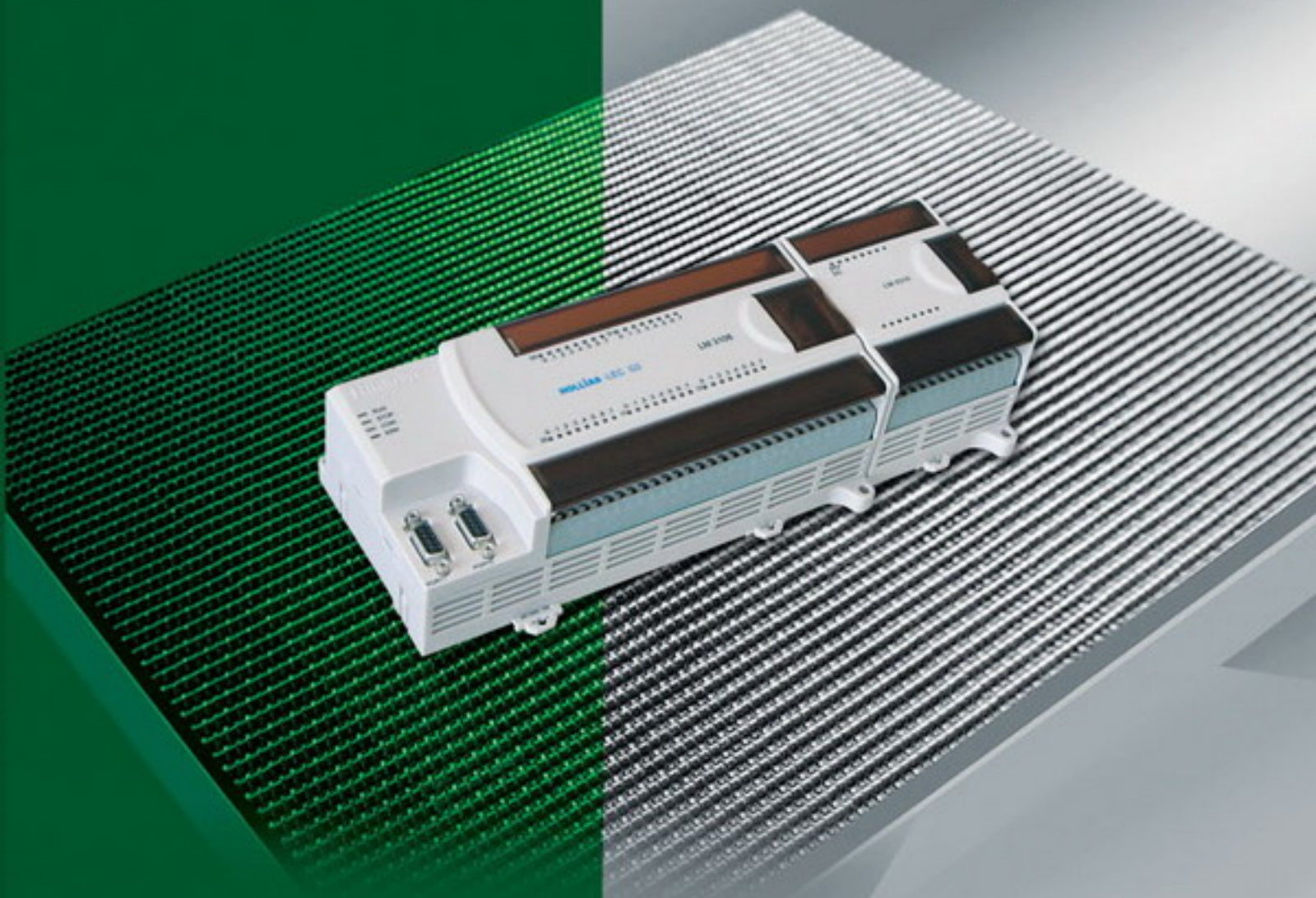
和利时 Hollysys



HOLLiAS-LEC G3系列小型一体化PLC

指令手册

北京和利时系统工程股份有限公司
杭州和利时自动化有限公司　PLC事业部

北京和利时系统工程股份有限公司
杭州和利时自动化有限公司　PLC事业部

地址：北京市海淀区西三旗建材城中路10号
邮编：100096
电话：(010)58981167/58981182
传真：(010)58981165/82923980
网址：http://www.hollysys.com

技术支持
邮箱：PLC@hollysys.com　电话：(010)58981171

版权申明

本手册内容，包括文字、图表、标志、标识、商标、产品型号、软件程序、版面设计等，均受《中华人民共和国著作权法》、《中华人民共和国商标法》、《中华人民共和国专利法》及与之适用的国际公约中有关著作权、商标权、专利权或其他财产所有权法律的保护，为北京和利时系统工程股份有限公司专属所有或持有。

本手册仅供商业用户阅读、查询，在未得到北京和利时系统工程股份有限公司特别授权的情况下，无论出于何种原因和目的，均不得用任何电子或机械方法，以任何形式复制和传递本手册的内容。否则本公司将依法追究法律责任。

我们已核对本手册中的内容、图表与所述指令相符，但误差难以避免，并不能保证完全一致。同时，我们会定期对手册的内容、图表进行检查、修改和维护，恕不另行通知。

本手册的说明、图表、简单程序及应用实例完全出于举例说明的目的，我们对其都进行了测试，但因为软件版本的更新和各种应用有许多未知的变化和要求，我们不承担根据本手册或本手册中的实例而构成的实际应用产生的责任。

北京和利时系统工程股份有限公司保留全部权利。

1993－2007 Copyright Hollysys

HOLLiAS LEC、HollySys、和利时 和利时 HollySys 的字样和徽标均为北京和利时系统工程股份有限公司的商标或注册商标。

Microsoft、Windows 和 WindowsNT 是微软公司在美国和/或其他国家分支机构的商标或注册商标。

手册中涉及到的其他商标或注册商标属于他们各自的拥有者。

联系方式

北京和利时系统工程股份有限公司
杭州和利时自动化有限公司

地址：北京市海淀区西三旗建材城中路 10 号
邮编：100096
电话：（+86）010-58981000
传真：（+86）010-58981165
网址：http：//www.hollysys.com
技术支持邮箱：PLC@hollysys.com
版本： 2007 年 5 月

前 言

HOLLiAS LEC G3 系列 PLC 是和利时公司推出的新一代小型一体化 PLC，包括多种 CPU 模块和扩展模块。同时，和利时公司还推出了功能强大的 PowerPro 编程软件及丰富的指令系统。

HOLLiAS LEC G3 系列 PLC 以其性能稳定、质量可靠、价格适中等优点，使之广泛应用于自动化领域的众多行业中，也赢得广大用户的好评。

包含内容

《HOLLiAS LEC G3 系列小型一体化 PLC 指令手册》是和利时公司对其 G3 系列 PLC 包含的所有指令详细介绍的技术手册，主要包含以下信息：

- G3 系列 PLC 指令概述
- G3 系列 PLC 的操作数与数据类型
- G3 系列 PLC 指令详细介绍
- G3 系列 PLC 部分实例

读者

本指令手册配合软件手册与硬件手册使用，适用于有一定 PLC 背景知识、掌握了 PowerPro 软件使用方法的工程师、编程人员。

使用本手册，需对 G3 系列 PLC 有一定的了解。

适用范围

本手册所讲述的所有指令适用于以下版本编程软件。

PowerPro1.0 中文版

PowerPro1.0 英文版

PowerPro2.0 中文版

PowerPro2.0 英文版

PowerPro2.1 中文版

PowerPro2.1 英文版

如何使用本手册

如果已经熟练掌握 PowerPro 编程软件，直接通过目录查找需要的指令。

如果刚刚开始学习 PowerPro 编程软件，建议阅读第 1 章 G3 PLC 指令概述。

如果对 PLC 所使用的操作数与数据类型不是很了解，建议阅读第 2、3 章。

附录 A 包含指令速查表、IEC 标准指令表、指令关联冲突速查表、硬件模块状态信息。

附录 B 包含一些应用实例，供使用者参考。

相关手册

《HOLLiAS LEC G3 系列小型一体化 PLC 软件手册》

《HOLLiAS LEC G3 系列小型一体化 PLC 硬件手册》

目 录

目 录 I
第 1 章 指令系统概述 1
1.1 指令的定义与分类 1
1.2 指令库的定义与分类 1
1.2.1 基本指令库 2
1.2.2 扩展指令库 6
1.3 指令库的添加 8
1.4 指令系统使用注意事项 9
第 2 章 操作数 11
2.1 常量 11
2.2 变量 13
2.3 地址 14
2.4 函数返回值 15
第 3 章 数据类型 16
3.1 标准数据类型 16
3.2 自定义数据类型 17
3.2.1 数组 17
3.2.2 指针 19
3.2.3 枚举 19
3.2.4 结构 20
第 4 章 基本指令 21
4.1 算术运算指令 21
4.1.1 ADD——加法指令 21
4.1.2 MUL——乘法指令 21
4.1.3 SUB——减法指令 22
4.1.4 DIV——除法指令 23
4.1.5 MOD——取余指令 23
4.2 赋值指令 24
4.3 逻辑运算指令 25
4.3.1 AND——与指令 25
4.3.2 OR——或指令 25
4.3.3 XOR——异或指令 26
4.3.4 NOT——取非指令 26
4.4 移位指令 27

4.4.1 SHL——左移指令 27
4.4.2 SHR——右移指令 27
4.4.3 ROL——循环左移指令 28
4.4.4 ROR——循环右移指令 29
4.5 选择指令 29
4.5.1 SEL——二选一指令 29
4.5.2 MAX——取最大值指令 30
4.5.3 MIN——取最小值指令 31
4.5.4 LIMIT——极限值指令 31
4.5.5 MUX——多选一指令 32
4.6 比较指令 33
4.6.1 GT——大于指令 33
4.6.2 LT——小于指令 34
4.6.3 GE——大于等于指令 34
4.6.4 LE——小于等于指令 35
4.6.5 EQ——等于指令 36
4.6.6 NE——不等于指令 36
4.7 数据类型转换指令 37
4.7.1 BOOL_TO_<TYPE>——布尔类型转换指令 39
4.7.2 BYTE_TO_<TYPE>——字节类型转换指令 40
4.7.3 WORD_TO_<TYPE>——字类型转换指令 42
4.7.4 DWORD_TO_<TYPE>——双字类型转换指令 44
4.7.5 SINT_TO_<TYPE>——短整型转换指令 45
4.7.6 USINT_TO_<TYPE>——无符号短整型转换指令 46
4.7.7 INT_TO_<TYPE>——整数类型转换指令 46
4.7.8 UINT_TO_<TYPE>——无符号整数类型转换指令 47
4.7.9 DINT_TO_<TYPE>——双整数类型转换指令 48
4.7.10 UDINT_TO_<TYPE>——无符号双整数类型转换指令 49
4.7.11 REAL_TO_<TYPE>——实数类型转换指令 51
4.7.12 TIME_TO_<TYPE>——时间类型转换指令 51
4.7.13 DATE_TO_<TYPE>——日期类型转换指令 52
4.7.14 DT_TO_<TYPE>——日期时间类型转换指令 53
4.7.15 TOD_TO_<TYPE>——时间类型转换指令 54
4.7.16 STRING_TO_<TYPE>——字符类型转换指令 55
4.7.17 TRUNC——截短转换指令 56
4.8 初等数学运算指令 57
4.8.1 ABS——绝对值指令 57
4.8.2 SQRT——平方根指令 58
4.8.3 LN——自然对数指令 58
4.8.4 LOG——常用对数指令 59
4.8.5 EXP——指数指令 59

4.8.6 SIN——正弦指令 60
4.8.7 COS——余弦指令 60
4.8.8 TAN——正切指令 61
4.8.9 ASIN——反正弦指令 61
4.8.10 ACOS——反余弦指令 62
4.8.11 ATAN——反正切指令 62
4.8.12 EXPT——幂指令 63
4.9 地址运算指令 63
4.9.1 ADR——取地址指令 63
4.9.2 ^——取地址内容指令 64
4.9.3 BITADR——位地址指令 65
4.9.4 INDEXOF——索引指令 65
4.9.5 SIZEOF——数据类型大小指令 66
4.10 调用指令 66
4.11 初始化操作指令 66
4.12 字符串处理指令（Standard.lib） 67
4.12.1 LEN——取字符串长度指令 67
4.12.2 LEFT——左边取字符串指令 68
4.12.3 RIGHT——右边取字符串指令 68
4.12.4 MID——中间取字符串指令 69
4.12.5 CONCAT——合并字符串指令 70
4.12.6 INSERT——插入字符串指令 70
4.12.7 DELETE——删除字符指令 71
4.12.8 REPLACE——替换字符串指令 71
4.12.9 FIND——查找字符串指令 72
4.13 库版本信息检查指令（Util.lib） 72
4.14 软件版本信息指令（SysLibC16x.lib） 73
4.15 检查指令（Check.lib） 74
4.15.1 CheckBounds——数组边界检查指令 74
4.15.2 CheckDivByte——字节型除数为零检查指令 75
4.15.3 CheckDivWord——字型除数为零检查指令 75
4.15.4 CheckDivDWord——双字型除数为零检查指令 76
4.15.5 CheckDivReal——实型除数为零检查指令 76
4.15.6 CheckRangeSigned——整型边界检查指令 76
4.15.7 CheckRangeUnsigned——无符号整型边界检查指令 78
4.16 BCD码转换指令（Util.lib） 78
4.16.1 BCD_TO_INT——BCD码转整型指令 78
4.16.2 INT_TO_BCD——整型转BCD码指令 79
4.17 位/字节操作指令（Util.lib） 80
4.17.1 EXTRACT——位提取指令 80

4.17.2 PACK——位整合指令 81
4.17.3 PUTBIT——位赋值指令 82
4.17.4 UNPACK——位拆分 82
4.18 高等数学运算指令（Util.lib） 84
4.18.1 DERIVATIVE——微分 84
4.18.2 INTEGRAL——积分 85
4.18.3 STATISTICS_INT——整型统计 87
4.18.4 STATISTICS_REAL——实型统计 88
4.18.5 VARIANCE——平方偏差 89
4.19 控制器指令（Util.lib） 90
4.19.1 P——比例控制器 90
4.19.2 PD——比例微分控制器 91
4.19.3 PID——比例积分微分控制器 93
4.19.4 PID_FIXCYCLE——比例积分微分控制器 95
4.20 信号发生器指令（Util.lib） 97
4.20.1 BLINK——脉冲信号发生器 97
4.20.2 GEN——典型周期信号发生器 98
4.21 函数操纵器指令（Util.lib） 100
4.21.1 CHARCURVE——特征曲线 100
4.21.2 RAMP_INT——整型限速 102
4.21.3 RAMP_REAL——实型限速 104
4.22 模拟量处理指令（Util.lib） 104
4.22.1 HYSTERESIS——滞后 104
4.22.2 LIMITALARM——上下限报警 105
4.23 双稳态指令（Standard.lib） 107
4.23.1 SR——置位优先双稳态器 107
4.23.2 RS——复位优先双稳态器 108
4.24 触发器指令（Standard.lib） 109
4.24.1 R_TRIG——上升沿检测触发器 109
4.24.2 F_TRIG——下降沿检测触发器 110
4.25 计数器（Standard.lib） 110
4.25.1 CTU——递增计数器 110
4.25.2 CTD——递减计数器 111
4.25.3 CTUD——递增递减计数器 112
4.26 定时器（Standard.lib） 114
4.26.1 TP——普通定时器 114
4.26.2 TON——通电延时定时器 115
4.26.3 TOF——断电延时定时器 116
4.26.4 RTC——实时时钟 117
第 5 章 扩展指令 119

5.1 模拟量模块处理指令（Hollysys_PLC_Analog.lib）.....119
5.1.1 Analog_IN——模拟量输入模块调用.....119
5.1.2 Analog_OUT——模拟量输出模块调用指令.....120
5.2 RS232 自由口通讯指令（Hollysys_PLC_Comm.lib）.....122
5.2.1 Set_COMM_PRMT——RS232 自由口通讯参数设置.....122
5.2.2 COMM_SEND——RS232 自由口通讯数据发送.....123
5.2.3 COMM_RECEIVE——RS232 自由口通讯数据接收.....124
5.2.4 Reset_COMM_PRMT——RS232 恢复协议设置.....125
5.3 RS485 自由口通讯指令（Hollysys_PLC_Comm2.lib）.....127
5.3.1 Set_COMM2_PRMT——RS485 自由口通讯参数设置.....127
5.3.2 COMM2_SEND——RS485 自由口通讯数据发送.....128
5.3.3 COMM2_RECEIVE——RS485 自由口通讯数据接收.....129
5.3.4 Reset_COMM2_PRMT——RS485 恢复协议设置.....130
5.4 Profibus-DP指令（Hollysys_PLC_DPSlave.lib）.....132
5.5 以太网指令（Hollysys_PLC_EtherNet.lib）.....133
5.6 正反动作PID控制器（Hollysys_PLC_Util.lib）.....135
5.7 Modbus校验指令（Hollysys_PLC_Modbus_CRC.lib）.....138
5.8 硬件实时时钟指令（Hollysys_PLC_HDRTC.lib）.....139
5.8.1 Set_HD_RTC——设置实时时钟（DT数据格式）.....139
5.8.2 Set_HD_RTC_X——设置实时时钟（普通数据格式）.....140
5.8.3 Get_HD_RTC——读取实时时钟日期/时间/星期.....141
5.9 实时时钟报警指令（Hollysys_PLC_HDRTCALM.lib）.....143
5.9.1 Get_HDRTC_ALM——读取硬件实时时钟报警时间/星期.....143
5.9.2 Set_HDRTC_ALM——设置硬件实时时钟报警时间/星期.....144
5.10 LM3104/5 实时时钟报警指令（Hollysys_PLC_HDRTCALM_N.lib）.....145
5.10.1 GET_HDRTC_ALM——获取实时时钟中断时间.....145
5.10.2 SET_HDRTC_ALM——设置实时时钟中断时间.....147
5.11 多段脉冲发送（Hollysys_PLC_PTOCtrl.lib）.....148
5.11.1 PTOCtrl_0——通道 1.1 多段脉冲发送.....148
5.11.2 PTOCtrl_1——通道 0.3 多段脉冲发送.....151
5.12 立即输出指令（Hollysys_PLC_IO.lib）.....153
5.12.1 OutPut_Bit——立即输出.....153
5.12.2 Set_INT_OutPut——设置中断立即输出.....154
5.13 工程量转换指令（Hollysys_PLC_Cnvt.lib）.....155
5.13.1 E_H——工程量数据转换为 16 进制数.....155
5.13.2 H_E——16 进制数转换为工程量数据.....157
5.14 随机数发生指令（Hollysys_PLC_Math.lib）.....158
5.15 Modbus从站地址指令（Hollysys_PLC_Ex.lib）.....159
5.15.1 SET_LOCAL_ADDRESS——设置Modbus从站通讯地址.....159

5.15.2 GET_LOCAL_ADDRESS——读取Modbus从站通讯地址 160
5.16 模拟电位器（Hollysys_PLC_Ex.lib） 161
5.17 系统看门狗复位（Hollysys_PLC_Ex.lib） 162
5.18 单相计数(Hollysys_PLC_Ex_CT.lib) 163
5.18.1 HD_CTUD_T2——T2 高速计数器 163
5.18.2 HD_CTUD_T3——T3 高速计数器 165
5.18.3 HD_CTUD_T4——T4 普通计数器 167
5.18.4 HD _CTU_T7——T7 高速计数器 168
5.19 两相计数（Hollysys_PLC_Ex_DCT.lib） 170
5.19.1 HD_DCTUD_T2——T2 两相高速计数器 170
5.19.2 HD_DCTUD_T3——T3 两相高速计数器 172
5.19.3 HD_DCTUD_T4——T4 两相普通计数器 173
5.20 两相 32 位计数（Hollysys_PLC_Ex_DCT32.lib） 175
5.21 中断定时器（Hollysys_PLC_Ex_TIMER.lib） 176
5.21.1 HD_TIMER_T7——中断定时器 176
5.21.2 HD_CLEAR_T7——重载定时器 178
5.21.3 HD_STOP_T7——停止定时器 179
5.22 外部中断（Hollysys_PLC_Ex_ExINT.lib） 180
5.22.1 Fast_ExINT——快速外部中断 180
5.22.2 Fast_ExINT_E——快速外部中断 181
5.23 脉冲输出指令（Hollysys_PLC_Ex_PT.lib） 183
5.23.1 PTO_PWM0——PTO/PWM脉冲输出 183
5.23.2 PTO_PWM1——PTO/PWM脉冲输出 185
5.24 脉冲加减速输出指令（Hollysys_PLC_Ex_PTRun.lib） 186
5.24.1 PTO_PWM0_RUN——PTO_PWM脉冲输出（加减速） 186
5.24.2 PTO_PWM1_RUN——PTO_PWM脉冲输出（加减速） 188
附录A 192
A.1 G3 指令速查表 192
A.2 IEC标准指令表 196
A.3 指令关联冲突速查表 198
A.4 硬件模块状态信息 200
A.4.1 专用寄存器区 200
A.4.2 模块诊断区 201
附录B 204
B.1 电机循环启动发送脉冲举例 204
B.2 PROFIBUS-DP模块使用举例 205
B.3 以太网指令使用举例 207

B.4 中断关联事件使用举例 213
B.5 自由口通讯使用举例 215
B.6 PID控制器使用举例 220
B.7 高速计数器使用举例 223
B.8 模拟电位器使用举例 224

第1章 指令系统概述

和利时公司 HOLLiAS LEC G3 系列小型一体化 PLC 为用户提供了丰富的指令，这些指令均可通过 G3 PLC 的编程软件 PowerPro 进行调用，操作简单，使用方便。

本手册将详细讲述 HOLLiAS LEC G3 系列 PLC 指令（简称 G3 指令）。请注意，PowerPro2.1 之前版本可以调用本手册绝大部分指令，PowerPro2.1 可以调用本手册介绍的所有指令。

1.1 指令的定义与分类

在可编程控制器中，使 CPU 完成某种操作或实现某种功能的命令及多个命令的组合称为指令，指令的集合称为指令系统。指令系统是可编程控制器硬件和软件的桥梁，是可编程控制器程序设计的基础。

PowerPro 提供了丰富的指令，按照功能不同可分为转换指令、比较指令、类型转换指令、逻辑运算指令、外部中断、两相计数等 51 种类型。为了便于理解和记忆，我们把这 51 种类型分为两大类，一类是基本指令，包括全部 IEC 标准规定指令、高等数学运算指令等，另一类是扩展指令，包括外部中断指令、脉冲输出、两相计数等与硬件端口有关的指令。扩展指令都是通过功能块方式实现的，而且对应库的名字都是以“Hollysys”开头的，具体见附录 A.1。

G3 指令在编程软件中有函数和功能块两种实现方式。函数和功能块都是 PowerPro 软件的程序组织单元，都是预先编好的、实现某种功能的程序，功能块输出可以是一个或多个结果，每一个功能块实例都有一个相关的标识符（即实例名称），函数则不需要标识符，而且只有一个输出结果（即函数的返回值），函数和功能块的具体概念可以参考《HOLLiAS LEC G3 系列小型一体化 PLC 软件手册》。附录 A.1 指令速查表中注明了指令的实现方式，FUN 表明指令是以函数方式实现，FB 则表明指令是以功能块方式实现的。

提示：

- 以函数方式实现的指令，在使用的时候都无需声明。
- 以功能块方式实现的指令，在使用的时候都需声明实例名。
- 在 G3 PLC 指令系统中有些指令需要先添加其所对应的库，才能被调用，还有部分指令没有封装在库中，可以直接被调用，在附录 A.1 中列出各个指令所在库。
- 创建工程时，Standard.lib（标准库）和 SYSLIBCALLBACK.lib（系统库）是自动添加到库管理器之中，其它库在使用时需要用户手动添加。

1.2 指令库的定义与分类

编写 PLC 程序的过程中，经常会引用一些有库指令，如字符串处理指令、触发器指令、计数器指令、PID 控制器等等。把这些具有相关功能的指令集合起来进行存储，建立专门的指令库。

指令库是 G3 PLC 指令代码的集合，所有的库都对应有库文件（库名.lib），调用某个库指

令，必需载入相应的库文件。

按照库中指令代码功能不同将其分为基本指令库、扩展指令库两类：

基本指令库指基本指令的集合。

扩展指令库指扩展指令的集合，其库名是以“Hollysys”开头。

按照库中指令执行代码所在位置的不同，指令库又可分为三类：

- 第一类：PowerPro 内部开放指令库。指令执行代码存在于库文件之中，可以使用 PowerPro 软件打开库文件，对指令的执行代码进行修改，用户也可以自己制作内部库。当程序下装到 PLC 之中，占用用户程序空间较大。
- 第二类：PowerPro 内部不开放指令库。指令执行代码存在于“库名.hex”文件之中，用户无法使用 PowerPro 软件打开库文件，对指令的执行代码进行修改。使用时应保证 hex 文件的文件名与 lib 文件的文件名一致，且存在于同一目录下。当程序下装到 PLC 之中，占用用户程序空间较大。
- 第三类：PowerPro 外部指令库。指令执行代码已经存在于 PLC 底层系统之中，用户无法修改此类库所包含的执行代码。当程序下装到 PLC 之中，占用用户程序空间较少。

本指令库分类方法有助于读者对 G3 指令系统的理解，进而更好的使用 PowerPro 软件设计程序，在下面介绍的基本指令库和扩展指令库中指出了各个指令库的类型（上面三种类型）。

提示：

- 使用 G3 指令时，必须添加相关的库文件（库名.lib）。
- PowerPro 内部指令库一经添加，即使不调用其中的指令，也会占用用户程序空间，因此在实际编程过程中，建议只添加需要的库。

1.2.1 基本指令库

✧ **标准指令库 Standard.lib**

Standard.lib 属于 PowerPro 外部指令库，在工程建立时自动添加，无需用户再次添加，包含的指令如图 1-2-1 所示。

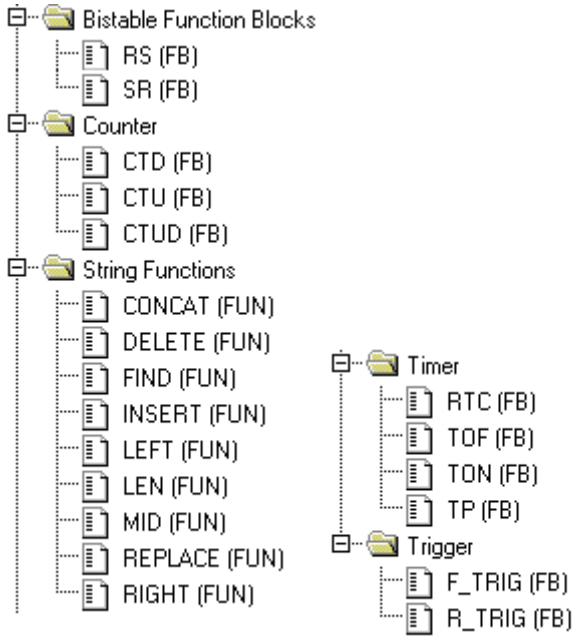


图 1-2-1

该库包含的 G3 指令的含义如表 1-2-1 所示。

表 1-2-1

双稳态指令（Bistable Function Blocks）	**计数器（Counter）**
RS（复位双稳态器） SR（置位双稳态器）	CTD（递减计数器） CTU（递增计数器） CTUD（递增递减计数器）
字符串指令（String Function）	**定时器（Timer）**
CONCAT（合并字符串指令） DELETE（删除字符串指令） FIND（查找字符串指令） INSERT（插入字符串指令） LEFT（左边取字符串指令） LEN（取字符串长度指令） MID（中间取字符串指令） REPLACE（替换字符串指令） RIGHT（右边取字符串指令）	RTC（实时时钟） TOF（断电延时定时器） TON（通电延时定时器） TP（普通定时器） **触发器（Trigger）** F_TRIG（下降沿检测触发器） R_TRIG（上升沿检测触发器）

✧ 应用指令库 Util.lib 和 Util_no_Real.lib

Util.lib 与 Util_no_Real.lib 属于 PowerPro 内部开放指令库，使用时需用户载入。Util.lib 包含的 G3 指令如图 1-2-2 所示。

- 应用指令库 Util.lib

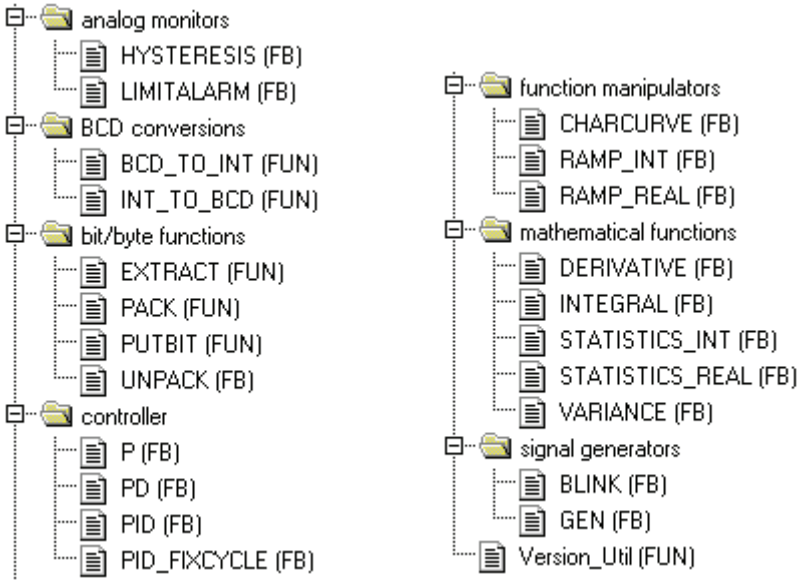


图 1-2-2

该库包含的 G3 指令的含义如表 1-2-2 所示。

表 1-2-2

模拟量处理指令	**BCD 码转换指令**
HYSTERESIS（滞后） LIMITALARM（上下限报警）	BCD_TO_INT（BCD 码转整型） INT_TO_BCD（整型转 BCD 码）
PID 控制器指令	**高等数学运算指令**
P（比例控制器） PD（比例微分控制器） PID（比例积分微分控制器） PID_FIXCYCLE （比例积分微分控制器，周期固定）	DERIVATIVE（微分） INTEGRAL（积分） STATISTICS_INT（整型统计） STATISTICS_REAL（实型统计） VARIANCE（平方偏差）
位转换指令	**函数操纵器指令**
EXTRACT（位提取） PACK（位整合） PUTBIT（位赋值） UNPACK（位拆分）	CHARCURVE（特征曲线） RAMP_INT（整型限速） RAMP_REAL（实型限速）
信号发生器指令	**库版本查看指令**
BLINK（脉冲信号发生器） GEN（典型周期信号发生器）	Version_Util（库版本查看）

- 应用指令库 Util_no_Real.lib

Util_no_Real.lib 包含的 G3 指令如图 1-2-3 所示。

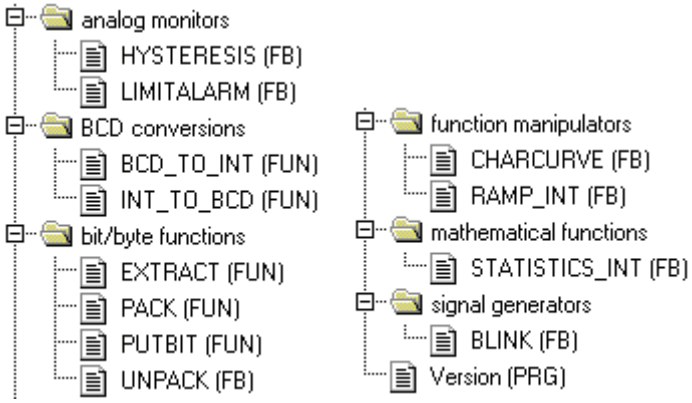


图 1-2-3

提示：

- 库 Util.lib 中指令的变量类型可为 Real 型，而库 Util_no_Real.lib 中指令的变量类型为非 Real 型，编译下载后占用用户程序空间相对少些。
- 库 Util_no_Real.lib 与库 Util.lib 不能同时添加。

✧ 系统指令库 SysLibCallback.lib 和 SysLibC16x.lib

SysLibCallback.lib 与 SysLibC16x.lib 属于 PowerPro 外部库，其包含的 G3 指令分别如图 1-2-4 与图 1-2-5 所示。SysLibCallback.lib 在工程建立时自动载入，无需用户再次载入，SysLibC16x.lib 使用时需用户载入。

- 系统指令库 SysLibCallback.lib

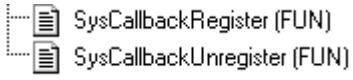


图 1-2-4

该库中指令可实现以下功能：

- 事件调用（SysCallbackRegister）
- 解除事件调用（SysCallbackUnregistrer）

- 系统指令库 SysLibC16x.lib

图 1-2-5

该库中指令可实现以下功能：

- 读取版本信息（Version）：SyslibGetVersion2300 可读取 G3 PLC 的版本信息

提示：
建立工程时 SysLibCallback.lib 库自动载入，当使用系统事件时，会自动调用该库中的指令。

✧ **检查指令库 check.lib**

check.lib 属于 PowerPro 内部开放库，其包含的 G3 指令分别如图 1-2-6 所示。

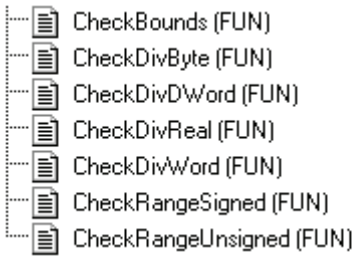


图 1-2-6

该库中指令可实现以下功能：

- 被除数为零检查功能
- 边界检查功能

✧ **IEC 动作指令库 Iecsfc.lib**

Iecsfc.lib 属于 PowerPro 内部开放库，其中只包含一个指令如图 1-2-7 所示。

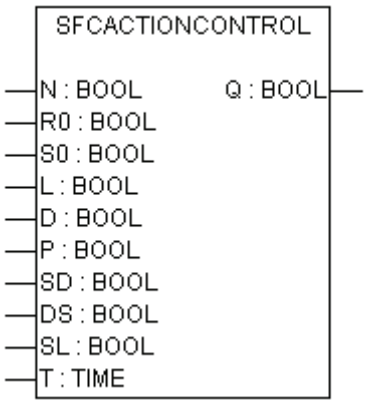


图 1-2-7

该库可实现以下功能：

- SFCActionControl：SFC 语言中 IEC 步关联动作的控制

提示：
在 SFC 编程语言中使用 IEC 步，必须加入该库，否则编译提示错误，当 IEC 步有关联动作时，系统自动调用 SFCActionControl 指令，IEC 步的具体使用方法参见软件手册。

1.2.2 扩展指令库

扩展指令库按 Util.lib 的补充指令库、指令库 Hollysys_PLC_***.lib、指令库 Hollysys_PLC_Ex_***.lib 三个部分进行介绍。

✧ **Util.lib 的补充指令库**

指令库 Hollysys_PLC_Util.lib 和 Hollysys_PLC_Modbus_CRC.lib 是指令库 Util.lib 的补充，属于 PowerPro 内部开放库，各包含的一个 G3 指令，如图 1-2-8 所示。

指令库 Hollysys_PLC_Util.lib　　Hollysys_PLC_Modbus_CRC.lib

图 1-2-8

✧ **指令库 Hollysys_PLC_***.lib**

这里***包括 ANALOG、COMM、COMM2、DPSLAVE、Ethernet、HDRTC、HDRTCALM、HDRTCALM_N、PTOCtrl、Cnvt、Math 和 IO，共 12 个指令库属于 PowerPro 外部库，每个指令库包含的 G3 指令见表 1-2-3。

表 1-2-3

Hollysys_PLC_ANALOG.lib **（模拟量模块处理指令库）**	**Hollysys_PLC_DPSLAVE.lib** **（DP 模块调用指令库）**
Analog_IN (FB) Analog_OUT (FB)	DP_Slave (FB)
Hollysys_PLC_COMM.lib **（RS232 口通讯指令库）**	**Hollysys_PLC_COMM2.lib** **（RS485 口通讯指令库）**
COMM_RECEIVE (FB) COMM_SEND (FB) Reset_COMM_PRMT (FB) Set_COMM_PRMT (FB)	COMM2_RECEIVE (FB) COMM2_SEND (FB) Reset_COMM2_PRMT (FB) Set_COMM2_PRMT (FB)
Hollysys_PLC_HDRTC.lib **（硬件实时时钟指令库）**	**Hollysys_PLC_HDRTCALM.lib** **（实时时钟报警指令库）**
Get_HD_RTC (FB) Set_HD_RTC (FB) Set_HD_RTC_X (FB)	Get_HDRTC_ALM (FB) Set_HDRTC_ALM (FB)
Hollysys_PLC_PTOCtrl.lib **（多段脉冲输出指令库）**	**Hollysys_PLC_IO.lib** **（立即输出指令库）**
PTOCtrl_0 (FB) PTOCtrl_1 (FB)	OutPut_Bit (FB) Set_INT_OutPut (FB)
Hollysys_PLC_Cnvt.lib **（工程量转换指令库）**	**Hollysys_PLC_Math.lib** **（数学指令库）**
E_H (FB) H_E (FB)	Rand (FB)
Hollysys_PLC_EtherNet.lib **（以太网模块调用指令库）**	**Hollysys_PLC_HDRTCALM_N.lib** **（LM3104/5 实时时钟报警指令库）**
EtherNet_TCP (FB)	Get_HDRTC_ALM (FB) Set_HDRTC_ALM (FB)

✧ **指令库 Hollysys_PLC_Ex_***.lib**

这里***包括 CT、DCT、DCT32、TIMER、EXINT、PT 和 PTRun 及 Hollysys_PLC_Ex.lib，共 8 个指令库，同属于 PowerPro 内部不开放库，每个指令库包含的 G3 指令见表 1-2-4。

表 1-2-4

Hollysys_PLC_Ex.lib（硬件功能指令库）	**Hollysys_PLC_Ex_CT.lib（单相计数指令库）**
Get_Local_Address (FB) HD_WDT_Reset (FB) POT (FB) Set_Local_Address (FB)	HD_CTU_T7 (FB) HD_CTUD_T2 (FB) HD_CTUD_T3 (FB) HD_CTUD_T4 (FB)
Hollysys_PLC_Ex_DCT.lib（两相计数指令库）	**Hollysys_PLC_Ex_TIMER.lib（中断定时器指令库）**
HD_DCTUD_T2 (FB) HD_DCTUD_T3 (FB) HD_DCTUD_T4 (FB)	HD_Clear_T7 (FB) HD_STOP_T7 (FB) HD_TIMER_T7 (FB)
Hollysys_PLC_Ex_DCT32.lib（两相 32 位计数指令库）	**Hollysys_PLC_Ex_ExINT.lib（外部中断指令库）**
HD_DCTUD32_T3 (FB)	Fast_ExINT (FB) Fast_ExINT_E (FB)
Hollysys_PLC_Ex_PT.lib（脉冲输出指令库）	**Hollysys_PLC_Ex_ PTRun.lib（脉冲加减速输出指令库）**
PTO_PWM0 (FB) PTO_PWM1 (FB)	PTO_PWM0_Run (FB) PTO_PWM1_Run (FB)

