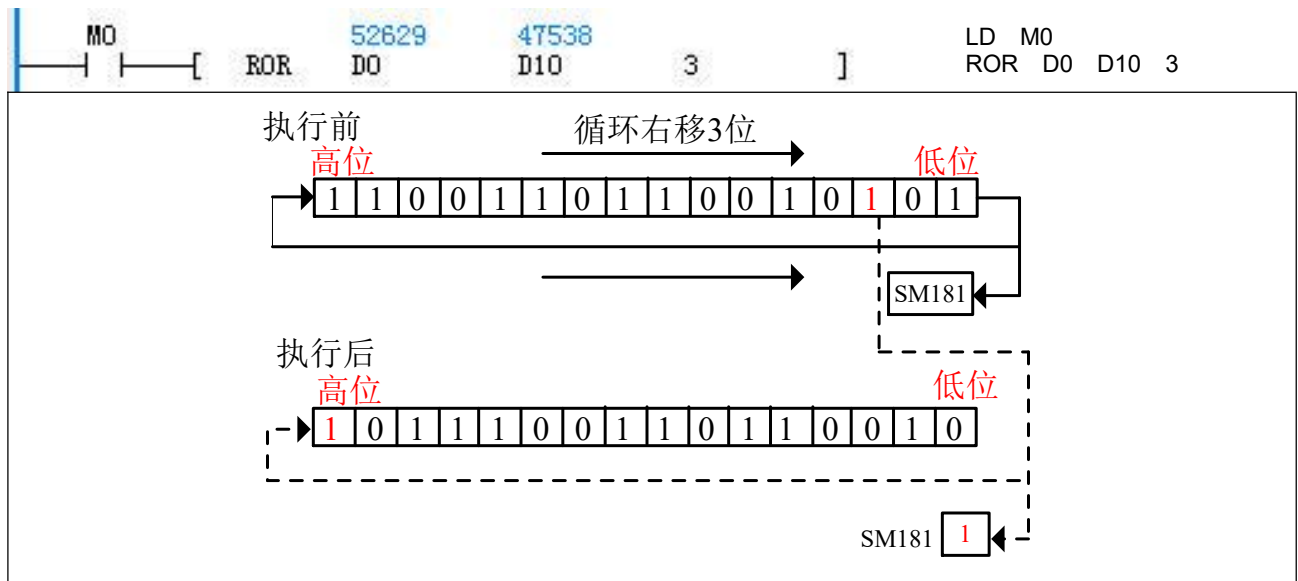
功能说明

当能流有效时，**S1** 的数据循环右移 **S2** 位后的结果赋予 **D**。同时，位移最终位被存入进位标志位（SM181）。

注意事项

S2 范围大于等于 0；当 **S1** 为 Kn 寻址时，Kn 必须等于 4。

使用示例



当 M0=ON 时，D0=2#1100110110010101 (52629) 循环右移 3 位，结果赋给 D10，位移最终位被存入进位标志位，D10=2#1011100110110010 (47538)，SM181=ON。

6.7.2 ROL: 16 位循环左移指令

梯形图: 										适用机型	APLC-TCP				
指令列表: ROL (S1) (D) (S2)										影响标志位	进位标志 SM181				
										步长	7				
操作数	类型	适用软元件													变址
S1	WORD	常数	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	√
D	WORD			KnY	KnM	KnS	KnLM		D		C	T	V	Z	√
S2	INT	常数	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	√

操作数说明

S1: 源操作数 1

D: 目的操作数

S2: 源操作数 2

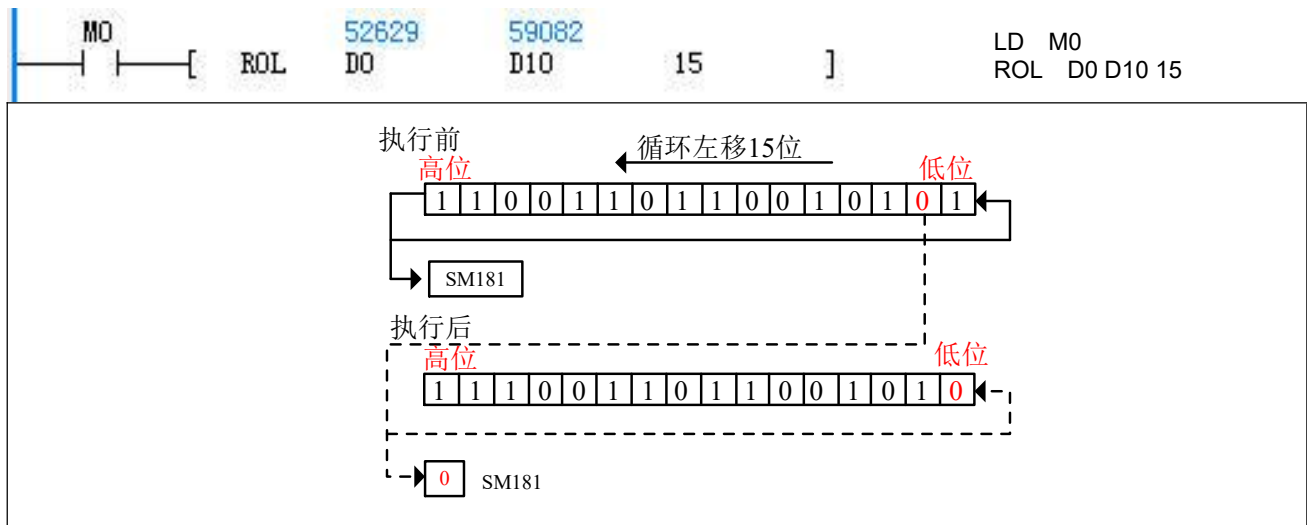
功能说明

当能流有效时，**S1** 的数据循环左移 **S2** 位后的结果赋予 **D**。同时，位移最终位被存入进位标志位（SM181）。

注意事项

S2 范围大于等于 0；当 **S1** 为 Kn 寻址时，Kn 必须等于 4。

使用示例



当 M0=ON 时，D0=2#1100110110010101 (52629) 循环左移 15 位，结果赋给 D10，位移最终位被存入进位标志位，D10=2#110011011001010 (59082)，SM181=OFF。

6.7.3 RCR: 16 位带进位循环右移指令

梯形图: --- ---[RCR (S1) (D) (S2)]										适用机型	APLC-TCP				
指令列表: RCR (S1) (D) (S2)										影响标志位	进位标志 SM181				
										步长	7				
操作数	类型	适用软元件													变址
S1	WORD	常数	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	√
D	WORD			KnY	KnM	KnS	KnLM		D		C	T	V	Z	√
S2	INT	常数	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	√

操作数说明

S1: 源操作数 1

D: 目的操作数

S2: 源操作数 2

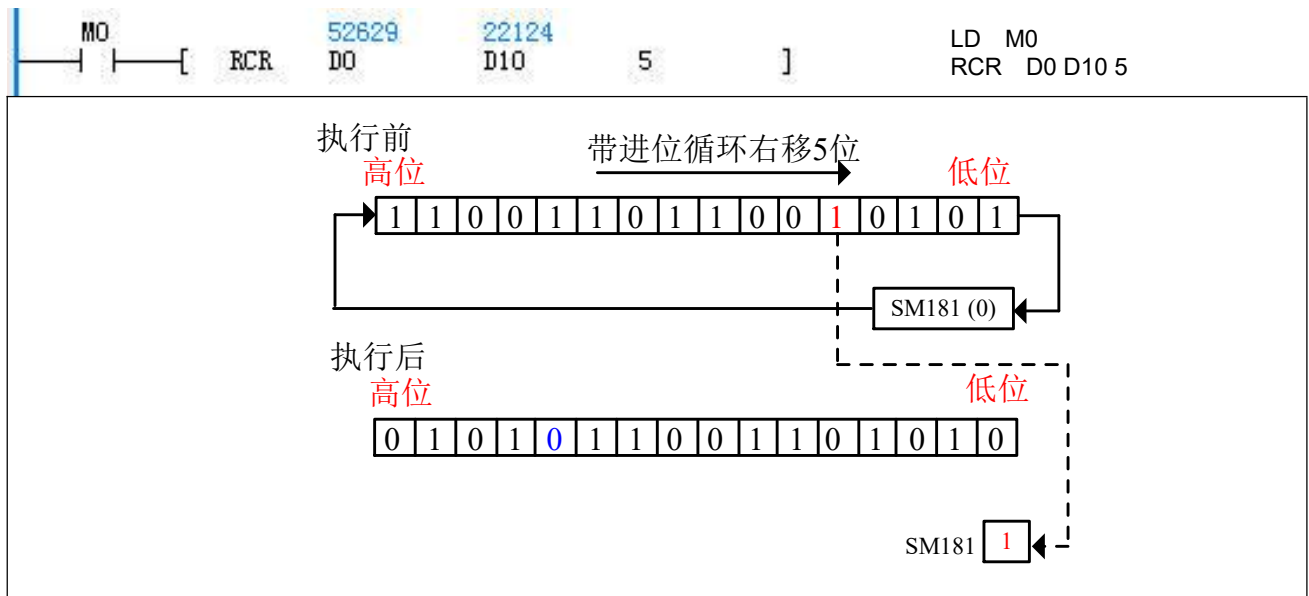
功能说明

当能流有效时, **S1** 的数据带进位 (SM181) 一起循环右移 **S2** 位后的结果赋予 **D**。

注意事项

S2 范围大于等于 0; 当 **S1** 为 Kn 寻址时, Kn 必须等于 4。

使用示例



当 M0=ON 时, D0=2#1100110110010101 (52629) 带进位 (SM181=OFF) 循环右移 5 位, 结果赋给 D10, D10=2#0101011001101100 (22124), SM181=ON。

6.7.4 RCL: 16 位带进位循环左移指令

梯形图: --- ---[RCL (S1) (D) (S2)]										适用机型	APLC-TCP				
										影响标志位	进位标志 SM181				
指令列表: RCL (S1) (D) (S2)										步长	7				
操作数	类型	适用软元件													变址
S1	WORD	常数	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	√
D	WORD			KnY	KnM	KnS	KnLM		D		C	T	V	Z	√
S2	INT	常数	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	√

操作数说明

S1: 源操作数 1

D: 目的操作数

S2: 源操作数 2

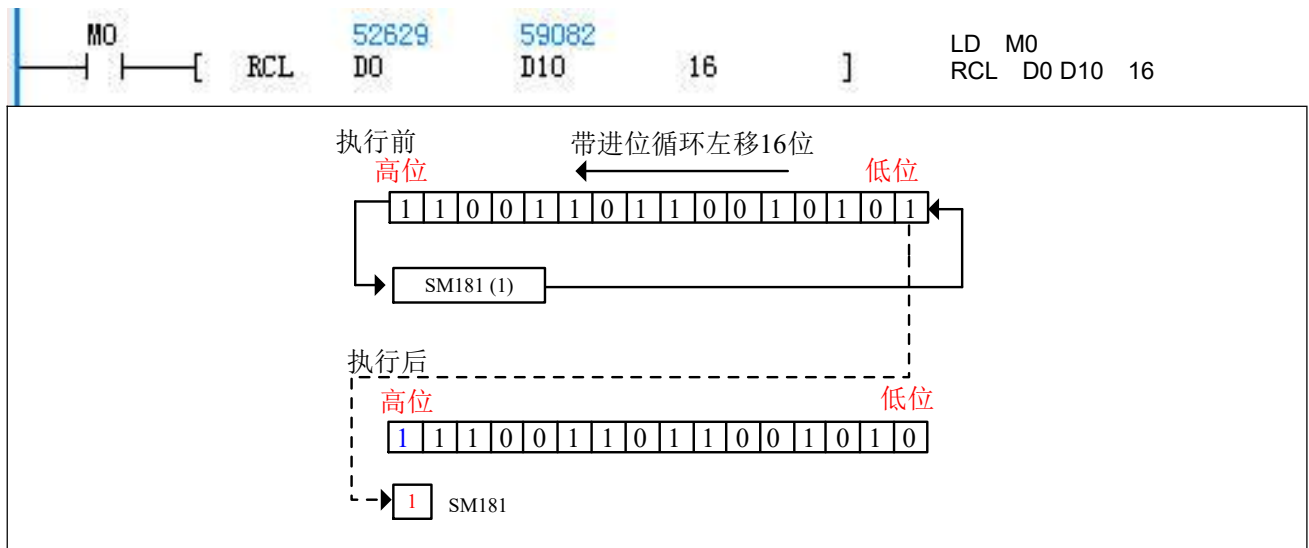
功能说明

当能流有效时, **S1** 的数据带进位 (SM181) 一起循环左移 **S2** 位后的结果赋予 **D**。

注意事项

S2 范围大于等于 0; 当 **S1** 为 Kn 寻址时, Kn 必须等于 4。

使用示例



当 M0=ON 时, D0=2#1100110110010101 (52629) 带进位 (SM181=ON) 循环左移 16 位, 结果赋给 D10, D10=2#1110011011001010 (59082), SM181=ON。

6.7.5 DROR: 32 位循环右移指令

梯形图： 									适用机型		APLC-TCP						
									影响标志位		进位标志 SM181						
指令列表： DROR (S1) (D) (S2)									步长		9						
操作数		类型		适用软元件												变址	
S1		DWORD		常数	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C		V		√
D		DWORD				KnY	KnM	KnS	KnLM		D		C		V		√
S2		INT		常数	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	√

操作数说明

S1: 源操作数 1

D: 目的操作数

S2: 源操作数 2

功能说明

当能流有效时，**S1** 的数据循环右移 **S2** 位后的结果赋予 **D**。同时，位移最终位被存入进位标志位（SM181）。

注意事项

S2 范围大于等于 0；当 **S1** 为 Kn 寻址时，Kn 必须等于 8。

使用示例



M0=ON 时，D0 (D1) =2#10110011100110001001110010101100 (3013123244) 循环右移 7 位，结果赋给 (D10, D11)，位移最终位被存入进位标志位，(D10, D11) =2#01011001011001110011000100111001 (1499935033)，SM181=OFF。

请参考 ROR 指令图例

6.7.6 DROL: 32 位循环左移指令

梯形图： 									适用机型		APLC-TCP					
指令列表： DROL (S1) (D) (S2)									影响标志位		进位标志 SM181					
步长									9							
操作数		类型	适用软元件													变址
S1		DWORD	常数	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C		V		√
D		DWORD			KnY	KnM	KnS	KnLM		D		C		V		√
S2		INT	常数	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	√

操作数说明

S1: 源操作数 1

D: 目的操作数

S2: 源操作数 2

功能说明

当能流有效时，**S1** 的数据循环左移 **S2** 位后的结果赋予 **D**。同时，位移最终位被存入进位标志位（SM181）。

注意事项

S2 范围大于等于 0；当 **S1** 为 Kn 寻址时，Kn 必须等于 8。

使用示例



当 M0=ON 时，(D0,D1)=2#10110011100110001001110010101100 (3013123244) 循环右移 30 位，结果赋给 (D10,D11)，位移最终位被存入进位标志位，(D10,D11)=2#00101100111001100010011100101011 (753280811)，SM181=ON。

请参考 ROL 指令图例

6.7.7 DRCR：32 位带进位循环右移指令

梯形图： 										适用机型	APLC-TCP					
										影响标志位	进位标志 SM181					
指令列表：DRCR (S1) (D) (S2)										步长	9					
操作数	类型	适用软元件												变址		
S1	DWORD	常数	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C		V		√	
D	DWORD			KnY	KnM	KnS	KnLM		D		C		V		√	
S2	INT	常数	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	√	

操作数说明

S1：源操作数 1

D：目的操作数

S2：源操作数 2

功能说明

当能流有效时，**S1** 的数据带进位（SM181）一起循环右移 **S2** 位后的结果赋予 **D**。

注意事项

S2 范围大于等于 0；当 **S1** 为 Kn 寻址时，Kn 必须等于 8。

使用示例



- 当 M0=ON 时，(D0,D1)=2#10110011100110001001110010101100 (3013123244) 带进位（SM181=OFF）循环右移 11 位，结果赋给 (D10,D11)，(D10,D11)=2#00101011000101100111001100010011 (722891539)，SM181=ON。
- 请参考 RCR 指令图例

6.7.8 DRCL：32 位带进位循环左移指令

梯形图： 									适用机型		APLC-TCP						
									影响标志位		进位标志 SM181						
指令列表： DRCL (S1) (D) (S2)									步长		9						
操作数		类型		适用软元件												变址	
S1		DWORD		常数	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C		V		√
D		DWORD				KnY	KnM	KnS	KnLM		D		C		V		√
S2		INT		常数	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	√

操作数说明

S1：源操作数 1

D：目的操作数

S2：源操作数 2

功能说明

当能流有效时，**S1** 的数据带进位（SM181）一起循环左移 **S2** 位后的结果赋予 **D**。

注意事项

S2 范围大于等于 0；当 **S1** 为 Kn 寻址时，Kn 必须等于 8。

使用示例



1M0=ON 时，(D0,D1)=2#10110011100110001001110010101100 (3013123244) 带进位 (SM181=OFF) 循环左移 25 位，结果赋给 (D10,D11)，(D10,D11)=2#001011000101100111001100010011100 (1488165020)，SM181=ON。

2. 请参考 RCL 指令图例

6.7.9 SHR：16 位右移指令

梯形图： 									适用机型		APLC-TCP					
指令列表：SHR (S1) (D) (S2)									影响标志位							
步长									7							
操作数	类型	适用软元件													变址	
S1	WORD	常数	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	√	
D	WORD			KnY	KnM	KnS	KnLM		D		C	T	V	Z	√	
S2	INT	常数	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	√	

操作数说明

S1：源操作数 1

D：目的操作数

S2：源操作数 2

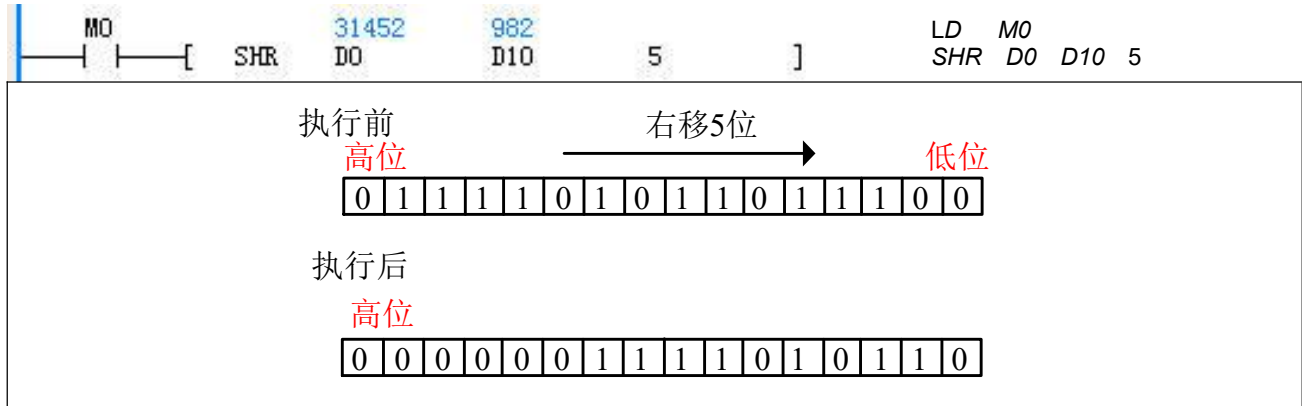
功能说明

当能流有效时，**S1** 的数据右移 **S2** 位后的结果赋予 **D**。

注意事项

S2 范围大于等于 0；当 **S1** 为 Kn 寻址时，Kn 必须等于 4。

使用示例



当 **M0=ON** 时，**D0=2#0111101011011100** (31452) 右移 5 位，结果赋给 **D10**，**D10=2#0000001111010110** (982)。

6.7.10 SHL：16 位左移指令

梯形图：										适用机型	APLC-TCP				
— — [SHL (S1) (D) (S2)]										影响标志位					
指令列表：SHL (S1) (D) (S2)										步长	7				
操作数	类型	适用软元件													变址
S1	WORD	常数	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	√
D	WORD			KnY	KnM	KnS	KnLM		D		C	T	V	Z	√
S2	INT	常数	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	√

操作数说明

S1：源操作数 1

D：目的操作数

S2：源操作数 2

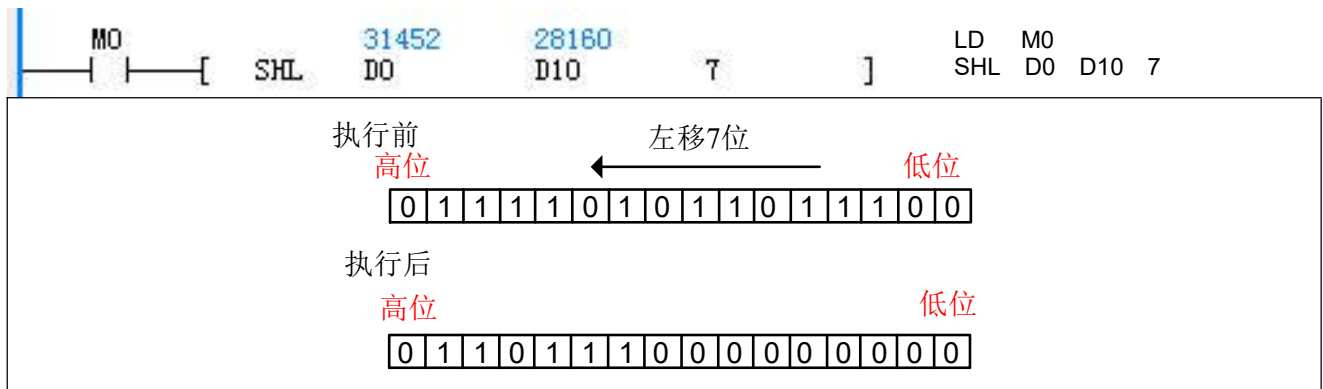
功能说明

当能流有效时，**S1** 的数据左移 **S2** 位后的结果赋予 **D**。

注意事项

S2 范围大于等于 0；当 **S1** 为 Kn 寻址时，Kn 必须等于 4。

使用示例



当 M0=ON 时，D0=2#0111101011011100（31452）左移 7 位，结果赋给 D10，D10=2#0110111000000000（28160）。

6.7.11 DSHR：32 位右移指令

梯形图：										适用机型	APLC-TCP				
[DSHR (S1) (D) (S2)]										影响标志位					
指令列表： DSHR (S1) (D) (S2)										步长	9				
操作数	类型	适用软元件													变址
S1	DWORD	常数	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C		V		√
D	DWORD			KnY	KnM	KnS	KnLM		D		C		V		√
S2	INT	常数	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	√

操作数说明

S1：源操作数 1

D：目的操作数

S2：源操作数 2

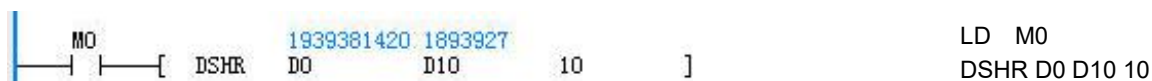
功能说明

当能流有效时，**S1** 的数据右移 **S2** 位后的结果赋予 **D**。

注意事项

S2 范围大于等于 0；当 **S1** 为 Kn 寻址时，Kn 必须等于 8。

使用示例



1. M0=ON 时，(D0,D1)=2#01110011100110001001110010101100（1939381420）右移 10 位，结果赋给 (D10,D11)，(D10,D11)=2#00000000000011100111001100010011（1893927）。

2. 请参考 SHR 指令图例

6.7.12 DSHL：32 位左移指令

梯形图： 										适用机型	APLC-TCP				
指令列表：DSHL (S1) (D) (S2)										影响标志位					
										步长	9				
操作数	类型	适用软元件												变址	
S1	DWORD	常数	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C		V		√
D	DWORD			KnY	KnM	KnS	KnLM		D		C		V		√
S2	INT	常数	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	√

操作数说明

S1：源操作数 1

D：目的操作数

S2：源操作数 2

功能说明

当能流有效时，**S1** 的数据左移 **S2** 位后的结果赋予 **D**。

注意事项

S2 范围大于等于 0；当 **S1** 为 Kn 寻址时，Kn 必须等于 8。

使用示例



1. M0=ON 时，(D0,D1) =2#01110011100110001001110010101100 (1939381420) 左移 15 位，结果赋给 (D10,D11)，(D10,D11) =2#01001110010101100000000000000000 (1314258944)。

2. 请参考 SHL 指令图例

6.7.13 SFTR：位串右移指令

梯形图： 										适用机型	APLC-TCP				
指令列表：SFTR (S1) (D) (S2) (S3)										影响标志位					
										步长	9				
操作数	类型	适用软元件												变址	
S1	BOOL		X	Y	M	S	LM	SM			C	T			√
D	BOOL			Y	M	S	LM				C	T			√
S2	INT	常数	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	√
S3	INT	常数	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	√

操作数说明

S1：源操作数 1

D：目的操作数

S2：源操作数 2

S3: 源操作数 3

功能说明

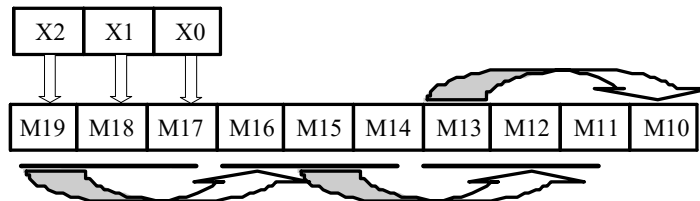
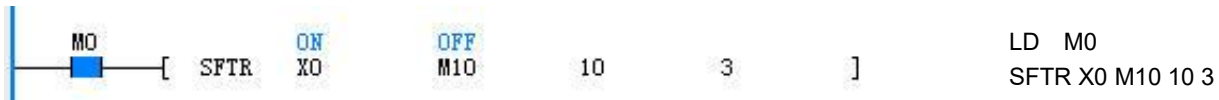
当能流有效时，将 **D** 单元开始的 **S2** 个单元的内容右移 **S3** 个单元，最右端 **S3** 个数据将被丢弃。同时，以 **S1** 单元开始的 **S3** 个单元的内容将被移入字串的左端。

注意事项

左右顺序，以大元件编号元件为左，小元件编号为右。

S2 范围大于等于零，**S3** 范围大于等于零。

使用示例



1. M0=ON 时，将 M10 单元开始的 10 个单元的内容以位为单位右移 3 个单元，最右端 M10~M12 将被丢弃。同时，X0 单元开始的 3 个单元的内容被移入位串的左端。
2. 执行前：X0=1, X1=0, X2=1。M10=0, M11=1, M12=1, M13=0, M14=0, M15=1, M16=0, M17=0, M18=0, M19=1。
3. 执行后：X0~X2 内容不变。M10=0, M11=0, M12=1, M13=0, M14=0, m15=0, m16=1, m17=1, m18=0, m19=1。

6.7.14 SFTL：位串左移指令

梯形图：									适用机型		APLC-TCP					
									影响标志位							
指令列表： SFTL (S1) (D) (S2) (S3)									步长		9					
操作数		类型	适用软元件												变址	
S1		BOOL		X	Y	M	S	LM	SM			C	T		√	
D		BOOL			Y	M	S	LM				C	T		√	
S2		INT	常数	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	√
S3		INT	常数	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	√

操作数说明

S1: 源操作数 1

D: 目的操作数

S2: 源操作数 2

S3: 源操作数 3

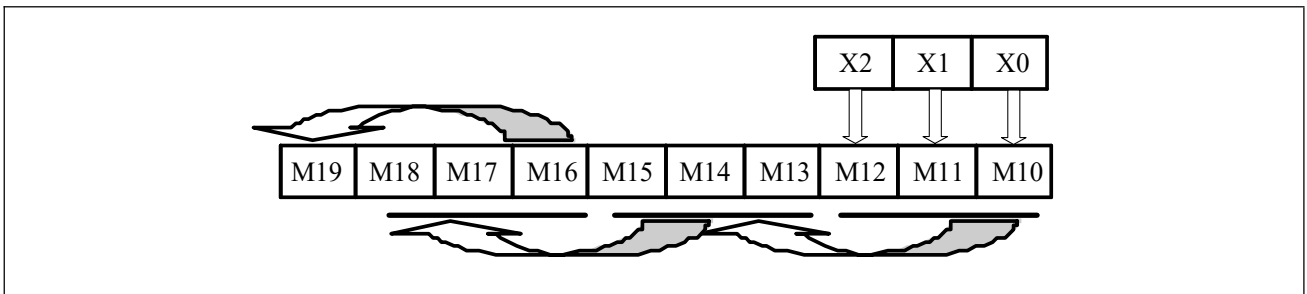
注意事项

1. 左右顺序，以大元件编号元件为左，小元件编号为右。
2. **S2** 范围大于等于零，**S3** 范围大于等于零。

功能说明

当能流有效时，将 **D** 单元开始的 **S2** 个单元的内容左移 **S3** 个单元，最左端 **S3** 个数据将被丢弃，同时，以 **S1** 单元开始的 **S3** 个单元的内容将被移入字串的右端。

使用示例



1. $M0=ON$ 时，将 $M10$ 单元开始的 10 个单元的内容以位为单位左移 3 个单元，最左端 $M17\sim M19$ 将被丢弃。同时， $X0$ 单元开始的 3 个单元的内容被移入位串的右端。
2. 执行前： $X0=1$ ， $X1=0$ ， $X2=1$ 。 $M10=0$ ， $M11=1$ ， $M12=1$ ， $M13=0$ ， $M14=0$ ， $M15=1$ ， $M16=0$ ， $M17=0$ ， $M18=0$ ， $M19=1$ 。
3. 执行后： $X0\sim X2$ 内容不变。 $M10=1$ ， $M11=0$ ， $M12=1$ ， $M13=0$ ， $M14=1$ ， $M15=1$ ， $M16=0$ ， $M17=0$ ， $M18=1$ ， $M19=0$ 。

6.8 外设指令

6.8.1 FROM：特殊模块缓冲寄存器字读指令

无

6.8.2 DFROM：特殊模块缓冲寄存器双字读指令

无

6.8.3 TO：特殊模块缓冲寄存器字写指令

无

6.8.4 DTO：特殊模块缓冲寄存器双字写指令

无

6.8.5 VRRD：读模拟电位器值指令

梯形图： 										适用机型	APLC-TCP				
										影响标志位					
指令列表：VRRD (S) (D)										步长	5				
操作数	类型	适用软元件													变址
S	WORD	常数													
D	WORD								D				V		√

操作数说明

S：指定的模拟电位器号

设定范围：0~255，否则报操作数错误。

D：模拟电位器读数值的保存（元件），数值大小为 0~255。

功能说明

读取指定的模拟电位器值，并存放到指定的元件中。

使用示例



M0 为 ON 时读入系统中 0 号模拟电位器的值，把读入的数值放到 D10 中。

6.8.6 REFF：设置输入滤波常数指令

梯形图： 										适用机型	APLC-TCP				
指令列表： REFF (S)										影响标志位					
										步长	3				
操作数	类型	适用软元件													变址
S	WORD	常数	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	√

操作数说明

S：输入滤波常数

- AT100：实际有效的数据为 0，8，16，32，64。当参数小于 8 按 0 处理，当参数小于 16 按 8 处理，当参数小于 32 按 16 处理，当参数小于 64 按 32 处理，如为其它数据按 64 处理。

功能说明

设置 X0~X17 的输入滤波常数。

注意事项

输入滤波常数仅对用做普通输入的端口有效，对用做高速输入的端口无效。

使用示例



X10 为 ON 时更改输入的滤波常数时间为 30ms。

6.8.7 REF：I/O 立即刷新指令

梯形图： 										适用机型	APLC-TCP				
指令列表： REF (D) (S)										影响标志位					
										步长	5				
操作数	类型	适用软元件													变址
D	BOOL		X	Y											
S	INT	常数													

操作数说明

D：要刷新的起始 X/Y 软元件

指定起始软元件号为 8 的整数倍。如 X0，X10，X20...或 Y0，Y10，Y20...，最低位为 0。

S：要刷新的端口数量

刷新点数应为 8，16，.....，256（8 的倍数，除此以外的数值是错误的）

功能说明

通常，PLC 的输入和输出都是在用户程序结束后的执行。在运算过程中，如果需要读取最新的输入状态或希望立即更新输出状态时，可以使用该指令。

注意事项

1. 对输入的端口（Xn，Yn）下标数应为 8 的整数倍。
2. 刷新的（端口）数量也应该为 8 的整数倍。
3. 在 FOR-NEXT 指令之间或 CJ 指令之间，一般使用 REF 立即处理。
4. 在有输入输出动作的中断处理被执行中，在中断子程序中进行输入输出刷新，获取最新的输入信息并且及时输出运算结果，使用 REF 指令。
5. 对继电器型的输出点要考虑到输出点的反应时间。

使用示例

当 M0 为 ON 时，Y0 到 Y7 的状态马上输出，不受扫描周期的影响。

6.8.8 EROMWR：EEPROM 写指令

梯形图：										适用机型	APLC-TCP				
— — [EROMWR (S1) (S2)]										影响标志位					
指令列表： EROMWR (S1) (S2)										步长	6				
操作数	类型	适用软元件													变址
S1	WORD								D						
S2	INT	常数	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	√

操作数说明

S1：写入原件的起始地址（只能为 D6000~D6999）

S2：写入元件的个数（ $S2 < 16$ ， $S1 + S2 < D7000$ ）

注意事项

一个写 EEPROM 指令操作会使扫描周期增加 2~5ms，建议 EROMWR 指令操作数 1 起始地址为 6000 加 16 的倍数，如 D6000，D6016，D6032 等。

功能说明

1. 通常，PLC 的一部分数据可以通过掉电保持功能保存。在运算过程中，如果需要将中间数据保存至 EEPROM 时，可以使用该指令。
2. 上升沿时指令执行。
3. 不能同时执行两条该指令，当 SM196（EEPROM 写空闲标志）为 ON 时表示 EEPROM 写操作空闲，为 OFF 时表示有 EROMWR 操作在执行。

```

graph TD
    M1[M1] --- J1(( ))
    J1 --- SET_M1000[SET M1000]
    J1 --- J2(( ))
    J2 --- RST_M1[RST M1]
    J2 --- J3(( ))
    J3 --- MOV_16[MOV 16 D6016]
    J3 --- J4(( ))
    J4 --- MOV_32[MOV 32 D6032]
    
    SM1[SM1] --- J5(( ))
    J5 --- SET_M1[SET M1]
    J5 --- J6(( ))
    J6 --- LD_M1000[LD M1000]
    J6 --- AND_SM196[AND SM196]
    J6 --- EROMWR_D6016_2[EROMWR D6016 2]
    
    M1001[M1001] --- J7(( ))
    J7 --- LD_M1001[LD M1001]
    J7 --- AND_SM196_2[AND SM196]
    J7 --- EROMWR_D6032_16[EROMWR D6032 16]
    J7 --- J8(( ))
    J8 --- RST_M1001[RST M1001]
  
```

通过 SM1 和 M1 使 M1000 在第二个扫描周期产生一个上升沿，保证了第一条 EROMWR 指令的执行；M1001 和 SM196 使第二条 EROMWR 指令产生上升沿，指令执行。

无

6.9 实时时钟指令

6.9.1 TRD：实时时钟读指令

梯形图： 								适用机型		APLC-TCP					
								影响标志位							
指令列表： TRD (D)								步长		3					
操作数	类型	适用软元件										变址			
D	WORD								D				V		√

操作数说明

D：读出系统时间所存放的起始单元，占有 **D** 所指定单元起始的 7 个连续单元

功能说明

读出系统中的时间，保存在 **D** 所指定的存储单元中。

注意事项

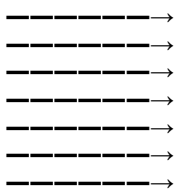
在系统出现时钟设置出错时，TRD 读时间不成功。

使用示例



M0 为 ON 时把系统的时间分别送到 D10 开始的 7 个单元中。

指令的执行结果如下：

实时时钟用的特殊数据寄存器	元件	项目	时钟数据		元件	项目
	SD100	年	2000~2099		D10	年
	SD101	月	1~12		D11	月
	SD102	日	1~31		D12	日
	SD103	时	0~23		D13	时
	SD104	分	0~59		D14	分
	SD105	秒	0~59		D15	秒
	SD106	星期	0~6		D16	星期

6.9.2 TWR：实时时钟写指令

梯形图： 										适用机型		APLC-TCP				
										影响标志位						
指令列表： TWR (S)										步长		3				
操作数		类型		适用软元件										变址		
S		WORD								D				V		√

操作数说明

S： 写入系统时间的软元件

时钟设定用数据	元件	项目	时钟数据		元件	项目
	D10	年	2000~2099		SD100	年
	D11	月	1~12		SD101	月
	D12	日	1~31		SD102	日
	D13	时	0~23		SD103	时
	D14	分	0~59		SD104	分
	D15	秒	0~59		SD105	秒
	D16	星期	0~6		SD106	星期

功能说明

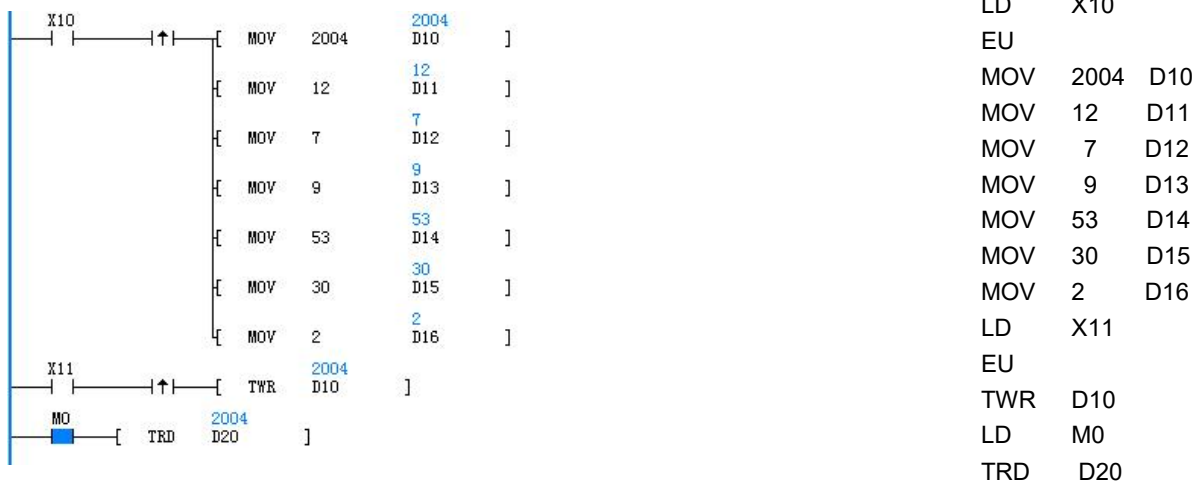
当系统时间跟实际时间不同时可以使用 TWR 指令更改系统的时间。

注意事项

1. 写入的时间数据必须要满足公历的要求否则指令执行不成功。
2. 建议使用边沿触发作为该指令执行条件。

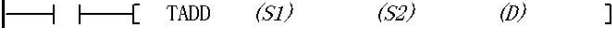
使用示例

通过 TWR 改变系统的时间，见下图：



1. 测到 X10 的上升沿给 D10 的连续 7 个单元写入时间设定值。
2. 检测到 X11 的上升沿把 D10 的连续 7 个单元的数值写到系统时间中。
3. 在 M0 为 ON 时读取系统时间存到 D20 中。

6.9.3 TADD：时钟加指令

梯形图： 										适用机型	APLC-TCP				
指令列表： TADD (S1) (S2) (D)										影响标志位	零标志 SM180 进位标志 SM181				
										步长	7				
操作数	类型	适用软元件										变址			
S1	WORD									D	SD			V	√
S2	WORD									D	SD			V	√
D	WORD									D				V	√

操作数说明

S1：时钟数据 1

在 **S1** 所指的 3 个储存单元内保存时间数据，对不满足时间格式的数据，系统提供指令操作数数值非法错误。

S2：时钟数据 2

在 **S2** 所指的 3 个储存单元内保存另一时间数据，对不满足时间格式的数据，系统提示指令操作数数值非法错误。

D：时间结果存储单元

按时间加处理完成的数据存储在 **D** 所指的 3 个储存单元内。根据处理完成的结果会影响进位标志 SM181，零标志 SM180。

功能说明

对时间格式的数据进行加法运算，运算规则按时间格式执行。

注意事项

参与运算的时间数据应符合时间格式：

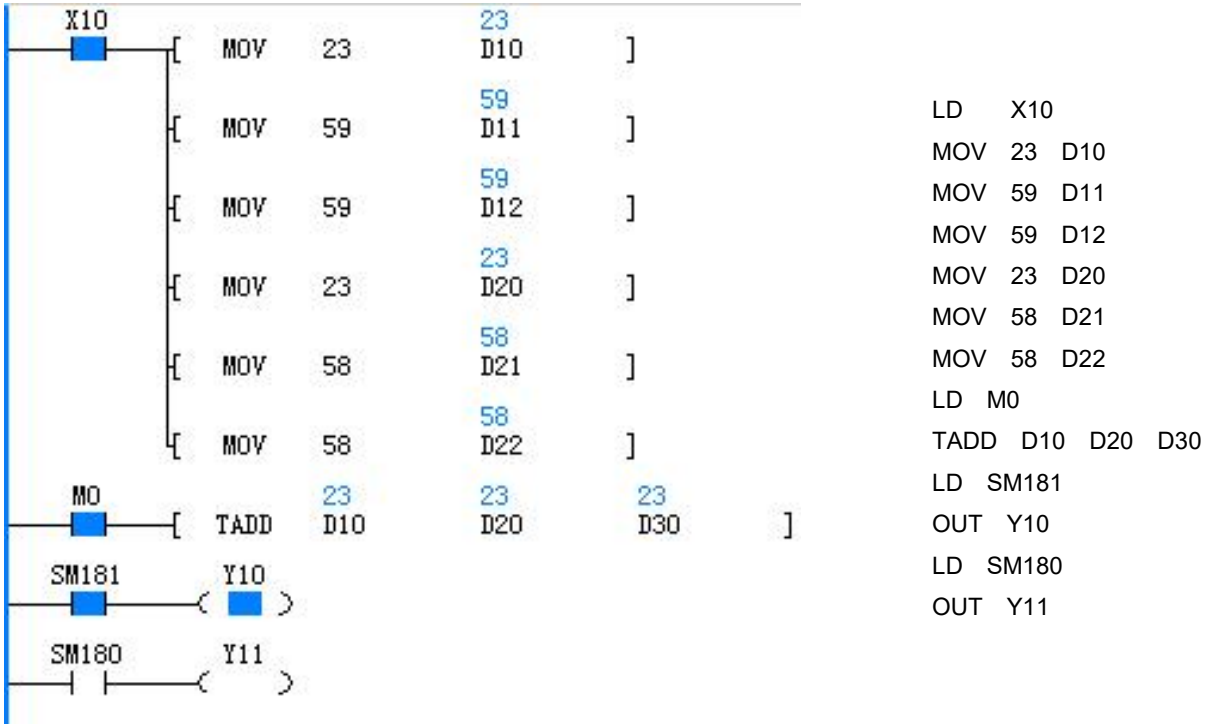
“时”的设定范围：0~23

“分”的设定范围：0~59

“秒”的设定范围：0~59

使用示例

S1		+	S2		=	D	
D10	23 时		D20	23 时		D30	23 时
D11	59 分		D21	58 分		D31	58 分
D12	59 秒		D22	58 秒		D32	57 秒



1. X10 为 ON 时送时间数据到 D10 开始的 3 点和 D20 开始的 3 个储存单元。
2. M0 为 ON 时把 D10 开始的 3 个储存单元加 D20 开始的 3 个储存单元，处理完成的结果保存在 D30 开始的 3 个储存单元。
3. 进位标志 (SM181) 置 ON，零标志 (SM180) 为 OFF。

6.9.4 TSUB：时钟减指令

梯形图： <div><div><div></div><div></div></div><div><div></div><div></div></div><div>[TSUB (S1) (S2) (D)]</div></div>									适用机型		APLC-TCP					
									影响标志位		零标志 SM180 借位标志 SM182					
指令列表： TADD (S1) (S2) (D)									步长		7					
操作数		类型	适用软元件										变址			
S1		WORD								D	SD			V		√
S2		WORD								D	SD			V		√
D		WORD								D				V		√

操作数说明

S1：时钟数据 1

在 S1 所指的 3 个储存单元内保存时间数据，对不满足时间格式的数据，系统提示指令操作数数值非法错误。

S2：时钟数据 2

在 S2 所指的 3 个储存单元内保存另一时间数据，对不满足时间格式的数据，系统提示指令操作数数值非法错误。

D：时间结果存储单元

按时间加处理完成的数据存储在 D 所指的 3 个储存单元内。根据处理完成的结果会影响借位标志 SM182，零标志 SM180。

功能说明

对时间格式的数据进行减法运算，运算规则按时间格式执行。

注意事项

参与运算的时间数据应符合时间格式：

“时”的设定范围：0~23

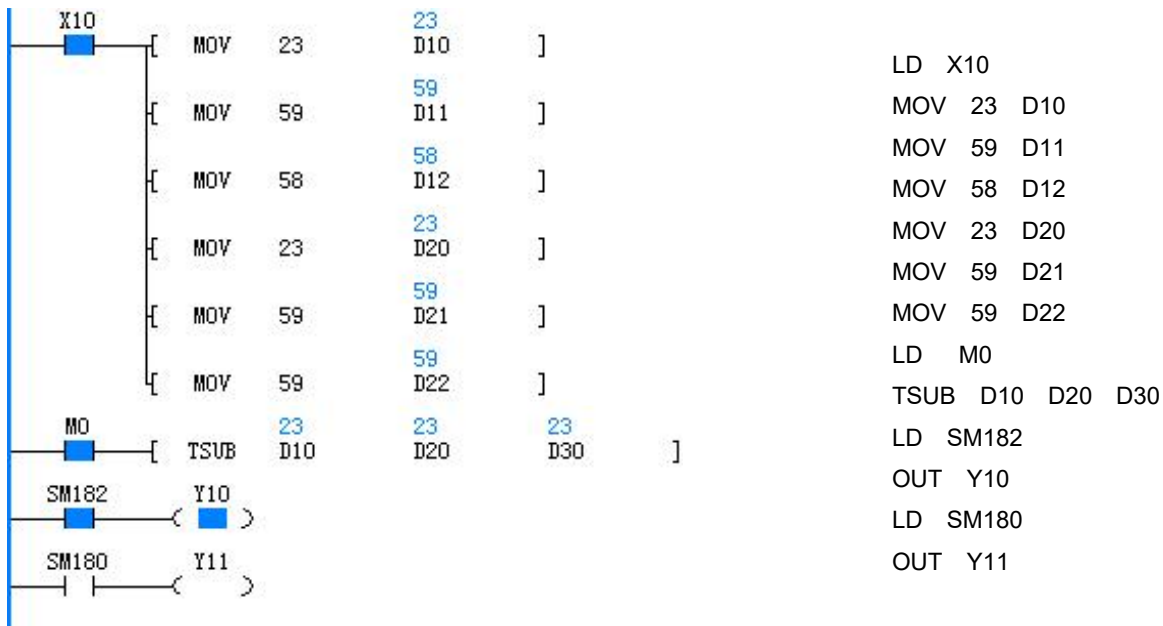
“分”的设定范围：0~59

“秒”的设定范围：0~59

使用示例

S1		-	S2		=	D	
D10	23 时		D20	23 时		D30	23 时
D11	59 分		D21	59 分		D31	59 分
D12	58 秒		D22	59 秒		D32	59 秒

示例说明



1. X10 为 ON 时送时间数据到 D10 开始的 3 点和 D20 开始的 3 个储存单元。
2. M0 为 ON 时把 D10 开始的 3 个储存单元减 D20 开始的 3 个储存单元，处理完成的结果保存在 D30 开始的 3 个储存单元。
3. 借位标志置（SM182）ON，零标志（SM180）为 OFF。

6.9.5 HOUR：计时表指令

梯形图： 									适用机型	APLC-TCP					
指令列表： HOUR (S) (D1) (D2)									影响标志位						
									步长	8					
操作数	类型	适用软元件													变址
S	INT	常数	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	√
D1	INT								D				V		√
D2	BOOL			Y	M	S	LM								

操作数说明

S：小时比较数据。数据范围 0~32767

D1：时间存储单元

D1 的数据单元保存小时，**D1+1** 的数据单元保存秒

D2：报警输出地址

当 **D1** 的数据大于等于 **S** 指定的数据时，报警点变为 ON 输出。

功能说明

对输入触点处于 ON 状态的时间以小时为单位进行判断。

注意事项

- 为了能在切断 PLC 的电源后仍然使用当前数据，请指定 **D1** 为停电保持用的软元件单元。若使用普通软元件单元，当 PLC 的电源被切断或进行 RUN → STOP 的操作时，当前数据将被清除。
- 即使报警输出 **D2** 为 ON，仍然能够继续计数。
- 本指令小时为 16 位整数数据。当小时的数据大于 32767 时又从 0 开始。

使用示例



- M0 为 ON 时设定 HOUR 指令的比较数据。
- M1 为 ON 时 HOUR 对输入触点进行时间累加。
- 当 M1 的 ON 状态累积时间大于等于 1000 时，M10 为 ON 状态。

6.9.6 DCMP: (=、<、>、<>、>=、<=) 日期比较指令

梯形图：									适用机型		APLC-TCP					
<div><div><div></div><div></div></div>DCMP=</div> <div><div><div></div><div></div></div>(S1)</div> <div><div><div></div><div></div></div>(S2)</div> <div><div><div></div><div></div></div>(D)</div> <div></div>									影响标志位							
<div><div><div></div><div></div></div>DCMP<</div> <div><div><div></div><div></div></div>(S1)</div> <div><div><div></div><div></div></div>(S2)</div> <div><div><div></div><div></div></div>(D)</div> <div></div>																
<div><div><div></div><div></div></div>DCMP></div> <div><div><div></div><div></div></div>(S1)</div> <div><div><div></div><div></div></div>(S2)</div> <div><div><div></div><div></div></div>(D)</div> <div></div>																
<div><div><div></div><div></div></div>DCMP<></div> <div><div><div></div><div></div></div>(S1)</div> <div><div><div></div><div></div></div>(S2)</div> <div><div><div></div><div></div></div>(D)</div> <div></div>																
<div><div><div></div><div></div></div>DCMP>=</div> <div><div><div></div><div></div></div>(S1)</div> <div><div><div></div><div></div></div>(S2)</div> <div><div><div></div><div></div></div>(D)</div> <div></div>																
<div><div><div></div><div></div></div>DCMP<=</div> <div><div><div></div><div></div></div>(S1)</div> <div><div><div></div><div></div></div>(S2)</div> <div><div><div></div><div></div></div>(D)</div> <div></div>																
指令列表：									步长		7					
DCMP = (S1) (S2) (D)																
DCMP < (S1) (S2) (D)																
DCMP > (S1) (S2) (D)																
DCMP < > (S1) (S2) (D)																
DCMP > = (S1) (S2) (D)																
DCMP < = (S1) (S2) (D)																
操作数		类型		适用软元件										变址		
S1		INT								D	SD			V		√
S2		INT								D	SD			V		√
D		BOOL				Y	M	S	LM				C	T		

操作数说明

S1: 日期比较数据 1, 占用 **S1** 指定单元起始 3 个字单元, 3 个单元的数据必须符合公历格式, 否则系统报操作数错误。

S2: 日期比较数据 2, 占用 **S2** 指定单元起始 3 个字单元, 3 个单元的数据必须符合公历格式, 否则系统报操作数错误。

D: 比较状态输出, 数据符合比较条件, D 置为 ON, 否则为 OFF。

功能说明

对分别以 **S1**、**S2** 为起始单元的日期数据进行 BIN 比较, 比较的结果赋予 **D**。

注意事项

以 **S1**、**S2** 为起始单元的日期数据必须符合公历纪年法, 否则将报操作数错误 (如: 2004, 9, 31 和 2003, 2, 29 等数据都不合法)。

使用示例

	LD	SM0	
	MOV	2004	D0
	MOV	10	D1
	MOV	25	D2
	MOV	2004	D10
	MOV	10	D11
	MOV	24	D12
	LD	X0	
	DCMP=	D0 D10	M0
	DCMP<	D0 D10	M1
	DCMP>	D0 D10	M2
	DCMP<>	D0 D10	M3
	DCMP>=	D0 D10	M4
	DCMP<=	D0 D10	M5

对分别以 D0、D10 为起始单元的日期数据进行 BIN 比较, 比较的结果赋予目的数据 (M0 等)。

6.9.7 TCMP: (=、<、>、<>、>=、<=) 时间比较指令

梯形图:										适用机型		APLC-TCP				
<div><div><div>┌───┐┌───┐</div><div>TCMP=</div><div>(S1) (S2) (D)</div></div><div><div>┌───┐┌───┐</div><div>TCMP<</div><div>(S1) (S2) (D)</div></div><div><div>┌───┐┌───┐</div><div>TCMP></div><div>(S1) (S2) (D)</div></div><div><div>┌───┐┌───┐</div><div>TCMP<></div><div>(S1) (S2) (D)</div></div><div><div>┌───┐┌───┐</div><div>TCMP>=</div><div>(S1) (S2) (D)</div></div><div><div>┌───┐┌───┐</div><div>TCMP<=</div><div>(S1) (S2) (D)</div></div></div>										影响标志位						
指令列表:										步长		7				
TCMP= (S1) (S2) (D)																
TCMP< (S1) (S2) (D)																
TCMP> (S1) (S2) (D)																
TCMP<> (S1) (S2) (D)																
TCMP>= (S1) (S2) (D)																
TCMP<= (S1) (S2) (D)																
操作数		类型		适用软元件										变址		
S1		INT								D	SD			V		√
S2		INT								D	SD			V		√
D		BOOL				Y	M	S	LM				C	T		

操作数说明

S1: 时间比较数据 1

占用 **S1** 指定单元起始 3 个字单元, 3 个单元的数据必须符合 24 小时制时间格式, 否则系统报操作数错误。

S2: 时间比较数据 2

占用 **S2** 指定单元起始 3 个字单元, 3 个单元的数据必须符合 24 小时制时间格式, 否则系统报操作数错误。

D: 比较状态输出, 数据符合比较条件, D 置为 ON, 否则为 OFF

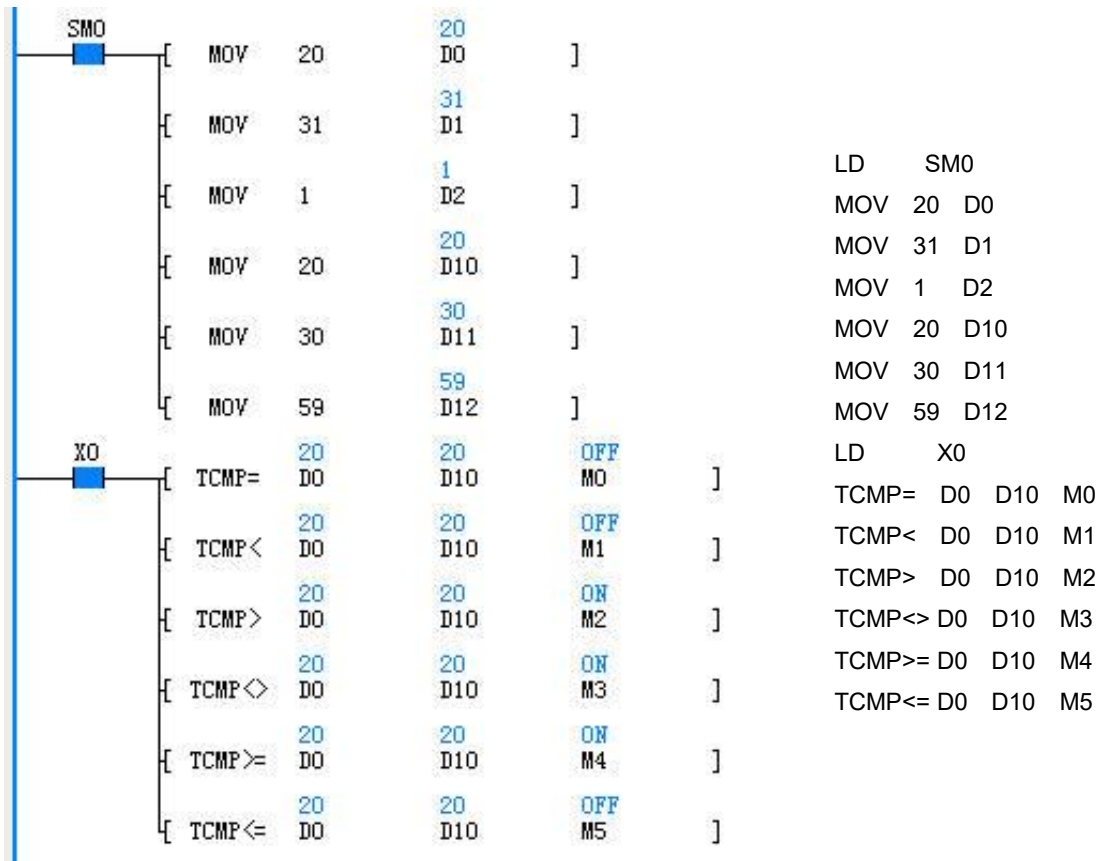
功能说明

对分别以 **S1**、**S2** 为起始单元的时间数据进行 BIN 比较, 比较的结果赋予 **D**。

注意事项

以 **S1**、**S2** 为起始单元的时间数据必须符合 24 小时制, 否则将报操作数错误 (如: 24, 10, 31 和 13, 59, 60 等数据都不合法)

使用示例



对分别以 D0、D10 为起始单元的时间数据进行 BIN 比较，比较的结果赋予目的数据（M0 等）。

6.10 高速 IO 指令

6.10.1 HCNT：高速计数器驱动指令

梯形图： 										适用机型	APLC-TCP				
指令列表：HCNT (D) (S)										影响标志位					
										步长	7				
操作数	类型	适用软元件										变址			
D	DINT										C				
S	DINT	常数	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C		V		√

操作数说明

D：指定计数器号，可设定范围：C236~C255

S：指定比较常数，为 32 位的有符号数据，数据范围

–2147483648~2147483647

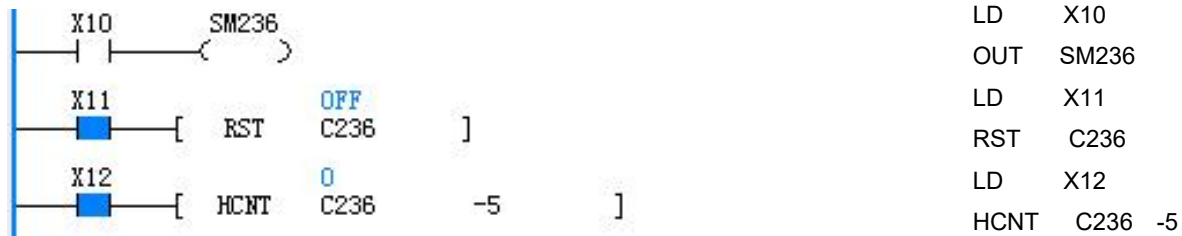
功能说明

驱动指定的硬件高速计数器，所有的高速计数器只有在持续驱动的情况下，才能进行硬件高速计数，同时根据 **S** 对高速计数器的常开触点的动作进行判断。

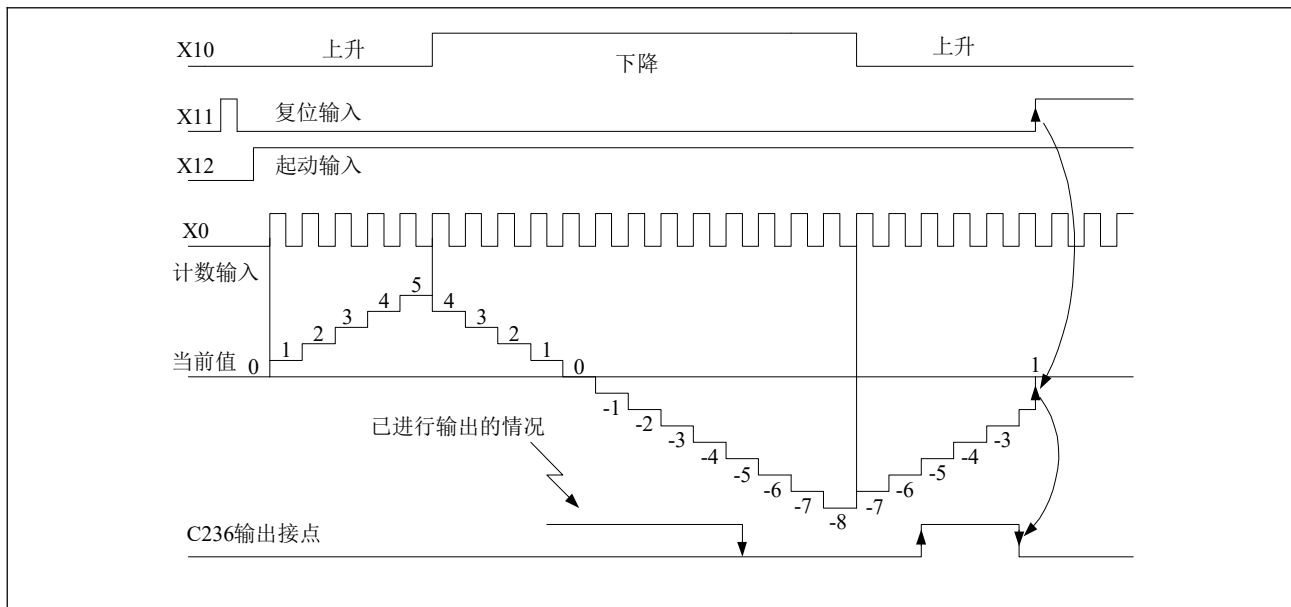
注意事项

HCNT 指令、SPD 指令、外部输入中断以及脉冲捕捉存在硬件冲突，要注意系统所有的高速 IO 的使用条件，使用时请参照高速 IO 指令说明。

使用示例



对程序的操作实例的时序操作如下：



1. X12 由 OFF 变为 ON 时初始化 C236 对应的硬件计数器，X0 为 C236 的脉冲输入端，C236 对 X0 外部脉冲计数。X12 为 OFF 时 X0 为一般的输入点，C236 不能对 X0 的外部脉冲计数。
2. 对触点的动作：当计数器 C236 的当前值由 -6→-5 增加时，C236 的触点被置位。当计数器 C236 的当前值由 -5→-6 减少时，C236 的触点被复位。
3. 当 X11 为 ON 时执行 RST 指令，C236 的数据清除，C236 的触点断开。
4. 对高速计数器的数据和其触点状态在停电状况下由用户在上层软件中的系统块中设置。

6.10.2 DHSCS：高速计数比较置位指令

梯形图： — — [DHSCS (S1) (S2) (D)]										适用机型	APLC-TCP				
指令列表：DHSCS (S1) (S2) (D)										影响标志位					
										步长	10				
操作数	类型	适用软元件										变址			
S1	DINT	常数	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C		V		√
S2	DINT										C				
D	BOOL			Y	M	S									

操作数说明

S1：高速计数器要比较的数据，为 32 位的 DINT 数据，数据范围 -2147483648~2147483647

S2：高速计数器，高速计数器适用范围为 C236~C255

D：输出位元件对象，对 Y，M，S 马上置位输出不受扫描周期的影响

注意事项

1. DHSCS 指令动作必须要和 HCNT 指令配合使用，只有 HCNT 驱动后的高速计数器，DHSCS 才能真正使用。
2. DHSCS 指令在脉冲输入时比较结果动作。因此，即使使用 DMOV 或 MOV 指令等更改高速计数器值 DHSCS 不会有动作。
3. DHSCS (DHSCI, DHSCR, DHSZ, DHSP, DHST) 和普通指令一样可以多次使用，但这些指令同时驱动个数限制在总计 6 条指令以下。对超过 6 条的有效指令不执行，有效的指令按指令的有效先后决定。
4. PLC 高速计数器的最大允许频率。如使用 DHSCS, DHSCI, DHSCR, DHSZ, DHSP, DHST 命令时，将受到很大的影响。具体参考高速 IO 使用的说明。

功能说明

1. 高速计数器只有在 HCNT 驱动指令的条件下根据计数输入 OFF→ON 以中断方式计数，当高速计数器的值等于 DHSCS 指令中的 **S1** 时，**D** 所指定的位元件马上置位，如果为 Y 元件，Y 元件马上输出。
2. 在希望高速计数器当前值的比较置位立即向外部输出比较结果时，可以使用该指令。

使用示例



1. M1 为 ON 时，C236 对 X0 由 OFF→ON 以中断的方式计数 (X0 的输入频率参考高速 IO 的使用说明)，当 C236 由 999→1000 时 C236 触点置位，由 1001→1000 时 C236 触点复位。C236 的触点驱动 Y11 时，Y11 的执行由用户程序的扫描决定。
2. 当 M2 为 ON 时，DHSCS 高速指令在满足注意事项所说的高速指令要求时，当 C236 到 2000 时 Y10 马上输出，不受扫描周期影响。
3. 当 M0 为 ON 时，SM236 驱动，C236 计数器减。当 M0 为 OFF 时，SM236 未驱动，C236 计数器增计数。

6.10.3 DHSCI：高速计数比较中断触发指令

梯形图： 									适用机型		APLC-TCP				
指令列表： DHSCI (S1) (S2) (S3)									影响标志位						
步长									10						
操作数	类型	适用软元件													变址
S1	DINT	常数	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C		V		√
S2	DINT										C				
S3	WORD	常数													

操作数说明

S1：高速计数器要比较的数据，为 32 位的 DINT 数据，数据范围 -2147483648~2147483647

S2：高速计数器，高速计数器适用范围为 C236~C255

S3：中断号。中断号范围：20~25

功能说明

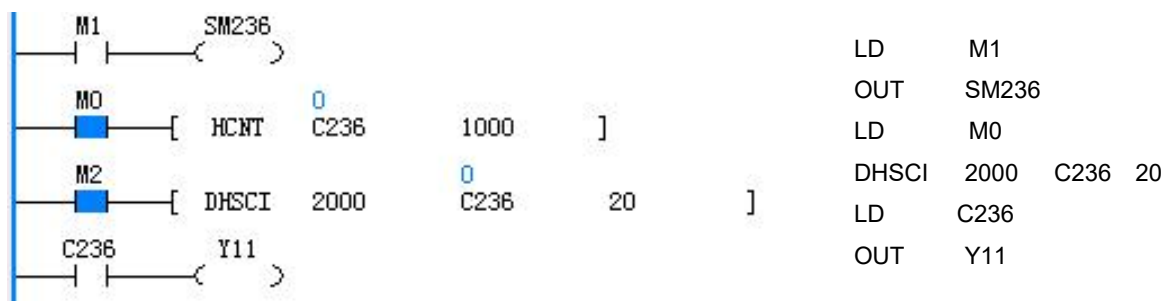
高速计数器只有在 HCNT 驱动指令的条件下根据计数输入 OFF→ON 以中断方式计数，当高速计数器的值等于 DHSCI 指令中的 **S1** 时，进入 **S3** 所指定的中断子程序中。用户在中断子程序中可以编写马上要执行的程序。

注意事项

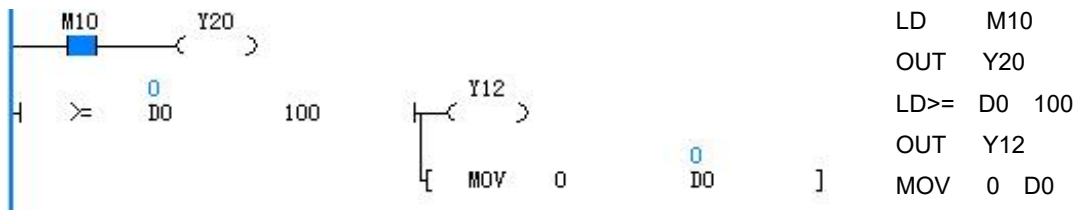
1. DHSCI 指令动作必须要和 HCNT 指令配合使用，只有 HCNT 驱动后的高速计数器，DHSCI 才能真正使用。
2. DHSCI 指令在脉冲输入时比较结果动作。因此，即使用 DMOV 或 MOV 指令等更改高速计数器值 DHSCI 不会有动作。
3. DHSCI (DHSCS, DHSCR, DHSZ, DHSP, DHST) 和普通指令一样可以多次使用，但这些指令同时驱动个数限制在总计 6 条指令以下。对超过 6 条的有效指令不执行，有效的指令按指令的有效先后决定。
4. PLC 高速计数器的最大允许频率。如使用 DHSCS, DHSCI, DHSCR, DHSZ, DHSP, DHST 命令时，将受到很大的影响。具体参考高速 IO 使用的说明。

使用示例

用户主程序如下：



用户中断号为 20 的中断程序如下：



1. M1 为 ON 时, C236 对 X0 由 OFF→ON 以中断的方式计数(X0 的输入频率参考高速 IO 的使用说明), 当 C236 由 999→1000 时 C236 触点置位, 由 1001→1000 时 C236 触点复位。C236 的触点驱动 Y11 时, Y11 的执行由用户程序的扫描决定。
2. 当 M2 为 ON 时, DHSCI 高速指令在满足注意事项所说的高速指令要求时, 当 C236 到 2000 时中断号为 20 的中断子程序马上响应, 执行在中断程序中的用户程序。
3. 当 M0 为 ON 时, SM236 驱动, C236 计数器减计数。当 M0 为 OFF 时, SM236 未驱动, C236 计数器增计数。
4. C236 在有脉冲输入的情况下, 当 C236 为 2000 时进入中断号为 20 的中断程序, 在 M10 为 ON 时 Y20 驱动, 但 Y20 的输出执行跟用户程序的扫描周期有关。同时也判断 D0 的数据大于 100 时驱动 Y12 和清除 D0 的数据。

6.10.4 DHSCR：高速计数比较复位指令

梯形图：										适用机型	APLC-TCP				
[DHSCR (S1) (S2) (D)]										影响标志位					
指令列表： DHSCR (S1) (S2) (D)										步长	10				
操作数	类型	适用软元件												变址	
S1	DINT	常数	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C		V		√
S2	DINT										C				
D	BOOL			Y	M	S					C				

操作数说明

S1：高速计数器要比较的数据，为 32 位的 DINT 数据，数据范围 -2147483648~2147483647

S2：高速计数器，高速计数器适用范围为 C236~C255

D：输出位元件对象，对 Y, M, S, C 马上复位输出不受扫描周期的影响。对 C 元件只能是 S2 本身

功能说明

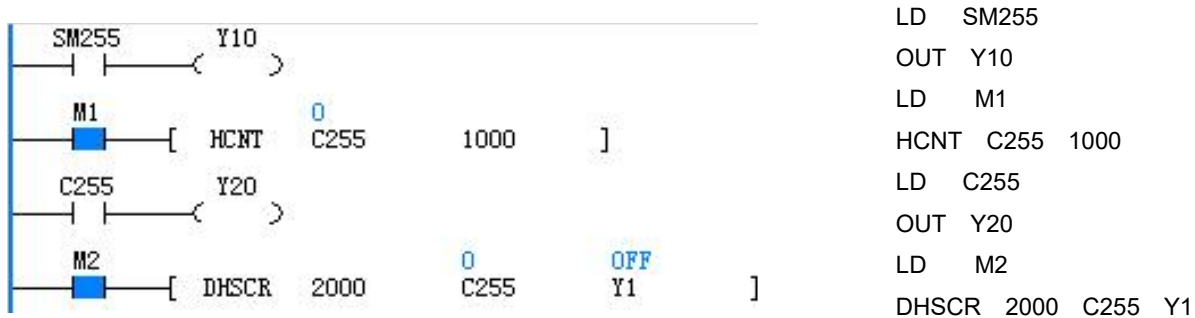
高速计数器只有在 HCNT 驱动指令的条件下根据计数输入 OFF→ON 以中断方式计数，当高速计数器的值等于 DHSCR 指令中的 S1 时, D 所指定的位元件马上复位，如果为 Y 元件, Y 元件马上输出。在希望高速计数器当前值的比较复位立即向外部输出比较结果时，使用 DHSCR 高速比较复位指令。

注意事项

1. DHSCR 指令动作必须要和 HCNT 指令配合使用，只有 HCNT 驱动后的高速计数器，DHSCR 才能真正使用。
2. DHSCR 指令在脉冲输入时比较结果动作。因此，即使使用 DMOV 或 MOV 指令等更改高速计数器值 DHSCR 不会有动作。

3. DHSCR (DHSCI, DHSCS, DHSZ, DHSP, DHST) 和普通指令一样可以多次使用, 但这些指令同时驱动个数限制在总计 6 条指令以下。对超过 6 条的有效指令不执行, 有效的指令按指令的有效先后决定。
4. PLC 高速计数器的最大允许频率。如使用 DHSCS, DHSCI, DHSCR, DHSZ, DHSP, DHST 命令时, 将受到很大的影响。具体参考高速 IO 使用的说明。

使用示例



1. M1 和 X7 同时为 ON 时, C255 对 X3 和 X4 的相位差以中断的方式计数 (相位差的输入频率参考高速 IO 的使用说明), 当 C255 由 999→1000 时 C255 触点置位, 由 1001→1000 时 C255 触点复位。C255 的触点驱动 Y20 时, Y20 的执行由用户程序的扫描周期决定。
2. 当 M2 为 ON 时, DHSCR 高速指令在满足注意事项所说的高速指令要求时, 当 C255 到 2000 时 Y1 马上输出, 不受扫描周期影响。
3. 当 X3 的输入脉冲超前 X4, SM255 为 ON, 当 X4 的输入脉冲超前 X3, SM255 为 OFF。
4. 当 X7 (C255 的启动信号) 为 OFF 时, C255 计数器不能计数。
5. 当 M1 和 X7 同时为 ON 时, 如果 X5 为 ON, C255 计数器清 0, 同时 C255 辅助触点也被清除。

6.10.5 DHSZ: 高速计数区间比较指令

梯形图:										适用机型	APLC-TCP				
— — [DHSZ (S1) (S2) (S3) (D)]										影响标志位					
指令列表: DHSZ (S1) (S2) (S3) (D)										步长	13				
操作数	类型	适用软元件												变址	
S1	DINT	常数	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C		V		√
S2	DINT	常数	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C		V		√
S3	DINT										C				
D	BOOL			Y	M	S									

操作数说明

S1: 高速计数器要比较的数据 1, 为 32 位的 DINT 数据, 数据范围 -2147483648~2147483647

S2: 高速计数器要比较的数据 2, 为 32 位的 DINT 数据, 数据范围 -2147483648~2147483647

S3: 高速计数器, 高速计数器适用范围为 C236~C255

D: 输出位元件对象, 对 Y, M, S 的处理不受扫描周期的影响

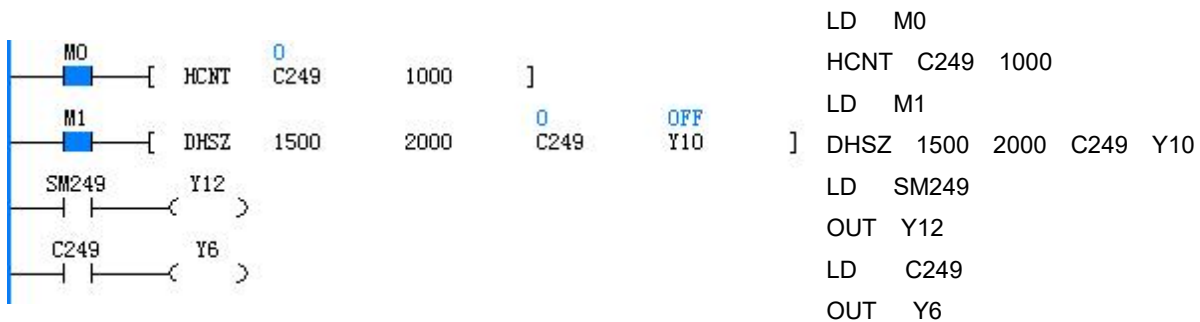
功能说明

1. 高速计数器只有在 HCNT 驱动指令的条件下根据计数输入 OFF→ON 以中断方式计数。
2. 当高速计数器的值小于指令中的 **S1** 时：**D** 所指定的位元件置位，**D** 所指定的位元件 + 1 复位，**D** 所指定的位元件 + 2 复位。
3. 当高速计数器的值大于等于 **S1** 小于等于 **S2** 时：**D** 所指定的位元件复位，**D** 所指定的位元件 + 1 置位，**D** 所指定的位元件 + 2 复位。
4. 当高速计数器的值大于 DHSZ 指令中的 **S2** 时：**D** 所指定的位元件复位，**D** 所指定的位元件 + 1 复位，**D** 所指定的位元件 + 2 置位。
5. 如果为 Y 元件，Y 元件马上输出相应的状态，输出动作与程序扫描周期无关。

注意事项

1. DHSZ 指令动作必须要和 HCNT 指令配合使用，只有 HCNT 驱动后的高速计数器，DHSZ 才能真正使用。
2. DHSZ 指令在脉冲输入时比较结果动作。因此，即使使用 DMOV 或 MOV 指令等更改高速计数器值，DHSZ 也不会有动作。
3. DHSCZ (DHSCI, DHSCS, DHSCR, DHSP, DHST) 和普通指令一样可以多次使用，但这些指令同时驱动的个数限制在总计 6 条指令以下。对超过 6 条的有效指令不执行，有效的指令按指令的有效先后决定。
4. PLC 高速计数器的最大允许频率。如使用 DHSCS, DHSCI, DHSCR, DHSZ, DHSP, DHST 命令时，将受到很大的影响。具体参考高速 IO 使用的说明。

使用示例



1. M0 和 X6 同时为 ON 时，C249 对 X0 由 OFF→ON 增计数，C249 对 X1 由 OFF→ON 减计数，（两相的输入频率参考高速 IO 的使用说明），当 C249 由 999→1000 时 C249 触点置位，由 1001→1000 时 C249 触点复位。C249 的触点驱动 Y6 时，Y6 的执行由用户程序的扫描决定。
2. 当 M1 为 ON 时，DHSZ 高速指令在满足注意事项所说的高速指令要求时，Y10，Y11 和 Y12 的状态如下：
 - (1) $C249 < 1000$: Y10, Y12: ON; Y11: OFF。
 - (2) $2000 \geq C249 \geq 1000$: Y10, Y12: OFF; Y11: ON。
 - (3) $C249 > 2000$: Y10, Y11: OFF; Y12: ON。
 Y10, Y11, Y12 的输出不受扫描周期影响。
3. 当 M0 和 X6 同时为 ON 时，如果 X0 由 OFF→ON 增计数时 SM249 复位。如果 X1 由 OFF→ON 减计数时 SM249 置位。
4. 当 X6 为 OFF 时，C249 计数器不能计数。
5. 当 M0 和 X6 同时为 ON 时，如果 X2 为 ON，C249 计数器清 0，同时 C249 辅助触点也被清除。

6.10.6 DHST：高速计数表格比较指令

梯形图： 									适用机型		APLC-TCP						
									影响标志位								
指令列表： DHST (S1) (S2) (S3)									步长		10						
操作数		类型	适用软元件											变址			
S1		DINT							D								
S2		INT	常数														
S3		DINT									C						

操作数说明

S1：表格比较的数据起始单元（D 元件起始号）。其后序列号相连的三个 D 元件用于指定高速计数器要比较的数据、Y 元件序号及其相应输出状态。这四个序列号相连的 D 元件合称一个记录。

S2：要比较的记录数量，数据范围为 1~128

S3：高速计数器，适用范围为 C236~C255

功能说明

1. 高速计数器只有在 HCNT 驱动指令的条件下，根据计数输入 OFF→ON 以中断方式计数。
2. 当高速计数器的值等于当前要比较记录的数据，则按记录的数据输出相应的 Y 元件状态，输出的对象只能为 Y 元件。
3. 输出动作与扫描周期无关，当前记录指定的 Y 元件将立即输出指定的状态。
4. 在希望用户程序按某个表格指定的比较数据和 Y 元件执行立即输出的操作时，使用 DHST 表格比较输出指令。

注意事项

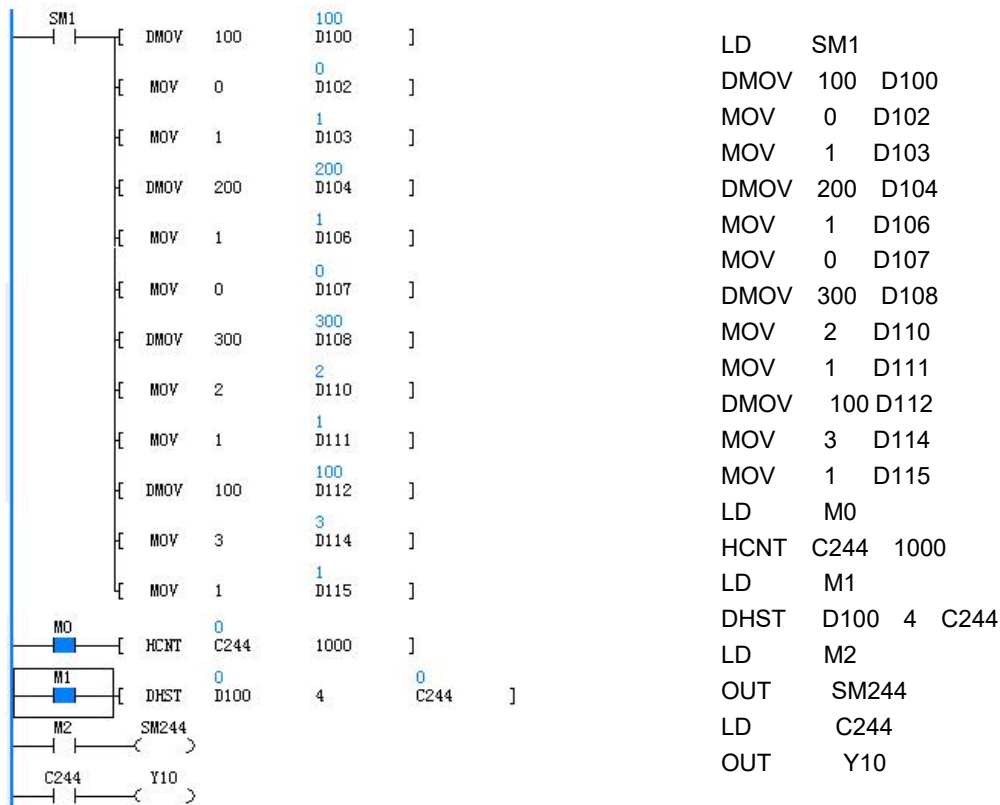
1. DHST 指令动作必须要和 HCNT 指令配合使用，只有 HCNT 驱动后的高速计数器，DHST 才能正确执行。
2. DHST 指令在脉冲输入时比较结果动作。因此，即使使用 DMOV 或 MOV 指令等更改高速计数器值，DHST 也不会有动作。
3. DHST (DHSCI, DHSCS, DHSCR, DHSP, DHSZ) 和普通指令一样可以多次使用，但这些指令同时驱动个数限制在总计 6 条指令以下。对超过 6 条的有指令不执行，有效的指令按指令的有效先后决定。在用户的指令中如果 DHSP 为有效指令，DHST 就不执行。反之 DHST 为有效指令，DHSP 就不执行。对用户程序中在同一时刻只能有一条指令（DHST 或 DHSP）有效。
4. PLC 高速计数器的最大允许频率。如使用 DHSCS, DHSCI, DHSCR, DHSZ, DHSP, DHST 命令时，将受到很大的影响。具体参考高速 I/O 使用的说明。

使用示例

表格数据如下表所示：

比较数据		输出 Y 编号	置位/复位	操作流程
高位	低位			
D100=0	D101=100	D102=0	D103=1	1 ↓
D104=0	D105=200	D106=1	D107=0	2 ↓
D108=0	D109=300	D110=2	D111=1	3 ↓
D112=0	D113=300	D114=3	D115=1	4 ↓ 返回到 1

梯形图如下所示：



1. 用户程序的第一个扫描周期，对 D100→D115 赋初值，产生要比较的表格。
2. M0 和 X6 同时为 ON 时，C244 对 X0 由 OFF→ON 计数，（输入频率参考高速 IO 的使用说明），当 C244 由 999→1000 时 C244 触点置位，由 1001→1000 时 C244 触点复位。C244 的触点驱动 Y10 时，Y10 的执行由用户程序的扫描周期决定。
3. 当 M1 为 ON 时，DHST 高速指令在满足注意事项所说的高速指令要求时，从表格的记录号 1 开始，在 1 号记录完成后才进入 2 号记录的比较，每次只有在前次指令完成后才进入下条记录比较。当最后条记录比较完成重新回到第 1 条记录比较，同时 SM185 置位。SD184 表示当前要比较的记录号，SD182 和 SD183 表示当前要比较的数据。对比较完成的结果马上输出，不受扫描周期影响。
4. 当 M2 为 ON 时，SM244 为 ON，C244 为减计数，如果 M2 为 OFF 时，SM244 为 OFF，C244 为增计数。
5. 当 X6 为 OFF 时，C244 计数器不能计数。
6. 当 M0 和 X6 同时为 ON 时，如果 X2 为 ON，C244 计数器清 0，同时 C244 辅助触点也被清除。

6.10.7 DHSP：高速计数表格比较脉冲输出指令

梯形图：										适用机型	APLC-TCP			
[DHSP (S1) (S2) (S3)]										影响标志位				
指令列表：DHSP (S1) (S2) (S3)										步长	10			
操作数	类型	适用软元件										变址		
S1	DINT									D				
S2	INT	常数												
S3	DINT										C			

操作数说明

S1: 表格比较的数据起始单元 (D 元件起始号)。其后序列号相连的三个 D 元件用于指定高速计数器要比较的数据, 以及输出到 SD180 和 SD181 数据。这四个序列号相连的 D 元件合称一个记录。

S2: 要比较的记录数量, 数据范围为 1~128

S3: 高速计数器, 适用范围为 C236~C255

功能说明

1. 高速计数器只有在 HCNT 驱动指令的条件下根据计数输入 OFF→ON 以中断方式计数。
2. 当高速计数器的值等于当前记录的比较数据, 则按当前记录的输出数据更改 SD180 和 SD181。
3. 在希望用户程序按某个表格决定高速输出或其它数据的赋值时, 使用 DHSP 表格比较输出指令。例如可指定 SD180 和 SD181 (双字) 为 PLSY 指令的输出频率操作数, 使 PLSY 输出频率按表格比较结果进行调整。

注意事项

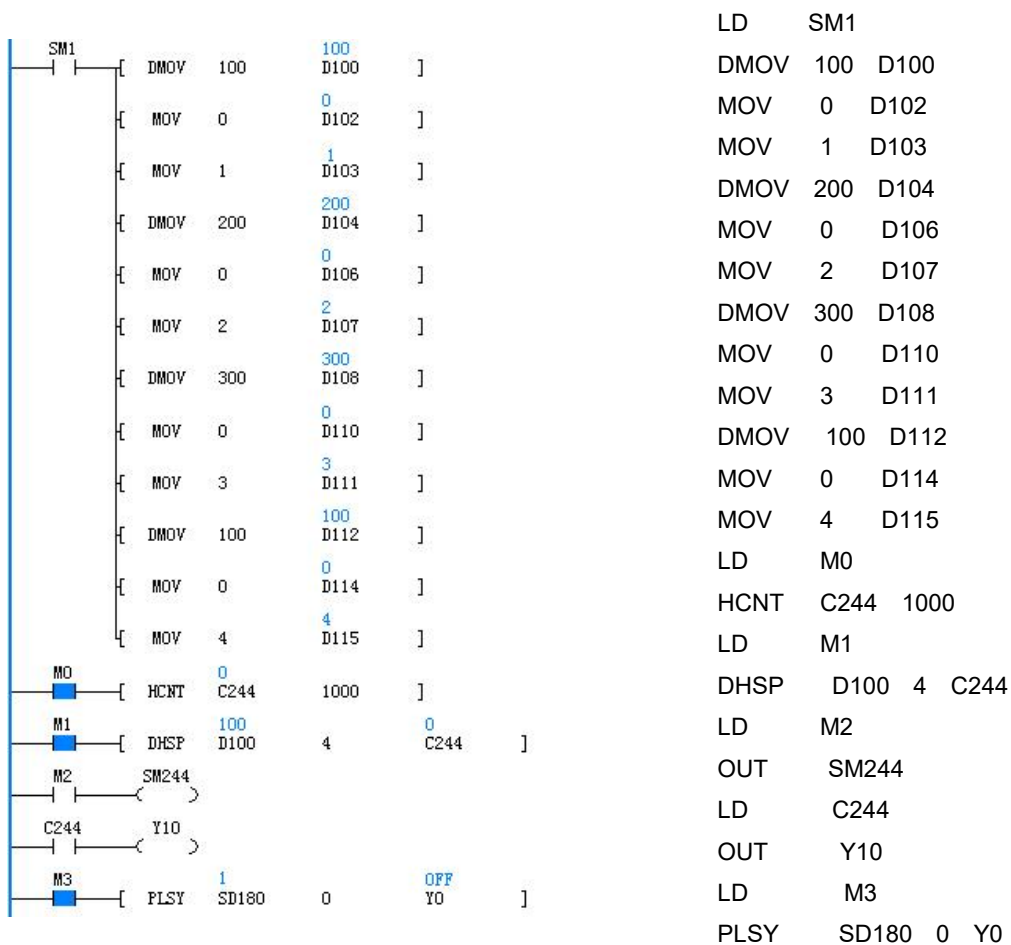
1. DHSP 指令动作必须要和 HCNT 指令配合使用, 只有 HCNT 驱动后的高速计数器, DHST 才能正确执行。
2. DHSP 和 PLSY 配合使用时送到 SD180 和 SD181 的数据要满足 PLSY 的频率输出。要求具体参考 PLSY 指令。
3. 当比较希望在最后一行停止时, 把最后表格送到 SD180 和 SD181 的数据设为 0。在此情况下, 其它的 DHST 和 DHSP 指令无效, 但此时的 DHSP 指令不占用其它高速指令的总条数。
4. DHSP 指令在脉冲输入时比较结果动作。因此, 即使使用 DMOV 或 MOV 指令等更改高速计数器值, DHST 也不会有动作。
5. DHSP (DHSCI, DHSCS, DHSCR, DHST, DHSZ) 和普通指令一样可以多次使用, 但这些指令同时驱动个数限制在总计 6 条指令以下。对超过 6 条的有指令不执行, 指令的有效性按其有效先后决定。
6. 在用户的指令中如果 DHSP 为有效指令, DHST 就不执行。反之 DHST 为有效指令, DHSP 就不执行。对用户程序中在同一时刻只能有一条指令 (DHST 或 DHSP) 有效, 其它为无效。
7. 注意 PLC 高速计数器的最大允许频率。如使用 DHSCS, DHSCI, DHSCR, DHSZ, DHSP, DHST 命令时, 将受到很大的影响。具体参考高速 IO 使用的说明

使用示例

表格数据如下表所示:

比较数据		输出到 SD180 和 SD181 数据		操作流程
高位	低位	高位	低位	
D100=0	D101=100	D102=0	D103=1	1 ↓
D104=0	D105=200	D106=0	D107=2	2 ↓
D108=0	D109=300	D110=0	D111=3	3 ↓
D112=0	D113=300	D114=0	D115=4	4 ↓ 从 1 返回

梯形图如下所示：



1. 用户程序扫描的第一个周期时，对 D100→D115 赋初值，产生要比较的表格数据。
2. M0 和 X6 同时为 ON 时，C244 对 X0 由 OFF→ON 计数，（输入频率参考高速 IO 的使用说明），当 C244 由 999→1000 时 C244 触点置位，由 1001→1000 时 C244 触点复位。C244 的触点驱动 Y10 时，Y10 的执行由用户程序的扫描周期决定。
3. 当 M1 为 ON 时，DHSP 高速指令在满足注意事项所说的高速指令要求时，从表格的记录号 1 开始，在 1 号记录完成后才进入 2 号记录的比较，每次只有在前次指令完成后才进入下条记录比较。当最后条记录比较完成重新回到第 1 条记录比较，同时 SM185 置位。SD184 表示当前要比较的记录号，SD182 和 SD183 表示当前要比较的数据。对比较完成的结果输出操作数分别放到 SD180 和 SD181 单元中，不受扫描周期影响。希望比较在最后一行停止时把表格中的最后一格送到 SD180 和 SD181 的数据设为 0。
4. 当 M2 为 ON 时，SM244 为 ON，C244 为减计数，如果 M2 为 OFF 时，SM244 为 OFF，C244 为增计数。
5. 当 X6 为 OFF 时，C244 计数器不能计数。
6. 当 M0 和 X6 同时为 ON 时，如果 X2 为 ON，C244 计数器清 0，同时 C244 触点也被清除。

6.10.8 SPD：测频指令

梯形图： — — [SPD (S1) (S2) (D)]										适用机型	APLC-TCP				
指令列表： SPD (S1) (S2) (D)										影响标志位					
步长										7					
操作数	类型	适用软元件													变址
S1	BOOL		X												
S2	WORD	常数	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	√
D	WORD								D				V		√

操作数说明

S1：输入点，可设定范围：X0~X5

S2：输入点检测的单位时间，以 ms 为单位，操作数 S2>0

D：检测脉冲数据保存单元，当计数超过 65535，自动溢出处理

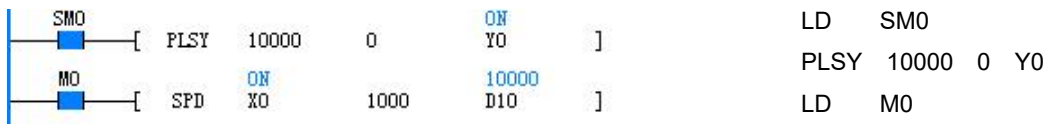
功能说明

检测 X0~X5 在指定的时间（ms）内输入脉冲个数，将结果存储在指定的软元件单元中。

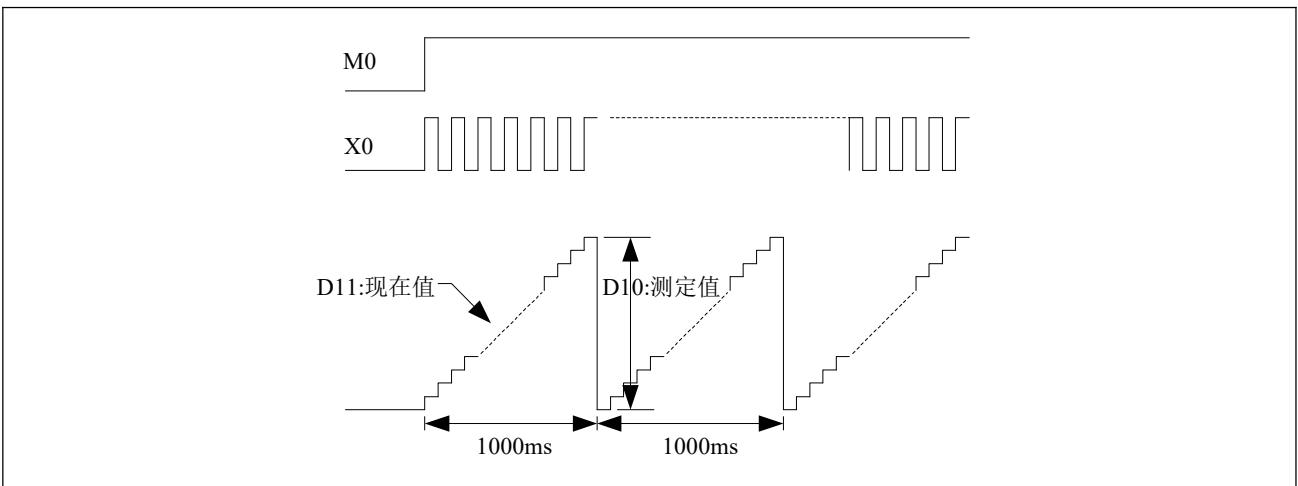
注意事项

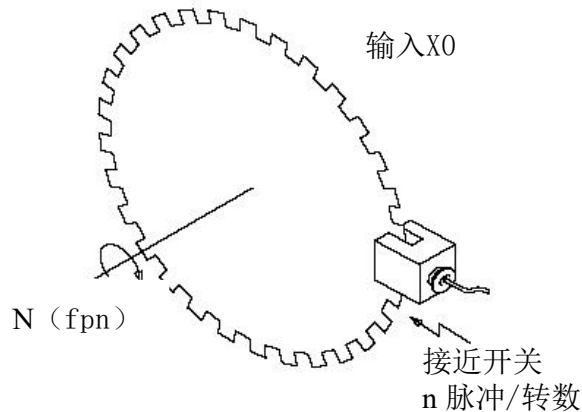
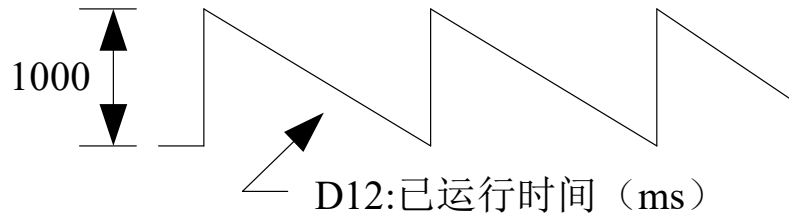
1. SPD 和 HCNT、外部输入中断、脉冲捕捉存在硬件冲突。具体请参见 6.10 高速 IO 指令。
2. SPD 的输入点只能为 X0~X5，其它的不支持。
3. SPD 的脉冲输入频率最大为 10kHz，超过 10kHz 检测有可能存在误差。
4. SPD 的输入脉冲频率必须参考系统总的脉冲频率输入输出要求。

使用示例



SPD X0 1000 D10 对程序的操作实例的时序操作如下：





1. M0 为 ON 时将 X0 指定的输入脉冲在 1000ms 内计数，将计数的结果保存在 D10 的存储单元中，其中 D11 为在 1000ms 内的计数的现在值，D12 为 1000ms 内已运行的时间。
2. D10 的数据和上图中的旋转速度成正比关系。
3. 对 X0 的每次 OFF→ON 计数每隔 1000ms 后保存到 D10 中。

6.10.9 PLSY：计数脉冲输出指令

梯形图： 									适用机型							
									影响标志位							
指令列表： PLSY (S1) (S2) (D)									步长		9					
操作数		类型	适用软元件											变址		
S1		DINT	常数	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C		V		√
S2		DINT	常数	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C		V		√
D		BOOL			Y											

操作数说明

S1：指定频率 (Hz)

可设定范围：1~100000 (Hz)。当 **S1** 小于等于 0 或大于 100000 时，系统报指令操作数非法，同时不占系统硬件资源。
在指令运行过程中更改 **S1** 的内容，输出的频率也随之发生变化。

S2：产生的脉冲量 (PLS)。

可设定范围：0~2147483647。设定操作数不在本范围之内时，系统报指令操作数非法，脉冲不输出，也不占用系统资源。

S2 为 0 时，在指令有效下脉冲始终输出。

在指令运行过程中更改 **S2** 的内容，在下次驱动有效的情况下操作数才有作用。

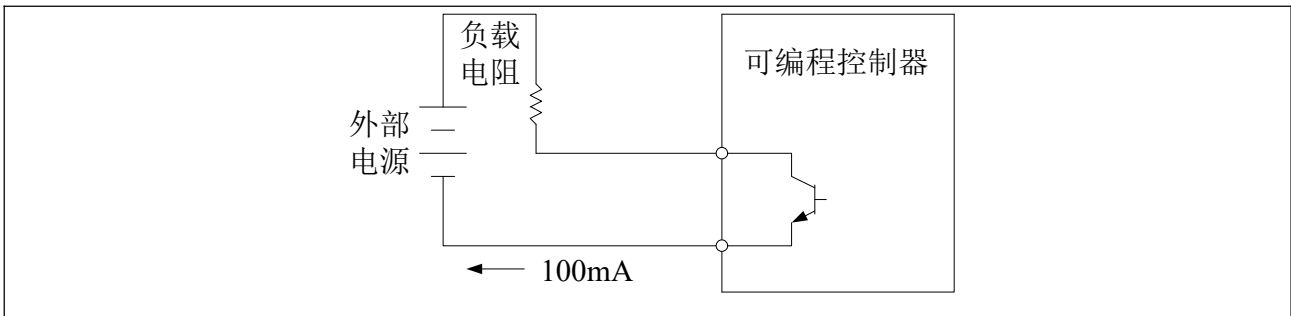
D: 高速脉冲输出点，只能指定 Y0 或 Y1

功能说明

根据指令指定的频率产生指定数量的高速脉冲输出。为了输出高速脉冲，PLC 的输出晶体管上的负载电流要大，但不能超过额定负载电流。

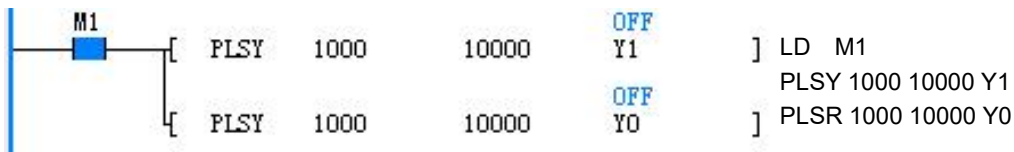
注意事项

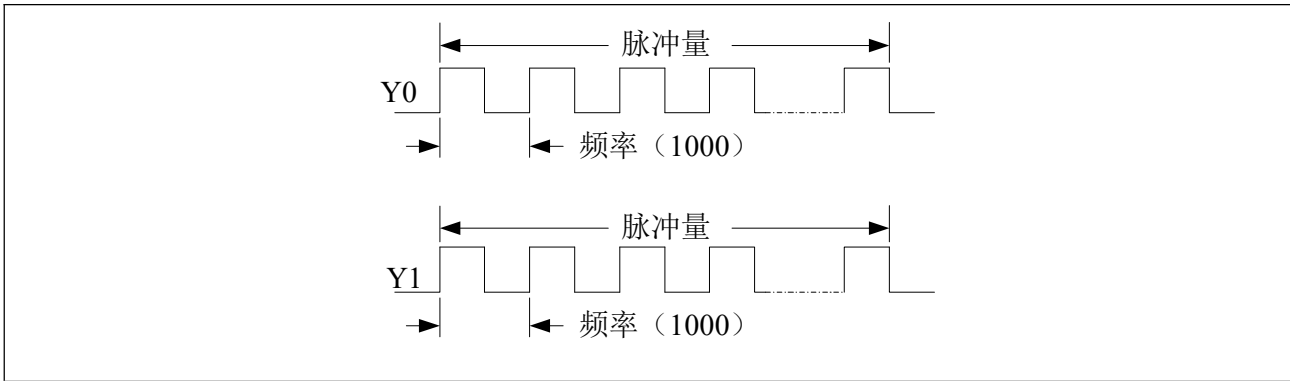
1. PLC 必须使用晶体管输出方式。
2. PLC 执行高速脉冲输出时，必须使用下列所述的 PLC 输出晶体管规定的负载电流。
3. 针对 PLSY, PWM, PLSR 的输出回路（晶体管）如下：



4. 在高负载时晶体管的 OFF 时间较长，在 PWM, PLSY, PLSR 指令时，要求晶体管输出端接相应的负载，当输出的波形不满足指令的操作数时，可以加大晶体管的负载电流（晶体管的负载 $\leq 100\text{mA}$ ）。
5. 在高速指令有效运行（包括输出完成）时，对同一端口的其它操作无效。只有在高速脉冲输出指令无效时，其它指令才能操作本端口。
6. 使用 2 个 PLSR 指令能够在 Y0 和 Y1 输出点得到各自独立的高速脉冲输出，也可和 PWM 或 PLSR 在不同的输出点（Y0, Y1）得到各自独立的高速脉冲输出。
7. 有多条 PWM、PLSY 或 PLSR 指令操作同一端口时，先有效的指令控制端口输出状态，后有效的指令对输出点的状态无影响。
8. 与其它高速指令（DHSCS, DHSCR, DHSZ, DHSP, DHST, HCNT）相同，PLSY 指令要满足系统中对高速输入和高速脉冲输出的要求。

使用示例





1. M1 为 ON 时，从 Y0、Y1 端口输出 10000 个频率为 1000Hz 的脉冲，完成 10000 个脉冲后，不再输出。当 M0 出现由 OFF 向 ON 跳变时，重新下一次输出。M0 为 OFF 时，端口输出 OFF。
2. 脉冲的占空比为 50%ON，50%OFF。输出控制不受扫描周期的影响，采用中断处理。在高频输出时，从 Y 端口的输出占空比跟负载有关系。从输出端子（Y0 和 PORT 0，Y1 和 PORT 1）得到的波形跟用户的输出负载有关系，在满足不能超过额定负载电流情况下，负载越小，输出波形越接近设定操作数。
3. SM80 对应 Y0 的输出使能，SM81 对应 Y1 的输出使能，为 1 的情况下输出脉冲。
4. SM82，SM83 对应 Y0 和 Y1 的输出标志，当输出完成或 M0 为 OFF 标志清除。
5. SD50 对应 Y0 在 PLSY 和 PLSR 指令的输出脉冲数量高位。
SD51 对应 Y0 在 PLSY 和 PLSR 指令的输出脉冲数量低位。SD52 对应 Y1 在 PLSY 和 PLSR 指令的输出脉冲数量高位。
SD53 对应 Y1 在 PLSY 和 PLSR 指令的输出脉冲数量低位。SD54 对应 Y0 和 Y1 在 PLSY 和 PLSR 指令的输出脉冲数量高位。
SD55 对应 Y0 和 Y1 在 PLSY 和 PLSR 指令的输出脉冲数量低位。
6. SD50~SD55 可以通过“DMOV *** SD5*”或“MOV *** SD5*”更改。也可通过监控更改。
7. 如果要使用输入脉冲个数控制 PLSY 的输出脉冲频率，具体参考 DHSP 指令。

6.10.10 PLSR：带加减速的计数脉冲输出指令

梯形图：									适用机型							
—— —— [PLSR (S1) (S2) (S3) (D)]									影响标志位							
指令列表： PLSR (S1) (S2) (S3) (D)									步长		10					
操作数		类型	适用软元件											变址		
S1	WORD	常数	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	√	
S2	DINT	常数	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C			V		√
S3	WORD	常数	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	√	
D1	BOOL			Y												

操作数说明

S1：最高频率（Hz）。可设定范围：10~20000（Hz）。

间接指定的操作数当大于 20000 按 20000 处理，当小于 10 按 10 处理，系统报指令操作数非法，高速脉冲输出按默认的数据输出。

S2: 总输出脉冲数 (PLS)。可设定范围: 110~2147483647。设定操作数不在本范围之内系统提示指令操作数数值非法错误, 同时脉冲不输出, 也不占用本指令对应的硬件资源。

S3: 加减速时间 (ms)

操作数 $S1 \times S3 < 100000$ 时, 系统按 $S3 = 100000 / S1$ 处理, 同时系统提示 PLSR 指令参数错误告警, 加减速时序不一定。

操作数 $S1 \times S3 > S2 \times 909$ 时, 系统按 $S3 = S2 \times 909 / S1$ 处理, 同时系统提示 PLSR 指令参数错误告警, 加减速时序不一定。

注意

对 AT100, 加减速时间不能低于 50ms。

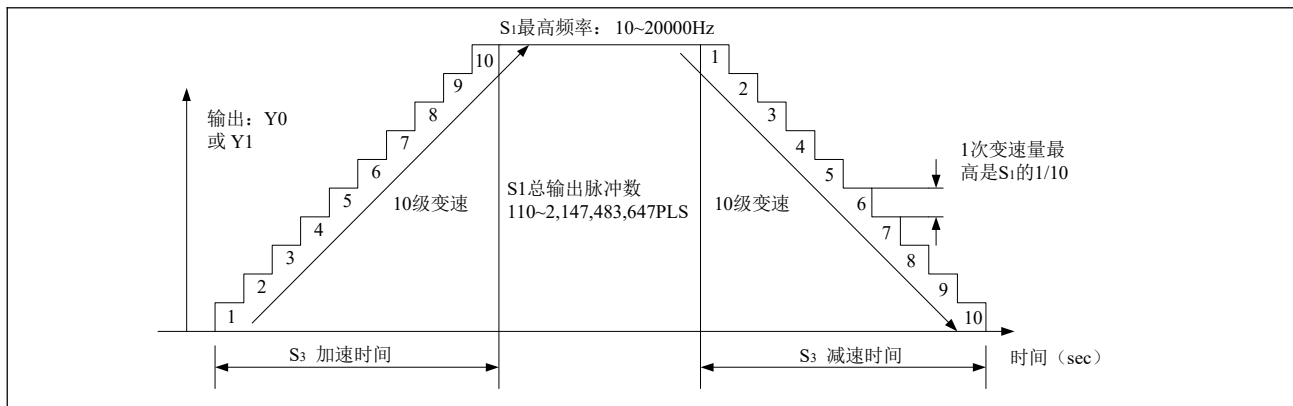
加减速时的变速次数按固定的 10 次处理, 每次变化量为 $S1/10$ 。

D: 高速脉冲输出点, 只能指定 Y0 或 Y1。

功能说明

带加速减速功能的定尺寸传送用的高速脉冲输出指令。针对指定的最高频率进行定加速, 在达到所指定的输出脉冲数后, 进行定减速。

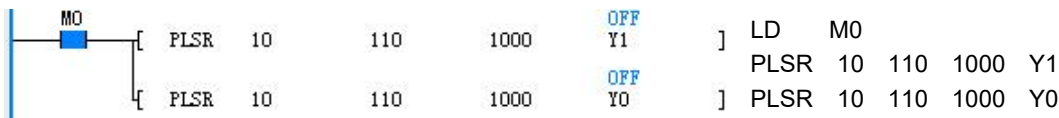
操作过程如图所示:



注意事项

1. 该命令的输出频率为 10~20000Hz, 加减速时的变速速度超过此范围时, 自动在范围内调整操作数, 本指令不受扫描周期的影响。
2. PLC 请使用晶体管输出, 在高速脉冲输出时请按规定负载电流接入输出晶体管。从输出端子 (Y0 和 PORT 0, Y1 和 PORT 1) 得到的波形跟用户的输出负载有关系, 在满足不能超过额定负载电流情况下, 负载越小, 输出波形越接近设定操作数。
3. 在高速指令有效运行 (包括输出完成) 时, 对同一端口的其它操作无效。只有在高速脉冲输出指令无效时, 其它指令才能操作本端口。
4. 使用 2 个 PLSR 指令能够在 Y0 和 Y1 输出端得到各自独立的高速脉冲输出, 也可和 PWM 或 PLSY 在不同的输出端 (Y0, Y1) 得到各自独立的高速脉冲输出。
5. 有多条 PWM、PLSY 或 PLSR 指令操作同一端口时, 先有效的指令控制端口输出状态, 后有效的对输出端的状态无影响。
6. 与其它高速指令 (DHSCS, DHSCR, DHSZ, DHSP, DHST, HCNT) 相同, PLSR 指令要满足系统中对高速输入和高速脉冲输出的要求。

使用示例



1. M0 为 ON 时按照设定值从 Y0、Y1 端口输出脉冲，完成 110 个脉冲后，不再输出。当 M0 出现由 OFF 向 ON 跳变时，重新下一次输出。M0 为 OFF 时，端口输出 OFF。
2. 在指令执行的过程中所有的操作数都不进行更改，按最先有效时的操作数处理，只有 M0 由 ON→OFF→ON 新操作数才有效。
3. SM80 对应 Y0 的输出使能，SM81 对应 Y1 的输出使能，SM80 和 SM81 为 1 的情况下对应的输出点输出中断。
4. SM82，SM83 对应 Y0 和 Y1 的输出标志，当输出完成或 M0 为 OFF 时，对应的 SM82 或 SM83 为 OFF。当正在输出时，对应的 SM82 或 SM83 为 ON 状态。
5. SD50~SD55 分别对应：
 - 1) SD50: Y0 在 PLSY 和 PLSR 指令的输出脉冲数量高位。
 - 2) SD51: Y0 在 PLSY 和 PLSR 指令的输出脉冲数量低位。
 - 3) SD52: Y1 在 PLSY 和 PLSR 指令的输出脉冲数量高位。
 - 4) SD53: Y1 在 PLSY 和 PLSR 指令的输出脉冲数量低位。
 - 5) SD54: Y0 和 Y1 在 PLSY 和 PLSR 指令的输出脉冲数量高位。
 - 6) SD55: Y0 和 Y1 在 PLSY 和 PLSR 指令的输出脉冲数量低位。
6. SD50~SD55 可以通过 (DMOV *** SD5×) 或 (MOV *** SD5×) 更改。也可通过监控更改。

6.10.11 PLS：包络线脉冲输出指令

梯形图：									适用机型							
┌──┴──┐ [PLS (S1) (S2) (D1)]									影响标志位							
指令列表：PLS (S1) (S2) (D1)									步长		7					
操作数		类型	适用软元件												变址	
S1	WORD	常数	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	√	
S2	DINT	常数	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C		V		√	
D1	BOOL			Y												

操作数说明

S1：参数指定的 D 元件起始地址

S2：输出段数 0~255

D：高速脉冲输出点（只能指定 Y0 或 Y1）

功能说明

1. 利用后台的指令向导生成 PLS 指令，以子程序的形式进行调用。能流一导通，系统就按照设置输出相应脉冲。用户可以控制运行或停止脉冲发生器，设置频率、脉冲量。
2. 当段的数目为 0 的时候，没有输出。

3. 使用 SM80, SM81 使能可以关断高速脉冲输出, 其它的标志位与高速共用。
4. 后台软件生成的子程序 PLS_SET 的内容 (设 n 为其 D 元件号, M 为总段数) :

LD SM0

DMOV 第 1 段步骤频率 Dn

DMOV 第 1 段步骤脉冲数 Dn + 2

DMOV 第 2 段步骤频率 Dn + 4

DMOV 第 2 段步骤脉冲数 Dn + 6

DMOV 第 3 段步骤频率 Dn + 8

DMOV 第 3 段步骤脉冲数 Dn + 10

.....

DMOV 第 M 段步骤频率 Dn + 4M - 4

DMOV 第 M 段步骤脉冲数 Dn + 4M - 2

DMOV 最大速度 Dn + 4M

MOV 最小速度 Dn + 4M + 2

MOV 加速时间 Dn + 4M + 3

MOV 减速时间 Dn + 4M + 4

注意事项

1. 建议使用 PTO 向导生成的 PLS 指令, 如果直接编写 PLS 指令, 注意各步骤的脉冲个数不能过小。在设定的加速度下, 每一个步骤的脉冲个数必须大于频率间变换所需要的最少脉冲数。

2. 用 P 表示某步骤输出的脉冲个数, F_N 表示第 N 段的频率, F_{\max} 、 F_{\min} 表示最高速度和最低速度, T_{up} 、 T_{down} 表示加速时间和减速时间。

1) 当步骤 N 的速度大于步骤 N-1 的速度时, 步骤 N 的脉冲个数必须满足下面的条件:

$$P \geq \frac{(F_N + F_{N-1}) \times (F_N - F_{N-1}) \times T_{up}}{2000 \times (F_{\max} - F_{\min})}$$

2) 当步骤 N 的速度小于步骤 N-1 的速度时, 步骤 N 的脉冲个数必须满足下面的条件:

$$P \geq \frac{(F_N + F_{N-1}) \times (F_N - F_{N-1}) \times T_{down}}{2000 \times (F_{\max} - F_{\min})}$$

3. 特别的:

1) 当 N=1 时, 步骤 N-1 的频率取 F_{\min} , 代入上述公式。

2) 当所有步骤数为 1 时, 即只有 1 段时, 脉冲个数必须满足下面的条件:

$$P \geq \frac{(F_1 + F_{\min}) \times (F_1 - F_{\min}) \times (T_{up} + T_{down})}{2000 \times (F_{\max} - F_{\min})}$$

3) 最后一个步骤的脉冲个数要满足下面的公式:

$$P \geq \frac{(F_M + F_{M-1}) \times (F_M - F_{M-1}) \times (T_{up} + T_{down})}{2000 \times (F_{\max} - F_{\min})}$$

4) 各步骤中指定的频率不能大于先前设定的最高速度, 也不能低于最低速度。

5) 各步骤的脉冲总数最大为 999, 999。

4. PLC 请使用晶体管输出, 在高速脉冲输出时请按规定负载电流接入输出晶体管。从输出端子 (Y0 和 PORT 0, Y1 和 PORT 1) 得到的波形跟用户的输出负载有关系, 在满足不能超过额定负载电流情况下, 负载越小, 输出波形越接近设定操作数。

5. 在高速指令有效运行 (包括输出完成) 时, 对同一端口的其它操作无效。只有在高速脉冲输出指令无效时, 其它指令才能操作本端口。

6. PLSY,PLSR,PLS, 定位指令可以利用 Y0,Y1 两个端口输出高速脉冲。请注意不要同时对同一个高速端口使用这些指令进行高速脉冲输出。

6.10.12 PLSB：带基底频率与加减速的计数脉冲输出指令

<div>梯形图：</div> <div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><</div></div>							
---	--	--	--	--	--	--	--

操作数说明

S1：基底频率 (HZ)

可设定范围：0－20000(Hz)。设定操作数不在本范围之内系统提供指令操作数数值非法错误，同时脉冲不输出输出，也不占用本指令对应的硬件资源。

S2：最高频率 (HZ)

可设定范围：10－20000 (Hz)。设定操作数不在本范围之内系统提供指令操作数数值非法错误，同时脉冲不输出输出，也不占用本指令对应的硬件资源。

S3：总数出脉冲数(PLS)

可设定范围：5-999999；设定操作数不在本范围之内系统提供指令操作数数值非法错误，同时脉冲不输出输出，也不占用本指令对应的硬件资源。

S4：加减速时间(mS)

可设定范围：0-10000；设定操作数不在本范围之内系统提供指令操作数数值非法错误，同时脉冲不输出输出，也不占用本指令对应的硬件资源。

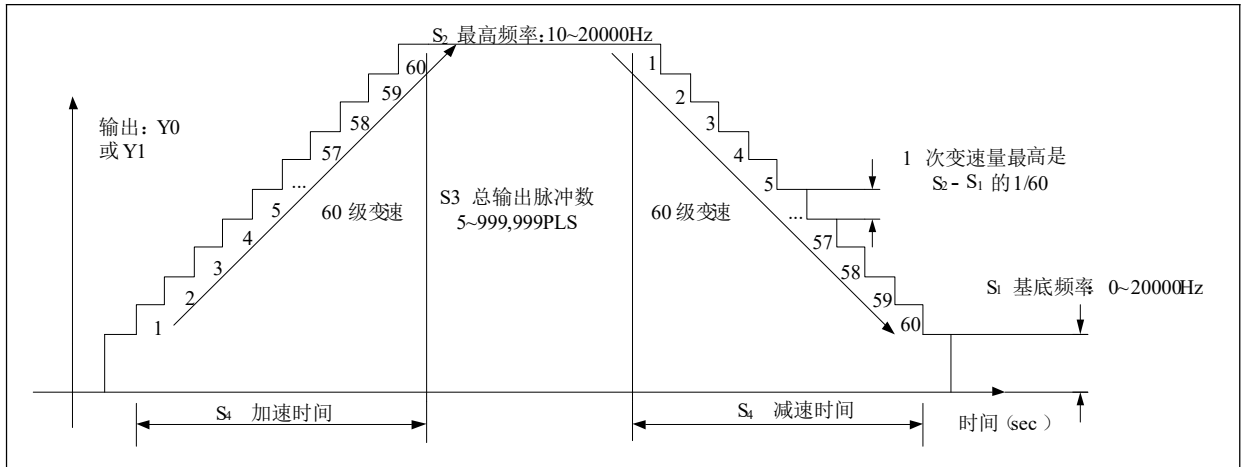
D：脉冲输出端口

只能指定 Y000 或 Y001。

功能说明

带基底频率与加速减速功能的定尺寸传送用的高速脉冲输出指令。针对指定的最高频率以基底频率开始进行定加速，在达到所指定的输出脉冲数后，进行定减速直至基底频率。

操作过程如图所示：



注意事项

1. 该命令的输出频率为 10~20000Hz，加减速时的变速速度超过此范围时，自动在范围内调整操作数，本指令不受扫描周期的影响。
2. 当用户设置的操作数不合理时（如脉冲个数与对应的加减速时间与频率相比过小），将按照用户设置操作数所决定加速度进行加减速，同时保证输出脉冲个数准确，但输出频率可能低于操作数设置值，加减速时间也可能小于设定值。
3. 本指令采取 60 级加减速，但在用户设置操作数不合理或脉冲数过小时，将适当减少加减速级数。
4. 可编程控制器请使用晶体管输出，在高频脉冲输出时请按规定负载电流接入输出晶体管；从输出端子（Y0 和 COM0，Y1 和 COM1）得到的波形跟用户的输出负载有关系，在满足不能超过额定负载电流情况下，负载小，输出波形越接近设定操作数。
5. 在高速指令有效运行(包括输出完成)时，对同一端口的其它操作无效；只有在高速输出指令无效时，其它指令才能操作本端口。
6. 使用 2 个 PLSB 指令能够在 Y000 和 Y001 输出端得到各自独立的脉冲输出，也可和 PWM 或 PLSY 在不同的输出端（Y000，Y001）得到各自独立的脉冲输出。
7. 有多条 PWM、PLSY 或 PLSB 指令操作同一端口时，先有效的指令控制端口输出状态，后有效的对输出端的状态无影响。
8. 与其它高速指令（DHSCS，DHSCR，DHSZ，DHSP，DHST，HCNT），SPD 指令要满足系统中对高速输入和高速输出的要求。
9. 高速指令、包络线指令、定位指令可以利用 Y0,Y1 两个端口输出高速脉冲。请注意不要同时对同一个高速端口使用这些指令进行高速输出。

使用示例

	PLSB	100	10000	10000	1000	OFF Y1
	PLSB	100	10000	10000	1000	OFF Y0
	LD	M0				
	PLSR	100	10000	10000	1000	Y1
	PLSR	100	10000	10000	1000	Y0

1. M0 为 ON 时按照设定值从 Y0、Y1 端口输出脉冲，完成 10000 个脉冲后，不再输出。当 M0 出现由 OFF 向 ON 跳变时，重新下一次输出。M0 为 OFF 时，端口输出 OFF。

2. 在指令执行的过程中所有的操作数都不进行更改，按最先有效时的操作数处理，只有 M0 由 ON→OFF→ON 新操作数才有效。
3. SM80 对应 Y0 的输出使能，SM81 对应 Y1 的输出使能，SM80 和 SM81 为 1 的情况下对应的输出点输出中断。
4. SM82，SM83 对应 Y0 和 Y1 的输出标志，当输出完成或 M0 为 OFF 时，对应的 SM82 或 SM83 为 OFF。当正在输出时，对应的 SM82 或 SM83 为 ON 状态。
5. SD50~SD55 分别对应：
 - 1) SD50: Y0 在 PLSB 指令的输出脉冲数量高位。
 - 2) SD51: Y0 在 PLSB 指令的输出脉冲数量低位。
 - 3) SD52: Y1 在 PLSB 指令的输出脉冲数量高位。
 - 4) SD53: Y1 在 PLSB 指令的输出脉冲数量低位。
 - 5) SD54: Y0 和 Y1 在 PLSB 指令的输出脉冲数量高位。
 - 6) SD55: Y0 和 Y1 在 PLSB 指令的输出脉冲数量低位。
6. SD50~SD55 可以通过 (DMOV *** SD5×) 或 (MOV *** SD5×) 更改。也可通过监控更改。

6.10.13 PWM：脉冲输出指令

梯形图： 									适用机型							
									影响标志位							
指令列表： PWM (S1) (S2) (D)									步长		7					
操作数		类型	适用软元件												变址	
S1	INT	常数	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	√	
S2	INT	常数	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	√	
D	BOOL			Y												