1.3 指令库的添加

使用库时，需要保证相应的库文件存在于如下目录：“PowerPro 安装目录\Library\”。

启动 PowerPro，选择“窗口/库管理器”，打开“库管理器”，点击右键，选择“添加库”，如图 1-3-1 所示。

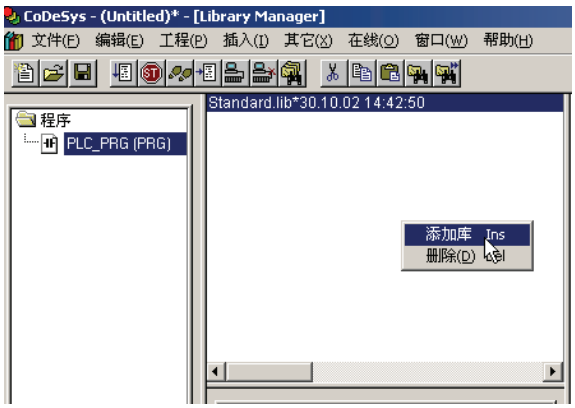


图 1-3-1

如图 1-3-2，选择需要的库文件，点击“打开”，不论哪种库只需要打开对应的*.lib 文件。

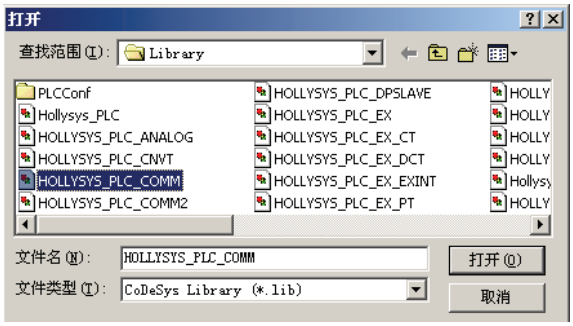


图 1-3-2

如图 1-3-3，上面选择的库被添加到列表中，该库所包含的指令显示在图中鼠标位置。

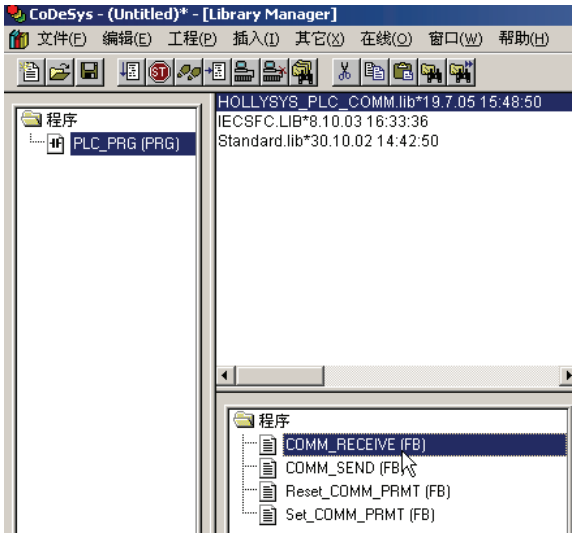


图 1-3-3

1.4 指令系统使用注意事项

- 上升沿使能，是指每当使能端由低电平变为高电平并一直保持高电平时，执行指令的相关功能。
- 上电/下装后首次使能有效，是指当使能端第一次由低电平变为高电平并一直保持高电平，执行指令的相关功能，以后再由低电平变为高电平将不会执行。如果要再次使用该指令，需要重新下装或再次上电。
- %MB0～%MB99 为系统使用，用户禁止使用，可以读取，但不能写，否则可能导致程序出错。
- 使用 Analog_IN、 Analog_OUT、DP_Slave 指令时，Address 设置必须与 PLC 配置中相关模块的节点 id 一致，只有当 EN 输入使能后才执行指令。
- 使用 Set_COMM_PRMT 指令设定自由口参数后，要想恢复原编程系统下装/调试功能，需将 RUN/STOP 开关拨到 STOP 位置，才可进行编程系统登录。

- 使用设置 Modbus 从站（SET_LOCAL_ADDRESS）指令时，如果所需通讯参数不是38400bps、8 位、无校验，则需要使用 Reset_COMM_PRMT 指令设置相应的通讯参数，然后再使用 SET_LOCAL_ADDRESS 指令设置从站地址，程序下装前需要将RUN/STOP 开关拨到 STOP 位置，程序下装后再将 RUN/STOP 开关拨到 RUN 位置运行程序。
- 只有使用 Set_COMM_PRMT 指令设定 RS232 自由口参数后，才可使用COMM_RECEIVE 指令进行数据接收，使用 COMM_SEND 指令进行数据发送。
- 只有使用 Set_COMM2_PRMT 指令设定 RS485 自由口参数后，才可使用COMM2_RECEIVE 指令进行数据接收，使用 COMM2_SEND 指令进行数据发送。
- 字型变量（%MW）地址必须定义在偶数地址，比如%MW200、%MW202、%MW204。
- 双字型变量（%MD）地址必须定义在偶数地址，比如%MD300、%MD304、%MD308。
- 在使用数学运算指令时，若输出数据定义的范围小于运算结果，则高位丢失。
- 部分指令由于硬件上冲突，所以不可以同时使用，请参考附录中的指令关联冲突速查表。
- 若指令的实例名声明在 RETAIN（掉电保持）区，该指令的所有输入/输出变量都要占用RETAIN内存区，所以建议不要在RETAIN内存区声明太多的实例，以免RETAIN区空间不足。
- 在梯形图（LD）编程环境下，插入使能运算符与插入功能块是两种不同的指令调用方式。其不同在于，如果采用插入使能运算符调用指令，当使能端低电平时，相应的指令代码不会被扫描，如果采用插入功能块调用指令，不论使能端低电平或高电平时，相应指令代码都会将使能端作为一个输入值来扫描。

第2章 操作数

在可编程控制器中，指令系统是可编程控制器硬件和软件的桥梁，是可编程控制器程序设计的基础。

与计算机的操作指令类似，可编程控制器指令的基本形式也是由操作码和操作数组成。操作码表示 CPU 所要执行的操作类型和所要完成的操作功能。操作数表示 CPU 所要操作的对象和目的。常量、变量、地址和函数调用返回值都可以作为操作数。

2.1 常量

✧ 布尔常量

布尔常量只有两个：逻辑值 TRUE 和 FALSE（也可表示为 1 和 0），TRUE 等价于 1，FALSE 等价于 0。

✧ 时钟常量

时钟常量一般用来操作时钟，由“T#”（或“t#”）加上“时钟值”构成，时钟值的单位包括天（d）、小时（h）、分（m）、秒（s）和毫秒（ms），它们的正确顺序为 d、h、m、s、ms，如“T#12h38m16s”表示“12 小时 38 分 16 秒”。

下面是**正确**的时钟常量：

```
T#18ms        (*18 毫秒的一个时钟常量*)
T#100s12ms    (*100 秒 12 毫秒的一个时钟常量，高单位允许超限*)
t#12h34m15s   (*12 小时 34 分 15 毫秒的一个常量*)
```

下面是**错误**的时钟常量：

```
t#5m68s       (*低单位不允许超限*)
15ms          (*没有 T# *)
t#4ms13d      (*顺序错误*)
```

✧ 日期常量

日期常量由“D#”(“d#”、“DATE#”或“date#”)加上“日期值”构成。例如：

```
DATE#2007-1-06    (*日期常量 2007 年 1 月 6 日*)
d#1980-09-22      (*日期常量 1980 年 9 月 22 日*)
```

✧ 时间常量

时间常量用于存储时间，由“TOD#”（“tod#”、“TIME_OF_DAY#”或“time_of_day#”）加上“时间值”构成。

时间值的格式为：小时:分钟:秒（可以用实数形式输入秒）。例如：

```
TOD#00:00:00                  (*时间常量为 0 点 0 时 0 分*)
TIME_OF_DAY#15:36:30.123      (*时间常量为 15 点 36 分 30.123 秒*)
```

✧ **日期时间常量**

日期常量和时间常量合并起来称为日期时间常量，由“DT#”（“dt#”、“DATE_AND_TIME#”或“date_and_time#”）加上“日期时间值”构成。例如：

DT#1980-09-22-15:45:18 (*日期时间常量为 1980 年 9 月 22 日 15 点 45 分 18 秒*)
date_and_time#2001-03-09-00:00:00 (*日期时间常量为 2001 年 3 月 9 日 0 点 0 分 0 秒*)

✧ **数字常量**

数字常量的数值可以是二进制、十进制、八进制和十六进制。如果整数值不是十进制值，可以用“进制”加符号“#”放在整数值前面来表示。十进制的 10 至 15 在十六进制中表示为 A 至 F。在数字中可以使用下划线连字符。

数字常量的数据类型可以是 BYTE、WORD、DWORD、SINT、USINT、INT、UINT、DINT、UDINT。

默认情况下，不允许“较大”的数据类型作为“较小”的数据类型使用。比如，DWORD 类型的常量不能简单地当作 INT 类型使用，必须使用数据类型转换指令进行转换之后才可以使用。例如：

14 (*十进制数 14*)
2#1001_0011 (*二进制数 1001_0011*)
8#67 (*八进制数 67*)
16#AE (*十六进制数 AE *)

✧ **实数常量**

实数常量用十进制小数和指数来表示，遵循标准的科学计数法格式。实数常量的数据类型是 REAL。例如：

7.4 (*实数 7.4*)
1.64e+009 (*实数 1.64e+009*)

✧ **字符串常量**

字符串常量在两个单引号之间，可以包含空格和特殊字符。例如：

' Abby and Craig ' (*字符串 Abby and Craig *)
':-)' (*字符串:-)*)

在字符串中，当在字符中出现以$开始的复合字符时，解释为下面的形式：

字符串	代表含义
$$	$字符
$’	单引号
$L 或$l	输入行
$N 或$n	新行
$P 或$p	输入页
$R 或$r	换行
$T 或$t	Tab 键

2.2 变量

变量在 POU（Program Organization Unit）的变量表或者全局变量表中声明，使用中应注意以下几点：

- 变量名不能包含空格和特殊字符，不能多次声明，不能和关键字使用相同的名字。
- 变量名不区分大小写。（如：VAR1、Var1 和 var1 表示相同的变量）
- 变量名识别下划线。（如：“A_BCD”和“AB_CD”被认为是两个不同的变量名）
- 变量名中不能有连续的 2 个下划线。（如：“A_ _B”是错误的变量名）

✧ **系统标志符**

系统标志符是隐含声明的变量，它在每个特定系统中是不同的。使用命令“插入/声明关键字”打开输入辅助对话框，选择系统变量类别，这样可以找到系统中可用的系统标志符。

✧ **变量访问的语法**

访问二维数组的元素：<字段名>[Index1, Index2]
访问结构变量：<结构名>.<变量名>
访问功能块和程序变量：<实例名>.<变量名>

✧ **访问变量的数据位**

在整型变量中，可以访问变量的每个数据位。
数据位附加在变量的后面，变量与数据位之间用“圆点”分隔，数据位从 0 开始编号。
可以访问变量数据位的数据类型包括：SINT、INT、DINT、USINT、UINT、UDINT、BYTE、WORD 和 DWORD。
定义在 VAR_IN_OUT 数据区的变量，不能访问变量的数据位。
例如：

```
a : INT;     (*定义整型变量 a*)
b : BOOL; (*定义布尔变量 b*)
...
a.2 := b;     (*将布尔变量 b 的值赋给整型变量 a 的第 2 位*)
```

出错处理：

- 如果数据位大于变量数据位的宽度（比如上例中若 a.16 := b），则会给出下列出错信息：

  Index '<n>' outside the valid range for variable '<var>'!
  因为整型变量的数据位的范围是 0－15，a.16 超出了范围。
- 如果变量类型不允许访问数据位（比如上例中若定义 a : REAL），则会给出下列出错信息：

  Invalid data type '<type>' for direct indexing
  因为实型变量不可以按位访问。

2.3 地址

✧ 地址格式

按照规定的地址格式显示内存中的地址。格式为：% 内存区范围 数据格式 地址。

内存区范围		数据格式	
I	输入区（Input）	X	单个位
Q	输出区（Output）	B	字节(8 位)
M	中间存储区（Memory location）	W	字（16 位）
		D	双字（32 位）

例如：

地址格式	对应地址
%QX7.5	输出区的地址 7，第 5 位
%IW4	输入区的地址 4，1 个字
%QB7	输出区的地址 7，1 个字节
%MD48	中间存储区的地址 48，双字

✧ 内存位置

在 PowerPro 中，内存地址按照字节排列，从 0 开始，其大小与 PLC 型号有关。例如 M 区（中间存储区）地址定义如表 2-1-1。

表 2-1-1

地址	字节定义	字定义	双字定义
0	%MB0	%MW0	%MD0
1	%MB1		
2	%MB2	%MW2	
3	%MB3		
4	%MB4	%MW4	%MD4
5	%MB5		
6	%MB6	%MW6	
7	%MB7		
……	……	……	……
4n	%MB4n	%MW4n	%MD4n
4n+1	%MB4n+1		
4n+2	%MB4n+2	%MW4n+2	
4n+3	%MB4n+3		

提示：

- 字型变量（%MW）必须定义在偶数地址，比如%MW0、%MW2、%MW4、%MW6……%MW4n。每个字型变量占用 2 个字节型变量地址。
- 双字型变量（%MD）必须定义在偶数地址，比如%MD0、%MD4……%MD4n。每个双字型变量占用 4 个字节型变量地址或者 2 个字型变量地址。

✧ 数据存储格式

PowerPro 软件中数据存储格式以 M 区为例，其中 MSB 表示最高有效位，LSB 最低有效位， 如下面所示：

- 字节

	MSB 7	LSB 0
%MB300	% MB300	

- 字

	MSB 15		LSB 0
%MW300	%MB301	%MB300	

- 双字

	MSB 31			LSB 0
%MD300	%MB303	%MB302	%MB301	%MB300

2.4 函数返回值

在 ST 语言中，调用函数的返回值可以直接作为操作数使用。例如：

Result:= Fct(7) + 3; (*函数 Fct 的返回值加上 3，然后赋值给 Result *)。

第3章 数据类型

编程时可以使用标准数据类型和用户自定义数据类型。数据类型用标识符来表示，规定了数据占用内存空间的大小及存储于其中的数据种类。

标准数据类型包括布尔型数据、整型数据、实型数据、字符串型数据和时间型数据，可以使用 PowerPro 提供的数据类型转换指令进行相互转化。

用户自定义数据类型包括数组、指针、枚举和结构。

3.1 标准数据类型

✧ 布尔型数据类型

布尔型变量的标识符为 BOOL，其值为“TRUE”和“FALSE”（也可表示为“1”和“0”）。“TRUE”等价于“1”，“FALSE”等价于“0”。

✧ 整型数据类型

整型数据类型的标识符包括 BYTE、WORD、DWORD、SINT、USINT、INT、UINT、DINT 和 UDINT 等。每一种不同数据类型的取值范围不同，整型数据类型的取值范围如表 3-1-1。

表 3-1-1

类型标识符	类型名称	数据下限	数据上限	存储空间
BYTE	字节型	0	255	8 Bit
WORD	字型	0	65535	16 Bit
DWORD	双字型	0	4294967295	32Bit
SINT	短整型	-128	127	8 Bit
USINT	无符号短整型	0	255	8 Bit
INT	整型	-32768	32767	16 Bit
UINT	无符号整型	0	65535	16 Bit
DINT	双整型	-2147483648	2147483647	32 Bit
UDINT	无符号双整型	0	4294967295	32 Bit

提示：
当较长的数据类型转换为较短的数据类型时，会丢失高位信息。

✧ 实型数据类型

实型数据类型也称为浮点型数据类型，用于表示有理数。实型数据类型的标识符为 REAL。REAL 型数据占用 32 位内存空间，即 4 个字节。

✧ **字符串型数据类型**

字符串型数据可以包含任意多个字符，其标识符为 STRING。在声明时所输入的大小决定了存储变量所需要的内存空间，它是指字符串中字符的数量，可以用圆括号或方括号括起来。如果没有给出大小说明，则缺省大小为 80 个字符。

例如：

str:STRING(35):='This is a String';　(*声明一个包含 35 个字符的字符串*)

✧ **时间型数据类型**

时间型数据类型用于处理时间，其标识符包括 TIME（缩写为 T）、TIME_OF_DAY（缩写为 TOD）、DATE（缩写为 D）和 DATE_AND_TIME（缩写为 DT）。

TIME 表示一个时间值，单位为毫秒，初始值为 0。

TOD 表示当天的时间，单位为毫秒，初始值为凌晨 0 点 0 分。

DATE 表示当前日期，单位为秒。初始值是 1970 年 1 月 1 日。

DT 表示当前日期和时间，单位为秒。初始值是 1970 年 1 月 1 日凌晨 0 点 0 分。

3.2 自定义数据类型

3.2.1 数组

一维、二维和三维数组属于基本的数据类型。在 POU 的变量表或者全局变量表中，都可以声明数组。数组的标识符为 ARRAY。

✧ **声明数组的语法**

<数组名>:ARRAY [<L1>..<U1>,<L2>..<U2>,<L3>..<U3>] OF <基本数据类型>;

L1、L2 和 L3 表示字段范围的最小值，U1、U2 和 U3 表示字段范围的最大值。字段范围必须是整数。例如：

Card_game: ARRAY [1..13, 1..4] OF INT;　(*定义一个整型的二维数组 Card_game*)

✧ **数组的初始化**

在数组定义时，可以初始化数组中所有元素，也可以不进行初始化。

举例 1：数组的完全初始化

```
Arr1:ARRAY [1..5] OF BYTE:= 1,2,3,4,5;
Arr2:ARRAY [1..2,3..4] OF INT := 1,3(7) ;                （*即 1,7,7,7 的缩写形式*）
Arr3:ARRAY [1..2,2..3,3..4] OF INT := 2(0),4(4),2,3;   （*即 0,0,4,4,4,4,2,3 的缩写形式*）
```

举例 2：结构数组的初始化

```
TYPE STRUCT1:
STRUCT
p1:int;
p2:int;
p3:dword;
END_STRUCT
```

```
END_ TYPE
ARRAY[1..3] OF STRUCT1:=(p1:=1,p2:=10,p3:= 3),(p1:=2,p2:=0,p3:=2),
                                    (p1:=4,p2:=5,p3:=1);
```

举例 3：数组的部分初始化

```
Arr1:ARRAY [1..10] OF BYTE:= 1,2;
```

对于那些没有预先赋值的元素，按照基本数据类型的缺省初始值进行初始化。在此例中，元素[3]到[10]被初始化为 0。

✧ **数组的访问**

在二维数组中访问数组元素，使用下列语法：

```
<Field_Name>[Index1,Index2]
```

例如：

数组的访问：

```
Card_game[9,2]
```

✧ **注意**

- 如果在工程中使用 CheckBounds 来定义函数，则可以自动进行数组越界错误检查。该函数名的关键字必须是 CheckBounds，CheckBounds 的程序如图 3-2-1 所示。

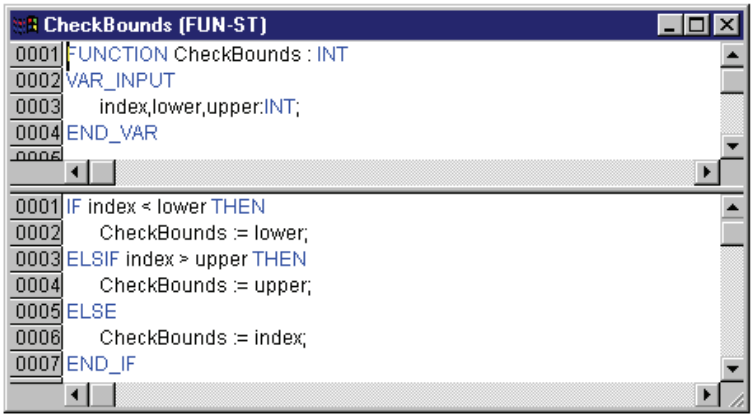


图 3-2-1

- 若工程中没有上面所述的 CheckBounds 程序，在图 3-2-2 的例子中，A[B]应该为 A[10]，从而超出了数组 A 的最大上界值 7，程序编译时出错。
- 但因为工程中定义了上面的 CheckBounds 函数，所以执行图 3-2-2 的程序时，A[B]会被当作 A[7]来使用，将布尔量 TRUE 赋值给 A[7]，编译时不出错。

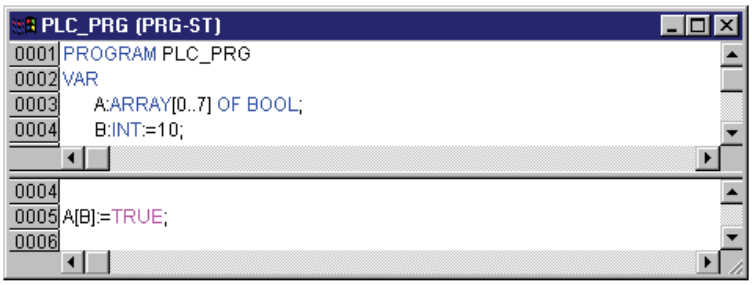


图 3-2-2

3.2.2 指针

当程序运行的时候，通过指针可以取得变量或功能块实例的地址。指针可以指向任何数据类型或功能块类型，包括用户自定义的数据类型。声明指针的语法：

<指针名>：POINTER TO <数据类型/功能块类型>。

通过取地址指令 ADR 可以将变量或指令实例的地址取出。通过在指针标识符的后面增加取地址内容指令“^”可以取得指针所指地址的数据。

例如：

程 序	含 义
pt:POINTER TO INT; Var_int1:INT := 5; Var_int2:INT; pt := ADR(Var_int1); Var_int2:= pt^;	(*定义一个整型数据的指针 pt*) (*定义整型变量 Var_int1，使其等于 5*) (*定义整型变量 Var_int2*) (*取出 Var_int1 变量的地址，将地址值赋给 pt*) (*将指针 pt 所指地址的值赋给 Var_int2，Var_int2=5*)

3.2.3 枚举

枚举是由一长串的数字常量组成的自定义数字类型，这些常量称为枚举值。

只要枚举值声明在 POU 内，在整个程序范围内都可以被识别。建议将枚举创建在 PowerPro 程序左下角选项卡中的 数据类型（Data types）里，通过在数据类型（Data types）上添加对象（Add Object）来创建一个新的枚举值。

枚举以关键字 TYPE 开始，以关键字 END_TYPE 结束。声明枚举的语法：

TYPE<标识符>:(<Enum_0>,<Enum_1>,…,<Enum_n>);
END_TYPE

✧ **注意**

- 枚举值中可以包含标识符变量，初始化时，该标识符变量被初始化为该枚举中的一个值。可以把任何数字量分配给枚举值。如果枚举值没有被初始化，则从 0 开始递增进行初始化。定义的枚举值与其他标准数据类型兼容，就像使用 INT 数据类型一样对其进行操作。

举例：

程 序	含 义

TYPE TRAFFIC_SIGNAL: (Red, Yellow, Green:=10);
END_TYPE
 	
TRAFFIC_SIGNAL1: TRAFFIC_SIGNAL;
TRAFFIC_SIGNAL1:=0;
FOR i:= Red TO Green DO
i := i + 1;
END_FOR
	(*每个颜色的初始值是 Red=0，Yellow=1，Green=10*)
	
	(*交通信号的值是 Red*)

- 相同的枚举值不能使用两次。

举例：

TRAFFIC_SIGNAL: (red, yellow, green);
COLOR: (blue, white, red);
错误：red 不能同时被 TRAFFIC_SIGNAL 和 COLOR 使用。

3.2.4 结构

结构创建在 PowerPro 软件左下角数据类型选项卡窗口中，通过在数据类型上添加对象来创建一个新的结构。

结构以关键字 TYPE 和 STRUCT 开始，以关键字 END_STRUCT 和 END_TYPE 结束。

✧ **声明结构的语法**

```
TYPE <结构名>:
STRUCT
<变量声明 1>
  :
<变量声明 n>
END_STRUCT
END_TYPE
```

<结构名>是一种可以在整个工程中被识别的数据类型，可以像标准数据类型一样引用，但不可以指定结构中变量的地址（即变量名之后不允许使用 AT 来指定该变量的地址）。

举例（定义名为 Polygonline 的结构）：

```
TYPE Polygonline:
STRUCT
   Start:ARRAY [1..2] OF INT;
   Point1:ARRAY [1..2] OF INT;
   Point2:ARRAY [1..2] OF INT;
   Point3:ARRAY [1..2] OF INT;
   Point4:ARRAY [1..2] OF INT;
End:ARRAY [1..2] OF INT;
END_STRUCT
END_TYPE
```

举例（初始化结构）：

```
Poly_1:polygonline:= (Start:=3,3,Point1 =5,2,Point2:=7,3,Point3:=8,5,
                      Point4:=5,7,End :=3,5);
```

✧ **结构成员的访问**

<结构名>.<结构成员名>

举例：

如果结构名为“Week”，其中一个成员名为“Monday”，则可以用下面的形式访问：

Week.Monday

第4章 基本指令

4.1 算术运算指令

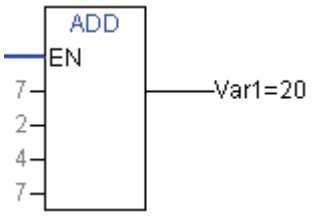
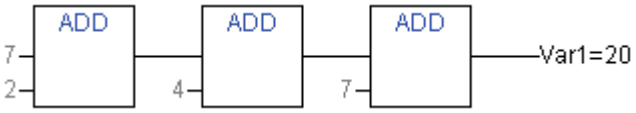
4.1.1 ADD——加法指令

- **功能**：两个（或者多个）变量或常量相加。
- **输入/输出数据类型**：BYTE、WORD、DWORD、SINT、USINT、INT、UINT、DINT、UDINT、REAL、TIME。

✧ **指令使用举例**

变量定义

	VAR	VAR_INPUT	VAR_OUTPUT	VAR_IN_OUT	CON
	名称	地址	类型	初始值	注释
0001	Var1		INT		

编程语言	程 序
梯形图（LD）	
结构化文本（ST）	Var1:=7+2+4+7; （*结果 Var1 为 20*）
指令列表（IL）	LD 7 ADD 2,4,7 ST Var1 （*结果 Var1 为 20*）
功能块（FBD）	

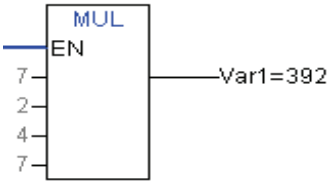
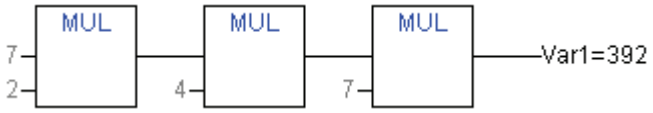
提示：

- TIME 型量也可以使用加法功能，两个 TIME 型量相加得到一个新的时间量。例如：t#45s + t#50s = t#1m35 s。
- 被选择的输出数据类型应可以存储输出结果，否则可能引起数据错误。MUL、SUB、DIV 指令同样。

4.1.2 MUL——乘法指令

- **功能**：两个（或者多个）变量或常量相乘。
- **输入/输出数据类型**：BYTE、WORD、DWORD、SINT、USINT、INT、UINT、DINT、UDINT、REAL。

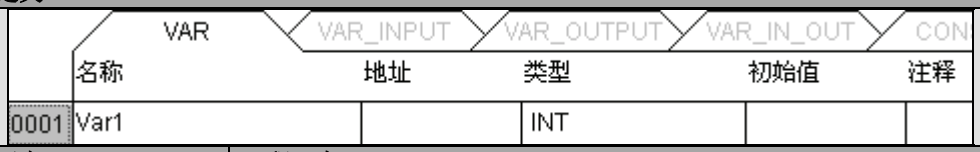
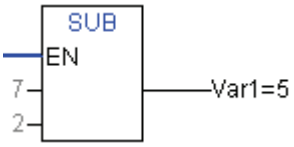
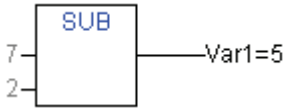
✧ 指令使用举例

变量定义	
编程语言	**程　序**
梯形图（LD）	
结构化文本（ST）	Var1:=7*2*4*7;　（*结果 Var1 为 392*）
指令列表（IL）	LD　7 MUL　2,4,7 ST　Var1　（*结果 Var1 为 392*）
功能块（FBD）	

4.1.3 SUB——减法指令

- **功能：**两个变量或常量相减。
- **输入/输出数据类型：**BYTE、WORD、DWORD、SINT、USINT、INT、UINT、DINT、UDINT、REAL、TOD。

✧ 指令使用举例

变量定义	
	
编程语言	**程　序**
梯形图（LD）	
结构化文本（ST）	Var1:=7-2;　（*结果 Var1 为 5*）
指令列表（IL）	LD　7 SUB　2 ST　Var1　（*结果 Var1 为 5*）
功能块（FBD）	

提示：

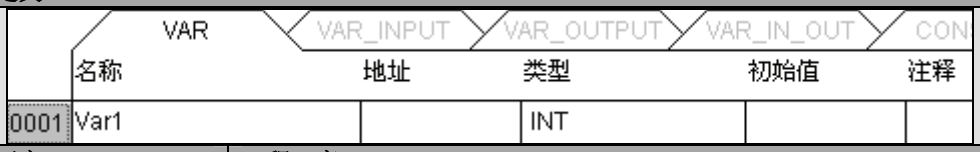
- TIME 型量也可以使用减法功能，两个 TIME 型量相减得到一个新的时间量。例如：t#1m35s - t#50s = t#45s ，但时间结果不能有负值。
- TOD 型量也可以使用减法功能，两个 TOD 型量相减得到一个新的 TIME 型数据，例如：TOD#23:40:30- TOD#00:30:20=T#1390m10s0ms ,但时间结果不能有负值。

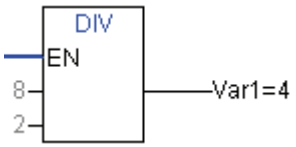
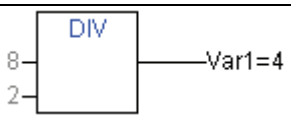
4.1.4 DIV——除法指令

- 功能：变量或常量相除。
- 输入/输出数据类型：BYTE、WORD、DWORD、SINT、USINT、INT、UINT、DINT、UDINT、REAL。

✧ **指令使用举例**

变量定义



编程语言	程　序
梯形图（LD）	
结构化文本（ST）	Var1:=8/2;　　（*结果 Var1 为 4*）
指令列表（IL）	LD　8 DIV　2 ST　Var1　　（*结果 Var1 为 4*）
功能块（FBD）	

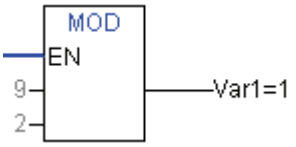
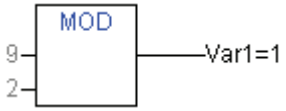
提示：

- 在工程中使用 DIV 指令时，可使用 CheckDivByte、CheckDivWord、CheckDivDWord 和 CheckDivReal 等指令（见 4.15 节）检查除数是否为零，避免了除数为零的现象。

4.1.5 MOD——取余指令

- **功能：**变量或常量相除取余，是一个整数。
- **输入/输出数据类型：**BYTE、WORD、DWORD、SINT、USINT、INT、UINT、DINT、UDINT。

✧ 指令使用举例

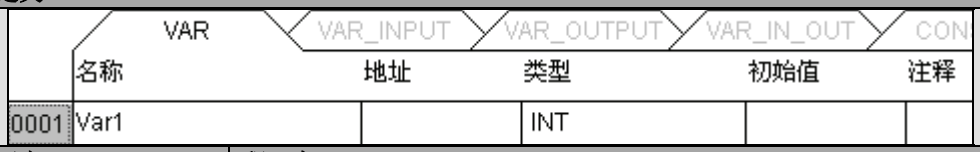
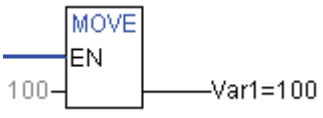
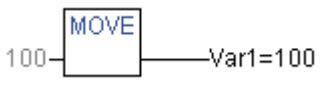
变量定义	
编程语言	**程　序**
梯形图（LD）	
结构化文本（ST）	Var1:=9 MOD 2;　（*结果 Var1 为 1*）
指令列表（IL）	LD　9 MOD　2 ST　Var1　（*结果 Var1 为 1*）
功能块（FBD）	

4.2 赋值指令

MOVE—赋值指令

- **功能：**将一个常量或者变量的值赋给另外一个变量。
- **输入/输出数据类型：** BYTE、WORD、DWORD、SINT、USINT、INT、UINT、DINT、UDINT、REAL、TIME、DT、BOOL、STRING、ARRAY。

✧ 指令使用举例

变量定义	
	
编程语言	**程　序**
梯形图（LD）	
结构化文本（ST）	Var1:=100;　（*结果 Var1 为 100*）
指令列表（IL）	LD　100 MOVE ST　Var1　（*结果 Var1 为 100*）
功能块（FBD）	

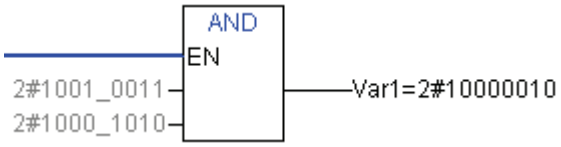
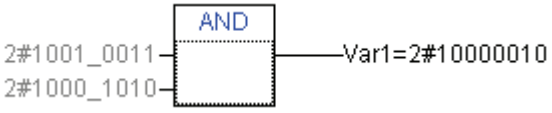
4.3 逻辑运算指令

4.3.1 AND——与指令

- **功能：**变量或常量的相与运算。
- **输入/输出数据类型：**BOOL、BYTE、WORD 和 DWORD。

✧ **指令使用举例**

变量定义

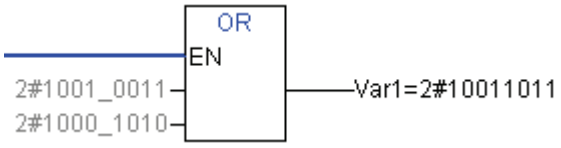
编程语言	程　序
梯形图（LD）	
结构化文本（ST）	Var1:=2#1001_0011 AND 2#1000_1010; （*结果 Var1 为 2#10000010*）
指令列表（IL）	LD　2#1001_0011 AND　2#1000_1010 ST　Var1　（*结果 Var1 为 2#10000010*）
功能块（FBD）	

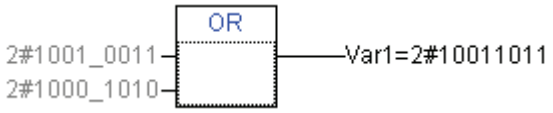
4.3.2 OR——或指令

- **功能：**变量或常量的相或运算。
- **输入/输出数据类型：**BOOL、BYTE、WORD 和 DWORD。

✧ **指令使用举例**

变量定义

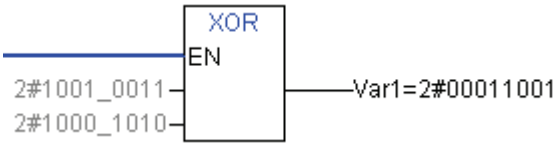
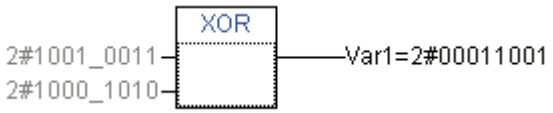
编程语言	程　序
梯形图（LD）	
结构化文本（ST）	Var1:=2#1001_0011 OR 2#1000_1010 ; （*结果 Var1 为 2#10011011*）
指令列表（IL）	LD　2#1001_0011

	OR 2#1000_1010 ST Var1 （*结果 Var1 为 2#10011011*）
功能块（FBD）	

4.3.3 XOR——异或指令

- **功能：**变量或常量的异或运算。
- **输入/输出数据类型：**BOOL、BYTE、WORD 和 DWORD。

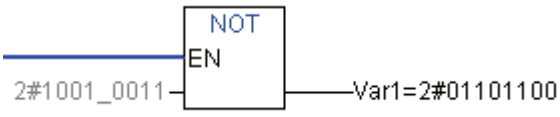
✧ **指令使用举例**

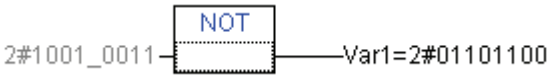
变量定义	
编程语言	**程 序**
梯形图（LD）	
结构化文本（ST）	Var1:=2#1001_0011 XOR 2#1000_1010 ; （*结果 Var1 为 2#00011001*）
指令列表（IL）	LD 2#1001_0011 XOR 2#1000_1010 ST Var1 （*结果 Var1 为 2#00011001*）
功能块（FBD）	

4.3.4 NOT——取非指令

- **功能：**变量或常量的取非运算， 逐位取非。
- **输入/输出数据类型：**BOOL、BYTE、WORD 和 DWORD。

✧ **指令使用举例**

变量定义	
编程语言	**程 序**
梯形图（LD）	

结构化文本（ST）	Var1:= NOT 2#1001_0011 ; （*结果 Var1 为 2#01101100*）
指令列表（IL）	LD 2#1001_0011 NOT ST Var1 （*结果 Var1 为 2#01101100*）
功能块（FBD）	

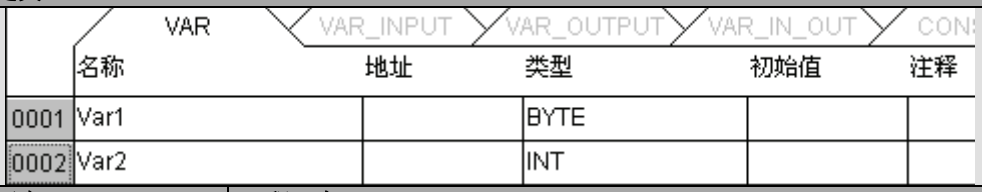
4.4 移位指令

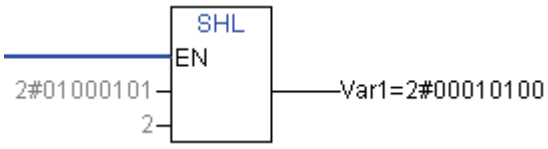
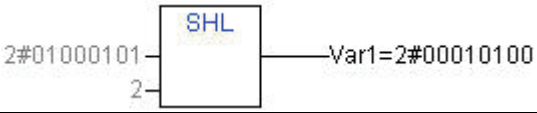
4.4.1 SHL——左移指令

- **功能：**对操作数进行按位左移，左边移出的位不作处理，右边自动补 0。
- **输入/输出数据类型：**BYTE、INT、WORD、DWORD、SINT、UINT。

✧ **指令使用举例**

变量定义



编程语言	**程 序**
梯形图（LD）	
结构化文本（ST）	Var1:=SHL(16#45,2); （*结果 Var1 为 16#14*） Var2:=SHL(16#45,2); （*结果 Var2 为 16#0114*） 注意：上面例子中，虽然输入变量的值一样，但因为输出数据类型的大小不同，所以得到的结果 Var1 和 Var2 不同。
指令列表（IL）	LD 16#45 SHL 2 ST Var1 （*结果 Var1 为 16#14*）
功能块（FBD）	

4.4.2 SHR——右移指令

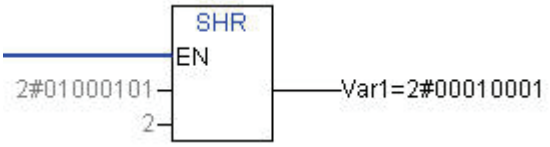
- **功能：**对操作数进行按位右移，右边移出的位不作处理，左边自动补 0。
- **输入/输出数据类型：**BYTE、INT、WORD、DWORD、SINT、UINT。

✧ **指令使用举例**

变量定义	

VAR | VAR_INPUT | VAR_OUTPUT | VAR_IN_OUT | CON

	名称	地址	类型	初始值	注释
0001	Var1		BYTE		
0002	Var2		INT		

编程语言	程　序
梯形图（LD）	
结构化文本（ST）	Var1:=SHR(16#45,2);　（*结果 Var1 为 16#11*） Var2:=SHR(16#45,2);　（*结果 Var2 为 16#0011*）
指令列表（IL）	LD　16#45 SHR　2 ST　Var1　（*结果 Var1 为 16#11*）
功能块（FBD）	

4.4.3　ROL——循环左移指令

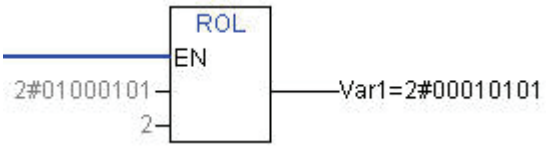
- **功能：**对操作数进行按位循环左移，左边移出的位直接补充到右边最低位。
- **输入/输出数据类型：**BYTE、INT、WORD、DWORD、SINT、UINT。

✧ **指令使用举例**

变量定义	

VAR | VAR_INPUT | VAR_OUTPUT | VAR_IN_OUT | CON

	名称	地址	类型	初始值	注释
0001	Var1		BYTE		
0002	Var2		INT		

编程语言	程　序
梯形图（LD）	
结构化文本（ST）	Var1:=ROL(16#45,2);　（*结果 Var1 为 16#15*） Var2:= ROL (16#45,2);　（*结果 Var2 为 16#0114*） 注意：在循环左移过程中，虽然输入变量的值一样，但因为输出数据类型的大小不同，所以得到的结果 Var1 和 Var2 不同。
指令列表（IL）	LD　16#45 ROL　2 ST　Var1　（*结果 Var1 为 16#15*）
功能块（FBD）	

4.4.4 ROR——循环右移指令

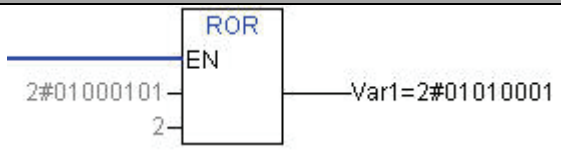
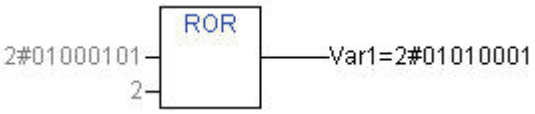
- **功能：**对操作数进行按位循环右移，右边移出的位直接补充到左边最高位。
- **输入/输出数据类型：**BYTE、INT、WORD、DWORD、SINT、UINT。

✧ **指令使用举例**

变量定义

VAR | VAR_INPUT | VAR_OUTPUT | VAR_IN_OUT | CON

	名称	地址	类型	初始值	注释
0001	Var1		BYTE		
0002	Var2		INT		

编程语言	程　序
梯形图（LD）	
结构化文本（ST）	Var1:=ROR(16#45,2)　（*结果 Var1 为 16#51*） Var2:= ROR (16#45,2)　（*结果 Var2 为 16#4011*） **注意：**在循环右移过程中，虽然输入变量的值一样，但因为输出数据类型的大小不同，所以得到的结果 Var1 和 Var2 不同。
指令列表（IL）	LD　16#45 ROR　2 ST　Var1　（*结果 Var1 为 16#51*）
功能块（FBD）	