操作数说明

S1：指定脉冲宽度（ms）

可设定范围：0~32767（ms），当 **S1** 大于 32767 时，系统报指令操作数非法，同时不占系统硬件资源。

在指令运行过程中更改 **S1** 的内容，输出脉冲也随之发生变化。

S2：指定脉冲周期（ms）

可设定范围：1~32767，设定操作数不在本范围之内系统报指令操作数非法，同时脉冲不输出，也不占用系统资源。

在指令运行过程中更改 **S2** 的内容，输出脉冲也随之发生变化。

S2 要大于等于 **S1**，否则系统报操作数错误，不输出脉冲，也不占用系统资源。

D：高速脉冲输出点，只能指定 Y0 或 Y1。

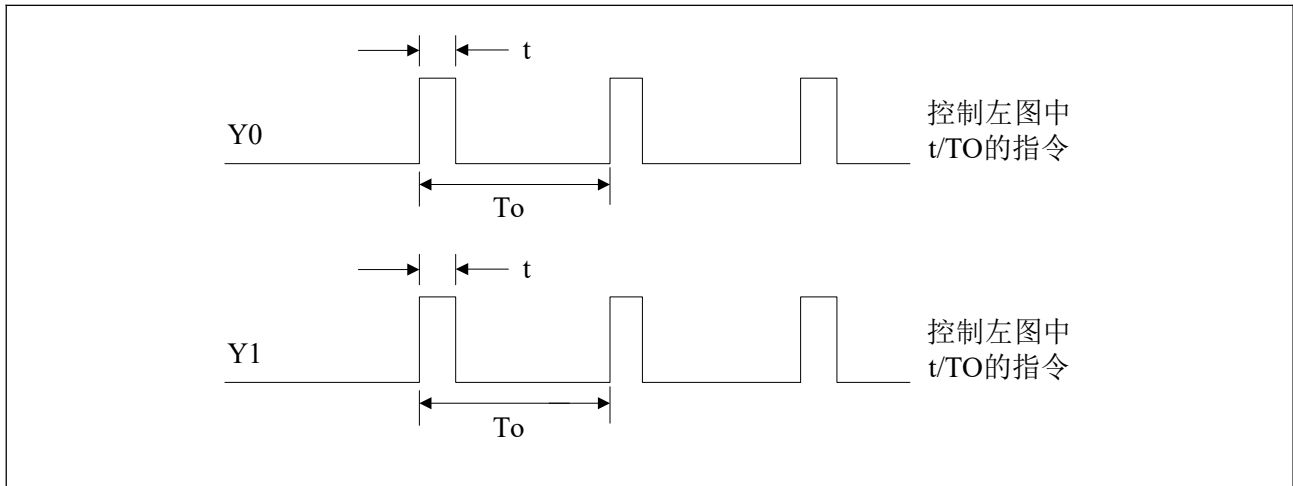
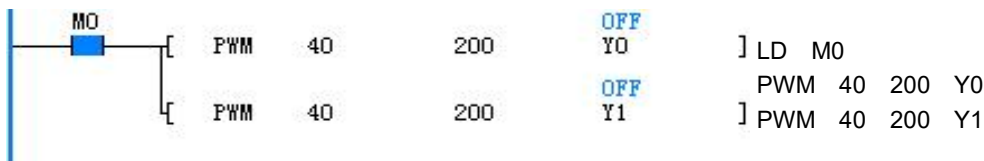
功能说明

在 **D** 所指定的端口输出宽度为 **S1**、周期为 **S2** 的 PWM 脉冲。

注意事项

1. 当 **S1** 为 0 时, Y0 或 Y1 端口输出一直为 OFF。当 **S1=S2** 时, Y0 或 Y1 端口输出一直为 ON。
2. 从输出端子 (Y0 和 PORT 0, Y1 和 PORT 1) 得到的波形跟用户的输出负载有关系, 在满足最大输出电流情况下, 负载越小, 输出波形越接近设定操作数。因此, 为了输出高速脉冲, PLC 的输出晶体管上的负载电流要大, 但不能超过额定负载电流。
3. 在高速指令有效运行 (包括输出完成) 时, 对同一端口的其它操作无效。只有在高速脉冲输出指令无效时, 其它指令才能操作本端口。
4. 使用 2 个 PWM 指令能够在 Y0 和 Y1 输出端得到各自独立的高速脉冲输出, 也可和 PLSY 或 PLSR 在不同的输出点 (Y0, Y1) 得到各自独立的高速脉冲输出。
5. 有多条 PWM、PLSY 或 PLSR 指令操作同一端口时, 先有效的指令控制端口输出状态, 后有效的指令对输出点的状态无影响。
6. 与其它高速指令 (DHSCS, DHSCR, DHSZ, DHSP, DHST, HCNT) 相同, PWM 指令要满足系统中对高速输入和高速脉冲输出的要求。

使用示例



其中 t 为脉冲宽度, T_0 为脉冲周期。

1. M0 为 ON 时, Y0、Y1 端口输出宽度为 40ms、周期为 200ms 的 PWM 脉冲。M0 为 OFF 时, 输出 OFF。输出状态不受扫描周期的影响。
2. SM80 对应 Y0 的输出禁止, SM81 对应 Y1 的输出禁止。SM80、SM81 为 ON 时, 输出中止。
3. SM82、SM83 对应 Y0 和 Y1 的输出标志, 当 M0 为 OFF 时, SM82、SM83 清为 OFF。

6.11 控制计算指令

6.11.1 PID：功能指令

<div>梯形图：</div> <div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div>[</div><div>PID</div><div>(S1)</div><div>(S2)</div><div>(S3)</div><div>(D)</div></div>									适用机型									
									影响标志位									
指令列表：PID(S1)(S2)(S3)(D)									步长		9							
操作数		类型		适用软元件													变址	
S1		INT								D							√	
S2		INT								D							√	
S3		INT								D							√	
D		INT								D							√	

操作数说明

D：执行程序时，输出运算结果（MV）

S1：设定目标值（SV）

S2：当前测量值（PV）

S3：采样时间（Ts）范围为 1~32767（ms），比运算周期短的时间数值无法执行。

S3+1：动作、报警及上下限功能设置字

位	设定值及其含义	
	0	1
0	正动作	逆动作
1	输入变化量报警无效	输入变化量报警有效
2	输出变化量报警无效	输出变化量报警有效
3~4	保留	
5	输出值上下限设定无效	输出值上下限设定有效
6~15	保留	

S3+2：输入滤波常数（α）范围 0~99[%]，为 0 时没有输入滤波功能。

S3+3：比例增益（Kp）范围 1~32767[%]。

S3+4：积分时间（TI）范围 0~32767（×100ms），为 0 时作为∞处理（无积分）。

S3+5：微分增益（KD）范围 0~100[%]，为 0 时无微分增益。

S3+6：微分时间（TD）范围 0~32767（×10ms），为 0 时无微分处理。

S3+7~S3+14：PID 运算内部数据存储寄存器。

S3+15：PID 输入变化量（增侧）报警设定值 0~32767（**S3+1** 的 BIT1=1 时）。

S3+16：PID 输入变化量（减侧）报警设定值 0~32767（**S3+1** 的 BIT1=1 时）。

S3+17：PID 输出变化量（增侧）报警设定值 0~32767（**S3+1** 的 BIT2=1 和 BIT5=0 时）。

输出上限设定值-32768~32767（**S3+1** 的 BIT2=0 和 BIT5=1 时）。

S3+18：PID 输出变化量（减侧）报警设定值 0~32767（**S3+1** 的 BIT2=1 和 BIT5=0 时）。

输出下限设定值-32768~32767 (**S3+1** 的 BIT2=0 和 BIT5=1 时)。

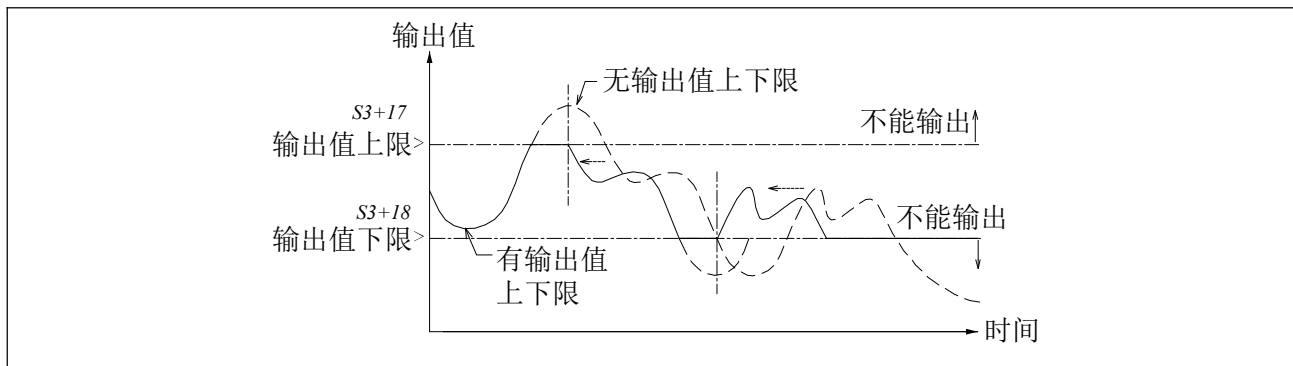
S3+19: PID 报警输出。

- BIT0: 输入变化量 (增侧) 溢出。
- BIT1: 输入变化量 (减侧) 溢出。
- BIT2: 输出变化量 (增侧) 溢出。
- BIT3: 输出变化量 (减侧) 溢出。

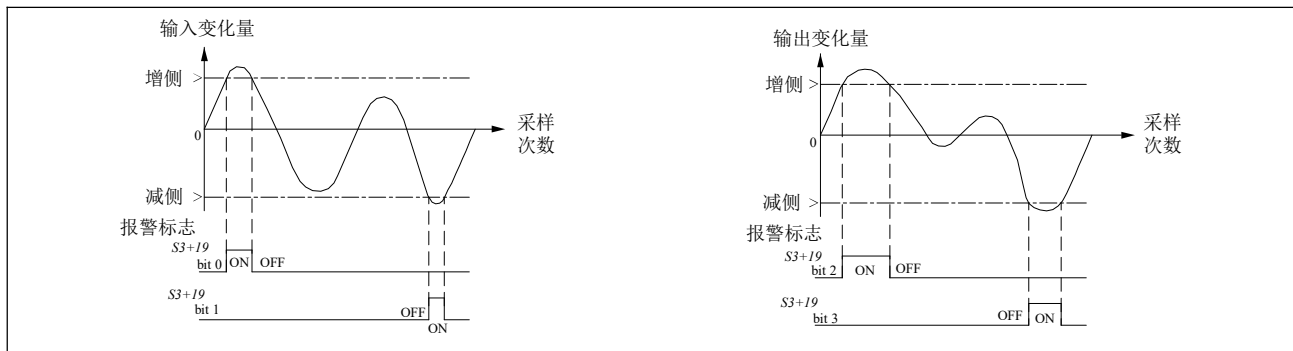
其中, **S3~S3+6** 是用户设定操作数。 **S3+15~S3+19** 是用户选择设定操作数。用户可采用后台软件中的 **PID** 指令向导设定各操作数。

功能说明

1. 当能流有效且达到采样时间时, 进行 **PID** 运算。
2. **PID** 指令可同时多次执行 (环路数目无限), 但要注意运算使用的 **S1**、**S2**、**S3** 或 **D** 软元件号不要重复覆盖。
3. **PID** 指令在定时中断子程序、一般子程序及主程序中可使用。在这种情况下, 执行 **PID** 指令前要先确认操作数设定单元及清除 **S3+7** 后的内部处理数据单元再使用。
4. 输入滤波常数有使测量值变化平滑的效果。
5. 微分增益有缓和输出值剧烈变化的效果。
6. 动作方向: 通过 **S3+1** 的 BIT0 来设定系统的正动作 (正反馈)、逆动作 (负反馈) 模式。
7. 输出上下限设定: 当设定输出上下限有效 (**S3+1** 的 BIT5=ON 且 BIT2=OFF) 时, 可以抑制 **PID** 控制的积分项过大。此时输出值如下图所示:



8. 报警设定: 当设定输出上下限有效 (**S3+1** 的 BIT1=ON, BIT2=ON 且 BIT5=OFF) 时, **PID** 指令将输入输出变化量与 **S3+15~S3+18** 单元里的设定值进行比较, 超过被设定的输入输出变化量时, **PID** 报警输出 **S3+19** 单元相应的各功能位在 **PID** 指令执行后立即被设置。由此用户可以进行输入变化量、输出变化量的监测。此时输出值如下图所示:



9. PID 指令的基本运算式：

动作方向	PID 运算公式
正动作	$\Delta MV = KP \left\{ (EV_n - EV_{n-1}) + \frac{T_s}{T_I} EV_n + D_n \right\}$ $EV_n = PV_{nf-1} - SV$ $D_n = \frac{T_D}{T_s + \alpha_D * T_D} (PV_{nf} + PV_{nf-2} - 2PV_{nf-1}) + \frac{\alpha_D * T_D}{T_s + \alpha_D * T_D} * D_{n-1}$ $MV_n = \sum \Delta MV$
逆动作	$\Delta MV = KP \left\{ (EV_n - EV_{n-1}) + \frac{T_s}{T_I} EV_n + D_n \right\}$ $EV_n = SV - PV_{nf-1}$ $D_n = \frac{T_D}{T_s + \alpha_D * T_D} (2PV_{nf-1} - PV_{nf} - PV_{nf-2}) + \frac{\alpha_D * T_D}{T_s + \alpha_D * T_D} * D_{n-1}$ $MV_n = \sum \Delta MV$

符号说明如下表所示：

符号	说明	符号	说明
EV_n	本次采样偏差	D_n	本次微分项
EV_{n-1}	1 个周期前的偏差	D_{n-1}	1 个周期前的微分项
SV	目标值	KP	比例增益
PV_{nf}	本次采样值（滤波后）	T_s	采样周期
PV_{nf-1}	1 个周期前采样值（滤波后）	T_I	积分时间
PV_{nf-2}	2 个周期前采样值（滤波后）	T_D	微分时间
ΔMV	输出变化量	α_D	微分增益
MV	本次的操作量		

使用示例

// PID 初始化程序，如果控制操作数不变，可以只执行一次

```
LD      SM1           //初始化程序只需运行一次
MOV     1000    D500   //设定目标值
MOV     500     D510   //采样时间 (Ts) 范围为 1~32767 (ms) 但比运算周期
                        //短的时间数值无法执行
MOV     7       D511   //动作方向
MOV     70     D512   //输入滤波常数 (α) 范围 0~99[%]为 0 时没有输入滤波功能
MOV     100    D513   //比例增益 (Kp) 范围 1~32767[%]
```

```
MOV    25      D514      //积分时间 (TI) 范围 0~32767 (×100ms) , 为 0 时作为
                               //∞处理 (无积分)

MOV    0        D515      //微分增益 (KD) 范围 0~100[%], 为 0 时无微分增益

MOV    63      D516      //微分时间 (TD) 范围 0~32767 (×10ms) , 为 0 时无微分
                               //处理

FMOV    0        D517      8  //清除 PID 运算中间数据存储区

MOV    2000     D525      //输入变化量 (增侧) 报警设定值 0~32767

MOV    2000     D526      //输入变化量 (减侧) 报警设定值 0~32767

MOV    2000     D527      //输出变化量 (增侧) 报警设定值 0~32767

MOV    2000     D528      //输出变化量 (减侧) 报警设定值 0~32767

//PID 指令执行运算

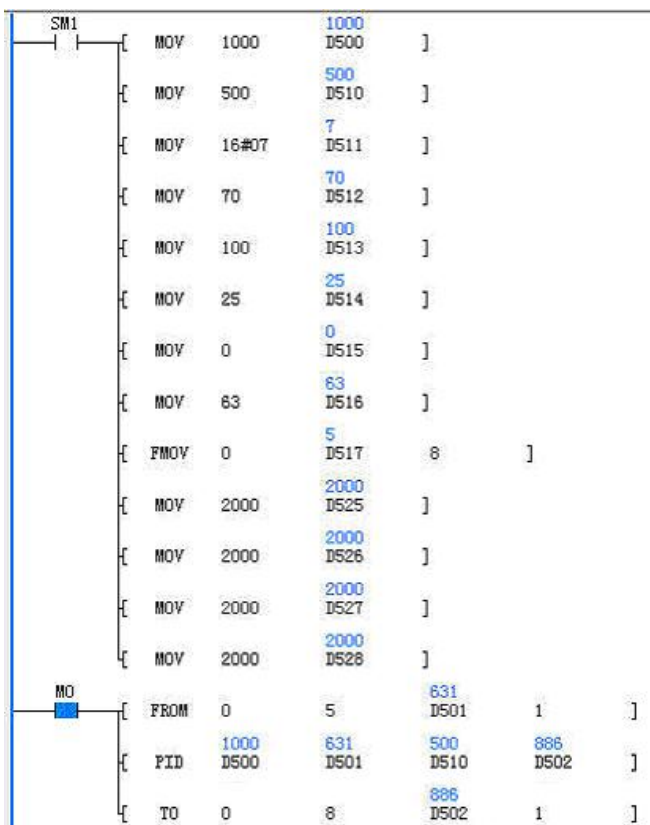
LD      M0                //用户控制 PID 运算程序

FROM    0    5    D501      1  //当前测量值读入 (用户可根据实际情况引入测量值)

PID      D500 D501 D510 D502  //PID 指令: PID S1 S2 S3 D

TO      0    8    D502      1  //PID 运算结果反馈到被控系统 (用户可根据实际情况
                               //处理 PID 运算结果)
```

以上指令的相关梯形图如下:



当主模块开始运行第一个扫描周期初始化 PID 各操作数，以后的扫描周期中不再初始化 PID 操作数。当 X2=ON 时，从外部 A/D 模块（实际应用中可以是其它方式）读入当前测量值，并填充到测量值单元。执行 PID 运算。将通过外部 D/A 模块（实际应用中可以是其它方式将运算结果转换成模拟信号，加入到被控系统当中。

注意事项

1. 对于 **D** 请指定在非停机保持的数据寄存器区。若指定在停机保持的数据寄存器区时，请在初次运行时，务必清 0 (LD SM0 MOV 0 D****)。
2. PID 指令需要占用 **S3** 起始的 20 个数据寄存器。
3. 采样时间 TS 的最大误差为 - (1 个扫描周期 + 1ms) ~ + (1 个扫描周期)。当 TS 值较小时，将影响 PID 效果，建议最好在定时中断中使用 PID 指令。
4. 当设置 PID 输出上下限有效时，如果上限值小于下限值时，系统报操作数错误，不执行 PID 运算。
5. 当设置输入、输出变化量报警有效时，**S3+15~S3+18** 单元的设定值不能为负值，否则系统报操作数错误，不执行 PID 运算。
6. 当 **S3+1** 的 BIT2 和 BIT5 同时为 ON 时，系统会认为设置无效（等同于 BIT2 和 BIT5 同时为 OFF），不进行上下限限制或变化量超值报警。
7. 当 PID 的控制操作数 (**S3~S3+6** 单元) 设定值不在有效范围内，系统会报操作数错误，不进行 PID 运算。
8. 采样时间小于等于 1 个扫描周期、运算过程中出现数据溢出、结果溢出等，不报警，PID 运算继续进行。
9. PID 指令首次执行前需要先将各操作数初始化，若运行过程中各操作数无变化，且控制操作数单元不会被其它程序覆盖，初始化程序可以只执行一次。PID 运算中间数据存储区中的数据若在 PID 运算过程中被改写，会导致运算结果有误。

6.11.2 RAMP：斜坡信号输出指令

梯形图： 									适用机型						
									影响标志位						
指令列表：RAMP (S1) (S2) (D1) (S3) (D2)									步长		12				
操作数	类型	适用软元件													变址
S1	INT	常数	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	√
S2	INT	常数	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	√
D1	INT								D				V		√
S3	INT	常数	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	√
D2	BOOL			Y	M	S	LM				C	T			

操作数说明

S1：起点值

S2：终点值

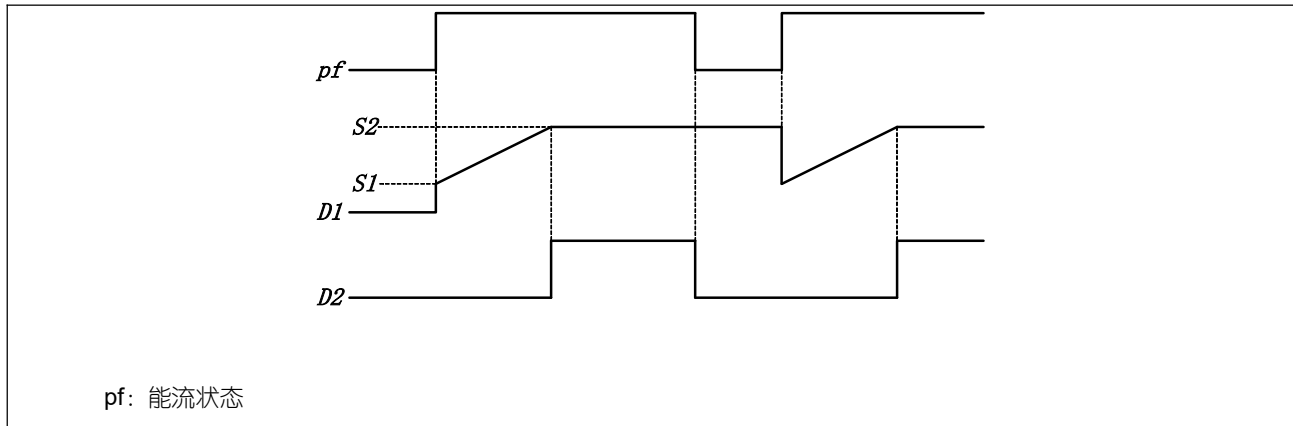
D1：输出值

S3：步数 (**S3**>0，否则报操作数错误，不执行运算)

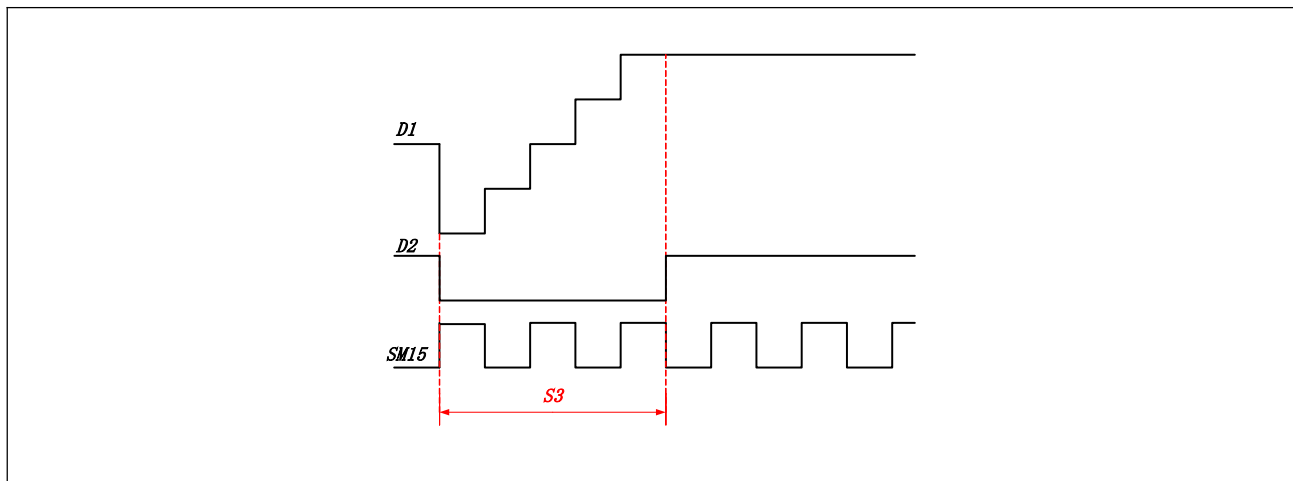
D2：输出状态 0

功能说明

当能流出现上升沿并保持 ON 时，每个扫描周期，按斜坡的高度和移动扫描次数，确定增量和当前输出值，到达 **S2** 后，输出值 (**D1**) 保持当前状态，并将状态输出位置成 ON。若能流出现下降沿，输出状态 (**D2**) 为 OFF，输出值 (**D1**) 保持当前状态，直到能流再次出现上升沿时，输出值 (**D1**) 初始化为 **S1** 的值，并继续产生下一次斜坡运算，如下图所示：



斜坡指令执行过程分解如下图 (**S3=5**)：



注意事项

1. 当计算步长不能整除时，采取“四舍五入”法
2. 每次上升沿来临，指令只会产生一次斜坡数据。
3. 当 **S1=S2** 时，**D1=S2**，**D2=ON**。
4. **RAMP**、**HACKLE**、**TRIANGLE** 指令在程序中的总数不能超过 100 条。

使用示例

//上电第一个扫描周期到来初始化寄存器

```
LD      SM1
MOV     0 D0
MOV     2000 D1
```

//X0=ON，执行斜坡函数指令

```
LD      X0
```

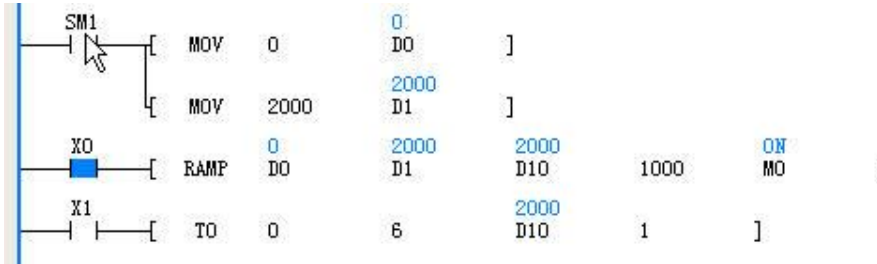
RAMP D0 D1 D10 1000 M0

//X1=ON, 将斜坡函数的输出值送到外部 DA 模块, 产生斜坡波形

LD X1

TO 0 6 D10 1

以上指令的梯形图如下:



1. 当 X0=ON 时, 每个扫描周期 D10 (第一个周期到来时 D10=D0=0) 增加 2 (2000/1000)。当 D10=D1=2000 时, D10 不再变化, 同时 M0=ON。斜坡函数产生过程中, 若能流出现下降沿, 输出状态 (D2) 为 OFF, 输出值 (D1) 保持当前状态, 直到下一个上升沿到来时, D10=D0, 重新开始一个新的斜坡过程。

2. 用户可以通过外部特殊模块将数据转换为模拟波形。

6.11.3 HACKLE: 锯齿波信号输出指令

<div>梯形图:</div> <div><div><div></div><div></div><div></div></div><div>HACKLE</div><div>(S1)</div><div>(S2)</div><div>(D1)</div><div>(S3)</div><div>(D2)</div><div>]</div></div>										适用机型						
										影响标志位						
指令列表: HACKLE (S1) (S2) (D1) (S3) (D2)										步长		12				
操作数		类型	适用软元件										变址			
S1		INT	常数	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	√
S2		INT	常数	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	√
D1		INT								D				V		√
S3		INT	常数	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	√
D2		BOOL			Y	M	S	LM				C	T			

操作数说明

S1: 起点值

S2: 终点值

D1: 输出值

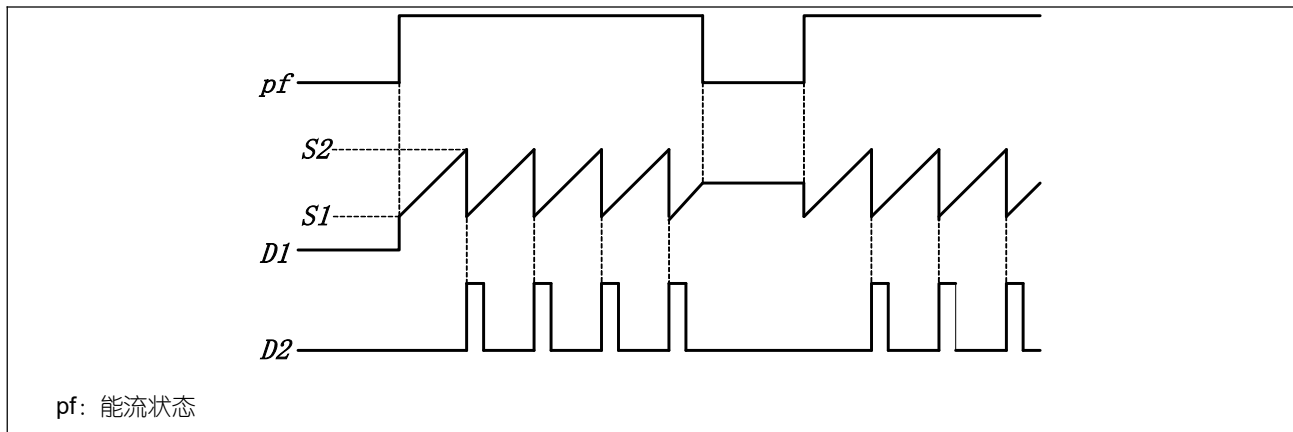
S3: 步数 (**S3**>0, 否则报操作数错误, 不执行运算)

D2: 输出状态

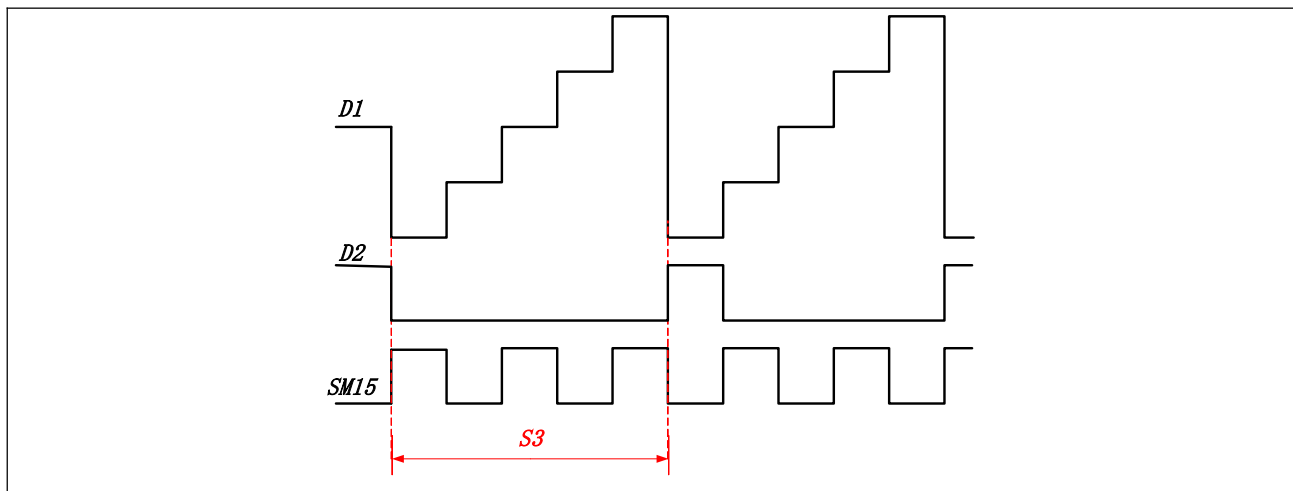
功能说明

当能流有效时, 每个扫描周期, 按锯齿波的高度和步数确定增量和当前输出值 (D1)。当输出值到达 S2 后, 初始化为 S1 值, 并将状态输出位 (D2) 置 ON。下一个扫描周期若能流持续为 ON, 将状态输出位 (D2) 置 OFF, 继续产生下一个锯齿波。

齿波。锯齿波函数产生过程中，若能流出现下降沿，输出状态（**D2**）为 OFF，输出值（**D1**）保持当前状态，直到能流再次出现上升沿时，输出值（**D1**）初始化为 **S1** 的值，并继续产生下一次锯齿波操作。如下图所示：



锯齿波指令执行过程分解如下图（**S3=5**）：



注意事项

1. 当计算步长不能整除时，采取“四舍五入”法
只要能流有效，指令会产生一系列连续的锯齿波数据
2. 当 **S1=S2** 时，**D1=S2**, **D2=ON**（不产生计数脉冲）
3. RAMP、HACKLE、TRIANGLE 指令在程序中的总数不能超过 100 条。

使用示例

//上电第一个扫描周期到来初始化寄存器

```
LD      SM1
```

```
MOV     0 D0
```

```
MOV2000 D1
```

//X0=ON，执行锯齿波函数指令

```
LD      X0
```

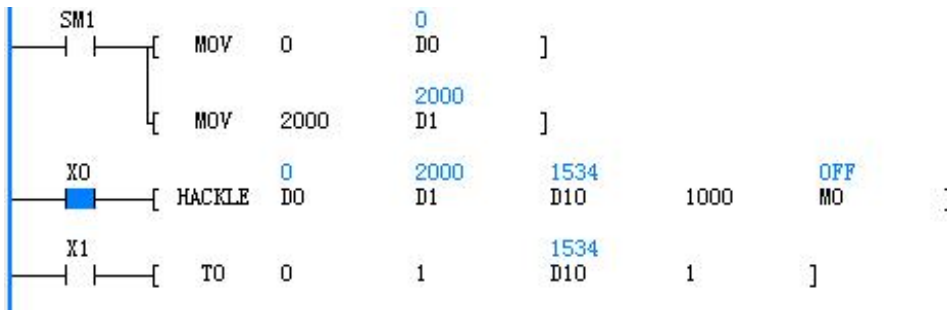
```
HACKLE  D0 D1 D10 1000 M0
```

//X1=ON，将斜坡函数的输出值送到外部 DA 模块，产生锯齿波形

```
LD      X1
```

TO 0 1 D10 1

以上指令的梯形图如下：



1. 当 X0=ON 时，每个扫描周期 D10 (第一个周期到来时 D10=D0=0) 增加 2 (2000/1000)。当 D10=D1=2000 时，M0=ON。下一个扫描周期，若 X0 保持 ON 状态，D10=D0=0，同时 M0=OFF，开始下一个锯齿波过程。若运行过程中，能流出现下降沿，则输出状态 (D2) 为 OFF，输出值 (D1) 保持当前状态，直到能流再次出现上升沿时，输出值 (D1) 初始化为 S1 的值，重新开始一个新的锯齿波过程。
2. 用户可以通过外部特殊模块将数据转换为模拟波形。

6.11.4 TRIANGLE：三角波信号输出指令

梯形图：									适用机型								
[TRIANGLE (S1) (S2) (D1) (S3) (D2)]									影响标志位								
指令列表：TRIANGLE (S1) (S2) (D1) (S3) (D2)									步长				12				
操作数		类型		适用软元件												变址	
S1		INT		常数	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	√
S2		INT		常数	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	√
D1		INT									D				V		√
S3		INT		常数	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	√
D2		BOOL				Y	M	S	LM				C	T			

操作数说明

S1：起点值

S2：终点值

D1：输出值

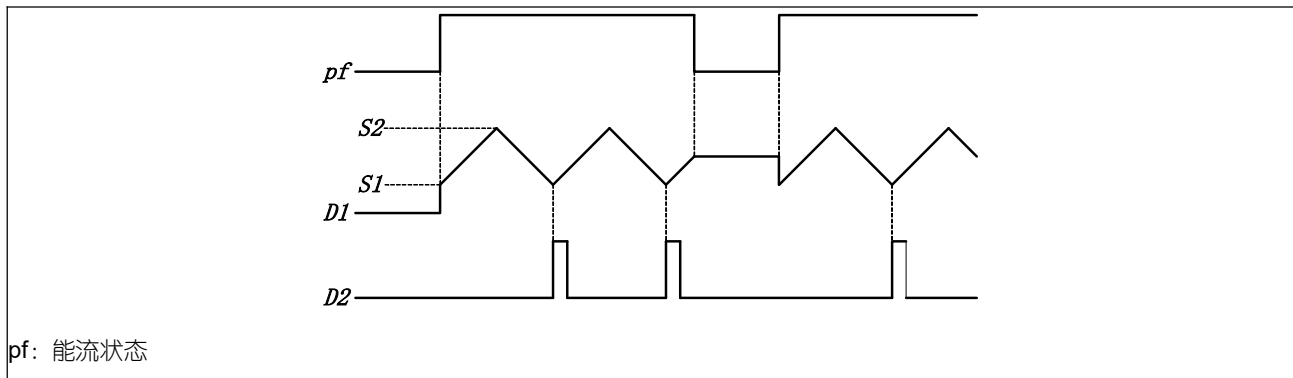
S3：步数 (S3>0，否则报操作数错误，不执行运算)

D2：输出状态

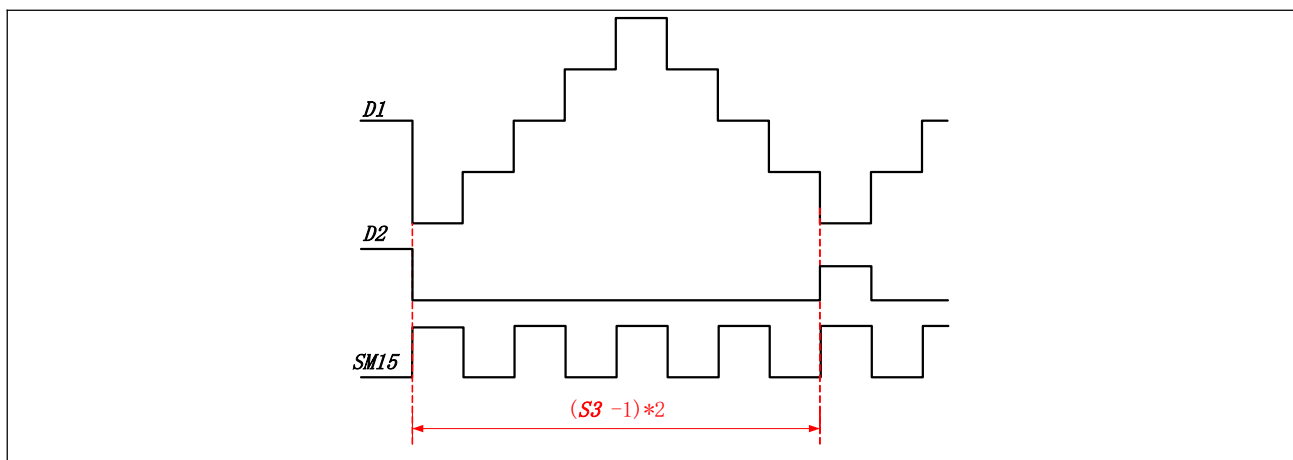
功能说明

当能流有效时，每个扫描周期，按三角波的高度和步数确定增量和当前输出值 (D1)。当输出值到达 S2 后，此时已完成三角波的前半坡，改变输出值的增量方向，继续产生后半坡。当输出值 (D1) 再次到达 S1 值时，将状态输出位 (D2) 置 ON。下一个扫描周期若能流持续为 ON，将状态输出位 (D2) 置 OFF，继续产生下一个三角波。三角波函数产生过程中，若能

流出现下降沿，输出状态（**D2**）为 OFF，输出值（**D1**）保持当前状态，直到能流再次出现上升沿时，输出值（**D1**）初始化为 **S1** 的值，重新开始一个新的三角波过程，如下图所示：



三角波指令执行过程分解如下图（**S3=5**）：



注意事项

1. 当计算步长不能整除时，采取“四舍五入”法。
2. 只要能流有效，指令会产生一系列连续的三角波数据。
3. 当 **S1=S2** 时，**D1=S2**，**D2=ON**（不产生计数脉冲）。
4. 三角波的周期 = $(S3 - 1) \times 2$ 。
5. RAMP、HACKLE、TRIANGLE 指令在程序中的总数不能超过 100 条。

使用示例

//上电第一个扫描周期到来初始化寄存器

```
LD      SM1
MOV     0 D0
MOV     2000 D1
```

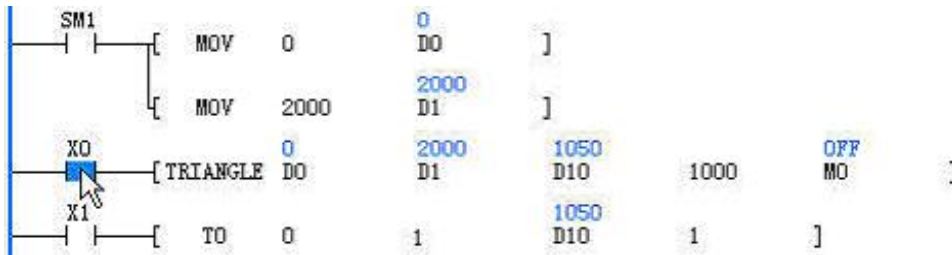
//X0=ON，执行三角波函数指令

```
LD      X0
TRIANGLE D0 D1 D10 1000 M0
```

//X1=ON，将斜坡函数的输出值送到外部 DA 模块，产生三角波形

```
LD      X1
TO      0 1 D10 1
```

以上指令的梯形图如下：



1. 当 X0=ON 时，每个扫描周期 D10（第一个周期到来时 D10=D0=0）增加 2（2000/1000）。当 D10=D1=2000 时，三角波半坡完成，此后每个扫描周期 D10 减 2。当 D10=D0=0 时，完整三角波完成，M0=ON。下一个扫描周期，若 X0 保持 ON 状态，M0=OFF，开始下一个三角波过程。若运行过程中，能流出现下降沿，则输出状态（D2）为 OFF，输出值（D1）保持当前状态，直到能流再次出现上升沿时，输出值（D1）初始化为 S1 的值，重新开始一个新的三角波过程。
2. 用户可以通过外部特殊模块将数据转换为模拟波形。

6.11.5 ABSD：凸轮绝对控制指令

无

6.11.6 DABSD：凸轮绝对控制指令

无

6.11.7 ALT：交替输出指令

无

6.12 通讯指令

6.12.1 Modbus：主站通讯指令

梯形图： 										适用机型	APLC-TCP				
指令列表：Modbus (S1) (S2) (S3)										影响标志位					
										步长	8				
操作数	类型	适用软元件													变址
S1	INT	常数													
S2	INT	D	V												
S3	INT	D													√

操作数说明

S1：指定的通讯通道

S2：发送数据起始地址

S3：接收数据起始地址

功能说明

1. 当作为主站，输入条件满足时，把从 **S2** 开始保存的数据发送出去，然后接收数据，并保存到 **S3** 开始的地址单元。
2. 当作为从站，接收和发送数据不需要指令控制。
3. 上升沿执行。

注意事项

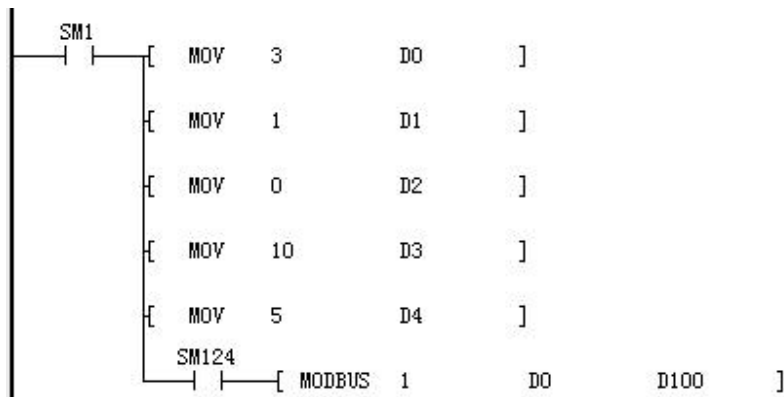
1. Modbus 发送数据，无论用户设置成 RTU 模式还是 ASCII 模式，都只需从 **S2** 开始保存 RTU 形式的数，不需要保存起始字符，结束字符和校验和。发送过程中，自动加上所需的起始字符，结束字符和校验和。
2. 发送的数据，不需设定发送的数据长度，系统会根据指令自动按系统内部设定长度进行发送，如下图所示：

S2	从站地址
S2+1	功能码
S2+2	数据1

S2+N+1	数据N

3. Modbus 的接收，无论用户设置成 RTU 模式还是 ASCII 模式，数据都是采用 RTU 形式存放，即用户设成 ASCII 模式，系统自动将其转换成十六进制，去掉起始字符、结束字符，保存在 **S3** 开始的数据区。
4. 发送和接收的数据都是保存在字元件的低字节，高字节没有使用。

使用示例



```
LD SM1
MOV 3 D0
MOV 1 D1
MOV 0 D2
MOV 10 D3
MOV 5 D4
AND SM124
Modbus 1 D0 D100
```

1. 将 Modbus 指令发送的数据放在 D0 开始的元件中。
2. 将接收到的数据保存到 D100 开始的元件中。
3. Modbus 接收到数据后，先进行 CRC 校验、地址校验、功能码校验，如果有错误，置错误标志（SM136），并将具体的错误信息记录在特殊寄存器 SD139 中。
4. SM114 和 SM124 串口空闲标志，也可以用在 MODBUS 中标识 MODBUS 的通讯状态。

Modbus 通讯错误代码表如下：

异常代码	异常代码意义
0x01	非法功能码
0x02	非法寄存器地址
0x03	数据个数错误
0x10	通讯超时，通讯时间超过了用户设定的通讯最大时间发生的错误
0x11	接收数据帧错误
0x12	操作数错误，设定操作数（模式或者主/从）错误
0x13	本身的站号与指令设定的站号相同，发生错误

详细应用方法举例请见第十章 通讯功能使用指南。

6.12.2 EVFWD：变频器正转指令

梯形图： 										适用机型	
指令列表：EVFWD (S1) (S2)										影响标志位	
操作数										步长	6
类型										适用软元件	变址
S1	INT	常数									
S2	WORD	常数	D	V							√

操作数说明

S1：指定的通讯通道（只支持通道 1）

S2：变频器的地址（支持广播方式，广播地址为 00，从站可设置地址为 1~247）

功能说明

1. 通过 Modbus 通讯协议控制变频器的正转。
2. 上升沿执行。

注意

Modbus 和与变频器通讯的指令总数不超过 128 个

使用示例



1. 设置串口 1，变频器地址#1,通过 Modbus 通讯协议控制变频器的正转。
2. 变频器接收到数据后，先进行 CRC 校验、地址校验、功能码校验，通讯完成，置通讯完成标志（SM135），如果有错误，置错误标志（SM136），并将具体的错误信息记录在特殊寄存器 SD139 中。

变频器指令通讯的错误代码表如下：

异常代码	异常代码意义
0x1	非法功能码。
0x2	非法寄存器地址。
0x3	数据错误，即数据超过上限或者下限。
0x4	从站操作失败，包括数据在上下限范围之内，但是数据无效引起的错误。
0x5	命令有效，正在处理中，主要应用在存储数据到非易失性存储中。
0x6	从站忙，请稍后再试，主要应用在存储数据到非易失性存储中。
0x18	信息帧错误：包括信息长度错误、校验错误。
0x20	参数不可修改
0x21	参数运行时不可修改（仅 EV3100 支持）
0x22	参数受密码保护

6.12.3 EVREV：变频器反转指令

梯形图： 										适用机型					
										影响标志位					
指令列表： EVREV (S1) (S2)										步长		6			
操作数		类型	适用软元件												变址
S1		INT	常数												
S2		WORD	常数	D	V										√

操作数说明

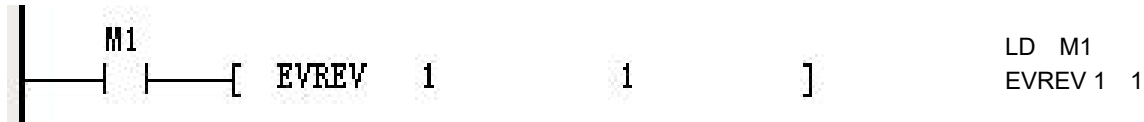
S1：指定的通讯通道（只支持通道 1）

S2：变频器的地址（支持广播方式，广播地址为 00，从站可设置地址为 1~247）

功能说明

1. 通过 Modbus 通讯协议控制变频器的反转。
2. 上升沿执行。

使用示例



1. 设置串口 1，变频器地址 #1，通过 Modbus 通讯协议控制变频器的反转。
2. 变频器接收到数据后，先进行 CRC 校验、地址校验、功能码校验，通讯完成，置通讯完成标志（SM135），如果有错误，置错误标志（SM136），并将具体的错误信息记录在特殊寄存器 SD139 中。

6.12.4 EVDFWD：变频器点动正转指令

梯形图： 									适用机型						
									影响标志位						
指令列表： EVDFWD (S1) (S2)									步长		6				
操作数		类型		适用软元件										变址	
S1		INT		常数											
S2		WORD		常数	D	V									√

操作数说明

S1：指定的通讯通道（只支持通道 1）

S2：变频器的地址（支持广播方式，广播地址为 00，从站可设置地址为 1~247）

功能说明

1. 通过 Modbus 通讯协议控制变频器的点动正转。
2. 上升沿执行。

使用示例



1. 设置串口 1，变频器地址#1,通过 Modbus 通讯协议控制变频器的点动正转。
2. 变频器接收到数据后，先进行 CRC 校验、地址校验、功能码校验，通讯完成，置通讯完成标志（SM135），如果有错误，置错误标志（SM136），并将具体的错误信息记录在特殊寄存器 SD139 中。

6.12.5 EVDREV：变频器点动反转指令

梯形图： —— ——[EVDREV (S1) (S2)]									适用机型						
									影响标志位						
指令列表： EVDREV (S1) (S2)									步长		6				
操作数	类型	适用软元件												变址	
S1	INT	常数													
S2	WORD	常数	D	V										√	

操作数说明

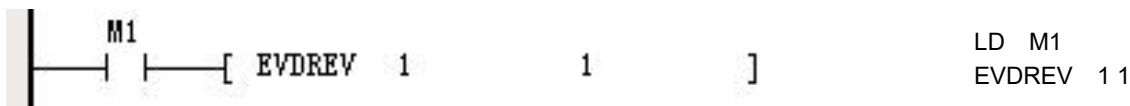
S1：指定的通讯通道（只支持通道 1）

S2：变频器的地址（支持广播方式，广播地址为 00，从站可设置地址为 1~247）

功能说明

1. 通过 Modbus 通讯协议控制变频器的点动反转。
2. 上升沿执行。

使用示例



1. 设置串口 1，变频器地址#1，过 Modbus 通讯协议控制变频器的点动反转。
2. 变频器接收到数据后，先进行 CRC 校验、地址校验、功能码校验，通讯完成，置通讯完成标志（SM135），如果有错误，置错误标志（SM136），并将具体的错误信息记录在特殊寄存器 SD139 中。

6.12.6 EVSTOP：变频器停止指令

梯形图： 										适用机型					
										影响标志位					
指令列表：EVSTOP (S1) (S2) (S3)										步长		8			
操作数		类型		适用软元件										变址	
S1		INT		常数											
S2		WORD		常数	D	V									√
S3		WORD		常数	D	V									√

操作数说明

S1：指定的通讯通道（只支持通道 1）

S2：变频器的地址（支持广播方式，广播地址为 00，从站可设置地址为 1~247）

S3：变频器的停止方式

停止方式有三种：方式 0 停车（按设定的减速时间停车）、方式 1 停车（自由停车）、方式 2 停车（最快减速时间停车）。

功能说明

1. 通过 Modbus 通讯协议控制变频器的停止。

2. 上升沿执行。

使用示例



1. 设置串口 1，变频器地址 #1，以及变频器的停止方式 0（按设定的减速时间停车），通过 Modbus 通讯协议控制变频器的停止。

2. 变频器接收到数据后，先进行 CRC 校验、地址校验、功能码校验，通讯完成，置通讯完成标志（SM135），如果有错误，置错误标志（SM136）并将具体的错误信息记录在特殊寄存器 SD139 中。

6.12.7 EVFRQ：设置变频器频率指令

梯形图： 										适用机型				
										影响标志位				
指令列表： EVFRQ (S1) (S2) (S3)										步长		8		
操作数	类型	适用软元件												变址
S1	INT	常数												
S2	WORD	常数	D	V										√
S3	WORD	常数	D	V										√

操作数说明

S1：指定的通讯通道（只支持通道 1）

S2：变频器的地址（支持广播方式，广播地址为 00，从站可设置地址为 1~247）

S3：变频器的频率

功能说明

1. 通过 Modbus 通讯协议控制变频器的运行频率。
2. 上升沿执行。

使用示例

	<pre>LD M1 EVFRQ 1 1 50</pre>
---	-------------------------------

1. 设置串口 1，变频器地址 #1，以及变频器的运行频率 50Hz，通过 Modbus 通讯协议控制变频器的运行频率。
2. 变频器接收到数据后，先进行 CRC 校验、地址校验、功能码校验，通讯完成，置通讯完成标志（SM135），如果有错误，置错误标志（SM136）并将具体的错误信息记录在特殊寄存器 SD139 中。

6.12.8 EVWRT：写单个寄存器值指令

梯形图： 									适用机型					
									影响标志位					
指令列表：EVWRT (S1) (S2) (S3) (S4)									步长		10			
操作数		类型	适用软元件										变址	
S1		INT	常数											
S2		WORD	常数	D	V									√
S3		WORD	常数	D	V									√
S4		WORD	常数	D	V									√

操作数说明

S1：指定的通讯通道（只支持通道 1）

S2：变频器的地址（支持广播方式，广播地址为 00，从站可设置地址为 1~247）

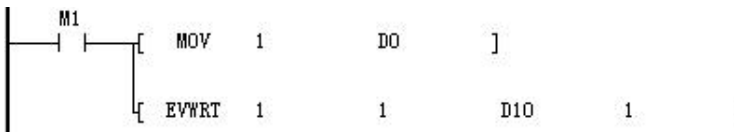
S3：寄存器地址

S4：寄存器值

功能说明

1. 通过 Modbus 通讯协议写入单个寄存器值。
2. 上升沿执行。

使用示例

	<pre>LD M1 MOV 1 D0 EVWRT 1 1 D10 1</pre>
---	---

1. 设置串口 1，变频器地址 #1，写入寄存器地址 1（数字频率控制），以及寄存器值 1（设定频率调电不存储），通过 Modbus 通讯协议写入单个寄存器值。

2. 变频器接收到数据后, 先进行 CRC 校验、地址校验、功能码校验, 通讯完成, 置通讯完成标志 (SM135), 如果有错误, 置错误标志 (SM136) 并将具体的错误信息记录在特殊寄存器 SD139 中。

6.12.9 EVRDST: 读取变频器状态指令

梯形图：										适用机型					
—— —— [EVRDST (S1) (S2) (S3) (D1)]										影响标志位					
指令列表： EVRDST (S1) (S2) (S3) (D1)										步长		10			
操作数	类型	适用软元件												变址	
S1	INT	常数													
S2	WORD	常数	D	V										√	
S3	WORD	常数	D	V										√	
D1	WORD	D												√	

操作数说明

S1: 指定的通讯通道 (只支持通道 1)

S2: 变频器的地址 (不支持广播方式, 从站可设置地址为 1~247)

S3: 读取状态信息选择

0: 运行状态字。1: 当前主设定的实际运行值。2: 变频器机型。3: 输出电流。4: 输出电压。5: 运行转速。6: 运行故障信息。

D1: 返回的状态存储地址

功能说明

1. 通过 Modbus 通讯协议读取变频器状态。
2. 上升沿执行。

使用示例

	LD M1 EVDRST 1 1 1 D0
--	--------------------------

1. 设置串口 1, 变频器地址 #1, 读取状态信息选择 1 (当前主设定的实际运行值), 返回的状态存储地址 D0, 通过 Modbus 通讯协议读取变频器状态。

2. 变频器接收到数据后, 先进行 CRC 校验、地址校验、功能码校验, 通讯完成, 置通讯完成标志 (SM135), 如果有错误, 置错误标志 (SM136) 并将具体的错误信息记录在特殊寄存器 SD139 中。

6.12.10 EVRD：读取变频器单个寄存器值指令

梯形图：										适用机型					
— — —[EVRD (S1) (S2) (S3) (D1)]										影响标志位					
指令列表：EV RD (S1) (S2) (S3) (D1)										步长		10			
操作数	类型	适用软元件													变址
S1	INT	常数													
S2	WORD	常数	D	V										√	
S3	WORD	常数	D	V										√	
D1	WORD	D												√	

操作数说明

S1：指定的通讯通道（只支持通道 1）

S2：变频器的地址（不支持广播方式，从站可设置地址为 1~247）

S3：读取寄存器地址

D1：返回值存储地址

功能说明

1. 通过 Modbus 通讯协议读取变频器单个寄存器值。
2. 上升沿执行。

使用示例

										<pre>LD M1 MOV 2 D10 EV RD 1 1 D10 D20</pre>				
--	--	--	--	--	--	--	--	--	--	--	--	--	--	--

1. 设置串口 1，变频器地址 #1，读取寄存器地址 2（变频器初始设定频率），返回值存储地址 D20，通过 Modbus 通讯协议读取变频器单个寄存器值。
2. 变频器接收到数据后，先进行 CRC 校验、地址校验、功能码校验，通讯完成，置通讯完成标志（SM135），如果有错误，置错误标志（SM136）并将具体的错误信息记录在特殊寄存器 SD139 中。

6.12.11 XMT：自由口发送指令

梯形图： 										适用机型	APLC-TCP				
指令列表：XMT (S1) (S2) (S3)										影响标志位					
										步长	7				
操作数	类型	适用软元件												变址	
S1	INT	常数													
S2	WORD	D	V												
S3	INT	常数	KnX	KnY	KnM	KnS	KnLM			D	SD	C	T	V	Z

操作数说明

S1：指定的通讯通道，取值范围为 0, 1

S2：发送数据起始地址

S3：发送的字节数

功能说明

当能流导通，且通讯条件满足时，按照用户指定的通道和地址发送数据。

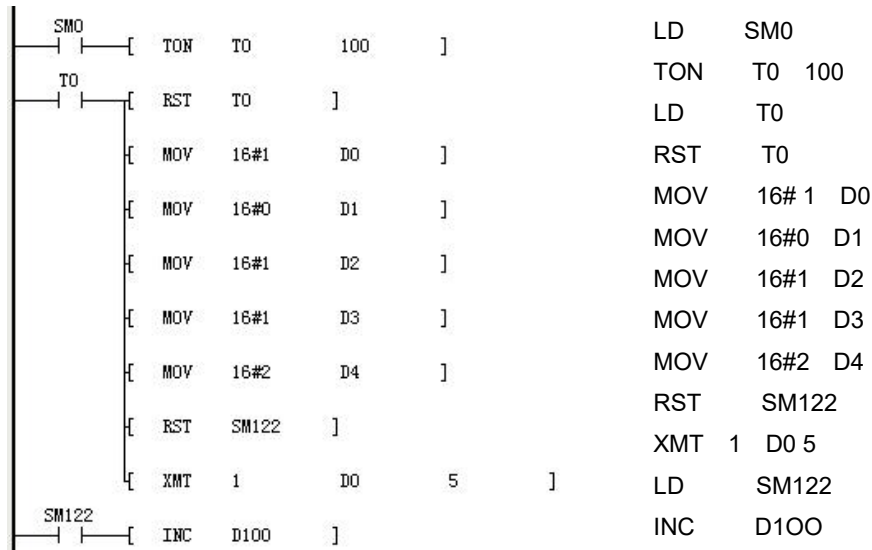
注意事项

1. 通讯帧的大小：通讯帧根据所选用的元件类型（D 或者 V）的不同，发送帧的结束字符不超过 D7999 或者 V63。
2. 停机的情况下，发送中止。

特殊寄存器

1. SM110/SM120：发送使能标志，当使用 XMT 指令时该位被置位，当发送结束后清除该位。当该位清零时，当前发送终止。
2. SM112/SM122：发送完成标志，当判断发送完成的时候，发送完成标志置位
3. SM114/SM124：空闲标志，当串口没有通讯任务的时候，置位。可以作为通讯的检测位。
4. 详细应用方法举例请见第十章 通讯功能使用指南。

使用示例



例程是每隔 10s 发送一帧数据。

利用串口 1 发送如下数据：

01	00	01	01	02
----	----	----	----	----

1. 首先，在系统块中将通讯口 1 设置为自由口，然后设定波特率、奇偶校验、数据位、停止位等。
2. 将要发送的数据写到发送缓存区内。
3. 发送数据之前先清发送完成标志（SM122）。
4. 当发送完成，发送完成标志（SM122）置位。

6.12.12 RCV：自由口接收指令

梯形图： 										适用机型	APLC-TCP				
										影响标志位					
指令列表：RCV (S1) (D) (S2)										步长	7				
操作数	类型	适用软元件												变址	
S1	INT	常数													
D	WORD	D	V												
S2	INT	常数	KnX	KnY	KnM	KnS	KnLM		D	SD	C	T	V	Z	

操作数说明

S1：指定的通讯通道，参数只支持 0，1

D：存放接收数据的起始地址

S2：接收的最大字节数

功能说明

当能流导通，且通讯条件满足时，按照用户指定的通道和地址接收数据。

注意事项

1. 通讯帧的大小：通讯帧根据所选用的元件类型（D 或者 V）的不同，接收帧的结束字符不超过 D7999 或者 V63。
2. 停机时，接收中止。
3. **S1** 取值范围为 0 和 1。

使用示例

										LD SM1 RCV 1 D20 5 LD SM123 INC D100					
---	--	--	--	--	--	--	--	--	--	---	--	--	--	--	--

1. 当能流导通的时候，RCV 指令将持续有效，如果只想接收一次，可以采用上升沿，或 SM1 等一次有效的特殊寄存器作能流输入。
2. 详细应用方法举例请见第十章 通讯功能使用指南。

特殊寄存器

SM111 (SM121) : 接收使能标志, 当使用 RCV 指令时该位被置位, 当接收结束后清除该位。当该位清零时, 当前接收终止。

SM113 (SM123) : 接收完成标志, 当接收完成, 接收完成标志置位。

SM114 (SM124) : 空闲标志, 当串口没有通讯任务的时候置位, 可以作为通讯的检测位。

SD111 (SD121) : 开始字符, 可以在系统块中设置。

SD112 (SD122) : 结束字符, 可以在系统块中设置。

SD113 (SD123) : 字符间超时时间, 也就是接收两个字符间的最大间隔时间, 可以在系统块中设置。

SD114 (SD124) : 帧超时时间, 从能流开始接通, 到接收结束的时间, 可以在系统块中设置。

SD115 (SD125) : 接收完成信息代码, 数据位定义如下:

用户终止接收标志	收到指定结束标志	收到最大字符数标志	字符间超时标志	(帧)接收超时标志	奇偶检验错误标志	保留, 用户可忽略
第 0 位	第 1 位	第 2 位	第 3 位	第 4 位	第 5 位	第 6~15 位

SD116 (SD126) : 当前收到的字符。

SD117 (SD127) : 当前收到的字符总数

6.12.13 MODRW: MODBUS 读写指令

无

6.13 校验指令

6.13.1 CCITT：校验指令

梯形图： 									适用机型	APLC-TCP					
									影响标志位						
指令列表：CCITT (S1) (S2) (D)									步长	7					
操作数	类型	适用软元件													变址
S1	WORD								D					V	√
S2	INT	常数	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	√
D	WORD								D					V	√

操作数说明

S1：待校验数据的起始单元

S2：待校验数据的数量（**S2**≥0，否则报操作数错误）

D：校验结果

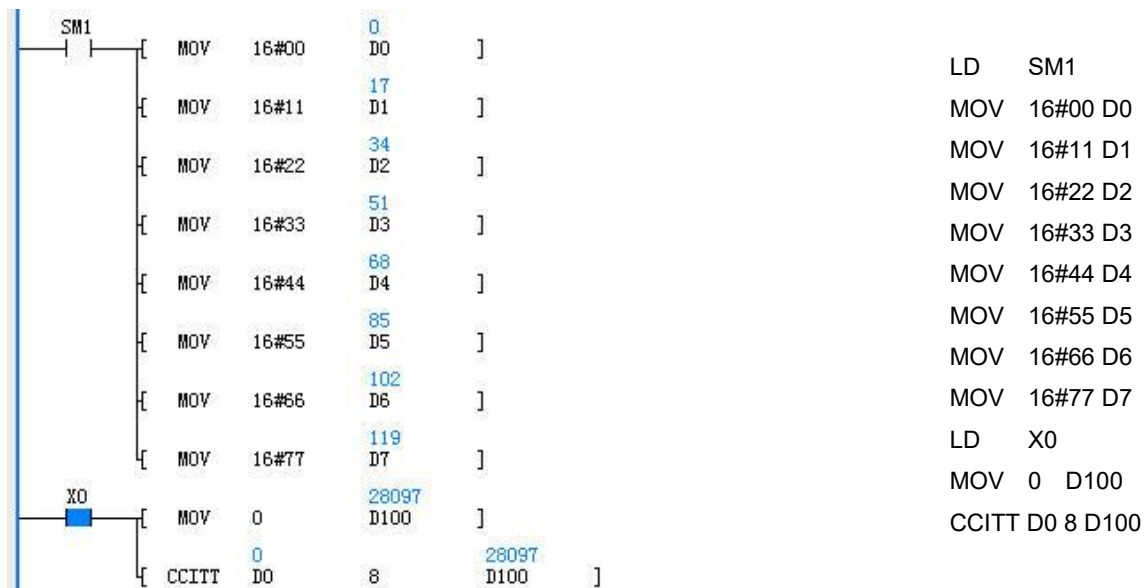
功能说明

- 将起始单元（**S1**）开始的 **S2** 个数据，进行 CCITT 校验运算，结果赋给 **D** 单元。
- CCITT 校验算法的多项式为： $X^{16} + X^{12} + X^5 + 1$

注意事项

- 在每次执行指令时，系统会将指令执行前 **D** 的内容带入运算，所以执行前要初始化 **D**。
- S2** 单元起始的校验数据区内的数据存储，默认为字节模式，即高字节被忽略为零，校验结果为 16 位。

使用示例



当 X0=ON 时，将 D0 起始的 8 个数据，进行 CCITT 校验运算，结果赋给 D100。

6.13.2 CRC16：校验指令

梯形图： 									适用机型	APLC-TCP					
指令列表：CRC16 (S1) (S2) (D)									影响标志位						
									步长	7					
操作数	类型	适用软元件												变址	
S1	WORD								D				V		√
S2	INT	常数	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	√
D	WORD								D				V		√

操作数说明

S1：待校验数据的起始单元

S2：待校验数据的数量（**S2**≥0，否则报操作数错误）

D：校验结果

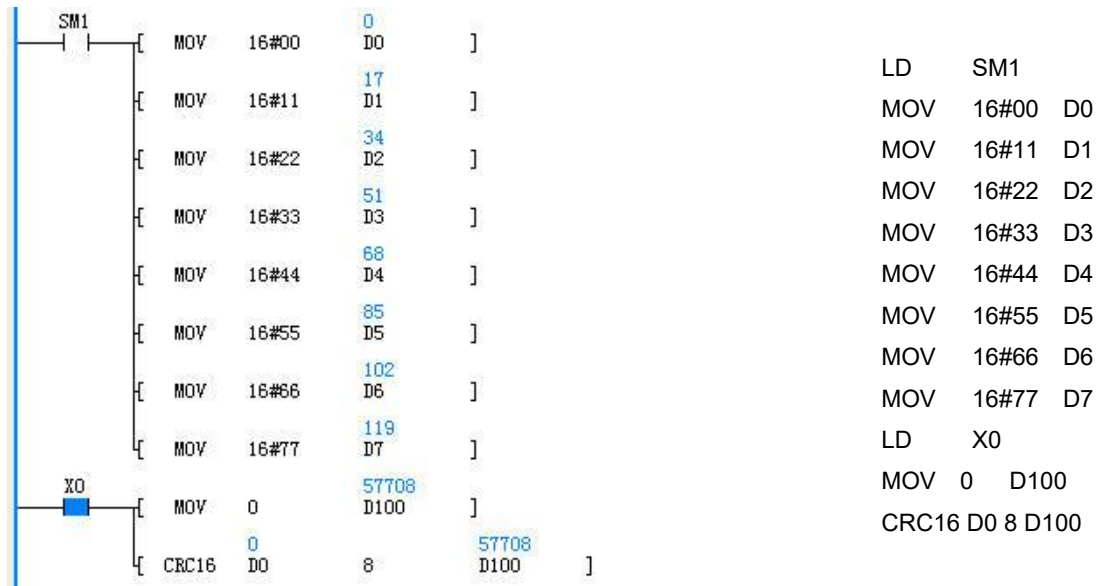
功能说明

1. 起始单元（**S1**）开始的 **S2** 个数据，进行 CRC16 校验运算，结果赋给 **D** 单元。
2. CRC16 校验算法的多项式为： $X^{16} + X^{15} + X^2 + 1$

注意事项

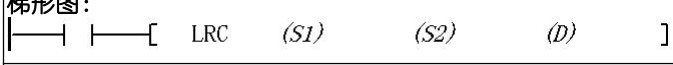
1. **D** 内容在每次执行指令时会将指令执行前的内容带入运算，执行前要初始化 **D**。
2. 如果使用标准 Modbus CRC 校验时，请给 **D** 元件（校验和）初始值赋 16#FFFF。且高低字节（高 8 位，低 8 位）需要交换。
3. **S2** 单元起始的校验数据区内的数据存储，默认为字节模式，即高字节被忽略为零，校验结果为 16 位。

使用示例



当 X0=ON 时，将 D0 起始的 8 个数据，进行 CRC16 校验运算，结果赋给 D100。

6.13.3 LRC：校验指令

梯形图： 										适用机型	APLC-TCP				
指令列表： LRC (S1) (S2) (D)										影响标志位					
步长										7					
操作数	类型	适用软元件													变址
S1	WORD								D				V		√
S2	INT	常数	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	√
D	WORD								D				V		√

操作数说明

S1：待校验数据的起始单元

S2：待校验数据的数量，（**S2**≥0，否则报操作数错误）

D：校验结果

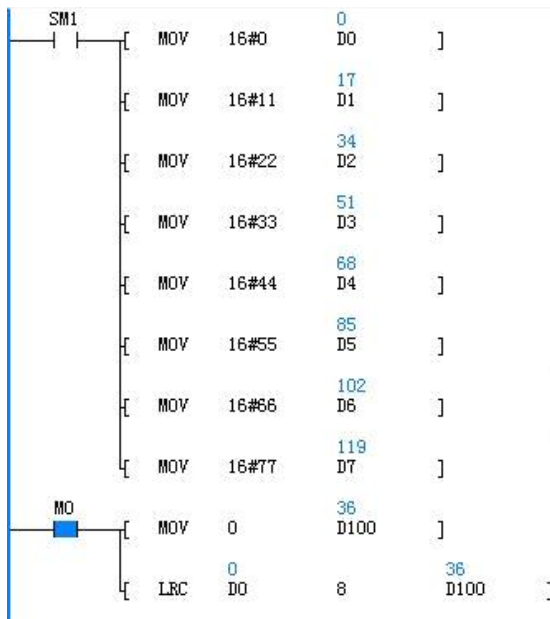
功能说明

将起始单元（**S1**）开始的 **S2** 个数据，进行 LRC 校验运算，结果赋给 **D** 单元。

注意事项

1. **D** 内容在每次执行指令时会将指令执行前的内容带入运算，执行前要初始化 **D**。
2. **S2** 单元起始的校验数据区内的数据存储，默认为字节模式，即高字节被忽略为零，校验结果为 8 位，存放在 **D** 的低字节。

使用示例



```

LD      SM1
MOV     16#00 D0
MOV     16#11 D1
MOV     16#22 D2
MOV     16#33 D3
MOV     16#44 D4
MOV     16#55 D5
MOV     16#66 D6
MOV     16#77 D7
LD      M0
MOV     0    D100
LRC     D0 8 D100

```

当 X0=ON 时，将 D0 起始的 8 个单元的数据，进行 LRC 校验运算后的结果赋给 D100。

6.14 增强型位处理指令

6.14.1 ZRST：批量位清零指令

梯形图： 										适用机型	APLC-TCP				
指令列表：ZRST (D) (S)										影响标志位					
										步长	5				
操作数	类型	适用软元件												变址	
D	BOOL			Y	M	S	LM				C	T			√
S	INT	常数	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	√

操作数说明

D：目的操作数

S：源操作数

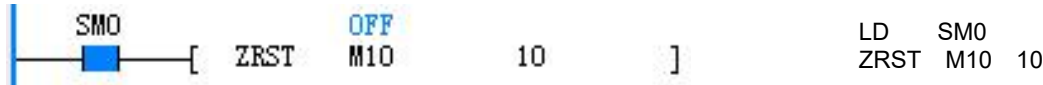
功能说明

当能流有效时，将 **D** 单元开始的 **S** 个连续位元件单元清零。

注意事项

1. 当被清除的位元件为 **C** 时，**C** 元件中的计数器值也将被清零。
2. 当被清除的位元件为 **T** 时，**T** 元件中的计时值也将被清零。

使用示例



当 **SM0=ON** 时，将 **M10** 开始的 **M10**、**M11**、**M12**...**M19** 这 10 个单元的数据全部清零。

6.14.2 ZSET：批量位置位指令

梯形图： 										适用机型	APLC-TCP				
指令列表：ZSET (D) (S)										影响标志位					
										步长	5				
操作数	类型	适用软元件												变址	
D	BOOL			Y	M	S	LM				C	T			√
S	INT	常数	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	√

操作数说明

D：目的操作数

S：源操作数

功能说明

当能流有效时，将 **D** 开始的 **S** 个连续位元件置 1。

使用示例



当 SMO=ON 时，将 M10 开始的 M10、M11、M12...M19 这 10 个单元的数据全部置 1。

6.14.3 DECO：解码指令

梯形图：									适用机型	APLC-TCP					
— — [DECO (S) (D)]									影响标志位						
指令列表：DATO (S) (D)									步长	5					
操作数	类型	适用软元件													变址
S	WORD	常数	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	√
D	INT			KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	√

操作数说明

S：源操作数

D：目的操作数

功能说明

当能流有效时，将字元件 **D** 中的第 **S** 位置为 1，其他位清为 0。

注意事项

1. **S** 的有效范围为 0~15。
2. **S** 大于 15 或小于 0 时，能流有效时，**D** 的值不会更改，但会报出指令操作数数值错误。

使用示例



能流有效时，将 D9 中的第 2 位置为 1，其他位清零。

6.14.4 ENCO：编码指令

梯形图：									适用机型	APLC-TCP					
— — [ENCO (S) (D)]									影响标志位						
指令列表：ENCO (S) (D)									步长	5					
操作数	类型	适用软元件													变址
S	INT	常数	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	√
D	INT			KnY	KnM	KnS	KnLM		D	SD	C	T	V	Z	√

操作数说明

S: 源操作数;

D: 目的操作数

功能说明

当能流有效时，字元件 **S** 中为“1”位的位编号，将被写入 **D** 中。

注意事项

当 **S** 中有多个位都出现“1”时，最小的位置编号将写入 **D** 中。如下图示例：



使用示例



当能流有效时，操作数 1 中为 2#0010，第 1 位为“1”，所以结果为 1，写入 D0。

6.14.5 BITS: 字中 ON 位统计指令

梯形图:									适用机型	APLC-TCP					
— [BITS (S) (D)]									影响标志位						
指令列表: BITS (S) (D)									步长	5					
操作数	类型	适用软元件													变址
S	INT	常数	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	√
D	INT	常数	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	√

操作数说明

S: 源操作数; **D**: 目的操作数

功能说明

当能流有效时，统计操作数 **S** 中为 1 位的个数，统计结果存入操作数 **D** 中。

使用示例



能流有效，BITS 指令中 **S** 为常数 16#F0F0，有 8 个位为“1”（ON 状态），计算结果为 8 存入 **D** (D1) 中。

6.14.6 DBITS：双字中 ON 位统计指令

梯形图： 									适用机型	APLC-TCP					
指令列表：DBITS (S) (D)									影响标志位						
									步长	6					
操作数	类型	适用软元件												变址	
S	DWORD	常数	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C		V		√
D	INT			KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	√

操作数说明

S：源操作数；**D**：目的操作数

功能说明

当能流有效时，统计双字 **S** 中为“1”位的个数，统计结果存入 **D** 中。

使用示例



能流有效，DBITS 指令中 **S** 为常数 16#FF0FF，有 16 个位为“1”（ON 状态），计算结果为 16，存入 **D**（D10）中。

6.15 字触点指令

6.15.1 BLD：字位触点 LD 指令

梯形图： 									适用机型	APLC-TCP					
指令列表：BLD (S1) (S2)									影响标志位						
									步长	5					
操作数	类型	适用软元件												变址	
S1	WORD		KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	√
S2	INT	常数	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	√

操作数说明

S1：源操作数

S2：指定位， $0 \leq S2 \leq 15$ ，否则报操作数错误。

功能说明

取 **S1** 单元内容的第 **S2** 位的状态用于驱动后段运算。

使用示例



取 D0（1000：2#0000001111101000）BIT5 的状态（ON）决定后段元件 Y0 的状态。

6.15.2 BLDI：字位触点 LDI 指令

梯形图： 										适用机型	APLC-TCP				
指令列表： BLDI (S1) (S2)										影响标志位					
										步长	5				
操作数	类型	适用软元件													变址
S1	WORD		KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	√
S2	INT	常数	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	√

操作数说明

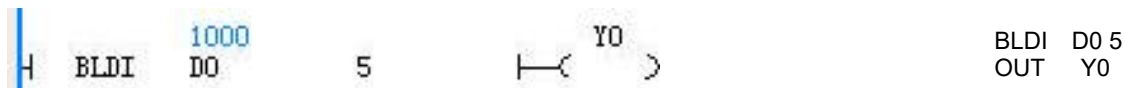
S1：源操作数

S2：指定位， $0 \leq S2 \leq 15$ ，否则报操作数错误。

功能说明

取 **S1** 单元内容的第 **S2** 位状态的逻辑非用于驱动后段运算。

使用示例



取 D0 (1000: 2#000000111101000) BIT5 状态 (ON) 的逻辑非 (OFF) 决定后段元件 Y0 的输出状态。

6.15.3 BAND：字位触点 AND 指令

梯形图 ^注 ： 										适用机型	APLC-TCP				
注：因为逻辑关系已经在图形里得到了体现，所以 BAND 指令在梯形图中显示为 BLD										影响标志位					
指令列表： BAND (S1) (S2)										步长	5				
操作数	类型	适用软元件													变址
S1	WORD		KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	√
S2	INT	常数	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	√

操作数说明

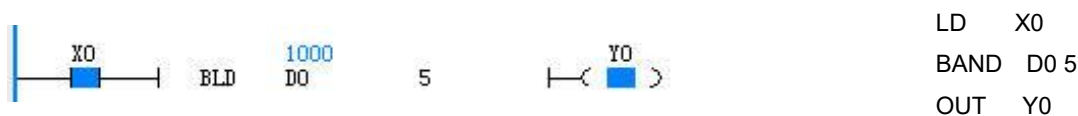
S1：源操作数

S2：指定位 ($0 \leq S2 \leq 15$ ，否则报操作数错误)

功能说明

取 **S1** 单元内容的第 **S2** 位的状态与其他节点串联用于驱动后段运算。

使用示例



取 D0 (1000: 2#000000111101000) BIT5 的状态 (ON) 与其他节点 (X0=ON) 串联决定后段元件 Y0 的输出状态。

6.15.4 BANI：字位触点 ANI 指令

梯形图 [※] : 										适用机型	APLC-TCP				
注：因为逻辑关系已经在图形里得到了体现，所以 BANI 指令在梯形图里显示为 BLDI										影响标志位					
指令列表：BANI (S1) (S2)										步长	5				
操作数	类型	适用软元件													变址
S1	WORD		KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	√
S2	INT	常数	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	√

操作数说明

S1：源操作数

S2：指定位 ($0 \leq S2 \leq 15$ ，否则报操作数错误)

功能说明

取 **S1** 单元内容的第 **S2** 位状态的逻辑非与其他节点串联用于驱动后段运算。

使用示例



```
LD    X0
BANI  D0 5
OUT   Y0
```

取 D0 (1000: 2#0000001111101000) BIT5 状态 (ON) 的逻辑非 (OFF) 与其他节点 (X0=ON) 串联决定后段元件 Y0 的输出状态。

6.15.5 BOR：字位触点 OR 指令

梯形图 [※] : 										适用机型	APLC-TCP				
注：因为逻辑关系已经在图形里得到了体现，所以 BOR 指令在梯形图里显示为 BLD										影响标志位					
指令列表：BOR (S1) (S2)										步长	5				
操作数	类型	适用软元件													变址
S1	WORD		KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	√
S2	INT	常数	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	√

操作数说明

S1：源操作数

S2：指定位 ($0 \leq S2 \leq 15$ ，否则报操作数错误)

功能说明

取 **S1** 单元内容的第 **S2** 位的状态与其他节点并联用于驱动后段运算。

使用示例



```
LD    X0
BOR   D0 5
OUT   Y0
```

取 D0 (1000: 2#000000111101000) BIT5 的状态 (ON) 与其他节点 (X0=ON) 并联决定后段元件 Y0 的输出状态。

6.15.6 BORI: 字位触点 ORI 指令

梯形图 [※] : 		适用机型	APLC-TCP												
		影响标志位													
指令列表: BORI (S1) (S2)		步长	5												
操作数	类型	适用软元件													变址
S1	WORD		KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	√
S2	INT	常数	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	√

操作数说明

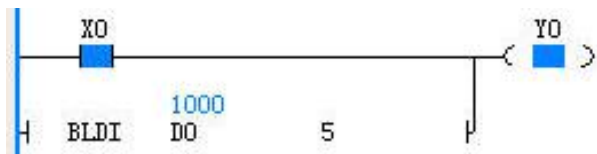
S1: 源操作数

S2: 指定位 ($0 \leq S2 \leq 15$, 否则报操作数错误)

功能说明

取 **S1** 单元内容的第 **S2** 位状态的逻辑非与其他节点并联用于驱动后段运算。

使用示例



```
LD    X0
BORI  D0 5
OUT   Y0
```

取 D0 (1000: 2#000000111101000) BIT5 状态 (ON) 的逻辑非 (OFF) 与其他节点 (X0=ON) 并联决定后段元件 Y0 的输出状态。

6.15.7 BOUT：字位线圈输出指令

梯形图： 										适用机型	APLC-TCP				
指令列表：BOUT (D) (S)										影响标志位					
										步长	5				
操作数	类型	适用软元件													变址
D	WORD			KnY	KnM	KnS	KnLM		D		C	T	V	Z	√
S	INT	常数	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	√

操作数说明

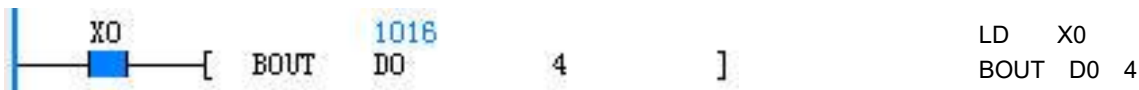
S1：源操作数

S2：指定位（ $0 \leq S2 \leq 15$ ，否则报操作数错误）

功能说明

将当前能流状态赋予 **D** 的 **S** 位。

使用示例



将当前能流状态（X0=ON）赋予 D0（1000：2#0000001111101000）的 BIT4。执行后 D0=1016（2#000000111111000）

6.15.8 BSET：字位线圈置位指令

梯形图： 										适用机型	APLC-TCP				
指令列表：BSET (D) (S)										影响标志位					
										步长	5				
操作数	类型	适用软元件													变址
D	WORD			KnY	KnM	KnS	KnLM		D		C	T	V	Z	√
S	INT	常数	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	√

操作数说明

D：目的操作数

S2：指定位（ $0 \leq S2 \leq 15$ ，否则报操作数错误）

功能说明

将 **D** 元件的 **S** 位置位。

使用示例



能流有效时，将 D0（1000：2#0000001111101000）的 BIT15 置位。执行后 D0=33768（2#1000001111101000）

6.15.9 BRST：字位线圈清除指令

梯形图： 										适用机型	APLC-TCP				
指令列表： BRST (D) (S)										影响标志位					
										步长	5				
操作数	类型	适用软元件													变址
D	WORD			KnY	KnM	KnS	KnLM		D		C	T	V	Z	√
S	INT	常数	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	√

操作数说明

D：目的操作数

S2：指定位（ $0 \leq S2 \leq 15$ ，否则报操作数错误）

功能说明

将 **D** 元件的 **S** 位清位。

使用示例



能流有效时，将 D0（1000：2#0000001111101000）的 BIT8 清位。执行后 D0=744（2#0000001011101000）

6.16 比较触点指令

6.16.1 LD (=, <, >, <>, >=, <=)：整数比较 LD※指令

梯形图：										适用机型		APLC-TCP				
										影响标志位						
指令列表：										步长		5				
LD=	(S1)	(S2)														
LD<	(S1)	(S2)														
LD>	(S1)	(S2)														
LD<>	(S1)	(S2)														
LD>=	(S1)	(S2)														
LD<=	(S1)	(S2)														
操作数	类型	适用软元件													变址	
S1	INT	常数	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	√	
S2	INT	常数	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	√	

操作数说明

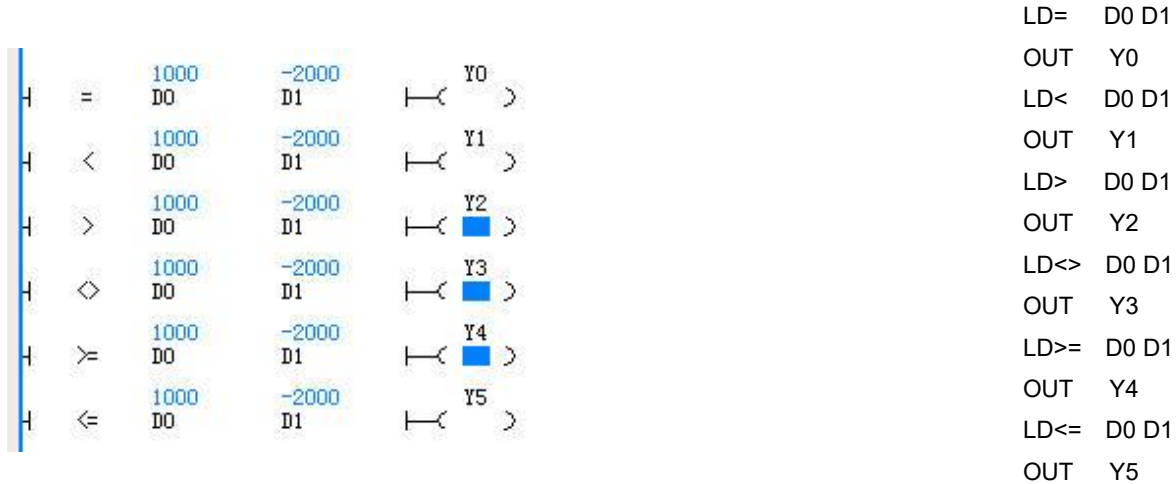
S1: 比较参数 1

S2: 比较参数 2

功能说明

对 **S1**、**S2** 单元内容进行 BIN 比较，比较的结果用于驱动后段运算。

使用示例



对 D0、D1 的数据进行 BIN 比较，比较的结果决定后段元件输出状态。

6.16.2 AND (=, <, >, <>, >=, <=) : 整数比较 AND※指令

梯形图：										适用机型		APLC-TCP					
<div><div><div><div></div><div></div></div><div><div></div><div></div></div></div><div>=</div><div>(S1)</div><div>(S2)</div><div><div><div></div><div></div></div><div><div></div><div></div></div></div><div>)</div></div> <div><div><div><div></div><div></div></div><div><div></div><div></div></div></div><div><</div><div>(S1)</div><div>(S2)</div><div><div><div></div><div></div></div><div><div></div><div></div></div></div><div>)</div></div> <div><div><div><div></div><div></div></div><div><div></div><div></div></div></div><div>></div><div>(S1)</div><div>(S2)</div><div><div><div></div><div></div></div><div><div></div><div></div></div></div><div>)</div></div> <div><div><div><div></div><div></div></div><div><div></div><div></div></div></div><div><></div><div>(S1)</div><div>(S2)</div><div><div><div></div><div></div></div><div><div></div><div></div></div></div><div>)</div></div> <div><div><div><div></div><div></div></div><div><div></div><div></div></div></div><div>>=</div><div>(S1)</div><div>(S2)</div><div><div><div></div><div></div></div><div><div></div><div></div></div></div><div>)</div></div> <div><div><div><div></div><div></div></div><div><div></div><div></div></div></div><div><=</div><div>(S1)</div><div>(S2)</div><div><div><div></div><div></div></div><div><div></div><div></div></div></div><div>)</div></div>										影响标志位							
指令列表： <div><div>AND=</div><div>(S1)</div><div>(S2)</div><div>AND<</div><div>(S1)</div><div>(S2)</div><div>AND></div><div>(S1)</div><div>(S2)</div><div>AND<></div><div>(S1)</div><div>(S2)</div><div>AND>=</div><div>(S1)</div><div>(S2)</div><div>AND<=</div><div>(S1)</div><div>(S2)</div></div>										步长		5					
操作数	类型	适用软元件													变址		
S1	INT	常数	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	√		
S1	INT	常数	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	√		

操作数说明

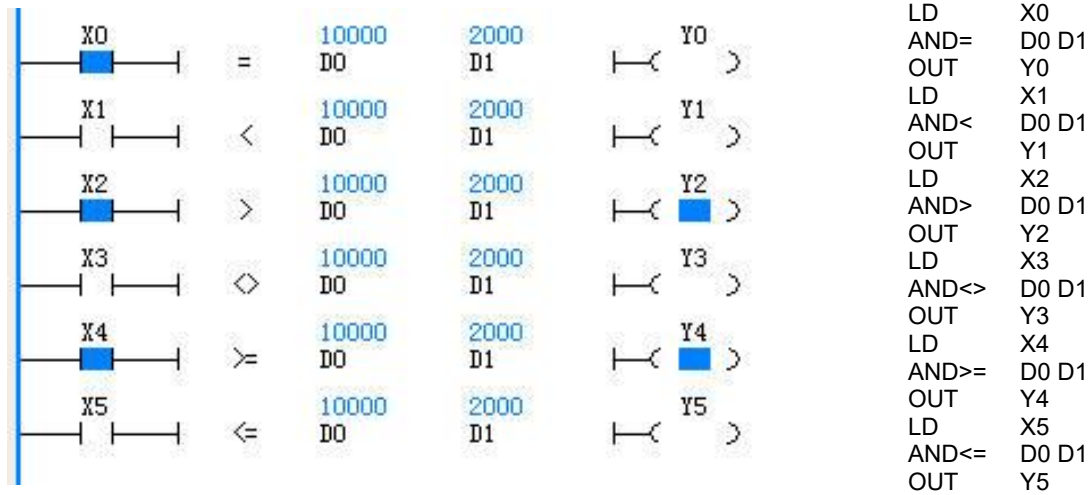
S1: 比较参数 1

S2: 比较参数 2

功能说明

对 **S1**、**S2** 单元内容进行 **BIN** 比较，比较的结果与其它节点串联用于驱动后段运算。

使用示例



对 **D0**、**D1** 的数据进行 **BIN** 比较，比较的结果与其它节点串联决定后段元件输出状态。

6.16.3 OR (=, <, >, <>, >=, <=) : 整数比较 OR※指令

梯形图:										适用机型		APLC-TCP						
										影响标志位								
指令列表:										步长		5						
指令列表:																		
指令列表:																		
指令列表:																		
指令列表:																		
指令列表:																		
指令列表:																		
指令列表:																		
指令列表:																		
指令列表:																		
指令列表:																		
指令列表:																		
指令列表:																		
指令列表:																		
指令列表:																		
指令列表:																		
指令列表:																		
指令列表:																		
指令列表:																		
指令列表:																		
指令列表:																		
指令列表:																		
指令列表:																		
指令列表:																		
指令列表:																		
指令列表:																		
指令列表:																		
指令列表:																		
指令列表:																		
指令列表:																		
指令列表:																		
指令列表:																		
指令列表:																		
指令列表:																		
指令列表:																		
指令列表:																		
指令列表:																		
指令列表:																		
指令列表:																		
指令列表:																		
指令列表:																		
指令列表:																		
指令列表:																		
指令列表:																		
指令列表:																		
指令列表:																		
指令列表:																		
指令列表:																		
指令列表:																		
指令列表:																		
指令列表:																		
指令列表:																		
指令列表:																		
指令列表:																		
指令列表:																		
指令列表:																		
指令列表:																		
指令列表:																		
指令列表:																		
指令列表:																		
指令列表:																		
指令列表:																		
指令列表:																		
指令列表:																		
指令列表:																		
指令列表:																		
指令列表:																		
指令列表:																		
指令列表:																		
指令列表:																		
指令列表:																		
指令列表:																		
指令列表:																		
指令列表:																		
指令列表:																		
指令列表:																		
指令列表:																		
指令列表:																		
指令列表:																		
指令列表:																		
指令列表:																		
指令列表:																		
指令列表:																		
指令列表:																		
指令列表:																		
指令列表:																		
指令列表:																		
指令列表:																		
指令列表:																		
指令列表:																		
指令列表:																		
指令列表:																		
指令列表:																		
指令列表:																		
指令列表:																		
指令列表:																		
指令列表:																		
指令列表:																		
指令列表:																		
指令列表:																		
指令列表:																		
指令列表:																		
指令列表:																		
指令列表:																		
指令列表:																		
指令列表:																		
指令列表:																		
指令列表:																		
指令列表:																		
指令列表:																		
指令列表:																		
指令列表:																		
指令列表:																		
指令列表:																		
指令列表:																		
指令列表:																		
指令列表:																		
指令列表:																		
指令列表:																		
指令列表:																		
指令列表:																		
指令列表:																		
指令列表:																		
指令列表:																		
指令列表:																		
指令列表:																		
指令列表:																		
指令列表:																		
指令列表:																		
指令列表:																		
指令列表:																		
指令列表:																		
指令列表:																		
指令列表:																		
指令列表:																		
指令列表:																		
指令列表:																		
指令列表:																		
指令列表:																		
指令列表:																		
指令列表:																		
指令列表:																		
指令列表:																		
指令列表:																		
指令列表:																		
指令列表:																		
指令列表:																		
指令列表:																		
指令列表:																		
指令列表:																		
指令列表:																		
指令列表:																		
指令列表:																		
指令列表:																		
指令列表:																		
指令列表:																		
指令列表:																		
指令列表:																		
指令列表:																		
指令列表:																		
指令列表:																		
指令列表:																		
指令列表:																		
指令列表:																		
指令列表:																		
指令列表:																		

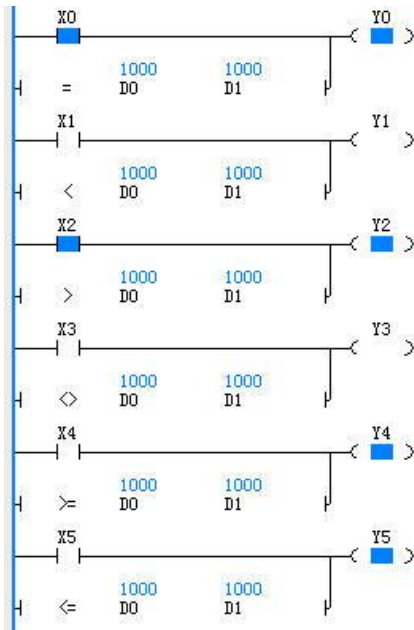
操作数说明

S1: 比较参数 1**S2**: 比较参数 2

功能说明

对 **S1**、**S2** 单元内容进行比较, 比较的结果与其它节点并联用于驱动后段运算。

使用示例



```
LD      X0
OR=     D0 D1
OUT     Y0
LD      X1
OR<     D0 D1
OUT     Y1
LD      X2
OR>     D0 D1
OUT     Y2
LD      X3
OR<>    D0 D1
OUT     Y3
LD      X4
OR>=    D0 D1
OUT     Y4
LD      X5
OR<=    D0 D1
OUT     Y5
```

对 D0、D1 的数据进行比较, 比较的结果与其它节点并联决定后段元件输出状态

6.16.4 LDD (=, <, >, <>, >=, <=) : 长整数比较 LDD※指令

梯形图：										适用机型		APLC-TCP					
<div><div><div></div><div>D=</div><div>(S1)</div><div>(S2)</div><div><div></div><div></div></div><div>)</div></div><div><div></div><div>D<</div><div>(S1)</div><div>(S2)</div><div><div></div><div></div></div><div>)</div></div><div><div></div><div>D></div><div>(S1)</div><div>(S2)</div><div><div></div><div></div></div><div>)</div></div><div><div></div><div>D<></div><div>(S1)</div><div>(S2)</div><div><div></div><div></div></div><div>)</div></div><div><div></div><div>D>=</div><div>(S1)</div><div>(S2)</div><div><div></div><div></div></div><div>)</div></div><div><div></div><div>D<=</div><div>(S1)</div><div>(S2)</div><div><div></div><div></div></div><div>)</div></div></div>										影响标志位							
指令列表： <div><div>LDD=</div><div>(S1)</div><div>(S2)</div><div>LDD<</div><div>(S1)</div><div>(S2)</div><div>LDD></div><div>(S1)</div><div>(S2)</div><div>LDD<></div><div>(S1)</div><div>(S2)</div><div>LDD>=</div><div>(S1)</div><div>(S2)</div><div>LDD<=</div><div>(S1)</div><div>(S2)</div></div>										步长		7					
操作数	类型	适用软元件													变址		
S1	DINT	常数	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C		V		√		
S2	DINT	常数	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C		V		√		

操作数说明

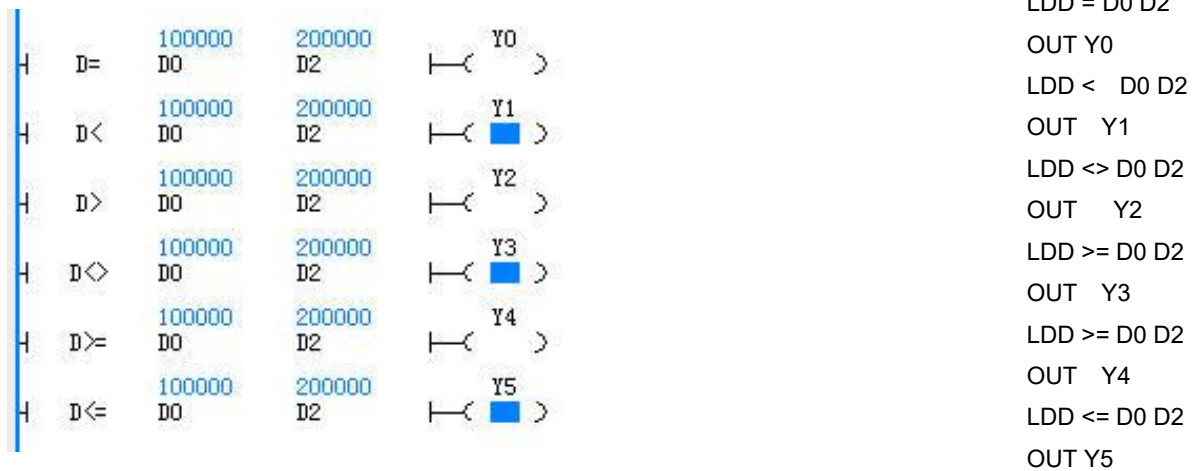
S1: 比较参数 1

S2: 比较参数 2

功能说明

对 **S1**、**S2** 单元内容进行比较，比较的结果用于驱动后段运算。

使用示例



比较 (D0,D1) 和 (D2,D3)，比较结果决定后段元件输出状态。

6.16.5 ANDD (=, <, >, <>, >=, <=) : 长整数比较 ANDD※指令

梯形图：									适用机型			APLC-TCP				
<div><div><div><div></div><div></div></div><div><div></div><div></div></div></div><div>D=</div><div>(S1)</div><div>(S2)</div><div><div><div></div><div></div></div><div><div></div><div></div></div></div><div>)</div></div> <div><div><div><div></div><div></div></div><div><div></div><div></div></div></div><div>D<</div><div>(S1)</div><div>(S2)</div><div><div><div></div><div></div></div><div><div></div><div></div></div></div><div>)</div></div> <div><div><div><div></div><div></div></div><div><div></div><div></div></div></div><div>D></div><div>(S1)</div><div>(S2)</div><div><div><div></div><div></div></div><div><div></div><div></div></div></div><div>)</div></div> <div><div><div><div></div><div></div></div><div><div></div><div></div></div></div><div>D<></div><div>(S1)</div><div>(S2)</div><div><div><div></div><div></div></div><div><div></div><div></div></div></div><div>)</div></div> <div><div><div><div></div><div></div></div><div><div></div><div></div></div></div><div>D>=</div><div>(S1)</div><div>(S2)</div><div><div><div></div><div></div></div><div><div></div><div></div></div></div><div>)</div></div> <div><div><div><div></div><div></div></div><div><div></div><div></div></div></div><div>D<=</div><div>(S1)</div><div>(S2)</div><div><div><div></div><div></div></div><div><div></div><div></div></div></div><div>)</div></div> <div>影响标志位</div>																
指令列表： ANDD = (S1) (S2) ANDD < (S1) (S2) ANDD > (S1) (S2) ANDD < > (S1) (S2) ANDD > = (S1) (S2) ANDD < = (S1) (S2)									步长			7				
操作数		类型		适用软元件											变址	
S1	DINT	常数	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C		V		√	
S2	DINT	常数	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C		V		√	

操作数说明

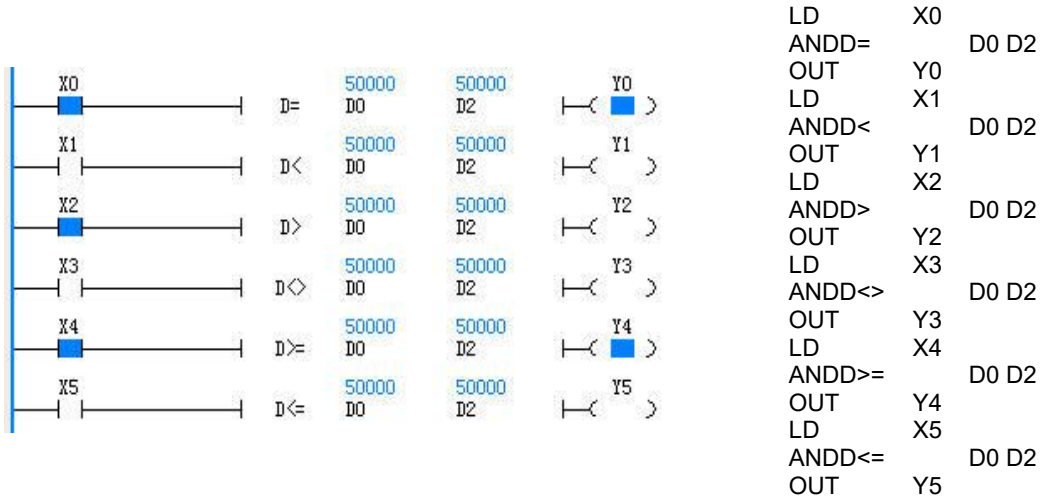
S1: 比较参数 1

S2: 比较参数 2

功能说明

对 **S1**、**S2** 单元内容进行比较，比较的结果与其它节点串联用于驱动后段运算。

使用示例



比较 (D0,D1) 和 (D2,D3)，比较的结果与其它节点串联决定后段元件输出状态。

6.16.6 ORD (=, <, >, <>, >=, <=) : 长整数比较 ORD※指令

梯形图：										适用机型		APLC-TCP					
										影响标志位							
指令列表：										步长		7					
ORD = (S1) (S2) ORD < (S1) (S2) ORD > (S1) (S2) ORD < > (S1) (S2) ORD > = (S1) (S2) ORD < = (S1) (S2)																	
操作数	类型	适用软元件													变址		
S1	DINT	常数	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C		V		√		
S2	DINT	常数	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C		V		√		

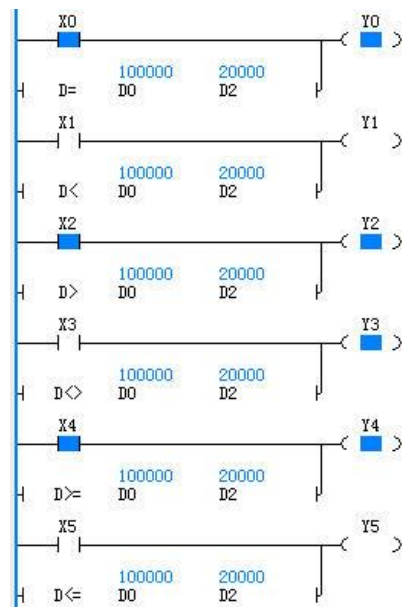
操作数说明

S1: 比较参数 1**S2**: 比较参数 2

功能说明

对 **S1**、**S2** 单元内容进行比较, 比较的结果与其它节点并联用于驱动后段运算。

使用示例



```
LD      X0
ORD=    D0 D2
OUT     Y0
LD      X1
ORD<    D0 D2
OUT     Y1
LD      X2
ORD<>   D0 D2
OUT     Y2
LD      X3
ORD>=   D0 D2
OUT     Y3
LD      X4
ORD>=   D0 D2
OUT     Y4
LD      X5
ORD<=   D0 D2
OUT     Y5
```

比较 (D0,D1) 和 (D2,D3), 比较的结果与其它节点并联决定后段元件输出状态。

6.16.7 LDR: 浮点数比较指令

梯形图：										适用机型		APLC-TCP					
<div><div><div></div><div></div><div></div><div></div><div></div><div></div></div><div>R= R< R> R<> R>= R<=</div><div>(S1) (S1) (S1) (S1) (S1) (S1)</div><div>(S2) (S2) (S2) (S2) (S2) (S2)</div><div><div><div></div><div></div><div></div><div></div><div></div><div></div></div><div>))))))</div></div></div>										影响标志位							
指令列表：										步长		7					
LDR = (S1) (S2)																	
LDR < (S1) (S2)																	
LDR > (S1) (S2)																	
LDR < > (S1) (S2)																	
LDR > = (S1) (S2)																	
LDR < = (S1) (S2)																	
操作数	类型	适用软元件										变址					
S1	REAL	常数								D				V		√	
S2	RAEL	常数								D				V		√	

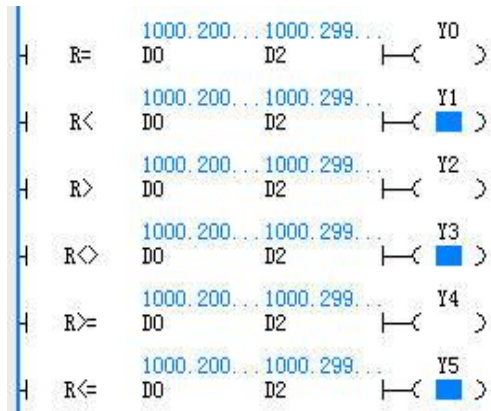
操作数说明

S1: 比较参数 1**S2**: 比较参数 2

功能说明

对 **S1**、**S2** 单元内容进行比较，比较的结果用于驱动后段运算。

使用示例



LDR= D0 D2
OUT Y0
LDR< D0 D2
OUT Y1
LDR> D0 D2
OUT Y2
LDR<> D0 D2
OUT Y3
LDR>= D0 D2
OUT Y4
LDR<= D0 D2
OUT Y5

比较 (D0,D1) 和 (D2,D3) ，比较的结果决定后段元件输出状态。

6.16.8 ANDR：浮点数比较指令

梯形图：										适用机型		APLC-TCP					
<div><div><div></div><div>R=</div><div>(S1)</div><div>(S2)</div><div></div><div></div></div><div><div></div><div>R<</div><div>(S1)</div><div>(S2)</div><div></div><div></div></div><div><div></div><div>R></div><div>(S1)</div><div>(S2)</div><div></div><div></div></div><div><div></div><div>R<></div><div>(S1)</div><div>(S2)</div><div></div><div></div></div><div><div></div><div>R>=</div><div>(S1)</div><div>(S2)</div><div></div><div></div></div><div><div></div><div>R<=</div><div>(S1)</div><div>(S2)</div><div></div><div></div></div></div>										影响标志位							
指令列表： <div><div>ANDR=</div><div>(S1)</div><div>(S2)</div></div> <div><div>ANDR<</div><div>(S1)</div><div>(S2)</div></div> <div><div>ANDR></div><div>(S1)</div><div>(S2)</div></div> <div><div>ANDR<></div><div>(S1)</div><div>(S2)</div></div> <div><div>ANDR>=</div><div>(S1)</div><div>(S2)</div></div> <div><div>ANDR<=</div><div>(S1)</div><div>(S2)</div></div>										步长		7					
操作数	类型	适用软元件												变址			
S1	REAL	常数							D					V		√	
S2	REAL	常数							D					V		√	

操作数说明

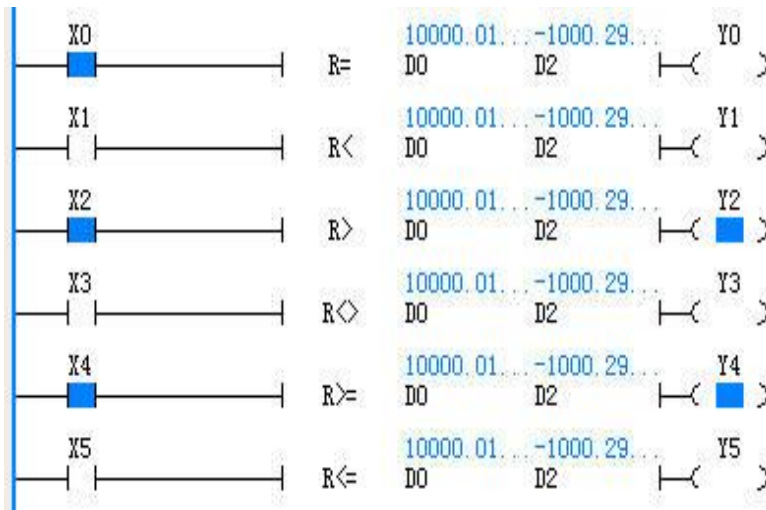
S1：比较参数 1

S2：比较参数 2

功能说明

对 **S1**、**S2** 单元内容进行比较，比较的结果与其它节点串联用于驱动后段运算。

使用示例



```

LD      X0
ANDR=   D0 D2
OUT     Y0
LD      X1
ANDR<   D0 D2
OUT     Y1
LD      X2
ANDR<>  D0 D2
OUT     Y2
LD      X3
OUT     Y3
LD      X4
ANDR>=  D0 D2
OUT     Y4
LD      X5
ANDR<=  D0 D2
OUT     Y5

```

比较 (D0,D1) 和 (D2,D3) , 比较的结果与其它节点串联决定后段元件输出状态。

6.16.9 ORR: 浮点数比较指令

梯形图：										适用机型		APLC-TCP					
										影响标志位							
指令列表：										步长		7					
ORR = (S1) (S2) ORR < (S1) (S2) ORR > (S1) (S2) ORR < > (S1) (S2) ORR > = (S1) (S2) ORR < = (S1) (S2)																	
操作数		类型		适用软元件										变址			
S1		REAL 常数									D				V		√
S2		REAL 常数									D				V		√

操作数说明

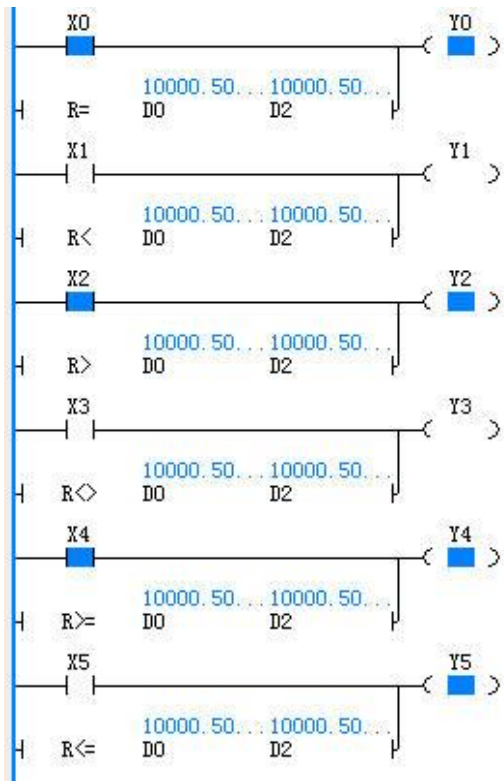
S1: 比较参数 1

S2: 比较参数 2

功能说明

对 **S1**、**S2** 单元内容进行比较，比较的结果与其它节点并联用于驱动后段运算。

使用示例



```
LD      X0
ORR=    D0 D2
OUT     Y0
LD      X1
ORR<    D0 D2
OUT     Y1
LD      X2
ORR>    D0 D2
OUT     Y2
LD      X3
ORR<>   D0 D2
OUT     Y3
LD      X4
ORR>=   D0 D2
OUT     Y4
LD      X5
ORR<=   D0 D2
OUT     Y5
```

比较 (D0,D1) 和 (D2,D3)，比较的结果与其它节点并联决定后段元件输出状态。

6.16.10 CMP：整数比较置位指令

无

6.16.11 LCMP：长整数比较置位指令

无

6.16.12 RCMP：浮点数比较置位指令

无

第七章 顺序功能图使用指南

本章介绍了顺序功能图编程的基本概念、基本和复杂流程的编程方法，以及编程的注意事项。

7.1 顺序功能图介绍	252
7.1.1 什么是顺序功能图	252
7.1.2 什么是 APLC 系列 PLC 的顺序功能图	252
7.1.3 顺序功能图的基本概念	252
7.1.4 编程图元及其连接规则	253
7.1.5 顺序功能图结构	254
7.1.6 顺序功能图程序的执行	258
7.2 顺序功能图与梯形图的对应关系	259
7.2.1 STL 指令与步进状态	259
7.2.2 SFC 状态转移指令	259
7.2.3 RET 指令与 SFC 程序段	259
7.2.4 SFC 状态跳转指令、重置指令	260
7.2.5 SFC 选择分支、并行分支及汇合	260
7.3 顺序功能图编程步骤	260
7.4 顺序功能图编程注意事项	261
7.4.1 常见的编程错误	261
7.4.2 编程技巧	263
7.5 顺序功能图编程实例	265
7.5.1 简单结构流程	265
7.5.2 选择结构	267
7.5.3 并行结构	271

7.1 顺序功能图介绍

7.1.1 什么是顺序功能图

顺序功能图 (Sequential Function Chart) 是一种近年来逐渐发展并流行起来的编程语言, 用于将 PLC 编程项目划分成结构化流程。它使用 IAT61131-3 标准中规定的编程元件和语言结构, 将复杂的系统过程划分为按顺序进行的多段流程步骤以及步骤间的转换过程, 从而实现顺序控制功能。

由于顺序功能图编程具有直观和流程化的特点, 分解后的每一步骤和每个转换条件都为相对简单的程序过程, 很适应顺序化控制的应用领域, 因此逐渐得到了广泛应用。

7.1.2 什么是 APLC 系列 PLC 的顺序功能图

APLC 系列 PLC 的顺序功能图是阿尔法 AT 系列 PLC 产品使用的一种编程语言。除了标准的顺序功能图功能之外, 还可内置一个或多个梯形图程序块。

用 APLC 系列 PLC 顺序功能图编写的程序可以转换成对应的梯形图和语句表程序。

APLC 系列 PLC 的顺序功能图程序还支持多个独立流程, 数量最多可达到 20 个。这些独立流程可以分别独立运行, 每个流程内的步进状态按工序分别扫描和转移。在各独立流程间可进行跳转。

7.1.3 顺序功能图的基本概念

顺序功能图有以下两个基本概念: 步进状态和转移。其它的概念, 如跳转、分支、多个独立流程等, 都是在此基础上派生而来的。

步进状态

1. 步进状态的定义

一个步进状态实际为一段独立的程序, 代表了顺序控制过程中的一个工作状态或一个工序。将多个步进状态进行有机的组合即可组成一个完整的顺序功能图程序。

2. 步进状态的执行

在顺序功能图程序中, 步进状态由固定的 S 元件来代表。

一个正在执行中的步进状态被称为是有效的步进状态, 其对应的 S 元件状态为 ON, 此时 PLC 扫描执行该步进状态内的所有指令序列。而未被执行的步进状态被称为无效的步进状态, 其对应的 S 元件状态为 OFF, 此时 PLC 不扫描执行相应的内部指令序列。

转移

顺序控制过程是一系列的步进状态切换过程。正在执行某个步进状态的 PLC, 在满足一定的逻辑条件的情况下, 会离开当前的步进状态, 进入并执行新的步进状态。这个切换过程被称为步进状态的转移。

转移的发生必须满足一定的逻辑条件, 这种逻辑条件被称为步进转移条件。

7.1.4 编程图元及其连接规则

编程图元

APLC 系列 PLC 顺序功能图由以下基本的编程图元构成。

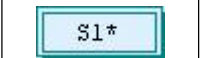


编程图元	图形表达	具体说明
初始步进符		代表一个起始的步进状态，一个步进状态的编号为指定的 S 元件号，编号不能重复。一个顺序功能图网络的执行必须由初始步进符开始。初始步进符对应的 S 软元件地址范围是 S0~S19
普通步进符		代表一个普通的步进状态，一个步进状态的编号为指定的 S 元件号，编号不能重复。普通步进符对应的 S 软元件地址范围是 S20--S991 软元件
转移符		代表转移，可内置使下一个步进有效的转移条件（内置梯形图）。用户可以自己定义其中的代码，达到转移条件时与转移符连接的下一个 S 软元件状态置位，进入下一步进状态。转移符必须连接在步进符之间
跳转符		跳转符，连接在转移符之后，达到转移条件时能使指定的 S 元件为 ON。用于步进状态的循环或跳转
重置符		重置符，连接在转移符之后，达到转移条件时能使指定的 S 元件为 OFF。用于顺序功能图程序段的结束
选择分支符		连接在步进符之后，分别代表多个相互独立的转移条件，当达到其中任何一个转移条件时，即结束上一步进状态，进入该转移条件下对应的步进分支。用于选择多个步进分支中的一个，选择一个分支后其它分支则不会再选中
选择汇合符		连接在选择分支汇合点，代表选择步进支路汇合。当达到其中一个支路的转移条件时，即转移到下一步进状态
并行分支符		连接在步进符之后，其后的多路分支共同等待同一个转移条件。当该转移条件成立后，同时使后面的这些多个步进分支有效并行执行
并行汇合符		连接在并行分支汇合点，转移条件代表了各个分支的结束条件之和。多个并行步进分支都执行完毕，满足转移条件后，才能使后面的一个步进状态生效
梯形图块		梯形图块用于表示顺序控制图流程以外的梯形图指令，可以用于起始步进的启动及通用操作

表 7-1 编程图元

编程图元连接规则

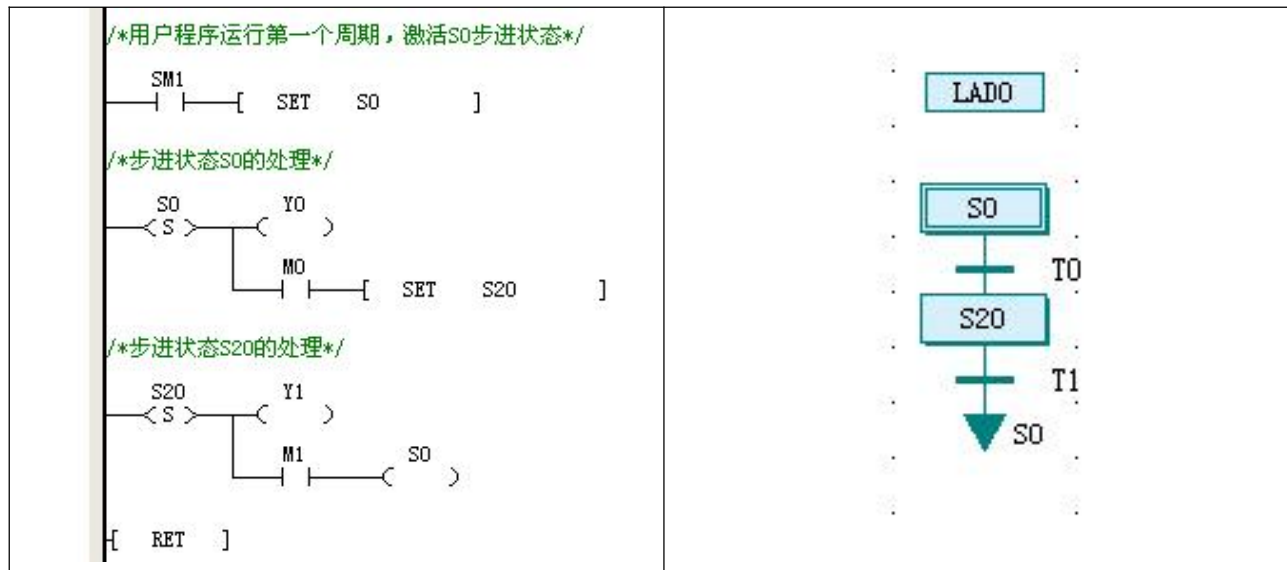
1. 初始步进符不能前接其它图元，后接图元必须是转移符，也可不参与连接。
2. 梯形图块不与任何其它图元相连接。
3. 与普通步进符直接相连的图元必须是转移符，普通步进符在图中不可孤立存在。
4. 重置符和跳转符应前接转移符，不能后接其它元件。
5. 转移符和跳转符不可孤立存在。

7.1.5 顺序功能图结构

SFC 流程结构分为简单顺序结构、选择结构和并行结构三类。除此之外，跳转也是选择结构的一种。

简单顺序结构

如下图为简单顺序结构及其梯形图表示的示例。



在简单顺序结构中，当步进转移条件满足时，由前一个步进状态顺序转移到下一个状态，其中无任何分支结构。最后一个步进状态当转移条件满足时，退出顺序功能图程序段，或者转移到初始步进状态。