4.5 选择指令

所有的选择指令在执行时均可以带有变量。为了能够更加清楚地说明问题，以下各例只使用常量。被选择的输入数据类型存储长度应不大于输出类型存储长度。

4.5.1 SEL——二选一指令

- **功能：**通过选择开关在两个输入数据中选择一个作为输出，选择开关为 FALSE 时输出为第一个输入数据，选择开关为 TRUE 时输出为第二个输入数据。
- **指令格式：**OUT := SEL(G, IN0, IN1)，其中 G 为选择开关，IN0 和 IN1 分别为第一个输入数据和第二个输入数据。
- **输入/输出数据类型：**

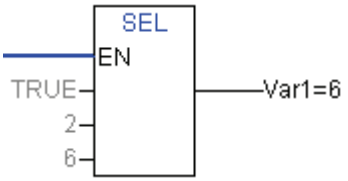
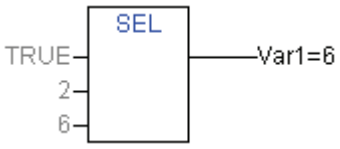
G 必须是 BOOL 类型，IN0、IN1 和输出数据可以是任意数据类型。

✧ 指令使用举例

变量定义

VAR | VAR_INPUT | VAR_OUTPUT | VAR_IN_OUT | CON

	名称	地址	类型	初始值	注释
0001	Var1		INT		
0002	Var2		INT		

编程语言	程　序
梯形图（LD）	
结构化文本（ST）	Var1:=SEL(TRUE,3,4); (*结果 Var1 为 4 *)
指令列表（IL）	LD TRUE SEL 2,6 ST Var1 (*结果 Var1 为 6*) LD FALSE SEL 2,6 ST Var2 (*结果 Var2 为 2*)
功能块（FBD）	

4.5.2 MAX——取最大值指令

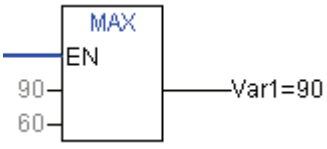
- **功能：**在多个输入数据中选择最大值作为输出。
- **指令格式：**OUT:=MAX(IN0, IN1)，其中 IN0 和 IN1 分别为第 1 个输入数据和第 2 个输入数据，OUT 是输出数据。
- **输入/输出数据类型：**IN0, IN1 和 OUT 可以是任意数据类型。

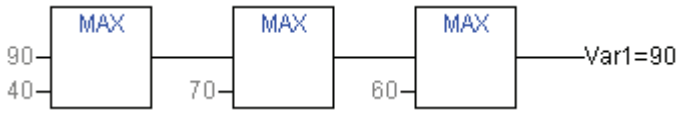
✧ 指令使用举例

变量定义

VAR | VAR_INPUT | VAR_OUTPUT | VAR_IN_OUT | CON

	名称	地址	类型	初始值	注释
0001	Var1		INT		
0002	Var2		INT		

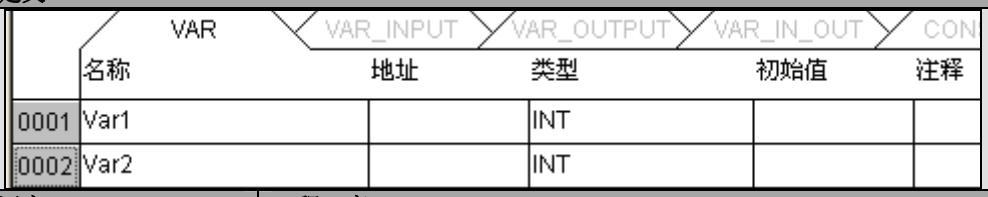
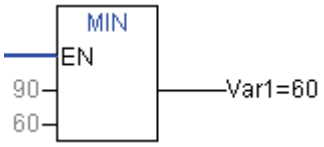
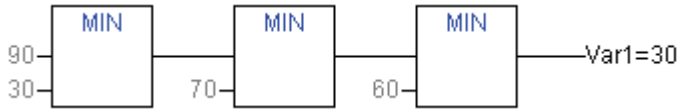
编程语言	程　序
梯形图（LD）	
结构化文本（ST）	Var1:=MAX(90,60); (*结果 Var1 为 90 *) Var2:=MAX(40,MAX(90,60)); (*结果 Var2 为 90 *)

指令列表（IL）	LD 90 MAX 40 MAX 70 MAX 60 ST Var1 （*结果 Var1 为 90*）
功能块（FBD）	

4.5.3 MIN——取最小值指令

- **功能：**在多个输入数据中选择最小值作为输出。
- **指令格式：**OUT:=MIN(IN0, IN1)，其中 IN0 和 IN1 分别为第 1 个输入数据和第 2 个输入数据，OUT 是输出数据。
- **输入/输出数据类型：**IN0， IN1 和 OUT 可以是任意数据类型。

✧ **指令使用举例**

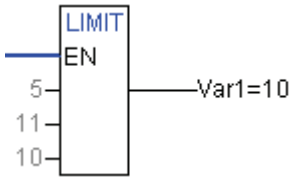
变量定义	
	
编程语言	**程 序**
梯形图（LD）	
结构化文本（ST）	Var1:=MIN(90,30); (*结果 Var1 为 30 *) Var2:=MIN(MIN(90,30),60); (*结果 Var2 为 30 *)
指令列表（IL）	LD 90 MIN 30 MIN 40 MIN 70 ST Var1 （*结果 Var1 为 30*）
功能块（FBD）	

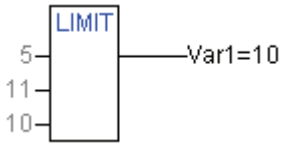
4.5.4 LIMIT——极限值指令

- **功能：**判断输入数据是否在最小值和最大值之间，若输入数据在二者之间，则直接把输入数据作为输出数据进行输出。若输入数据大于最大值，则把最大值作为输出值。若输入数据小于最小值，则把最小值作为输出值。
- **指令格式：** OUT := LIMIT(Min, IN, Max)

- **输入/输出数据类型：**IN 和 OUT 可以是任意数据类型。

✧ **指令使用举例**

<table>
<tr><td colspan="2">变量定义
</td></tr>
<tr><td>编程语言</td><td>程　序</td></tr>
<tr><td>梯形图（LD）</td><td> 

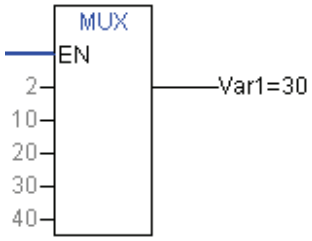
（*11 为输入数据，5 为最小值，10 为最大值*）</td></tr>
<tr><td>结构化文本（ST）</td><td>Var1:=LIMIT(30,90,80);　(*结果 Var1 为 80 *)</td></tr>
<tr><td>指令列表（IL）</td><td>LD　90
LIMIT　30，80
ST　Var1　(*结果 Var1 为 80*)</td></tr>
<tr><td>功能块（FBD）</td><td> 

（*11 为输入数据，5 为最小值，10 为最大值*）</td></tr>
</table>

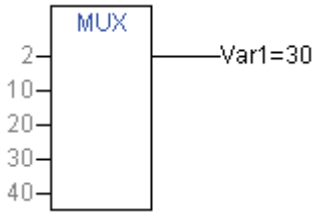
4.5.5　MUX——多选一指令

- **功能：**通过控制数在多个输入数据中选择一个作为输出。
- **指令格式：** OUT:=MUX(K,IN0,…,INn)，其中 K 为控制数，IN0,…,INn 为输入数据，OUT 为输出结果。控制数为 K 时选择第 INk 个输入数据作为输出。
- **输入/输出数据类型：**IN0,…, INn 和 OUT 可以是任意数据类型，K 必须是 BYTE、WORD、DWORD、SINT、USINT、INT、UINT、DINT 或 UDINT。

✧ **指令使用举例**

<table>
<tr><td colspan="2">变量定义
</td></tr>
<tr><td>编程语言</td><td>程　序</td></tr>
<tr><td>梯形图（LD）</td><td> 

（*2 为控制数据，对应于 30，所以输出 30 *）</td></tr>
</table>

结构化文本（ST）	Var1:=MUX(0,30,40,50,60,70,80);（*结果 Var1 为 30*）
指令列表（IL）	LD 0 MUX 30，40，50，60，70，80 ST Var1 （*结果 Var1 为 30*）
功能块（FBD）	 （*2 为控制数据，对应于 30，所以结果为 30*）

4.6 比较指令

所有的比较指令在执行时均可以带有变量。为了能够更加清楚地说明问题，以下各例只使用常量。

4.6.1 GT——大于指令

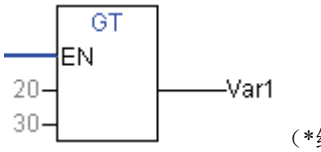
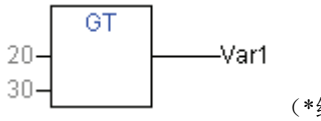
- **功能：**判断两个操作数的大小，当第一个数大于第二个数时输出 TRUE，否则输出为 FALSE。
- **输入/输出数据类型：**

输入数据类型：BOOL、BYTE、WORD、DWORD、SINT、USINT、INT、UINT、DINT、UDINT、REAL、TIME、DATE、TOD、DT 和 STRING；

输出数据类型：BOOL。

✧ **指令使用举例**

变量定义					
VAR / VAR_INPUT / VAR_OUTPUT / VAR_IN_OUT / CON					
	名称	地址	类型	初始值	注释
0001	Var1		BOOL		

编程语言	程 序
梯形图（LD）	 （*结果 Var1 为 FALSE*）
结构化文本（ST）	Var1:=20>30;
指令列表（IL）	LD 20 GT 30 ST Var1 （*结果 Var1 为 FALSE*）
功能块（FBD）	 （*结果 Var1 为 FALSE*）

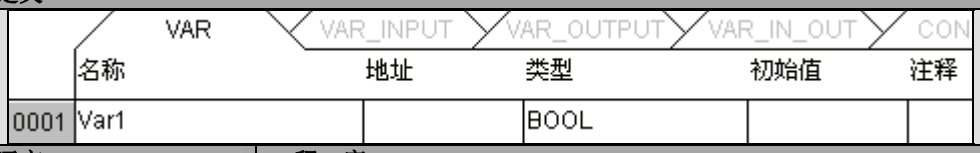
4.6.2 LT——小于指令

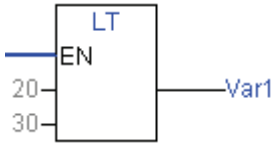
- **功能：**判断两个操作数的大小，当第一个数小于第二个数时返回 TRUE，否则结果为 FALSE。
- **输入/输出数据类型：**

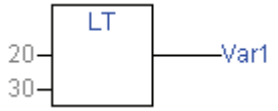
输入数据类型：BOOL、BYTE、WORD、DWORD、SINT、USINT、INT、UINT、DINT、UDINT、REAL、TIME、DATE、TOD、DT 和 STRING；

输出数据类型：BOOL。

✧ **指令使用举例**

<table>
<tr><td colspan="2">变量定义</td></tr>
<tr><td></td><td>


</td></tr>
<tr><td>编程语言</td><td>程　序</td></tr>
<tr><td>梯形图（LD）</td><td>


（*结果 Var1 为 TRUE*）
</td></tr>
<tr><td>结构化文本（ST）</td><td>VAR1:=20<30;　　（*结果 Var1 为 TRUE*）</td></tr>
<tr><td>指令列表（IL）</td><td>LD　20
LT　30
ST　Var1　　（*结果 Var1 为 TRUE*）</td></tr>
<tr><td>功能块（FBD）</td><td>


（*结果 Var1 为 TRUE*）
</td></tr>
</table>

4.6.3 GE——大于等于指令

- **功能：**判断两个操作数的大小，当第一个数大于等于第二个数时返回 TRUE，否则结果为 FALSE。
- **输入/输出数据类型：**

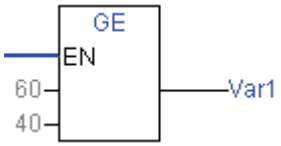
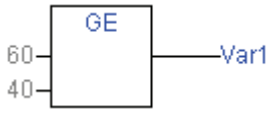
输入数据类型：BOOL、BYTE、WORD、DWORD、SINT、USINT、INT、UINT、DINT、UDINT、REAL、TIME、DATE、TOD、DT 和 STRING；

输出数据类型：BOOL。

✧ **指令使用举例**

<table>
<tr><td>变量定义</td></tr>
<tr><td>

</td></tr>
</table>

编程语言	程　序
梯形图（LD）	 （*结果 Var1 为 TRUE*）
结构化文本（ST）	VAR1:=60>=40;　　（*结果 Var1 为 TRUE*）
指令列表（IL）	LD　60 GE　40 ST　Var1　　（*结果 Var1 为 TRUE*）
功能块（FBD）	 （*结果 Var1 为 TRUE*）

4.6.4　LE——小于等于指令

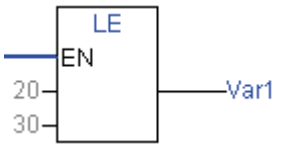
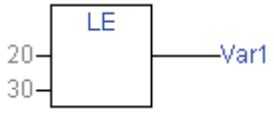
- **功能：**判断两个操作数的大小，当第一个数小于等于第二个数时返回 TRUE，否则结果为 FALSE。
- **输入/输出数据类型：**

输入数据类型：BOOL、BYTE、WORD、DWORD、SINT、USINT、INT、UINT、DINT、UDINT、REAL、TIME、DATE、TOD、DT 和 STRING；

输出数据类型：BOOL。

✧ **指令使用举例**

变量定义					
	VAR / VAR_INPUT / VAR_OUTPUT / VAR_IN_OUT / CON				
	名称	地址	类型	初始值	注释
0001	Var1		BOOL		

编程语言	程　序
梯形图（LD）	 （*结果 Var1 为 TRUE*）
结构化文本（ST）	VAR1:=20<=30;　　（*结果 Var1 为 TRUE*）
指令列表（IL）	LD　20 LE　30 ST　Var1　　（*结果 Var1 为 TRUE*）
功能块（FBD）	 （*结果 Var1 为 TRUE*）

4.6.5 EQ——等于指令

- **功能：**判断两个操作数是否相等，当第一个数等于第二个数时返回 TRUE，否则结果为 FALSE。
- **输入/输出数据类型：**

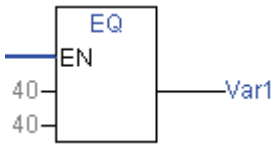
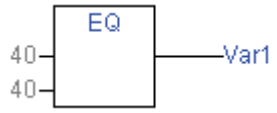
输入数据类型：BOOL、BYTE、WORD、DWORD、SINT、USINT、INT、UINT、DINT、UDINT、REAL、TIME、DATE、TOD、DT 和 STRING；

输出数据类型：BOOL。

✧ **指令使用举例**

变量定义

	VAR	VAR_INPUT	VAR_OUTPUT	VAR_IN_OUT	CON
	名称	地址	类型	初始值	注释
0001	Var1		BOOL		

编程语言	程　序
梯形图（LD）	 （*结果 Var1 为 TRUE*）
结构化文本（ST）	VAR1:=40=40;　　（*结果 Var1 为 TRUE*）
指令列表（IL）	LD　40 EQ　40 ST　Var1　　（*结果 Var1 为 TRUE*）
功能块（FBD）	 （*结果 Var1 为 TRUE*）

4.6.6 NE——不等于指令

- **功能：**判断两个操作数是否不相等，当第一个数不等于第二个数时返回 TRUE，否则结果为 FALSE。
- **输入/输出数据类型：**

输入数据类型：BOOL、BYTE、WORD、DWORD、SINT、USINT、INT、UINT、DINT、UDINT、REAL、TIME、DATE、TOD、DT 和 STRING；

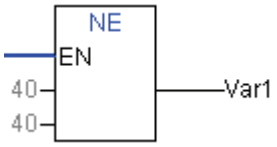
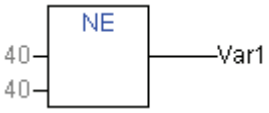
输出数据类型：BOOL。

✧ 指令使用举例

变量定义

VAR | VAR_INPUT | VAR_OUTPUT | VAR_IN_OUT | CON

	名称	地址	类型	初始值	注释
0001	Var1		BOOL		

编程语言	程　序
梯形图（LD）	 （*结果 Var1 为 FALSE*）
结构化文本（ST）	VAR1:=40<>40;　　（*结果 Var1 为 FALSE*）
指令列表（IL）	LD　40 NE　40 ST　Var1　　（*结果 Var1 为 FALSE*）
功能块（FBD）	 （*结果 Var1 为 FALSE*）

4.7 数据类型转换指令

PowerPro 提供了 240 个数据类型转换指令，用于各种数据类型之间相互转换。

✧ 语法：<TYPE1>_TO_<TYPE2>

- 禁止将“较大的”数据类型隐含地转换为“较小的”数据类型使用，当从较大数据类型转为较小数据类型时，有可能丢失信息。
- 如果被转换的值超出目标数据类型的存储范围，则这个数的高字节将被忽略。例如将 INT 类型转换为 BYTE 类型，或者将 DINT 类型转换为 WORD 类型。
- <TYPE>_TO_STRING 的转换中，字符串是从左边开始生成的。如果定义的字符串长度小于<TYPE>的长度，右边部分会被截去。

✧ 数据类型转换指令列表

表 4-7-1 列出了所有的数据类型转换指令，本小节讲述数据类型转换指令。

表 4-7-1

BOOL_TO_<TYPE>	BYTE_TO_<TYPE>	DATE_TO_<TYPE>	DINT_TO_<TYPE>
BOOL_TO_BYTE	BYTE_TO_BOOL	DATE_TO_BOOL	DINT_TO_BOOL
BOOL_TO_DATE	BYTE_TO_DATE	DATE_TO_BYTE	DINT_TO_BYTE
BOOL_TO_DINT	BYTE_TO_DINT	DATE_TO_DINT	DINT_TO_DATE
BOOL_TO_DT	BYTE_TO_DT	DATE_TO_DT	DINT_TO_DT
BOOL_TO_DWORD	BYTE_TO_DWORD	DATE_TO_DWORD	DINT_TO_DWORD
BOOL_TO_INT	BYTE_TO_INT	DATE_TO_INT	DINT_TO_INT
BOOL_TO_REAL	BYTE_TO_REAL	DATE_TO_REAL	DINT_TO_REAL
BOOL_TO_SINT	BYTE_TO_SINT	DATE_TO_SINT	DINT_TO_SINT
BOOL_TO_STRING	BYTE_TO_STRING	DATE_TO_STRING	DINT_TO_STRING
BOOL_TO_TIME	BYTE_TO_TIME	DATE_TO_TIME	DINT_TO_TIME
BOOL_TO_TOD	BYTE_TO_TOD	DATE_TO_TOD	DINT_TO_TOD
BOOL_TO_UDINT	BYTE_TO_UDINT	DATE_TO_UDINT	DINT_TO_UDINT
BOOL_TO_UINT	BYTE_TO_UINT	DATE_TO_UINT	DINT_TO_UINT
BOOL_TO_USINT	BYTE_TO_USINT	DATE_TO_USINT	DINT_TO_USINT
BOOL_TO_WORD	BYTE_TO_WORD	DATE_TO_WORD	DINT_TO_WORD
DT_TO_<TYPE>	**DWORD_TO_<TYPE>**	**INT_TO_<TYPE>**	**WORD_TO_<TYPE>**
DT_TO_BOOL	DWORD_TO_BOOL	INT_TO_BOOL	WORD_TO_BOOL
DT_TO_BYTE	DWORD_TO_BYTE	INT_TO_BYTE	WORD_TO_BYTE
DT_TO_DATE	DWORD_TO_DATE	INT_TO_DATE	WORD_TO_DATE
DT_TO_DINT	DWORD_TO_DINT	INT_TO_DINT	WORD_TO_DINT
DT_TO_DWORD	DWORD_TO_DT	INT_TO_DT	WORD_TO_DT
DT_TO_INT	DWORD_TO_INT	INT_TO_DWORD	WORD_TO_DWORD
DT_TO_REAL	DWORD_TO_REAL	INT_TO_REAL	WORD_TO_INT
DT_TO_SINT	DWORD_TO_SINT	INT_TO_SINT	WORD_TO_REAL
DT_TO_STRING	DWORD_TO_STRING	INT_TO_STRING	WORD_TO_SINT
DT_TO_TIME	DWORD_TO_TIME	INT_TO_TIME	WORD_TO_STRING
DT_TO_TOD	DWORD_TO_TOD	INT_TO_TOD	WORD_TO_TIME
DT_TO_UDINT	DWORD_TO_UDINT	INT_TO_UDINT	WORD_TO_TOD
DT_TO_UINT	DWORD_TO_UINT	INT_TO_UINT	WORD_TO_UDINT
DT_TO_USINT	DWORD_TO_USINT	INT_TO_USINT	WORD_TO_UINT
DT_TO_WORD	DWORD_TO_WORD	INT_TO_WORD	WORD_TO_USINT
REAL_TO_<TYPE>	**SINT_TO_<TYPE>**	**STRING_TO_<TYPE>**	**TIME_TO_<TYPE>**
REAL_TO_BOOL	SINT_TO_BOOL	STRING_TO_BOOL	TIME_TO_BOOL
REAL_TO_BYTE	SINT_TO_BYTE	STRING_TO_BYTE	TIME_TO_BYTE
REAL_TO_DATE	SINT_TO_DATE	STRING_TO_DATE	TIME_TO_DATE
REAL_TO_DINT	SINT_TO_DINT	STRING_TO_DINT	TIME_TO_DINT
REAL_TO_DT	SINT_TO_DT	STRING_TO_DT	TIME_TO_DT
REAL_TO_DWORD	SINT_TO_DWORD	STRING_TO_DWORD	TIME_TO_DWORD
REAL_TO_INT	SINT_TO_INT	STRING_TO_INT	TIME_TO_INT
REAL_TO_SINT	SINT_TO_REAL	STRING_TO_REAL	TIME_TO_REAL
REAL_TO_STRING	SINT_TO_STRING	STRING_TO_SINT	TIME_TO_SINT
REAL_TO_TIME	SINT_TO_TIME	STRING_TO_TIME	TIME_TO_STRING
REAL_TO_TOD	SINT_TO_TOD	STRING_TO_TOD	TIME_TO_TOD
REAL_TO_UDINT	SINT_TO_UDINT	STRING_TO_UDINT	TIME_TO_UDINT
REAL_TO_UINT	SINT_TO_UINT	STRING_TO_UINT	TIME_TO_UINT
REAL_TO_USINT	SINT_TO_USINT	STRING_TO_USINT	TIME_TO_USINT
REAL_TO_WORD	SINT_TO_WORD	STRING_TO_WORD	TIME_TO_WORD
TOD_TO_<TYPE>	**UDINT_TO_<TYPE>**	**UINT_TO_<TYPE>**	**USINT_TO_<TYPE>**
TOD_TO_BOOL	UDINT_TO_BOOL	UINT_TO_BOOL	USINT_TO_BOOL
TOD_TO_BYTE	UDINT_TO_BYTE	UINT_TO_BYTE	USINT_TO_BYTE
TOD_TO_DATE	UDINT_TO_DATE	UINT_TO_DATE	USINT_TO_DATE
TOD_TO_DINT	UDINT_TO_DINT	UINT_TO_DINT	USINT_TO_DINT
TOD_TO_DT	UDINT_TO_DT	UINT_TO_DT	USINT_TO_DT
TOD_TO_DWORD	UDINT_TO_DWORD	UINT_TO_DWORD	USINT_TO_DWORD
TOD_TO_INT	UDINT_TO_INT	UINT_TO_INT	USINT_TO_INT
TOD_TO_REAL	UDINT_TO_REAL	UINT_TO_REAL	USINT_TO_REAL
TOD_TO_SINT	UDINT_TO_SINT	UINT_TO_SINT	USINT_TO_SINT
TOD_TO_STRING	UDINT_TO_STRING	UINT_TO_STRING	USINT_TO_STRING
TOD_TO_TIME	UDINT_TO_TIME	UINT_TO_TIME	USINT_TO_TIME
TOD_TO_UDINT	UDINT_TO_TOD	UINT_TO_TOD	USINT_TO_TOD
TOD_TO_UINT	UDINT_TO_UINT	UINT_TO_UDINT	USINT_TO_UDINT
TOD_TO_USINT	UDINT_TO_USINT	UINT_TO_USINT	USINT_TO_UINT
TOD_TO_WORD	UDINT_TO_WORD	UINT_TO_WORD	USINT_TO_WORD

4.7.1 BOOL_TO_<TYPE>——布尔类型转换指令

- **功能：**把布尔数据类型转换为其它数据类型。
- **输入/输出数据类型**（参见表 4-7-1）：

输出为数字类型时，如果输入是 TRUE，则输出 1，如果输入是 FALSE，则输出为 0；输出为字符串类型时，如果输入是 TRUE，则输出字符串'TRUE'，如果输入是 FALSE，则输出为字符串'FALSE'。

✧ **指令使用举例**

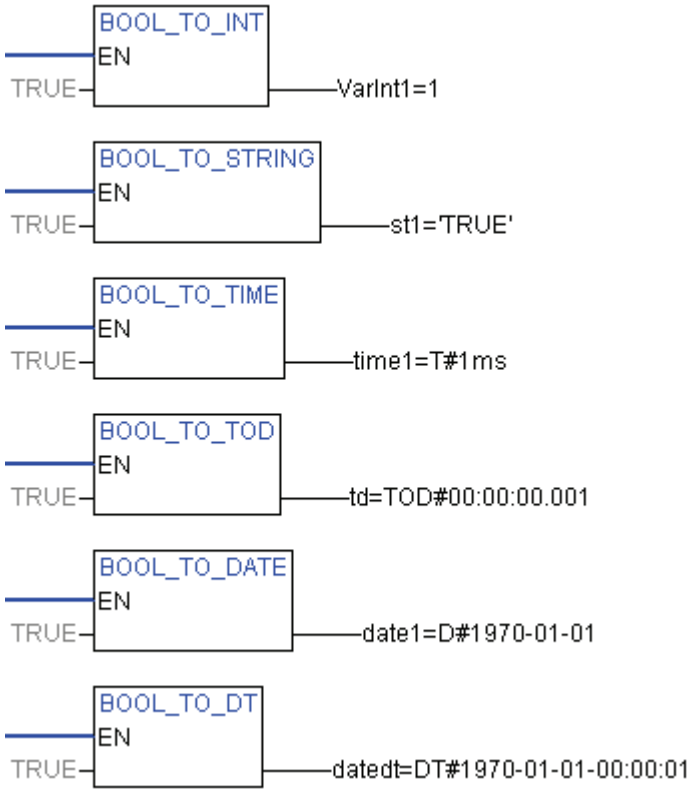
变量定义

VAR | VAR_INPUT | VAR_OUTPUT | VAR_IN_OUT | CONSTANT

	名称	地址	类型	初始值	注释
0001	VarInt1		INT		
0002	st1		STRING		
0003	time1		TIME		
0004	td		TOD		
0005	date1		DATE		
0006	datedt		DT		

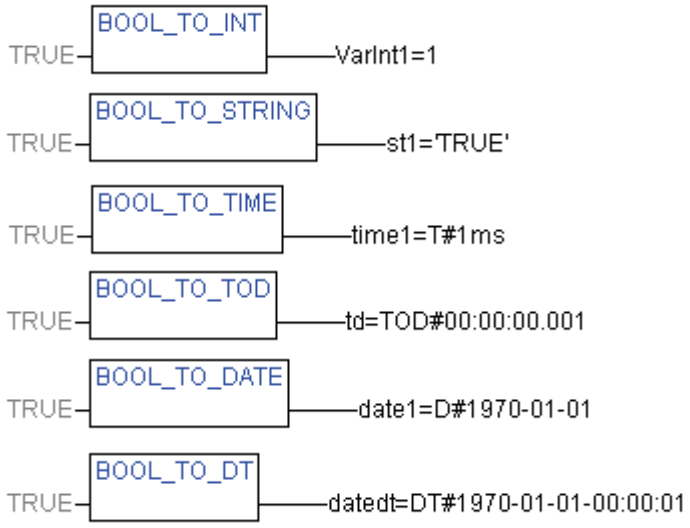
编程语言 | **程 序（部 分）**

梯形图（LD）



结构化文本（ST）

```
VarInt1:=BOOL_TO_INT(TRUE);     (*结果为 1*)
st1:=BOOL_TO_STRING(TRUE);      (*结果为'TRUE'*)
time1:=BOOL_TO_TIME(TRUE);      (*结果为 T#1ms*)
td:=BOOL_TO_TOD(TRUE);      (*结果为 TOD#00:00:00.001*)
```

	date1:=BOOL_TO_DATE(TRUE); (*结果为 D#1970-01-01*) datedt :=BOOL_TO_DT(TRUE); (*结果为 DT#1970-01-01-00:00:01*)
指令列表（IL）	LD TRUE BOOL_TO_INT ST VarInt1 (*结果为 1*) LD TRUE BOOL_TO_STRING ST st1 (*结果为'TRUE'*) LD TRUE BOOL_TO_TIME ST time1 (*结果为 T#1ms*) LD TRUE BOOL_TO_TOD ST td (*结果为 TOD#00:00:00.001*) LD TRUE BOOL_TO_DATE ST date1 (*结果为 D#1970-01-01*) LD TRUE BOOL_TO_DT ST datedt (*结果为 DT#1970-01-01-00:00:01*)
功能块（FBD）	

4.7.2 BYTE_TO_<TYPE>——字节类型转换指令

- **功能：**把字节类型转换为其他数据类型。
- **输入/输出数据类型**（参见表 4-7-1）：

当 BYTE_TO_BOOL 时，输入不等于 0 时输出为 TRUE，当输入等于 0 时输出为 FALSE；
当 BYTE_TO_TIME、BYTE_TO_TOD 时，输入将以毫秒值进行转换；
当 BYTE_TO_DATE、BYTE_TO_DT 时，输入将以秒值进行转换。

✧ 指令使用举例

变量定义

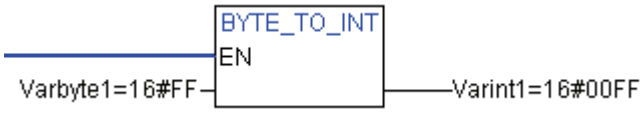
VAR | VAR_INPUT | VAR_OUTPUT | VAR_IN_OUT | CONSTANT

	名称	地址	类型	初始值	注释
0001	Varbool1		BOOL		
0002	Varbyte1		BYTE		
0003	Varint1		INT		
0004	Vartime1		TIME		
0005	Vardt1		DT		
0006	Varreal1		REAL		
0007	Varstring1		STRING		

编程语言 | **程 序（部 分）**

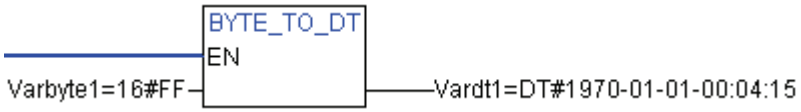
梯形图（LD）

(*结果为 TRUE *)

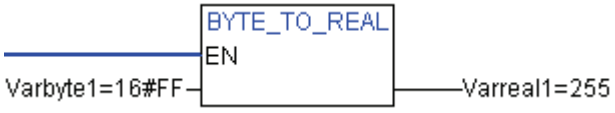


(*结果为 16# 00FF *)

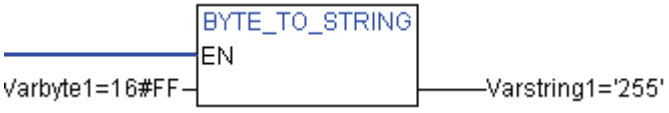
(*结果为 T#255ms *)



(*结果为 DT#1977-01-01-00:04:05 *)



(*结果为 255 *)



(*结果为字符串'255'*)

结构化文本（ST）

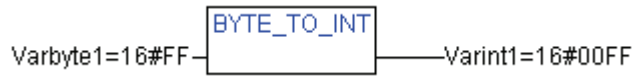
```
Varbyte1:=16#FF                                    (*Varbyte1 取值*)
Varbool1:=BYTE_TO_BOOL(Varbyte1);                  (*结果为 TRUE *)
Varint1:=BYTE_TO_INT(Varbyte1);                    (*结果为 16# FF *)
Vartime1:=BYTE_TO_TIME(Varbyte1);                  (*结果为 T#255ms *)
Vardt1:=BYTE_TO_DT(Varbyte1);
(*结果为 DT#1977-01-01-00:04:05 *)
Varreal1:=BYTE_TO_REAL(Varbyte1);                  (*结果为 255 *)
Varstring1:=BYTE_TO_STRING(Varbyte1); (*结果为字符串'255'*)
```

指令列表（IL）

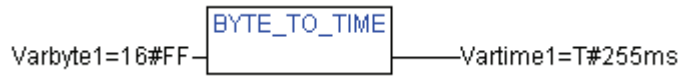
```
LD      16#FF
ST      Varbyte1                    (*Varbyte1 取值*)
LD      Varbyte1
BYTE_TO_BOOL
ST      Varbool1                    (*结果为 TRUE *)
LD      Varbyte1
BYTE_TO_INT
ST      Varint1                     (*结果为 16# FF *)
LD      Varbyte1
BYTE_TO_TIME
ST      Vartime1                    (*结果为 T#255ms *)
LD      Varbyte1
BYTE_TO_DT
ST      Vardt1          (*结果为 DT#1970-01-01-00:04:05 *)
LD      Varbyte1
BYTE_TO_REAL
ST      Varreal1                    (*结果为 255 *)
LD      Varbyte1
BYTE_TO_STRING
ST      Varstring1                  (*结果为字符串'255'*)
```

功能块（FBD）

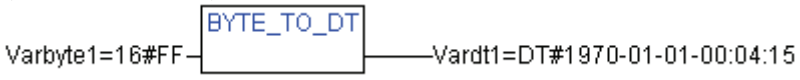
(*结果为 TRUE *)



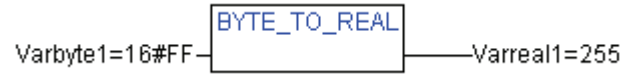
(*结果为 16# 00FF *)



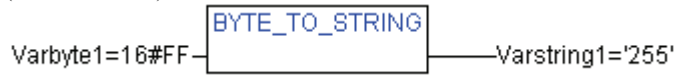
(*结果为 T#255ms *)



(*结果为 DT#1977-01-01-00:04:05 *)



(*结果为 255 *)



(*结果为字符串'255'*)

4.7.3 WORD_TO_<TYPE>——字类型转换指令

- **功能：**把字类型转换为其它数据类型。
- **输入/输出数据类型**（参见表 4-7-1）：

当 WORD_TO_BOOL 时，输入不等于 0 时输出为 TRUE，当输入等于 0 时输出为 FALSE；
当 WORD_TO_TIME、WORD_TO_TOD 时，输入将以毫秒值进行转换；
当 WORD_TO_DATE、WORD_TO_DT 时，输入将以秒值进行转换。

✧ 指令使用举例

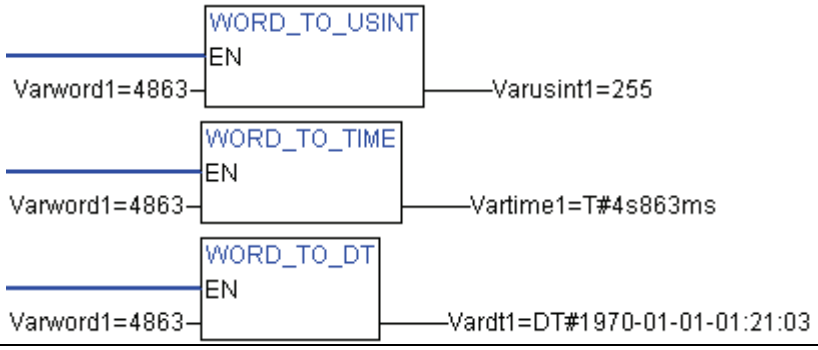
变量定义

VAR | VAR_INPUT | VAR_OUTPUT | VAR_IN_OUT | CONSTANT

	名称	地址	类型	初始值	注释
0001	Varusint1		USINT		
0002	Varword1		WORD		
0003	Vartime1		TIME		
0004	Vardt1		DT		

编程语言 程 序（部 分）

梯形图（LD）



结构化文本（ST）

```
Varword1:=4863;                          (*Varword1 取值*)
Varusint1:=WORD_TO_USINT(Varword1);      (*结果 255 *)
```

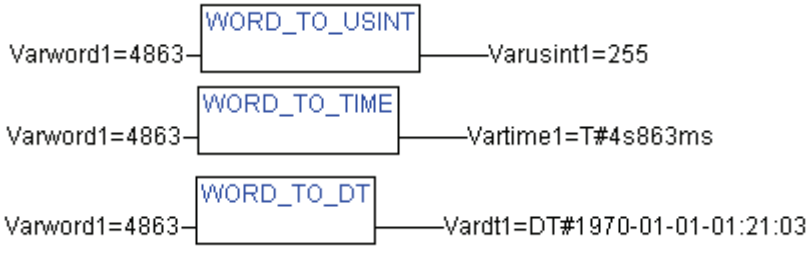
说明：如果将整数 4863（十六进制为 16#12FF）保存为 USINT 型变量，则会丢失高位数据，只显示低位数据 255（十六进制为 16#FF）。

```
Vartime1:=WORD_TO_TIME(Varword1);        (*结果 T#4s863ms*)
Vardt1:=WORD_TO_DT(Varword1);
(*结果 DT#1970-01-01-01:21:03 *)
```

指令列表（IL）

```
LD      4863
ST      Varword1                         (*Varword1 取值*)
LD      Varword1
WORD_TO_USINT
ST      Varusint1                        (*结果 255 *)
LD      Varword1
WORD_TO_TIME
ST      Vartime1                         (*结果 T#4s863ms*)
LD      Varword1
WORD_TO_DT
ST      Vardt1                   (*结果 DT#1970-01-01-01:21:03 *)
```

功能块（FBD）



4.7.4 DWORD_TO_<TYPE>——双字类型转换指令

- **功能：**把双字类型转换为其它数据类型。
- **输入/输出数据类型**（参见表 4-7-1）：

当 DWORD_TO_BOOL 时，输入不等于 0 时输出为 TRUE，当输入等于 0 时输出为 FALSE；
当 DWORD_TO_TIME、DWORD_TO_TOD 时，输入将以毫秒值进行转换；
当 DWORD_TO_DATE、DWORD_TO_DT 时，输入将以秒值进行转换。

✧ **指令使用举例**

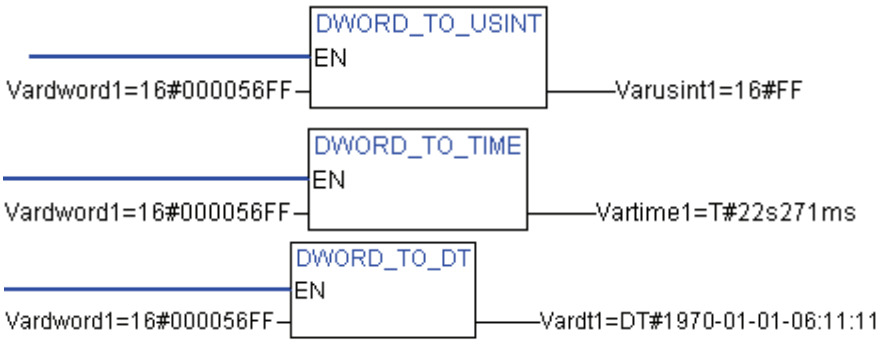
变量定义

VAR | VAR_INPUT | VAR_OUTPUT | VAR_IN_OUT | CONSTANT

	名称	地址	类型	初始值	注释
0001	Varusint1		USINT		
0002	Vardword1		DWORD		
0003	Vartime1		TIME		
0004	Vardt1		DT		

编程语言 | **程 序（部 分）**

梯形图（LD）



结构化文本（ST）

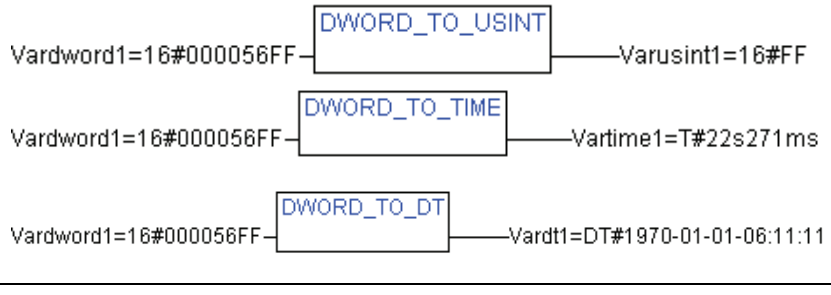
```
Varword1:=16#56FF;                          (*Varword1 取值*)
Varusint1:=DWORD_TO_USINT(Vardword1);       (*结果 255 *)
```