1. 梯形图块

梯形图块用于启动顺序功能图程序段，即设置初始步进符的 `S` 元件为 `ON`，上图例程中采用上电启动的方式。

梯形图块还用于非顺序功能图的其它通用程序段。

2. 初始步进状态

示例中由梯形图块来启动初始步进状态。S 元件范围 0~19。

3. 普通步进符

用于顺序过程中的编程。S 元件范围 20~1023 (AT100 系列适用)。

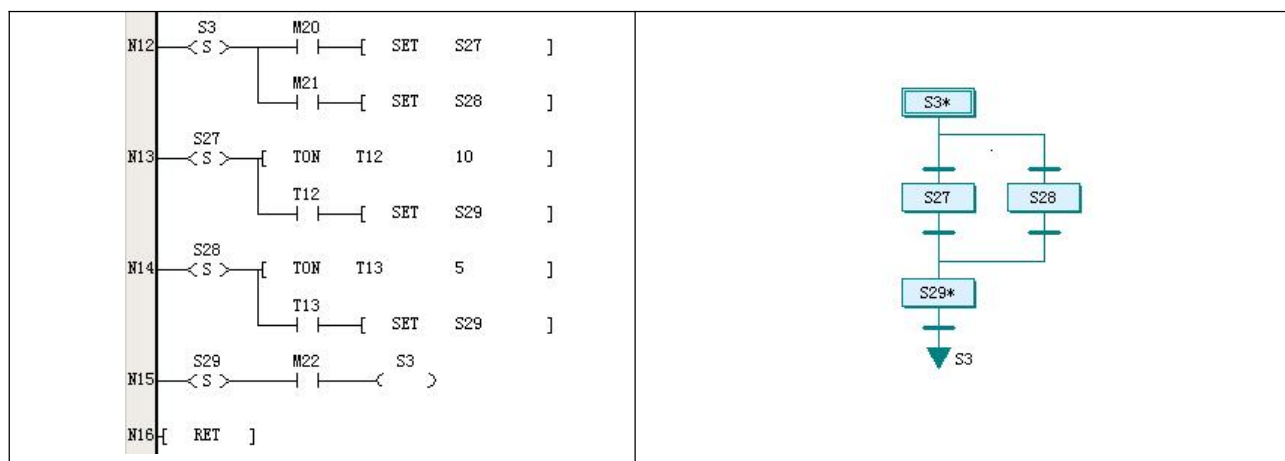
4. 转移或重置

示例程序最后一个转移符连接跳转符，跳转到初始步进状态。这是一种连续循环操作的流程。

最后一个转移符也可以连接重置符，复位最后一个步进状态。重置之后，这一次简单顺序结构的流程操作就完成了，然后等待下一次流程操作的开始。

选择分支结构

选择分支结构示例如下图所示，左图为梯形图，右图为对应的顺序功能图。



1. 选择分支

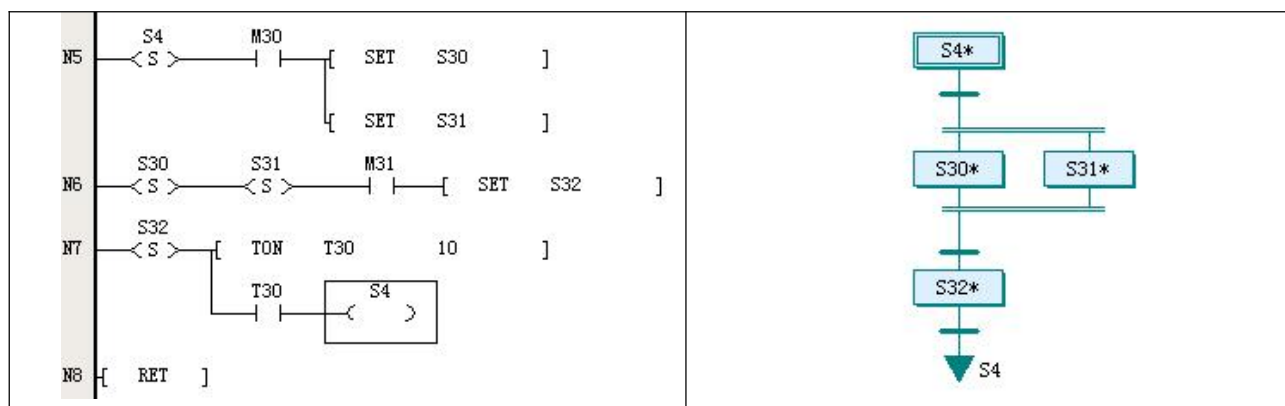
根据各分支转移条件，选择激活相应支路上的步进状态。用户必须确保分支中的转移条件互斥。因此选择结构在流程运行时，一次只能选择一个分支。如上图程序所示，第 N12 行程序中，S27、S28 两个步进状态分别由 M20、M21 作为转移条件，当保证 M20、M21 不会同时置位时，S27 和 S28 就只能是二者选其一。

2. 选择汇合

选择分支汇合符处，所有的分支都连接到同一个步进状态，转移条件各自独立。如上图程序所示，第 N13 行中 S27 步进状态的转移条件是 T12 计时时间到；而第 N14 行中 S28 步进状态的转移条件是 T13 计时时间到。转移结果都是进入到下一个步进状态 S29。

并行分支结构

并行结构示例如下图所示，左图为梯形图，右图为对应的顺序功能图。



1. 并行分支

当并行分支结构的转移条件满足时，并行分支结构下接的各步进状态同时激活。这也是一种常见的顺控结构，即在一定条件下，将并行地启动处理多项工序。如图中 N5 行程序所示，M30 为转移条件，当 M30 置位，则 S30 和 S31 步进状态同时有效。

2. 并行汇合

当并行汇合结构的转移条件满足时，将会使并行分支结构上接的各步进状态同时失效，转移到后续步进状态。如图 N6 行程序所示，当处在 S30 和 S31 步进状态下，M31 置位时，即转移到 S32 步进状态，结束 S30 和 S31 步进状态。

并行汇合的转移条件要保证汇合之前进行处理的各项独立步骤能全部完成，然后才能进行转移。

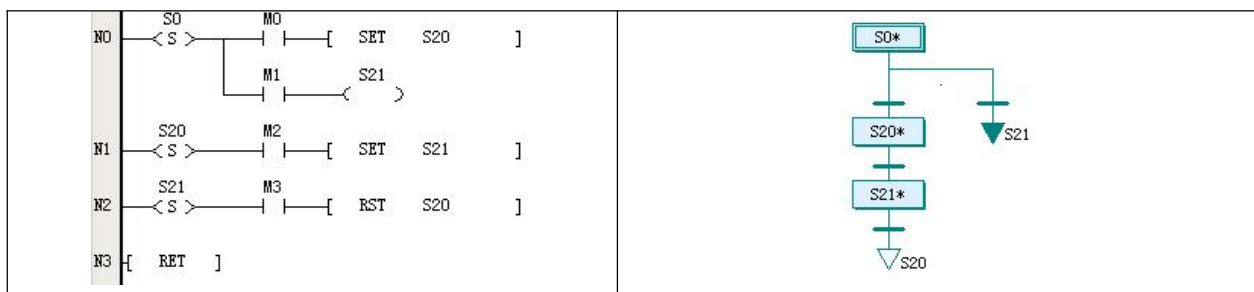
跳转

跳转结构常用于以下用途：跨越部分步进状态；循环：：返回起始步进状态或某一普通步进状态；转移到其它流程。

1. 跨越部分步进状态

一个流程中，根据一定的转移条件，当不需要顺序执行时，可以采用跳转符转移到需要的步进状态，跨越部分步进状态。

如下为示例图。左边为梯形图，右边为对应的顺序功能图。



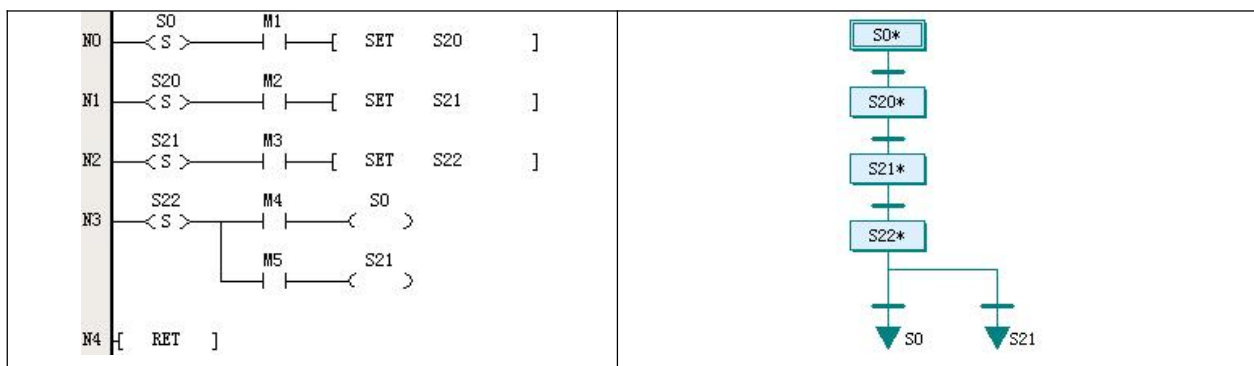
在顺序功能图中，采用 S21 跳转符表示跳转，S20 步进状态被跨越。跳转之前实际上是选择分支结构。

在梯形图中，第 N0 行的第二支路就是跳转指令。跳转指令采用的是 OUT 线圈的形式，而不是顺序转移的 SET 指令形式。在 S0 步进状态运行时，当 M1 为 ON，则实现跳转，跨越到 S21 状态。

2. 循环

一个流程中，根据一定的转移条件，需要在部分或全部步进状态间循环时，采用跳转符实现循环的功能。在这段流程最后的转移时，跳转到之前的普通步进符，实现部分步进状态循环功能；如果是跳转到初始步进符，则实现全部步进状态循环功能。

如下为同时实现上述两种循环结构的示例程序，左图为梯形图，右图为对应的顺序功能图。



在顺序功能图中，在 S22 步进状态下，当其中一个转移条件满足时，跳转到 S21，重新运行 S21 步进状态。在另一个转移条件下，将跳转到 S0 初始步进状态，重新运行全部步进状态。

在梯形图中，这两个循环的跳转都在第 N3 行中实现，可见跳转指令的 OUT 线圈。

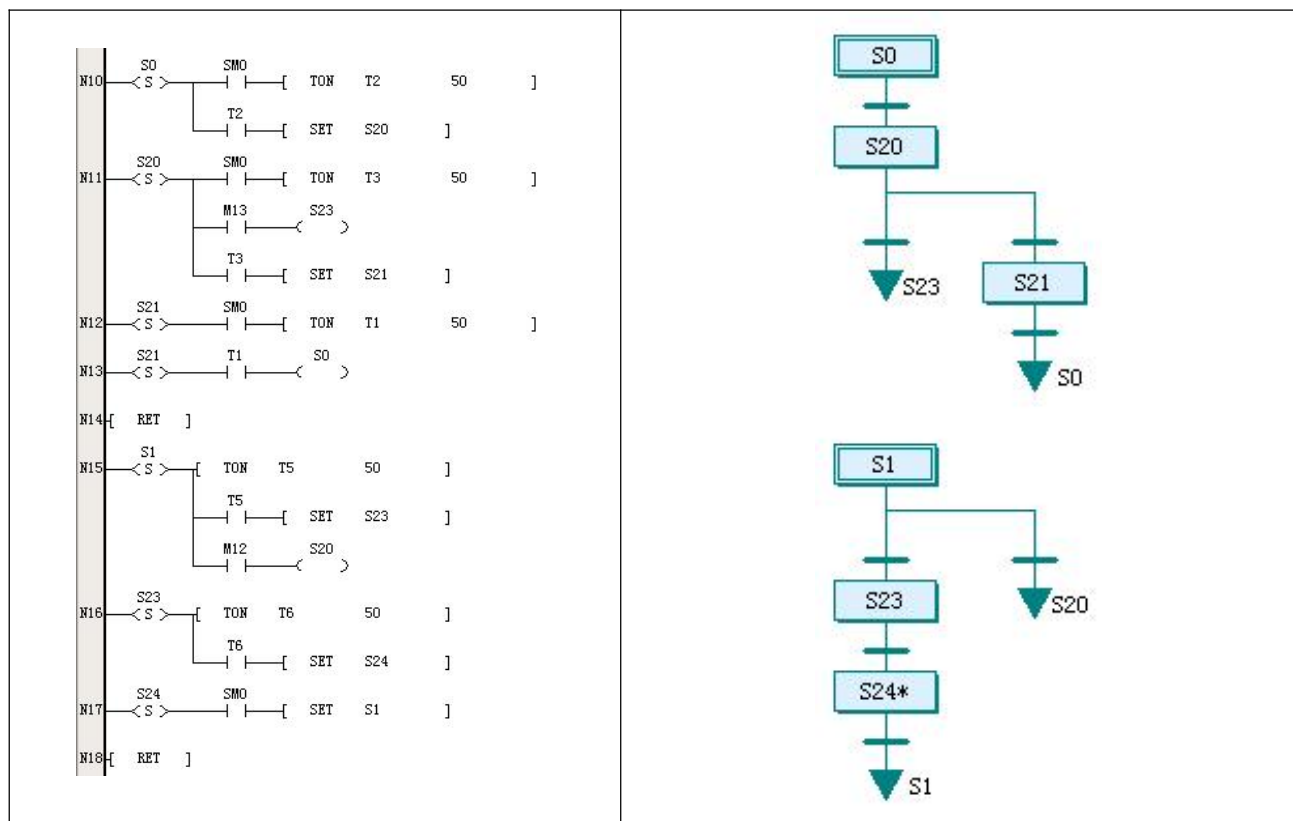
3. 在不同的独立流程间跳转

APLC 系列 PLC 顺序功能图程序中可以同时存在多个独立流程，支持在这些流程间的跳转。在一个独立流程中可以设置转移条件，在该条件满足的时候，直接转移到另一个独立流程。可以跳转到另一个流程的初始步进状态，也可以转到普通步进状态。

注意

在多个流程之间跳转会增加 PLC 程序的复杂性，必须谨慎对待。

如下图为实现了从一个独立流程跳转到另一个流程的示例程序。左图为梯形图，右图为对应的顺序功能图。



在顺序功能图中，在上面的流程 S20 步进状态下，可根据转移条件跳转到 S23 步进状态；而在下面的流程步进状态 S1 下，也可根据转移条件跳转到 S20 步进状态。

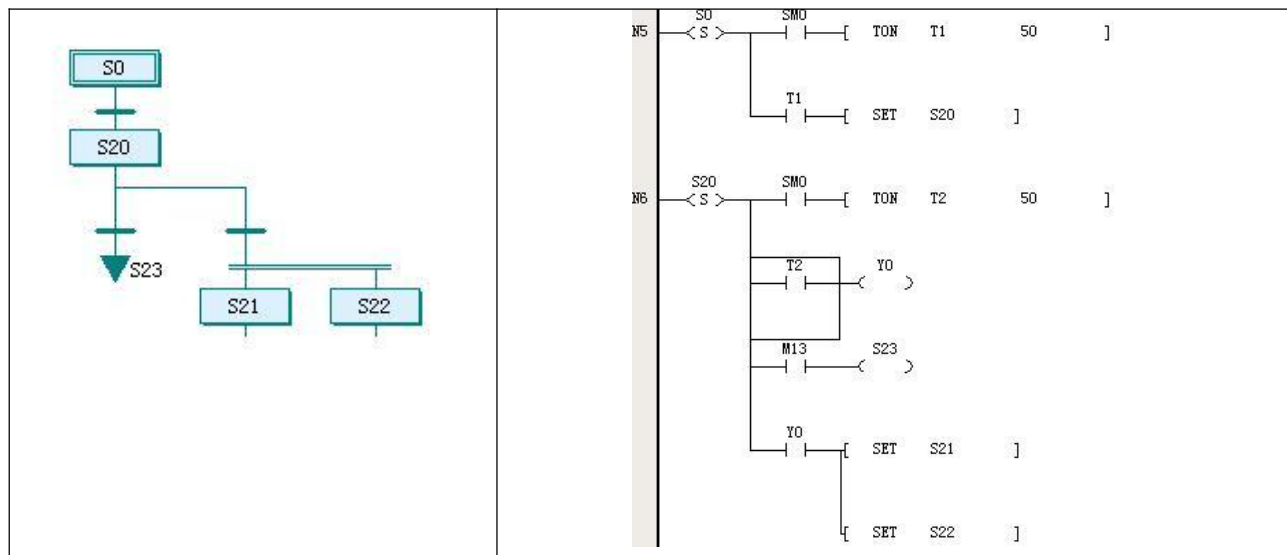
在示例图中，可以看到这种跳转是基于选择分支结构的，因此在发生不同流程间的跳转时，发生跳转的那个流程中的步进状态将全部无效。例如示例图中上面的流程 S20 步进状态下，如果转移到了下面的流程 S23 步进状态，则 S20 将会置 OFF，该独立流程所有的步进状态 S0、S20、S21 都为 OFF，也就是处于无效状态。

7.1.6 顺序功能图程序的执行

顺序功能图程序的执行过程与普通梯形图相同之处在于：两者都是连续地从上到下、自左及右地周期扫描。

顺序功能图程序的执行过程与普通梯形图不同之处在于：顺序功能图的各步进状态会按顺序条件切换有效无效状态，相应有效的步进状态的内部指令序列才会得到扫描执行，无效的步进状态内部指令则不会被扫描执行；而普通梯形图主程序的全部程序行则在每次扫描周期中都会被经过和执行。

如下图所示，右边的梯形图程序由左边的顺序功能图程序转换而来。当 S20 步进状态有效时，T2 计时器会被扫描并计时，在 T2 完成之前不会进入 S21、S22 状态；在 M13 为 OFF 时，不会进入 S23，它们的内部指令都不会被扫描执行。



各 S 元件之间根据步进转移条件进行 ON/OFF 切换，结果就导致上一步进状态转移到下一步进状态。在一个 S 元件由 ON 跳转为 OFF 时，其内部指令的输出软元件会被复位或清零。请参见 5.3.1 STL：SFC 状态装载指令。

注意

1. APLC 系列 PLC 顺序功能图程序中一般都会同时包含有顺序功能图和梯形图块。梯形图块用于处理流程以外的事务，包括启动顺序功能图的操作，不受任何 S 元件的控制。在每个扫描周期中，PLC 扫描到的这些梯形图块的程序行都会被执行。

2. 由于 S 元件的状态变化会对步进状态内置指令产生影响，且上下步进状态切换也有一个过程，因此在进行顺序功能图编程时对软件元件的操作和指令的使用有一些需要注意的事项。详细情况请参阅 7.4 顺序功能图编程注意事项。

7.2 顺序功能图与梯形图的对应关系

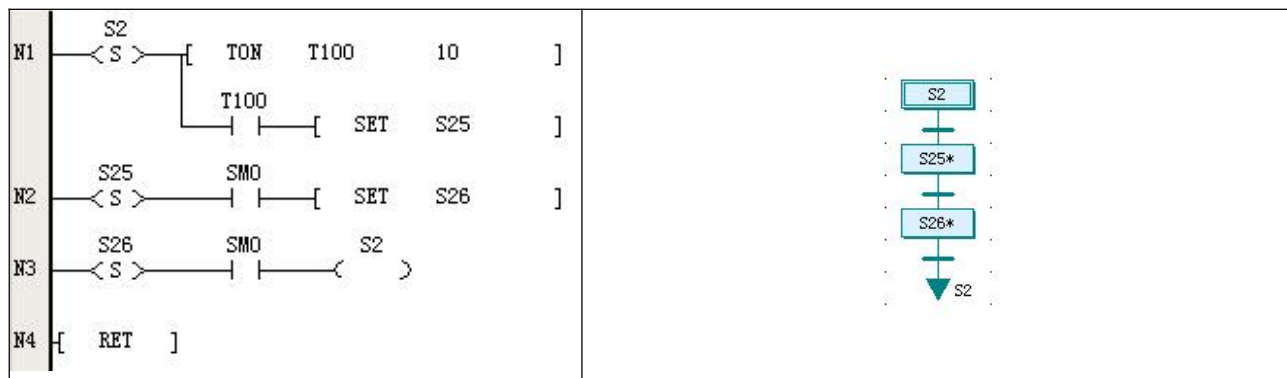
SFC 程序可用梯形图来表示。利用梯形图可以了解 SFC 程序结构的实际意义。

在梯形图中，SFC 编程的各种图元符号都有对应的 SFC 指令，而对应的流程也是有特定结构的。

7.2.1 STL 指令与步进状态

在梯形图中是以 STL 指令来启动一个步进状态的。每个步进状态都以 S 元件为标志。

如下左图所示为一个简单顺序结构示例程序的梯形图程序。右图为该流程的 SFC 程序。



梯形图中，S2 步进状态是从 STL 状态装载指令开始。其后的 TON 计时器语句为 S2 步进状态内部指令序列。一个步进状态内部指令序列可以有多个语句，与普通梯形图程序基本一致，实际上就是一段相对完整的程序片段。

初始步进状态与普通步进状态区别仅仅是选用 S 元件的范围不同。

STL 指令的详细说明请参阅 5.3.1 STL：SFC 状态装载指令。其中要注意的是，步进状态由 ON 到 OFF 跳变时，内置的 OUT、TON、TOF、PWM、HCNT、PLSY、PLSR、DHSCS、SPD、DHSCI、DHSCR、DHSZ、DHST、DHSP、BOUT 所对应的目的操作数将被清除。

注意

由于 PLC 是连续周期性扫描的，因此，当一个步进状态转移到下一步步进状态时，原步进状态的那些内置语句要到下一次扫描时才会受到 ON 到 OFF 跳变的影响。请参阅 7.4.1 常见的编程错误。

7.2.2 SFC 状态转移指令

如上图所示，右图的转移符在左图梯形图中是用 SFC 状态转移指令实现的。

转移条件是 SET 语句前面的常开触点元件组合而成。常开触点元件是由内置语句或外部操作来控制的。

当 SFC 状态转移指令能流有效时，将指定步进状态置为有效，同时使当前有效的步进状态置为失效，完成步进状态转移的动作。

7.2.3 RET 指令与 SFC 程序段

如上图所示，右图的 SFC 程序，由 S2 初始步进符开始，经过 2 个普通步进符后再返回 S2 步进符。在梯形图中，SFC 程序段落的结束必须是以 RET 指令来标记的。

RET 指令只能用在主程序中。

7.2.4 SFC 状态跳转指令、重置指令

上图中，跳转符 S2 在梯形图中是 N3 行所示。采用 OUT 指令，实现了跳转。跳转可在同一流程中，也可以在不同独立流程之间。

如果采用了重置符 S26，在梯形图中 N3 行则为 RST 指令，实现对上一步进状态 S26 的复位。

7.2.5 SFC 选择分支、并行分支及汇合

选择分支的梯形图示例请参阅 7.1.5 顺序功能图结构中的选择分支结构。

并行分支的梯形图示例请参阅 7.1.5 顺序功能图结构中的并行分支结构。

7.3 顺序功能图编程步骤

1. 分析流程，确定程序流程结构

程序流程结构可分为简单顺序结构、选择结构、并行结构，跳转也是选择结构的一种。采用 SFC 编程时，第一步要确定是哪一种流程结构。比如单个对象连续通过前后顺序步骤完成操作，一般是简单顺序结构；有多个产品加工选项，各选项参数不同，且不能同时加工的，则应该确定为选择结构；多个机械装置联合运行却又相对独立的，则可能是并行结构。

2. 确定主要步骤和主要转移条件，得出流程草图

确定了流程结构，接下来是大体上确定主要步骤和主要转移条件。把流程结构划分为细化的操作过程，每个操作过程就是步骤，而操作过程结束的重要标志则是转移条件。这样就能得到流程的草图。

3. 根据流程草图做出 SFC 顺序功能图

打开 YD-station 编程软件的 SFC 编程界面，将流程草图变为 SFC 顺序功能图。此时已经能够得到可执行的 PLC 程序，只是程序有待完善。

4. 做出输入输出点表，确定各步骤操作对象及实际转移条件

输入点多为转移条件，输出点多为操作对象。根据点表，还可以进一步修正顺序功能图。

5. 步骤和转移条件的输入

在 SFC 编程界面中，使用鼠标右键点击 SFC 图元，可以弹出相应的右键菜单，选择内置梯形图选项，可以打开该图元的内置梯形图编辑工作区，输入梯形图程序及条件。

6. 编写梯形图程序块

不要忘记在程序中编写一些通用处理功能的梯形图程序块，比如顺序流程的启动，还有停机、报警等等通用的操作。这些都需要放在梯形图块中。

注意

启动停机操作关系到人身和设备安全。要考虑到 SFC 编程的特殊性，停机时尽可能把全部该停止运行的输出关闭。

7.4 顺序功能图编程注意事项

由于 STL 语句具有一些特点，且 PLC 是按语句顺序周期性扫描的，因此 SFC 编程有几个比较重要的注意事项。

7.4.1 常见的编程错误

1. 重用步进状态符

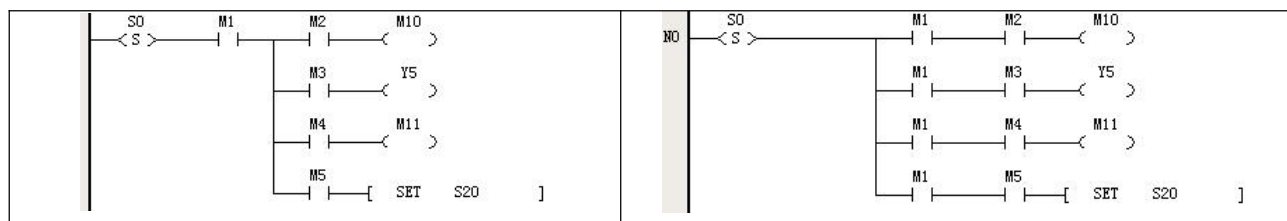
在同一个 PLC 程序中，用于顺序控制编程的每一个步进状态符都是对应于一个唯一的 S 元件的，不可重用。

在采用梯形图输入时必须注意这个要求。

2. 转移条件后再分支路

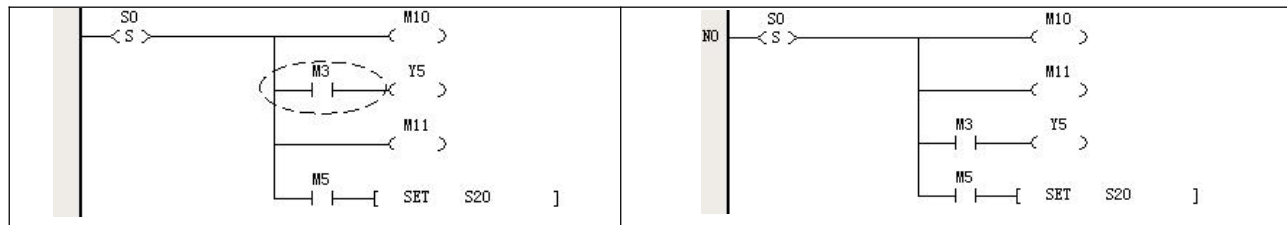
转移条件后不可再分出带条件的支路。如以下左图程序将不能通过编译，因为 M1 已经成为了转移条件，其后不能再分支。

应该修改为右图程序，可正确编译。



3. 常开常闭触点与输出线圈使用错误

当一个支路里使用了常开或常闭触点指令后，其后的支路里面的输出线圈不能直接连接到内部母线上，否则不能通过编译，如下左图所示。将支路顺序修改为右图所示，则可通过编译。

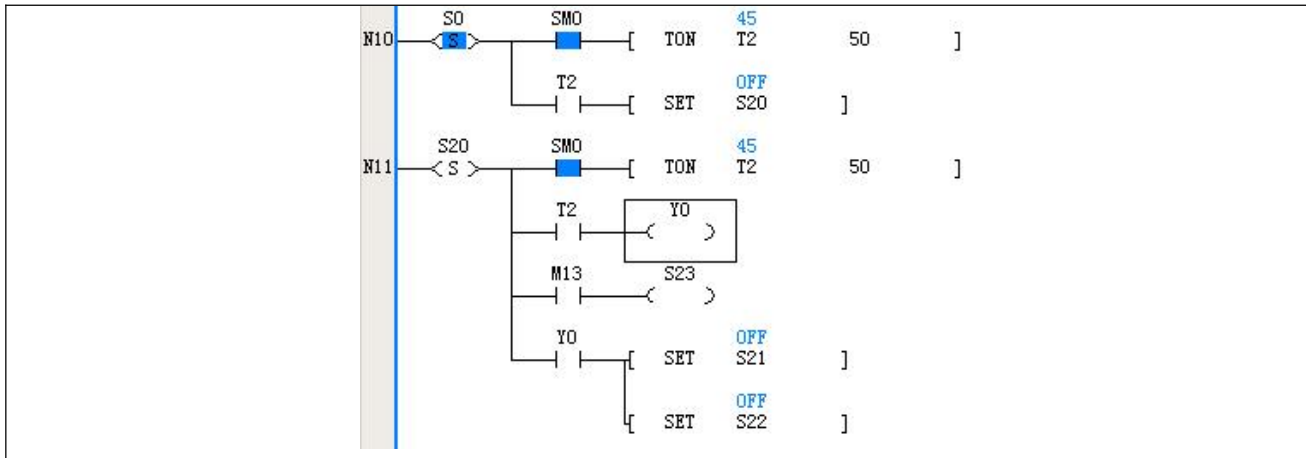


4. 在相邻的步进状态重复使用软元件

PLC 执行程序时是按指令顺序循环扫描的。当从上一步进状态转移到下一步进状态时，上一步进状态内的指令序列刚扫描结束，下一步进状态指令序列也已经开通扫描，形成控制输出。

根据上述分析可知，STL 指令在由 ON 变为 OFF 时，虽然会复位其内部的一些元件（见 5.3.1 STL：SFC 状态装载指令），但这个复位操作只能在下次扫描周期时进行。在步进状态转移的瞬间，上一步进状态内部元件仍然保持原有数据和状态直到下次扫描经过该步进状态。

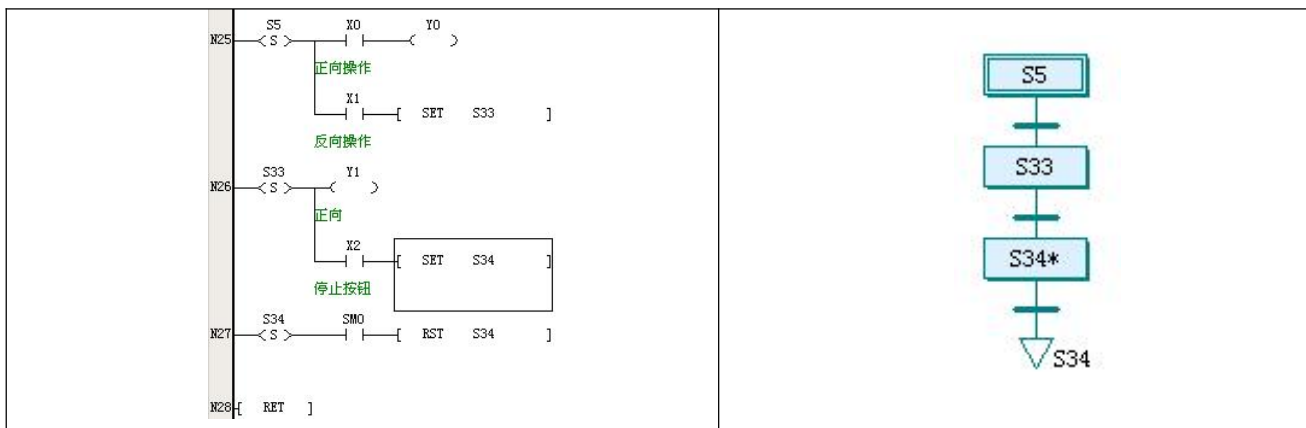
如下图所示，上下两个相联的步进状态中同时使用了 T2 计时器。当步进状态由 S0 转向 S20 时，T2 元件会保持计数值和接通状态。S20 步进状态因此不能按照用户最初的设计来执行计时操作，而是直接进入下面的 S21 和 S22 步进状态。因此，在不同的步进状态中，编程软元件虽然可以重复使用，但是最好不要用在相邻的步进状态中，否则可能会造成意外的结果。



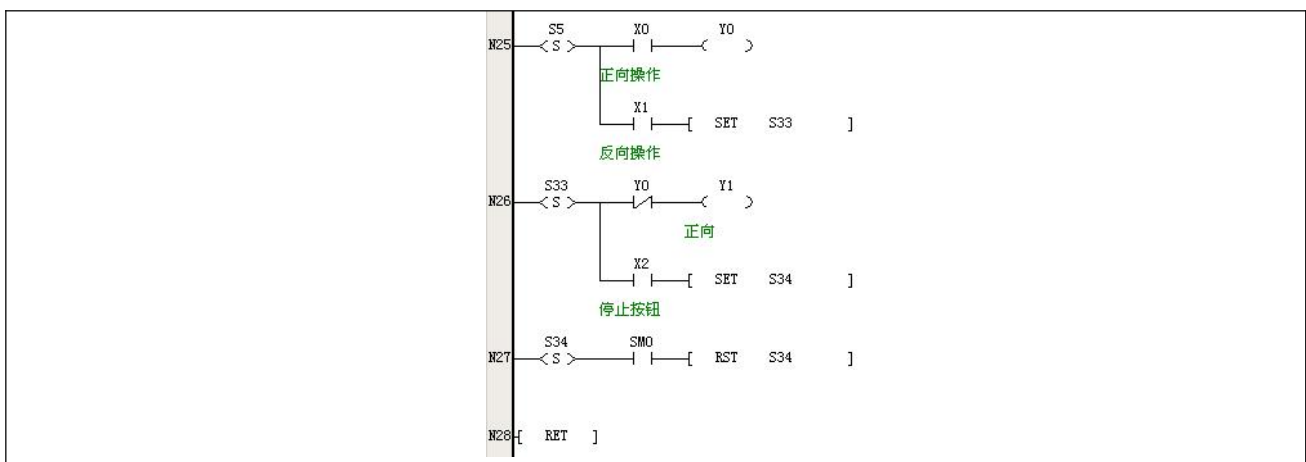
5. 软元件未能互锁

在 SFC 编程中，有些软元件之间可能因为步进状态转移时的特殊情况而出现矛盾。此时需要对其进行互锁。

例如下图的正反顺序操作程序示例。 $Y0$ 和 $Y1$ 分别是设备运行正向和反向控制输出。 $X0$ 为正向操作， $X1$ 为反向操作， $X2$ 为停止按钮。要求 $Y0$ 和 $Y1$ 互锁，即不能同时为 ON。然而在该例程中，当设备正向运行时，接通 $X1$ 使 $S5$ 步进状态转移到 $S33$ 步进状态的时刻， $Y0$ 和 $Y1$ 同时为 ON 且时间为一个程序扫描周期。



因此，程序中应该增加互锁的语句。如下为示例，在上图程序中 $Y1$ 输出线圈前增加 $Y0$ 的常闭触点以作为互锁。

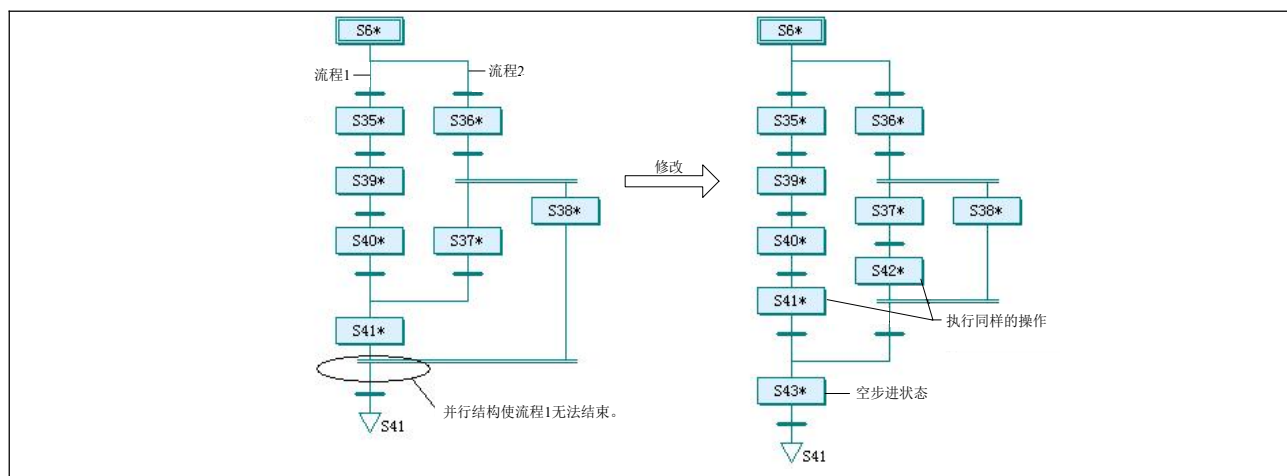


6. 跳转与转移混用

跳转多是应用在不同流程、不相邻步进状态之间切换。转移则是相邻步进状态之间切换的操作。如果将应该使用跳转的地方由 OUT 线圈改为 SET 语句，或者在应该使用转移的地方把 SET 语句改为 OUT 线圈，则都不能通过编译。

7. 选择分支转移处为并行分支汇合结构导致无法结束流程

选择分支是多选一的流程，如果其中混合了并行分支，则可能会发生导致选择分支运行无法结束的流程错误。如下图所示。左图程序中，流程 1 在执行到步进状态 S41 时，由于转移条件为并行分支，而此刻系统不会再运行流程 2，导致该处转移条件永远无法实现，从而出现流程错误。



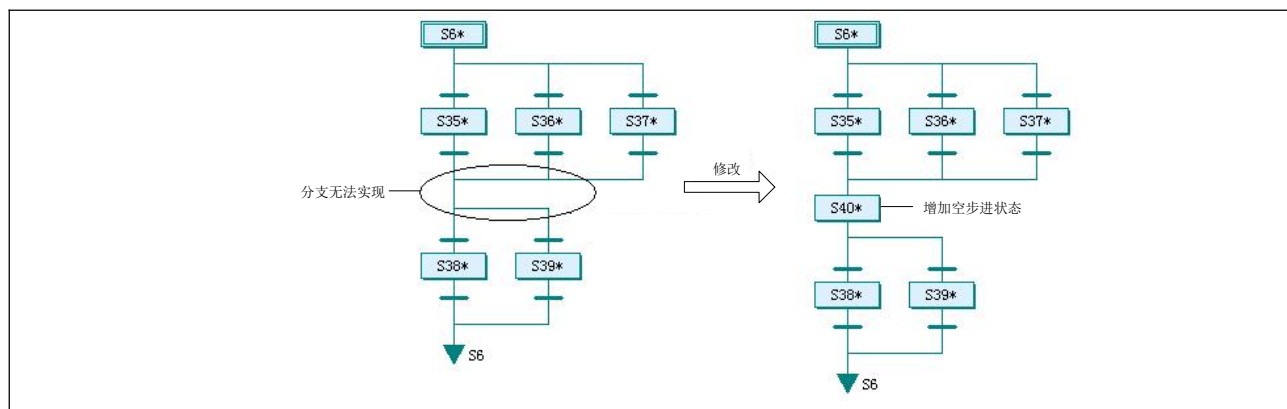
修改方法如右图所示，增加步进状态 S42，功能与 S41 完全一致；增加 S43 空步进状态，仅作为编程结构要素，没有实质操作。S38、S41、S43 的转移条件需要编程者设计，比如都采用原 S41 的转移条件即可。

7.4.2 编程技巧

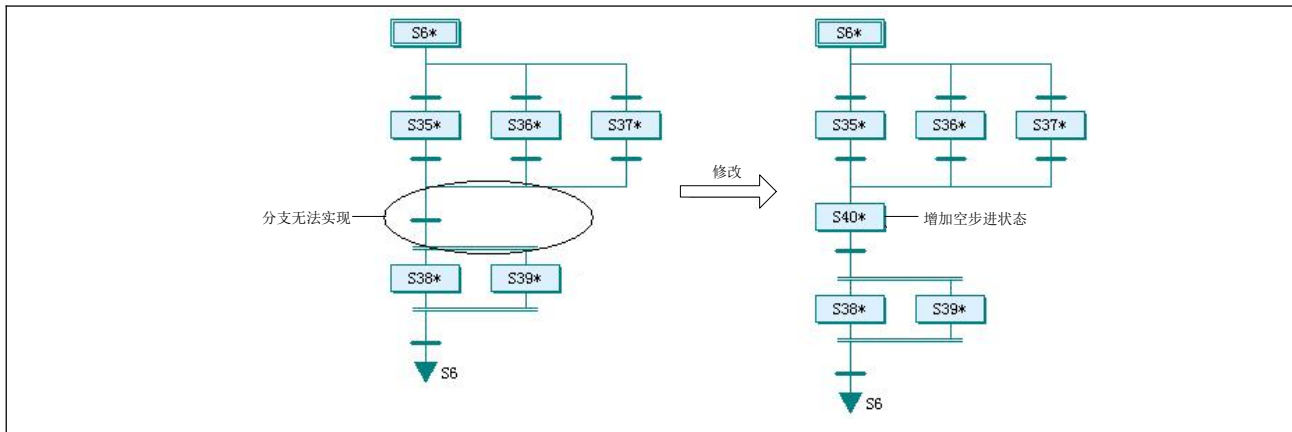
1. 巧用空步进状态

一些有语法问题的分支设计，需要采用空步进状态来解决分支难题。所谓的空步进状态，就是在步进状态中没有安排有实质内容的操作，直接等待进行转移。如下是一些示例。

下面左图中，选择分支汇合后立即接上另一个选择分支，这是不能编译通过的。可按照右图来修改，增加一个空步进状态。



下面左图中，选择分支后立即接上另一个并行也是不行的。可按照右图来修改，增加一个空步进状态即可。

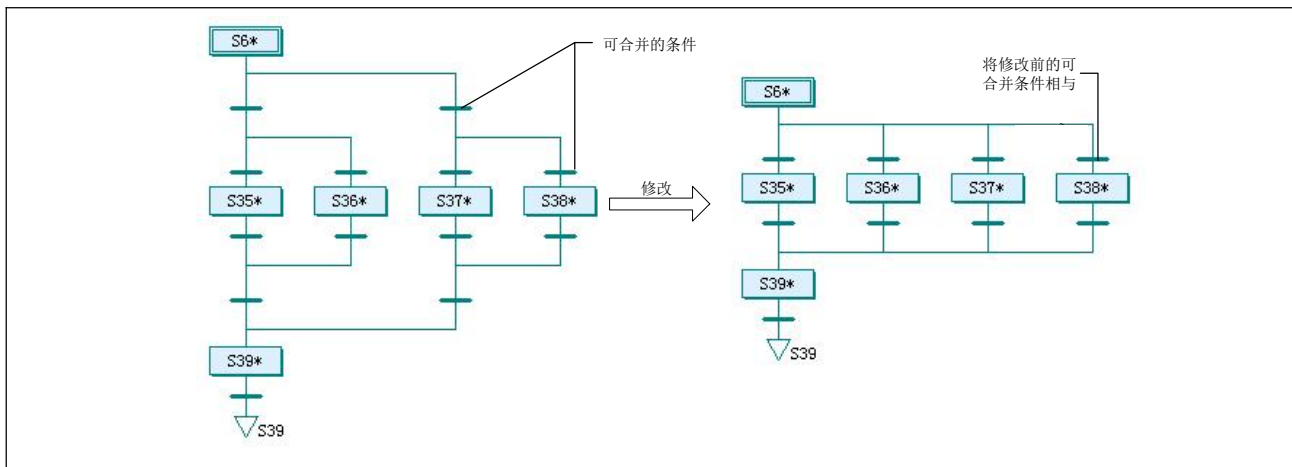


其它问题分支如并行汇合之后接并行分支、并行分支之后接选择分支，也可这种采用增加空步进状态的方式解决问题。

2. 合并分支和转移条件

一些看起来复杂的分支，实际上是设计时分析不当造成的，可以适当合并或简化。

如下图，设计者先做了第一个选择分支，然后分别再做两个选择分支。实际上只需要采用一个四条支路的选择分支即可，原设计的上下两级转移符则合并为转移条件相与的一级转移符。



3. 利用停电保持功能

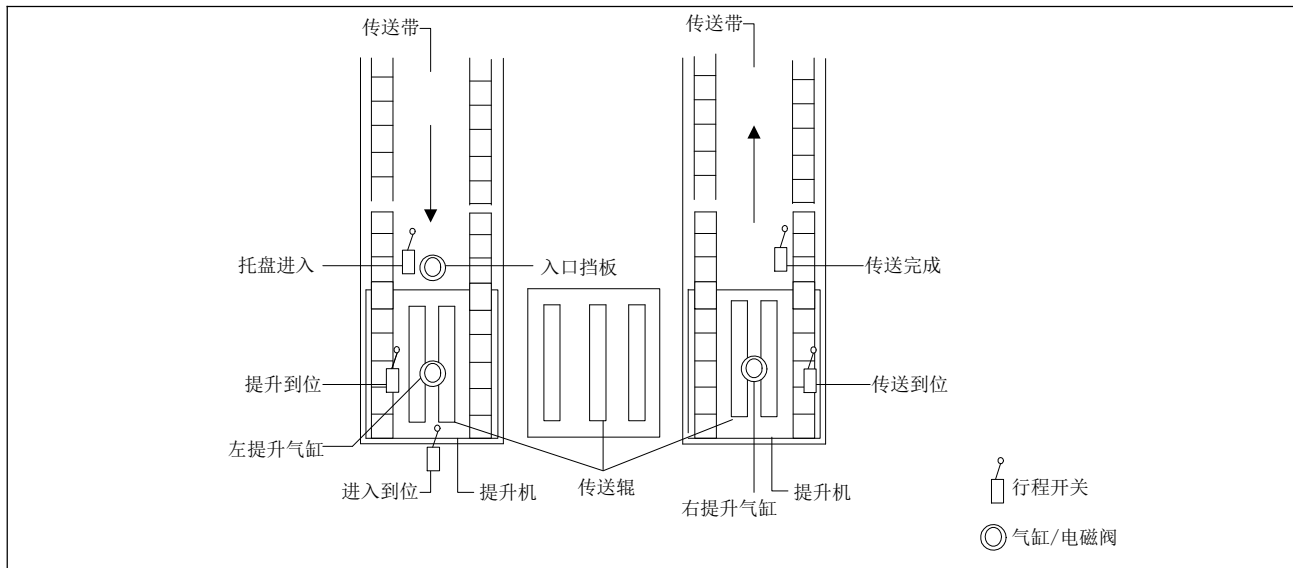
可用停电保持设置来保持 S 元件的值，在恢复上电后还可以从停电时的步进状态重新运行。

7.5 顺序功能图编程实例

本节中各示例只能作为简明的顺序功能图编程示范案例，操作和条件是经过简化的。设备配置设计表达的是粗略的概念，不能当作针对实际设备的设计，仅供学习参考。

7.5.1 简单结构流程

以下示例是一台工件托盘提升传送机。该传送机采用气缸提升装置和传送辊将工件托盘从一条传送带传送到另一条传送带上去。如下图为传送带和工件托盘提升传送机俯视图。

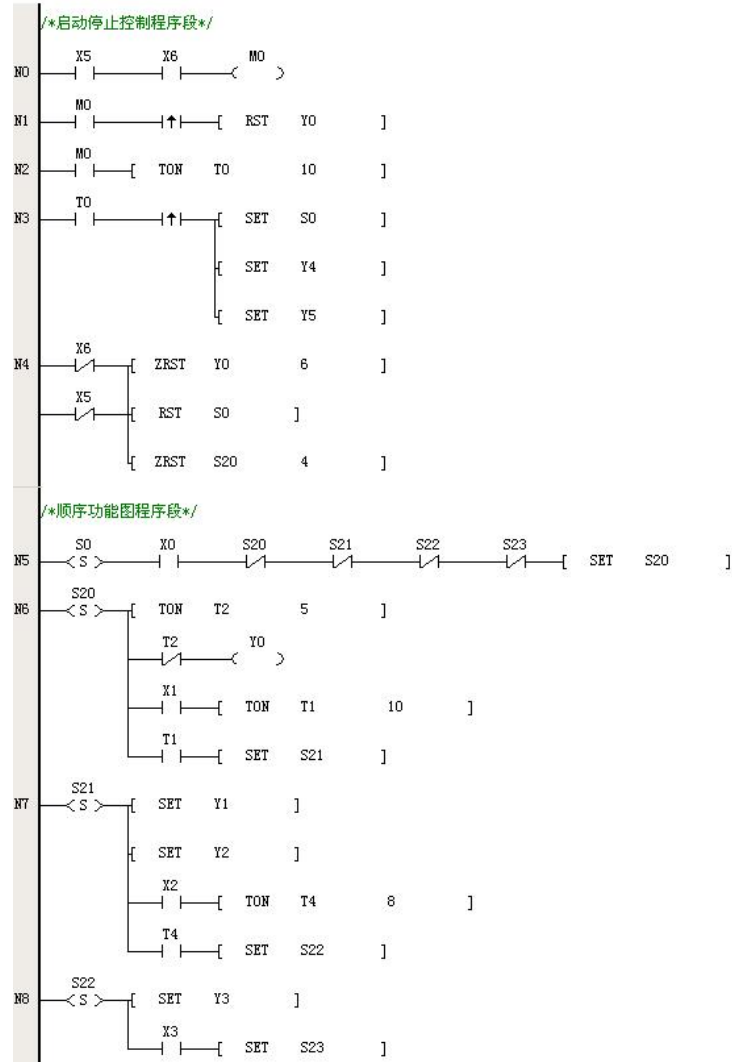
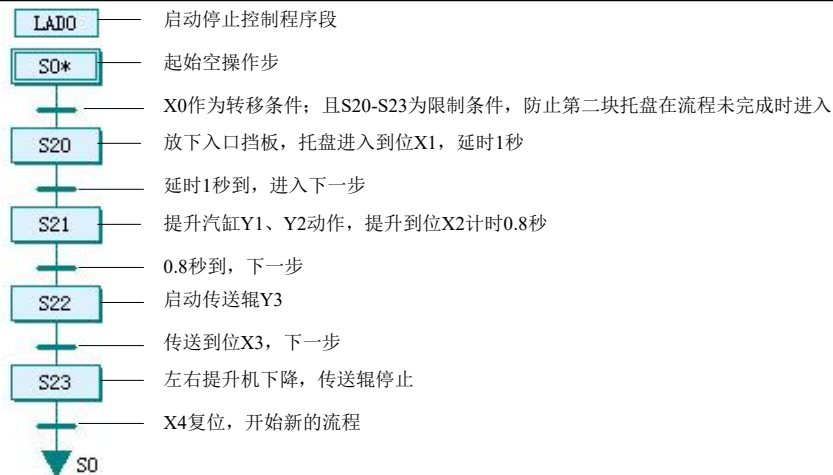


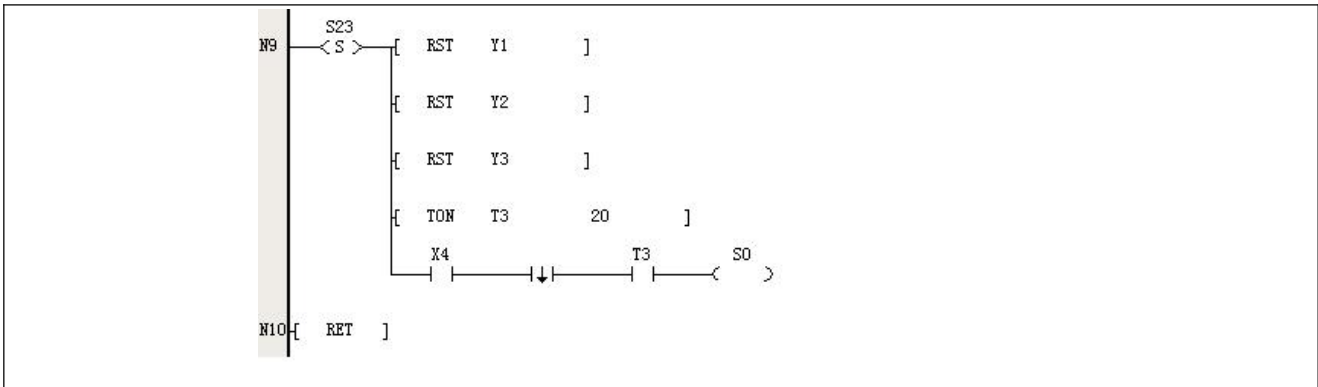
设备启动后，工件托盘沿左边的传送带传送到提升传送机入口，触动“托盘进入行程开关”。当整个传送机上没有工件托盘传送时，入口挡板下降，将工件托盘输送进入提升传送机。等到工件托盘完全进入左边的提升机，触动“进入到位行程开关”，则提升气缸动作，提升机升起，到位后会触动“提升到位行程开关”。传送辊电动机在提升到位后启动，将工件托盘传送到右边的提升机，到位后会触动“传送到位行程开关”，之后提升机气缸动作，提升机下降。工件托盘落到右边传送带上，被带离提升机。当传送完成行程开关复位时，一个完整的提升传送流程结束，随后进入下一个提升传送流程。下表为输入点输出点表。

序号	点地址	监控对象	序号	点地址	监控对象
1	X0	托盘进入行程开关	8	Y0	入口挡板气缸电磁阀
2	X1	进入到位行程开关	9	Y1	左提升机气缸电磁阀
3	X2	提升到位行程开关	10	Y2	右提升机气缸电磁阀
4	X3	传送到位行程开关	11	Y3	传送辊电机接触器
5	X4	传送完成行程开关	12	Y4	左传送带电机接触器
6	X5	启动开关	13	Y5	右传送带电机接触器
7	X6	紧急开关辅助信号			

可以看出这是一个简单顺序流程。每块托盘经过几个连续步骤被传送，没有其它选择或并行步骤，也没有并行流程。以顺序功能图设计程序，比较常规逻辑设计要简便快速，条理更清晰。

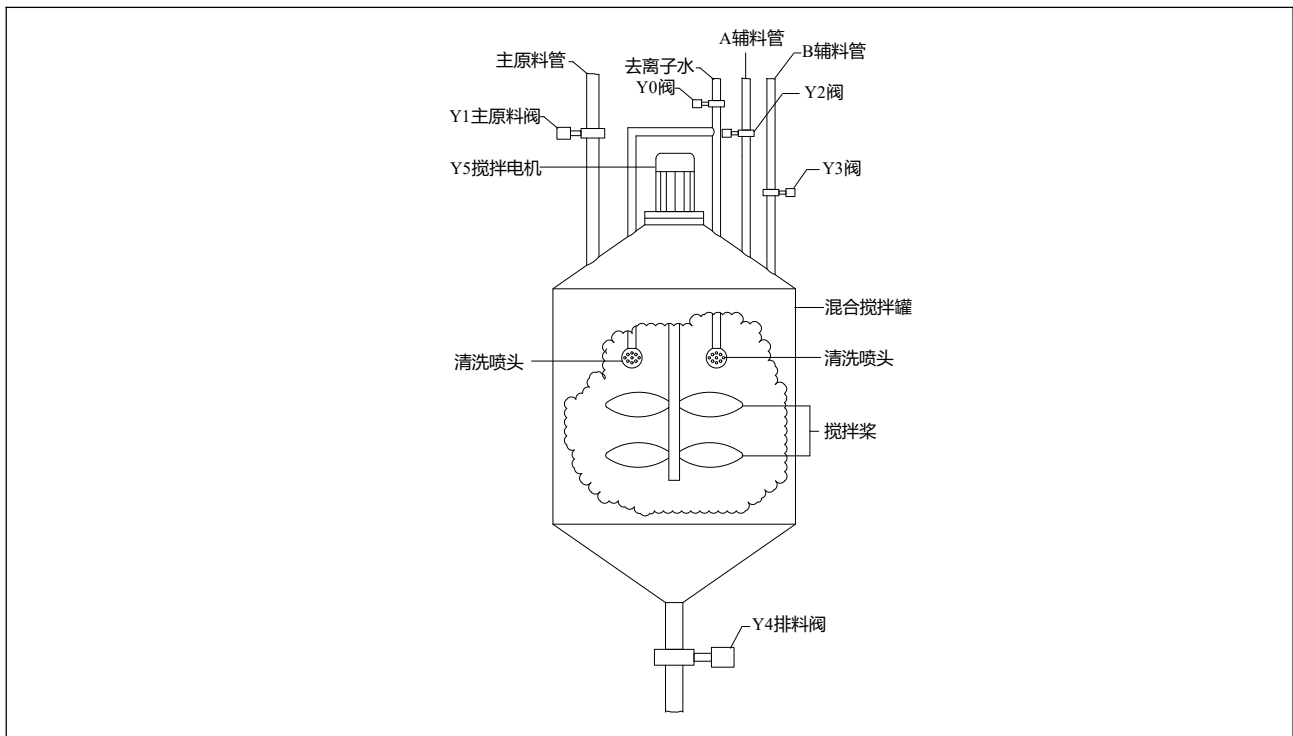
如下为顺序功能图程序，以及对照的梯形图程序。





7.5.2 选择结构

以下示例是一个物料混合加工工序流程。通过这个流程，可以生产 A、B 两种品种的产品。如下图所示为生产装置示意图。



运行时，第一步是通过触摸屏选择下一批次产品的品种 A 或 B，然后开始生产。第二步加主原料，重量达 2000kg 时，停止加料；第三步是添加辅助原料，生产 A 类型产品时添加 A 辅料 500kg，生产 B 类型产品时添加 B 辅料 500kg；第四步是搅拌 20 分钟；第五步是排料，当剩余物料少于 20kg 且延时时间到，则完成排料。这些都完成后，重新进入下一批次生产流程。

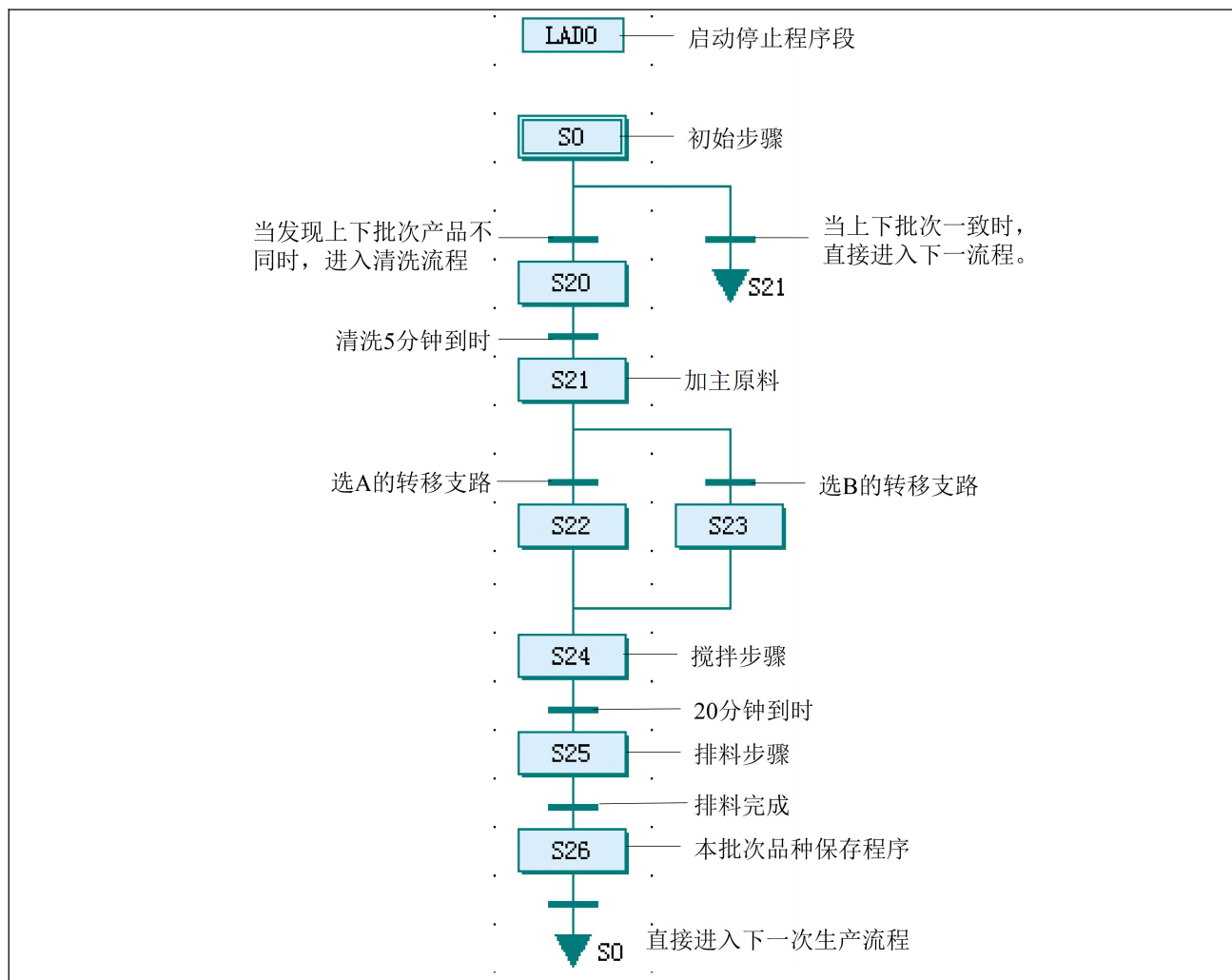
如果是第一次开机生产，或者上一批次产品品种与下一批次不同，则在加主原料前打开去离子水和排料阀，清洗 5 分钟。

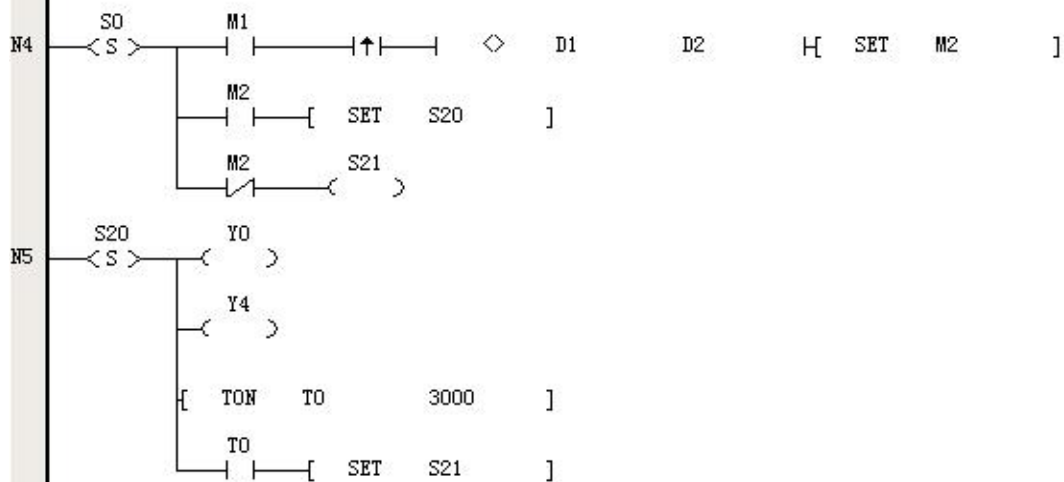
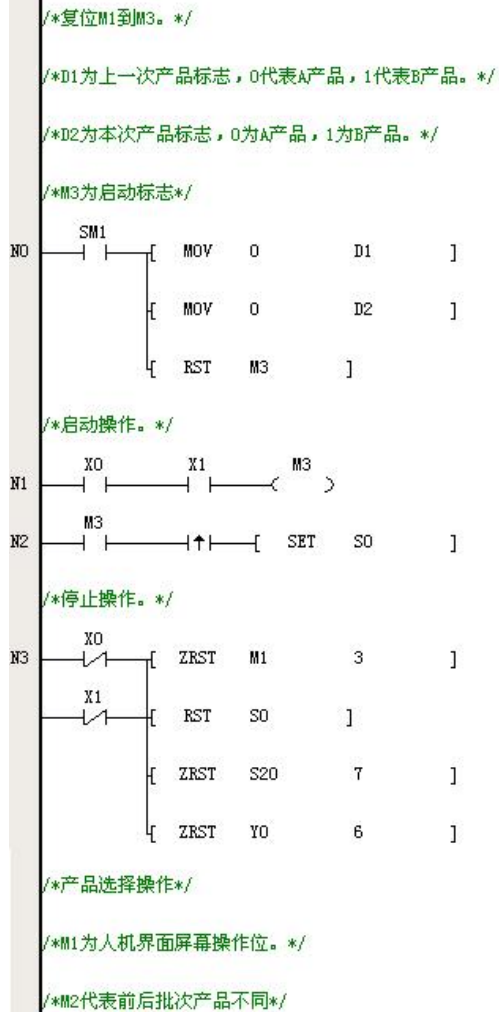
如下为输入点输出点表。

序号	点地址	监控对象	序号	点地址	监控对象
1	X0	去离子水阀开状态	10	X11	排料阀开状态
2	X1	去离子水阀关状态	11	X12	排料阀关状态
3	X2	主原料阀开状态	12	Y0	去离子水电磁阀
4	X3	主原料阀关状态	13	Y1	主原料电磁阀
5	X4	A 辅料阀开状态	14	Y2	A 辅料电磁阀
6	X5	A 辅料阀关状态	15	Y3	B 辅料电磁阀
7	X6	B 辅料阀开状态	16	Y4	排料电磁阀
8	X7	B 辅料阀关状态	17	Y5	搅拌电机接触器
9	X10	搅拌电机运行状态			

可见这是一个选择结构的流程，生产产品时，只能从 A 或 B 中选择一种。生产完毕后才可能切换品种。同时，流程中还有一个选择和跳转的结构，即清洗步骤。

如下为顺序功能图程序，以及对照的梯形图程序。

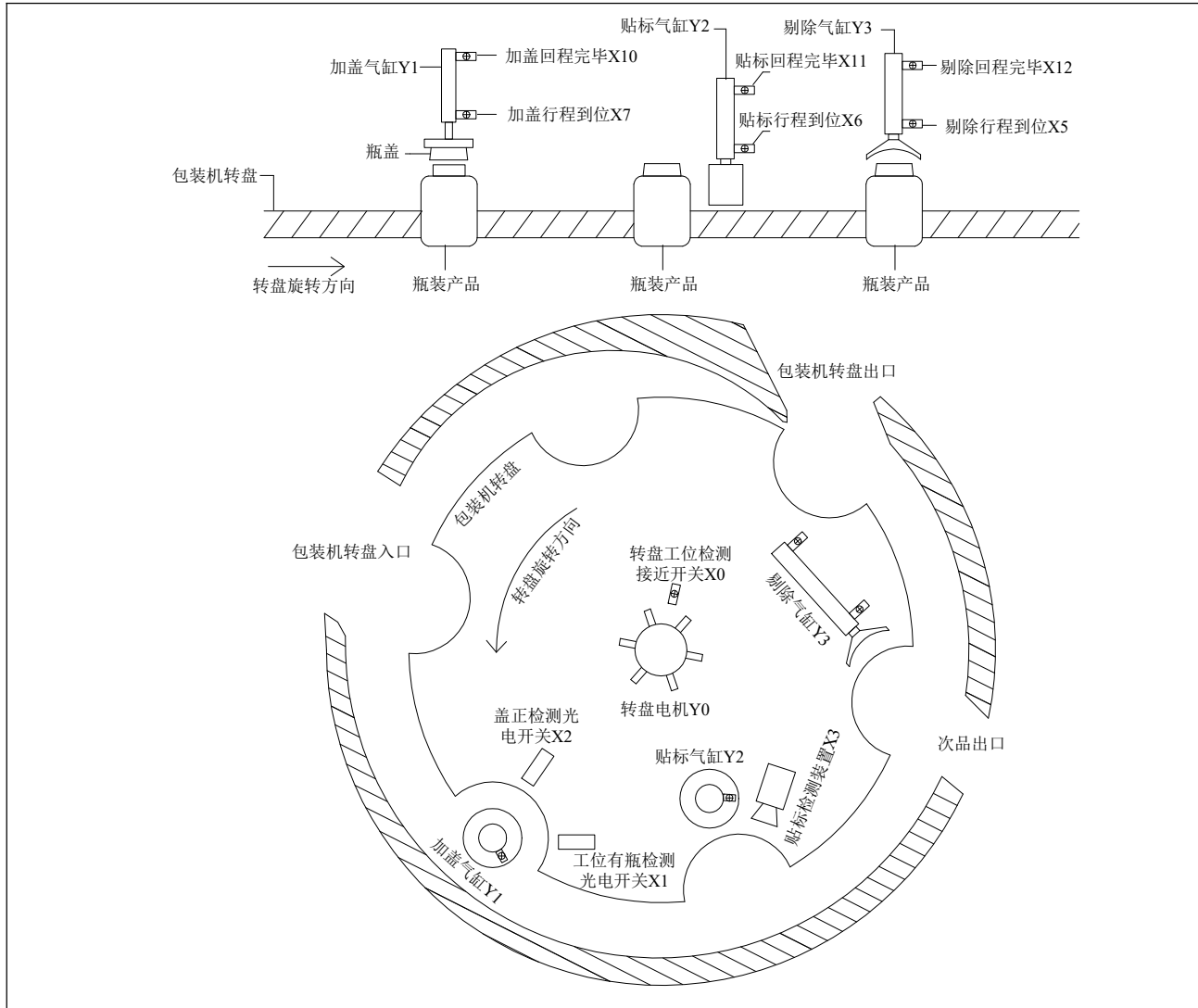






7.5.3 并行结构

以下示例是一个瓶装产品包装机。该包装机是将瓶装产品加上封盖，然后贴上产品标签。在这过程中，对瓶盖和标签进行检测，检测到有问题的产品则通过后一步剔除装置来剔除，正品可直接送下道工序。如果上道工序没有瓶子送过来，则相关的加盖、贴标工序都不动作。三道工序同时进行，转盘一次走一个工位。如下图为生产装置示意图。



运行时，转盘 1 次走 1 个工位，由 X0 接近开关检测。在每个工位转盘都会停顿直到所有操作完成。加盖、贴标、剔除机构均由气缸驱动，并分别检测气缸行程到位和气缸回程完毕信号。

如下所示为输入点输出点表：

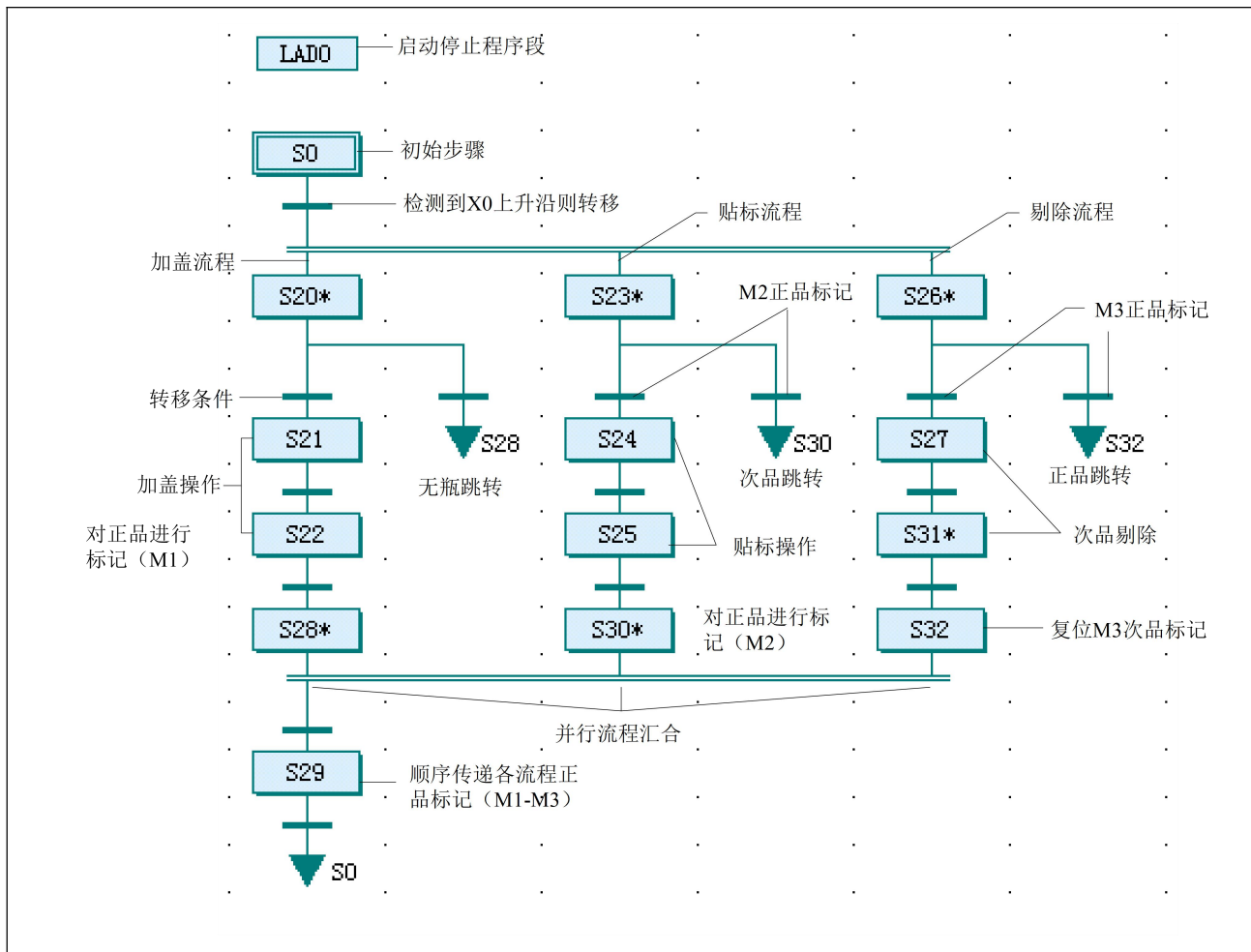
序号	点地址	监控对象	序号	点地址	监控对象
1	X0	转盘工位检测接近开关	8	X10	加盖回程完毕
2	X1	工位有瓶检测光电开关	9	X11	贴标回程完毕
3	X2	盖正检测光电开关	10	X12	剔除回程完毕
4	X3	贴标检测装置	11	Y0	转盘电机
5	X5	剔除行程到位	12	Y1	加盖气缸
6	X6	贴标行程到位	13	Y2	贴标气缸
7	X7	加盖行程到位	14	Y3	剔除气缸

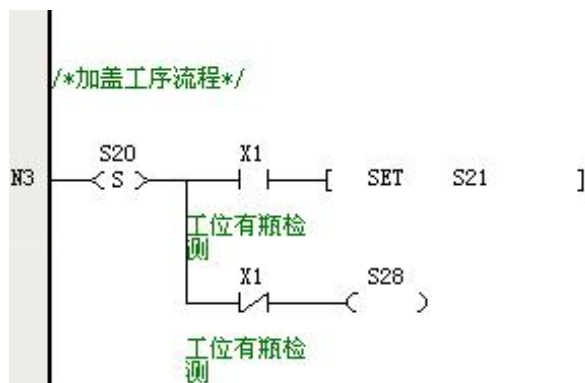
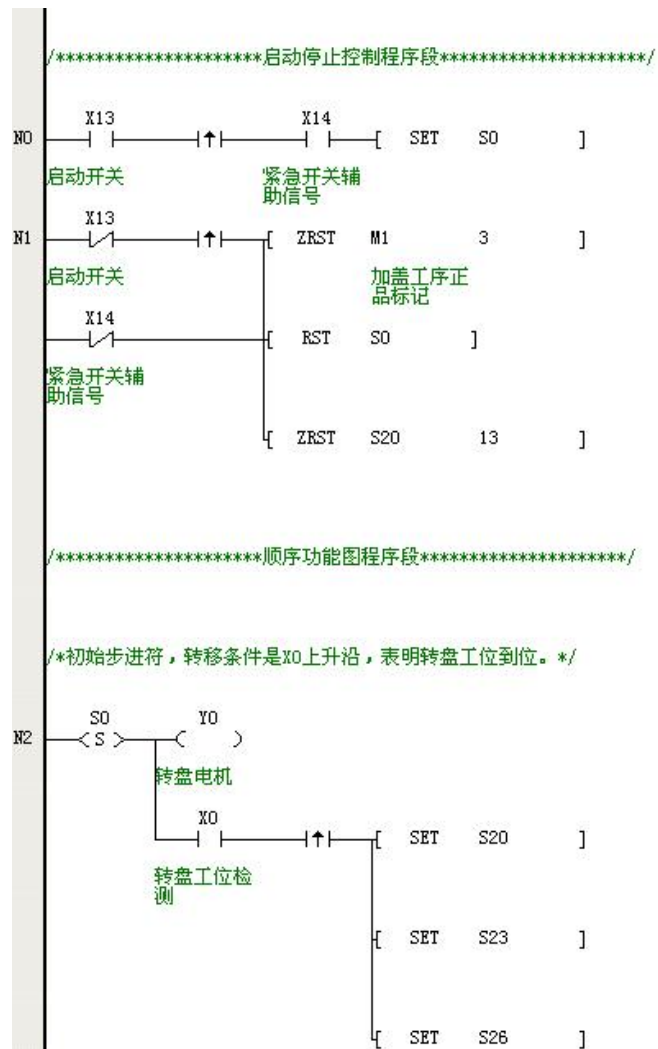
通过分析生产流程，可知这是一个并行分支结构的流程，在转盘转动步骤完成后，三个工位的操作并列进行，直到全部操作完毕，设备才会进行下一步操作。如下为顺序功能图程序，以及对照的梯形图程序。

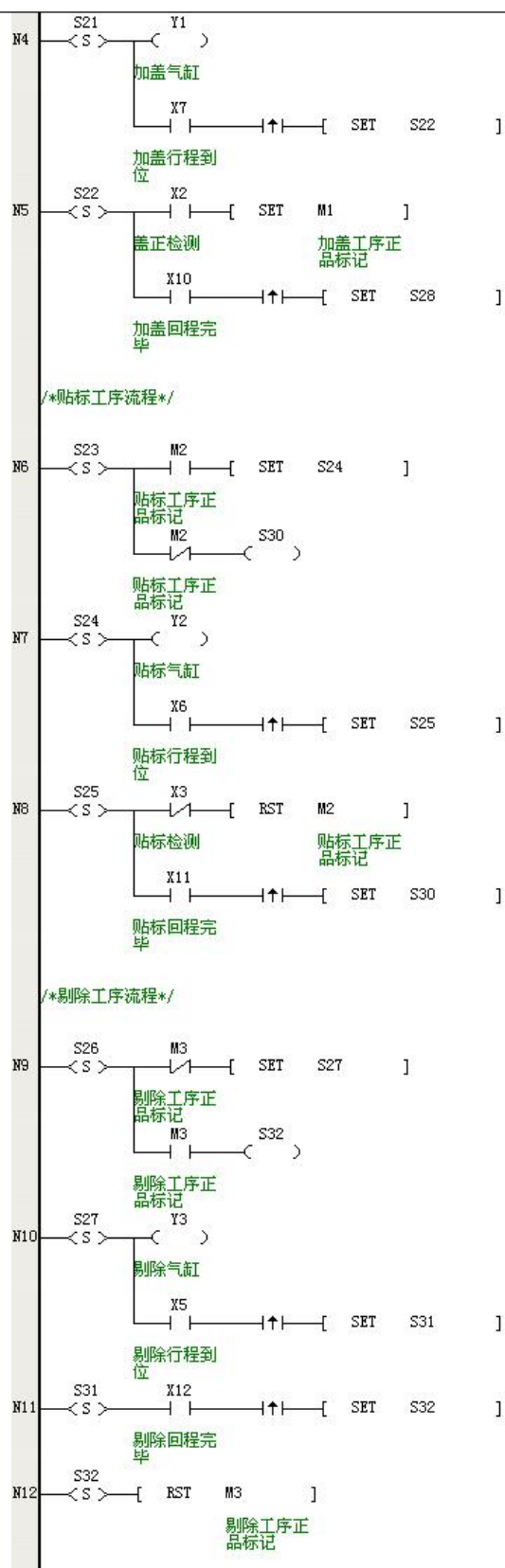
程序中，M1-M3 分别为加盖、贴标、剔除这三个工序的正品标志。

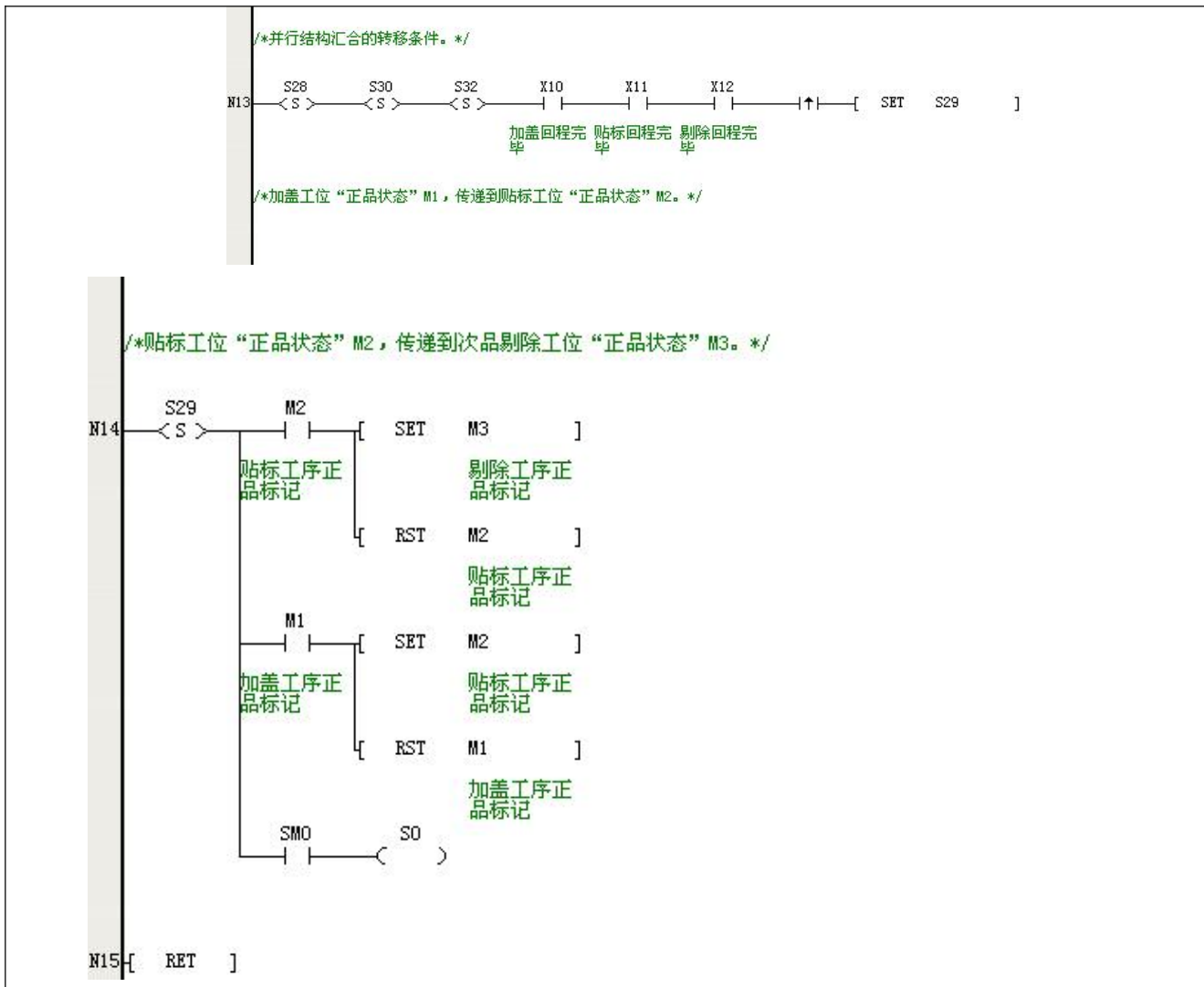
当加盖流程进行到 S22 时，通过 X2 检测加盖是否成功（盖正），盖正则置位 M1，表明加盖工序产生了正品；当贴标流程进行到 S25 时，通过 X3 检测标签是否贴上，未正确贴上时则复位 M2，表明贴标工序产生次品；全部工序结束时，在 S29 步骤，将 M2 状态传递给 M3，再将 M1 状态传递给 M2。

加盖流程根据 X1 检测是否有瓶到位，无瓶到位则不执行加盖操作；贴标流程开始运行时，M2 为 ON 时表明加盖工序的正品传到了，进行贴标操作，M2 为 OFF 表明次品到位，不执行贴标操作；剔除流程根据 M3 标志来选择执行，为 ON 时表明正品，不执行剔除操作，为 OFF 时则剔除次品，这两种情况运行到 S32 步骤时复位 M3 以等待下一步骤和工序。









第八章 高速输入输出

APLC 系列不支持此功能。

第九章 中断程序使用指南

本章详细讲述了各种中断的机理、处理过程与使用方法。

9.1 理解中断程序	278
9.2 中断事件的处理过程	279
9.3 使用定时中断	279
9.4 使用外部中断	282
9.5 电源失电中断	284
9.6 使用串口中断	284

9.1 理解中断程序

当中断事件发生时，正常扫描周期将被打断，中断程序将被优先调用执行，这就是中断处理机制。对于时效性要求高、事件触发型的控制任务，用户常常需要采用这种特殊的系统处理机制。

系统为用户提供了多种可编程的中断资源，每种中断资源都会触发一类中断事件，每类中断事件都被独立编号。

为处理特定的中断事件，用户必须编写相应的处理程序，即中断程序，是构成用户程序的独立的程序体。

每一个中断程序必须指定一个中断事件编号，这样用户中断程序和指定了事件编号的中断事件就建立了一一对应的关系。当该类中断事件的中断请求被响应时，系统根据中断事件编号调用相应的用户中断程序。

以下是 APLC 系列小型 PLC 提供的中断资源列表：

中断事件编号	对应中断事件	中断使能控制 SM	中断事件编号	对应中断事件	中断使能控制 SM
0	X0 输入上升沿中断	SM40	20	高速计数器中断 0	SM65
1	X1 输入上升沿中断	SM41	21	高速计数器中断 1	SM65
2	X2 输入上升沿中断	SM42	22	高速计数器中断 2	SM65
3	X3 输入上升沿中断	SM43	23	高速计数器中断 3	SM65
4	X4 输入上升沿中断	SM44	24	高速计数器中断 4	SM65
5	X5 输入上升沿中断	SM45	25	高速计数器中断 5	SM65
6	X6 输入上升沿中断	SM46	26	定时中断 0	定时设定值:SD66 使能控制: SM66
7	X7 输入上升沿中断	SM47	27	定时中断 1	定时设定值:SD67 使能控制: SM67
10	X0 输入下降沿中断	SM40	28	定时中断 2	定时设定值:SD68 使能控制: SM68
11	X1 输入下降沿中断	SM41	29	电源失电中断	SM56
12	X2 输入下降沿中断	SM42	30	通讯口 0 的字符发送中断	SM48
13	X3 输入下降沿中断	SM43	31	通讯口 0 的字符接收中断	SM49
14	X4 输入下降沿中断	SM44	32	通讯口 0 的帧发送中断	SM50
15	X5 输入下降沿中断	SM45	33	通讯口 0 的帧接收中断	SM51
16	X6 输入下降沿中断	SM46	34	通讯口 1 的字符发送中断	SM52
17	X7 输入下降沿中断	SM47	35	通讯口 1 的字符接收中断	SM53
18	PTO (Y0) 输出完成中断	SM63	36	通讯口 1 的帧发送中断	SM54
19	PTO (Y1) 输出完成中断	SM64	37	通讯口 1 的帧接收中断	SM55

9.2 中断事件的处理过程

1. 当某一中断事件发生，且该中断事件已被使能，该中断事件编号将被加入到中断请求队列记录中，中断请求队列是一个深度为 8 的先入先出队列。
2. 系统处理中断请求：
 - (1) 系统检测到中断请求队列不为空时，系统打断正常的用户程序的执行流程。
 - (2) 系统查询中断请求队列的队首记录，队首记录着最先发生的中断事件的编号，该中断事件编号所对应的用户中断程序将被调用执行。
 - (3) 当对应的中断程序被执行完毕时（执行了中断返回指令），该中断请求被处理完毕，请求队列的队首记录将被从队列中删除，请求队列中次先记录变为队首记录，后续的记录也将向前移动一个位置。
 - (4) 系统再次检测到中断请求队列是否为空，不为空的话，从头循环执行以上步骤，直至中断请求队列为空。
 - (5) 当中断请求队列为空时，系统回到被打断的主程序的执行流程继续执行。
3. 系统一次只处理一个中断请求，如系统正处理某一中断请求，新近发生的中断事件不会被立即响应，但将被记录到中断请求队列的队尾，等待系统处理完之前的中断请求后，再处理该中断请求。
4. 当中断请求队列中的记录数达到 8 时，系统将自动屏蔽新的中断事件，新的中断事件不能加入中断队列。直至中断请求队列中所有请求被处理完毕，而且被打断的主程序也被执行完毕，系统才解除屏蔽。

注意

1. 中断程序时间不应过长，否则将导致其它中断事件被屏蔽（中断请求丢失），系统扫描时间过长，主程序的执行效率低下等异常情况。
2. 中断程序中禁止调用其他用户子程序。
3. 要想在中断中立即刷新 I/O，请使用立即刷新指令（REF），注意 REF 执行时间跟所要刷新 I/O 点数有关。
4. 要使某类中断事件产生中断请求，应确保中断请求所属的中断事件标志应被使能（每类中断事件使能/禁止控制，都有相关的 SM 元件。如要使能某类中断事件，应将相应的 SM 元件置为 ON）且全局中断使能标志是被打开的。
5. 如产生了相应的中断请求，但用户程序中没有对应的中断程序与之对应，系统也将响应该中断请求，但只做空操作。

9.3 使用定时中断

定时中断的说明

定时中断是按照设定的定时值，系统定时地产生的中断事件。

定时中断程序主要应用于要求定时处理，且要求系统及时处理的场合，如定时采样模拟输入量、定时按一定波型刷新模拟输出量。

设定相应 SD 元件的值，可设定中断定时值，所设定定时值的计时单位为毫秒（ms），当定时值小于等于设定值时，系统不触发中断事件（中断定时值的最小设定值，推荐大于 4ms）。

设定相应 SM 元件 ON/OFF 状态，可使能/禁止定时中断。

APLC 系列 PLC 给用户提供了 3 个定时中断资源，如下所示：

表 9-1 定时中断资源一览表

定时中断	中断事件号	定时设定值 (SD)	使能控制 (SM)
0	26	SD66	SM66
1	27	SD67	SM67
2	28	SD68	SM68

注意

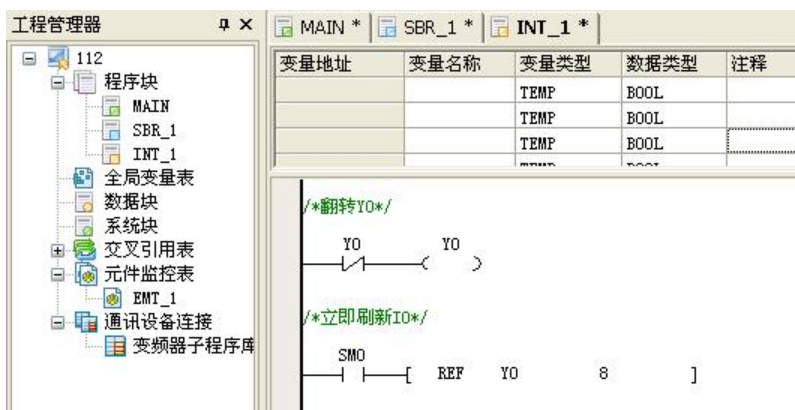
1. 当禁止定时中断时，已加入中断队列中的定时中断仍被执行。
2. 当定时中断被禁止后再使能时，定时的计时将从零开始。

如果想在程序运行下，改变中断定时值的设定，建议按照如下步骤操作：先禁止定时中断，改变定时值的设定，再使能该中断。

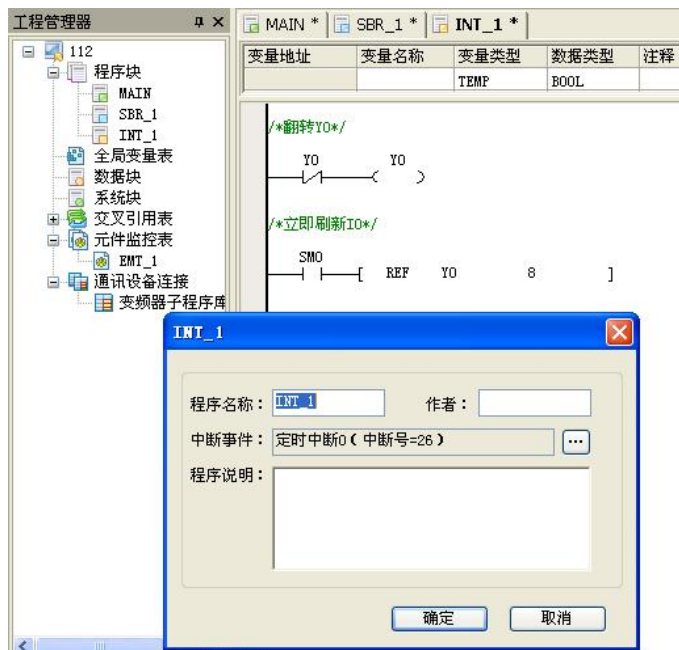
定时中断的使用示例

该示例是利用定时中断 0 功能，一秒钟翻转 Y0 输出一次，使 Y0 产生定时闪烁的效果。

1. 编写中断程序，写出定时到来，中断触发时的处理代码。



2. 为该中断程序指定对应的中断事件编号：



3. 在主程序中，编写设置定时中断和使能定时中断的代码。

112

程序块

MAIN

SBR_1

INT_1

全局变量表

数据块

系统块

交叉引用表

元件监控表

EMT_1

通讯设备连接

变频器子程序库

MAIN *

SBR_1 *

INT_1 *

变量地址	变量名称	变量类型	数据类型
		TEMP	BOOL
		TEMP	BOOL
		TEMP	BOOL
		TEMP	BOOL

```

/*设定定时中断的时间为1S（1000ms）*/
SM1
| | [ MOV 1000 SD66 ]

/*使能定时中断，置SM66状态ON，打开允许中断标志*/
SM1
| | [ SET SM66 ]

/*允许已使能的定时中断，产生中断请求*/
SM0
| | [ EI ]

```

9.4 使用外部中断

外部中断说明

外部中断和 PLC 的实际输入点有关，对外部的中断分为输入上升沿中断或输入下降沿中断。在用户程序中与外部事件有关的动作放到外部中断程序中，系统对外部信号的响应频率最高为 1k。超过 1k 以上的外部事件有可能丢失。对同一端口不能同时使用上升沿和下降沿中断。所有的外部中断只用在总中断控制 EI 有效和对应的中断使能 SM 有效的情况才能有效。具体关系如下：

中断号	对应的使能位	中断号	对应的使能位
0 或 10	SM40	4 或 14	SM44
1 或 11	SM41	5 或 15	SM45
2 或 12	SM42	6 或 16	SM46
3 或 13	SM43	7 或 17	SM47

体关系如下：

外部中断编号如下：

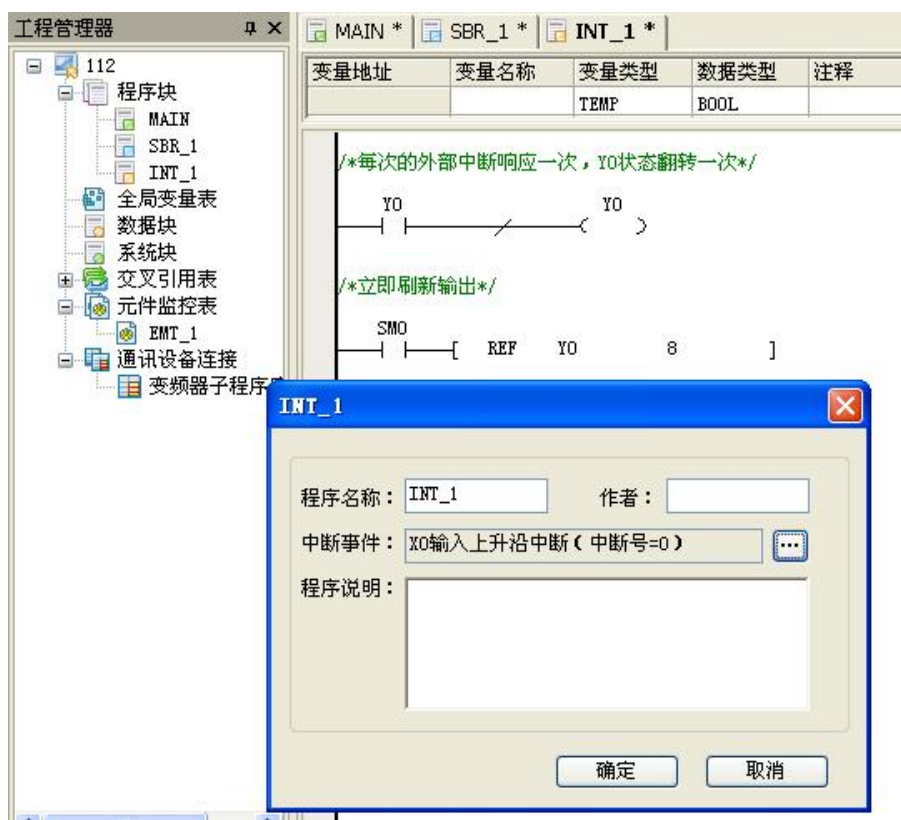
中断编号	对应中断源	中断编号	对应中断源
0	X0 输入上升沿中断	9	保留
1	X1 输入上升沿中断	10	X0 输入下降沿中断
2	X2 输入上升沿中断	11	X1 输入下降沿中断
3	X3 输入上升沿中断	12	X2 输入下降沿中断
4	X4 输入上升沿中断	13	X3 输入下降沿中断
5	X5 输入上升沿中断	14	X4 输入下降沿中断
6	X6 输入上升沿中断	15	X5 输入下降沿中断
7	X7 输入上升沿中断	16	X6 输入下降沿中断
8	保留	17	X7 输入下降沿中断

对 X0 – X7 的单个输入脉冲频率<200Hz。

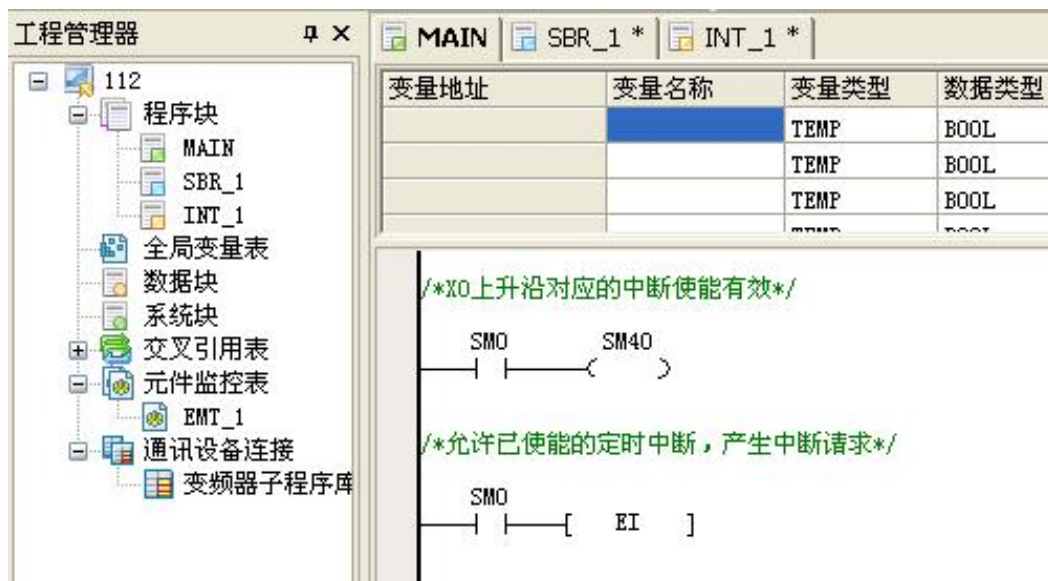
外部中断使用示例

该示例是利用 X0 对应的外部中断 0 功能，跟据 X0 的上升沿输入事件对 Y0 翻转输出。

1. 编写中断程序，每进一次中断 Y0 的状态就翻转一次，同时马上输出。对每个中断都要选择他所对应的中断号，具体操作如下图所示。



2. 在主程序里写入 EI 指令，并使 X0 输入上升沿中断对应的中断使能 SM40 有效。



9.5 使用电源失电中断

在使能标志 **SM56** 置位情况下，当主模块检测到电源失电的情况下，触发电源失电中断，用户可以在该中断子程序内进行相关处理。此功能只适用于 **AT100** 系列。

由于电源失电中断子程序执行在系统没有外供电源的情况下，所以请控制电源失电中断子程序的执行时间不能超过 **5ms**，否则会导致掉电保持元件无法完全保存的情况出现。

9.6 使用串口中断

串口中断说明

当串口在自由口协议模式下，根据串口的发送、接收事件，系统将产生中断事件。

对于每一个串口，系统给用户提供了 **4** 个中断资源，串口中断程序主要应用于要求对串口的字符与帧接收、发送操作做特殊处理，且要求系统及时处理的场合，可不受扫描时间影响立即响应发送接收完字符或帧的一些处理。

设定相应 **SM** 元件 **ON/OFF** 状态，可使能/禁止串口中断，当禁止串口中断时，已加入中断队列中的串口中断仍被执行。在字符发送中断处理子程序中请不要在常通能流后调用串口发送指令（**XMT**），这样可能会导致中断子程序嵌套而阻塞用户程序的执行。

帧接收、帧发送中断是指在串口发送指令（**XMT**）、串口接收指令（**RCV**）完成后出发的中断事件。

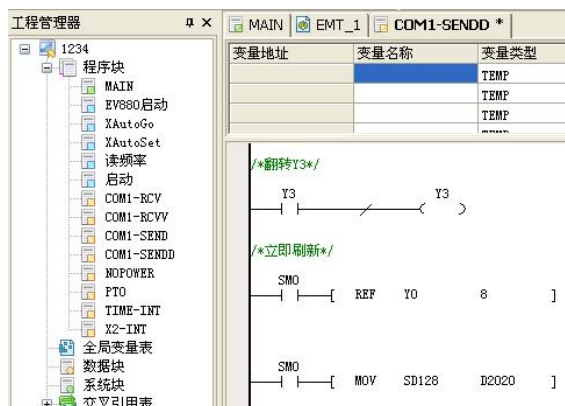
串口中断资源一览表：

中断事件编号	对应中断事件	中断使能控制 SM 元件编号
30	通讯口 0 的字符发送中断	SM48
31	通讯口 0 的字符接收中断	SM49
32	通讯口 0 的帧发送中断	SM50
33	通讯口 0 的帧接收中断	SM51
34	通讯口 1 的字符发送中断	SM52
35	通讯口 1 的字符接收中断	SM53
36	通讯口 1 的帧发送中断	SM54
37	通讯口 1 的帧接收中断	SM55

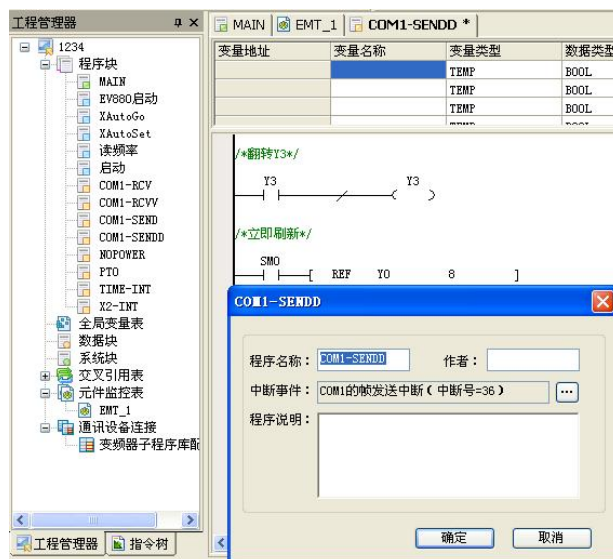
串口中断使用示例

该示例是利用串口帧发送中断功能，每一个帧发送完成翻转 Y3 输出一次，使 Y3 根据字符发送帧的频率产生闪烁的效果。

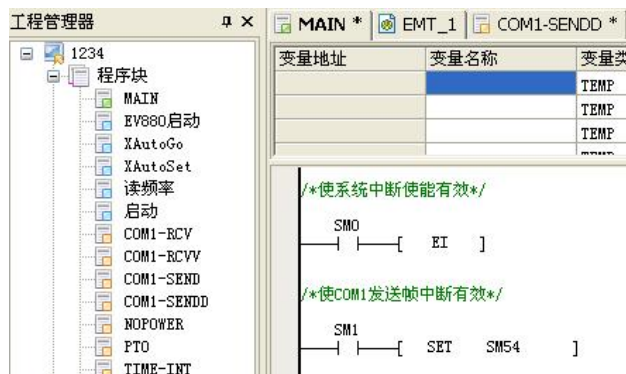
1. 编写中断程序，写出串口发送帧完成，中断触发时的处理代码。



2. 为该中断程序指定对应的中断事件编号：



3. 在主程序中，编写使能串口发送帧中断的代码。



详细串口中断使用例子请参看第十章 通讯功能使用指南。

第十章 通讯功能使用指南

本章详细介绍了 APLC 系列小型 PLC 的通讯功能，包括通讯资源、通讯协议，并通过实例加以说明。

10.1 通讯资源	287
10.2 编程口通讯协议	287
10.3 自由口通讯协议	287
10.3.1 简介	287
10.3.2 自由口参数设置	287
10.3.3 自由端口指令	289
10.4 Modbus 通讯协议	291
10.4.1 简介	291
10.4.2 链路特性	291
10.4.3 RTU 传输模式	291
10.4.4 ASCII 传输模式	291
10.4.5 支持的 Modbus 功能码	292
10.4.6 PLC 元件的寻址方式	292
10.4.7 Modbus 从站	293
10.4.8 元件的读写	293
10.4.9 对双字元件的处理	294
10.4.10 对 LONG INT 的处理	295
10.4.11 诊断功能码	295
10.4.12 异常码	295
10.4.13 Modbus 参数设置	296
10.4.14 Modbus 指令	296
10.5 ModbusTCP 通讯协议	297
10.5.1 简介	297
10.5.2 网络设置	297

10.1 通讯资源

APLC 系列小型 PLC 适用的波特率：

通信协议	适用的波特率
自由□协议，Modbus 通讯协议	38400, 19200, 9600, 4800, 2400, 1200
ModbusTCP 通讯协议（从站）	

APLC 系列小型 PLC 支持的通信协议：

	通讯□	通讯□类型	支持的协议
APLC-TCP	通讯□ 0	USB	编程□协议
	通讯□ 1	RS485	自由□协议，Modbus 通讯协议（主站、从站）
	通讯□ 2	RS485	自由□协议，Modbus 通讯协议（主站、从站）
	网□ 1	RJ45	ModbusTCP（从站）

10.2 编程□通讯协议

编程□协议是上位机软件 YD-station 与主模块通讯的专用协议，不对外开放。

10.3 自由□通讯协议

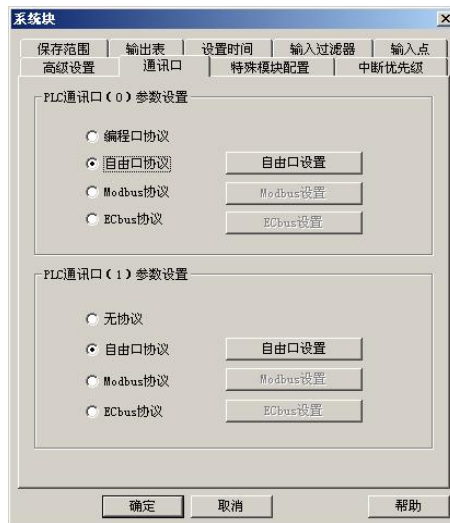
10.3.1 简介

自由□协议是用户定义数据文件格式的通讯方式，可由指令实现数据的发送和接收。自由端口协议支持 ASCII 和二进制两种数据格式。只有在 PLC 位于 RUN（运行）模式时才能使用自由端口通讯。

自由端口的通讯指令包括 XMT（自由□发送指令）、RCV（自由□接收指令）。

10.3.2 自由□参数设置

在系统块中选择**通讯□**选项，在 **PLC 通讯□（1）参数设置**或 **PLC 通讯□（2）参数设置**中选择**自由□协议**，从而激活相应的**自由□设置**按钮，如下图所示：



点击任意一个自由口设置按钮，进入自由口参数设置界面，如下图所示：



可配置内容如下：

选项	设置内容	注释
波特率	38400, 19200, 9600, 4800, 2400, 1200, 默认为 9600	—
数据位	设定 7 或 8, 默认为 8	—
奇偶校验位	设定为无校验、奇校验、偶校验, 默认为无校验	—
停止位	设定 1 或 2, 默认为 1	—
允许起始字符检测	允许或禁止, 默认为禁止	—
起始字符检测	0~255 (对应 00~FF)	检测到用户指定的起始字符, 开始接收, 并将接收到的字符 (包括起始字符) 保存到用户指定的缓存区中
允许结束字符检测	允许或禁止, 默认为禁止	—
结束字符检测	0~255 (对应 00~FF)	当接收到用户设定的结束字符, 结束接收, 并将结束字符保存到缓存区内
允许字符间超时时间	允许或禁止, 默认为禁止	—
字符间超时时间	0~65535ms	当接收到的两个字符间的时间超过用户设定的字符间超时时间, 接收中止
帧超时时间使能	有效或无效, 默认为无效	—
帧超时时间	0~65535ms	当 RCV 的能流导通, 并且通讯条件满足, 从通讯串口开始接收计时, 当计时时间到, 还没有接收完一帧, 中止接收

10.3.3 自由端口指令

注意事项

自由端口指令 **XMT** 和 **RCV** 可以用来向指定的通讯口发送和接收数据,自由端口指令的详细使用方法请参见 **6.12.11 XMT: 自由口发送指令**和 **6.12.12 RCV: 自由口接收指令**。

需要注意的是,如果在某个端口上使用自由端口指令,需先在 **YD-station** 软件的系统设置中,设置该通讯口使用自由端口协议并设置通讯参数,设置完毕后下载系统设置到 **PLC** 中,并且重新启动才能生效。

程序示例

例程 1: 通过通讯口 1 发送数据,然后接收数据,发送的数据为 5 个字节,接收的为 6 个字节。

发送的数据为:

01	FF	00	01	02
----	----	----	----	----

接收的数据为:

01	FF	02	03	05	FE
----	----	----	----	----	----

将接收到的数据保存到 **D10** 开始的地址,每一个字节保存到一个 **D** 元件中,保存的方式如下表所示:

01	FF	02	03	05	FE
D10	D11	D12	D13	D14	D15

<pre> graph TD SM1 --> MOV1[MOV 16#1 D0] MOV1 --> MOV2[MOV 16#FF D1] MOV2 --> MOV3[MOV 16#0 D2] MOV3 --> MOV4[MOV 16#1 D3] MOV4 --> MOV5[MOV 16#2 D4] MOV5 --> RST1[RST SM122] RST1 --> XMT[XMT 1 D0 5] XMT --> RST2[RST SM123] RST2 --> RCV[RCV 1 D10 6] RCV --> BLD[BLD SD125 2] BLD --> INC[INC D100] INC --> RST3[RST SM120] RST3 --> RST4[RST SM121] T5[T5] </pre>	<ol style="list-style-type: none"> 1. 首先应该在系统块中将通讯口的设置修改成自由口通讯,并设置波特率、奇偶校验位等参数。 2. SM1 一次能流有效时,将待发送的数据保存到 D0 开始的通讯缓存区中,利用 XMT 指令发送数据,发送前复位 SM122 (发送结束标志)。 3. 发送完成 SM122 置位,利用上升沿,开始接收数据,接收的最大长度为 6。 4. 接收完成 SM123 置位,根据接收完成信息寄存器 (SD125) 的内容,进行相应的操作。 5. 利用 X5 作为中断发送和接收的使能位
--	--

例程 2：通过通讯口 1 发送数据，然后接收数据。

<div><div><div>SM1</div><div>[</div><div>MOV</div><div>16#1</div><div>D0</div><div>]</div></div><div><div>[</div><div>MOV</div><div>16#FF01</div><div>K4M0</div><div>]</div></div><div><div>[</div><div>MOV</div><div>K2M0</div><div>D1</div><div>]</div></div><div><div>[</div><div>MOV</div><div>K2M8</div><div>D2</div><div>]</div></div><div><div>[</div><div>MOV</div><div>16#1</div><div>D3</div><div>]</div></div><div><div>[</div><div>MOV</div><div>16#2</div><div>D4</div><div>]</div></div><div><div>[</div><div>RST</div><div>SM122</div><div>]</div></div><div><div>[</div><div>XMT</div><div>1</div><div>D0</div><div>5</div><div>]</div></div><div><div>SM122</div><div>[</div><div>RST</div><div>SM123</div><div>]</div></div><div><div>[</div><div>RCV</div><div>1</div><div>D10</div><div>6</div><div>]</div></div><div><div>SM123</div><div>[</div><div>BLD</div><div>SD125</div><div>2</div><div>H</div><div>INC</div><div>D100</div><div>]</div></div><div><div>X5</div><div>[</div><div>RST</div><div>SM120</div><div>]</div></div><div><div>[</div><div>RST</div><div>SM121</div><div>]</div></div></div>	<p>和例程 1 不同的是，当要将一个字元件的高低字节都发送出去的时候，首先要将这个字元件拆成高、低字节两部分。</p> <p>例如要发送 D2 这个字元件的内容，可先将它的高、低字节分别存在 D3 和 D4 的低中，然后发送 D3、D4。也可以采用先将数据保存到一个 K4MX（例如本程序的 K4M0）的元件中，然后分别取高字节 K2M0 为高字节、K2M8 为低字节的方法</p>
--	--

10.4 Modbus 通讯协议

10.4.1 简介

APLC 系列小型 PLC 串口通讯可以选用 Modbus 通讯协议，支持 ASCII 和 RTU 两种通讯模式，可以设成主站或者从站。

10.4.2 链路特性

1. 物理层：RS232、RS485
2. 链路层：异步传输
 - (1) 数据位：7 位（ASCII）或者 8 位（RTU）
 - (2) 传输速率：1200, 2400, 4800, 9600, 19200, 38400
 - (3) 校验方式：偶校验、奇校验或无校验
 - (4) 停止位：停止位 1 位或者 2 位
3. 组网配置：最多 31 个设备，地址范围 1~31。支持广播。

10.4.3 RTU 传输模式

1. 十六进制数据。
2. 字符间间隔应该少于 1.5 个字符时间。
3. 没有帧头和帧尾，帧间间隔至少为 3.5 个字符时间。
4. 使用 CRC16 校验。
5. RTU 帧的最大帧长度是 256 个字节，帧结构如下表：

帧构成	地址	功能码	数据	CRC
字节数	1	1	0~252	2

6. 字符间隔时间计算：
 通讯波特率为 19200，那么 1.5 个字符时间= $1/19200 \times 11 \times 1.5 \times 1000 = 0.86\text{ms}$
 3.5 个字符间隔= $1/19200 \times 11 \times 3.5 \times 1000 = 2\text{ms}$ 。

10.4.4 ASCII 传输模式

1. 使用 ASCII 数据通讯。
2. 帧使用“: (3A)”作为头，CRLF (0D 0A) 两个字符作为尾
3. 允许的字符间的间隔时间是 1s。
4. 使用 LRC 校验。
5. ASCII 的帧结构比 RTU 的帧要长，ASCII 模式传送一个字节（HEX）需要两个字符编码，ASCII 的数据域（2×252）的最大长度是 RTU 数据域（252）的两倍，ASCII 的最大帧长为 513 个字符，帧构成如下表：

帧构成	头	地址	功能码	数据	LRC	尾
字节数	1	2	2	0~2*252	2	2

10.4.5 支持的 Modbus 功能码

仅支持 Modbus 通讯协议中的功能码 01, 02, 03, 05, 06, 08, 15, 16。

10.4.6 PLC 元件的寻址方式

1. 读写元件功能码和与元件对应关系：

功能码	功能码名称	Modicon 数据地址	可操作元件类型	注释
01	读线圈	0 ^{注1} :xxxx	Y、X、M、SM、S、T、C	读位
02	读离散量输入	1 ^{注2} :xxxx	X	读位
03	读寄存器	4 ^{注3} :xxxx ^{注4}	D、SD、Z、T、C	读字
05	写单个线圈	0:xxxx	Y、M、SM、S、T、C	写位
06	写单个寄存器	4:xxxx	D、SD、Z、T、C	写字
15	写多个线圈	0:xxxx	Y、M、SM、S、T、C	写位
16	写多个寄存器	4:xxxx	D、SD、Z、T、C	写字

注：

- 0 表示线圈
- 1 表示离散量输入
- 4 表示寄存器
- xxxx 表示范围 1~9999。每一种类型有独立的逻辑地址范围 1~9999（协议地址是从 0 开始的）。
- 0、1、4 并不具备物理上的意义，不参与实际的寻址。
- 用户不应该使用功能码 05, 15 对 X 元件写入。如果对 X 元件写入，并且写入的操作数和数据都是正确的，系统不会返回错误信息，但系统不会对写入的命令作任何操作。

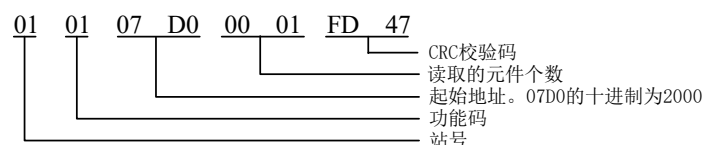
2. PLC 元件与 Modbus 通讯协议地址的对应关系：

元件	类型	物理元件	协议地址	支持的功能码	注释
Y	位元件	Y0~Y377 (8 进制编码) 共 256 点	0000~0255	01、05、15	输出的状态，元件编号为 Y0~Y7，Y10~Y17
X	位元件	X0~X377 (8 进制编码) 共 256 点	1200~01455 0000~0255	01、05、15 02	输入的状态，支持两种地址，元件编号同上
M	位元件	M0~M1999	2000~3999	01、05、15	
SM	位元件	SM0~SM255	4400~4655	01、05、15	
S	位元件	S0~S991	6000~6991	01、05、15	
T	位元件	T0~T255	8000~8255	01、05、15	T 元件的状态
C	位元件	C0~C255	9200~9455	01、05、15	C 元件的状态

元件	类型	物理元件	协议地址	支持的功能码	注释
D	字元件	D0~D7999	0000~7999	03、06、16	
SD	字元件	SD0~SD255	8000~8255	03、06、16	
Z	字元件	Z0~Z15	8500~8515	03、06、16	
T	字元件	T0~T255	9000~9255	03、06、16	T 元件的当前值
C	字元件	C0~C199	9500~9699	03、06、16	C 元件 (WORD) 的当前值
C	双字元件	C200~C255	9700~9811	03、16	C 元件 (DWORD) 的当前值

注：

协议地址是在数据传输中使用的地址，协议地址与 Modicon 的数据的逻辑地址有对应关系，协议地址是从 0 开始，Modicon 的数据的逻辑地址是从 1 开始的，也就是说协议地址 + 1 = Modicon 的数据的逻辑地址，例如：M0 协议地址是 2000，它对应的 Modicon 的数据的逻辑地址是 0：2001，在实际中对 M0 的读写是通过协议地址完成，例如对 M0 元件的读取帧（主站发出）：



10.4.7 Modbus 从站

Modbus 从站不主动发送任何报文，只有接收到对本地寻址的报文后才根据具体情况看是否响应主站。从站仅支持 Modbus 功能码 01，02，03，05，06，08，15，16，其余的响应均为“非法功能码”（除广播帧）。

10.4.8 元件的读写

除了功能码 08，APLC-TCP 支持的其他功能码都是对元件读写操作的，原则上最多允许一帧读 2000 个位元件，写 1968 个位元件，读取 125 个字元件，写 120 个字元件。但由于实际的协议地址对不同元件是分开的，不连续（例如 Y377 的协议地址是 255，X0 的协议地址是 1200），因此在对元件的读写操作时，一次读取的元件只能是一种类型的元件，而读取元件的最多数目也与实际定义的该类型元件个数有关系，例如读取 Y 元件，Y0~Y377（共 256 点），协议地址范围 0~255，对应的 Modicon 的数据的逻辑地址是 1~256，在读取 Y 元件时最多允许读取 256 个元件。

举例如下：

1. 主站发送：01 01 00 00 01 00 3D 9A

01 地址，功能码 01，00 00 起始地址，01 00 读取元件个数 3D 9A 校验

从站应答：会返回正确的应答

2. 主站发送：01 01 00 00 01 01 FC 5A

主站读取起始地址从 0000 的 01 01（257）个元件，超出了定义的 Y 元件个数范围

从站应答：01 81 03 00 51

从站应答是非法的数据值，原因是 257 大于 256，而 256 是最大允许的 Y 元件个数

3. 主站发送：01 01 00 64 00 A0 7D AD

主站读取起始地址 00 64 (十进制 100)，元件个数 00 A0 (十进制 160)

从站应答: 01 81 02 C1 91

从站应答非法数据地址，从协议地址 100 开始的 Y 元件只有 156 个，读 160 个非法。

4. 主站发送: 01 04 00 02 00 0A D1 CD

主站发送功能码 04 的帧

从站应答: 01 84 01 82 C0

从站应答非法的功能码，APLC-TCP 不支持功能码 04

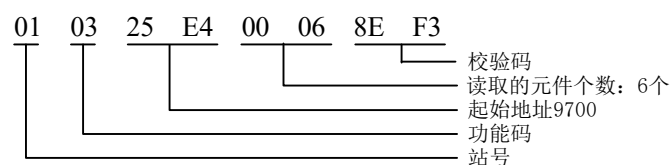
注意

1. X 元件不支持写入 (也就是对 X 元件写入是个无效的操作)。SM, SD 元件的可写属性请查阅附录一 特殊辅助继电器及附录二 特殊数据寄存器 (当元件不可写入时，对其写入也是个无效的操作)。
2. 从站的地址都是 01，后两个字节都是 CRC 校验码，第二字节是功能码。

10.4.9 对双字元件的处理

C 元件的当前计数值为字元件或双字元件，C200~C255 为双字元件。对 C200~C255 的读写也是通过读写寄存器的功能码 (03、16) 来完成。每两个寄存器的地址对应一个 C 双字元件，读写时只能成对的读写寄存器。

例如: 读取 C200~C202 三个 C 双字元件 RTU 帧:



返回的数据中 9700 9701 两个地址表示 C200 的内容，9700 为高 16 位，9701 为低 16 位。

在读取双字元件时如果读取的开始地址不是偶数，会返回异常码非法地址，如果读取的寄存器个数不是偶数，会返回异常码非法数据。

举例如下:

主站发送: 01 03 25 E5 00 04 5E F2

主站发送读取开始地址是 25 E5 (十进制 9701) 的四个字元件

从站响应: 01 83 02 C0 F1

从站应答: 非法的数据地址

主站发送: 01 03 25 E4 00 05 CE F2

主站读取开始地址是 25 E4 的 5 个字元件

从站应答: 01 83 03 01 31

从站返回非法的数据

10.4.10 对 LONG INT 的处理

对于一个 LONG INT 类型数据的存储，可能存在两个 D 元件内，例如：D3, D4 存一个 LONG INT 型的数，AT100 PLC 认为 D3 存储的是高 16 位，D4 存储的是低 16 位，当主站通过 Modbus 读取 LONG INT 数据时，读回数据后也应该按照 EMERSON PLC 对 LONG INT 的存储原则重组 32 位的数据。FLOAT 的存储原则等同于 LONG INT 的存储原则。

10.4.11 诊断功能码

诊断功能码用来提供测试主站和从站之间的通讯，或者从站的各种内部差错状态。支持的诊断子功能码如下表所示：

功能码	子功能码	子功能码名称	功能码	子功能码	子功能码名称
08	00	返回询问数据	08	12	返回总线通讯错误计数
08	01	重新启动通讯选项	08	13	返回总线异常错误计数
08	04	强制只听模式	08	14	返回从站报文计数
08	10	清除计数器	08	15	返回从站无响应计数
08	11	返回总线报文计数	08	18	返回总线字符超限计数

10.4.12 异常码

当主站发送的命令，在正常的响应中，从站在数据域中返回数据或统计值。在异常的响应中，服务器在数据域中返回异常码，异常码如下表：

异常代码	异常代码意义
0x01	非法功能码
0x02	非法寄存器地址
0x03	非法的数据

另外，从站在以下几种情况接收到数据会不返回响应报文：

- (1) 广播帧中有错误，例如数据错误，地址错误等。
- (2) 字符超限不返回，例如 RTU 帧大于 256 个字节。
- (3) 当 RTU 传输模式，字符间的间隔时间超时，相当于收到错误帧，不返回。
- (4) 从站在只听模式不返回。
- (5) 从站接收到错误的 ASCII 错误帧，包括帧尾错误，帧中的字符范围错误。

注意

读取站具有强制元件，读取的只是程序运行的值，可能与强制值不符

10.4.13 Modbus 参数设置

设置系统块中的通讯口

在通讯口界面中有两个串口的选择，PORT 0 和 PORT 1，PORT 0 只支持 Modbus 从站，PORT 1 支持 Modbus 主站和从站。

设置 Modbus 通讯协议参数

在 Modbus 通讯协议操作数界面中，有个默认值按钮，默认值是 Modbus 通讯协议推荐的通讯设置。参数设置选项如下表所示。

选项	设置内容
站号	0~31
波特率	38400, 19200, 9600, 4800, 2400, 1200
数据位	设定 7 或 8, ASCII 模式 7 位, RTU 模式 8 位
奇偶校验位	设定为无校验、奇校验、偶校验
停止位	设定 1 或 2, 奇、偶校验时设定为 1, 无校验时设定为 2
Modbus 主/从	可设为主站或从站, 通讯口 1 可设为主站或从站, 通讯口 0 不可设为主站
传输模式	选择 RTU 模式或 ASCII 模式
主模式超时时间	主站等待从站响应的超时时间
注：当操作数在系统块中设定并下载后，不是立即有效，必须运行一次，才能生效。	