说明：如果将整数 16#56FF（十进制为 22271）保存为 USINT 型变量，则会丢失高位数据，只显示低位数据 255（十六进制为 16#FF）。

```
Vartime1:=DWORD_TO_TIME(Vardword1);     (*结果 T#22s271ms*)
Vardt1:=DWORD_TO_DT(Vardword1);
(*结果 DT#1970-01-01-06:11:11 *)
```

指令列表（IL）

```
LD      16#56FF
ST      Vardword1                       (*Vardword1 取值*)
LD      Vardword1
DWORD_TO_USINT
ST      Varusint1                       (*结果 255 *)
LD      Vardword1
DWORD_TO_TIME
ST      Vartime1                        (*结果 T#22s271ms*)
LD      Vardword1
DWORD_TO_DT
ST      Vardt1                  (*结果 DT#1970-01-01-06:11:11 *)
```

功 能 块（FBD）	

4.7.5 SINT_TO_<TYPE>——短整型转换指令

- **功能：**把短整型转换为其它数据类型。
- **输入/输出数据类型**（参见表 4-7-1）：

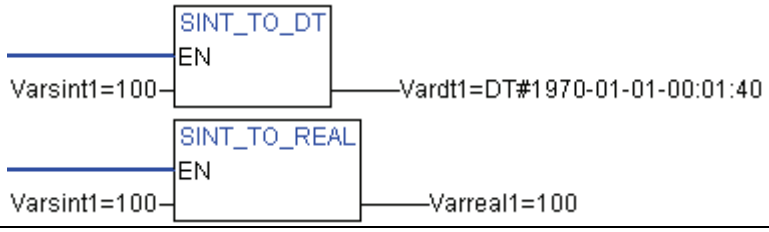
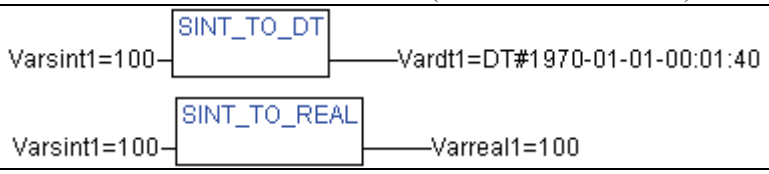
当 SINT_TO_BOOL 时，输入不等于 0 时输出为 TRUE，当输入等于 0 时输出为 FALSE；
当 SINT_TO_TIME、SINT _TO_TOD 时，输入将以毫秒值进行转换；
当 SINT_TO_DATE、SINT _TO_DT 时，输入将以秒值进行转换。

✧ **指令使用举例**

变量定义

VAR | VAR_INPUT | VAR_OUTPUT | VAR_IN_OUT | CONSTANT

	名称	地址	类型	初始值	注释
0001	Varsint1		SINT		
0002	Vardt1		DT		
0003	Varreal1		REAL		

编程语言	程 序（部 分）
梯形图（LD）	
结构化文本（ST）	Varsint1:=100; (*Varsint1 取值*) Vardt1:=SINT_TO_DT(Varsint1); (*结果 DT#1970-01-01-00:01:40 *) Varreal1:=SINT_TO_REAL(Varsint1); (*结果为 100 .0*)
指令列表（IL）	LD 100 ST Varsint1 (*Varsint1 取值*) LD Varsint1 SINT_TO_DT ST Vardt1 (*结果 DT#1970-01-01-00:01:40 *) LD Varsint1 SINT_TO_REAL ST Varreal1 (*结果 Varrea1 为 100.0*)
功能块（FBD）	

4.7.6 USINT_TO_<TYPE>——无符号短整型转换指令

- **功能：**把无符号短整型转换为其它数据类型。
- **输入/输出数据类型**（参见表 4-7-1）：

当 USINT_TO_BOOL 时，输入不等于 0 时输出为 TRUE，当输入等于 0 时输出为 FALSE；
当 USINT_TO_TIME、USINT _TO_TOD 时，输入将以毫秒值进行转换；
当 USINT_TO_DATE、USINT _TO_DT 时，输入将以秒值进行转换。

✧ **指令使用举例**

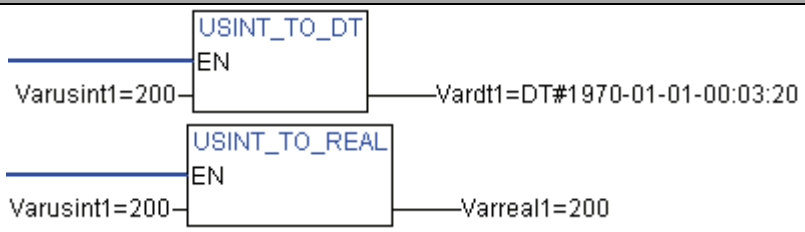
变量定义

VAR | VAR_INPUT | VAR_OUTPUT | VAR_IN_OUT | CONSTANT

	名称	地址	类型	初始值	注释
0001	Varusint1		USINT		
0002	Vardt1		DT		
0003	Varreal1		REAL		

编程语言 | **程 序（部 分）**

梯形图（LD）



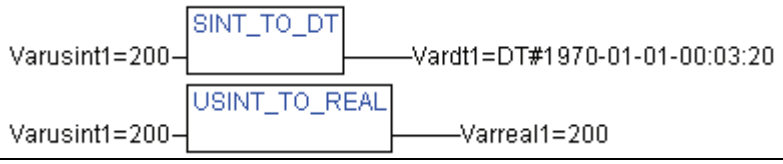
结构化文本（ST）

```
Varusint1:=200;                                    (*Varusint1 取值*)
Vardt1:=USINT_TO_DT(Varusint1);(*结果 DT#1970-01-01-00:03:20 *)
Varreal1:=USINT_TO_REAL(Varusint1);          (*结果为 200 .0*)
```

指令列表（IL）

```
LD      200
ST      Varusint1                          (*Varusint1 取值*)
LD      Varusint1
USINT_TO_DT
ST      Vardt1                  (*结果 DT#1970-01-01-00:03:20 *)
LD      Varusint1
USINT_TO_REAL
ST      Varreal1                  (*结果 Varreal1 为 200.0*)
```

功能块（FBD）



4.7.7 INT_TO_<TYPE>——整数类型转换指令

- **功能：**把整型数据类型转换为其它数据类型。
- **输入/输出数据类型**（参见表 4-7-1）：

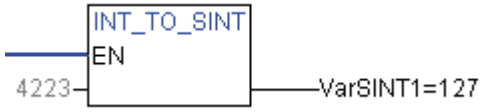
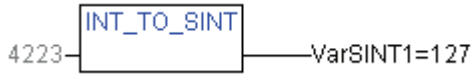
当 INT_TO_BOOL 时，输入不等于 0 时输出为 TRUE，当输入等于 0 时输出为 FALSE；
当 INT_TO_TIME、INT _TO_TOD 时，输入将以毫秒值进行转换；
当 INT_TO_DATE、INT _TO_DT 时，输入将以秒值进行转换。

✧ 指令使用举例

变量定义

VAR | VAR_INPUT | VAR_OUTPUT | VAR_IN_OUT | CONS

	名称	地址	类型	初始值	注释
0001	VarSINT1		SINT		
0002	VarREAL1		REAL		

编程语言	程 序（部 分）
梯形图（LD）	
结构化文本（ST）	VarSINT1 := INT_TO_SINT(4223); (*结果 VarSINT1 为 127 *) 说明：如果将整数 4223（十六进制为 16#107F）保存为 SINT 型变量，则会丢失高位数据，只显示低位数据 127（十六进制为 16#7F）。
指令列表（IL）	LD 2 INT_TO_REAL ST VarREAL1 (*结果 VarREAL1 为 2.0*)
功能块（FBD）	

4.7.8 UINT_TO_<TYPE>——无符号整数类型转换指令

➢ **功能：**无符号整数类型转换为其它数据类型。

➢ **输入/输出数据类型**（参见表 4-7-1）：

当 UINT_TO_BOOL 时，输入不等于 0 时输出为 TRUE，当输入等于 0 时输出为 FALSE；

当 UINT_TO_TIME、UINT _TO_TOD 时，输入将以毫秒值进行转换；

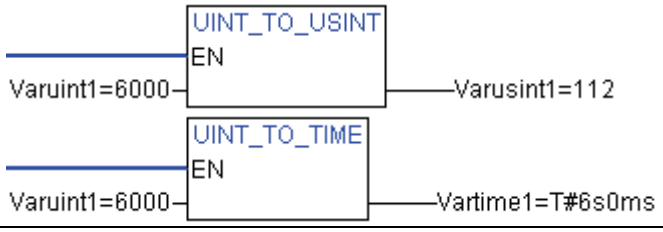
当 UINT_TO_DATE、UINT _TO_DT 时，输入将以秒值进行转换。

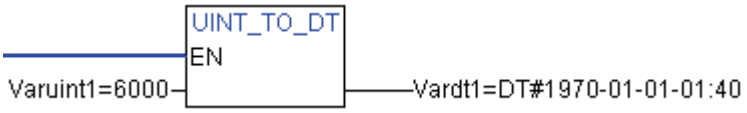
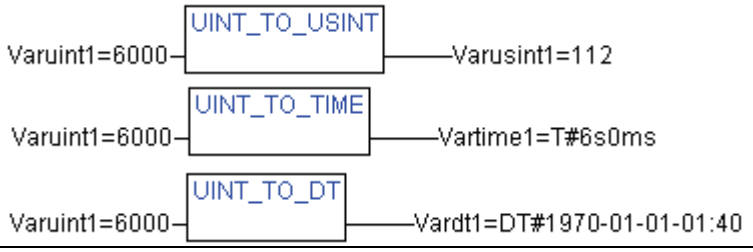
✧ 指令使用举例

变量定义

VAR | VAR_INPUT | VAR_OUTPUT | VAR_IN_OUT | CONSTANT

	名称	地址	类型	初始值	注释
0001	Varuint1		UINT		
0002	Varusint1		USINT		
0003	Vartime1		TIME		
0004	Vardt1		DT		

编程语言	程 序（部 分）
梯形图（LD）	

	
结构化文本（ST）	Varuint1:=6000; Varusint1:=UINT_TO_USINT(Varuint1); 说明：如果将整数 6000（十六进制为 16#1770）保存为 SINT 型变量，则会丢失高位数据，只显示低位数据 112（十六进制为 16#70）。 Vartime1:=UINT_TO_TIME(Varuint1); Vardt1:=UINT_TO_DT(Varuint1);
指令列表（IL）	LD 6000 ST Varuint1 (*Varuint1 取值*) LD Varuint1 UINT_TO_USINT ST Varusint1 (*结果 112*) LD Varuint1 UINT_TO_TIME ST Vartime1 (*结果 T#6s0ms*) LD Varuint1 UINT_TO_DT ST Vardt1 (*结果 DT#1970-01-01-01:40:00 *)
功能块（FBD）	

4.7.9 DINT_TO_<TYPE>——双整数类型转换指令

- **功能：**双整数类型类型转换为其它数据类型。
- **输入/输出数据类型**（参见表 4-7-1）：

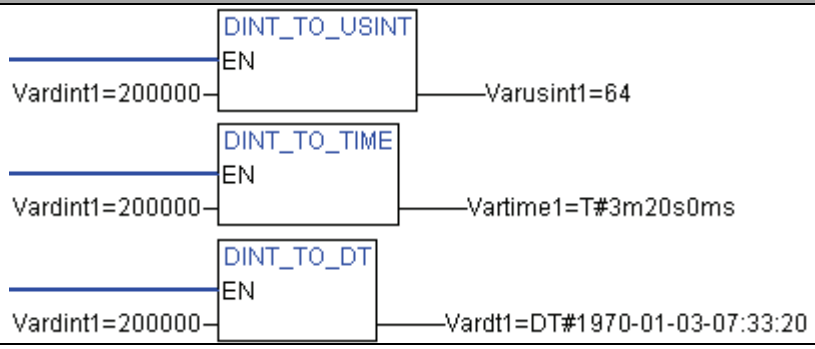
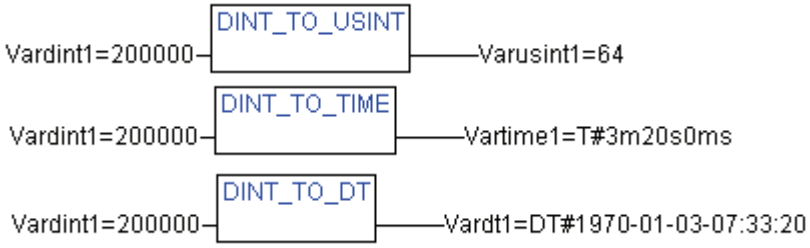
当 DINT_TO_BOOL 时，输入不等于 0 时输出为 TRUE，当输入等于 0 时输出为 FALSE；
当 DINT_TO_TIME、DINT _TO_TOD 时，输入将以毫秒值进行转换；
当 DINT_TO_DATE、DINT _TO_DT 时，输入将以秒值进行转换。

✧ **指令使用举例**

变量定义

VAR | VAR_INPUT | VAR_OUTPUT | VAR_IN_OUT | CONSTANT

	名称	地址	类型	初始值	注释
0001	Vardint1		DINT		
0002	Varusint1		USINT		
0003	Vartime1		TIME		
0004	Vardt1		DT		

编程语言	程 序（部 分）
梯形图（LD）	
结构化文本（ST）	Vardint1:=200000; Varusint1:=DINT_TO_USINT(Vardint1); (*结果 64*) 说明：如果将整数 200000（十六进制为 16#30D40）保存为 USINT 型变量，则会丢失高位数据，只显示低位数据 64（十六进制为 16#40）。 Vartime1:=DINT_TO_TIME(Vardint1); (*结果 T#3m20s0ms*) Vardt1:=DINT_TO_DT(Vardint1); (*结果 DT#1970-01-03-07:33:20 *)
指令列表（IL）	LD 200000 ST Vardint1 (*Vardint1 取值*) LD Vardint1 DINT_TO_USINT ST Varusint1 (*结果 64*) LD Vardint1 DINT_TO_TIME ST Vartime1 (*结果 T#3m20s0ms*) LD Vardint1 DINT_TO_DT ST Vardt1 (*结果 DT#1970-01-03-07:33:20 *)
功能块（FBD）	

4.7.10 UDINT_TO_<TYPE>——无符号双整数类型转换指令

- **功能：**无符号双整数类型转换为其它数据类型。
- **输入/输出数据类型**（参见表 4-7-1）：

当 UDINT_TO_BOOL 时，输入不等于 0 时输出为 TRUE，当输入等于 0 时输出为 FALSE；
当 UDINT_TO_TIME、UDINT _TO_TOD 时，输入将以毫秒值进行转换；
当 UDINT_TO_DATE、UDINT _TO_DT 时，输入将以秒值进行转换。

✧ 指令使用举例

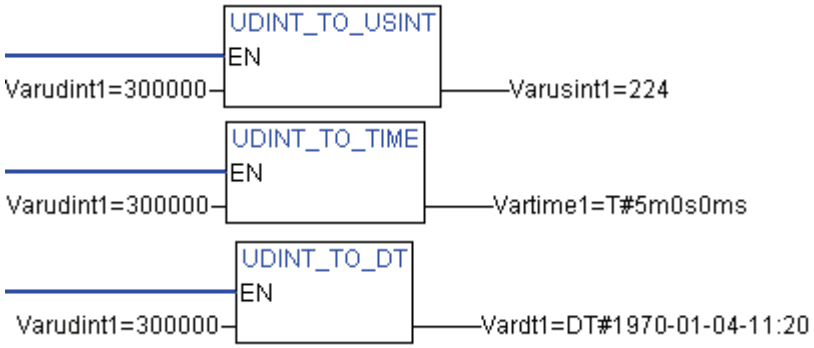
变量定义

VAR | VAR_INPUT | VAR_OUTPUT | VAR_IN_OUT | CONSTANT

	名称	地址	类型	初始值	注释
0001	Varudint1		UDINT		
0002	Varusint1		USINT		
0003	Vartime1		TIME		
0004	Vardt1		DT		

编程语言 | **程 序（部 分）**

梯形图（LD）



结构化文本（ST）

```
Varudint1:=300000;
Varusint1:= UDINT_TO_USINT(Varudint1);          (*结果 224*)
```

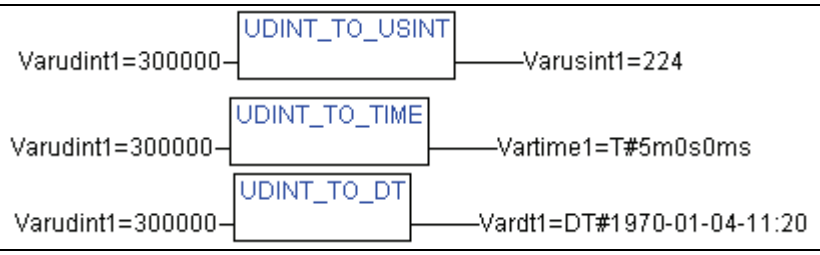
说明：如果将整数 300000（十六进制为 16#493E0）保存为 USINT 型变量，则会丢失高位数据，只显示低位数据 224（十六进制为 16#E0）。

```
Vartime1:=UDINT_TO_TIME(Varudint1);             (*结果 T#5m0s0ms*)
Vardt1:=UDINT_TO_DT(Varudint1);
(*结果 DT#1970-01-04-11:20:00 *)
```

指令列表（IL）

```
LD      300000
ST      Varudint1                    (*Varudint1 取值*)
LD      Varudint1
UDINT_TO_USINT
ST      Varusint1                    (*结果 224*)
LD      Varudint1
UDINT_TO_TIME
ST      Vartime1                     (*结果 T#5m0s0ms*)
LD      Varudint1
UDINT_TO_DT
ST      Vardt1          (*结果 DT#1970-01-04-11:20:00 *)
```

功能块（FBD）



4.7.11 REAL_TO_<TYPE>——实数类型转换指令

- **功能：**把浮点数转换为其它类型数据。把浮点数转换为其它类型数据时，先将值四舍五入成整数值，然后转成新的变量类型。
- **输入/输出数据类型**（参见表 4-7-1）：

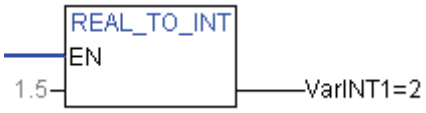
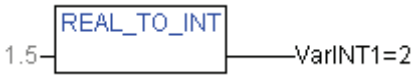
当 REAL_TO_BOOL 时，输入不等于 0 时输出为 TRUE，当输入等于 0 时输出为 FALSE；
当 REAL_TO_TIME、REAL_TO_TOD 时，输入将以毫秒值进行转换；
当 REAL_TO_DATE、REAL_TO_DT 时，输入将以秒值进行转换。

✧ **指令使用举例**

变量定义

VAR | VAR_INPUT | VAR_OUTPUT | VAR_IN_OUT | CONSTANT

	名称	地址	类型	初始值	注释
0001	Varint1		INT		
0002	Varint2		INT		
0003	Varint3		INT		
0004	Varint4		INT		

编程语言	程　序（部 分）
梯形图（LD）	
结构化文本（ST）	VarINT1:= REAL_TO_INT(1.5);　(*结果 VarINT1 为 2*) VarINT2:= REAL_TO_INT(1.4);　(*结果 VarINT2 为 1*) VarINT3:= REAL_TO_INT(-1.5);　(*结果 VarINT3 为-2*) VarINT4:= REAL_TO_INT(-1.4);　(*结果 VarINT4 为-1*)
指令列表（IL）	LD　2.7 REAL_TO_INT ST　VarINT1　(*结果 VarINT1 为 3*)
功能块（FBD）	

4.7.12 TIME_TO_<TYPE>——时间类型转换指令

- **功能：**把时间型数据转换为其它类型数据，时间在内部以毫秒为单位存储成 DWORD 类型（对于 TIME_OF_DAY 变量从凌晨 00：00 开始）。
- **输入/输出数据类型**（参见表 4-7-1）：

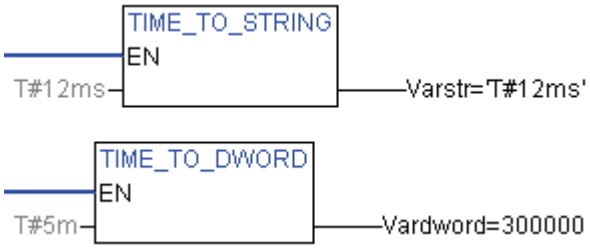
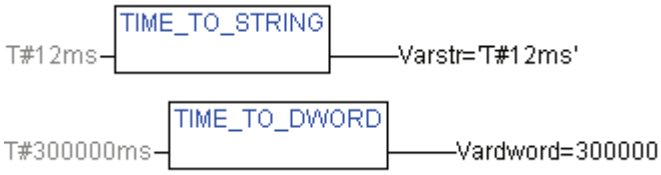
当 TIME_TO_BOOL 时，输入不等于 0 时输出为 TRUE，当输入等于 0 时输出为 FALSE。

✧ **指令使用举例**

变量定义

VAR | VAR_INPUT | VAR_OUTPUT | VAR_IN_OUT | CONS

	名称	地址	类型	初始值	注释
0001	Varstr		STRING		
0002	Vardword		DWORD		

编程语言	程　序（部 分）
梯形图（LD）	
结构化文本（ST）	Varstr:=TIME_TO_STRING(T#12ms);　(*结果为‘T#12ms’*) Vardword:=TIME_TO_DWORD(T#5m);　(*结果为 300000*)
指令列表（IL）	LD　T#12ms TIME_TO_STRING ST　Varstr　(*结果为‘T#12ms’*) LD　T#300000ms TIME_TO_DWORD ST　Vardword　(*结果为 300000*)
功能块（FBD）	

4.7.13　DATE_TO_<TYPE>——日期类型转换指令

- **功能：**把日期型数据转换为其它类型数据，日期在内部以秒为单位存储，时间从 1970 年 1 月 1 日开始。
- **输入/输出数据类型**（参见表 4-7-1）：

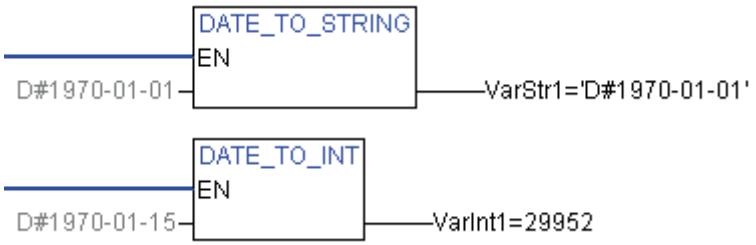
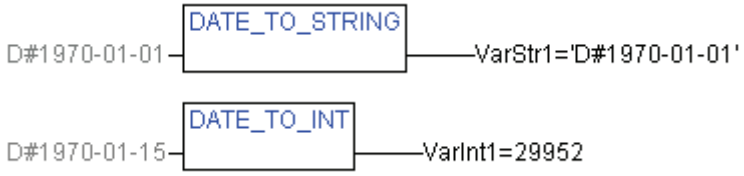
当 DATE_TO_BOOL 时，输入不等于 0 时输出为 TRUE，当输入等于 0 时输出为 FALSE。

✧ **指令使用举例**

变量定义

VAR | VAR_INPUT | VAR_OUTPUT | VAR_IN_OUT | CONS

	名称	地址	类型	初始值	注释
0001	Varint1		INT		
0002	VarStr1		STRING		

编程语言	程　序（部 分）
梯形图（LD）	
结构化文本（ST）	VarStr1:=DATE_TO_STRING(D#1970-01-01); (*结果为'D#1970-01-01' *) VarInt1:=DATE_TO_INT(D#1970-01-15);　(*结果为 29952*) 说明：将 D#1970-01-15（十进制 14×24×3600=1209600=16#127500）保存为 INT 型变量，则会丢失高位数据，只显示低位数据 16#7500,转换十进制数为 29952。
指令列表（IL）	LD　　D#1970-01-01 DATE_TO_STRING ST　　VarStr1　　(*结果为‘TRUE’*) LD　　D#1970-01-15 DATE_TO_INT ST　　VarInt1　　(*结果为 29952*)
功能块（FBD）	

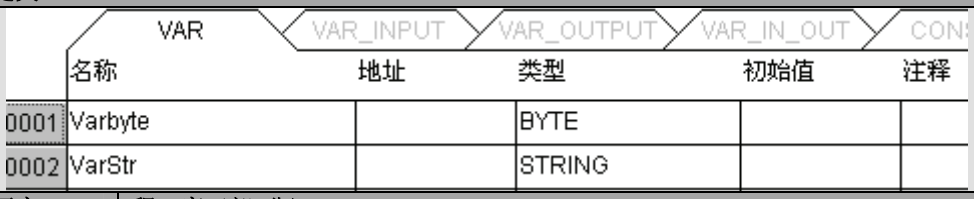
4.7.14　DT_TO_<TYPE>——日期时间类型转换指令

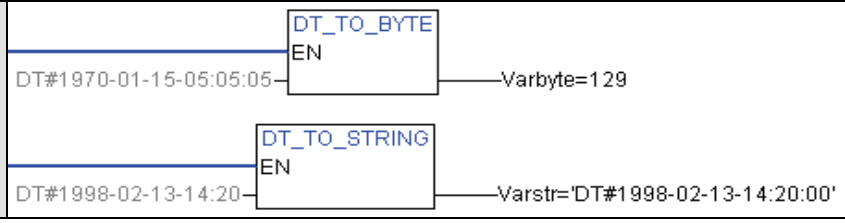
- **功能：**把日期时间型数据转换为其它类型数据，日期在内部以秒为单位存储，时间从 1970 年 1 月 1 日开始。
- **输入/输出数据类型**（参见表 4-7-1）：

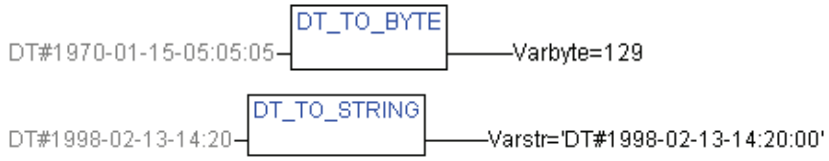
当 DT_TO_BOOL 时，输入不等于 0 时输出为 TRUE，当输入等于 0 时输出为 FALSE。

✧　**指令使用举例**

变量定义



编程语言	程　序（部 分）
梯形图（LD）	

结构化文本（ST）	Varbyte:=DT_TO_BYTE(DT#1970-01-15-05:05:05); (*结果 Varbyte 为 129*) 说明：将 DT#1970-01-15-05:05:05 转换成秒数，值为(((14*24+5)*60+5)*60+5)=1227905=16#12BC81，保存为 BYTE 型变量，则会丢失高 24 位数据，只显示低 8 位数据，16#81= 129。 Varstr:=DT_TO_STRING(DT#1998-02-13-14:20); (*结果 Varstr 为‘DT#1998-02-13-14:20’)
指令列表（IL）	LD DT#1970-01-15-05:05:05 DT_TO_BYTE ST Varbyte (*结果 Varbyte 为 129*) LD DT#1998-02-13-14:20 DT_TO_STRING ST Varstr (*结果 Varstr 为‘DT#1998-02-13-14:20’)
功能块（FBD）	

4.7.15 TOD_TO_<TYPE>——时间类型转换指令

- **功能：**把时间型数据转换为其它类型数据，日期在内部以毫秒为单位进行转化。
- **输入/输出数据类型**（参见表 4-7-1）：

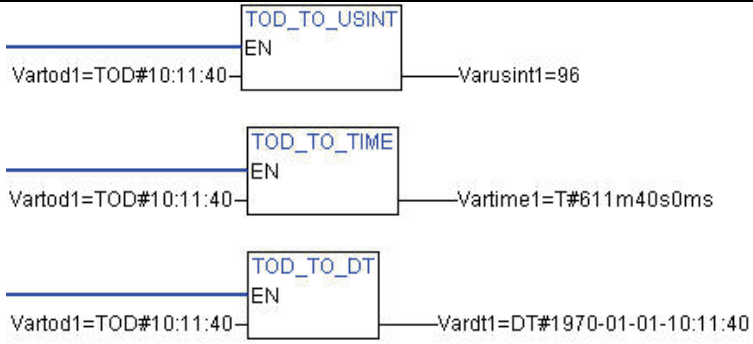
当 TOD_TO_BOOL 时，输入不等于 0 时输出为 TRUE，当输入等于 0 时输出为 FALSE。

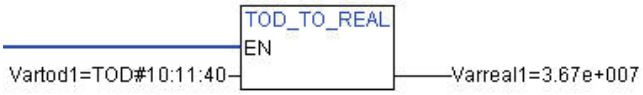
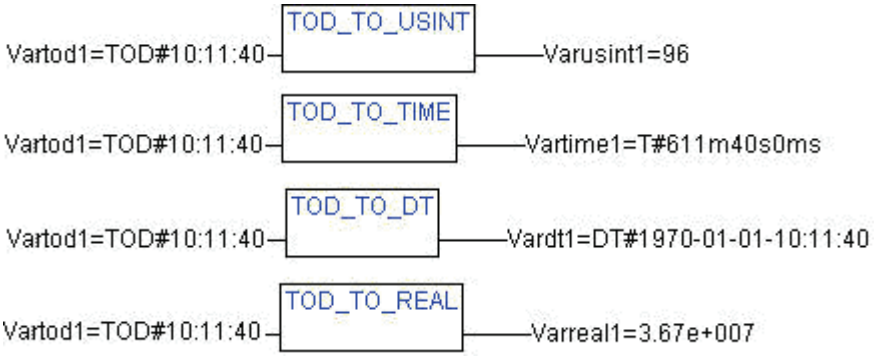
✧ **指令使用举例**

变量定义

VAR | VAR_INPUT | VAR_OUTPUT | VAR_IN_OUT | CONSTANT

	名称	地址	类型	初始值	注释
0001	Vartod1		TOD		
0002	varbool1		BOOL		
0003	Varusint1		USINT		
0004	Vartime1		TIME		
0005	Vardt1		DT		
0006	Varreal1		REAL		

编程语言	程 序（部 分）
梯形图（LD）	

	
结构化文本（ST）	Vartod1:=TOD#10:11:40; (*Vartod1 取值*) Varusint1:=TOD_TO_USINT(Vartod1); (*结果为 96*) Vartime1:=TOD_TO_TIME(Vartod1); (*结果为 T#611m40s0ms*) Vardt1:=TOD_TO_DT(Vartod1); (*结果为 DT#1970-01-01-10:11:40*) Varreal1:=TOD_TO_REAL(Vartod1); (*结果为 3.67e+007*)
指令列表（IL）	LD TOD#10:11:40 ST Vartod1 (*Vartod1 取值*) LD Vartod1 TOD_TO_USINT ST Varusint1 (*结果为 96*) LD Vartod1 TOD_TO_TIME ST Vartime1 (*结果为 T#611m40s0ms*) LD Vartod1 TOD_TO_DT ST Vardt1 (*结果为 DT#1970-01-01-10:11:40*) LD Vartod1 TOD_TO_REAL ST Varreal1 (*结果为 3.67e+007*)
功能块（FBD）	

4.7.16 STRING_TO_<TYPE>——字符类型转换指令

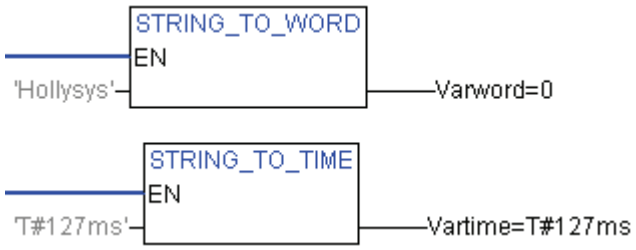
- **功能：**把字符串转换为其它类型数据，字符串型变量必须包含一个有效的目标变量值，否则转换结果为 0。
- **输入/输出数据类型：**（参见表 4-7-1）

✧ **指令使用举例**

变量定义

VAR | VAR_INPUT | VAR_OUTPUT | VAR_IN_OUT | CON

	名称	地址	类型	初始值	注释
0001	Varword		WORD		
0002	Vartime		TIME		

编程语言	程　序（部 分）
梯形图（LD）	
结构化文本（ST）	Varword:=STRING_TO_WORD('Hollysys'); (*结果为 0*) Vartime:=STRING_TO_TIME('T#127ms');　(*结果为 T#127ms*)
指令列表（IL）	LD　'Hollysys' STRING_TO_WORD ST　Varword　(*结果 Varword 为 0*) LD　'T#127ms' STRING_TO_TIME ST　Vartime　(*结果 Vartime 为 T#127ms*)
功能块（FBD）	

4.7.17　TRUNC——截短转换指令

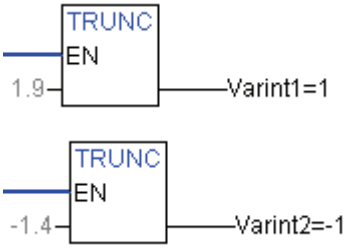
- **功能：**该指令将数据截去小数部分，只保留整数部分。
- **输入/输出数据类型：**输入为 REAL 型，输出为 INT、WORD、DWORD 型。

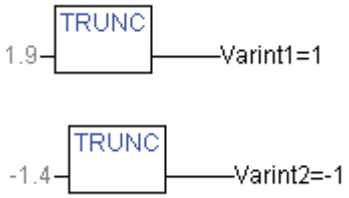
✧　**指令使用举例**

变量定义

VAR | VAR_INPUT | VAR_OUTPUT | VAR_IN_OUT | CON

	名称	地址	类型	初始值	注释
0001	Varint1		INT		
0002	Varint2		INT		

编程语言	程　序
梯形图（LD）	
结构化文本（ST）	Varint1:=TRUNC(1.9);　(*结果 Varint1 为 1*) Varint2:=TRUNC(-1.4);　(*结果 Varint2 为-1*)

指令列表（IL）	LD 1.9 TRUNC ST Varint1 (*结果 Varint1 为 1*) LD -1.4 TRUNC ST Varint2 (*结果 Varint2 为-1*)
功能块（FBD）	

提示：

- 当从较大数据类型转为较小数据类型时，有可能丢失信息。
- 本指令只是截取整数部分，如果想四舍五入取整，可以使用 REAL_TO_INT 指令。

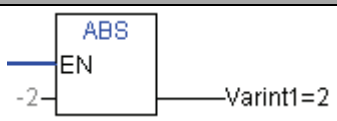
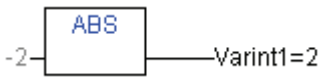
4.8 初等数学运算指令

4.8.1 ABS——绝对值指令

- **功能：**把输入数据的绝对值赋予输出变量。
- **输入/输出数据类型：**见下表

输入数据类型	输出数据类型
INT	INT、WORD、DWORD、DINT、UINT、REAL
REAL	REAL
BYTE	INT、BYTE、WORD、DWORD、DINT、UINT、REAL
WORD	WORD、DWORD、DINT、REAL
DWORD	DWORD、DINT、REAL
SINT	INT、BYTE、WORD、DWORD、DINT、UINT、REAL
USINT	INT、BYTE、WORD、DWORD、DINT、UINT、REAL
UINT	WORD、DWORD、DINT、UINT、REAL
DINT	DWORD、DINT、REAL
UDINT	DWORD、DINT、UDINT、REAL

✧ **指令使用举例**

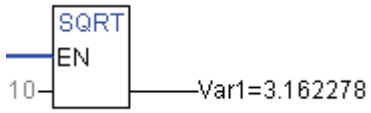
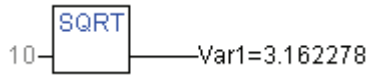
变量定义	
编程语言	**程　序**
梯形图（LD）	
结构化文本（ST）	i:=ABS(-2); (*结果 Varint1 为 2*)
指令列表（IL）	LD -2 ABS ST Varint1 (*结果 Varint1 为 2*)
功能块（FBD）	

4.8.2 SQRT——平方根指令

- **功能：**对输入数据求平方根，输入数据为非负数。
- **输入/输出数据类型：**

输入数据类型可以是 BYTE、WORD、DWORD、INT、DINT、REAL、SINT、USINT、UINT、UDINT，输出数据必须是 REAL 类型。

✧ **指令使用举例**

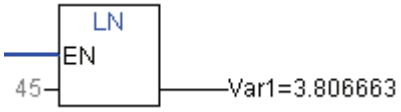
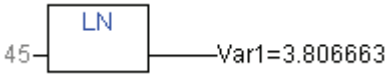
变量定义	
编程语言	**程　序**
梯形图（LD）	
结构化文本（ST）	Var1:=SQRT(10); (*结果 Var1 为 3.162278*)
指令列表（IL）	LD　10 SQRT ST　Var1　(*结果 Var1 为 3.162278*)
功能块（FBD）	

4.8.3 LN——自然对数指令

- **功能：**对输入数据求自然对数，输入数据必须为正数。
- **输入/输出数据类型：**

输入数据类型可以是 BYTE、WORD、DWORD、INT、DINT、REAL、SINT、USINT、UINT、UDINT，输出数据必须是 REAL 型。

✧ **指令使用举例**

变量定义	
编程语言	**程　序**
梯形图（LD）	
结构化文本（ST）	Var1:=LN(45);　(*结果 Var1 为 3.806663*)
指令列表（IL）	LD　45 LN ST　Var1　(*结果 Var1 为 3.806663*)
功能块（FBD）	

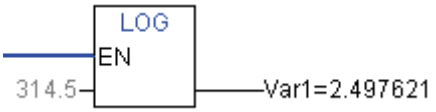
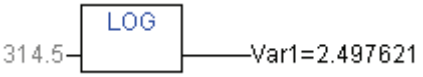
4.8.4 LOG——常用对数指令

- **功能：**对输入数据求以 10 为底的对数，输入数据必须为正数。
- **输入/输出数据类型：**

输入数据类型可以是 BYTE、WORD、DWORD、INT、DINT、REAL、SINT、USINT、UINT、UDINT，输出数据必须是 REAL 型。

✧ **指令使用举例**

变量定义

编程语言	程　序
梯形图（LD）	
结构化文本（ST）	Var1:=LOG(314.5);　(*结果 Var1 为 2.497621 *)
指令列表（IL）	LD　314.5 LOG ST　Var1　(*结果 Var1 为 2.497621 *)
功能块（FBD）	

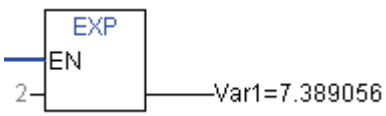
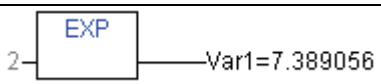
4.8.5 EXP——指数指令

- **功能：**以输入数据为指数的幂计算，即 $y = e^x$，其中 x 为输入，y 为输出。
- **输入/输出数据类型：**

输入数据类型可以是 BYTE、WORD、DWORD、INT、DINT、REAL、SINT、USINT、UINT、UDINT，输出数据必须是 REAL 型。

✧ **指令使用举例**

变量定义

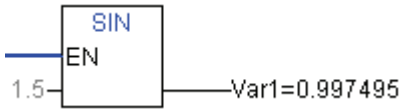
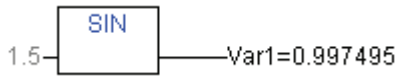
编程语言	程　序
梯形图（LD）	
结构化文本（ST）	Var1:=EXP(2);　（*结果 Var1 为 7.389056*）
指令列表（IL）	LD　2 EXP ST　Var1　（*结果 Var1 为 7.389056*）
功能块（FBD）	

4.8.6 SIN——正弦指令

- **功能：**求输入数据的正弦值，输入数据以弧度表示。弧度(rad) = 角度 $*\frac{\pi}{180^{o}}$
- **输入/输出数据类型：**输入数据类型可以是 BYTE、WORD、DWORD、INT、DINT、REAL、SINT、USINT、UINT、UDINT，输出数据必须是 REAL 型。

✧ **指令使用举例**

变量定义

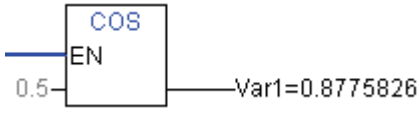
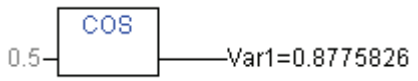
编程语言	程　序
梯形图（LD）	
结构化文本（ST）	Var1:=SIN(1.5);　　(*结果 Var1 为 0.997495 *)
指令列表（IL）	LD　1.5 SIN ST　Var1　　(*结果 Var1 为 0.997495 *)
功能块（FBD）	

4.8.7 COS——余弦指令

- **功能：**求输入数据的余弦值，输入数据以弧度表示。弧度(rad) = 角度 $*\frac{\pi}{180^{o}}$
- **输入/输出数据类型：**输入数据类型可以是 BYTE、WORD、DWORD、INT、DINT、REAL、SINT、USINT、UINT、UDINT，输出数据必须是 REAL 型。