10.4.14 Modbus 指令

PLC 作为 Modbus 主站使用时，可以通过系统提供的 Modbus 指令发送 Modbus 数据帧并接收回复，Modbus 指令的详细信息使用方法请参见 6.12.1 *Modbus：主站通讯指令*。

当设置 PLC 为主站时，在系统块中设置 Modbus 参数时，有个主模式的超时时间，为了能够保证接收数据的正确性，设置时应确保这个时间应该比 Modbus 从站的一个循环的扫描周期要长，并且有余量。

10.5 ModbusTCP 通讯协议

10.5.1 简介

APLC 系列小型 PLC 网口通讯可以选用 ModbusTCP 通讯协议(从站)。

元件寻址方式参考 10.4.6 PLC 元件寻址方式，功能码参考 10.4.7 Modbus 从站。

10.5.2 网络设置

在系统块中选择网口选项，



保存范围	输出表	设置时间	输入过滤器	输入点	高级设置
通讯口	网口	特殊模块配置	中断优先级	通信模块	

MAC地址: 085700D6ED2C [更新]

IP与端口号: 192 . 168 . 1 . 254 5000

子网掩码: 255 . 255 . 255 . 0

网关: 192 . 168 . 1 . 1

单元地址: 8

[确定] [取消] [帮助]

- (1) MAC 地址：用户可写入 MAC 地址，点击更新会自己生成 MAC 地址；
- (2) IP 与端口号：默认端口号为 5000；
- (3) 子掩码：默认 255.255.255.0；
- (4) 网关：网关的网段需与 IP 的网段相同；
- (5) 单元地址：ModbusTCP 网络的单元地址（使用默认值）。

附录一 特殊辅助继电器

所有特殊辅助继电器，在 **STOP→RUN** 时被系统初始化，在系统设置中被设置的特殊辅助继电器将在前面的初始化完成后，重新根据系统块设置中的设置值赋值。

注意

保留的 **SD**，**SM** 表格中不列出，保留的 **SM** 元件读写属性默认为只读（**R**）。

1. PLC 工作状态标志

地址	名称	动作与功能	R/W	APLC-TCP
SM0	监控运行位	RUN 状态下，始终为高， STOP 状态下始终为零	R	√
SM1	初始运行脉冲位	用户程序从 STOP 到 RUN ，置高一个运行周期后置低	R	√
SM2	上电标志位	当发生系统上电后置为高，当用户程序运行一个周期后置低	R	√
SM3	系统错误	上电后或 STOP 到 RUN 时检测有系统错误发生时置位，如没有任何系统错误发生，该位清零	R	√
SM4	电池电压过低	当电池电压过低时置位，当检测到电池电压高于 2.4V 清除该位	R	
SM5	交流停电检测位	检测到交流失电时置位（检测交流失电时间窗口长 40ms），延时停电检测时间（在 SD05 中设定）后，如掉电情况消失清除该位	R	
SM6	24Vdc 失电	检测 24Vdc 失电时置位（检测时间窗口长 50ms），延迟 50ms 后，再次检测 24Vdc 失电时，如失电情况消失，清除该位	R	√
SM7	无电池工作模式	当该位置 1 时，系统电池失效情况下，不报电池备份数据丢失错误和强制表丢失错误。（只能通过系统块配置）	R	
SM8	恒定扫描模式	该位置位后，扫描时间恒定（只能通过系统块配置）	R	√
SM9	输入点开机模式	该位置位后，设定的 X 输入点 ON 时 PLC 可由 STOP 进入 RUN 状态（只能通过系统块配置）	R	√

2. 时钟运行位

地址	名称	动作与功能	R/W	APLC-TCP
SM10	10ms 时钟	10ms 为周期的时钟振荡（半个周期翻转，用户程序运行时首个半个周期为 0）	R	√
SM11	100ms 时钟	100ms 为周期的时钟振荡（半个周期翻转，用户程序运行时首个半个周期为 0）	R	√
SM12	1s 时钟	1s 为周期的时钟振荡（半个周期翻转，用户程序运行时首个半个周期为 0）	R	√
SM13	1min 时钟	1min 为周期的时钟振荡（半个周期翻转，用户程序运行时首个半个周期为 0）	R	√
SM14	1hour 时钟	1hour 为周期的时钟振荡（半个周期翻转，用户程序运行时首个半个周期为 0）	R	√
SM15	扫描周期振荡位	该位每一扫描周期翻转一次（用户程序运行时首次周期为 0）	R	√

3. 用户程序执行错误

地址	名称	动作与功能	R/W	APLC-TCP
SM20	指令执行错误	指令执行错误，置位。同时具体的错误类型代码填入 SD20。执行应用指令正确后清零	R	√
SM21	指令元件编号下标溢出	指令执行错误，置位。同时具体的错误类型代码填入 SD20	R	√
SM22	指令参数非法	指令执行错误，置位。同时具体的错误类型代码填入 SD20。执行应用指令正确后清零	R	√

4. 中断控制

地址	名称	动作与功能	R/W	APLC-TCP
SM40	X0 输入上升/下降沿中断使能标志位	置 1 时，使能 X0 上升沿（下降沿）中断	R/W	√
SM41	X1 输入上升/下降沿中断标志位	置 1 时，使能 X1 上升沿（下降沿）中断	R/W	√
SM42	X2 输入上升/下降沿中断使能标志位	置 1 时，使能 X2 上升沿（下降沿）中断	R/W	√
SM43	X3 输入上升/下降沿中断使能标志位	置 1 时，使能 X3 上升沿（下降沿）中断	R/W	√
SM44	X4 输入上升/下降沿中断使能标志位	置 1 时，使能 X4 上升沿（下降沿）中断	R/W	√
SM45	X5 输入上升/下降沿中断使能标志位	置 1 时，使能 X5 上升沿（下降沿）中断	R/W	√
SM46	X6 输入上升/下降沿中断使能标志位	置 1 时，使能 X6 上升沿（下降沿）中断	R/W	√
SM47	X7 输入上升/下降沿中断使能标志位	置 1 时，使能 X7 上升沿（下降沿）中断	R/W	√
SM48	通讯口 0 的字符发送中断使能标志位	置 1 时，允许	R/W	√
SM49	通讯口 0 的字符接收中断使能标志位	置 1 时，允许	R/W	√
SM50	通讯口 0 的帧发送中断使能标志位	置 1 时，允许	R/W	√
SM51	通讯口 0 的帧接收中断使能标志位	置 1 时，允许	R/W	√
SM52	通讯口 1 的字符发送中断使能标志位	置 1 时，允许	R/W	√
SM53	通讯口 1 的字符接收中断使能标志位	置 1 时，允许	R/W	√
SM54	通讯口 1 的帧发送中断使能标志位	置 1 时，允许	R/W	√
SM55	通讯口 1 的帧接收中断使能标志位	置 1 时，允许	R/W	√
SM56	电源失电中断	置 1 时，允许	R/W	√
SM63	PTO (Y0) 输出完成中断使能标志位	置 1 时，使能 PTO (Y0) 输出完成中断	R/W	√
SM64	PTO (Y1) 输出完成中断使能标志位	置 1 时，使能 PTO (Y1) 输出完成中断	R/W	√
SM65	高速计数器中断使能标志位	置 1 时，使能高速计数器中断	R/W	√
SM66	定时中断 0 使能标志位	置 1 时，使能定时中断 0	R/W	√
SM67	定时中断 1 使能标志位	置 1 时，使能定时中断 1	R/W	√
SM68	定时中断 2 使能标志位	置 1 时，使能定时中断 2	R/W	√

5. 外设指令

地址	名称	动作与功能	R/W	APLC-TCP
----	----	-------	-----	----------

地址	名称	动作与功能	R/W	APLC-TCP
SM70	打印模式选择	置 1 时, 1-16 个字符, 0 时固定 8 个字符	R/W	
SM71	打印进行中	置 1 时, 正在打印	R	

6. 高速脉冲输出控制

地址	名称	功能	R/W	
SM80	Y0 高速脉冲输出控制	Y0 高速脉冲输出停止指令	R/W	√
SM81	Y1 高速脉冲输出控制	Y1 高速脉冲输出停止指令	R/W	√
SM82	Y0 高速脉冲输出监视	Y0 高速脉冲输出监视 (busy 时为 ON, ready 时为 OFF)	R	√
SM83	Y1 高速脉冲输出监视	Y1 高速脉冲输出监视 (busy 时为 ON, ready 时为 OFF)	R	√
SM84	PWM 时基单位	为 ON 时, 时基单位为微秒; 为 OFF 时, 时基单位为毫秒	R/W	
SM85	清零功能有效	定位指令原点回归 (ZRN) CLR 信号输出功能有效	R/W	√
SM86	Y000 中断驱动脉冲输出有效	为 ON 时, 中断程序和子程序中可调用 PLSY 指令, 主程序中调用将随能流连续反复驱动	R/W	√
SM87	Y001 中断驱动脉冲输出有效	为 ON 时, 中断程序和子程序中可调用 PLSY 指令, 主程序中调用将随能流连续反复驱动	R/W	√

7. 脉冲捕捉监视位

地址	名称	功能	R/W	APLC-TCP
SM90	输入 X0 脉冲捕捉监视位	输入 X0 上升沿脉冲捕捉	R/W	√
SM91	输入 X1 脉冲捕捉监视位	输入 X1 上升沿脉冲捕捉	R/W	√
SM92	输入 X2 脉冲捕捉监视位	输入 X2 上升沿脉冲捕捉	R/W	√
SM93	输入 X3 脉冲捕捉监视位	输入 X3 上升沿脉冲捕捉	R/W	√
SM94	输入 X4 脉冲捕捉监视位	输入 X4 上升沿脉冲捕捉	R/W	√
SM95	输入 X5 脉冲捕捉监视位	输入 X5 上升沿脉冲捕捉	R/W	√
SM96	输入 X6 脉冲捕捉监视位	输入 X6 上升沿脉冲捕捉	R/W	√
SM97	输入 X7 脉冲捕捉监视位	输入 X7 上升沿脉冲捕捉	R/W	√

注:

- 由 STOP 到 RUN 时清除。在本端口上有 HCNT 高速计数器驱动指令和 SPD 测频指令, 端口的脉冲捕捉无效。在其它情况都有效。具体说明参考 6.10.8 SPD: 测频指令和 6.10.1 HCNT: 高速计数器驱动指令。
- 对 X0~X7 脉冲的总输入数在硬件方式 (脉冲捕捉, SPD 指令, HCNT 且没有高速比较指令时) 是不超过 80k, 软件处理 (在驱动了的高速计数器中使用 DHSCS, DHSCI, DHSZ, DHSP, DHST 指令) 是不超过 30k

8. 自由端口 (PORT 0)

地址	名称	动作与功能	R/W	APLC-TCP
----	----	-------	-----	----------

SM110	端口 0 发送使能标志	当使用 XMT 指令时该位被置位，当发送结束后该位将被自动清零。用户也可对其手动清零，以中止端口 0 的当前发送任务。当又有能流导通的时候，该端口的发送任务可以继续	R/W	√
SM111	端口 0 接收使能标志	当使用 RCV 指令时该位被置位，当发送结束后该位将被自动清零。用户也可对其手动清零，以中止端口 0 的当前发送任务。当又有能流导通的时候，该端口的发送任务可以继续	R/W	√
SM112	端口 0 发送完成标志	发送完成置位	R/W	√
SM113	端口 0 接收完成标志	接收完成置位	R/W	√
SM114	端口 0 空闲标志	当端口没有通讯任务的时候，标志位置位	R	√

注意

SM112-SM114 是针对于适用 PORT 0 的所有通信协议而言的一个接收、完成与空闲标志。比如：APLC-TCP PLC 的 PORT 0 可用于 Modbus 及 FREEMPORT 协议，无论采用那种协议，SM112-SM114 都适用。

9. 自由端口 (PORT 1)

地址	名称	动作与功能	R/W	APLC-TCP
SM120	端口 1 发送使能标志	当使用 XMT 指令时该位被置位，当发送结束后该位将被自动清零。用户也可对其手动清零，以中止端口 1 的当前发送任务。当又有能流导通的时候，该端口的发送任务可以继续	R/W	√
SM121	端口 1 接收使能标志	当使用 RCV 指令时该位被置位，当发送结束后该位将被自动清零。用户也可对其手动清零，以中止端口 1 的当前发送任务。当又有能流导通的时候，该端口的发送任务可以继续	R/W	√
SM122	端口 1 发送完成标志	发送完成置位	R/W	√
SM123	端口 1 接收完成标志	接收完成置位	R/W	√
SM124	端口 1 空闲标志	当端口没有通讯任务的时候，标志位置位	R	√

注意

SM122-SM124 是针对于适用 PORT 1 的所有通信协议而言的一个接收、完成与空闲标志。比如：PORT 1 可用于 Modbus 及 FREEMPORT 协议，无论采用那种协议，SM122-SM124 都适用。

10. 扩展自由端口 (PORT 2)

地址	名称	动作与功能	R/W	APLC-TCP
SM130	端口 2 发送使能标志	当使用 XMT 指令时该位被置位，当发送结束后该位将被自动清零。用户也可对其手动清零，以中止端口 1 的当前发送任务。	R/W	

地址	名称	动作与功能	R/W	APLC-TCP
		当又有能流导通的时候，该端口的发送任务可以继续		
SM131	端口 2 接收使能标志	当使用 RCV 指令时该位被置位，当发送结束后该位将被自动清零。用户也可对其手动清零，以中止端口 1 的当前发送任务。当又有能流导通的时候，该端口的发送任务可以继续	R/W	
SM132	端口 2 发送完成标志	发送完成置位	R/W	
SM133	端口 2 接收完成标志	接收完成置位	R/W	
SM134	端口 2 空闲标志	当端口没有通讯任务的时候，标志位置位	R	

注意

SM132-SM134 是针对适用于 PORT 2 的所有通信协议而言的一个接收、完成与空闲标志。比如：PLC 的 485 通信模块可用于 Modbus 及 FREEMPORT 协议，无论采用那种协议，SM132-SM134 都适用。

11. Modbus 通讯

地址	名称	动作与功能	R/W	APLC-TCP
SM135	端口 1 Modbus 的通讯完成	通讯完成时置位	R/W	√
SM136	端口 1 Modbus 的通讯错误	通讯错误时置位	R/W	√
SM137	端口 2 Modbus 的通讯完成	通讯完成时置位	R/W	
SM138	端口 2 Modbus 的通讯错误	通讯错误时置位	R/W	

12. 集成模拟量通道的使能标志

地址	名称	动作与功能	R/W	APLC-TCP
SM172	AD 通道 0 的使能标志	置 1，则使能 AD 通道 0 的采样	R/W	
SM173	AD 通道 1 的使能标志	置 1，则使能 AD 通道 1 的采样	R/W	
SM174	AD 通道 0 的电压电流使能标志	置 1，电流输入，0 为电压输入	R/W	
SM175	AD 通道 1 的电压电流使能标志	置 1，电流输入，0 为电压输入	R/W	
SM178	DA 通道 0 的使能标志	置 1，则使能 DA 通道 0 的输出	R/W	

13. 运算标志位

地址	名称	动作与功能	R/W	APLC-TCP
SM180	零标志位	当相关操作结果为零时，相关指令执行时对该位置位。用户可对其手动清零	R/W	√
SM181	进位/溢出标志位	当相关操作有进位时，相关指令执行时对该位置位，用户可对其手动清零	R/W	√
SM182	借位	当相关操作有借位时，相关指令执行时对该位置位，用户可手动清除、设置该位	R/W	√
SM185	表格比较标志	当整个表格记录完成时置位	R/W	√

14. ASCII 码转换指令标志

地址	名称	动作与功能	R/W	APLC-TCP
SM186	ASC 指令存储方式标志	0: 每个字高低字节各存放 1 个 ASCII 码 1: 每个字的低字节存放 1 个 ASCII 码	R/W	√

15. 系统总线错误标志

地址	名称	动作与功能	R/W	APLC-TCP
SM190	主模块总线错误标志位	1. 上电编址正确清除 2. STOP→RUN 无此错误清除 3. 下载新程序时清除 4. 该位引起系统停机	R	√
SM191	一般模块总线错误标志位	1. 当发生一般模块总线操作错误, 该位置位, 系统报警 2. 系统故障消除标志自动清除	R	√
SM192	特殊模块总线错误标志位	1. 当发生特殊模块总线操作错误, 该位置位, 系统报警 2. 系统故障消除标志自动清除	R	√

16. 实时时钟错误标志

地址	名称	动作与功能	R/W	APLC-TCP
SM193	读写实时时钟错误	当发生实时时钟错误, 该位置位; 系统故障消除标志自动清除	R	

17. EEPROM 标志

地址	名称	动作与功能	R/W	APLC-TCP
SM196	EEPROM 写空闲标志	当 EEPROM 没有写处理时, 置位	R	√

18. 增/减型计数器计数方向

地址号	对应计数器地址号	功能	R/W	APLC-TCP
SM200	C200	当 SM2 __ 为高电平，其对应的 C2 __ 变为减型计数 当 SM2 __ 为低电平，其对应的 C2 __ 变为增型计数	R/W	√
SM201	C201		R/W	√
SM202	C202		R/W	√
SM203	C203		R/W	√
SM204	C204		R/W	√
SM205	C205		R/W	√
SM206	C206		R/W	√
SM207	C207		R/W	√
SM208	C208		R/W	√
SM209	C209		R/W	√
SM210	C210		R/W	√
SM211	C211		R/W	√
SM212	C212		R/W	√
SM213	C213		R/W	√
SM214	C214		R/W	√
SM215	C215		R/W	√
SM216	C216		R/W	√
SM217	C217		R/W	√
SM218	C218		R/W	√
SM219	C219		R/W	√
SM220	C220		R/W	√
SM221	C221		R/W	√
SM222	C222		R/W	√
SM223	C223		R/W	√
SM224	C224		R/W	√
SM225	C225		R/W	√
SM226	C226	当 SM2 __ 为高电平，其对应的 C2 __ 变为减型计数 当 SM2 __ 为低电平，其对应的 C2 __ 变为增型计数	R/W	√
SM227	C227		R/W	√
SM228	C228		R/W	√
SM229	C229		R/W	√
SM230	C230		R/W	√
SM231	C231		R/W	√
SM232	C232		R/W	√
SM233	C233		R/W	√
SM234	C234		R/W	√
SM235	C235		R/W	√

19. 高速计数器的计数方向及监控

区分	地址号	名称	寄存器内容	R/W	APLC-TCP
单相单端计数输入	SM236	C236	其对应的 SM2 __ 变为高电平和低电平分别对应计数器的减和增	R/W	√
	SM237	C237		R/W	√
	SM238	C238		R/W	√
	SM239	C239		R/W	√
	SM240	C240		R/W	√
	SM241	C241		R/W	√
	SM242	C242		R/W	√
	SM243	C243		R/W	√
	SM244	C244		R/W	√
单相增减计数输入	SM245	C245	当单相增减计数输入计数器和双相计数输入计数器的 C2 __ 处于减型计数模式时，其对应的 SM2 __ 变为高电平。增型计数时，为低电平	R/W	√
	SM246	C246		R/W	√
	SM247	C247		R/W	√
	SM248	C248		R/W	√
	SM249	C249		R/W	√
双相计数输入	SM250	C250		R/W	√
	SM251	C251		R/W	√
	SM252	C252		R/W	√
	SM253	C253		R/W	√
	SM254	C254		R/W	√
	SM255	C255		R/W	√

20. 增强定位

地址号	名称	功能	R/W	
SM260	中断输入功能指定有效	Y0、Y1 适用 DVIT。当不使用中断输入指定功能，那么 Y0 对应 X0 的中断，Y1 对应 X1 的中断；当使用指定则将该位置位，然后对应的 SD240 的每一个 4 位对应着每一个输出 (Y) 的输入	R/W	
SM280	清零功能有效	定位指令原点回归 (ZRN) CLR 信号输出功能有效(Y0) DSZR/ZRN	R/W	
SM281	清零信号指定软件有效	对应的 SD230 内的数值为 Y (N) 表示清零信号，不指定则 Y0 为 Y2。适用 DSZR	R/W	
SM282	原点回归方向	Y0 适用 DSZR	R/W	
SM283	正转极限	Y0 适用 DSZR/DVIT	R/W	
SM284	反转极限	Y0 适用 DSZR/DVIT	R/W	
SM285	近点信号逻辑反转	Y0 适用 DSZR	R/W	
SM286	零点信号逻辑反转	Y0 适用 DSZR	R/W	
SM287	中断信号逻辑反转	Y0 适用 DVIT，不适用法律中断输入指令	R/W	
SM288	定位指令驱动中	Y0 适用 DSZR/DVIT	R/W	
SM289	用户中断输入指令	Y0 适用 DVIT	R/W	
SM290	清零功能有效	Y1 适用 DSZR/ZRN	R/W	
SM291	清零信号指定软件有效	对应的 SD 内的数值为 Y (N) 表示清零信号，不指定则 Y1 为 Y3 适用 DSZR	R/W	
SM292	原点回归方向	Y1 适用 DSZR	R/W	
SM293	正转极限	Y1 适用 DSZR/DVIT	R/W	
SM294	反转极限	Y1 适用 DSZR/DVIT	R/W	
SM295	近点信号逻辑反转	Y1 适用 DSZR	R/W	
SM296	零点信号逻辑反转	Y1 适用 DSZR	R/W	
SM297	中断信号逻辑反转	Y1 适用 DVIT	R/W	
SM298	定位指令驱动中	Y1 适用 DSZR/DVIT	R/W	
SM299	用户中断输入指令	Y1 适用 DVIT	R/W	

附录二 特殊数据寄存器

注意

- SD50~SD55 以外的所有特殊数据寄存器，在 STOP→RUN 时被系统初始化。
- 保留的 SD，SM 表格中不列出，保留的 SD 元件读写属性默认为只读（R）。

1. PLC 工作状态数据

地址	名称	动作与功能	R/W	APLC-TCP	范围
SD00	PLC 类型	10 代表 AT100,	R	√	
SD01	版本号	例如：100 为 1.00	R	√	
SD02	用户程序的容量	例如：8 表示 8k 步程序	R	√	
SD03	系统错误代码	存储发生的系统错误代码	R	√	
SD04	电池电压值	以 0.1V 为单位，3.6V 为 36	R		
SD05	交流失电检测延迟时间设置值	如设置值小于 10ms 按 10ms 处理。 如设置值大于 100ms 按 100ms 处理。 (只能通过系统块配置)	R		10~100ms
SD07	扩展 I/O 模块个数		R	√	
SD08	特殊模块个数		R	√	
SD09	设定运行控制的输入点 采用 10 进制 (X0 显示为 0, X10 显示为 8, 最大到 15) (只能通过系统块配置)		R	√	0~15
SD10	主模块 IO 的点数	高字节：输入。低字节：输出	R	√	
SD11	扩展模块 IO 的点数	高字节：输入。低字节：输出	R	√	
SD12	主模块模拟 IO 的点数	高字节：输入。低字节：输出	R	√	

2. 运行错误代码 FIFO 区

地址	名称	动作与功能	R/W	APLC-TCP	范围
SD20	保留运行错误代码 0	按队列顺序, 保留 5 条最近的运行错误类型代码, SD20 总保存新近发生的错误的类型代码	R	√	
SD21	保留运行错误代码 1		R	√	
SD22	保留运行错误代码 2		R	√	
SD23	保留运行错误代码 3		R	√	
SD24	保留运行错误代码 4		R	√	

3. FROM/TO 错误

地址	名称	R/W	APLC-TCP	范围
SD25	使用 FROM/TO 指令时, 发生了错误的特殊模块编号 (从 0 开始)	R	√	初始值为 255
SD26	刷新 I/O 操作时, 发生了错误的 I/O 芯片编号 (从 0 开始)	R	√	初始值为 255

4. 扫描时间

地址	名称	动作与功能	R/W	APLC-TCP	范围
SD30	当前扫描值	当前扫描时间 (ms 为单位)	R	√	
SD31	最小扫描时间	扫描时间的最小值 (ms 为单位)	R	√	
SD32	最大扫描时间	扫描时间的最大值 (ms 为单位)	R	√	
SD33	恒定扫描时间设定值	初始值为 0ms (只能通过系统块配置), 以 1ms 为单位, 当恒定扫描时间大于用户监控超时设定值时, 作用户程序超时报警。当用户程序某个扫描周期大于恒定扫描时, 该周期恒定扫描模式自动失效, 不作报警处理。 当 SD33 设定值大于 1000ms 时, 按 1000 处理	R	√	0~1000ms
SD34	用户程序超时设定值	初始值为 100ms (只能通过系统块配置)。 当 SD34 值小于 100 时, 按 100 处理; 当 SD34 值大于 1000 时, 按 1000 处理	R	√	100~1000ms

注意

- SD30、SD31、SD32 有 1ms 的误差。
- 当恒定扫描时间设定值 SD33 与用户程序超时设定值 SD34 值相近时, 受系统工况及用户程序影响, 易发生用户程序超时错误, 建议用户程序超时设定值大于恒定扫描时间设定值 (SD33) 5ms。

5. 输入滤波常数设置

地址	名称	动作与功能	R/W	APLC-TCP	范围
SD35	输入滤波常数	(只能通过系统块配置)	R	√	0~60

6. 高速脉冲输出监控

地址	名称	R/W	APLC-TCP	范围
SD50	PLSR/PLSY 指令输出 Y0 脉冲总数 (高位)	R/W		
SD51	PLSR/PLSY 指令输出 Y0 脉冲总数 (低位)	R/W		
SD52	PLSR/PLSY 指令输出 Y1 脉冲总数 (高位)	R/W		
SD53	PLSR/PLSY 指令输出 Y1 脉冲总数 (低位)	R/W		
SD54	PLSR/PLSY 指令输出 Y1, Y0 脉冲总数 (高位)	R/W		
SD55	PLSR/PLSY 指令输出 Y1, Y0 脉冲总数 (低位)	R/W		
SD56	PLS 指令输出 Y0 目前执行到的段数	R		
SD57	PLS 指令输出 Y1 目前执行到的段数	R		

7. 定时中断周期

地址	名称	寄存器内容	R/W	APLC-TCP	范围
SD66	定时中断 0 周期设置值	当值不在 1~32767 范围内时该中断不触发	R/W	√	1~32767ms
SD67	定时中断 1 周期设置值	当值不在 1~32767 范围内时该中断不触发	R/W	√	1~32767ms
SD68	定时中断 2 周期设置值	当值不在 1~32767 范围内时该中断不触发	R/W	√	1~32767ms

注: 系统在处理用户定时中断时有 ± 1 ms 误差, 为了保证定时中断能正常工作, 建议用户定时中断周期设置值大于等于 5ms。

8. 定位指令

地址	名称	R/W	APLC-TCP	初始值
SD80	Y0 输出定位指令的当前值（高位）	R/W	√	0
SD81	Y0 输出定位指令的当前值（低位）	R/W	√	
SD82	Y1 输出定位指令的当前值（高位）	R/W	√	0
SD83	Y1 输出定位指令的当前值（低位）	R/W	√	
SD84	执行 ZRN、DRVI、DRVA 指令时的基底速度	R/W	√	0
SD85	执行 ZRN、DRVI、DRVA 指令时的最高速度（高位）	R/W	√	100.000
SD86	执行 ZRN、DRVI、DRVA 指令时的最高速度（低位）	R/W	√	
SD87	执行 ZRN、DRVI、DRVA 指令时的加减速时间	R/W	√	100
SD88	包络线上升时间（毫秒）	R/W	√	100
SD89	包络线下降时间（毫秒）	R/W	√	100

9. 实时时钟

地址	名称	寄存器内容	R/W	APLC-TCP	范围
SD100	年	实时时钟用	R		2000~2099
SD101	月	实时时钟用	R		1~12 月
SD102	日	实时时钟用	R		1~31 日
SD103	小时	实时时钟用	R		0~23 小时
SD104	分	实时时钟用	R		0~59 分钟
SD105	秒	实时时钟用	R		0~59 秒
SD106	星期	实时时钟用	R		0（周日）~6（周六）
用户只能通过 TWR 指令或上位机设置					

10. 自由端口接收控制及状态（通讯口 1）

地址	名称	寄存器内容	R/W	APLC-TCP	范围
SD120	自由端口 1 模式状态字	SD120.0~SD120.2 端口波特率 b2, b1, b0 000: 38,400 波特率 001: 19,200 波特率 010: 9,600 波特率 011: 4,800 波特率 100: 2,400 波特率 101: 1,200 波特率 110: 57,600 波特率 111: 115,200 波特率	R/W	√	
		SD120.3 停止位 0: 1 位停止位 1: 2 位停止位			
		SD120.4 奇偶校验 0: 偶校验 1: 奇校验			
		SD120.5 奇偶校验允许 0: 不校验 1: 校验			
		SD120.6 每个字符的数据位 每个字符的数据位 0: 8 位字符 1: 7 位字符			
		SD120.7 自由口接收开始字符模式 1: 有特定起始字符 0: 无特定起始字符			
		SD120.8 自由端口接收结束字符模式 1: 有特定结束字符 0: 无特定结束字符			
		SD120.9 自由口字符间超时有效 1: 有字符间超时有效 0: 无字符间超时有效			
		SD120.10 自由口帧间超时有效 1: 有帧间超时 0: 无帧间超时			
		SD120.11 保留			
		SD120.12 高低字节有效 0: 字元件低字节有效 1: 字元件高低字节有效			
		SD120.13~SD120.15 保留			
SD121	开始字符		R/W	√	
SD122	结束字符		R/W	√	
SD123	字符间超时时间	默认 0ms (忽略字符间超时)	R/W	√	0~32767ms
SD124	帧超时时间	默认 0ms (忽略帧超时)	R/W	√	0~32767ms
SD125	接收完成信息代码	第 0 位: 用户终止接收置位 第 1 位: 收到指定结束字置位 第 2 位: 收到最大字符数置位 第 3 位: 字符间超时置位 第 4 位: (帧) 接收超时置位 第 5 位: 奇偶检验错误时置位 第 6~15 位: 保留, 用户可忽略	R	√	
SD126	当前收到的字符		R	√	
SD127	当前收到的字符总数		R	√	
SD128	当前发送的字符		R	√	

11. Modbus (PORT1) 设定

地址	名称	R/W	APLC-TCP	范围
SD135	本站站号设定 (通讯口 1)	R/W	√	MOD (1~32) , EMR (0~31)
SD136	最大超时时间设定 (通讯口 1 发送之后和接收之前)	R/W	√	
SD137	重试次数 (通讯口 1)	R/W	√	0~100
SD138	预留	R/W	√	1~13
SD139	Modbus 主站的错误代码 (通讯口 1)	R	√	

12. 扩展自由端口接收控制及状态（通讯口 2）

地址	名称	寄存器内容	R/W	APLC-TCP	范围
SD150	自由端口 2 模式状态字	SD150.0~SD150.2 端口波特率 b2, b1, b0 000: 38,400 波特率 001: 19,200 波特率 010: 9,600 波特率 011: 4,800 波特率 100: 2,400 波特率 101: 1,200 波特率 110: 57,600 波特率 111: 115,200 波特率	R/W	√	
		SD150.3 停止位			
		SD150.4 奇偶校验			
		SD150.5 奇偶校验允许			
		SD150.6 每个字符的数据位			
		SD150.7 自由口接收开始字符模式			
		SD150.8 自由端口接收结束字符模式			
		SD150.9 自由口字符间超时有效			
		SD150.10 自由口帧间超时有效			
		SD150.11			
		SD150.12 高低字节有效			
		SD150.13~SD150.15			
SD151	开始字符		R/W	√	
SD152	结束字符		R/W	√	
SD153	字符间超时时间	默认 0ms（忽略字符间超时）	R/W	√	0~32767ms
SD154	帧超时时间	默认 0ms（忽略帧超时）	R/W	√	0~32767ms
SD155	接收完成信息代码	第 0 位：用户终止接收置位 第 1 位：收到指定结束字置位 第 2 位：收到最大字符数置位 第 3 位：字符间超时置位 第 4 位：（帧）接收超时置位 第 5 位：奇偶检验错误时置位 第 6~15 位：保留，用户可忽略	R	√	
SD156	当前收到的字符		R	√	
SD157	当前收到的字符总数		R	√	
SD158	当前发送的字符		R	√	

13. Modbus (PORT2) 设定

地址	名称	R/W	APLC-TCP	范围
SD140	本站站号设定 (通讯口 2)	R/W	√	MOD (1~32)
SD141	通讯口 2 最大超时时间设定 (发送之后和接收之前)	R/W	√	
SD142	通讯口 2 重试次数	R/W	√	0~100
SD144	Modbus 主站的错误代码 (通讯口 2)	R	√	

14. 集成模拟量通道的设定与读取

地址号	名称	R/W	APLC-TCP	范围
SD172	A/D 通道 0 的采样平均值	R		- 10000~ + 10000
SD173	A/D 通道 0 的采样次数	R/W		1~1000
SD174	A/D 通道 1 的采样平均值	R		- 10000~ + 10000
SD175	A/D 通道 1 的采样次数	R/W		1~1000
SD178	D/A 通道 0 的输出值	R/W		- 10000~ + 10000

注：SD173 和 SD175 的默认值为 8

15. DHSP 和 DHST 指令使用

地址	名称	R/W	APLC-TCP	范围
SD180	DHSP 表格比较输出数据的高位	R/W		
SD181	DHSP 表格比较输出数据的低位	R/W		
SD182	DHST 或 DHSP 表格要比较数据高位	R/W		
SD183	DHST 或 DHSP 表格要比较数据低位	R/W		
SD184	当前正执行表格的记录号	R/W		

16. 发生错误标志

地址	名称	R/W	APLC-TCP	范围
SD191	一般模块总线错误的模块号	R	√	
SD192	特殊模块总线错误的模块号	R	√	

17. 增强定位指令

地址	名称	R/W	APLC-TCP
SD200	Y0 输出定位指令底当前值数据寄存器使用（高位）	R/W	
SD201	Y0 输出定位指令底当前值数据寄存器使用（低位）	R/W	
SD202	Y0 执行 ZRN, PLSV, DRVI, DRVA, DSZR, DVIT 指令时的最高速度 (10~100000) (高位)	R/W	
SD203	Y0 执行 ZRN, PLSV, DRVI, DRVA, DSZR, DVIT 指令时的最高速度 (10~100000) (低位)	R/W	
SD204	Y0 执行 ZRN, PLSV, DRVI, DRVA, DSZR, DVIT 指令时的基底速度(最高速度的 1/10 以下)	R/W	
SD205	Y0 执行 ZRN, DRVI, DRVA, DSZR, DVIT 指令时由基底速度 (SD204) 上升到最高速度 (SD202, SD203) 的加减速时间。 (50ms~5000ms)	R/W	
SD207	Y0 爬行速度,适用于 DSZR	R/W	
SD208	Y0 原点回归速度, 适用于 DSZR (高位)	R/W	
SD209	Y0 原点回归速度, 适用于 DSZR (低位)	R/W	
SD210	Y1 输出定位指令底当前值数据寄存器使用（高位）	R/W	
SD211	Y1 输出定位指令底当前值数据寄存器使用（低位）	R/W	
SD212	Y1 执行 ZRN, PLSV, DRVI, DRVA, DSZR, DVIT 指令时的最高速度 (10~100000) (高位)	R/W	
SD213	Y1 执行 ZRN, PLSV, DRVI, DRVA, DSZR, DVIT 指令时的最高速度 (10~100000) (低位)	R/W	
SD214	Y1 执行 ZRN, PLSV, DRVI, DRVA, DSZR, DVIT 指令时的基底速度(最高速度的 1/10 以下)	R/W	
SD215	Y1 执行 ZRN, DRVI, DRVA, DSZR, DVIT 指令时由基底速度 (SD214) 上升到最高速度 (SD212, SD213) 的加减速时间。 (50ms~5000ms)	R/W	
SD217	Y1 爬行速度, 适用于 DSZR	R/W	
SD218	Y1 原点回归速度, 适用于 DSZR (高位)	R/W	
SD219	Y1 原点回归速度, 适用于 DSZR (低位)	R/W	

附录三 保留元件表

说明	APLC-TCP	
变频器指令发送缓存区	D7940	D7969
变频器指令接收缓存区	D7970	D7999
保留网络共享区	D7700	D7763
保留网络共享区	M1400	M1911
EROMWR 指令操作区	D6000	D6999

附录四 Modbus 通讯错误代码

异常代码	异常代码意义
0x01	非法功能码
0x02	非法寄存器地址
0x03	数据个数错误
0x10	通讯超时，通讯时间超过了用户设定的通讯最大时间发生的错误
0x11	接收数据帧错误
0x12	参数错误，指令参数设置了不支持的功能码，或者非法功能码
0x13	本身的站号与指令设定的站号相同，发生错误
0x14	元件地址溢出（接收或发送的数据量超出元件存储空间）
0x16	接收和发送地址不匹配
0x17	接收和发送功能码不匹配
0x18	接收和发送的元件起始地址不匹配
0x19	接收到的数据长度不匹配
0x1a	校验错误
0x1c	指令参数设置了不支持的功能码或者非法功能码
0x1e	元件个数设置越界

附录五 变频器指令的错误代码

异常代码	异常代码意义
0x1	非法功能码
0x2	非法寄存器地址
0x3	数据错误，即数据超过上限或者下限
0x4	从站操作失败，包括数据在上下限范围之内，但是数据无效引起的错误
0x5	命令有效，正在处理中，主要应用在存储数据到非易失性存储中
0x6	从站忙，请稍后再试，主要应用在存储数据到非易失性存储中
0x18	信息帧错误：包括信息长度错误、校验错误
0x20	参数不可修改
0x21	参数运行时不可修改（保留）
0x22	参数受密码保护

附录六 系统错误代码表

错误代码	含义	错误类型	说明	APLC-TCP
0	无错误发生			√
1~9	系统保留			√
系统硬件错误				
10	SRAM 错误	系统错误	停止用户程序 错误灯常亮；消除本错误需停电检查硬件；	√
11	FLASH 错误	系统错误	停止用户程序 错误灯常亮；消除本错误需停电检查硬件；	√
12	通讯口错误	系统错误	停止用户程序 错误灯常亮；消除本错误需停电检查硬件；	√
13	实时时钟错误	系统错误	停止用户程序 错误灯常亮；消除本错误需停电检查硬件；	
14	I2C 错误	系统错误	停止用户程序 错误灯常亮；消除本错误需停电检查硬件；	√
外设错误 (20~23)				
20	本机 I/O 严重错误	系统错误	停止用户程序 错误灯常亮。消除本错误需停电检查硬件	√
21	扩展 I/O 严重错误	系统错误	错误灯闪烁 错误消失，自动清除	
22	特殊模块严重错误	系统错误	错误灯闪烁 错误消失，自动清除	√
23	实时时钟错误刷新 (系统刷新时发现读出错误的时 间)	系统错误	错误灯闪烁 错误消失，自动清除	
24	EEPROM 读写操作错误	系统错误	错误灯闪烁 错误消失，自动清除	√
25	本地模拟量错误	系统错误	错误灯闪烁 错误消失，自动清除	
26	系统特殊模块配置错误	系统错误	错误灯闪烁； 错误消失，自动清除	√
存储错误 (40~45)				
40	用户程序文件错误	系统错误	停止用户程序 (错误灯常亮) 消除条件：下载新程序/格式化	√
41	系统配置文件错误	系统错误	停止用户程序 错误灯常亮 消除条件：下载新系统配置文件/格式化	√
42	数据块文件错误	系统错误	停止用户程序 (错误灯常亮) 消除条件：下载新数据块文件/格式化	√
43	电池备份数据丢失错误	系统错误	不停止用户程序 (错误灯闪烁) 消除条件：清除元件/格式化/复位后检测无 错	
44	强制表丢失错误	系统错误	不停止用户程序 (错误灯闪烁) 消除条件：清除元件/强制操作/格式化/复位 后检测无错	√
45	用户信息文件错误	系统错误	不停止用户程序 (错误灯不指示) 消除条件：下载新程序和新数据块文件/格式 化	√
46~59	保留			
指令执行错误 (60~75)				
60	用户程序编译错误	执行错误	停止用户程序 (错误灯常亮)	√

错误代码	含义	错误类型	说明	APLC-TCP
61	用户程序运行超时	执行错误	停止用户程序（错误灯常亮）	√
62	执行到非法的用户程序指令	执行错误	停止用户程序（错误灯常亮）	√
63	指令操作数的元件类型非法	执行错误	停止用户程序（错误灯常亮）	√
64	指令操作数数值非法	执行错误	不停止用户程序执行，错误灯不指示，但会在 SD20 中指示错误类型代码	√
65	指令操作数元件编号范围超出	执行错误		√
66	子程序栈溢出	执行错误		√
67	用户中断请求队列溢出	执行错误		√
68	非法的标号跳转或子程序调用	执行错误	不停止用户程序执行，错误灯不指示，但会在 SD20 中指示错误类型代码	√
69	零除错误	执行错误		√
70	栈定义非法	执行错误	当栈尺寸、栈中元素个数小于零 栈中元素个数大于栈尺寸限制	√
71	保留			√
72	未定义用户子程序或中断子程序	执行错误		√
73	特殊模块地址无效	执行错误		√
74	访问特殊模块出错	执行错误		√
75	I/O 立即刷新出错	执行错误		√
76	时钟设置出错	执行错误		√
77	PLSR 指令参数错误	执行错误		√
78	特殊模块 BFM 缓存器超界	执行错误		
79	ABS 数据读取超时	执行错误		√
80	ABS 数据读取和校验错误	执行错误		√

附录七 Modbus 通讯协议

1. Modbus 通讯协议概述

APLC 系列小型 PLC 有两个 RS485 通讯口，一个以太网口，其通讯口 1 和通讯口 2 支持 Modbus 主站和从站（可由后台软件设置），以太网口支持 ModbusTCP 从站协议：

1. RS485 或 RS232 接口，物理接口采用 RS232 三线制
2. 支持 RTU、ASCII 格式（不支持 ASCII 结束字符的更改）
3. 作为 Modbus 从站，从站的地址范围 1~31
4. 支持广播模式，广播对写入有效，还对诊断的部分子功能码有效
5. 波特率可设置为 38, 400、19, 200 bps、9, 600 bps、4, 800 bps、2, 400 bps、1, 200 bps（默认为 19200，8 位数据，1 位停止位，偶校验）
6. 数据域：支持 2×252 个字节（ASCII 模式）、252 字节（RTU 模式）

2. 支持的 Modbus 功能码及元件寻址方式

从站支持的功能码 01, 02, 03, 05, 06, 08, 15, 16（十进制）。

读写元件功能码与元件对应关系：

功能码	功能码名称	Modicon 数据地址	可操作元件类型	注释
01	读线圈	0 ^{注1} :xxxx	Y、X、M、SM、S、T、C	读位
02	读离散量输入	1 ^{注2} :xxxx	X	读位
03	读寄存器	4 ^{注3} :xxxx ^{注4}	D、SD、Z、T、C	读字
05	写单个线圈	0:xxxx	Y、M、SM、S、T、C	写位
06	写单个寄存器	4:xxxx	D、SD、Z、T、C	写字
15	写多个线圈	0:xxxx	Y、M、SM、S、T、C	写位
16	写多个寄存器	4:xxxx	D、SD、Z、T、C	写字

注：

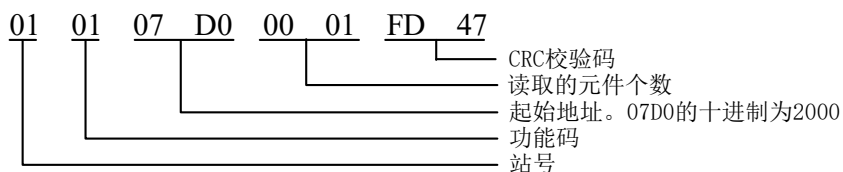
1. 0 表示线圈
2. 1 表示离散量输入
3. 4 表示寄存器
4. xxxx 表示范围 1~9999。每一种类型有独立的逻辑地址范围 1~9999（协议地址是从 0 开始的）。
5. 0、1、4 并不具备物理上的意义，不参与实际的寻址。
6. 用户不应该使用功能码 05, 15 对 X 元件写入。如果对 X 元件写入，并且写入的操作数和数据都是正确的，系统不会返回错误信息，但系统不会对写入的命令作任何操作。

PLC 元件与 Modbus 传输中的协议地址的对应关系：

元件	类型	物理元件	协议地址	支持的功能码	注释
Y	位元件	Y0~Y377 (8 进制编码) 共 256 点	0000~0255	01、05、15	输出的状态，元件编号为 Y0~Y7，Y10~Y17
X	位元件	X0~X377 (8 进制编码) 共 256 点	1200~01455 0000~0255	01、05、15 02	输入的状态，支持两种地址，元件编号同上
M	位元件	M0~M1999	2000~3999	01、05、15	
SM	位元件	SM0~SM255	4400~4655	01、05、15	
S	位元件	S0~S991	6000~6991	01、05、15	
T	位元件	T0~T255	8000~8255	01、05、15	T 元件的状态
C	位元件	C0~C255	9200~9455	01、05、15	C 元件的状态
D	字元件	D0~D7999	0000~7999	03、06、16	
SD	字元件	SD0~SD255	8000~8255	03、06、16	
Z	字元件	Z0~Z15	8500~8515	03、06、16	
T	字元件	T0~T255	9000~9255	03、06、16	T 元件的当前值
C	字元件	C0~C199	9500~9699	03、06、16	C 元件 (WORD) 的当前值
C	双字元件	C200~C255	9700~9811	03、16	C 元件 (DWORD) 的当前值

注：

协议地址是在数据传输中使用的地址，协议地址与 Modicon 的数据的逻辑地址有对应关系，协议地址是从 0 开始，Modicon 的数据的逻辑地址是从 1 开始的，也就是说协议地址 + 1 = Modicon 的数据的逻辑地址，例如：M0 协议地址是 2000，它对应的 Modicon 的数据的逻辑地址是 0：2001，在实际中对 M0 的读写是通过协议地址完成，例如对 M0 元件的读取帧（主站发出）：



异常响应描述：

异常代码	异常代码意义
0x01	非法功能码
0x02	非法寄存器地址
0x03	非法数据

在阅读过程中有几个地方需要注意：

1. X、Y 元件采用 8 进制。X0~X377 共 256 点，Y0~Y377 共 256 点，排列为 Y0~Y7，Y10~Y17，Y20~Y27 等。
2. X 元件可以有两种方式寻址，一种是协议地址为 1200~1455，对应的功能码 01、05、15，另一种协议地址 0~255 对应的功能码 02。
3. 对双字元件的处理，C 元件是计数器，它既有状态，又有当前的值，C200~C255 是 32 位的，但协议编址时，在这个范围的每个 C 元件占两个协议地址，例如：C200 它的协议地址为 9700~9701，通过 Modbus 读取时，应该起始协议地址为偶数，读取的元件数为偶数。
4. 对于大多数 SM、SD 元件可能通过 Modbus 写入时，实际的值是写不进去的，但是 PLC 从站也会返回写入完成的 OK 信息。这是允许的。

3. Modbus 功能码叙述

3.1 读取线圈状态 (0x01)

在 APLC 系列 PLC 中读取位元件，读取的最大个数是 256 个。

1. 请求帧

地址	功能码 (01H)	起始地址		元件个数		校验码 (CRC 或 LRC)
		H	L	H	L	

2. 响应帧

如果读取的地址不是 8 的倍数，剩下的位由 0 填充（由高位开始填充）。

地址	功能码 (01H)	读取的元件个数 n (字节数)	读取的数据 No.1	读取的数据 No.n	校验码 (CRC 或 LRC)
			<div style="border: 1px solid black; padding: 2px; display: inline-block;"> <div style="border: 1px solid black; width: 100%; height: 100%; position: relative;"> <div style="position: absolute; top: 0; left: 0; right: 0; height: 10px;"></div> <div style="position: absolute; bottom: 0; left: 0; right: 0; height: 10px;"></div> <div style="position: absolute; top: 50%; left: 50%; transform: translate(-50%, -50%);"> <div style="border: 1px solid black; padding: 2px; display: inline-block;"> <div style="border: 1px solid black; width: 100%; height: 100%; position: relative;"> <div style="position: absolute; top: 0; left: 0; right: 0; height: 10px;"></div> <div style="position: absolute; bottom: 0; left: 0; right: 0; height: 10px;"></div> <div style="position: absolute; top: 50%; left: 50%; transform: translate(-50%, -50%);"> <div style="border: 1px solid black; padding: 2px; display: inline-block;"> <div style="border: 1px solid black; width: 100%; height: 100%; position: relative;"> <div style="position: absolute; top: 0; left: 0; right: 0; height: 10px;"></div> <div style="position: absolute; bottom: 0; left: 0; right: 0; height: 10px;"></div> <div style="position: absolute; top: 50%; left: 50%; transform: translate(-50%, -50%);"> <div style="border: 1px solid black; padding: 2px; display: inline-block;"> <div style="border: 1px solid black; width: 100%; height: 100%; position: relative;"> <div style="position: absolute; top: 0; left: 0; right: 0; height: 10px;"></div> <div style="position: absolute; bottom: 0; left: 0; right: 0; height: 10px;"></div> <div style="position: absolute; top: 50%; left: 50%; transform: translate(-50%, -50%);"> <div style="border: 1px solid black; padding: 2px; display: inline-block;"> <div style="border: 1px solid black; width: 100%; height: 100%; position: relative;"> <div style="position: absolute; top: 0; left: 0; right: 0; height: 10px;"></div> <div style="position: absolute; bottom: 0; left: 0; right: 0; height: 10px;"></div> <div style="position: absolute; top: 50%; left: 50%; transform: translate(-50%, -50%);"> <div style="border: 1px solid black; padding: 2px; display: inline-block;"> <div style="border: 1px solid black; width: 100%; height: 100%; position: relative;"> <div style="position: absolute; top: 0; left: 0; right: 0; height: 10px;"></div> <div style="position: absolute; bottom: 0; left: 0; right: 0; height: 10px;"></div> <div style="position: absolute; top: 50%; left: 50%; transform: translate(-50%, -50%);"> <div style="border: 1px solid black; padding: 2px; display: inline-block;"> <div style="border: 1px solid black; width: 100%; height: 100%; position: relative;"> <div style="position: absolute; top: 0; left: 0; right: 0; height: 10px;"></div> <div style="position: absolute; bottom: 0; left: 0; right: 0; height: 10px;"></div> <div style="position: absolute; top: 50%; left: 50%; transform: translate(-50%, -50%);"> <div style="border: 1px solid black; padding: 2px; display: inline-block;"> <div style="border: 1px solid black; width: 100%; height: 100%; position: relative;"> <div style="position: absolute; top: 0; left: 0; right: 0; height: 10px;"></div> <div style="position: absolute; bottom: 0; left: 0; right: 0; height: 10px;"></div> <div style="position: absolute; top: 50%; left: 50%; transform: translate(-50%, -50%);"> <div style="border: 1px solid black; padding: 2px; display: inline-block;"> <div style="border: 1px solid black; width: 100%; height: 100%; position: relative;"> <div style="position: absolute; top: 0; left: 0; right: 0; height: 10px;"></div> <div style="position: absolute; bottom: 0; left: 0; right: 0; height: 10px;"></div> <div style="position: absolute; top: 50%; left: 50%; transform: translate(-50%, -50%);"> <div style="border: 1px solid black; padding: 2px; display: inline-block;"> <div style="border: 1px solid black; width: 100%; height: 100%; position: relative;"> <div style="position: absolute; top: 0; left: 0; right: 0; height: 10px;"></div> <div style="position: absolute; bottom: 0; left: 0; right: 0; height: 10px;"></div> <div style="position: absolute; top: 50%; left: 50%; transform: translate(-50%, -50%);"> <div style="border: 1px solid black; padding: 2px; display: inline-block;"> <div style="border: 1px solid black; width: 100%; height: 100%; position: relative;"> <div style="position: absolute; top: 0; left: 0; right: 0; height: 10px;"></div> <div style="position: absolute; bottom: 0; left: 0; right: 0; height: 10px;"></div> <div style="position: absolute; top: 50%; left: 50%; transform: translate(-50%, -50%);"> <div style="border: 1px solid black; padding: 2px; display: inline-block;"> <div style="border: 1px solid black; width: 100%; height: 100%; position: relative;"> <div style="position: absolute; top: 0; left: 0; right: 0; height: 10px;"></div> <div style="position: absolute; bottom: 0; left: 0; right: 0; height: 10px;"></div> <div style="position: absolute; top: 50%; left: 50%; transform: translate(-50%, -50%);"> <div style="border: 1px solid black; padding: 2px; display: inline-block;"> <div style="border: 1px solid black; width: 100%; height: 100%; position: relative;"> <div style="position: absolute; top: 0; left: 0; right: 0; height: 10px;"></div> <div style="position: absolute; bottom: 0; left: 0; right: 0; height: 10px;"></div> <div style="position: absolute; top: 50%; left: 50%; transform: translate(-50%, -50%);"> <div style="border: 1px solid black; padding: 2px; display: inline-block;"> <div style="border: 1px solid black; width: 100%; height: 100%; position: relative;"> <div style="position: absolute; top: 0; left: 0; right: 0; height: 10px;"></div> <div style="position: absolute; bottom: 0; left: 0; right: 0; height: 10px;"></div> <div style="position: absolute; top: 50%; left: 50%; transform: translate(-50%, -50%);"> <div style="border: 1px solid black; padding: 2px; display: inline-block;"> <div style="border: 1px solid black; width: 100%; height: 100%; position: relative;"> <div style="position: absolute; top: 0; left: 0; right: 0; height: 10px;"></div> <div style="position: absolute; bottom: 0; left: 0; right: 0; height: 10px;"></div> <div style="position: absolute; top: 50%; left: 50%; transform: translate(-50%, -50%);"> <div style="border: 1px solid black; padding: 2px; display: inline-block;"> <div style="border: 1px solid black; width: 100%; height: 100%; position: relative;"> <div style="position: absolute; top: 0; left: 0; right: 0; height: 10px;"></div> <div style="position: absolute; bottom: 0; left: 0; right: 0; height: 10px;"></div> <div style="position: absolute; top: 50%; left: 50%; transform: translate(-50%, -50%);"> <div style="border: 1px solid black; padding: 2px; display: inline-block;"> <div style="border: 1px solid black; width: 100%; height: 100%; position: relative;"> <div style="position: absolute; top: 0; left: 0; right: 0; height: 10px;"></div> <div style="position: absolute; bottom: 0; left: 0; right: 0; height: 10px;"></div> <div style="position: absolute; top: 50%; left: 50%; transform: translate(-50%, -50%);"> <div style="border: 1px solid black; padding: 2px; display: inline-block;"> <div style="border: 1px solid black; width: 100%; height: 100%; position: relative;"> <div style="position: absolute; top: 0; left: 0; right: 0; height: 10px;"></div> <div style="position: absolute; bottom: 0; left: 0; right: 0; height: 10px;"></div> <div style="position: absolute; top: 50%; left: 50%; transform: translate(-50%, -50%);"> <div style="border: 1px solid black; padding: 2px; display: inline-block;"> <div style="border: 1px solid black; width: 100%; height: 100%; position: relative;"> <div style="position: absolute; top: 0; left: 0; right: 0; height: 10px;"></div> <div style="position: absolute; bottom: 0; left: 0; right: 0; height: 10px;"></div> <div style="position: absolute; top: 50%; left: 50%; transform: translate(-50%, -50%);"> <div style="border: 1px solid black; padding: 2px; display: inline-block;"> <div style="border: 1px solid black; width: 100%; height: 100%; position: relative;"> <div style="position: absolute; top: 0; left: 0; right: 0; height: 10px;"></div> <div style="position: absolute; bottom: 0; left: 0; right: 0; height: 10px;"></div> <div style="position: absolute; top: 50%; left: 50%; transform: translate(-50%, -50%);"> <div style="border: 1px solid black; padding: 2px; display: inline-block;"> <div style="border: 1px solid black; width: 100%; height: 100%; position: relative;"> <div style="position: absolute; top: 0; left: 0; right: 0; height: 10px;"></div> <div style="position: absolute; bottom: 0; left: 0; right: 0; height: 10px;"></div> <div style="position: absolute; top: 50%; left: 50%; transform: translate(-50%, -50%);"> <div style="border: 1px solid black; padding: 2px; display: inline-block;"> <div style="border: 1px solid black; width: 100%; height: 100%; position: relative;"> <div style="position: absolute; top: 0; left: 0; right: 0; height: 10px;"></div> <div style="position: absolute; bottom: 0; left: 0; right: 0; height: 10px;"></div> <div style="position: absolute; top: 50%; left: 50%; transform: translate(-50%, -50%);"> <div style="border: 1px solid black; padding: 2px; display: inline-block;"> <div style="border: 1px solid black; width: 100%; height: 100%; position: relative;"> <div style="position: absolute; top: 0; left: 0; right: 0; height: 10px;"></div> <div style="position: absolute; bottom: 0; left: 0; right: 0; height: 10px;"></div> <div style="position: absolute; top: 50%; left: 50%; transform: translate(-50%, -50%);"> <div style="border: 1px solid black; padding: 2px; display: inline-block;"> <div style="border: 1px solid black; width: 100%; height: 100%; position: relative;"> <div style="position: absolute; top: 0; left: 0; right: 0; height: 10px;"></div> <div style="position: absolute; bottom: 0; left: 0; right: 0; height: 10px;"></div> <div style="position: absolute; top: 50%; left: 50%; transform: translate(-50%, -50%);"> <div style="border: 1px solid black; padding: 2px; display: inline-block;"> <div style="border: 1px solid black; width: 100%; height: 100%; position: relative;"> <div style="position: absolute; top: 0; left: 0; right: 0; height: 10px;"></div> <div style="position: absolute; bottom: 0; left: 0; right: 0; height: 10px;"></div> <div style="position: absolute; top: 50%; left: 50%; transform: translate(-50%, -50%);"> <div style="border: 1px solid black; padding: 2px; display: inline-block;"> <div style="border: 1px solid black; width: 100%; height: 100%; position: relative;"> <div style="position: absolute; top: 0; left: 0; right: 0; height: 10px;"></div> <div style="position: absolute; bottom: 0; left: 0; right: 0; height: 10px;"></div> <div style="position: absolute; top: 50%; left: 50%; transform: translate(-50%, -50%);"> <div style="border: 1px solid black; padding: 2px; display: inline-block;"> <div style="border: 1px solid black; width: 100%; height: 100%; position: relative;"> <div style="position: absolute; top: 0; left: 0; right: 0; height: 10px;"></div> <div style="position: absolute; bottom: 0; left: 0; right: 0; height: 10px;"></div> <div style="position: absolute; top: 50%; left: 50%; transform: translate(-50%, -50%);"> <div style="border: 1px solid black; padding: 2px; display: inline-block;"> <div style="border: 1px solid black; width: 100%; height: 100%; position: relative;"> <div style="position: absolute; top: 0; left: 0; right: 0; height: 10px;"></div> <div style="position: absolute; bottom: 0; left: 0; right: 0; height: 10px;"></div> <div style="position: absolute; top: 50%; left: 50%; transform: translate(-50%, -50%);"> <div style="border: 1px solid black; padding: 2px; display: inline-block;"> <div style="border: 1px solid black; width: 100%; height: 100%; position: relative;"> <div style="position: absolute; top: 0; left: 0; right: 0; height: 10px;"></div> <div style="position: absolute; bottom: 0; left: 0; right: 0; height: 10px;"></div> <div style="position: absolute; top: 50%; left: 50%; transform: translate(-50%, -50%);"> <div style="border: 1px solid black; padding: 2px; display: inline-block;"> <div style="border: 1px solid black; width: 100%; height: 100%; position: relative;"> <div style="position: absolute; top: 0; left: 0; right: 0; height: 10px;"></div> <div style="position: absolute; bottom: 0; left: 0; right: 0; height: 10px;"></div> <div style="position: absolute; top: 50%; left: 50%; transform: translate(-50%, -50%);"> <div style="border: 1px solid black; padding: 2px; display: inline-block;"> <div style="border: 1px solid black; width: 100%; height: 100%; position: relative;"> <div style="position: absolute; top: 0; left: 0; right: 0; height: 10px;"></div> <div style="position: absolute; bottom: 0; left: 0; right: 0; height: 10px;"></div> <div style="position: absolute; top: 50%; left: 50%; transform: translate(-50%, -50%);"> <div style="border: 1px solid black; padding: 2px; display: inline-block;"> <div style="border: 1px solid black; width: 100%; height: 100%; position: relative;"> <div style="position: absolute; top: 0; left: 0; right: 0; height: 10px;"></div> <div style="position: absolute; bottom: 0; left: 0; right: 0; height: 10px;"></div> <div style="position: absolute; top: 50%; left: 50%; transform: translate(-50%, -50%);"> <div style="border: 1px solid black; padding: 2px; display: inline-block;"> <div style="border: 1px solid black; width: 100%; height: 100%; position: relative;"> <div style="position: absolute; top: 0; left: 0; right: 0; height: 10px;"></div> <div style="position: absolute; bottom: 0; left: 0; right: 0; height: 10px;"></div> <div style="position: absolute; top: 50%; left: 50%; transform: translate(-50%, -50%);"> <div style="border: 1px solid black; padding: 2px; display: inline-block;"> <div style="border: 1px solid black; width: 100%; height: 100%; position: relative;"> <div style="position: absolute; top: 0; left: 0; right: 0; height: 10px;"></div> <div style="position: absolute; bottom: 0; left: 0; right: 0; height: 10px;"></div> <div style="position: absolute; top: 50%; left: 50%; transform: translate(-50%, -50%);"> <div style="border: 1px solid black; padding: 2px; display: inline-block;"> <div style="border: 1px solid black; width: 100%; height: 100%; position: relative;"> <div style="position: absolute; top: 0; left: 0; right: 0; height: 10px;"></div> <div style="position: absolute; bottom: 0; left: 0; right: 0; height: 10px;"></div> <div style="position: absolute; top: 50%; left: 50%; transform: translate(-50%, -50%);"> <div style="border: 1px solid black; padding: 2px; display: inline-block;"> <div style="border: 1px solid black; width: 100%; height: 100%; position: relative;"> <div style="position: absolute; top: 0; left: 0; right: 0; height: 10px;"></div> <div style="position: absolute; bottom: 0; left: 0; right: 0; height: 10px;"></div> <div style="position: absolute; top: 50%; left: 50%; transform: translate(-50%, -50%);"> <div style="border: 1px solid black; padding: 2px; display: inline-block;"> <div style="border: 1px solid black; width: 100%; height: 100%; position: relative;"> <div style="position: absolute; top: 0; left: 0; right: 0; height: 10px;"></div> <div style="position: absolute; bottom: 0; left: 0; right: 0; height: 10px;"></div> <div style="position: absolute; top: 50%; left: 50%; transform: translate(-50%, -50%);"> <div style="border: 1px solid black; padding: 2px; display: inline-block;"> <div style="border: 1px solid black; width: 100%; height: 100%; position: relative;"> <div style="position: absolute; top: 0; left: 0; right: 0; height: 10px;"></div> <div style="position: absolute; bottom: 0; left: 0; right: 0; height: 10px;"></div> <div style="position: absolute; top: 50%; left: 50%; transform: translate(-50%, -50%);"> <div style="border: 1px solid black; padding: 2px; display: inline-block;"> <div style="border: 1px solid black; width: 100%; height: 100%; position: relative;"> <div style="position: absolute; top: 0; left: 0; right: 0; height: 10px;"></div> <div style="position: absolute; bottom: 0; left: 0; right: 0; height: 10px;"></div> <div style="position: absolute; top: 50%; left: 50%; transform: translate(-50%, -50%);"> <div style="border: 1px solid black; padding: 2px; display: inline-block;"> <div style="border: 1px solid black; width: 100%; height: 100%; position: relative;"> <div style="position: absolute; top: 0; left: 0; right: 0; height: 10px;"></div> <div style="position: absolute; bottom: 0; left: 0; right: 0; height: 10px;"></div> <div style="position: absolute; top: 50%; left: 50%; transform: translate(-50%, -50%);"> <div style="border: 1px solid black; padding: 2px; display: inline-block;"> <div style="border: 1px solid black; width: 100%; height: 100%; position: relative;"> <div style="position: absolute; top: 0; left: 0; right: 0; height: 10px;"></div> <div style="position: absolute; bottom: 0; left: 0; right: 0; height: 10px;"></div> <div style="position: absolute; top: 50%; left: 50%; transform: translate(-50%, -50%);"> <div style="border: 1px solid black; padding: 2px; display: inline-block;"> <div style="border: 1px solid black; width: 100%; height: 100%; position: relative;"> <div style="position: absolute; top: 0; left: 0; right: 0; height: 10px;"></div> <div style="position: absolute; bottom: 0; left: 0; right: 0; height: 10px;"></div> <div style="position: absolute; top: 50%; left: 50%; transform: translate(-50%, -50%);"> <div style="border: 1px solid black; padding: 2px; display: inline-block;"> <div style="border: 1px solid black; width: 100%; height: 100%; position: relative;"> <div style="position: absolute; top: 0; left: 0; right: 0; height: 10px;"></div> <div style="position: absolute; bottom: 0; left: 0; right: 0; height: 10px;"></div> <div style="position: absolute; top: 50%; left: 50%; transform: translate(-50%, -50%);"> <div style="border: 1px solid black; padding: 2px; display: inline-block;"> <div style="border: 1px solid black; width: 100%; height: 100%; position: relative;"> <div style="position: absolute; top: 0; left: 0; right: 0; height: 10px;"></div> <div style="position: absolute; bottom: 0; left: 0; right: 0; height: 10px;"></div> <div style="position: absolute; top: 50%; left: 50%; transform: translate(-50%, -50%);"> <div style="border: 1px solid black; padding: 2px; display: inline-block;"> <div style="border: 1px solid black; width: 100%; height: 100%; position: relative;"> <div style="position: absolute; top: 0; left: 0; right: 0; height: 10px;"></div> <div style="position: absolute; bottom: 0; left: 0; right: 0; height: 10px;"></div> <div style="position: absolute; top: 50%; left: 50%; transform: translate(-50%, -50%);"> <div style="border: 1px solid black; padding: 2px; display: inline-block;"> <div style="border: 1px solid black; width: 100%; height: 100%; position: relative;"> <div style="position: absolute; top: 0; left: 0; right: 0; height: 10px;"></div> <div style="position: absolute; bottom: 0; left: 0; right: 0; height: 10px;"></div> <div style="position: absolute; top: 50%; left: 50%; transform: translate(-50%, -50%);"> <div style="border: 1px solid black; padding: 2px; display: inline-block;"> <div style="border: 1px solid black; width: 100%; height: 100%; position: relative;"> <div style="position: absolute; top: 0; left: 0; right: 0; height: 10px;"></div> <div style="position: absolute; bottom: 0; left: 0; right: 0; height: 10px;"></div> <div style="position: absolute; top: 50%; left: 50%; transform: translate(-50%, -50%);"> <div style="border: 1px solid black; padding: 2px; display: inline-block;"> <div style="border: 1px solid black; width: 100%; height: 100%; position: relative;"> <div style="position: absolute; top: 0; left: 0; right: 0; height: 10px;"></div> <div style="position: absolute; bottom: 0; left: 0; right: 0; height: 10px;"></div> <div style="position: absolute; top: 50%; left: 50%; transform: translate(-50%, -50%);"> <div style="border: 1px solid black; padding: 2px; display: inline-block;"> <div style="border: 1px solid black; width: 100%; height: 100%; position: relative;"> <div style="position: absolute; top: 0; left: 0; right: 0; height: 10px;"></div> <div style="position: absolute; bottom: 0; left: 0; right: 0; height: 10px;"></div> <div style="position: absolute; top: 50%; left: 50%; transform: translate(-50%, -50%);"> <div style="border: 1px solid black; padding: 2px; display: inline-block;"> <div style="border: 1px solid black; width: 100%; height: 100%; position: relative;"> <div style="position: absolute; top: 0; left: 0; right: 0; height: 10px;"></div> <div style="position: absolute; bottom: 0; left: 0; right: 0; height: 10px;"></div> <div style="position: absolute; top: 50%; left: 50%; transform: translate(-50%, -50%);"> <div style="border: 1px solid black; padding: 2px; display: inline-block;"> <div style="border: 1px solid black; width: 100%; height: 100%; position: relative;"> <div style="position: absolute; top: 0; left: 0; right: 0; height: 10px;"></div> <div style="position: absolute; bottom: 0; left: 0; right: 0; height: 10px;"></div> <div style="position: absolute; top: 50%; left: 50%; transform: translate(-50%, -50%);"> <div style="border: 1px solid black; padding: 2px; display: inline-block;"> <div style="border: 1px solid black; width: 100%; height: 100%; position: relative;"> <div style="position: absolute; top: 0; left: 0; right: 0; height: 10px;"></div> <div style="position: absolute; bottom: 0; left: 0; right: 0; height: 10px;"></div> <div style="position: absolute; top: 50%; left: 50%; transform: translate(-50%, -50%);"> <div style="border: 1px solid black; padding: 2px; display: inline-block;"> <div style="border: 1px solid black; width: 100%; height: 100%; position: relative;"> <div style="position: absolute; top: 0; left: 0; right: 0; height: 10px;"></div> <div style="position: absolute; bottom: 0; left: 0; right: 0; height: 10px;"></div> <div style="position: absolute; top: 50%; left: 50%; transform: translate(-50%, -50%);"> <div style="border: 1px solid black; padding: 2px; display: inline-block;"> <div style="border: 1px solid black; width: 100%; height: 100%; position: relative;"> <div style="position: absolute; top: 0; left: 0; right: 0; height: 10px;"></div> <div style="position: absolute; bottom: 0; left: 0; right: 0; height: 10px;"></div> <div style="position: absolute; top: 50%; left: 50%; transform: translate(-50%, -50%);"> <div style="border: 1px solid black; padding: 2px; display: inline-block;"> <div style="border: 1px solid black; width: 100%; height: 100%; position: relative;"> <div style="position: absolute; top: 0; left: 0; right: 0; height: 10px;"></div> <div style="position: absolute; bottom: 0; left: 0; right: 0; height: 10px;"></div> <div style="position: absolute; top: 50%; left: 50%; transform: translate(-50%, -50%);"> <div style="border: 1px solid black; padding: 2px; display: inline-block;"> <div style="border: 1px solid black; width: 100%; height: 100%; position: relative;"> <div style="position: absolute; top: 0; left: 0; right: 0; height: 10px;"></div> <div style="position: absolute; bottom: 0; left: 0; right: 0; height: 10px;"></div> <div style="position: absolute; top: 50%; left: 50%; transform: translate(-50%, -50%);"> <div style="border: 1px solid black; padding: 2px; display: inline-block;"> <div style="border: 1px solid black; width: 100%; height: 100%; position: relative;"> <div style="position: absolute; top: 0; left: 0; right: 0; height: 10px;"></div> <div style="position: absolute; bottom: 0; left: 0; right: 0; height: 10px;"></div> <div style="position: absolute; top: 50%; left: 50%; transform: translate(-50%, -50%);"> <div style="border: 1px solid black; padding: 2px; display: inline-block;"> <div style="border: 1px solid black; width: 100%; height: 100%; position: relative;"> <div style="position: absolute; top: 0; left: 0; right: 0; height: 10px;"></div> <div style="position: absolute; bottom: 0; left: 0; right: 0; height: 10px;"></div> <div style="position: absolute; top: 50%; left: 50%; transform: translate(-50%, -50%);"> <div style="border: 1px solid black; padding: 2px; display: inline-block;"> <div style="border: 1px solid black; width: 100%; height: 100%; position: relative;"> <div style="position: absolute; top: 0; left: 0; right: 0; height: 10px;"></div> <div style="position: absolute; bottom: 0; left: 0; right: 0; height: 10px;"></div> <div style="position: absolute; top: 50%; left: 50%; transform: translate(-50%, -50%);"> <div style="border: 1px solid black; padding: 2px; display: inline-block;"> <div style="border: 1px solid black; width: 100%; height: 100%; position: relative;"> <div style="position: absolute; top: 0; left: 0; right: 0; height: 10px;"></div> <div style="position: absolute; bottom: 0; left: 0; right: 0; height: 10px;"></div> <div style="position: absolute; top: 50%; left: 50%; transform: translate(-50%, -50%);"> <div style="border: 1px solid black; padding: 2px; display: inline-block;"> <div style="border: 1px solid black; width: 100%; height: 100%; position: relative;"> <div style="position: absolute; top: 0; left: 0; right: 0; height: 10px;"></div> <div style="position: absolute; bottom: 0; left: 0; right: 0; height: 10px;"></div> <div style="position: absolute; top: 50%; left: 50%; transform: translate(-50%, -50%);"> <div style="border: 1px solid black; padding: 2px; display: inline-block;"> <div style="border: 1px solid black; width: 100%; height: 100%; position: relative;"> <div style="position: absolute; top: 0; left: 0; right: 0; height: 10px;"></div> <div style="position: absolute; bottom: 0; left: 0; right: 0; height: 10px;"></div> <div style="position: absolute; top: 50%; left: 50%; transform: translate(-50%, -50%);"> <div style="border: 1px solid black; padding: 2px; display: inline-block;"> <div style="border: 1px solid black; width: 100%; height: 100%; position: relative;"> <div style="position: absolute; top: 0; left: 0; right: 0; height: 10px;"></div> <div style="position: absolute; bottom: 0; left: 0; right: 0; height: 10px;"></div> <div style="position: absolute; top: 50%; left: 50%; transform: translate(-50%, -50%);"> <div style="border: 1px solid black; padding: 2px; display: inline-block;"> <div style="border: 1px solid black; width: 100%; height: 100%; position: relative;"> <div style="position: absolute; top: 0; left: 0; right: 0; height: 10px;"></div> <div style="position: absolute; bottom: 0; left: 0; right: 0; height: 10px;"></div> <div style="position: absolute; top: 50%; left: 50%; transform: translate(-50%, -50%);"> <div style="border: 1px solid black; padding: 2px; display: inline-block;"> <div style="border: 1px solid black; width: 100%; height: 100%; position: relative;"> <div style="position: absolute; top: 0; left: 0; right: 0; height: 10px;"></div> <div style="position: absolute; bottom: 0; left: 0; right: 0; height: 10px;"></div> <div style="position: absolute; top: 50%; left: 50%; transform: translate(-50%, -50%);"> <div style="border: 1px solid black; padding: 2px; display: inline-block;"> <div style="border: 1px solid black; width: 100%; height: 100%; position: relative;"> <div style="position: absolute; top: 0; left: 0; right: 0; height: 10px;"></div> <div style="position: absolute; bottom: 0; left: 0; right: 0; height: 10px;"></div> <div style="position: absolute; top: 50%; left: 50%; transform: translate(-50%, -50%);"> <div style="border: 1px solid black; padding: 2px; display: inline-block;"> <div style="border: 1px solid black; width: 100%; height: 100%; position: relative;"> <div style="position: absolute; top: 0; left: 0; right: 0; height: 10px;"></div> <div style="position: absolute; bottom: 0; left: 0; right: 0; height: 10px;"></div> <div style="position: absolute; top: 50%; left: 50%; transform: translate(-50%, -50%);"> <div style="border: 1px solid black; padding: 2px; display: inline-block;"> <div style="border: 1px solid black; width: 100%; height: 100%; position: relative;"> <div style="position: absolute; top: 0; left: 0; right: 0; height: 10px;"></div> <div style="position: absolute; bottom: 0; left: 0; right: 0; height: 10px;"></div> <div style="position: absolute; top: 50%; left: 50%; transform: translate(-50%, -50%);"> <div style="border: 1px solid black; padding: 2px; display: inline-block;"> <div style="border: 1px solid black; width: 100%; height: 100%; position: relative;"> <div style="position: absolute; top: 0; left: 0; right: 0; height: 10px;"></div> <div style="position: absolute; bottom: 0; left: 0; right: 0; height: 10px;"></div> <div style="position: absolute; top: 50%; left: 50%; transform: translate(-50%, -50%);"> <div style="border: 1px solid black; padding: 2px; display: inline-block;"> <div style="border: 1px solid black; width: 100%; height: 100%; position: relative;"> <div style="position: absolute; top: 0; left: 0; right: 0; height: 10px;"></div> <div style="position: absolute; bottom: 0; left: 0; right: 0; height: 10px;"></div> <div style="position: absolute; top: 50%; left: 50%; transform: translate(-50%, -50%);"> <div style="border: 1px solid black; padding: 2px; display: inline-block;"> <div style="border: 1px solid black; width: 100%; height: 100%; position: relative;"> <div style="position: absolute; top: 0; left: 0; right: 0; height: 10px;"></div> <div style="position: absolute; bottom: 0; left: 0; right: 0; height: 10px;"></div> <div style="position: absolute; top: 50%; left: 50%; transform: translate(-50%, -50%);"> <div style="border: 1px solid black; padding: 2px; display: inline-block;"> <div style="border: 1px solid black; width: 100%; height: 100%; position: relative;"> <div style="position: absolute; top: 0; left: 0; right: 0; height: 10px;"></div> <div style="position: absolute; bottom: 0; left: 0; right: 0; height: 10px;"></div> <div style="position: absolute; top: 50%; left: 50%; transform: translate(-50%, -50%);"> <div style="border: 1px solid black; padding: 2px; display: inline-block;"> <div style="border: 1px solid black; width: 100%; height: 100%; position: relative;"> <div style="position: absolute; top: 0; left: 0; right: 0; height: 10px;"></div> <div style="position: absolute; bottom: 0; left: 0; right: 0; height: 10px;"></div> <div style="position: absolute; top: 50%; left: 50%; transform: translate(-50%, -50%);"> <div style="border: 1px solid black; padding: 2px; display: inline-block;"> <div style="border: 1px solid black; width: 100%; height: 100%; position: relative;"> <div style="position: absolute; top: 0; left: 0; right: 0; height: 10px;"></div> <div style="position: absolute; bottom: 0; left: 0; right: 0; height: 10px;"></div> <div style="position: absolute; top: 50%; left: 50%; transform: translate(-50%, -50%);"> <div style="border: 1px solid black; padding: 2px; display: inline-block;"> <div style="border: 1px solid black; width: 100%; height: 100%; position: relative;"> <div style="position: absolute; top: 0; left: 0; right: 0; height: 10px;"></div> <div style="position: absolute; bottom: 0; left: 0; right: 0; height: 10px;"></div> <div style="position: absolute; top: 50%; left: 50%; transform: translate(-50%, -50%);"> <div style="border: 1px solid black; padding: 2px; display: inline-block;"> <div style="border: 1px solid black; width: 100%; height: 100%; position: relative;"> <div style="position: absolute; top: 0; left: 0; right: 0; height: 10px;"></div> <div style="position: absolute; bottom: 0; left: 0; right: 0; height: 10px;"></div> <div style="position: absolute; top: 50%; left: 50%; transform: translate(-50%, -50%);"> <div style="border: 1px solid black; padding: 2px; display: inline-block;"> <div style="border: 1px solid black; width: 100%; height: 100%; position: relative;"> <div style="position: absolute; top: 0; left: 0; right: 0; height: 10px;"></div> <div style="position: absolute; bottom: 0; left: 0; right: 0; height: 10px;"></div> <div style="position: absolute; top: 50%; left: 50%; transform: translate(-50%, -50%);"> <div style="border: 1px solid black; padding: 2px; display: inline-block;"> <div style="border: 1px solid black; width: 100%; height: 100%; position: relative;"> <div style="position: absolute; top: 0; left: 0; right: 0; height: 10px;"></div> </div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div>			

3.4 强置 (写) 单线圈 (0x05)

强置 (写) 单线圈是向从站写入位元件值, 允许广播 (broadcast), 即写入所有从站的相同元件。最多支持 1 个位元件。

1. 请求帧

地址	功能码 (05H)	起始地址		写入的元件值		校验码 (CRC 或 LRC)
		H	L	H	L	

注: 写入元件的值为 0xFF00(ON,1)或者 0x0000(OFF,0)

2. 响应帧

响应帧是请求帧的重复。

地址	功能码 (05H)	起始地址		写入的元件值		校验码 (CRC 或 LRC)
		H	L	H	L	

3.5 预置 (写) 单寄存器 (0x06)

强置 (写) 单寄存器是向从站写入字元件值, 允许广播, 即写入所有从站的相同元件。最多支持 1 个字元件。

1. 请求帧

地址	功能码 (06H)	起始地址		写入的元件值		校验码 (CRC 或 LRC)
		H	L	H	L	

2. 响应帧

响应帧是请求帧的重复。

地址	功能码 (06H)	起始地址		写入的元件值		校验码 (CRC 或 LRC)
		H	L	H	L	

3.6 回送诊断校验 (0x08)

回送诊断校验, 可以得到诊断寄存器并得到通讯错误信息。

诊断码	含义
0x00	请求帧返回
0x 01	重启通讯选项
0x 04	从站进入 LISTEN ONLY 模式
0x0a	清除计数器和诊断寄存器
0x0b	返回总线报文计数
0x0c	返回总线 CRC 错误计数
0x0d	返回从站异常差错计数
0x0e	返回从站报文计数
0x0f	返回从站无响应计数
0x12	返回总线字符超限计数

子功能码的帧描述如下：

请求帧返回（0x00）：

1. 请求帧

地址	功能码 (0x08H)	功能字码		任意字符		校验码 (CRC 或 LRC)
		(0x00H)	(0x00H)	H	L	

2. 响应帧

将请求帧原样返回

地址	功能码 (0x08H)	功能字码		任意字符		校验码 (CRC 或 LRC)
		(0x00H)	(0x00H)	H	L	

重新启动通讯选项（0x01）：

当收到该帧时，可以将 PLC 带出只听模式。（支持广播帧）

1. 请求帧

当数据域正常为 0x00 00 或者 0xff 00。

地址	功能码 (0x08H)	功能字码		数据域		校验码 (CRC 或 LRC)
		0x00H	0x01H	H	L	

2. 响应帧

地址	功能码 (0x08H)	功能字码		数据域		校验码 (CRC 或 LRC)
		0x00H	0x01H	H	L	

从站进入 LISTEN ONLY 模式（0x04）：

从站进入 LISTEN ONLY 模式，命令都不执行，也不作回应，从站只认重新启动通讯选项命令，并且当收到该命令后，从站进入在线模式。（支持广播帧）

1. 请求帧

地址	功能码 (0x08H)	功能字码		数据域		校验码 (CRC 或 LRC)
		(0x00H)	(0x04H)	0x00H	0x00H	

2. 响应帧

无返回

清计数器和诊断寄存器（0x0A）：

清除所有的计数器（支持广播帧）。

1. 请求帧

地址	功能码 (0x08H)	功能字码		数据域		校验码 (CRC 或 LRC)
		(0x00H)	(0x0AH)	0x00H	0x00H	

2. 响应帧

地址	功能码 (0x08H)	功能字码		数据域		校验码 (CRC 或 LRC)
		(0x00H)	(0x0AH)	0x00H	0x00H	

返回总线报文计数 (0x0B) :

记录从上一次启动、清除计数器或加电之后, 从站在到的所有主站发出的报文总数, 不包括 CRC 错误的报文。

1. 请求帧

地址	功能码 (0x08H)	功能字码		数据域		校验码 (CRC 或 LRC)
		(0x00H)	(0x0BH)	0x00H	0x00H	

2. 响应帧

地址	功能码 (0x08H)	功能字码		数据域		校验码 (CRC 或 LRC)
		(0x00H)	(0x0BH)	H	L	

CRC 错误计数值 (0x0C) :

记录从上一次启动、清除计数器或加电之后, 从站在到的 CRC 错误的数量。

1. 请求帧

地址	功能码 (0x08H)	功能字码		数据域		校验码 (CRC 或 LRC)
		(0x00H)	(0x0CH)	0x00H	0x00H	

2. 响应帧

功能码 (0x08H)	功能字码		数据域		校验码 (CRC 或 LRC)
	(0x00H)	(0x0CH)	H	L	

返回从站异常差错计数 (0x0D) :

记录从上一次启动、清除计数器或加电之后, 从站检测到的异常差错数量, 也包括广播报文中的检测到的差错。

1. 请求帧

地址	功能码 (0x08H)	功能字码		数据域		校验码 (CRC 或 LRC)
		(0x00H)	(0x0DH)	0x00H	0x00H	

2. 响应帧

地址	功能码 (0x08H)	功能字码		数据域		校验码 (CRC 或 LRC)
		(0x00H)	(0x0DH)	H	L	

返回从站报文计数 (0x0E) :

记录从上一次启动、清除计数器或加电之后, 从站收到的对从站寻址的报文数目。

1. 请求帧

地址	功能码 (0x08H)	功能字码		数据域		校验码 (CRC 或 LRC)
		(0x00H)	(0x0EH)	0x00H	0x00H	

2. 响应帧

地址	功能码 (0x08H)	功能字码		数据域		校验码 (CRC 或 LRC)
		(0x00H)	(0x0EH)	H	L	

返回从站无响应计数 (0x0F) :

记录从上一次启动、清除计数器或加电之后, 从站没有返回的报文数量。

1. 请求帧

地址	功能码 (0x08H)	功能字码		数据域		校验码 (CRC 或 LRC)
		(0x00H)	(0x0FH)	0x00H	0x00H	

2. 响应帧

地址	功能码 (0x08H)	功能字码		数据域		校验码 (CRC 或 LRC)
		(0x00H)	(0x0FH)	H	L	

从站收到字符超限计数值 (0x12) :

记录从上一次启动、清除计数器或加电之后, 由于收到的字符超限导致无法寻址的的报文数量。

1. 请求帧

地址	功能码 (0x08H)	功能字码		数据域		校验码 (CRC 或 LRC)
		(0x00H)	(0x12H)	0x00H	0x00H	

2. 响应帧

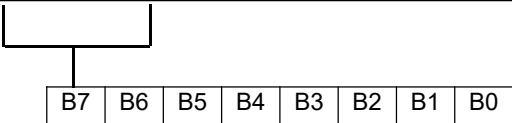
地址	功能码 (0x08H)	功能字码		数据域		校验码 (CRC 或 LRC)
		(0x00H)	(0x12H)	H	L	

3.7 强置 (写) 多线圈(0x0F)

最多能写 1968 个位元件 (0x07b0) ,根据元件的定义范围有变化。

1. 请求帧

地址	功能码 (0FH)	起始地址		元件个数		字节数 (n)	写入元件 值 No.1	写入元件值 No.N	校验码(CRC 或 LRC)
		H	L	H	L					



2. 响应帧

地址	功能码 (0FH)	起始地址		元件个数		校验码 (CRC 或 LRC)
		H	L	H	L	

3.8 预置（写）多寄存器(0x10 Hex)

最多写 120 个寄存器（0x78）。

1. 请求帧

地址	功能码 (0x10H)	起始地址		元件 个数 (n)	写入元件值 No.1		写入元件值 No.N		校验码(CRC 或 LRC)
		H	L		H	L		H	L	

2. 响应帧

地址	功能码 (0x10H)	起始地址		元件个数		校验码 (CRC 或 LRC)
		H	L	H	L	

3.9 故障响应帧（0x80+功能码）

响应帧：

地址	功能码	异常代码（见 2. 支持的 Modbus 功能码及元件寻址方式）	校验码 (CRC 或 LRC)
----	-----	----------------------------------	--------------------

功能码是截获所请求帧的功能码 + 0x80

3.10 注意事项

1. 参考软元件的地址划分，每次读取的软元件类型为同一类，例如不能在一帧中将 X 和 Y 元件一起读回来。
2. 读取该类软元件的地址和数据范围不能超过协议中规定的范围。举例如下：

已知 Y 元件的协议地址范围 0000~0255（Y0~Y377）：

- 如果读取的起始地址是 1，读取的元件个数是 256，则会返回地址错误（异常码 02），因为从 1 起始的 Y 元件只有 255 个；
- 如果读取的起始地址是 0，读取的元件个数是 257，则会返回数据错误（异常码 03），因为读取元件的个数超过了 256，实际只定义了 256 个 Y 元件；
- 如果读取的起始地址为 0，读取元件的个数是 256 则会返回 256 个元件的状态；

也就是必须保证读取的元件是实际定义的（在范围内）。读、写字元件和位元件都是如此。

4. Modbus 通讯控制举例

Modbus 从站不主动发送任何报文，只有接收到对本地的信息后才根据具体情况看是否响应主站的报文。从站仅支持 Modbus 功能码 01，02，03，05，06，08，15，16，其余的均以不合法的功能码响应（除广播帧）。

对元件读写的处理：

除了 08 功能码，其他支持的功能码都是对元件读写的，原则上讲最多允许一帧读 2000 个位元件，写 1968 个位元件，读取 125 个字元件，写 120 个字元件。但由于实际的协议地址对不同的元件是分开的，不连续（例如 Y377 的协议地址是 255，X0 的协议地址是 1200），因此在对元件的读写操作时，一次读取的元件只能是一种类型的元件，而读写元件的最多数目也与实际定义的元件个数有关系，例如读取 Y 元件，Y0~Y377，协议地址范围 0~255，对应的 Modicon 的数据的逻辑地址是 1~256，在读取 Y 元件时最多允许读取 256 个元件，举例如下：

1. 主站发送：01 01 00 00 01 00 3D 9A

01 地址，01 功能码，00 00 起始地址，01 00 读取元件个数 3D 9A 校验

从站应答：会返回正确的应答

2. 主站发送：01 01 00 00 01 01 FC 5A

主站读取 01 01 个元件也就是 257 个，超出了定义的 Y 元件的范围

从站应答：01 81 03 00 51

从站应答是非法的数据值，原因是 257>256，256 是最大允许的 Y 元件数

3. 主站发送：01 01 00 64 00 A0 7D AD

主站读取起始地址 00 64（十进制 100），元件个数 00 A0（十进制 160）

从站应答：01 81 02 C1 91

从站应答非法数据地址，从 100 开始的 Y 元件只有 156 个读 160 个非法。

4. 主站发送：01 01 01 2C 00 0A 7C 38

主站读取 01 2C（十进制 300）的 10 个位元件

从站应答：01 81 02 C1 91

从站应答非法数据地址，由于协议地址 300 没有位元件的定义。

5. 主站发送：01 04 00 02 00 0A D1 CD

主站发送功能码 04 的帧

从站应答：01 84 01 82 C0

从站应答非法的功能码

6. 主站发送：01 02 00 00 00 0A F8 0D

主站读取输入元件（X 元件）从起始地址 00 00 读 10 个（X0~X9）

从站返回：01 02 02 00 00 B9 B8

从站返回了正确信息 02 个字节内容是 00 00

7. 主站发送：01 01 04 B0 00 0A BC DA

主站读取 04 B0（十进制 1200）开始的 10 个位元件，也是 X0~X9

从站应答：01 01 02 00 00 B9 FC

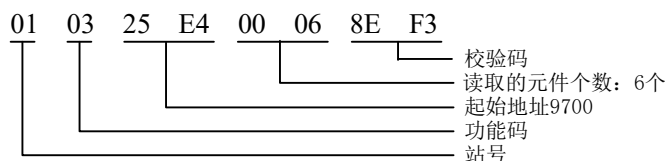
从站应答 02 个字节内容是 00 00

注意

1. 从站的地址都是 01，后两个字节都是 CRC 校验码，第二字节是功能码。
2. X 元件不支持写入。

对双字元件的处理：

1. C 元件的当前计数值为字元件或双字元件，C200~C255 为双字元件，对 C200~C255 的读写也是通过读写寄存器的功能码（03、16）来完成，每两个寄存器的地址对应一个 C 双字元件，读写时只能成对的读取寄存器。例如读取 C200~C202 三个 C 双字元件 RTU 帧：



返回的数据中 9700、9701 两个地址表示 C200 的内容，9700 为高 16 位，9701 为低 16 位。

2. 在读取双字元件时如果读取的开始地址不是偶数，会返回错误代码非法地址。举例如下：

主站发送：01 03 25 E5 00 04 5E F2

主站发送读取开始地址是 25 E5（十进制 9701）的四个字元件

从站响应：01 83 02 C0 F1

从站应答：非法的数据地址

3. 如果读取的个数不是偶数，会返回错误代码非法的数据，举例如下：

主站发送：01 03 25 E4 00 05 CE F2

主站读取开始地址是 25E4 的 5 个字元件

从站应答：01 83 03 01 31

从站返回非法的数据

对 LONG INT 类型数据的处理：

根据 APLC 系列 PLC 元件的存储方式，一个 LONG INT 类型，可能存在两个 D 元件内，例如：D3, D4 存一个 LONG INT 型的数，EMERSON PLC 认为 D3 存储的是高 16 位，D4 存储的是低 16 位，当主站通过 Modbus 读取 LONG INT 数据时，读回数据后也应该按照 EMERSON PLC 对 LONG INT 的存储原则重组 32 位的数据。

FLOAT 的存储原则等同于 LONG INT 的存储原则。

5. 对广播的描述

从站支持广播，但不是每一个功能码都支持，从站支持的功能码 01, 02, 03, 05, 06, 08, 15, 16（十进制），其中为 01, 02, 03 读取元件不支持广播，发送了广播会没有返回；05, 06, 15, 16 是写元件支持广播的功能码，发送了广播没有返回，但从站会处理接收的数据；08 是诊断功能码，它的子功能码 0x01, 0x04, 0x0A（16 进制）支持广播，其他的不支持广播。

附录八 ASCII 码字符编码表

ASCII HEX 码		高 3 位							
		0	1	2	3	4	5	6	7
低 4 位	0	NUL	DLE	SPACE	0	@	P	` (顿号)	p
	1	SOH	DC1	!	1	A	Q	a	q
	2	STX	DC2	"	2	B	R	b	r
	3	ETX	DC3	#	3	C	S	c	s
	4	EOT	DC4	\$	4	D	T	d	t
	5	ENQ	NAK	%	5	E	U	e	u
	6	ACK	SYN	&	6	F	V	f	v
	7	BEL	ETB	, (单引号)	7	G	W	g	w
	8	BS	CAN	(8	H	X	h	x
	9	HT	EM)	9	I	Y	i	y
	A	LF	SUB	*	:	J	Z	j	z
	B	VT	ESC	+	;	K	[k	{
	C	FF	FS	, (逗号)	<	L		l	1 (竖线)
	D	CR	GS	- (减号)	=	M]	m	}
	E	SO	RS	.	>	N	^	n	~
	F	SI	US	/	?	O	- (下划线)	o	DEL

附录九 指令排序索引表

指令		指令功能说明	步长	影响标志位	APLC-TCP	页数
	AND<>	整数比较 AND<>指令	5		√	242
	AND=	整数比较 AND=指令	5		√	242
	AND>	整数比较 AND>指令	5		√	242
	AND>=	整数比较 AND>=指令	5		√	242
	ANDD<	长整数比较 AND<指令	7		√	245
	ANDD<=	长整数比较 AND<=指令	7		√	245
	ANDD<>	长整数比较 AND<>指令	7		√	245
	ANDD=	长整数比较 AND=指令	7		√	245
	ANDD>	长整数比较 AND>指令	7		√	245
	ANDD>=	长整数比较 AND>=指令	7		√	245
	ANDR<	浮点数比较 AND>指令	7		√	248
	ANDR<=	浮点数比较 AND<=指令	7		√	248
	ANDR<>	浮点数比较 AND<>指令	7		√	248
	ANDR=	浮点数比较 AND=指令	7		√	248
	ANDR>	浮点数比较 AND>指令	7		√	248
	ANDR>=	浮点数比较 AND>=指令	7		√	248
	ANI	常闭触点与指令	1		√	77
	ASC	ASCII 码转换指令	19		√	152
	ATI	ASCII 码数转换 16 位 16 进制指令	7		√	154
	ALT	交替输出指令	11	零、进位、借位		217
	ABSD	凸轮绝对控制指令	9	零、进位、借位		217
B	BAND	字位触点 AND 指令	5		√	237
	BANI	字位触点 ANI 指令	5		√	238
	BCD	字转换 16 位 BCD 码指令	5		√	147
	BIN	16 位 BCD 码转换字指令	5		√	148
	BITS	字中 ON 位统计指令	5		√	235
	BLD	字位触点 LD 指令	5		√	236
	BLDI	字位触点 LDI 指令	5		√	237
	BMOV	块数据传输指令	7		√	109
	BOR	字位触点 OR 指令	5		√	238
	BORI	字位触点 ORI 指令	5		√	239
	BOUT	字位线圈输出指令	5		√	240
	BRST	字位线圈清除指令	5		√	241
	BSET	字位线圈置位指令	5		√	240
C	CALL	用户子程序调用	由子程序所带的参数决定		√	105

指令		指令功能说明	步长	影响标志位	APLC-TCP	页数
	CCITT	CCITT 校验指令	7		√	230
	CFEND	用户主程序条件结束	1		√	104
	CIRET	用户中断子程序条件返回	1		√	105
	CJ	条件跳转指令	3		√	103
	COS	浮点数 COS 指令	7	零	√	138
	CRC16	CRC16 校验指令	7		√	231
	CSRET	用户子程序条件返回	1		√	106
	CTR	16 位循环计数指令	5		√	93
	CTU	16 位增计数器指令	5		√	92
	CMP	整数比较置位指令	7		√	250
D	DADD	长整数加法指令	10	零、进位、借位	√	125
	DBCD	双字转换 32 位 BCD 码指令	7		√	147
	DBIN	32 位 BCD 码转换双字指令	7		√	149
	DBITS	双字中 ON 位统计指令	6		√	236
	DCMP<	日期小于比较指令	7			182
	DCMP<=	日期大于等于比较指令	7			182
	DCMP<>	日期不等比较指令	7			182
	DCMP=	日期相等比较指令	7			182
	DCMP>	日期大于比较指令	7			182
	DCMP>=	日期大于等于比较指令	7			182
	DCNT	32 位计数指令	7		√	94
	DDAT	长整数减一指令	4		√	129
	DDIV	长整数除法指令	10		√	127
	DAT	整数减一指令	3		√	123
	DATO	解码指令	5		√	234
	DFLT	长整数转换浮点数指令	7		√	145
	DFMOV	数据块双字填充指令	9		√	111
	DFROM	特殊模块缓冲寄存器双字读指令	10		√	172
	DGBIN	32 位格雷码转换双字指令	7		√	151
	DGRY	双字转换 32 位格雷码指令	7		√	150
	DHSCI	高速计数比较中断触发指令	10			188
	DHSCR	高速计数比较复位指令	10			189
	DHSCS	高速计数比较置位指令	10			186
	DHSP	高速计数表格比较脉冲输出指令	10			193
	DHST	高速计数表格比较指令	10			192
	DHSZ	高速计数区间比较指令	13			190
	DI	中断禁止指令	1		√	105
	DINC	长整数增一指令	4		√	128
	DINT	浮点数转换长整数指令	7	零、进位、借位	√	146

指令		指令功能说明	步长	影响标志位	APLC-TCP	页数
	DIV	整数除法指令	7		√	121
	DMOV	双字数据传输指令	7		√	108
	DMUL	长整数乘法指令	10		√	126
	DNEG	长整数取负指令	7		√	130
	DRCL	32 位带进位循环左移指令	9	进位	√	166
	DRCR	32 位带进位循环右移指令	9	进位	√	165
	DROL	32 位循环左移指令	9	进位	√	164
	DROR	32 位循环右移指令	9	进位	√	163
	DSHL	32 位左移指令	9		√	169
	DSHR	32 位右移指令	9		√	168
	DSQT	长整数算术平方根指令	7		√	127
	DSUB	长整数减法指令	10	零、进位、借位	√	125
	DSUM	长整数累加指令	9	零	√	131
	DTI	长整数转换整数指令	6		√	143
	DTO	特殊模块缓冲寄存器双字写指令	10		√	172
D	DVABS	长整数绝对值指令	7		√	129
	DWAND	双字与指令	10		√	157
	DWINV	双字取非指令	10		√	159
	DWOR	双字或指令	10		√	158
	DWXOR	双字异或指令	10		√	158
	DXCH	双字交换指令	7		√	113
	DABSD	凸轮绝对控制指令	11	零、进位、借位		217
E	ED	下降沿检测指令	1		√	82
	EI	中断使能指令	1		√	104
	ENCO	编码指令	5		√	234
	EROMWR	EEPROM 写指令	7		√	174
	EU	上升沿检测指令	2		√	82
	EVDFWD	变频器点动正转指令	6			221
	EVDREV	变频器点动反转指令	6			222
	EVFRQ	设置变频器频率指令	8			223
	EVFWD	变频器正转指令	6			220
	EVRD	读取变频器单个寄存器值指令	10			226
	EVRDST	读取变频器状态指令	10			225
	EVREV	变频器反转指令	6			221
	EVSTOP	变频器停止指令	8			223
	EVWRT	写单个寄存器值指令	10		√	224
	EXP	浮点数自然数幂指令 EXP	7	零、进位	√	141
F	FIFO	先入先出指令	7	零	√	114
	FLT	整数转换浮点数指令	6		√	144

指令	指令功能说明	步长	影响标志位	APLC-TCP	页数
	FMOV	数据块填充指令	7	√	110
	FOR	循环指令	3	√	100
	FROM	特殊模块缓冲寄存器字读指令	9	√	172
G	GBIN	16 位格雷码转换字指令	5	√	151
	GRY	字转换为 16 位格雷码指令	5	√	149
H	HACKLE	锯齿波信号输出指令	12	√	213
	HCNT	高速计数器驱动指令	7	√	185
	HOURL	计时表指令	8	√	181
I	INC	整数增一指令	3	√	122
	INT	浮点数转换整数指令	6	零、进位、借位	145
	INV	能流取反指令	1	√	83
	ITA	16 位 16 进制数转换 ASCII 码指令	7	√	153
	ITD	整数转换长整数指令	6	√	143
L	LBL	跳转标号定义指令	3	√	102
	LD	常开触点指令	1	√	76
	LD<	整数比较 LD<指令	5	√	241
	LD<=	整数比较 LD<=指令	5	√	241
	LD<>	整数比较 LD<>指令	5	√	241
	LD=	整数比较 LD=指令	5	√	241
	LD>	整数比较 LD>指令	5	√	241
	LD>=	整数比较 LD>=指令	5	√	241
	LDD<	长整数比较 LD<指令	7	√	244
	LDD<=	长整数比较 LD<=指令	7	√	244
	LDD<>	长整数比较 LD<>指令	7	√	244
	LDD=	长整数比较 LD=指令	7	√	244
	LDD>	长整数比较 LD>指令	7	√	244
	LDD>=	长整数比较 LD>=指令	7	√	244
	LDI	常闭触点指令	1	√	76
	LDR<	浮点数比较 LD<指令	7	√	247
	LDR<=	浮点数比较 LD<=指令	7	√	247
	LDR<>	浮点数比较 LD<>指令	7	√	247
	LDR=	浮点数比较 LD=指令	7	√	247
	LDR>	浮点数比较 LD>指令	7	√	247
	LDR>=	浮点数比较 LD>=指令	7	√	247
	LIFO	后入先出指令	7	零	115
	LN	浮点数自然对数指令 LN	7	零、进位	140
	LRC	LRC 校验指令	7	√	232
M	MC	主控指令	3	√	84
	MCR	主控清除指令	1	√	85
	Modbus	Modbus 主站通讯指令	8	√	218
	MOV	字数据传输指令	5	√	107
	MPP	输出能流栈出栈指令	1	√	81
	MPS	输出能流入栈指令	1	√	80
	MRD	读输出能流栈顶值指令	1	√	81
	MUL	整数乘法指令	8	√	120

指令	指令功能说明	步长	影响标志位	APLC-TCP	页数
MODRW	MODBUS 读写指令	14		√	229
N	NEG	整数取负指令	5	√	124
	NEXT	循环返回	1	√	100
	NOP	空操作指令	1	√	84
	OR	常开触点或指令	1	√	78
O	OR<	整数比较 OR<指令	5	√	243
	OR<=	整数比较 OR<=指令	5	√	243
	OR<>	整数比较 OR<>指令	5	√	243
	OR=	整数比较 OR=指令	5	√	243
	OR>	整数比较 OR>指令	5	√	243
	OR>=	整数比较 OR>=指令	5	√	243
	ORB	能流块或指令	1	√	80
	ORD<	长整数比较 OR<指令	7	√	246
	ORD<=	长整数比较 OR<=指令	7	√	246
	ORD<>	长整数比较 OR<>指令	7	√	246
	ORD=	长整数比较 OR=指令	7	√	246
	ORD>	长整数比较 OR>指令	7	√	246
	ORD>=	长整数比较 OR>=指令	7	√	246
	ORI	常闭触点或指令	1	√	78
	ORR<	浮点数比较 OR>指令	7	√	249
	ORR<=	浮点数比较 OR<=指令	7	√	249
	ORR<>	浮点数比较 OR<>指令	7	√	249
	ORR=	浮点数比较 OR=指令	7	√	249
	ORR>	浮点数比较 OR>指令	7	√	249
	ORR>=	浮点数比较 OR>=指令	7	√	249
	OUT	线圈输出指令	1	√	79
	OUT Sxx	SFC 状态跳转	3	√	87
P	PID	PID 功能指令	9		207
	PLS	包络线指令	7		201
	PLSR	带加速的计数脉冲输出指令	10		199
P	PLSY	计数脉冲输出指令	9		197
	POWER	浮点数求幂指令	10	零、进位	√ 139
	PUSH	数据入栈指令	7	进位	√ 113
	PWM	PWM 脉冲输出指令	7		205
	PLSB	带基底频率与加减速的计数脉冲输出指令	12	零、进位、借位	203
R	RADD	浮点数加法指令	10	零、进位	√ 133
	RAMP	斜坡信号输出指令	12		211
	RCL	16 位带进位循环左移指令	7	进位	√ 163

指令		指令功能说明	步长	影响标志位	APLC-TCP	页数
	RCR	16 位带进位循环右移指令	7	进位	√	162
	RCV	自由口接收（RCV）指令	7		√	228
	RDIV	浮点数除法指令	10	零、进位	√	135
	REF	I/O 立即刷新指令	5		√	173
	REFF	设置输入滤波常数指令	3		√	173
R	RET	SFC 程序结束	1		√	87
	RMOV	浮点数数据传输指令	7		√	109
	RMUL	浮点数乘法指令	10	零、进位	√	134
	RNEG	浮点数取负指令	7		√	137
	ROL	16 位循环左移指令	7	进位	√	161
	ROR	16 位循环右移指令	7	进位	√	160
	RSQT	浮点数算术平方根指令	7	零	√	135
	RST	线圈清除指令	1		√	84
	RST Sxx	SFC 状态清除	3		√	87
	RSUB	浮点数减法指令	10	零、进位	√	133
	RSUM	浮点数累加指令	9		√	141
	RVABS	浮点数绝对值指令	7		√	136
	RCMP	浮点数比较置位指令	9		√	250
	S	SEG	字转换 7 段码	5		√
SET		线圈置位指令	1		√	83
SET Sxx		SFC 状态转移	3		√	86
SFTL		位串左移指令	9		√	170
SFTR		位串右移指令	9		√	169
SHL		16 位左移指令	7		√	167
SHR		16 位右移指令	7		√	166
SIN		浮点数 SIN 指令	7	零	√	137
SPD		SPD 测频指令	7		√	196
SQT		整数算术平方根指令	5		√	121
STL		SFC 状态装载指令	3		√	86
STOP		用户程序停止	1		√	105
SUB		整数减法指令	7	零、进位、借位	√	119
SUM		整数累加指令	8	零	√	131
SWAP		高低字节交换指令	3		√	112
T	TADD	时钟加指令	7	零、进位		178
	TAN	浮点数 TAN 指令	7	零、进位	√	139
	TCMP<	时间小于比较指令	7			183
	TCMP<=	时间大于等于比较指令	7			183
	TCMP<>	时间不等比较指令	7			183
	TCMP=	时间相等比较指令	7			183
	TCMP>	时间大于比较指令	7			183

指令	指令功能说明	步长	影响标志位	APLC-TCP	页数
	TCMP>=	时间大于等于比较指令	7		183
	TMON	不重触发单稳计时指令	5		91
	TO	特殊模块缓冲寄存器字写指令	9	√	172
T	TOF	断开延时时指令	5	√	90
	TON	接通延时时指令	5	√	88
	TONR	记忆型接通延时时指令	5	√	89
	TRD	实时时钟读指令	3	√	176
	TRIANGLE	三角波信号输出指令	12	√	215
	TSUB	时钟减指令	7	零、借位	179
	TWR	实时时钟写指令	3	√	177
V	VABS	整数绝对值指令	5	√	123
	VRRD	读模拟电位器值指令	5	√	172
W	WAND	字与指令	7	√	155
	WDT	用户程序看门狗清零	1	√	104
	WINV	字取非指令	5	√	156
	WOR	字或指令	7	√	155
	WSFL	字串左移动指令	9	√	117
	WSFR	字串右移动指令	9	√	116
	WXOR	字异或指令	7	√	156
X	XCH	字交换指令	5	√	112
	XMT	自由口发送 (XMT) 指令	7	√	227
	ZRST	批量位清零指令	5	√	233
	ZSET	批量位置位指令	5	√	233

附录十 指令分类索引表

指令	指令功能说明	步长	影响标志位	APLC-TCP	页数
基本指令	LD	常开触点指令	1	√	76
	LDI	常闭触点指令	1	√	76
	AND	常开触点与指令	1	√	77
	ANI	常闭触点与指令	1	√	77
	OR	常开触点或指令	1	√	78
	ORI	常闭触点或指令	1	√	78
	OUT	线圈输出指令	1	√	79
	SET	线圈置位指令	1	√	83
	RST	线圈清除指令	1	√	84
	ANB	能流块与指令	1	√	79
	ORB	能流块或指令	1	√	80
	INV	能流取反指令	1	√	83
	NOP	空操作指令	1	√	84
	MPS	输出能流入栈指令	1	√	80
	MRD	读输出能流栈顶值指令	1	√	81
	MPP	输出能流栈出栈指令	1	√	81
	MC	主控指令	3	√	84
	MCR	主控清除指令	1	√	85
	EU	上升沿检测指令	2	√	82
	ED	下降沿检测指令	2	√	82
	TON	接通延时计时指令	5	√	88
	TOF	断开延时计时指令	5	√	90
	TMON	不重触发单稳计时指令	5	√	91
	TONR	记忆型接通延时计时指令	5	√	89
	CTU	16 位增计数器指令	5	√	92
	CTR	16 位循环计数指令	5	√	93
	DCNT	32 位计数指令	7	√	94
程序流控制指令	LBL	跳转标号定义指令	3	√	102
	CJ	条件跳转指令	3	√	103
	CALL	用户子程序调用	由程序决定	√	105
	CSRET	用户子程序条件返回	1	√	106
	CFEND	用户主程序条件结束	1	√	104
	CIRET	用户中断子程序条件返回	1	√	105
	FOR	循环指令	3	√	100
	NEXT	循环返回	1	√	100
	WDT	用户程序看门狗清零	1	√	104
	STOP	用户程序停止	1	√	105

指令		指令功能说明	步长	影响标志位	APLC-TCP	页数
	EI	中断使能指令	1		√	104
	DI	中断禁止指令	1		√	105
SFC 指令	STL	SFC 状态装载指令	3		√	86
	SET Sxx	SFC 状态转移	3		√	86
	OUT Sxx	SFC 状态跳转	3		√	87
	RST Sxx	SFC 状态清除	3		√	87
	RET	SFC 程序结束	1		√	87
数据传输指令	MOV	字数据传输指令	5		√	107
	DMOV	双字数据传输指令	7		√	108
	RMOV	浮点数数据传输指令	7		√	109
	BMOV	块数据传输指令	7		√	109
	SWAP	高低字节交换指令	3		√	112
数据流	XCH	字交换指令	5		√	112
	DXCH	双字交换指令	7		√	113
	FMOV	数据块填充指令	7		√	110
数据流	DFMOV	数据块双字填充指令	9		√	111
	WSFR	字串右移动指令	9		√	116
	WSFL	字串左移动指令	9		√	117
	PUSH	数据入栈指令	7	进位	√	113
	FIFO	先入先出指令	7	零	√	114
	LIFO	后入先出指令	7	零	√	115
整数/长整数 算数运算指令	ADD	整数加法指令	7	零、进位、借位	√	119
	DADD	长整数加法指令	10	零、进位、借位	√	125
	SUB	整数减法指令	7	零、进位、借位	√	119
	DSUB	长整数减法指令	10	零、进位、借位	√	125
	INC	整数增一指令	3		√	122
	DINC	长整数增一指令	4		√	128
	DAT	整数减一指令	3		√	123
	DDAT	长整数减一指令	4		√	129
	MUL	整数乘法指令	8		√	120
	DMUL	长整数乘法指令	10		√	126
	DIV	整数除法指令	7		√	121
	DDIV	长整数除法指令	10		√	127
	VABS	整数绝对值指令	5		√	123
	DVABS	长整数绝对值指令	7		√	129
	NEG	整数取负指令	5		√	124
	DNEG	长整数取负指令	7		√	130

指令	指令功能说明	步长	影响标志位	APLC-TCP	页数
	SQT	整数算术平方根指令	5	√	121
	DSQT	长整数算术平方根指令	7	√	127
	SUM	整数累加指令	8	零	√ 131
	DSUM	长整数累加指令	9	零	√ 131
浮点运算指令	RADD	浮点数加法指令	10	零、进位	√ 133
	RSUB	浮点数减法指令	10	零、进位	√ 133
	RMUL	浮点数乘法指令	10	零、进位	√ 134
	RDIV	浮点数除法指令	10	零、进位	√ 135
	RVABS	浮点数绝对值指令	7		√ 136
	RNEG	浮点数取负指令	7		√ 137
	RSQT	浮点数算术平方根指令	7	零	√ 135
	SIN	浮点数 SIN 指令	7	零	√ 137
	COS	浮点数 COS 指令	7	零	√ 138
	TAN	浮点数 TAN 指令	7	零、进位	√ 139
	LN	浮点数自然对数指令 LN	7	零、进位	√ 140
	EXP	浮点数自然数幂指令 EXP	7	零、进位	√ 141
	POWER	浮点数求幂指令	10	零、进位	√ 139
	RSUM	浮点数累加指令	9		√ 141
字/双字逻辑运算	WAND	字与指令	7		√ 155
	DWAND	双字与指令	10		√ 157
	WOR	字或指令	7		√ 155
	DWOR	双字或指令	10		√ 158
	WXOR	字异或指令	7		√ 156
	DWXOR	双字异或指令	10		√ 158
	WINV	字取非指令	5		√ 156
	DWINV	双字取非指令	7		√ 159
位移动旋转指令	ROR	16 位循环右移指令	7	进位	√ 160
	DROR	32 位循环右移指令	9	进位	√ 163
	ROL	16 位循环左移指令	7	进位	√ 161
	DROL	32 位循环左移指令	9	进位	√ 164
	RCR	16 位带进位循环右移指令	7	进位	√ 162
	DRCR	32 位带进位循环右移指令	9	进位	√ 165
	RCL	16 位带进位循环左移指令	7	进位	√ 163
	DRCL	32 位带进位循环左移指令	9	进位	√ 166
	SHR	16 位右移指令	7		√ 166
	DSHR	32 位右移指令	9		√ 168
位移动旋转指令	SHL	16 位左移指令	7		√ 167
	DSHL	32 位左移指令	9		√ 169
	SFTL	位串左移指令	9		√ 170

指令	指令功能说明	步长	影响标志位	APLC-TCP	页数
SFTR	位串右移指令	9		√	169
增强行位处理指令	DATO	5		√	234
	ENCO	5		√	234
	BITS	5		√	235
	DBITS	6		√	236
	ZRST	5		√	233
	ZSET	5		√	233
高速 I/O 指令	HCNT	7			185
	DHSCS	10			186
	DHSCR	10			189
	DHSCI	10			188
	DHSZ	13			190
	DHST	10			192
	DHSP	10			193
	SPD	7			196
	PLSY	9			197
	PLSR	10			199
	PLSB	12	零、进位、借位		203
	PWM	7			205
	PLS	7			205
控制计算指令	PID	9			207
	RAMP	12			211
	TRIANGLE	12			215
	HACKLE	12			213
	ABSD	9	零、进位、借位		217
	DABSD	11	零、进位、借位		217
	ALT	11	零、进位、借位		217
外设指令	FROM	9			172
	DFROM	10			172
	TO	9			172
外设指令	DTO	10			172
	VRRD	5			172
	REFF	3		√	173
	REF	5		√	173

指令		指令功能说明	步长	影响标志位	APLC-TCP	页数
	EROMWR	EEPROM 写指令	7			174
	PR	打印指令	5			175
实时时钟指令	TRD	实时时钟读指令	3			176
	TWR	实时时钟写指令	3			177
	TADD	时钟加指令	7	零、进位		178
	TSUB	时钟减指令	7	零、借位		179
	HOUR	计时表指令	8			181
比较触点指令	LD=	整数比较 LD=指令	5		√	241
	LDD=	长整数比较 LD=指令	7		√	244
	LDR=	浮点数比较 LD=指令	7		√	247
比较触点指令	LD>	整数比较 LD>指令	5		√	241
	LDD>	长整数比较 LD>指令	7		√	244
	LDR>	浮点数比较 LD>指令	7		√	247
	LD>=	整数比较 LD>=指令	5		√	241
	LDD>=	长整数比较 LD>=指令	7		√	244
	LDR>=	浮点数比较 LD>=指令	7		√	247
	LD<	整数比较 LD<指令	5		√	241
	LDD<	长整数比较 LD<指令	7		√	244
	LDR<	浮点数比较 LD<指令	7		√	247
	LD<=	整数比较 LD<=指令	5		√	241
	LDD<=	长整数比较 LD<=指令	7		√	244
	LDR<=	浮点数比较 LD<=指令	7		√	247
	LD<>	整数比较 LD<>指令	5		√	241
	LDD<>	长整数比较 LD<>指令	7		√	244
	LDR<>	浮点数比较 LD<>指令	7		√	247
	AND=	整数比较 AND=指令	5		√	242
	ANDD=	长整数比较 AND=指令	7		√	245
	ANDR=	浮点数比较 AND=指令	7		√	248
	AND>	整数比较 AND>指令	5		√	242
	ANDD>	长整数比较 AND>指令	7		√	245
	ANDR>	浮点数比较 AND>指令	7		√	248
	AND>=	整数比较 AND>=指令	5		√	242
	ANDD>=	长整数比较 AND>=指令	7		√	245
	ANDR>=	浮点数比较 AND>=指令	7		√	248
	AND<	整数比较 AND<指令	5		√	242
	ANDD<	长整数比较 AND<指令	7		√	245
	ANDR<	浮点数比较 AND>指令	7		√	248
	AND<=	整数比较 AND<=指令	5		√	242
	ANDD<=	长整数比较 AND<=指令	7		√	245

指令		指令功能说明	步长	影响标志位	APLC-TCP	页数
	ANDR<=	浮点数比较 AND<=指令	7		√	248
比较触点指令	AND<>	整数比较 AND<>指令	5		√	242
	ANDD<>	长整数比较 AND<>指令	7		√	245
	ANDR<>	浮点数比较 AND<>指令	7		√	248
	OR=	整数比较 OR=指令	5		√	243
	ORD=	长整数比较 OR=指令	7		√	246
	ORR=	浮点数比较 OR=指令	7		√	249
	OR>	整数比较 OR>指令	5		√	243
	ORD>	长整数比较 OR>指令	7		√	246
	ORR>	浮点数比较 OR>指令	7		√	249
	OR>=	整数比较 OR>=指令	5		√	243
	ORD>=	长整数比较 OR>=指令	7		√	246
	ORR>=	浮点数比较 OR>=指令	7		√	249
	OR<	整数比较 OR<指令	5		√	243
	ORD<	长整数比较 OR<指令	7		√	246
	ORR<	浮点数比较 OR>指令	7		√	249
	OR<=	整数比较 OR<=指令	5		√	243
	ORD<=	长整数比较 OR<=指令	7		√	246
	ORR<=	浮点数比较 OR<=指令	7		√	249
	OR<>	整数比较 OR<>指令	5		√	243
	ORD<>	长整数比较 OR<>指令	7		√	246
比较触点指令	ORR<>	浮点数比较 OR<>指令	7		√	249
	CMP	整数比较置位指令	7		√	250
比较触点指令	LCMP	长整数比较置位指令	9		√	250
	RCMP	浮点数比较置位指令	9		√	250

指令	指令功能说明	步长	影响标志位	APLC-TCP	页数
数值转换指令	ITD	整数转换长整数指令	6	√	143
	DTI	长整数转换整数指令	6	√	143
	FLT	整数转换浮点数指令	6	√	144
	DFLT	长整数转换浮点数指令	7	√	145
	INT	浮点数转换整数指令	6	零、进位、借位	√ 145
	DINT	浮点数转换长整数指令	7	零、进位、借位	√ 146
	BCD	字转换 16 位 BCD 码指令	5	√	147
	DBCD	双字转换 32 位 BCD 码指令	7	√	147
	BIN	16 位 BCD 码转换字指令	5	√	148
	DBIN	32 位 BCD 码转换双字指令	7	√	149
	GRY	字转换为 16 位格雷码指令	5	√	149
	DGRY	双字转换 32 位格雷码指令	7	√	150
	GBIN	16 位格雷码转换字指令	5	√	151
	DGBIN	32 位格雷码转换双字指令	7	√	151
	SEG	字转换 7 段码	5	√	152
	ASC	ASCII 码转换指令	19	√	152
	ITA	16 位 16 进制数转换 ASCII 码指令	7	√	153
	ATI	ASCII 码数转换 16 位 16 进制指令	7	√	154
字触点指令	BLD	字位触点 LD 指令	5	√	236
	BLDI	字位触点 LDI 指令	5	√	237
	BAND	字位触点 AND 指令	5	√	237
	BANI	字位触点 ANI 指令	5	√	238
	BOR	字位触点 OR 指令	5	√	238
	BORI	字位触点 ORI 指令	5	√	239
	BSET	字位线圈置位指令	5	√	240
	BRST	字位线圈清除指令	5	√	241
	BOUT	字位线圈输出指令	5	√	240
通讯指令	Modbus	Modbus 主站通讯指令	8	√	218
	XMT	自由口发送 (XMT) 指令	7	√	227
	RCV	自由口接收 (RCV) 指令	7	√	228
	EVFWD	变频器正转指令	6	√	220
	EVREV	变频器反转指令	6	√	221
	EVDFWD	变频器点动正转指令	6	√	221
	EVDREV	变频器点动反转指令	6	√	222
	EVSTOP	变频器停止指令	8	√	223
	EVFRQ	设置变频器频率指令	8	√	223
	EVWRT	写单个寄存器值指令	10	√	224

指令		指令功能说明	步长	影响标志位	APLC-TCP	页数
	EVRDST	读取变频器状态指令	10		√	225
	EVRD	读取变频器单个寄存器值指令	10		√	226
	MODRW	MODBUS 读写指令	14		√	229
校验指令	CCITT	CCITT 校验指令	7		√	230
	CRC16	CRC16 校验指令	7		√	231
	LRC	LRC 校验指令	7		√	232
日期比较指令	DCMP=	日期相等比较指令	7			182
	DCMP>	日期大于比较指令	7			182
	DCMP<	日期小于比较指令	7			182
	DCMP>=	日期大于等于比较指令	7			182
	DCMP<=	日期大于等于比较指令	7			182
	DCMP<>	日期不等比较指令	7			182
时间比较指令	TCMP=	时间相等比较指令	7			183
	TCMP>	时间大于比较指令	7			183
	TCMP<	时间小于比较指令	7			183
	TCMP>=	时间大于等于比较指令	7			183
	TCMP<=	时间大于等于比较指令	7			183
	TCMP<>	时间不等比较指令	7			183



关注公众号 获取技术支持