✧ **指令使用举例**

变量定义

编程语言	程　序
梯形图（LD）	
结构化文本（ST）	Var1:=COS(0.5);　　(*结果 Var1 为 0.8775826 *)
指令列表（IL）	LD　0.5 COS ST　Var1　　(*结果 Var1 为 0.8775826 *)
功能块（FBD）	

4.8.8 TAN——正切指令

- **功能：**求输入数据的正切值，输入数据以弧度表示。弧度(rad) = 角度 * $\frac{\pi}{180^o}$
- **输入/输出数据类型：**输入数据类型可以是 BYTE、WORD、DWORD、INT、DINT、REAL、SINT、USINT、UINT、UDINT，输出数据必须是 REAL 型。

✧ **指令使用举例**

变量定义					
VAR / VAR_INPUT / VAR_OUTPUT / VAR_IN_OUT / CON					
	名称	地址	类型	初始值	注释
0001	Var1		REAL		

编程语言	程　序
梯形图（LD）	TAN, EN, 0.5, Var1=0.5463025
结构化文本（ST）	Var1:=TAN(0.5);　(*结果 Var1 为 0.5463025 *)
指令列表（IL）	LD 0.5 TAN ST Var1　(*结果 Var1 为 0.5463025 *)
功能块（FBD）	TAN, 0.5, Var1=0.5463025

4.8.9 ASIN——反正弦指令

- **功能：**求输入数据的反正弦值。
- **输入/输出数据类型：**

输入数据类型可以是 BYTE、WORD、DWORD、INT、DINT、REAL、SINT、USINT、UINT、UDINT，输出数据必须是 REAL 型，输出数据以弧度表示。

✧ **指令使用举例**

变量定义					
VAR / VAR_INPUT / VAR_OUTPUT / VAR_IN_OUT / CON					
	名称	地址	类型	初始值	注释
0001	Var1		REAL		

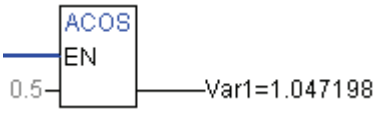
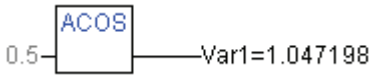
编程语言	程　序
梯形图（LD）	ASIN, EN, 0.5, Var1=0.5235988
结构化文本（ST）	Var1:=ASIN(0.5);　(*结果 Var1 为 0.5235988 *)
指令列表（IL）	LD 0.5 ASIN ST Var1　(*结果 Var1 为 0.5235988 *)
功能块（FBD）	ASIN, 0.5, Var1=0.5235988

4.8.10 ACOS——反余弦指令

- **功能：**求输入数据的反余弦值。
- **输入/输出数据类型：**

输入数据类型可以是 BYTE、WORD、DWORD、INT、DINT、REAL、SINT、USINT、UINT、UDINT，输出数据必须是 REAL 型，输出数据以弧度表示。

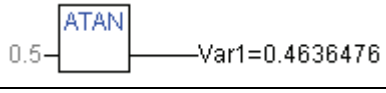
✧ **指令使用举例**

变量定义	
编程语言	**程　序**
梯形图（LD）	
结构化文本（ST）	Var1:=ACOS(0.5);　　(*结果 Var1 为 1.047198 *)
指令列表（IL）	LD　　0.5 ACOS ST　　Var1　　(*结果 Var1 为 1.047198 *)
功能块（FBD）	

4.8.11 ATAN——反正切指令

- **功能：**求输入数据的反正切值。
- **输入/输出数据类型：**输入数据类型可以是 BYTE、WORD、DWORD、INT、DINT、REAL、SINT、USINT、UINT、UDINT，输出数据必须是 REAL 型，输出数据以弧度表示。

✧ **指令使用举例**

变量定义	
编程语言	**程　序**
梯形图（LD）	
结构化文本（ST）	Var1:=ATAN(0.5);　(*结果 Var1 为 0.4636476 *)
指令列表（IL）	LD　　0.5 ATAN ST　　Var1　　(*结果 Var1 为 0.4636476 *)
功能块（FBD）	

4.8.12 EXPT——幂指令

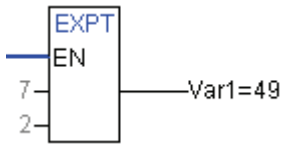
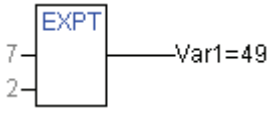
- **功能：**对输入数据求幂，输入数据 1 为幂底数，输入数据 2 为幂指数。
- **输入/输出数据类型：**输入数据的类型可以是 BYTE、WORD、DWORD、INT、DINT、REAL、SINT、USINT、UINT、UDINT，输出数据必须是 REAL 型。

✧ **指令使用举例**

变量定义

VAR | VAR_INPUT | VAR_OUTPUT | VAR_IN_OUT | CON

	名称	地址	类型	初始值	注释
0001	Var1		REAL		

编程语言	程　序
梯形图（LD）	
结构化文本（ST）	Var1:=EXPT(7,2);　　(*结果 Var1 为 49 *)
指令列表（IL）	LD　　7 EXPT　　2 ST　　Var1　　(*结果 Var1 为 49 *)
功能块（FBD）	

4.9 地址运算指令

4.9.1 ADR——取地址指令

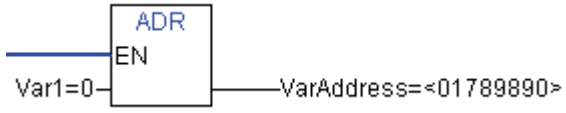
- **功能：**取得输入变量的内存地址，并输出。该地址可以在程序内当作指针使用，也可以作为指针传送给函数。

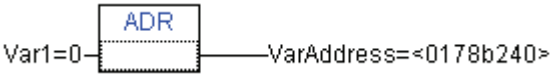
✧ **指令使用举例**

变量定义

VAR | VAR_INPUT | VAR_OUTPUT | VAR_IN_OUT | CONS

	名称	地址	类型	初始值	注释
0001	Var1		BYTE		
0002	VarAddress		POINTER TO BYTE		

编程语言	程　序
梯形图（LD）	
结构化文本（ST）	VarAddress:= ADR(Var1);

指令列表（IL）	LD Var1 ADR ST VarAddress
功能块（FBD）	

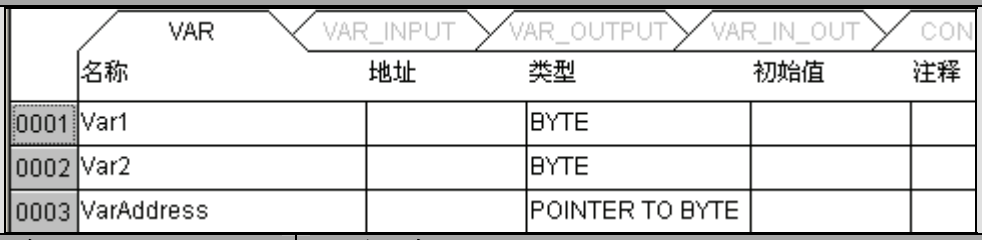
4.9.2 ^——取地址内容指令

➢ **功能：**

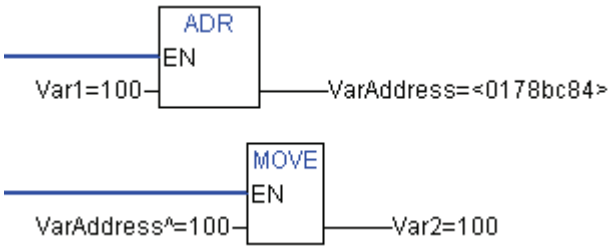
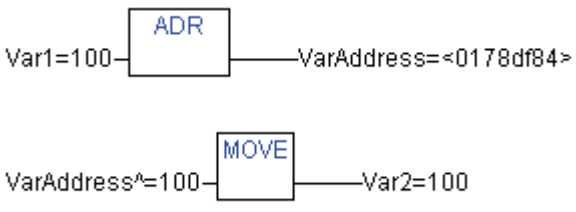
在指针变量后增加“^”符号，以取得该指针所指地址的数据。

✧ **指令使用举例**

变量定义



	名称	地址	类型	初始值	注释
0001	Var1		BYTE		
0002	Var2		BYTE		
0003	VarAddress		POINTER TO BYTE		

编程语言	**程　序**
梯形图（LD）	
结构化文本（ST）	VarAddress:= ADR(Var1); Var2:= VarAddress^;　(*结果 Var2 为 100 *)
指令列表（IL）	LD　Var1 ADR ST　VarAddress LD　VarAddress^ ST　Var2
功能块（FBD）	

4.9.3 BITADR——位地址指令

➢ **功能**：取得 BOOL 量的位地址，下例中 MX300.7 的地址为 300*8+7=2407。

✧ **指令使用举例**

变量定义

VAR | VAR_INPUT | VAR_OUTPUT | VAR_IN_OUT | CON

	名称	地址	类型	初始值	注释
0001	Varbool1	%MX300.7	BOOL		
0002	Bitadr1		DWORD		

编程语言	程　序
梯形图（LD）	
结构化文本（ST）	Bitadr1:=BITADR(Varbool1)；
指令列表（IL）	LD　Varbool1 BITADR ST　Bitadr1
功能块（FBD）	

4.9.4 INDEXOF——索引指令

➢ **功能**：在 POU 中执行索引指令，可以寻找 POU 的索引号，其输入为 POU 的名称，输出为 INT 的数据。

✧ **指令使用举例**

变量定义

VAR | VAR_INPUT | VAR_OUTPUT | VAR_IN_OUT | CON

	名称	地址	类型	初始值	注释
0001	Var1		INT		

编程语言	程　序
梯形图（LD）	（POU2 可以是程序、功能块、函数）
结构化文本（ST）	Var1:=INDEXOF(POU2);　（*结果 Var1 为 38*）
指令列表（IL）	LD　POU2 INDEXOF ST　Var1　（*结果 Var1 为 38*）
功能块（FBD）	（*结果 Var1 为 38*）

4.9.5 SIZEOF——数据类型大小指令

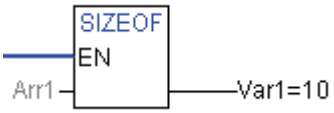
➢ **功能：**取得数据类型所需字节数。

✧ **指令使用举例**

变量定义

VAR | VAR_INPUT | VAR_OUTPUT | VAR_IN_OUT | CONSTANT

	名称	地址	类型	初始值	注释
0001	Arr1		ARRAY[0..4] OF INT		
0002	Var1		INT		

编程语言	程　序
梯形图（LD）	
结构化文本（ST）	Var1:=SIZEOF(arr1);　（*结果 Var1 为 10*）
指令列表（IL）	LD　　arr1 SIZEOF ST　　Var1　　（*结果 Var1 为 10*）
功能块（FBD）	

4.10 调用指令

CAL—调用指令

➢ **功能：**调用功能块或者程序。在 IL 语言中使用 CAL 运算来调用功能块或者程序。被调用功能块/程序的输入变量位于该功能块/程序名称右侧的括号内。

✧ **指令使用举例**

在程序中调用名称为 Inst 的功能块，其输入变量 Par1 等于 0、 Par2 等于 TRUE。IL 中调用如下：

CAL Inst(Par1:=0, Par2:=TRUE)

4.11 初始化操作指令

INI—初始化操作指令

➢ **功能：**用于初始化在程序中使用的功能块程序的内部保持型变量。

✧ **指令使用举例**

语法: <bool-Variable> := INI(<FB-instance, TRUE|FALSE)

在程序中调用名称为 FB-instance 的功能块，其输入变量分别为 Par1=FB-instance、Par2=TRUE|FALSE，输出为初始化完成标志。

主程序 PLC_PRG 变量定义：

VAR | VAR_INPUT | VAR_OUTPUT | VAR_IN_OUT | CONSTANT

	名称	地址	类型	初始值	注释
0001	VarBOOL2		BOOL		
0002	VarBOOL1		BOOL		
0003	funb1		funb		

funb（FB）变量定义：

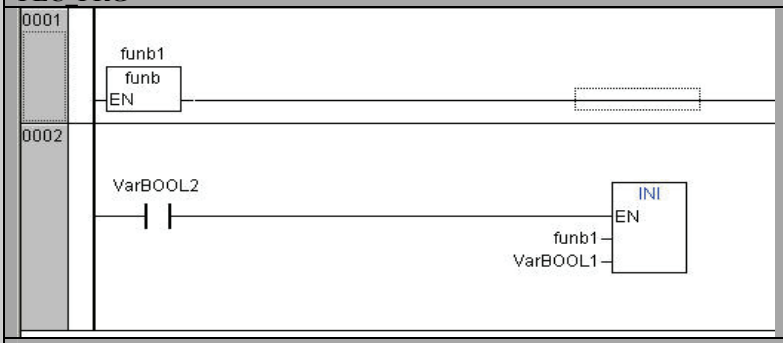
VAR | VAR_INPUT | VAR_OUTPUT | VAR_IN_OUT | CONSTANT | RETAIN

	名称	地址	类型	初始值	注释
0001	Rtrig		R_TRIG		
0002	var1		INT	1	
0003	var2		INT	2	
0004	var3		INT	3	

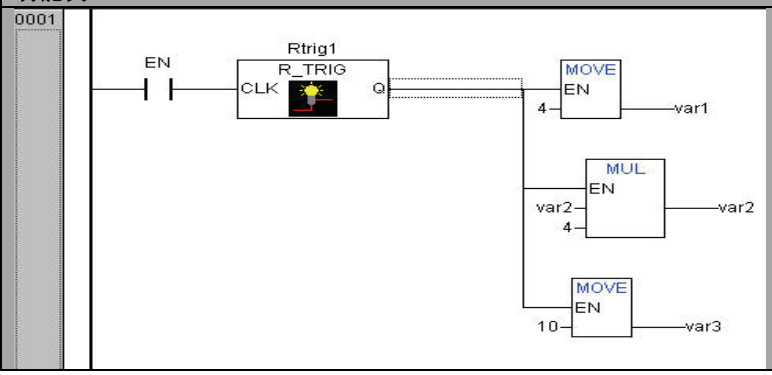
编程语言：梯形图（LD）

程　序

PLC_PRG



功能块 funb



程序说明：

- 程序运行第一个扫描周期将变量 var1、var2、var3 的值改为 4、8、10，强制变量 VarBOOL2 接通，INI 指令执行，将三个保持型变量恢复为初始值 1、2、3。

4.12 字符串处理指令（Standard.lib）

4.12.1 LEN——取字符串长度指令

- **功能：**计算字符串的长度。
- **输入/输出数据类型：**输入是 STRING 类型，输出是 INT 类型。

✧ 指令使用举例

变量定义	
VAR / VAR_INPUT / VAR_OUTPUT / VAR_IN_OUT / CON 名称 地址 类型 初始值 注释 0001 VarINT1 INT	
编程语言	程 序
梯形图（LD）	LEN EN 'Hollysys'–STR –VarINT1=8
结构化文本（ST）	Var1INT1 := LEN (‘Hollysys’); (*结果 Var1INT1 为 8*)
指令列表（IL）	LD ‘Hollysys’ LEN ST VarINT1 (*结果 Var1INT1 为 8*)
功能块（FBD）	LEN 'Hollysys'–STR –VarINT1=8

4.12.2 LEFT——左边取字符串指令

- **功能：**从字符串左边取字符串。
- **指令格式：**LEFT（STR,SIZE），其中输入 STR 是 STRING 类型，为输入字符串，SIZE 是 INT 型，为从输入字符串左边开始获取的字符个数，输出数据类型是 STRING 型。

✧ 指令使用举例

变量定义	
VAR / VAR_INPUT / VAR_OUTPUT / VAR_IN_OUT / CONS 名称 地址 类型 初始值 注释 0001 VarSTRING STRING	
编程语言	程 序
梯形图（LD）	LEFT EN 'Hollysys'–STR 3–SIZE –VarSTRING='Hol'
结构化文本（ST）	VarSTRING := LEFT (‘Hollysys’,3); (*结果为' Hol '*)
指令列表（IL）	LD ' Hollysys ' LEFT 3 ST VarSTRING (*结果为' Hol '*)
功能块（FBD）	LEFT 'Hollysys'–STR 3–SIZE –VarSTRING='Hol'

4.12.3 RIGHT——右边取字符串指令

- **功能：**从字符串右边取字符串。
- **指令格式：**RIGHT（STR,SIZE），其中输入 STR 是 STRING 类型，为输入字符串，SIZE 是 INT 型，为从字符串右边开始获取的字符个数，输出数据类型是 STRING 型。

✧ 指令使用举例

变量定义					
	VAR / VAR_INPUT / VAR_OUTPUT / VAR_IN_OUT / CON				
	名称	地址	类型	初始值	注释
	0001 VarSTRING		STRING		

编程语言	程 序
梯形图（LD）	RIGHT EN 'Hollysys'—STR 3—SIZE —VarSTRING='sys'
结构化文本（ST）	VarSTRING := RIGHT(‘Hollysys’,3); (*结果为' sys '*)
指令列表（IL）	LD ' Hollysys ' RIGHT 3 ST VarSTRING (*结果为' sys '*)
功能块（FBD）	RIGHT 'Hollysys'—STR 3—SIZE —VarSTRING='sys'

4.12.4 MID——中间取字符串指令

- **功能：**从字符串中间取字符串。
- **指令格式：**MID(STR,LEN,POS)，其中输入 STR 是 STRING 类型，为输入字符串。LEN 和 POS 是 INT 型，该指令从 POS 开始从左往右获取 LEN 个字符，输出数据类型是 STRING 型。

✧ 指令使用举例

变量定义					
	VAR / VAR_INPUT / VAR_OUTPUT / VAR_IN_OUT / CON				
	名称	地址	类型	初始值	注释
	0001 VarSTRING		STRING		

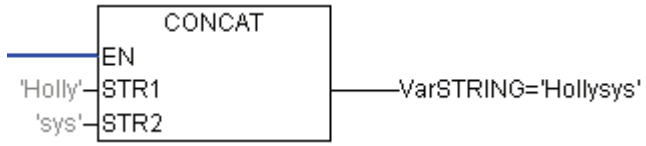
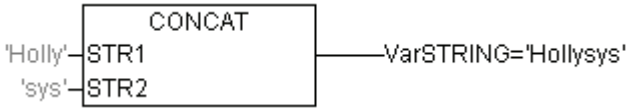
编程语言	程 序
梯形图（LD）	MID EN 'Hollysys'—STR 2—LEN 4—POS —VarSTRING='ly'
结构化文本（ST）	VarSTRING:= MID(‘Hollysys’,2,4); (*结果为‘ly’ *)
指令列表（IL）	LD ‘Hollysys’ RIGHT 2,4 ST VarSTRING (*结果为‘ly’ *)
功能块（FBD）	MID 'Hollysys'—STR 2—LEN 4—POS —VarSTRING='ly'

4.12.5 CONCAT——合并字符串指令

- **功能：**把两个字符串按前后顺序结合成一个字符串，输入和输出都是 STRING 型。

✧ **指令使用举例**

变量定义

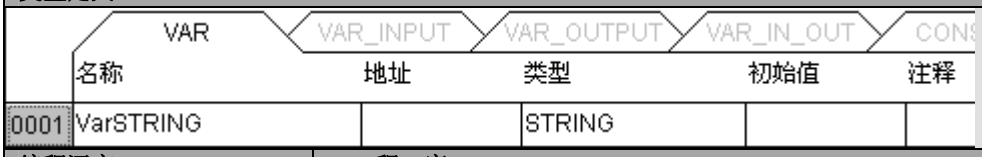
编程语言	程　序
梯形图（LD）	
结构化文本（ST）	VarSTRING := CONCAT ('Holly', 'sys'); (*结果为'Hollysys'*)
指令列表（IL）	LD 'Holly' CONCAT ‘sys’ ST VarSTRING (*结果为'Hollysys'*)
功能块（FBD）	

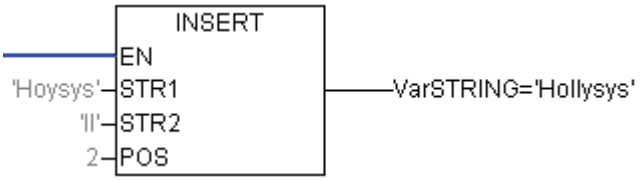
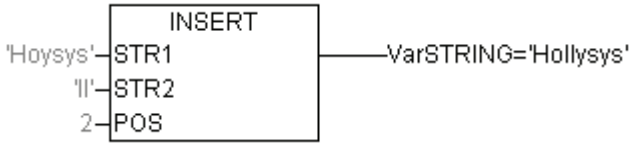
4.12.6 INSERT——插入字符串指令

- **功能：**把一个字符串插入到另一个字符串中。
- **指令格式：**INSERT(STR1,STR2,POS)。输入 STR1 和 STR2 是 STRING 类型，POS 是 INT 型，该指令把 STR2 插入到 STR1 的 POS 位置之后。输出数据类型是 STRING 型。

✧ **指令使用举例**

变量定义



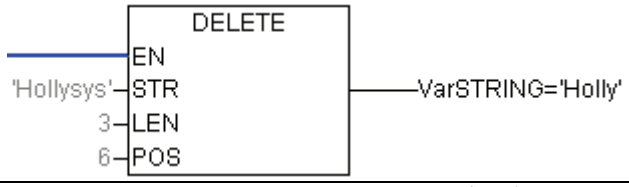
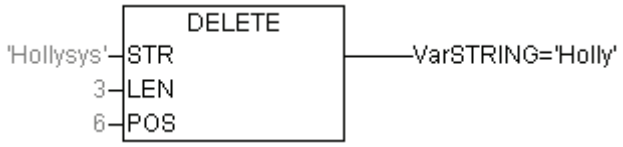
编程语言	程　序
梯形图（LD）	
结构化文本（ST）	VarSTRING := INSERT ('Hoysys', ‘ll’,2); (*结果为 'Hollysys'*)
指令列表（IL）	LD 'Hoysys' INSERT ‘ll’,2 ST VarSTRING (*结果为 'Hollysys'*)
功能块（FBD）	

4.12.7 DELETE——删除字符指令

- **功能：**从字符串中删除字符。
- **指令格式：**DELETE(STR,LEN,POS)。输入 STR 是 STRING 类型，为输入字符串。LEN 和 POS 是 INT 型，该指令从输入字符的位置 POS 处开始从左往右删除 LEN 个字符，输出数据类型是 STRING 型。

✧ **指令使用举例**

变量定义

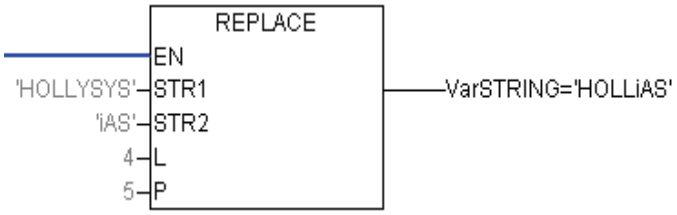
编程语言	程　序
梯形图（LD）	
结构化文本（ST）	VarSTRING := DELETE ('Hollysys',3,6);　(*结果为'Holly'*)
指令列表（IL）	LD　'Hollysys' DELETE　3,6 ST　VarSTRING　(*结果为'Holly'*)
功能块（FBD）	

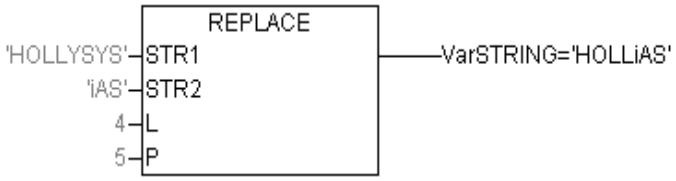
4.12.8 REPLACE——替换字符串指令

- **功能：**用一个字符串替代另一字符串中的部分内容。
- **指令格式：**REPLACE(STR1,STR2,L,P)。输入 STR1 和 STR2 是 STRING 类型，为输入字符串。L 和 P 是 INT 型，该指令用 STR2 代替 STR1 中从 P 位置开始的 L 个字符。输出数据类型是 STRING 型。

✧ **指令使用举例**

变量定义

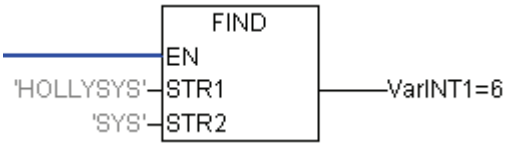
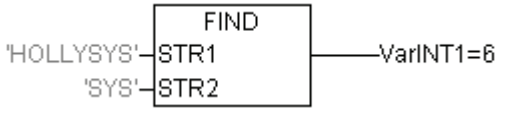
编程语言	程　序
梯形图（LD）	

结构化文本（ST）	VarSTRING:= REPLACE ('HOLLYSYS', 'iAS',4,5); (*结果为'HOLLiAS'*)
指令列表（IL）	LD 'HOLLYSYS' REPLACE 'iAS', 4,5 ST VarSTRING (*结果为'HOLLiAS'*)
功能块（FBD）	

4.12.9 FIND——查找字符串指令

- **功能**：在一个字符串中查找与另一字符串完全相同的内容。
- **指令格式**：FIND(STR1,STR2)。输入 STR1 和 STR2 都是 STRING 类型，为输入字符串，返回数据类型为 INT 型。指令功能为在第一个字符串 STR1 中查找与字符串 STR2 完全相同的部分，返回该相同部分在字符串 STR1 的起始位置。若没有完全相同的部分，输出结果为 0。

✧ **指令使用举例**

变量定义	
编程语言	**程 序**
梯形图（LD）	
结构化文本（ST）	VarINT1 := FIND ('HOLLYSYS', 'SYS'); (*结果为 6*)
指令列表（IL）	LD 'HOLLYSYS' FIND 'SYS' ST VarINT1 (*结果为 6*)
功能块（FBD）	

4.13 库版本信息检查指令（Util.lib）

Version_Util——库版本信息检查指令

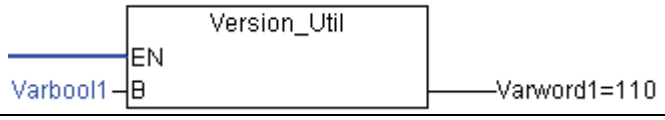
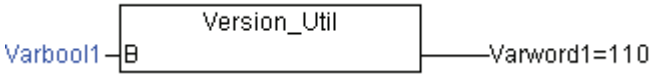
- **功能**：读取当前软件的库版本信息码。
- **输入/输出数据类型**：输入类型：BOOL； 输出类型：WORD；

✧ 指令使用举例

变量定义

VAR | VAR_INPUT | VAR_OUTPUT | VAR_IN_OUT | CONSTANT

	名称	地址	类型	初始值	注释
0001	Varbool1		BOOL		
0002	Varword1		WORD		

编程语言	程 序
梯形图（LD）	
结构化文本（ST）	Varword1:=Version_Util(Varbool1);
指令列表（IL）	LD Varbool1 Version_Util ST Varword1
功能块（FBD）	

4.14 软件版本信息指令（SysLibC16x.lib）

SyslibGetVersion2300——PLC 版本信息检查指令

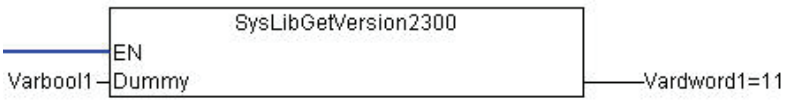
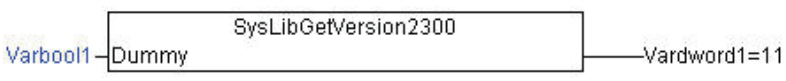
- **功能：**读取当前版本信息码。
- **输入/输出数据类型：**输入类型：BOOL； 输出类型：DWORD；

✧ 指令使用举例

变量定义

VAR | VAR_INPUT | VAR_OUTPUT | VAR_IN_OUT | CONSTANT

	名称	地址	类型	初始值	注释
0001	Varbool1		BOOL		
0002	Vardword1		DWORD		

编程语言	程 序
梯形图（LD）	
结构化文本（ST）	Vardword1:= SysLibGetVersion2300 (Varbool1);
指令列表（IL）	LD Varbool1 SysLibGetVersion2300 ST Vardword1
功能块（FBD）	

4.15 检查指令（Check.lib）

4.15.1 CheckBounds——数组边界检查指令

- **功能：**数组边界检查，若超出数组边界值，则程序按边界值进行计算。
- **输入/输出数据类型：**

输入类型：Index, lower, upper: DINT；输出类型：CheckBounds : DINT

✧ **指令使用举例**

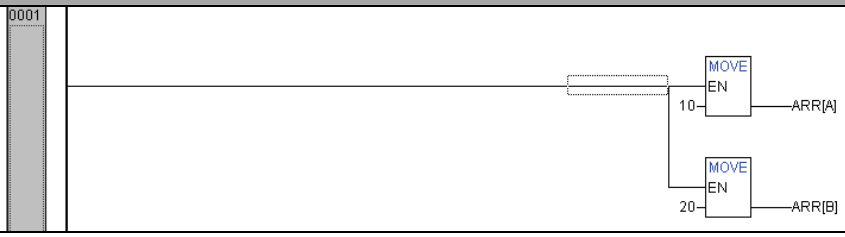
变量定义

VAR | VAR_INPUT | VAR_OUTPUT | VAR_IN_OUT | CONSTANT

	名称	地址	类型	初始值	注释
0001	Arr		ARRAY[1..7] OF BYT		
0002	A		INT	-2	
0003	B		INT	10	

编程语言 | **程 序**

梯形图（LD）



结构化文本（ST）

```
ARR[A]:=10;
ARR[B]:=20;
```

指令列表（IL）

```
LD    10
ST    Arr[A]
LD    20
ST    Arr[B]
```

功能块（FBD）

程序说明：

只要添加了 Check.lib 库，不需要调用 CheckBounds 指令，就可以检测数组边界，程序中 A 和 B 初始值分别为-2 和 10，超过 1 至 7 范围，所以 Arr[A]和 Arr[B]自动对应数组 Arr[1]和 Arr[7]，也就是 10 赋值给 Arr[1]而 20 赋值给 Arr[7]。

4.15.2 CheckDivByte——字节型除数为零检查指令

- **功能：**用于字节型除法运算中的除数为零检查，当除数为零时，按照除数等于 1 进行运算。
- **输入/输出数据类型：**

输入类型：divisor: BYTE；输出类型：CheckDivByte: BYTE。

✧ **指令使用举例**

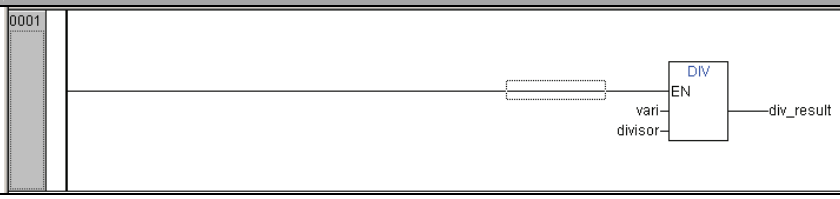
变量定义

VAR | VAR_INPUT | VAR_OUTPUT | VAR_IN_OUT | CONSTANT

	名称	地址	类型	初始值	注释
0001	divisor		BYTE	0	
0002	vari		BYTE	100	
0003	div_result		BYTE	10	

编程语言 | **程　序**

梯形图（LD）



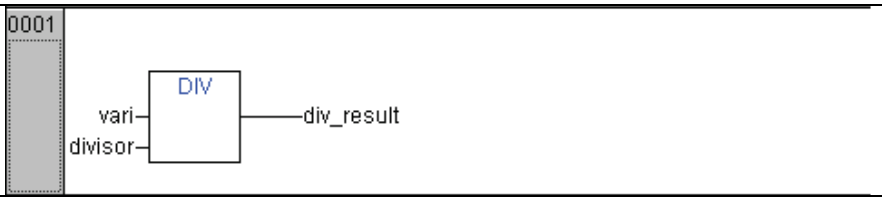
结构化文本（ST）

```
Div_result:=vari/divsor;
```

指令列表（IL）

```
LD      vari
DIV     divisor
ST      div_result
```

功能块（FBD）



程序说明：

只要添加了 Check.lib 库，不需要调用 CheckDivByte 指令，就可以检测 0 除数，程序中 divisor 为 0，当 vari 除以 divisor 时就相当于除以 1。

4.15.3 CheckDivWord——字型除数为零检查指令

- **功能：**用于字型除法运算中的除数为零检查，当除数为零时，按照除数等于 1 进行运算。
- **输入/输出数据类型：**

输入类型：divisor: WORD；输出类型：CheckDivByte: WORD。

✧ **指令使用举例**

见“4.15.2 CheckDivByte”指令使用举例，只是把变量类型改为 WORD 型。

4.15.4 CheckDivDWord——双字型除数为零检查指令

- **功能：**

用于双字型除法运算中的除数为零检查，当除数为零时，按照除数等于 1 进行运算。

- **输入/输出数据类型：**

输入类型：divisor: DWORD；输出类型：CheckDivByte: DWORD。

✧ **指令使用举例**

见“4.15.2 CheckDivByte”指令使用举例，只是把变量类型改为 DWORD 型。

4.15.5 CheckDivReal——实型除数为零检查指令

- **功能：**用于实数型除法运算中的除数为零检查，当除数为零时，按照除数等于 1 进行运算。
- **输入/输出数据类型：**

输入类型：divisor: REAL；输出类型：CheckDivByte: REAL。

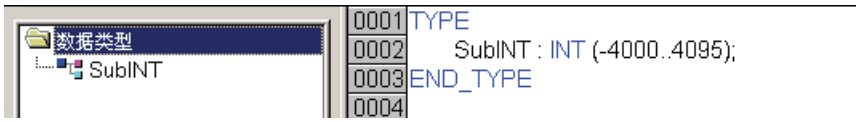
✧ **指令使用举例**

见 4.15.2CheckDivByte 指令使用举例，只是把变量类型改为 REAL 型。

4.15.6 CheckRangeSigned——整型边界检查指令

- **功能：**用于检查整型的 subrange 类型变量的范围越界。如果所赋值小于 subrange 类型变量的下边界，则该变量值为所定义的边界最小值；如果所赋值大于 subrange 类型变量的上边界，则该变量值为所定义的边界最大值。
- **Subrange 类型：**是某种基本数据类型取值范围的子集，它可以在数据类型中定义见下图，也可以在程序的变量定义中直接定义见指令使用举例中变量定义。

✧ **在数据类型中声明 Subrange 类型**

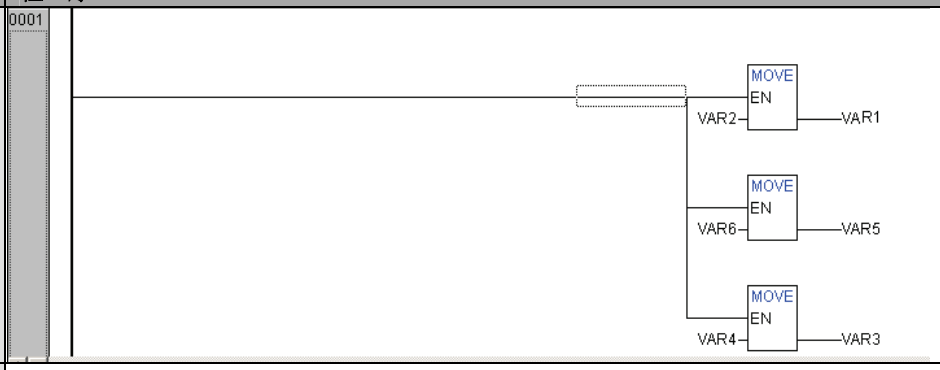
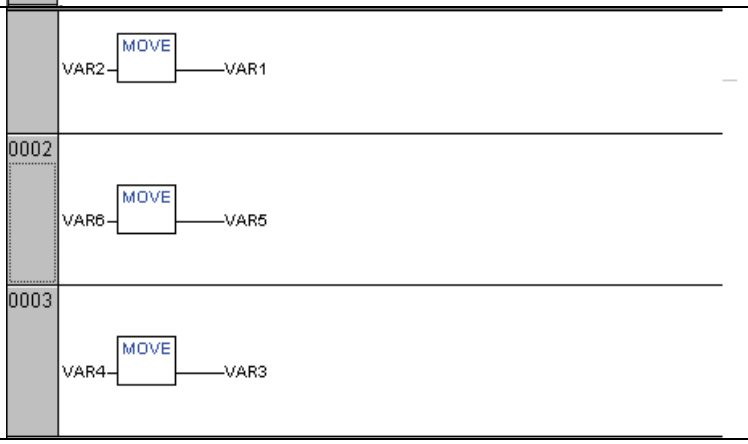


✧ **指令使用举例**

变量定义

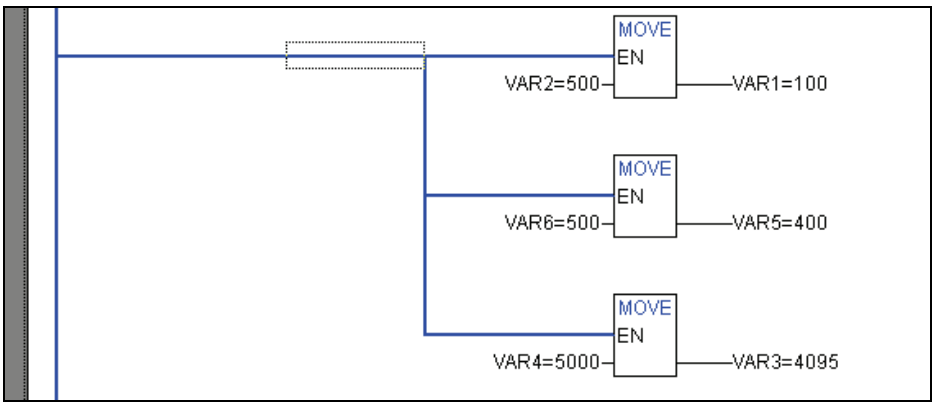
VAR | VAR_INPUT | VAR_OUTPUT | VAR_IN_OUT | CONSTANT | R

	名称	地址	类型	初始值	注释
0001	VAR1		INT(-100..100)		
0002	VAR2		INT	500	
0003	VAR3		subint		
0004	VAR4		INT	5000	
0005	VAR5		WORD(100..400)		
0006	VAR6		WORD	500	

编程语言	程　序
梯形图（LD）	
结构化文本（ST）	VAR1:=VAR2; VAR5:=VAR6; VAR3:=VAR4
指令列表（IL）	0001 LD VAR2 0002 ST VAR1 0003 LD VAR6 0004 ST VAR5 0005 LD VAR4 0006 ST VAR3
功能块（FBD）	

程序说明：

只要添加了 Check.lib 库，不需要调用 CheckRangeSigned 指令，在程序中定义了变量 VAR3 为 SubINT 类型，SubINT 类型的取值范围为-4000~4095，在程序中将 VAR4=5000， i 赋值给 VAR3，由于 VAR4 的值 5000 已经超过了 4095 上限，所以运行后，VAR3 的值取上边界值。同样道理 VAR1 和 VAR5 的值也是取上边界值，运行结果如下：



4.15.7 CheckRangeUnsigned——无符号整型边界检查指令

- **功能**：用于检查无符号整型的subrange类型变量的范围越界。如果所赋值小于subrange类型变量的下边界，则该变量值为所定义的边界最小值；如果所赋值大于 subrange类型变量的上边界，则该变量值为所定义的边界最大值。

✧ **指令使用举例**

参照 4.15.6CheckRangesigned 指令使用举例，只是把变量类型定义成无符号整型。

4.16 BCD 码转换指令（Util.lib）

BCD 码的一个字节包含 0 到 99 之间的整数。每个十进制位对应 4 位，十位数存储在 4-7 位，个位数存储在 0-3 位。BCD 码格式和 16 进制表达方式很相似，差别在于 BCD 字节值是 0-99，而 16 进制是 0-FF。

BCD 码用 4 位二进制数表示一个十进制数位，整个十进制数用一串 BCD 码来表示。例如，十进制数 59 表示成 BCD 码为 0101 1001，但表示成二进制为 2#111011。十进制 51 转换成 BCD 码，5 的二进制是 0101，1 的二进制是 0001，那么 51 转换 BCD 码为 0101 0001。

4.16.1 BCD_TO_INT——BCD 码转整型指令

- **功能：**该指令将 BCD 码转为 INT 值。
- **输入/输出数据类型：**

输入 B：BYTE 型，输入 BCD 码的二进制形式（或者该二进制对应的十进制和十六进制）。比如 BCD 码 49，表示为 2#100 1001（或者对应 10#73、16#49），则此处输入 2#100 1001（或者 10#73，16#49）；

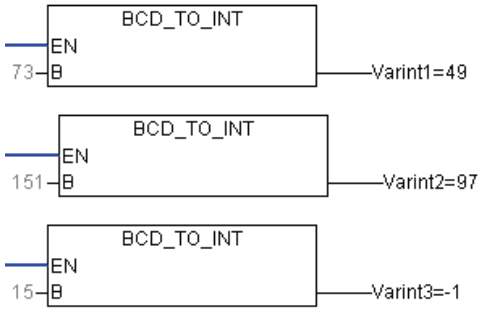
输出：INT 型，该 BCD 码所代表的实际值，如果输入的字节不是 BCD 码，输出是-1。

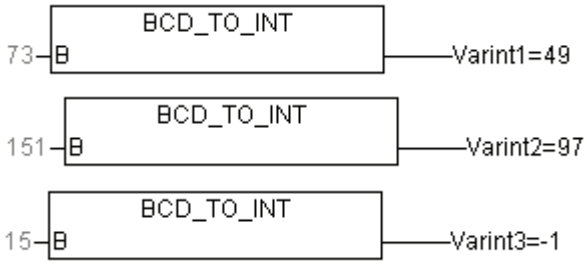
✧ **指令使用举例**

变量定义

VAR | VAR_INPUT | VAR_OUTPUT | VAR_IN_OUT | CONSTANT

名称	地址	类型	初始值	注释
0001 Varint1		INT		
0002 Varint2		INT		
0003 Varint3		INT		

编程语言	程　序
梯形图（LD）	

结构化文本（ST）	Varint1:=BCD_TO_INT(73); (*结果为 49 *) Varint2:=BCD_TO_INT(151); (*结果为 97*) Varint3:=BCD_TO_INT(15); (*输出-1，因为不是 BCD 码格式*)
指令列表（IL）	LD 73 BCD_TO_INT ST Varint1 (*结果为 49 *)
	LD 151 BCD_TO_INT ST Varint2 (*结果为 97*)
	LD 15 BCD_TO_INT ST Varint3 (*输出-1，因为不是 BCD 码格式*)
功能块（FBD）	

4.16.2 INT_TO_BCD——整型转 BCD 码指令

- **功能：**将整数值转换成 BCD 码，当整数值不能转换成 BCD 码字节时，输出数值 255。
- **输入/输出数据类型：**

输入 I：INT 型，如果整数值为 49，则此处输入整型数据 49。

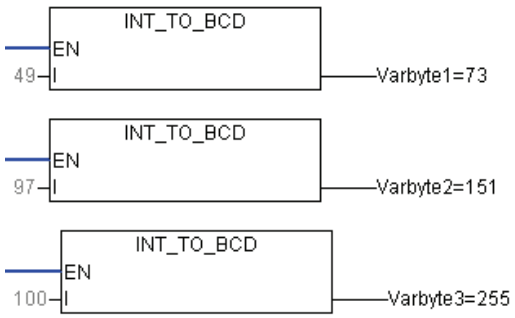
输出：BYTE 型，转换完的 BCD 码的值，比如将 49 转换为 BCD 码为 2#100 1001，则此处输出 2#100 1001（或者该二进制对应的 10#73、16#49）。

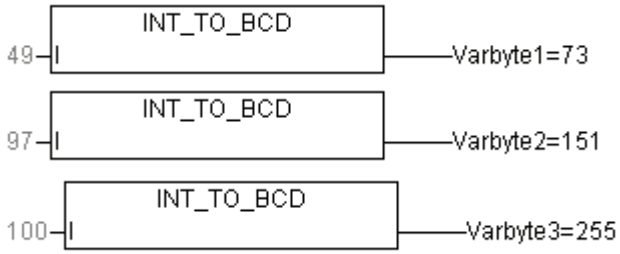
✧ **指令使用举例**

变量定义

VAR | VAR_INPUT | VAR_OUTPUT | VAR_IN_OUT | CONSTANT

	名称	地址	类型	初始值	注释
0001	Varbyte1		BYTE		
0002	Varbyte2		BYTE		
0003	Varbyte3		BYTE		

编程语言	**程 序**
梯形图（LD）	

结构化文本（ST）	Varbyte1:=INT_TO_BCD(49); (*结果为 73*) Varbyte2:= INT_TO_BCD(97); (*结果为 151*) Varbyte3:= INT_TO_BCD(100); (*错误！输出：255*)
指令列表（IL）	LD 49 INT_TO_BCD ST Varbyte1 (*结果为 73*)
	LD 97 INT_TO_BCD ST Varbyte2 (*结果为 151*)
	LD 100 INT_TO_BCD ST Varbyte3 (*错误！输出：255*)
功能块（FBD）	

4.17 位/字节操作指令（Util.lib）

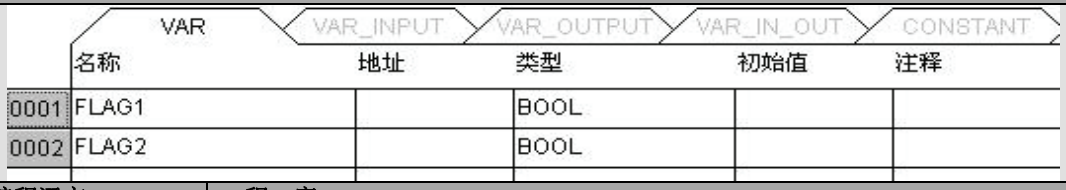
4.17.1 EXTRACT——位提取指令

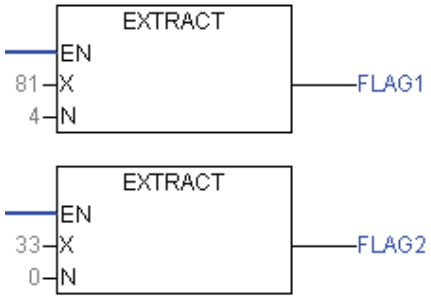
- **功能：**提取输入变量 X 二进制数的第 N 位（N=0,1...）并输出该位数值。
- **输入/输出数据类型：**

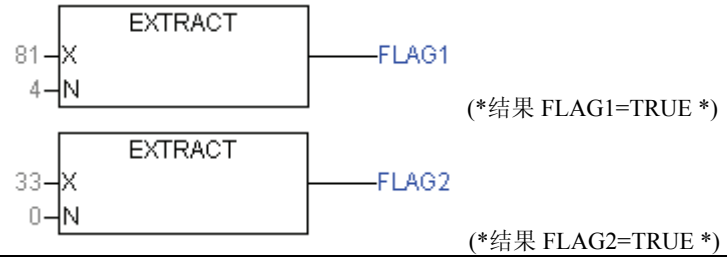
输入变量 X 是 DWORD 类型，N 是 BYTE 型；输出变量是 BOOL 类型。

✧ **指令使用举例**

变量定义



编程语言	**程 序**
梯形图（LD）	 (*结果 FLAG1=TRUE，FLAG2=TRUE *)

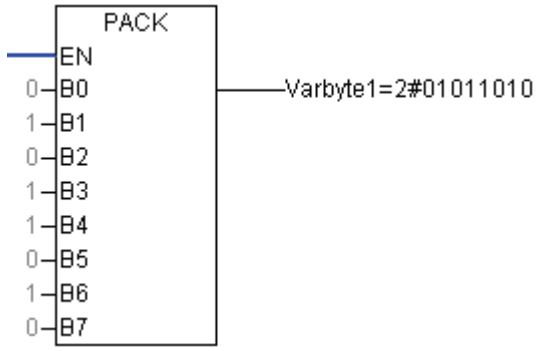
结构化文本（ST）	FLAG1:=EXTRACT(X:=81, N:=4); (*结果：TRUE，因为 81 的二进制数是 1010001，所以第四位是 1*) FLAG2:=EXTRACT(X:=33, N:=0); (*结果：TRUE，因为 33 的二进制数是 100001，所以第 0 位是 1*)
指令列表（IL）	LD 81 EXTRACT 4 ST FLAG1 (*结果：TRUE，因为 81 的二进制数是 1010001，所以第四位是 1*)
	LD 33 EXTRACT 0 ST FLAG2 (*结果：TRUE，因为 33 的二进制数是 100001，所以第 0 位是 1*)
梯形图（LD）	

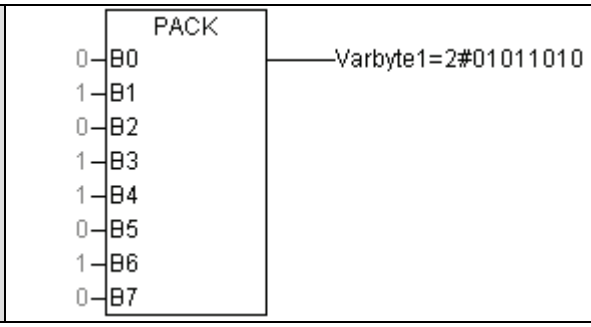
4.17.2 PACK——位整合指令

- **功能：**把输入位 B0、B1、……、B7 合成为一个字节，与这个指令相对应的指令是 UNPACK。
- **输入/输出数据类型：**

输入 B0、B1、……、B7 均为 BOOL 类型；输出数据为 BYTE 型。

✧ **指令使用举例**

变量定义					
编程语言	**程　序**				
梯形图（LD）					
结构化文本（ST）	Varbyte1:= PACK(0,1,0,1,1,0,1,0); (*结果为 2#01011010*)				
指令列表（IL）	LD FALSE PACK TRUE,FALSE,TRUE,TRUE,FALSE,TRUE,FALSE ST Varbyte1 (*结果为 2#01011010*)				

功能块（FBD）	

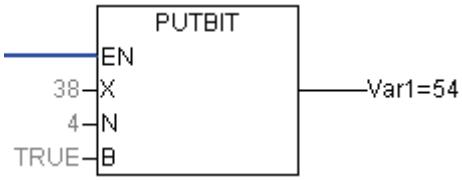
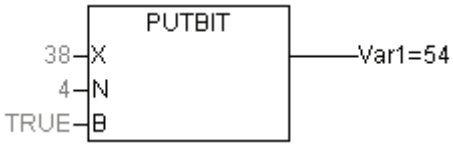
4.17.3 PUTBIT——位赋值指令

- **功能：**将输入变量 X 值的第 N 位（N=0,1...）赋值为 B，并输出 X 转变后的值。
- **输入/输出数据类型：**
  输入变量 X 为 DWORD 型、N 为 BYTE 型和 B 为 BOOL 型；输出为 DWORD 型。

✧ **指令使用举例**

变量定义

	VAR	VAR_INPUT	VAR_OUTPUT	VAR_IN_OUT	CONSTANT
	名称	地址	类型	初始值	注释
0001	Var1		DWORD		
0002	Var2		DWORD		

编程语言	程　序
梯形图（LD）	
结构化文本（ST）	Var2:=38;　　(*二进制 100110*) Var1:=PUTBIT(Var2,4,TRUE);　(*结果：54 = 2#110110*)
指令列表（IL）	LD　38　　(*二进制 100110*) PUTBIT　4,TRUE ST　Var1　　(*结果：54 = 2#110110*)
功能块（FBD）	

4.17.4 UNPACK——位拆分

- **功能：**将字节型的输入 B 拆分转换成 8 个 BOOL 类型的输出变量 B0,…,B7，与 PACK 指令相反。
- **输入/输出变量：**
  输入数据为 BYTE 型；输出 B0、B1、……、B7 均为 BOOL 类型。

✧ 指令使用举例

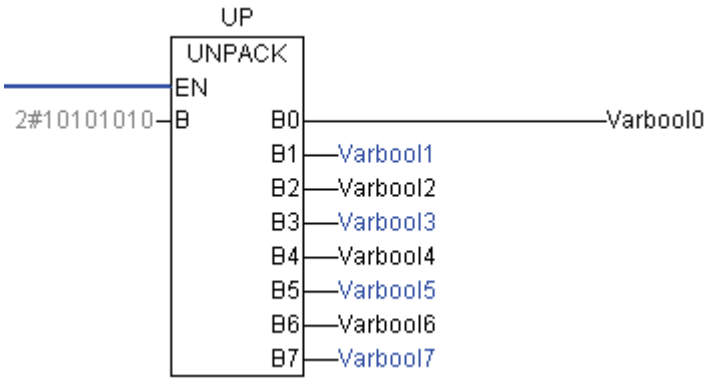
变量定义

VAR | VAR_INPUT | VAR_OUTPUT | VAR_IN_OUT | CONSTANT

	名称	地址	类型	初始值	注释
0001	UP		UNPACK		
0002	Varbool0		BOOL		
0003	Varbool1		BOOL		
0004	Varbool2		BOOL		
0005	Varbool3		BOOL		
0006	Varbool4		BOOL		
0007	Varbool5		BOOL		
0008	Varbool6		BOOL		
0009	Varbool7		BOOL		

编程语言 | **程 序**

梯形图（LD）

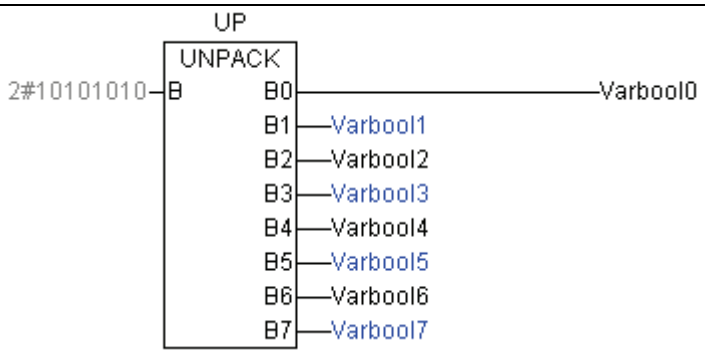


结构化文本（ST）

```
UP(B:=2#10101010);
Varbool0:=UP.B0;
Varbool1:=UP.B1;
Varbool2:=UP.B2;
Varbool3:=UP.B3;
Varbool4:=UP.B4;
Varbool5:=UP.B5;
Varbool6:=UP.B6;
Varbool7:=UP.B7;
```

指令列表（IL）

```
CAL   UP(B := 2#10101010)
LD    UP.B1
ST    Varbool1
LD    UP.B2
ST    Varbool2
LD    UP.B3
ST    Varbool3
LD    UP.B4
ST    Varbool4
LD    UP.B5
ST    Varbool5
```

	LD UP.B6 ST Varbool6 LD UP.B7 ST Varbool7 LD UP.B0 ST Varbool0
功能块（FBD）	

4.18 高等数学运算指令（Util.lib）

4.18.1 DERIVATIVE——微分

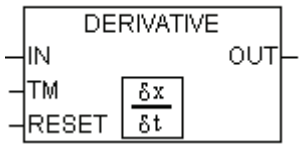
- **功能：**该指令对连续输入的变量进行微分运算。为了获得最好结果，DERIVATIVE 指令只对最新的四个输入值进行微分，减小因输入参数不精确产生的误差。

  微分迭代公式：

$$OUT = \frac{3*[IN(k) - IN(k-3)] + IN(k-1) - IN(k-2)}{3*TM(k-2) + 4*TM(k-1) + 3*TM(k)}$$

  k-3、k-2、k-1、k 为连续四次输入值的标记。

- **指令示例**

梯形图语言示例


- **参数说明**

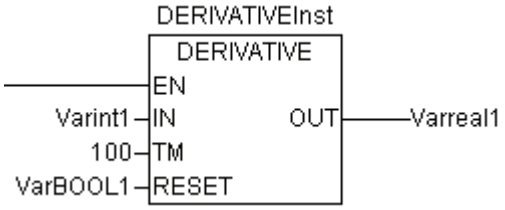
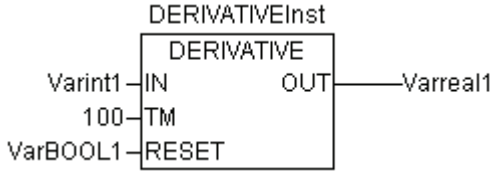
输入参数	数据类型	功能描述	参数值说明
IN	REAL	连续输入的变量	
TM	DWORD	微分时间	毫秒
RESET	BOOL	复位信号	值是 TRUE 时，重新启动指令
输出参数	**数据类型**	**功能描述**	**参数值说明**
OUT	REAL	微分结果输出	

✧ 指令使用举例

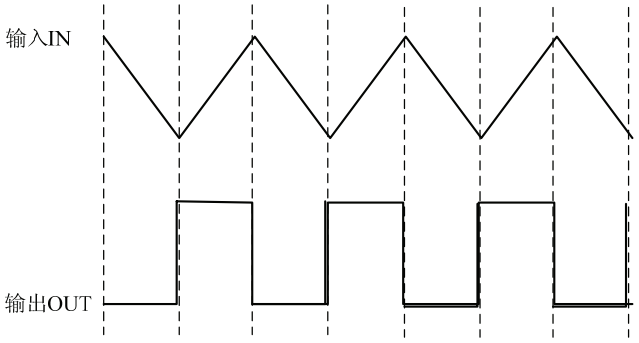
变量定义

VAR | VAR_INPUT | VAR_OUTPUT | VAR_IN_OUT | CONSTANT

	名称	地址	类型	初始值	注释
0001	DERIVATIVEInst		DERIVATIVE		
0002	Varreal1		REAL		
0003	Varint1		INT		
0004	VarBOOL1		BOOL		

编程语言	程 序
梯形图（LD）	
结构化文本（ST）	DERIVATIVEInst (IN:=Varint1,TM:=100,RESET:=VarBOOL1); Varreal1:=DERIVATIVEInst.OUT;
指令列表（IL）	CAL DERIVATIVEInst(IN := Varint1, TM := 100, RESET := VarBOOL1) LD DERIVATIVEInst.OUT ST Varreal1
功能块（FBD）	

输入输出对应关系如下图所示。



4.18.2 INTEGRAL——积分

➢ **功能：**该指令对连续输入的变量进行积分运算。

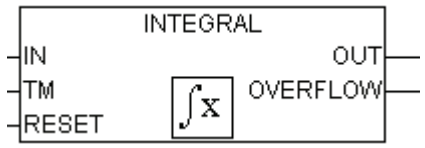
积分迭代公式： $A(k) = A(k-1) + TM * IN(k-1)$

$$B(k) = B(k-1) + TM * IN(k)$$

$$OUT(k) = \frac{A(k) + B(k)}{2}$$

k-1 、k 为连续两次输入值的标记。

➢ 指令示例

梯形图语言示例


➢ 参数说明

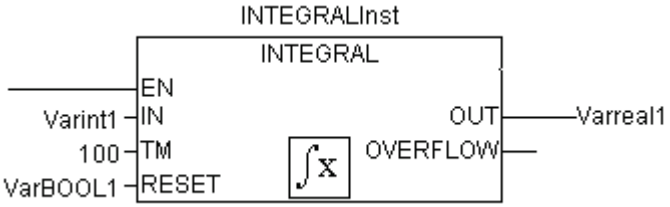
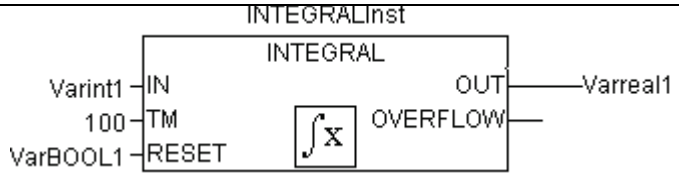
输入参数	数据类型	功能描述	参数值说明
IN	REAL	连续输入的变量	
TM	DWORD	积分时间	
RESET	BOOL	复位信号	其值是 TRUE 时，重新启动指令
输出参数	**数据类型**	**功能描述**	**参数值说明**
OUT	REAL	积分结果输出	
OVERFLOW	BOOL	溢出标志	其值是 TRUE 时，说明计算溢出

✧ 指令使用举例

变量定义

VAR | VAR_INPUT | VAR_OUTPUT | VAR_IN_OUT | CONSTANT

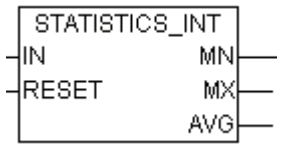
	名称	地址	类型	初始值	注释
0001	INTEGRALInst		INTEGRAL		
0002	Varreal1		REAL		
0003	Varint1		REAL		
0004	VarBOOL1		BOOL		

编程语言	程　序
梯形图（LD）	
结构化文本（ST）	INTEGRALInst(IN := Varint1, TM := 100, RESET := VarBOOL1); Varreal1:=INTEGRALInst.OUT;
指令列表（IL）	CAL INTEGRALInst(IN := Varint1, TM := 100, RESET := VarBOOL1) LD　　INTEGRALInst.OUT ST　　Varreal1
功能块（FBD）	

输入输出对应关系如下图所示。

4.18.3 STATISTICS_INT——整型统计

- **功能**：统计输入整型数据最小值、最大值和平均值的计算。
- **指令示例**

梯形图语言示例


- **参数说明**

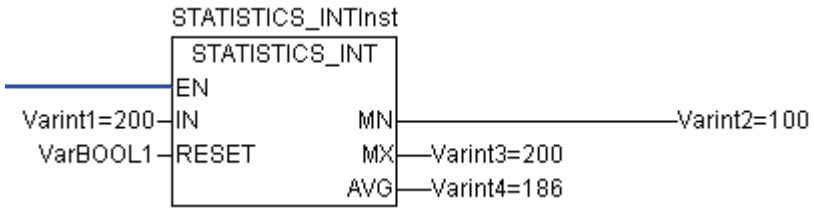
输入参数	数据类型	功能描述	参数值说明
IN	INT	输入	
RESET	BOOL	初始化	其值是 TRUE 时，重新初始化
输出参数	**数据类型**	**功能描述**	**参数值说明**
MN	INT	最小值	
MX	INT	最大值	
AVG	INT	平均值	

✧ **指令使用举例**

变量定义

VAR | VAR_INPUT | VAR_OUTPUT | VAR_IN_OUT | CONSTANT

	名称	地址	类型	初始值	注释
0001	STATISTICS_INTInst		STATISTICS_INT		
0002	VarBOOL1		BOOL		
0003	Varint1		INT		
0004	Varint2		INT		
0005	Varint3		INT		
0006	Varint4		INT		

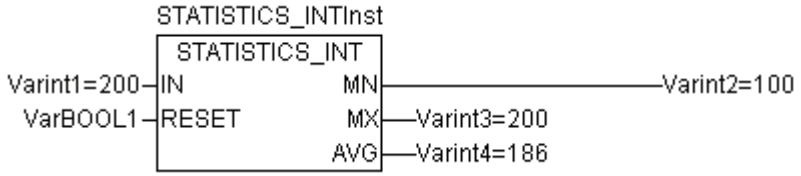
编程语言	程　序
梯形图（LD）	

结构化文本（ST）

```
STATISTICS_INTInst(IN := Varint1, RESET := VarBOOL1);
Varint3:=STATISTICS_INTInst.MX;
Varint4:=STATISTICS_INTInst.AVG;
Varint2:=STATISTICS_INTInst.MN;
```

指令列表（IL）

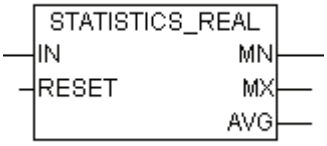
```
CAL STATISTICS_INTInst(IN := Varint1, RESET := VarBOOL1)
LD      STATISTICS_INTInst.MX
ST      Varint3
LD      STATISTICS_INTInst.AVG
ST      Varint4
LD      STATISTICS_INTInst.MN
```

	ST Varint2
功能块（FBD）	

4.18.4 STATISTICS_REAL——实型统计

- **功能：**统计输入实型数据的最小值、最大值和平均值。
- **指令示例**

梯形图语言示例



- **参数说明**

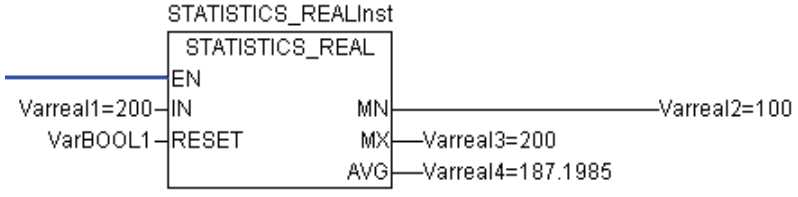
输入参数	数据类型	功能描述	参数值说明
IN	REAL	输入	
RESET	BOOL	初始化	其值是 TRUE 时，重新初始化
输出参数	**数据类型**	**功能描述**	**参数值说明**
MN	REAL	最小值	
MX	REAL	最大值	
AVG	REAL	平均值	

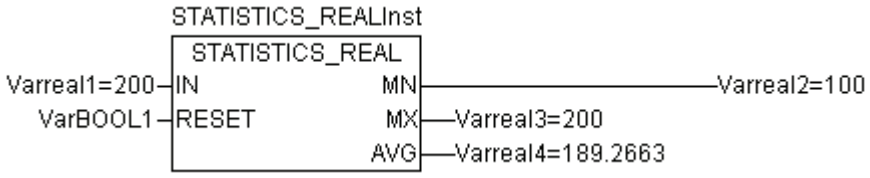
✧ **指令使用举例**

变量定义

VAR | VAR_INPUT | VAR_OUTPUT | VAR_IN_OUT | CONSTANT

	名称	地址	类型	初始值	注释
0001	STATISTICS_REALInst		STATISTICS_REAL		
0002	VarBOOL1		BOOL		
0003	Varreal1		REAL		
0004	Varreal2		REAL		
0005	Varreal3		REAL		
0006	Varreal4		REAL		

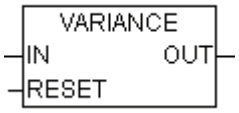
编程语言	程　序
梯形图（LD）	
结构化文本（ST）	STATISTICS_REALInst(IN :=Varreal1,RESET:=VarBOOL1); Varreal3:=STATISTICS_REALInst.MX; Varreal4:=STATISTICS_REALInst.AVG; Varreal2:=STATISTICS_REALInst.MN;

指令列表（IL）	CAL STATISTICS_REALInst(IN := Varreal1, RESET := VarBOOL1) LD STATISTICS_REALInst.MX ST Varreal3 LD STATISTICS_REALInst.AVG ST Varreal4 LD STATISTICS_REALInst.MN ST Varreal2
功能块（FBD）	

提示：
该指令与 STATISTICS_INT 功能相同，只是输入和输出数据类型为 REAL 型。

4.18.5 VARIANCE——平方偏差

- **功能：**该指令计算变量输入值的平方偏差。标准偏差可以由平方偏差的平方根得到。
- **指令示例**

梯形图语言示例


- **参数说明**

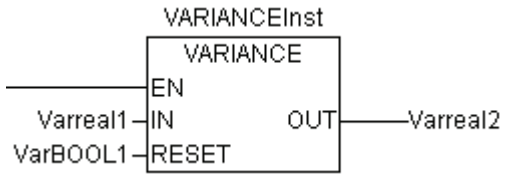
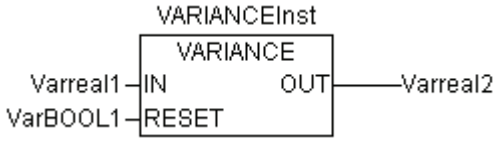
输入参数	数据类型	功能描述	参数值说明
IN	REAL	输入	
RESET	BOOL	复位	其值是 TRUE 时，指令复位
输出参数	**数据类型**	**功能描述**	**参数值说明**
OUT	REAL	平方偏差	

✧ **指令使用举例**

变量定义

VAR | VAR_INPUT | VAR_OUTPUT | VAR_IN_OUT | CONSTANT

	名称	地址	类型	初始值	注释
0001	VARIANCEInst		VARIANCE		
0002	VarBOOL1		BOOL		
0003	Varreal1		REAL		
0004	Varreal2		REAL		

编程语言	程　序
梯形图（LD）	
结构化文本（ST）	VARIANCEInst(IN := Varreal1, RESET := VarBOOL1); Varreal2:=VARIANCEInst.OUT;
指令列表（IL）	CAL VARIANCEInst(IN := Varreal1, RESET := VarBOOL1) LD　　VARIANCEInst.OUT ST　　Varreal2
功能块（FBD）	

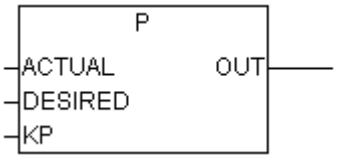
4.19 控制器指令（Util.lib）

4.19.1 P——比例控制器

- **功能：**该指令为比例控制器。
  控制方程：OUT=ACTUAL+（DESIRED-ACTUAL）*KP。
- **输入/输出数据类型：**

输入的测量值 ACTUAL、设定值 DESIRED 和比例因子 KP 都是 REAL 型。输出值 OUT 是 REAL 型。

- **指令示例**

梯形图语言示例


- **参数说明**

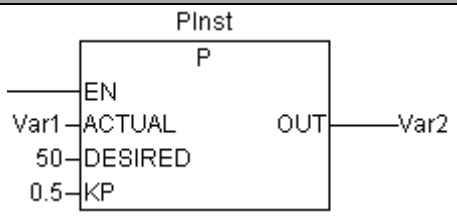
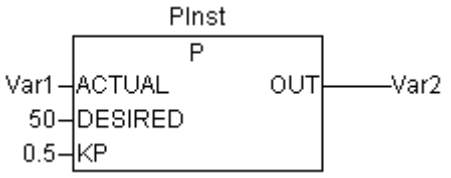
输入参数	数据类型	功能描述	参数值说明
ACTUAL	REAL	测量值	
DESIRED	REAL	设定值	
KP	REAL	比例系数	
输出参数	**数据类型**	**功能描述**	**参数值说明**
OUT	REAL	输出值	

✧ 指令使用举例

变量定义

VAR | VAR_INPUT | VAR_OUTPUT | VAR_IN_OUT | CONSTANT

	名称	地址	类型	初始值	注释
0001	PInst		P		
0002	Var1		REAL		
0003	Var2		REAL		

编程语言	程 序
梯形图（LD）	
结构化文本（ST）	PInst(ACTUAL := Var1, DESIRED := 50, KP := 0.5); Var2:=PInst.OUT;
指令列表（IL）	CAL PInst(ACTUAL := Var1, DESIRED := 50, KP := 0.5) LD PInst.OUT ST Var2
功能块（FBD）	

4.19.2 PD——比例微分控制器

➢ **功能：**该指令为比例微分控制器。

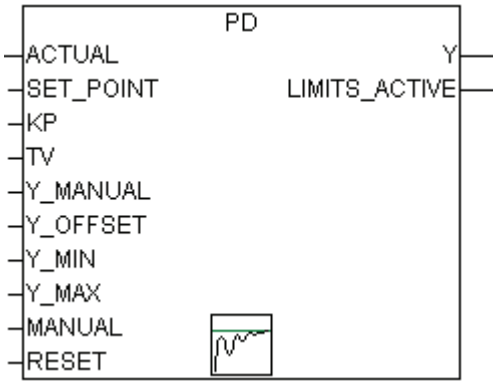
控制方程： $\Delta = \mathrm{SET}_POINT - ACTUAL$

$$Y = KP * (\Delta + TV * \frac{\sigma\Delta}{\sigma t}) + Y_OFFSET$$

该指令会自动计算 $\frac{\sigma\Delta}{\sigma \mathrm{t}}$，用户无需考虑。

➢ **指令示例**

梯形图语言示例



➢ 参数说明

输入参数	数据类型	功能描述	参数值说明
ACTUAL	REAL	测量值	
SET_POINT	REAL	设定值	
KP	REAL	比例系数	
TV	REAL	微分时间	单位为秒（s）
Y_MANUAL	REAL	手动值	MANUAL=TRUE 时，Y= Y_MANUAL
Y_OFFSET	REAL	输出值的偏移量	
Y_MIN	REAL	输出值的最小值	
Y_MAX	REAL	输出值的最大值	
MANUAL	BOOL	手自动选择	TRUE 时为手动调节，FALSE 时为自动调节
RESET	BOOL	重置	TRUE 时重置该控制器，正常运行时应置 FALSE
输出参数	**数据类型**	**功能描述**	**参数值说明**
Y	REAL	输出值	
LIMITS_ACTIVE	BOOL	输出超限标志	输出值超限时等于 TRUE，即超出（Y_MIN，Y_MAX）

✧ 指令使用举例

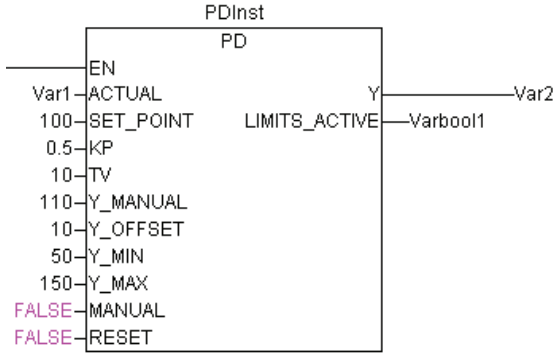
变量定义

VAR | VAR_INPUT | VAR_OUTPUT | VAR_IN_OUT | CONSTANT

	名称	地址	类型	初始值	注释
0001	PDInst		PD		
0002	Var1		REAL		
0003	Var2		REAL		
0004	Varbool1		BOOL		

编程语言 | **程　序**

梯形图（LD）

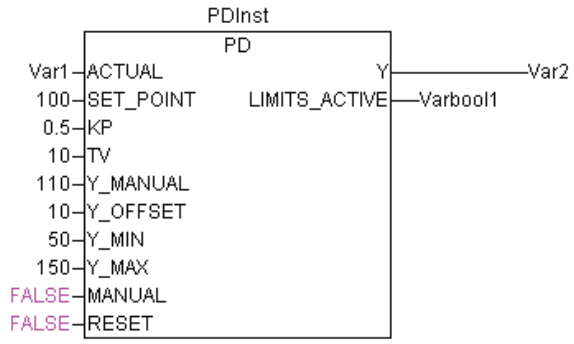


结构化文本（ST）

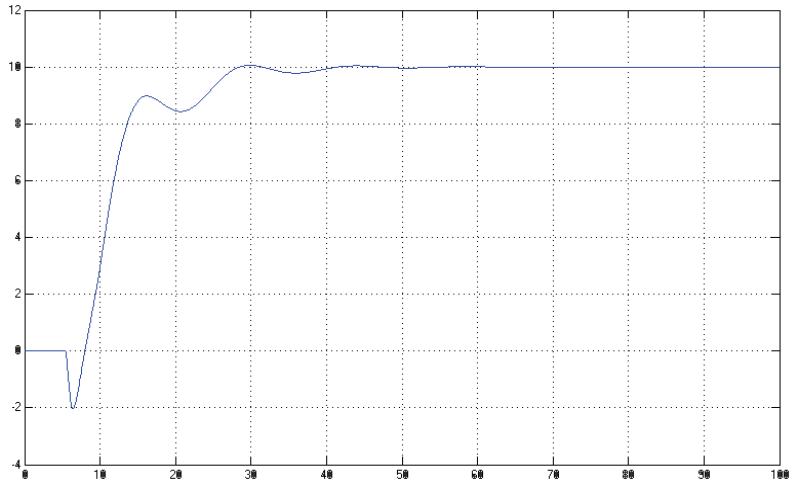
```
PDInst(ACTUAL := Var1, SET_POINT := 100, KP := 0.5, TV := 10,
Y_MANUAL := 110, Y_OFFSET := 10, Y_MIN := 50,
Y_MAX := 150, MANUAL := FALSE, RESET := FALSE);
Varbool1:=PDInst.LIMITS_ACTIVE;
Var2:=PDInst.Y;
```

指令列表（IL）

```
CAL PDInst(ACTUAL := Var1, SET_POINT := 100, KP := 0.5,
TV := 10, Y_MANUAL := 110, Y_OFFSET := 10, Y_MIN := 50,
Y_MAX := 150, MANUAL := FALSE, RESET := FALSE)
LD      PDInst.LIMITS_ACTIVE
ST      Varbool1
LD      PDInst.Y
ST      Var2
```

功能块（FBD）	

调整波形如下图所示。



4.19.3 PID——比例积分微分控制器

- **功能：**该指令为比例积分微分控制器。当 TV=0 时，PID 控制器作为 PI 控制器。

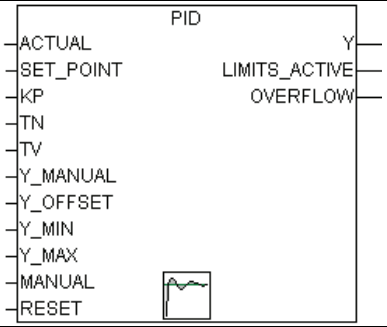
控制方程：

$$Y = KP * (\Delta + \frac{1}{TN} * \int \Delta(t)dt + TV * \frac{\sigma\Delta}{\sigma t}) + Y_OFFSET$$

$$\Delta = SET_POINT - ACTUAL$$

该指令会自动计算 $\int \Delta(t)dt$ 与 $\frac{\sigma\Delta}{\sigma t}$，用户无需考虑。

- **指令示例**

梯形图语言示例


➢ 参数说明

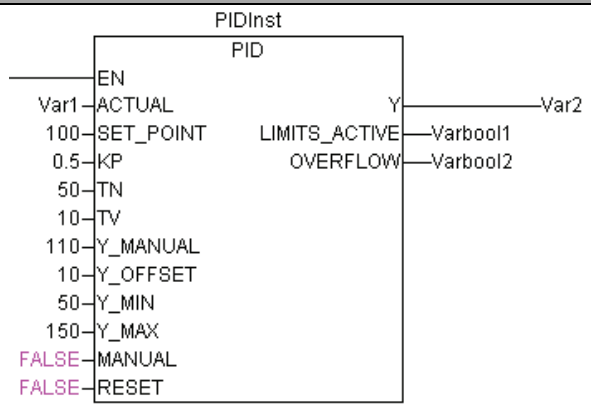
输入参数	数据类型	功能描述	参数值说明
ACTUAL	REAL	测量值	
SET_POINT	REAL	设定值	
KP	REAL	比例系数	
TN	REAL	积分时间	单位为秒（s）
TV	REAL	微分时间	单位为秒（s）
Y_MANUAL	REAL	手动值	MANUAL=TRUE 时， Y= Y_MANUAL
Y_OFFSET	REAL	输出值的偏移量	
Y_MIN	REAL	输出值的最小值	
Y_MAX	REAL	输出值的最大值	
MANUAL	BOOL	手自动选择	值为 TRUE 时为手动调节，值为 FALSE 时为自动调节
RESET	BOOL	重置	值为 TRUE 时重置该控制器，正常运行时应置 FALSE
输出参数	**数据类型**	**功能描述**	**参数值说明**
Y	REAL	输出值	
LIMITS_ACTIVE	BOOL	输出超限标志	输出值超限时等于 TRUE，即超出（Y_MIN，Y_MAX）
OVERFLOW	BOOL	输出溢出标志	积分溢出时等于 TRUE

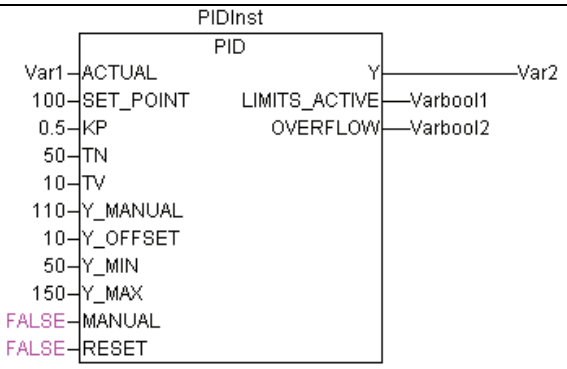
✧ 指令使用举例

变量定义

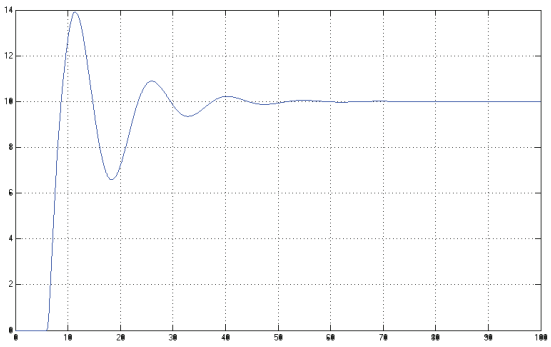
VAR | VAR_INPUT | VAR_OUTPUT | VAR_IN_OUT | CONSTANT

	名称	地址	类型	初始值	注释
0001	PIDInst		PID		
0002	Var1		REAL		
0003	Var2		REAL		
0004	Varbool1		BOOL		
0005	Varbool2		BOOL		

编程语言	程　序
梯形图（LD）	
结构化文本（ST）	PIDInst(ACTUAL := Var1, SET_POINT := 100, KP := 0.5, TN := 50, TV := 10, Y_MANUAL := 110, Y_OFFSET := 10, Y_MIN := 50, Y_MAX := 150, MANUAL := FALSE, RESET := FALSE); Varbool1:=PIDInst.LIMITS_ACTIVE; Varbool2:=PIDInst.OVERFLOW; Var2:=PIDInst.Y;
指令列表（IL）	CAL PIDInst(ACTUAL := Var1, SET_POINT := 100, KP := 0.5, TN := 50, TV := 10,

	Y_MANUAL := 110, Y_OFFSET := 10, Y_MIN := 50, Y_MAX := 150, MANUAL := FALSE, RESET := FALSE) LD PIDInst.LIMITS_ACTIVE ST Varbool1 LD PIDInst.OVERFLOW ST Varbool2 LD PIDInst.Y ST Var2
功能块（FBD）	

调整波形如下图所示。



4.19.4 PID_FIXCYCLE——比例积分微分控制器

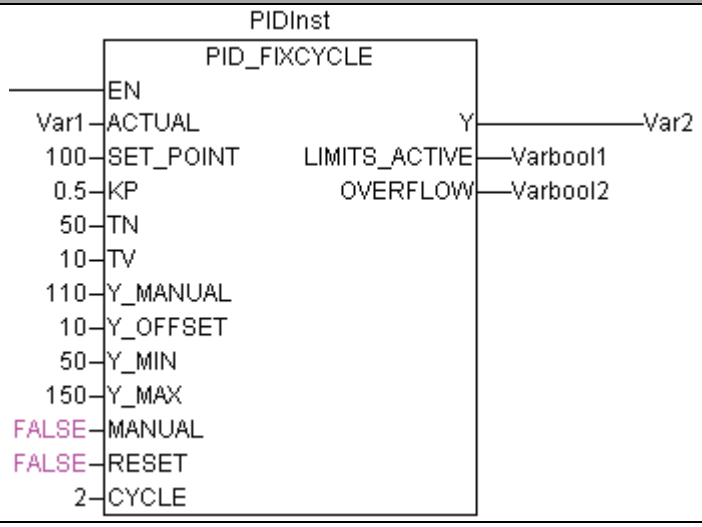
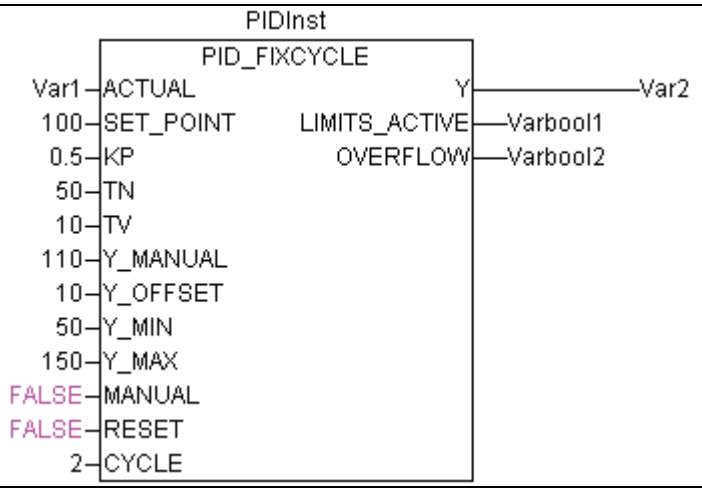
- **功能：**该指令为比例积分微分控制器。与前面介绍过的 PID 控制器对比，就是多了一个采样周期的参数 CYCLE，CYCLE 是一个 REAL 型的输入参数，是用来设定积分和微分的时间步长，单位是秒。控制方程和参数说明详见前面 PID 指令。

✧ **指令使用举例**

变量定义

VAR | VAR_INPUT | VAR_OUTPUT | VAR_IN_OUT | CONSTANT

	名称	地址	类型	初始值	注释
0001	Var1		REAL		
0002	Var2		REAL		
0003	Varbool1		BOOL		
0004	Varbool2		BOOL		
0005	PIDInst		PID_FIXCYCLE		

编程语言	程　序
梯形图（LD）	
结构化文本（ST）	PIDInst(ACTUAL := Var1, SET_POINT := 100, KP := 0.5, TN := 50, TV := 10, Y_MANUAL := 110, Y_OFFSET := 10, Y_MIN := 50, Y_MAX := 150, MANUAL := FALSE, RESET := FALSE, CYCLE := 2); Varbool1:=PIDInst.LIMITS_ACTIVE; Varbool2:=PIDInst.OVERFLOW; Var2:= PIDInst.Y;
指令列表（IL）	CAL PIDInst(ACTUAL := Var1, SET_POINT := 100, KP := 0.5, TN := 50, TV := 10, Y_MANUAL := 110, Y_OFFSET := 10, Y_MIN := 50, Y_MAX := 150, MANUAL := FALSE, RESET := FALSE, CYCLE := 2) LD PIDInst.LIMITS_ACTIVE ST Varbool1 LD PIDInst.OVERFLOW ST Varbool2 LD PIDInst.Y ST Var2
功能块（FBD）	

调整波形如前面 PID 指令所示。

4.20 信号发生器指令（Util.lib）

4.20.1 BLINK——脉冲信号发生器

- **功能：**该指令产生脉冲信号。脉冲信号发生器开始工作，输出高电平时间为TIMEHIGH，输出低电平时间为 TIMELOW，周期循环输出。

- **参数说明**

输入参数	数据类型	功能描述	参数值说明
ENABLE	BOOL	使能	TRUE 时，指令开始工作
TIMELOW	TIME	输出低电平时间	
TIMEHIGH	TIME	输出高电平时间	
输出参数	**数据类型**	**功能描述**	**参数值说明**
OUT	BOOL	脉冲信号输出值	

✧ **指令使用举例**

变量定义

VAR | VAR_INPUT | VAR_OUTPUT | VAR_IN_OUT | CONSTANT

	名称	地址	类型	初始值	注释
0001	Varbool1		BOOL		
0002	PIDInst		PID_FIXCYCLE		
0003	BLINKInst		BLINK		

编程语言 | **程　序**

梯形图（LD）

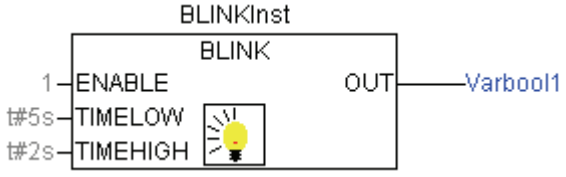
结构化文本（ST）

```
BLINKInst(TIMELOW := T#5s,TIMEHIGH := T#2s);
Varbool1:=BLINKInst.OUT;
```

指令列表（IL）

```
CAL BLINKInst(TIMELOW := T#5s,TIMEHIGH := T#2s)
LD      BLINKInst.OUT
ST      Varbool1
```

功能块（FBD）



当指令执行时，OUT 如图 4-20-1 所示波形输出。

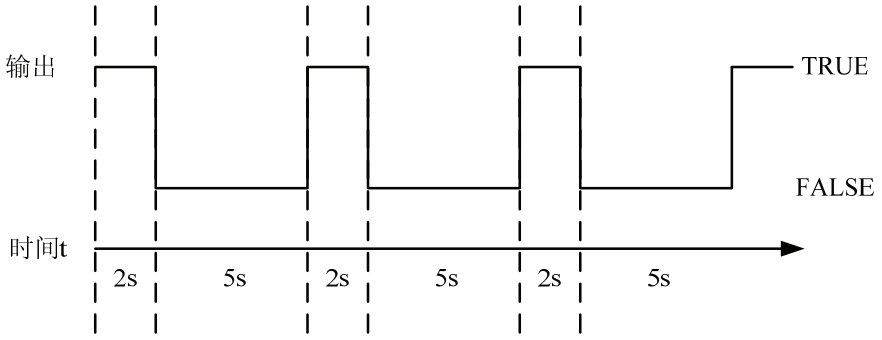


图 4-20-1

4.20.2 GEN——典型周期信号发生器

- **功能：**该指令用于生成典型的周期信号，如：三角波、零起点三角波、上升锯齿波、下降锯齿波、方波、正弦波和余弦波共七种。
- **参数说明**

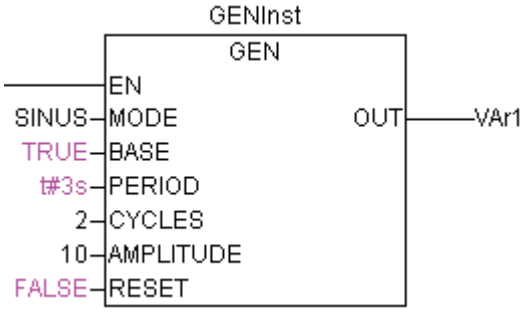
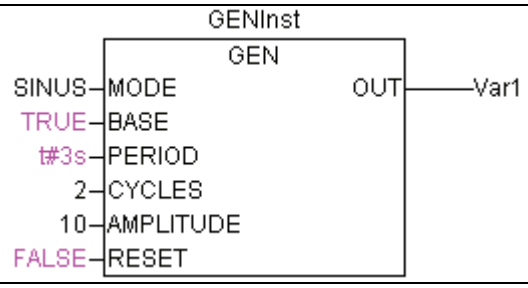
输入参数	数据类型	功能描述	参数值说明
MODE	GEN_MODE	指定要产生的信号类型	直接在 MODE 处输入 TRIANGLE、TRIANGLE_POS、SAWTOOTH_RISE、SAWTOOTH_FALL、RECTANGLE、SINUS、COSINUS，则产生对应的波形
		TRIANGLE	三角波
		TRIANGLE_POS	零起点三角波
		SAWTOOTH_RISE	上升锯齿波
		SAWTOOTH_FALL	下降锯齿波
		RECTANGLE	方波
		SINUS	正弦波
		COSINU	余弦波
BASE	BOOL	循环方式选择	当 BASE 为 TRUE 时，信号发生器与定义的循环周期有关。当 BASE 为 FALSE 时，信号发生器与特定的发生的个数有关
PERIOD	TIME	循环周期	
CYCLES	INT	发生的个数	
AMPLITUDE	INT	信号的振幅	
RESET	BOOL	初始化	当 RESET=TRUE 时，信号发生器被重新设置为 0
输出参数	**数据类型**	**功能描述**	**参数值说明**
OUT	INT	波形信号输出值	

✧ 指令使用举例

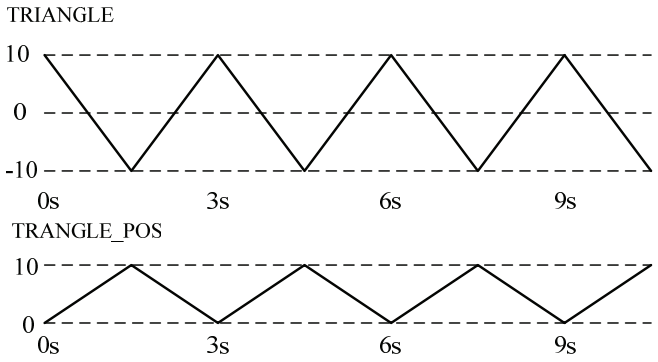
变量定义

VAR | VAR_INPUT | VAR_OUTPUT | VAR_IN_OUT | CONSTANT

	名称	地址	类型	初始值	注释
0001	GENInst		GEN		
0002	Var1		INT		

编程语言	程　序
梯形图（LD）	
结构化文本（ST）	GENInst(MODE:=SINUS, BASE:=TRUE, PERIOD:=T#3s, CYCLES:=2, AMPLITUDE:=10, RESET:=FALSE); Var1:=GENInst.OUT;
指令列表（IL）	CAL GENInst(MODE:=SINUS, BASE:=TRUE, PERIOD:=T#3s, CYCLES:=2, AMPLITUDE:=10, RESET:=FALSE) LD GENInst.OUT ST Var1
功能块（FBD）	

根据 MODE 处输入不同，产生的波形如图 4-20-2 所示。



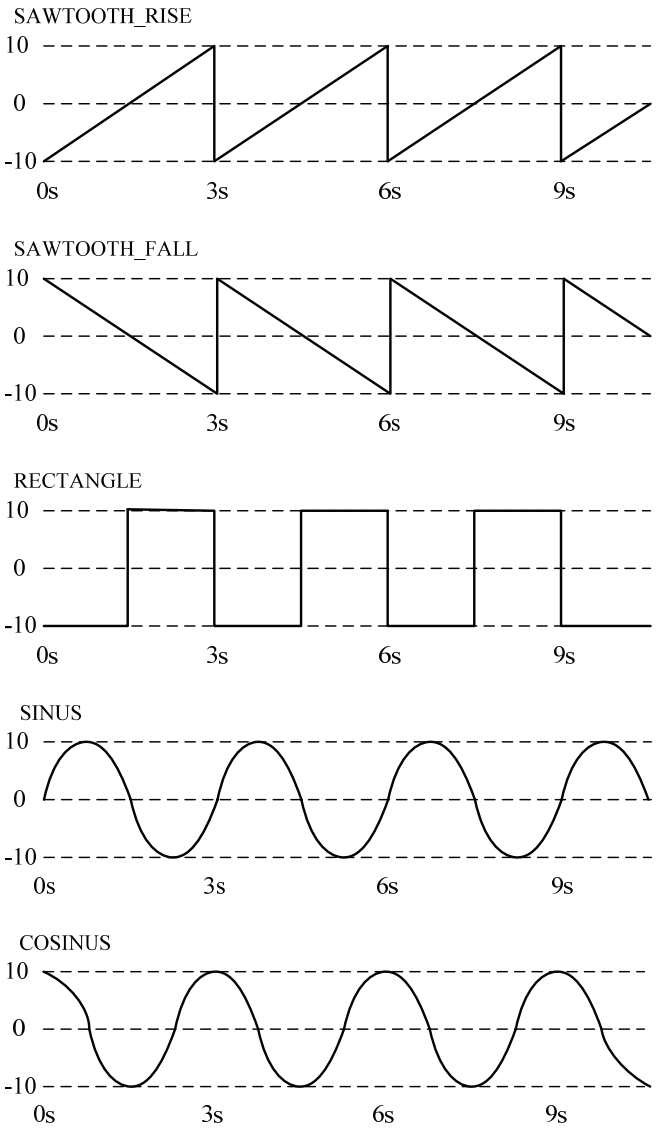


图 4-20-2

4.21 函数操纵器指令（Util.lib）

4.21.1 CHARCURVE——特征曲线

- **功能：**输入的 POINT 类型数组 P[0..N-1]在 XY 坐标图上定义了一条曲线，输入值 IN 为坐标图上的 X 轴上的点，输出值 OUT 为坐标图上该曲线所对应的 Y 轴的值。
- **参数说明**

输入参数	数据类型	功能描述	参数值说明
IN	INT	输入坐标图上 X 轴的值	
N	BYTE	指定义曲线所使用数组中的点数	
P	ARRAY[0..n] OF POINT	（2≤n≤11）	用来在 XY 坐标上定义曲线

输出参数	数据类型	功能描述	参数值说明
OUT	INT	输出值	坐标图上曲线对应的 Y 轴的值
ERR	BYTE	显示错误类型	ERR=1：数组中的点 P[0]..P[N-1]中的 X 值有错误。 ERR=2：输入值 IN 不在 P[0].X 和 P[N-1].X 之间，即超出了数组定义曲线的 X 轴的范围。此时 OUT 输出 IN 包含在限制值 P[0].Y 和 P[N-1].Y 之间所对应的数据。 ERR=4：输入 N 小于 2，或者大于 11。
P	BYTE	N 输出值	

✧ **指令使用举例**

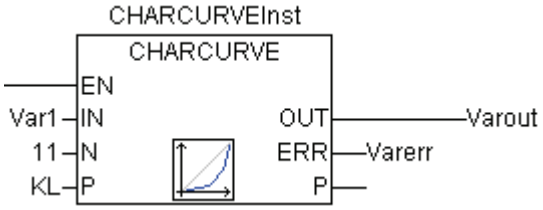
变量定义

VAR | VAR_INPUT | VAR_OUTPUT | VAR_IN_OUT | CONSTANT | RETAIN

	名称	地址	类型	初始值	注释
0001	CHARCURVEInst		CHARCURVE		
0002	Var1		INT		
0003	Varout		INT		
0004	Varerr		BYTE		
0005	KL		ARRAY[0..10] OF POINT	(X:=0,Y:=0),(X:=250,Y:=50),(X:=500,Y:=150),(X:=750,Y:=400),7((X:=1000,Y:=1000));	

编程语言 | **程 序**

梯形图（LD）



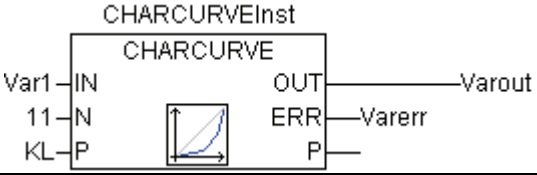
结构化文本（ST）

```
CHARCURVEInst(IN := Var1, N := 11, P := KL);
Varerr:=CHARCURVEInst.ERR;
Varout:=CHARCURVEInst.OUT;
```

指令列表（IL）

```
CAL     CHARCURVEInst(IN := Var1, N := 11, P := KL)
LD      CHARCURVEInst.ERR
ST      Varerr
LD      CHARCURVEInst.OUT
ST      Varout
```

功能块（FBD）



上例中，当指令执行时，根据输入值的变化，对应的输出值如图 4-21-1，坐标曲线由数组 KL 确定，输入 IN 为 X 轴上的值，输出 OUT 为该曲线对应的 Y 轴上的值。

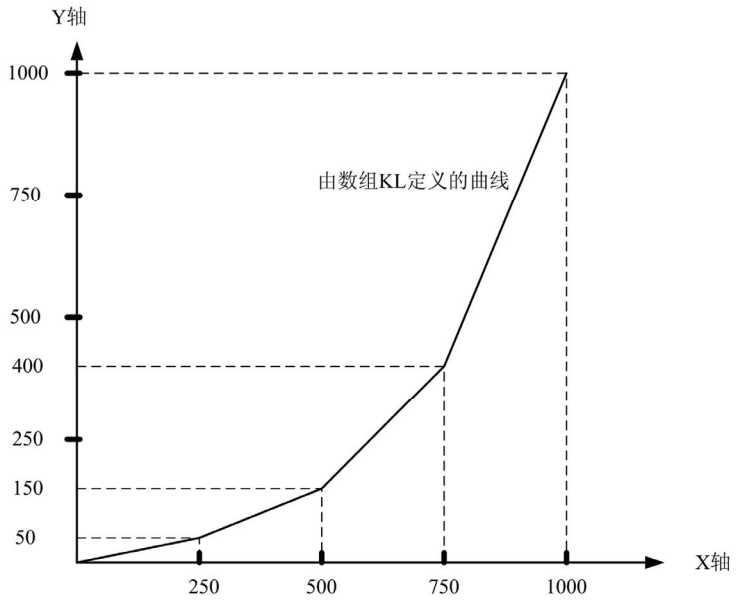


图 4-21-1

4.21.2 RAMP_INT——整型限速

- **功能：**限制整型输入函数的升降速度。
- 参数说明

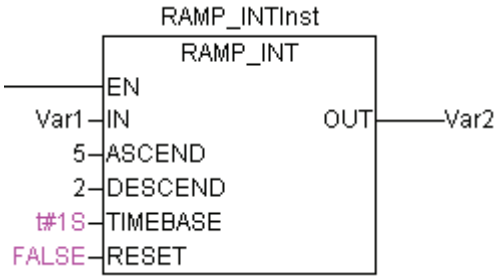
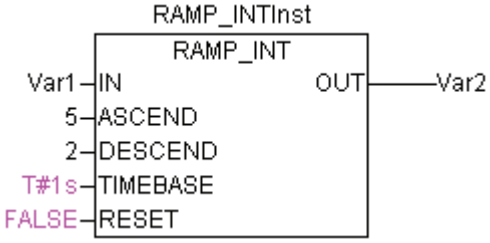
输入参数	数据类型	功能描述	参数值说明
IN	INT	目标值	若当前设定值大于先前值，则按照设定的时间和上升速率进行加法运算，由 OUT 即时输出，若当前设定值小于先前值，则按照设定的时间和下降速率进行减法运算，由 OUT 即时输出
ASCEND	INT	上升的增量	在规定时间（TIMEBASE）内上升的数量
DESCEND	INT	下升的增量	在规定时间（TIMEBASE）内下降的数量
TIMEBASE	TIME	时间基数	规定上升或者下降的速率
RESET	BOOL	初始化	设置为 TRUE 时，RAMP_INT 被重新初始化
输出参数	**数据类型**	**功能描述**	**参数值说明**
OUT	INT	即时数据输出	

✧ **指令使用举例**

变量定义

VAR | VAR_INPUT | VAR_OUTPUT | VAR_IN_OUT | CONSTANT

	名称	地址	类型	初始值	注释
0001	RAMP_INTInst		RAMP_INT		
0002	Var1		INT		
0003	Var2		INT		

编程语言	程　序
梯形图（LD）	
结构化文本（ST）	RAMP_INTInst(IN := Var1, ASCEND := 5, DESCEND := 2, TIMEBASE := T#1000ms, RESET := FALSE); Var2:=RAMP_INTInst.OUT;
指令列表（IL）	CAL　RAMP_INTInst(IN := Var1, ASCEND := 5, DESCEND := 2, TIMEBASE := T#1000ms, RESET := FALSE) LD　RAMP_INTInst.OUT ST　Var2
功能块（FBD）	

上例中，当指令执行时，根据输入值 IN 的变化，对应的输出值如图 4-21-2 所示。

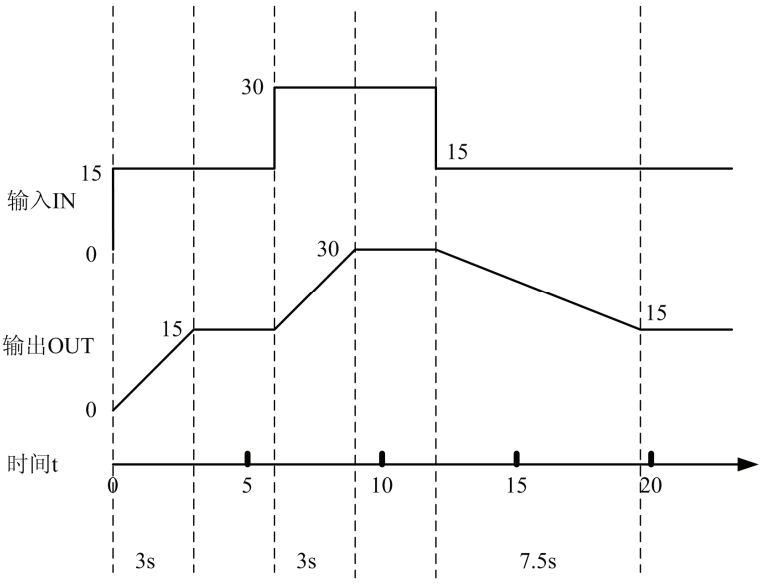


图 4-21-2

4.21.3 RAMP_REAL——实型限速

- **功能：**限制实型输入函数的升降速度。
- **参数说明**

输入参数	数据类型	功能描述	参数值说明
IN	REAL	目标值	若当前设定值大于先前值，则按照设定的时间和上升速率进行加法运算，由 OUT 即时输出，若当前设定值小于先前值，则按照设定的时间和下降速率进行减法运算，由 OUT 即时输出
ASCEND	REAL	上升的增量	在规定时间（TIMEBASE）内上升的数量
DESCEND	REAL	下升的增量	在规定时间（TIMEBASE）内下降的数量
TIMEBASE	TIME	时间基数	规定上升或者下降的速率
RESET	BOOL	初始化	设置为 TRUE 时，RAMP_INT 被重新初始化
输出参数	**数据类型**	**功能描述**	**参数值说明**
OUT	REAL	即时数据输出	

指令使用举例及输出图请参见 RAMP_INT 指令所述。

4.22 模拟量处理指令（Util.lib）

4.22.1 HYSTERESIS——滞后

- **功能：**该指令的输入包括三个 INT 类型的数值 IN、HIGH 和 LOW。如果 IN 小于下限值 LOW，OUT 为 TRUE，保持至 IN 大于上限值 HIGH。此时，OUT 由 TRUE 变为 FALSE，保持至 IN 小于下限值 LOW。OUT 由 FALSE 变为 TRUE，保持至 IN 大于上限值 HIGH，OUT 变为 FALSE，如此循环。
- **参数说明**

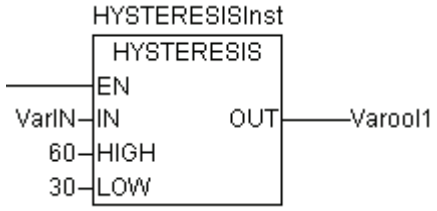
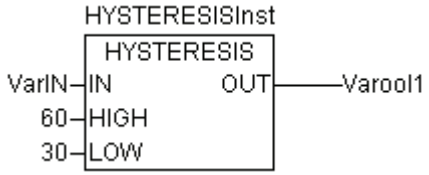
输入参数	数据类型	功能描述	参数值说明
IN	INT	输入值	
HIGH	INT	上限值	
LOW	INT	下限值	
输出参数	**数据类型**	**功能描述**	**参数值说明**
OUT	BOOL	输出值	低下限值后为 TRUE，直到上限值后为恢复 FALSE

✧ **指令使用举例**

变量定义

VAR | VAR_INPUT | VAR_OUTPUT | VAR_IN_OUT | CONSTANT

	名称	地址	类型	初始值	注释
0001	HYSTERESISInst		HYSTERESIS		
0002	Varool1		BOOL		
0003	VarIN		INT		

编程语言	程　序
梯形图（LD）	
结构化文本（ST）	HYSTERESISInst(IN := VarIN, HIGH := 60, LOW := 30); Varool1:=HYSTERESISInst.OUT;
指令列表（IL）	CAL　HYSTERESISInst(IN := VarIN, HIGH := 60, LOW := 30) LD　HYSTERESISInst.OUT ST　Varool1
功能块（FBD）	

上例中，当指令执行时，根据输入值的变化，对应的输出值如图 4-22-1 所示。

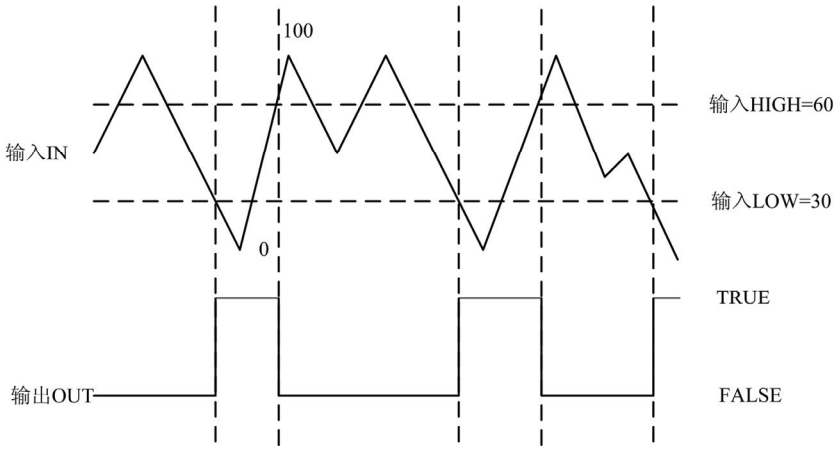


图 4-22-1

4.22.2 LIMITALARM——上下限报警

- **功能：**如果 IN 超出上限 HIGH，则 O 为 TRUE，U 和 IL 为 FALSE。如果 IN 低于下限 LOW，则 U 为 TRUE，O 和 IL 为 FALSE。如果 IN 在下限 LOW 和上限 HIGH 之间，则 IL 为 TRUE，O 和 U 为 FALSE。

➢ 参数说明

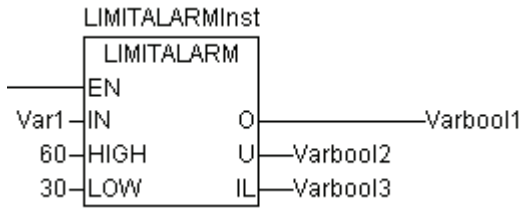
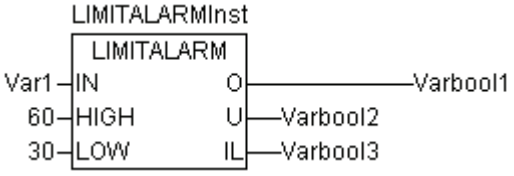
输入参数	数据类型	功能描述	参数值说明
IN	INT	输入值	
HIGH	INT	上限值	
LOW	INT	下限值	
输出参数	**数据类型**	**功能描述**	**参数值说明**
O	BOOL	输出值	
U	BOOL	输出值	
IL	BOOL	输出值	

✧ 指令使用举例

变量定义

VAR | VAR_INPUT | VAR_OUTPUT | VAR_IN_OUT | CONSTANT

	名称	地址	类型	初始值	注释
0001	LIMITALARMInst		LIMITALARM		
0002	Var1		INT		
0003	Varbool1		BOOL		
0004	Varbool2		BOOL		
0005	Varbool3		BOOL		

编程语言	程 序
梯形图（LD）	
结构化文本（ST）	LIMITALARMInst(IN := Var1, HIGH := 60, LOW := 30); Varbool2:=LIMITALARMInst.U; Varbool3:= LIMITALARMInst.IL; Varbool1:= LIMITALARMInst.O;
指令列表（IL）	CAL LIMITALARMInst(IN := Var1, HIGH := 60, LOW := 30) LD LIMITALARMInst.U ST Varbool2 LD LIMITALARMInst.IL ST Varbool3 LD LIMITALARMInst.O ST Varbool1
功能块（FBD）	

上例中，当指令执行时，根据输入值的变化，对应的输出值如下图 4-22-2 所示。

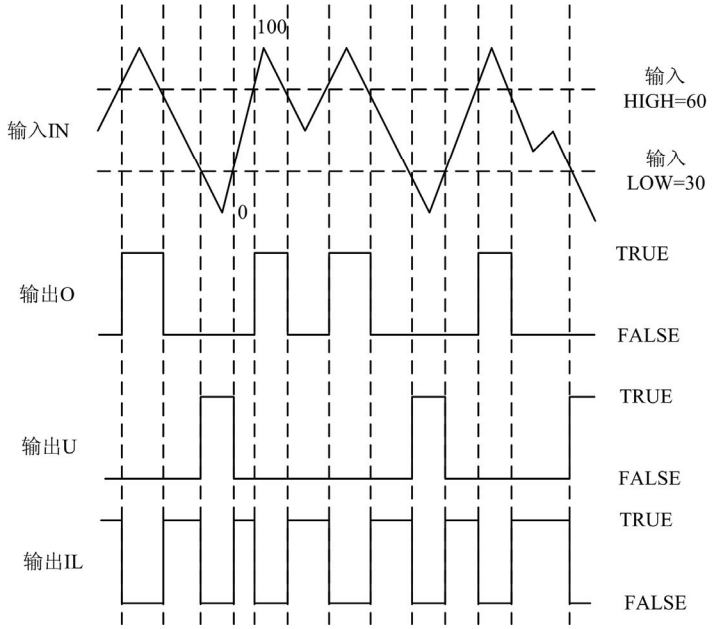


图 4-22-2

4.23 双稳态指令（Standard.lib）

4.23.1 SR——置位优先双稳态器

- **功能：**置位双稳态触发器，置位优先。
  逻辑关系：Q1=(NOT RESET AND Q1) OR SET1
  其中 SET1 为置位信号，RESET 为复位信号。
- **输入/输出数据类型：**均为 BOOL 型。

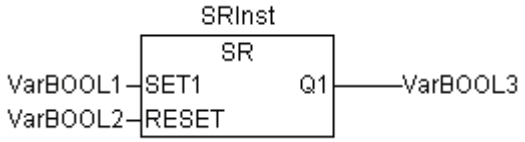
✧ **指令使用举例**

变量定义

VAR | VAR_INPUT | VAR_OUTPUT | VAR_IN_OUT | CONSTANT

	名称	地址	类型	初始值	注释
0001	SRInst		SR		
0002	VarBool1		BOOL		
0003	VarBool2		BOOL		
0004	VarBool3		BOOL		

编程语言	程　序
梯形图（LD）	SRInst SR EN VarBOOL1 — SET1　Q1 — VarBOOL3 VarBOOL2 — RESET 说明：VarBOOL3＝（NOT VarBOOL2 AND VarBOOL3）OR VarBOOL1
结构化文本（ST）	SRInst(SET1:= VarBOOL1 , RESET:=VarBOOL2);

	VarBOOL3 := SRInst.Q1 ;
指令列表（IL）	CALSRInst(SET1:= VarBOOL1, RESET:= VarBOOL2) LD SRInst.Q1 ST VarBOOL3
功能块（FBD）	

✧ **指令的真值表**

指令	**SET1**	**RESET**	**输出**
SR	0	0	保持原状态
	1	0	1
	0	1	0
	1	1	1

4.23.2 RS——复位优先双稳态器

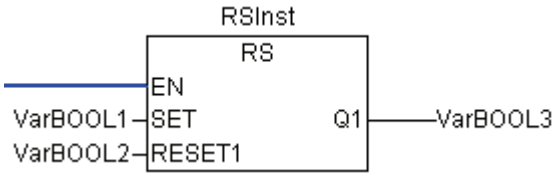
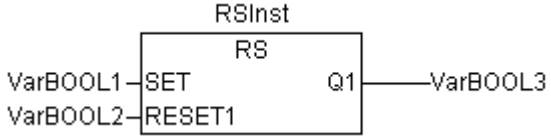
- **功能：**复位双稳态触发器，复位优先。
  逻辑关系：Q1=NOT RESET1 AND (Q1 OR SET)
  其中 SET 为置位信号，RESET1 为复位信号。
- **输入/输出数据类型：**均为 BOOL。

✧ **指令使用举例**

变量定义

VAR | VAR_INPUT | VAR_OUTPUT | VAR_IN_OUT | CONSTANT

	名称	地址	类型	初始值	注释
0001	RSInst		RS		
0002	VarBool1		BOOL		
0003	VarBool2		BOOL		
0004	VarBool3		BOOL		

编程语言	**程 序**
梯形图（LD）	 说明：VarBOOL3＝NOT VarBOOL2 AND (VarBOOL3 OR VarBOOL1)
结构化文本（ST）	RSInst(SET:= VarBOOL1 , RESET1:=VarBOOL2); VarBOOL3 := RSInst.Q1 ;
指令列表（IL）	CAL RSInst(SET := VarBOOL1, RESET1 := VarBOOL2) LD RSInst.Q1 ST VarBOOL3
功能块（FBD）	

✧ 指令的真值表

指令	SET1	RESET	输出
RS	0	0	保持原状态
	1	0	1
	0	1	0
	1	1	1

4.24 触发器指令（Standard.lib）

触发器包含上升沿检测触发器 R_TRIG 和下降沿检测触发器 F_TRIG，分别用于检测上升沿和下降沿。

4.24.1 R_TRIG——上升沿检测触发器

➢ **功能**：用于检测上升沿。

逻辑关系：Q := CLK AND NOT M;

M := CLK;

M 是初始值为 TRUE 的一个中间变量，只要 CLK 是 FALSE，Q 和 M 就是 FALSE。每次调用指令时，Q 返回 FALSE。当 CLK 检测到上升沿时，Q 返回 TRUE。

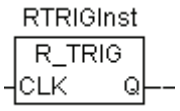
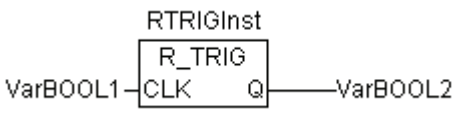
➢ **输入/输出数据类型：** CLK、Q 类型为 BOOL。

✧ 指令使用举例

变量定义

VAR | VAR_INPUT | VAR_OUTPUT | VAR_IN_OUT | CONSTANT

	名称	地址	类型	初始值	注释
0001	RTRIGInst		R_TRIG		
0002	VarBOOL1		BOOL		
0003	VarBOOL2		BOOL		

编程语言	程　序
梯形图（LD）	
结构化文本（ST）	RTRIGInst(CLK:= VarBOOL1); VarBOOL2 := RTRIGInst.Q;
指令列表（IL）	CAL　RTRIGInst(CLK := VarBOOL1) LD　RTRIGInst.Q ST　VarBOOL2
功能块（FBD）	

4.24.2 F_TRIG——下降沿检测触发器

- **功能：**用于检测下降沿。
  逻辑关系：Q := NOT CLK AND NOT M;
  M := NOT CLK;

M 是初始值为 FALSE 的一个中间变量，只要 CLK 是 TRUE，Q 和 M 保持 FALSE。CLK 是 FALSE，Q 首次返回 TRUE，M 设置为 TRUE。每次调用指令时，Q 返回 FALSE，当 CLK 检测到下降沿时，Q 为 TRUE。

- **输入/输出数据类型：**均为 BOOL 型。

✧ **指令使用举例**

变量定义

VAR | VAR_INPUT | VAR_OUTPUT | VAR_IN_OUT | CONSTANT

	名称	地址	类型	初始值	注释
0001	FTRIGInst		F_TRIG		
0002	VarBOOL1		BOOL		
0003	VarBOOL2		BOOL		

编程语言	程　序
梯形图（LD）	
结构化文本（ST）	FTRIGInst(CLK:= VarBOOL1); VarBOOL2 := FTRIGInst.Q;
指令列表（IL）	CAL FTRIGInst(CLK := VarBOOL1) LD FTRIGInst.Q ST VarBOOL2
功能块（FBD）	

4.25 计数器（Standard.lib）

计数器包括递增计数器 CTU、递减计数器 CTD 和增减计数器 CTUD。

4.25.1 CTU——递增计数器

- **功能：**递增计数器指令。
- **参数说明**

输入参数	数据类型	功能描述	参数值说明
CU	BOOL	计数输入	当 CU 有从 FALSE 到 TRUE 的上升沿，CV 加 1
RESET	BOOL	初始化	设置为 TRUE 时，CTU 被重新初始化
PV	WORD	计数器设定值	0-65535

输出参数	数据类型	功能描述	参数值说明
Q	BOOL	计数标志输出	当 CV 大于或等于上限 PV 时，Q 为 TRUE
CV	WORD	当前计数值	

✧ 指令使用举例

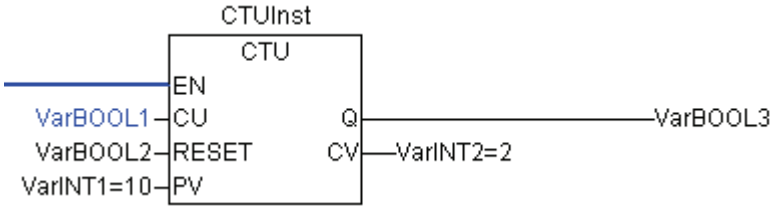
变量定义

VAR | VAR_INPUT | VAR_OUTPUT | VAR_IN_OUT | CONSTANT

	名称	地址	类型	初始值	注释
0001	CTUInst		CTU		
0002	VarBOOL1		BOOL		
0003	VarBOOL2		BOOL		
0004	VarBOOL3		BOOL		
0005	VarINT1		INT		
0006	VarINT2		INT		

编程语言 | **程　序**

梯形图（LD）



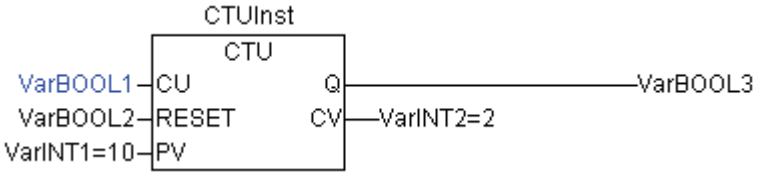
结构化文本（ST）

```
CTUInst(CU:= VarBOOL1, RESET:=VarBOOL2 , PV:= VarINT1);
VarBOOL3 := CTUInst.Q;
VarINT2 := CTUInst.CV;
```

指令列表（IL）

```
CAL   CTUInst(CU := VarBOOL1, RESET := VarBOOL2, PV :=VarINT1)
LD      CTUInst.Q
ST       VarBOOL3
LD       CTUInst.CV
ST       VarINT2
```

功能块（FBD）



4.25.2 CTD——递减计数器

➢ **功能：**递减计数器。

➢ **参数说明**

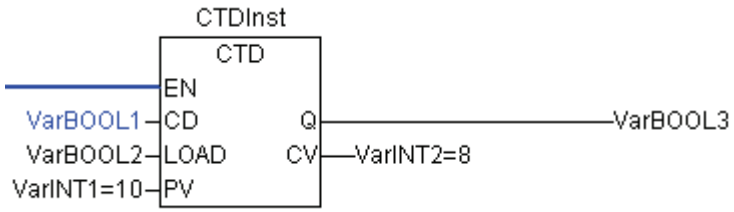
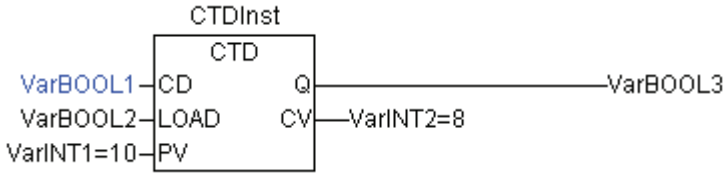
输入参数	数据类型	功能描述	参数值说明
CD	BOOL	计数输入	当 CD 有从 FALSE 到 TRUE 的上升沿，若 CV 大于 0，CV 减 1（CV 的值不小于 0）
LOAD	BOOL	初始化	LOAD 为 TRUE 时，计数变量 CV 装载为 PV
PV	WORD	计数器设定值	0-65535
输出参数	**数据类型**	**功能描述**	**参数值说明**
Q	BOOL	计数标志输出	当 CV 等于 0 时，Q 为 TRUE
CV	WORD	当前计数值	

✧ 指令使用举例

变量定义

VAR | VAR_INPUT | VAR_OUTPUT | VAR_IN_OUT | CONSTANT

	名称	地址	类型	初始值	注释
0001	CTDInst		CTD		
0002	VarBOOL1		BOOL		
0003	VarBOOL2		BOOL		
0004	VarBOOL3		BOOL		
0005	VarINT1		INT		
0006	VarINT2		INT		

编程语言	程 序
梯形图（LD）	
结构化文本（ST）	CTDInst(CD:= VarBOOL1, LOAD:=VarBOOL2 , PV:= VarINT1); VarBOOL3 := CTDInst.Q; VarINT2 := CTDInst.CV;
指令列表（IL）	CAL CTDInst(CD := VarBOOL1, LOAD := VarBOOL2, PV :=VarINT1) LD CTDInst.Q ST VarBOOL3 LD CTDInst.CV ST VarINT2
功能块（FBD）	

4.25.3 CTUD——递增递减计数器

- **功能：**递增递减计数器。
- **参数说明**

输入参数	数据类型	功能描述	参数值说明
CU	BOOL	计数输入	当 CU 有从 FALSE 到 TRUE 的上升沿，CV 加 1
CD	BOOL	计数输入	当 CD 有从 FALSE 到 TRUE 的上升沿，若 CV 大于 0，CV 减 1（CV 的值不小于 0）
RESET	BOOL	复位输入	RESET 为 TRUE 时，计数变量 CV 初始化为 0
LOAD	BOOL	初始化	LOAD 为 TRUE 时，计数变量 CV 装载为 PV
PV	WORD	计数器设定值	0-65535

输出参数	数据类型	功能描述	参数值说明
QU	BOOL	计数标志输出	当 CV 等于 PV 时，QU 为 TRUE
QD	BOOL	计数标志输出	当 CV 等于 0 时，QD 为 TRUE
CV	WORD	当前计数值	

✧ 指令使用举例

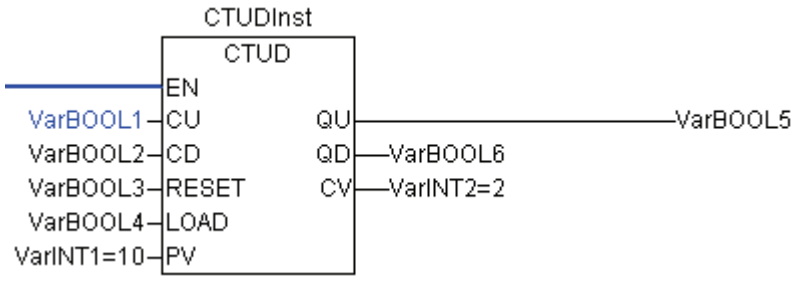
变量定义

VAR | VAR_INPUT | VAR_OUTPUT | VAR_IN_OUT | CONSTANT

	名称	地址	类型	初始值	注释
0001	CTUDInst		CUTD		
0002	VarBOOL1		BOOL		
0003	VarBOOL2		BOOL		
0004	VarBOOL3		BOOL		
0005	VarBOOL3		BOOL		
0006	VarBOOL4		BOOL		
0007	VarBOOL5		BOOL		
0008	VarBOOL6		BOOL		
0009	VarINT1		INT		
0010	VarINT2		INT		

编程语言 | **程 序**

梯形图（LD）



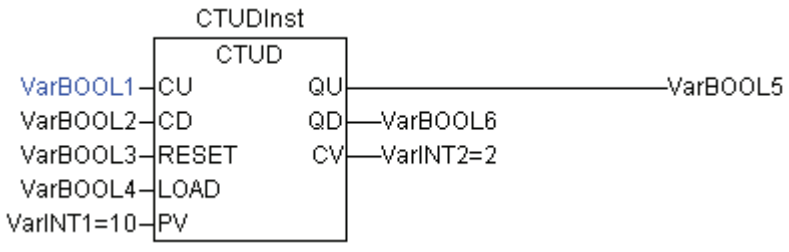
结构化文本（ST）

```
CTUDInst(CU:=VarBOOL1,CD:=VarBOOL2,RESET:=VarBOOL3,
LOAD:=VarBOOL4,PV:=VarINT1)
VarBOOL5 := CTUDInst.QU
VarBOOL6 := CTUDInst.QD
VarINT2 := CTUDInst.CV
```

指令列表（IL）

```
CAL     CTUDInst(CU:=VarBOOL1,CD:=VarBOOL2,
RESET:=VarBOOL3,LOAD:=VarBOOL4,PV:=VarINT1)
LD      CTUDInst.QU
ST      VarBOOL5
LD      CTUDInst.QD
ST      VarBOOL6
LD      CTUDInst.CV
ST      VarINT2
```

功能块（FBD）



4.26 定时器（Standard.lib）

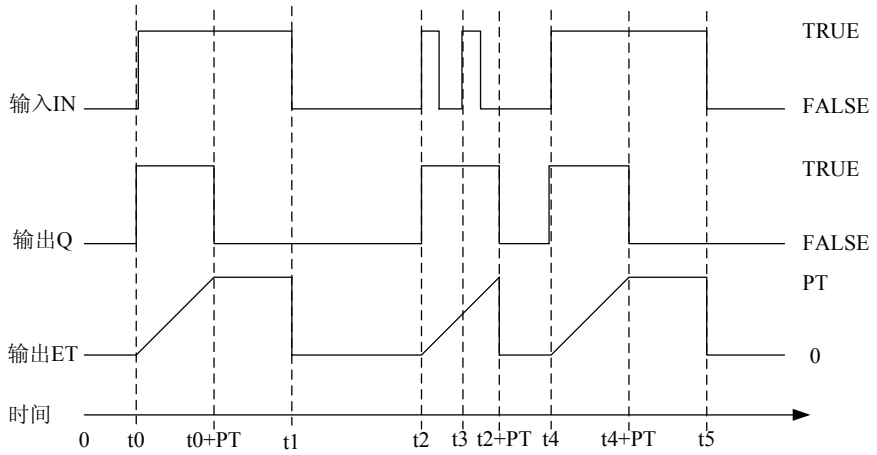
定时器包括普通定时器 TP、通电延时定时器 TON、断电延时定时器 TOF 和实时时钟 RTC，下面分别描述。

4.26.1 TP——普通定时器

- **功能：**普通定时器。
- **参数说明**

输入参数	数据类型	功能描述	参数值说明
IN	BOOL	定时器初始化信号	当 IN 变成 TRUE 时，ET 以毫秒计时直到 ET 等于 PT，然后 ET 保持常数
PT	TIME	定时时间值	
输出参数	**数据类型**	**功能描述**	**参数值说明**
Q	BOOL	定时器输出	若 IN 为 FALSE，则 Q 为 FALSE，ET 为 0。当 IN 为 TRUE 时，定时器开始工作，Q 为 TRUE，在 ET 小于等于 PT 前，IN 无效，ET 等于 PT 时，Q 为 FALSE
ET	TIME	当前时间值	当 IN 变成 TRUE 时，ET 以毫秒计时直到 ET 等于 PT，然后 ET 保持常数。计时完毕后，当 IN 为 FALSE 时，ET 等于 0

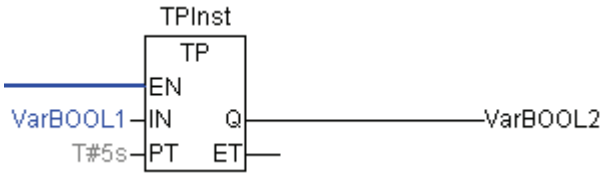
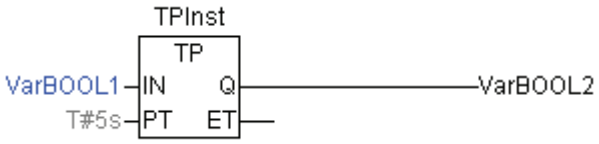
- **TP 时间顺序图**



✧ **指令使用举例**

变量定义

	名称	地址	类型	初始值	注释
0001	TPInst		TP		
0002	VarBOOL1		BOOL		
0003	VarBOOL2		BOOL		

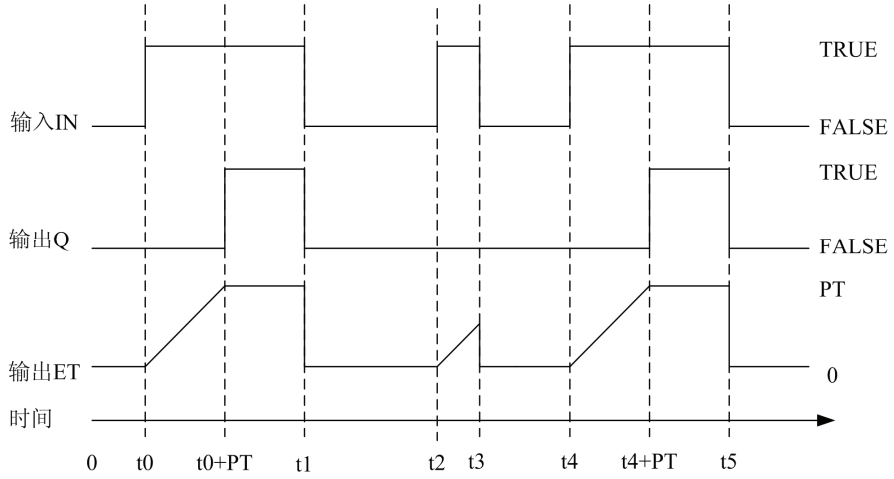
编程语言	程　序
梯形图（LD）	
结构化文本（ST）	TPInst(IN:= VarBOOL1,PT:= T#5s); VarBOOL2:=TPInst.Q;
指令列表（IL）	CAL　TPInst(IN:= VarBOOL1,PT:=T#5s) LD　TPInst.Q ST　VarBOOL2
功能块（FBD）	

4.26.2　TON——通电延时定时器

- **功能：**通电延时定时器。
- **参数说明**

输入参数	数据类型	功能描述	参数值说明
IN	BOOL	定时器初始化信号	当 IN 变成 TRUE 时，ET 以毫秒计时直到 ET 等于 PT，然后 ET 保持常数
PT	TIME	定时时间值	
输出参数	**数据类型**	**功能描述**	**参数值说明**
Q	BOOL	定时器输出	当 IN 为 FALSE 时，Q 为 FALSE，ET 为 0。当 IN 为 TRUE 时，定时器开始工作，ET 等于 PT 时，Q 为 TRUE
ET	TIME	当前时间值	当 IN 变成 TRUE 时，ET 以毫秒计时直到 ET 等于 PT，然后 ET 保持常数。无论何时，当 IN 为 FALSE 时，ET 等于 0

- **TON 时间顺序图**

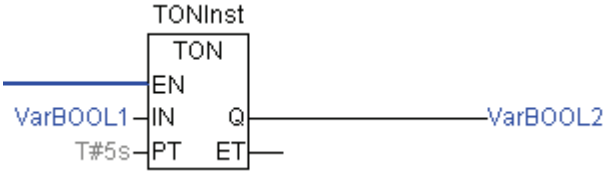
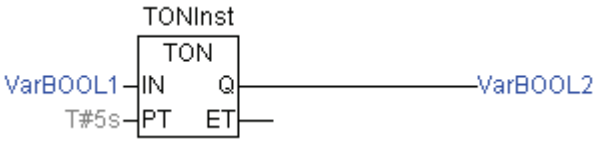


✧ 指令使用举例

变量定义

VAR | VAR_INPUT | VAR_OUTPUT | VAR_IN_OUT | CONSTANT

	名称	地址	类型	初始值	注释
0001	TONInst		TON		
0002	VarBOOL1		BOOL		
0003	VarBOOL2		BOOL		

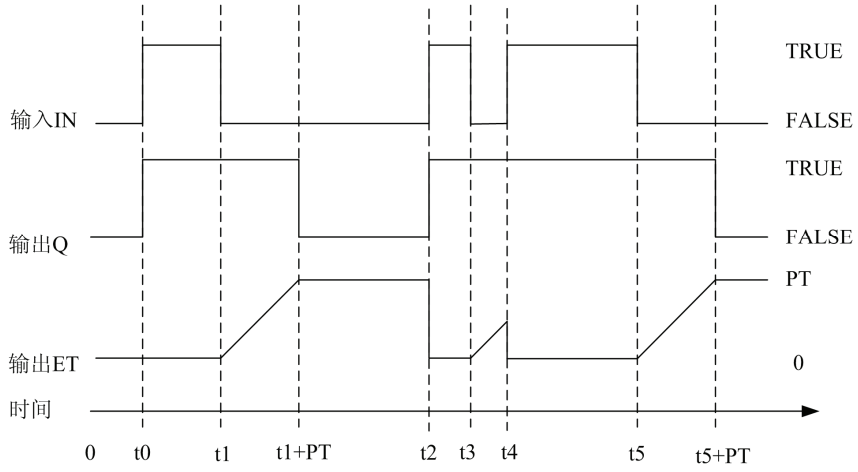
编程语言	程　序
梯形图（LD）	
结构化文本（ST）	TONInst(IN:=VarBOOL1,PT:= T#5s);
指令列表（IL）	CAL　TONInst(IN:= VarBOOL1,PT:=T#5s) LD　TONInst.Q ST　VarBOOL2
功能块（FBD）	

4.26.3 TOF——断电延时定时器

- **功能：**断电延时定时器。
- **参数说明**

输入参数	数据类型	功能描述	参数值说明
IN	BOOL	定时器初始化信号	当 IN 由 TRUE 变成 FALSE 时，ET 以毫秒计时直到 ET 等于 PT，然后 ET 保持常数
PT	TIME	定时时间值	
输出参数	**数据类型**	**功能描述**	**参数值说明**
Q	BOOL	定时器输出	当 IN 为 FALSE 且 ET 等于 PT 时，Q 为 FALSE。否则，Q 为 TRUE
ET	TIME	当前时间值	当 IN 为 FALSE 时，ET 以毫秒计数直到 ET 等于 PT，然后 ET 保持常数。无论何时，当 IN 为 TRUE 时，ET 等于 0，Q 为 TRUE

➢ **TOF 时间顺序图**



✧ **指令使用举例**

变量定义

VAR | VAR_INPUT | VAR_OUTPUT | VAR_IN_OUT | CONSTANT

	名称	地址	类型	初始值	注释
0001	TOFInst		TOF		
0002	VarBOOL1		BOOL		
0003	VarBOOL2		BOOL		

编程语言	程　序
梯形图（LD）	TOFInst TOF EN VarBOOL1 IN Q VarBOOL2 T#5s PT ET
结构化文本（ST）	TOFInst(IN:= VarBOOL1,PT:=T#5s); VarBOOL2 :=TOFInst.Q;
指令列表（IL）	CAL　TOFInst(IN:= VarBOOL1,PT:= T#5s) LD　TOFInst.Q ST　VarBOOL2
功能块（FBD）	TOFInst TOF VarBOOL1 IN Q VarBOOL2 T#5s PT ET

4.26.4　RTC——实时时钟

➢ **功能：**在给定时间启动，返回当前日期和时间。

➢ 参数说明

输入参数	数据类型	功能描述	参数值说明
EN	BOOL	使能信号	启动该指令
PDT	DT	时间基值	
输出参数	**数据类型**	**功能描述**	**参数值说明**
Q	BOOL	定时器输出	EN 为 FALSE 时，Q 为 FALSE， EN 为 TRUE 时，Q 为 TRUE

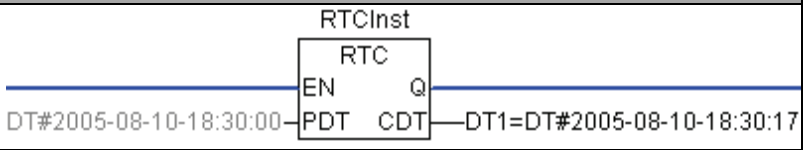
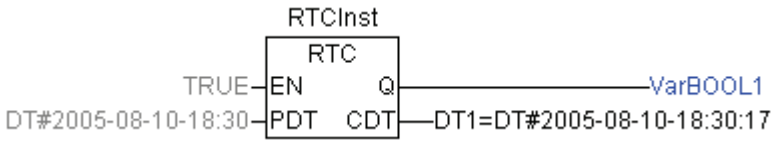
CDT	DT	当前时间值	EN 为 FALSE 时，CDT 为 1970-01-01-00:00:00，EN 为 TRUE 时，CDT 从 PDT 中的时间开始计时

✧ **指令使用举例**

变量定义

VAR | VAR_INPUT | VAR_OUTPUT | VAR_IN_OUT | CONSTANT

	名称	地址	类型	初始值	注释
0001	RTCInst		RTC		
0002	DT1		DT		
0003	VarBOOL1		BOOL		

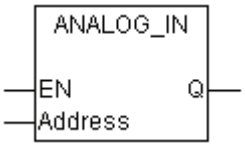
编程语言	程　序
梯形图（LD）	
结构化文本（ST）	RTCInst(PDT:=DT#2005-08-10-18:30:31); DT1:=RTCInst.CDT; VarBOOL1:=RTCInst.Q;
指令列表（IL）	CAL　RTCInst(PDT := DT#2005-08-10-18:30:31) LD　RTCInst.CDT ST　DT1 LD　RTCInst.Q ST　VarBOOL1
功能块（FBD）	

第5章 扩展指令

5.1 模拟量模块处理指令（Hollysys_PLC_Analog.lib）

5.1.1 Analog_IN——模拟量输入模块调用

✧ 指令示例

梯形图语言示例


✧ 参数说明

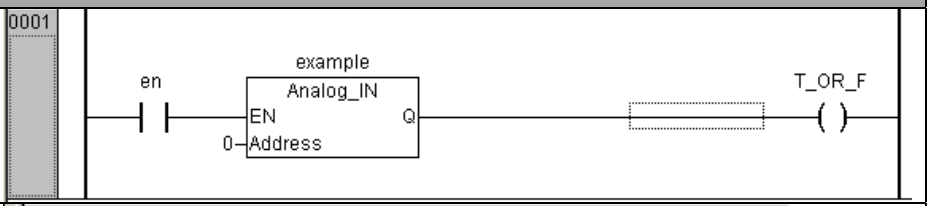
输入数据存放地址	含 义		
%IWxx	模拟量输入通道依次对应 PLC 配置中所规定的 I 区地址		
输入参数	**数据类型**	**功能描述**	**参数值说明**
EN	BOOL	使能	0：无效，不对模拟量输入模块进行扫描 1：使能，对模拟量输入模块进行扫描
Address	BYTE	该模块节点 id	0-7（与 PLC 配置里的节点 id 一致）
输出参数	**数据类型**	**功能描述**	**参数值说明**
Q	BOOL	是否读到有效数据	0：未读到数据 1：读到有效数据

✧ 指令使用举例（梯形图和结构化文本）

变量定义

VAR | VAR_INPUT | VAR_OUTPUT | VAR_IN_OUT | CONSTANT | RETAIN

	名称	地址	类型	初始值	注释
0001	T_OR_F		BOOL		
0002	en		BOOL		
0003	example		Analog_IN		

编程语言	程 序
梯形图（LD）	
结构化文本（ST）	0001 example(EN:= en,Address:=0); 0002 T_OR_F:=example.Q;

例子中 Address 处的输入值应与图 5-1-1 中 PLC 配置表中鼠标处的节点 id 一致。

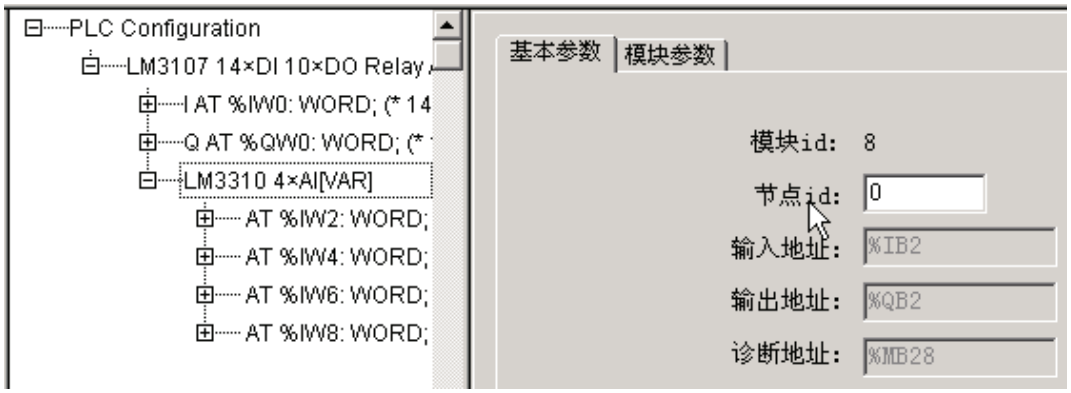


图 5-1-1

程序说明：

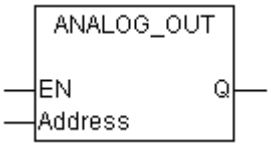
- EN 置位并保持时，对模拟量输入模块进行扫描，如上图所示，4 个通道的扫描值分别存储在%IW2、%IW4、%IW6、%IW8 寄存器中。
- EN 复位时，不对模拟量输入模块进行扫描。

提示：

- 如果要使用多个模拟量输入模块(包括 LM3310、LM3311、LM3312、LM3313 和 LM3314)，则需要配置多个 Analog_IN 指令，每个功能块对应的 Address 与对应模块的节点 id 保持一致。
- 在调用 Analog_IN 指令之前先要进行 PLC 配置（如图 5-1-1 所示）和添加库文件 Hollysys_PLC_Analog.lib。

5.1.2 Analog_OUT——模拟量输出模块调用指令

✧ **指令示例**

梯形图语言示例


✧ **参数说明**

数据输出存放地址	含 义		
%QWxx	PLC 配置中所规定的 Q 区地址，依次对应模拟量输出模块的通道		
输入参数	**数据类型**	**功能描述**	**参数值说明**
EN	BOOL	使能	0：无效，不对模拟量输出模块进行扫描 1：使能，对模拟量输出模块进行扫描
Address	BYTE	该模块节点 id	0-7（与 PLC 配置里的节点 id 一致）
输出参数	**数据类型**	**功能描述**	**参数值说明**
Q	BOOL	是否正确输出数据	0：未输出数据 1：正确输出数据

✧ 指令使用举例（梯形图和结构化文本）

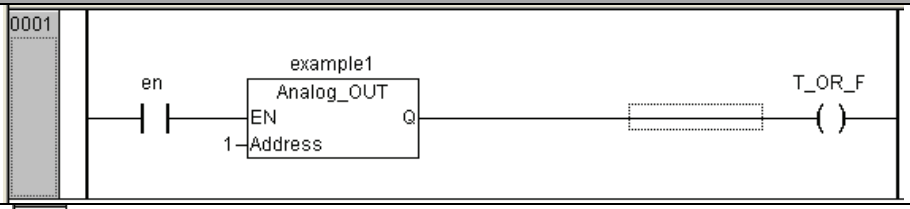
变量定义

VAR | VAR_INPUT | VAR_OUTPUT | VAR_IN_OUT | CONSTANT

	名称	地址	类型	初始值	注释
0001	T_OR_F		BOOL		
0002	en		BOOL		
0003	example1		Analog_OUT		

编程语言 | **程 序**

梯形图（LD）



结构化文本（ST）

```
0002 example1(en:=en,address:=1);
0003 T_OR_F:=example1.Q;
```

例中 Address 处的输入值应与图 5-1-2 中 PLC 配置表中鼠标处的节点 id 一致。

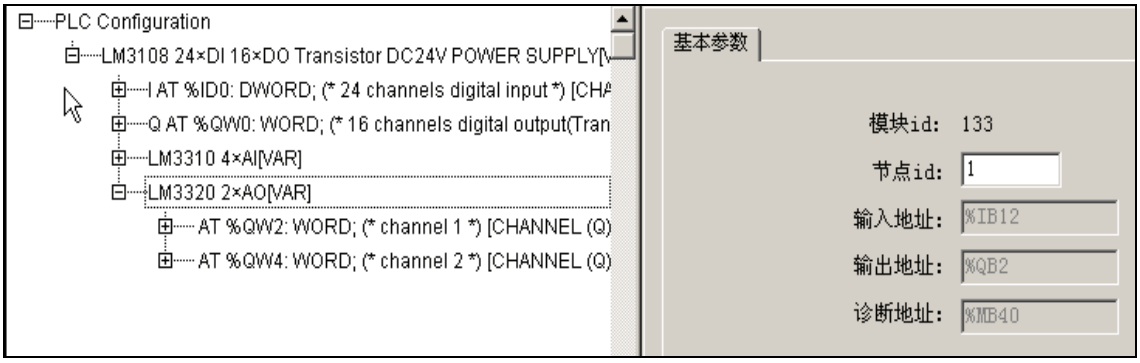


图 5-1-2

程序说明：

- EN 置位并保持时，对模拟量输出模块进行扫描，如上图所示，2 个通道的输出值分别存储在%QW2、%QW4 寄存器中。
- EN 复位时，不对模拟量输出模块进行扫描。

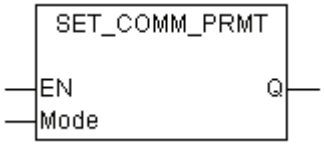
提示：

- 如果要使用多个模拟量输出模块，则需要配置多个 Analog_OUT 指令，每个指令对应的 Address 与对应模块的节点 id 保持一致。
- 在调用 Analog_OUT 指令之前，先要进行 PLC 配置（如图 5-1-2 所示）和添加库文件 Hollysys_PLC_Analog.lib。

5.2 RS232 自由口通讯指令（Hollysys_PLC_Comm.lib）

5.2.1 Set_COMM_PRMT——RS232 自由口通讯参数设置

✧ 指令示例

梯形图语言示例


✧ 参数说明

输入参数	数据类型	功能描述	参数值说明
EN	BOOL	使能	0：无效 1：上升沿使能
Mode	BYTE	通讯模式配置 （不支持 7 位无校验，这时默认为 7 位偶校验）	**Bits** 7 6 5 4 3 2 1 0 校验 位数 波特率 未定义 详见下面 Mode 参数详细说明表
输出参数	**数据类型**	**功能描述**	**值**
Q	BOOL	设置完成标志	0：设置未完成 1：设置已完成

Mode 参数详细说明表

校验		位数	波特率			未定义		设置说明
7	6	5	4	3	2	1	0	
0	0							保留现所有通讯参数（启动自由口）
0	1							偶校验
1	0							无校验（不支持 7 位无校验，这时默认为 7 位偶校验）
1	1							奇校验
		0						8 位
		1						7 位
			0	0	0			38400bps
			0	0	1			19200bps
			0	1	0			9600bps
			0	1	1			4800bps
			1	0	0			2400bps
			1	0	1			1200bps
			1	1	0			600bps
			1	1	1			300bps

提示：

- 使用 Set_COMM_PRMT 指令设定自由口参数后，要想再次下装/调试程序，需将 RUN/STOP 开关拨到 STOP 位置，才可登录。

✧ 指令使用举例（梯形图和结构化文本）

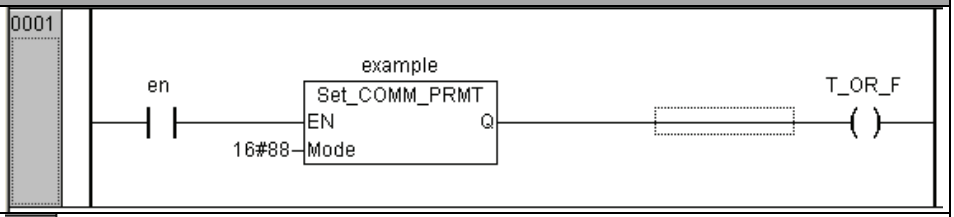
变量定义

VAR | VAR_INPUT | VAR_OUTPUT | VAR_IN_OUT | CONSTANT

	名称	地址	类型	初始值	注释
0001	T_OR_F		BOOL		
0002	en		BOOL		
0003	example		Set_COMM_PRMT		

编程语言 | **程 序**

梯形图（LD）



结构化文本（ST）

```
0002 example(EN:=en, Mode:=16#88);
0003 T_OR_F:=example.Q;
```

如下表，Mode 十六进制为 88（二进制为 10001000），则自由口为无校验、8 位、波特率 9600bps。

Mode=16#88							
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
校验		位数	波特率			未定义	
1	0	0	0	1	0	0	0

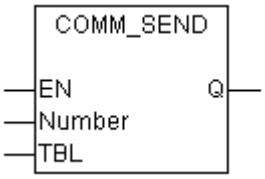
程序说明：

- EN 置位并保持时，RS232 口设置成无校验、8 位、波特率 9600bps 的自由口。
- EN 复位时，保持原设置不变。

5.2.2 COMM_SEND——RS232 自由口通讯数据发送

✧ 指令示例

梯形图语言示例



✧ 参数说明

输入参数	数据类型	功能描述	参数值说明
EN	BOOL	使能	0：无效 1：上升沿使能
Number	BYTE	发送字节数	
TBL	INT	M 区发送缓存区首地址	例如：TBL=X，则首地址为%MBX

输出参数	数据类型	功能描述	参数值说明
Q	BOOL	发送完成标志	0：发送未完成 1：发送已完成

✧ 指令使用举例（梯形图和结构化文本）

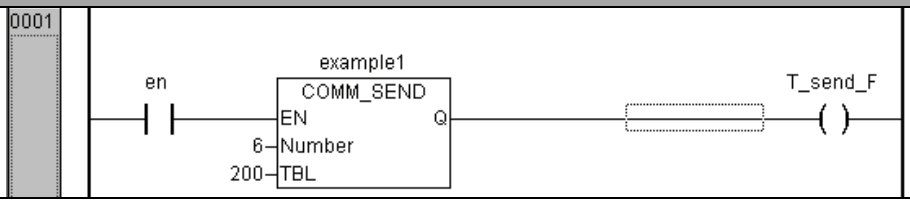
变量定义

VAR | VAR_INPUT | VAR_OUTPUT | VAR_IN_OUT | CONSTANT

	名称	地址	类型	初始值	注释
0001	en		BOOL		
0002	T_send_F		BOOL		
0003	example1		COMM_SEND		

编程语言：程 序

梯形图（LD）



结构化文本（ST）

```
0006 example1(EN:=en,Number:=6,TBL:=200);
0007 T_send_F:=example1.Q;
```

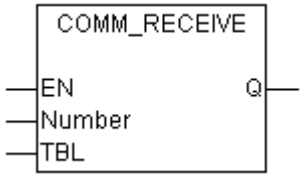
程序说明：

- 在 RS232 自由口模式下，EN 置位并保持时，将%MB200 的连续 6 个字节（即%MB200-%MB205）发送出去，Q 等于 TRUE。
- EN 复位时，不进行发送。

5.2.3 COMM_RECEIVE——RS232 自由口通讯数据接收

✧ 指令示例

梯形图语言示例



✧ 参数说明

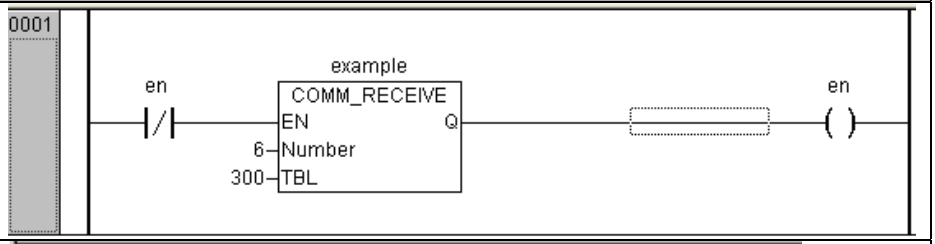
输入参数	数据类型	功能描述	参数值说明
EN	BOOL	使能	0：无效 1：上升沿使能
Number	BYTE	接收字节数	
TBL	INT	M 区接收缓存区首地址	例如：TBL=X，则首地址为%MBX
输出参数	**数据类型**	**功能描述**	**参数值说明**
Q	BOOL	接收完成标志	0：接收未完成 1：接收已完成

✧ 指令使用举例（梯形图和结构化文本）

变量定义

VAR | VAR_INPUT | VAR_OUTPUT | VAR_IN_OUT | CONSTANT

	名称	地址	类型	初始值	注释
0001	en		BOOL		
0002	example		COMM_RECEIVE		

编程语言	程　序
梯形图（LD）	
结构化文本（ST）	0001 example(EN:= NOT en,Number:=6,TBL:=300); 0002 en:=example.Q;

程序说明：

- 在 RS232 自由口模式下，上电 EN 等于 FALSE，当开始运行时第一个扫描周期接收 6 个字节数据分别保存到%MB300-%MB305 中，Q 等于 TRUE，从而 EN 等于 TRUE。第二个扫描周期时该指令不进行接收，Q 等于 FALSE，从而 EN 等于 FALSE。第三个扫描周期再次接收 6 个字节数据分别保存到%MB300-%MB305 中，如此循环接收数据。

5.2.4 Reset_COMM_PRMT——RS232 恢复协议设置

✧ 指令示例

梯形图语言示例

✧ 参数说明

输入参数	数据类型	功能描述	参数值说明
EN	BOOL	使能	0：无效 1：上升沿使能
Mode	BYTE	通讯模式配置（不支持 7 位无校验，这时默认为 7 位偶校验）	Bits 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 校验 | 位数 | 波特率 | 未定义 详见下面 Mode 参数详细说明表
输出参数	**数据类型**	**功能描述**	**参数值说明**
Q	BOOL	设置完成标志	0：设置未完成 1：设置已完成

Mode 参数详细说明表

校验		位数	波特率			未定义		设置说明
7	6	5	4	3	2	1	0	
0	0							保留现所有通讯参数（恢复 G3 专有协议）
0	1							偶校验
1	0							无校验（不支持 7 位无校验，这时默认为 7 位偶校验）
1	1							奇校验
		0						8 位
		1						7 位
			0	0	0			38400bps
			0	0	1			19200bps
			0	1	0			9600bps
			0	1	1			4800bps
			1	0	0			2400bps
			1	0	1			1200bps
			1	1	0			600bps
			1	1	1			300bps

提示：

- RS232 口默认协议是 MODBUS 协议和 G3 专有协议，使用 Set_COMM_PRMT 指令将端口设置为自由口后，需要使用本指令恢复原协议，同时也可以改变通讯参数。
- G3 专有协议通讯参数为无校验、8 位、波特率 38400bps，即 Mode 值等于 16#80。

✧ 指令使用举例（梯形图和结构化文本）

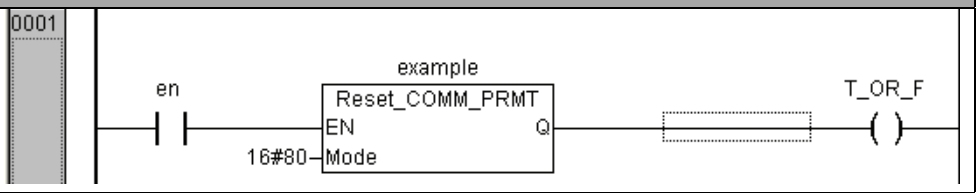
变量定义

VAR | VAR_INPUT | VAR_OUTPUT | VAR_IN_OUT | CONSTANT

	名称	地址	类型	初始值	注释
0001	en		BOOL		
0002	example		Reset_COMM_PRM		
0003	T_OR_F		BOOL		

编程语言 | **程　序**

梯形图（LD）



结构化文本（ST）

```
0001 example(EN:=en,Mode:=16#80);
0002 T_OR_F:=example.Q;
```

如下表，Mode 十六进制为 80（二进制为 10000000），则 G3 专有协议为无校验、8 位、波特率 38400bps。

Mode=16#80							
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
校验		位数	波特率			未定义	
1	0	0	0	0	0	0	0

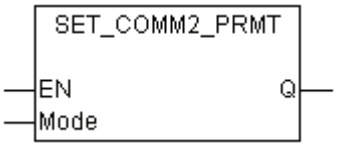
程序说明：

- EN 置位并保持时，将 RS232 口设置成无校验、8 位、波特率 38400bps 的 G3 专有协议模式。

5.3 RS485 自由口通讯指令（Hollysys_PLC_Comm2.lib）

5.3.1 Set_COMM2_PRMT——RS485 自由口通讯参数设置

✧ **指令示例**

梯形图语言示例


✧ **参数说明**

输入参数	数据类型	功能描述	参数值说明
EN	BOOL	使能	0：无效 1：上升沿使能
Mode	BYTE	通讯模式配置	Bits 7 6 5 4 3 2 1 0 校验 位数 波特率 未定义 详见 Mode 参数详细说明表
输出参数	**数据类型**	**功能描述**	**参数值说明**
Q	BOOL	设置完成标志	0：设置未完成 1：设置已完成

Mode 参数详细说明表

校验		位数	波特率			未定义		设置说明
7	6	5	4	3	2	1	0	
0	0							保留现所有通讯参数（启动自由口）
0	1							偶校验
1	0							无校验（不支持 7 位无校验，这时默认为 7 位偶校验）
1	1							奇校验
		0						8 位
		1						7 位
			0	0	0			38400bps
			0	0	1			19200bps
			0	1	0			9600bps
			0	1	1			4800bps
			1	0	0			2400bps
			1	0	1			1200bps
			1	1	0			600bps
			1	1	1			300bps

✧ 指令使用举例（梯形图和结构化文本）

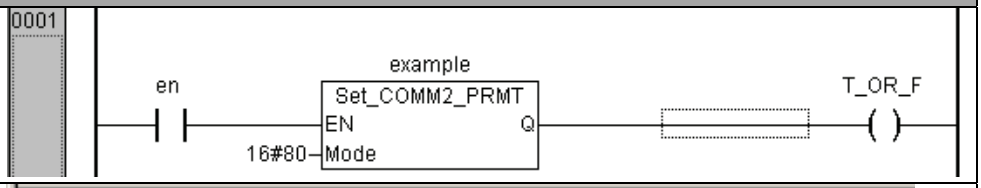
变量定义

VAR | VAR_INPUT | VAR_OUTPUT | VAR_IN_OUT | CONSTANT

	名称	地址	类型	初始值	注释
0001	en		BOOL		
0002	example		Set_COMM2_PRMT		
0003	T_OR_F		BOOL		

编程语言 | **程 序**

梯形图（LD）



结构化文本（ST）

```
0001 example(EN:=en,Mode:=16#80);
0002 T_OR_F:=example.Q;
```

如下表，Mode 十六进制为 80（二进制为 10000000），则自由口为无校验、8 位、波特率 38400bps。

Mode=16#80							
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
校验		位数	波特率			未定义	
1	0	0	0	0	0	0	0

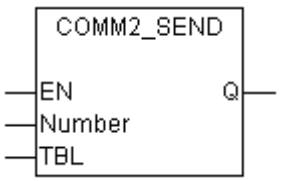
程序说明：

- EN 置位并保持时，RS485 口设置成无校验、8 位、波特率 38400bps 的自由口。
- EN 复位时，保持原设置不变。

5.3.2 COMM2_SEND——RS485 自由口通讯数据发送

✧ 指令示例

梯形图语言示例



✧ 参数说明

输入参数	功能描述	数据类型	值
EN	使能	BOOL	0：无效 1：上升沿使能
Number	发送字节数	BYTE	
TBL	M 区发送缓存区首地址	INT	例如：TBL=X，则首地址为%MBX

输出参数	功能	数据类型	值
Q	发送完成标志	BOOL	0：发送未完成 1：发送已完成

✧ 指令使用举例（梯形图和结构化文本）

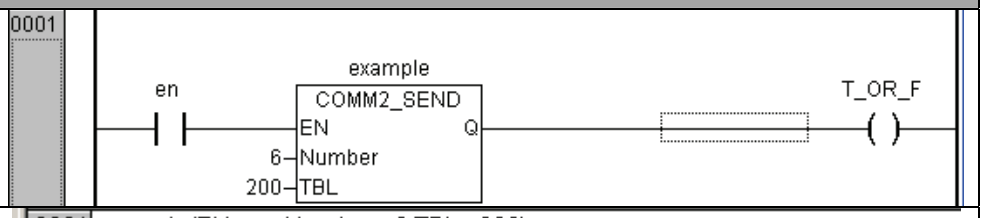
变量定义

VAR | VAR_INPUT | VAR_OUTPUT | VAR_IN_OUT | CONSTANT

	名称	地址	类型	初始值	注释
0001	en		BOOL		
0002	example		COMM2_SEND		
0003	T_OR_F		BOOL		

编程语言：程 序

梯形图（LD）



结构化文本（ST）

```
0001 example(EN:=en,Number:=6,TBL:=200);
0002 T_OR_F:=example.Q;
```

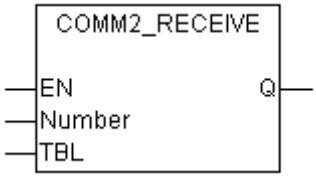
程序说明：

- 在 RS485 自由口模式下，EN 置位并保持时，将%MB200 的连续 6 个字节（即%MB200-%MB205）发送出去，Q 等于 TRUE，保持。
- EN 复位时，不进行发送。

5.3.3 COMM2_RECEIVE——RS485 自由口通讯数据接收

✧ 指令示例

梯形图语言示例



✧ 参数说明

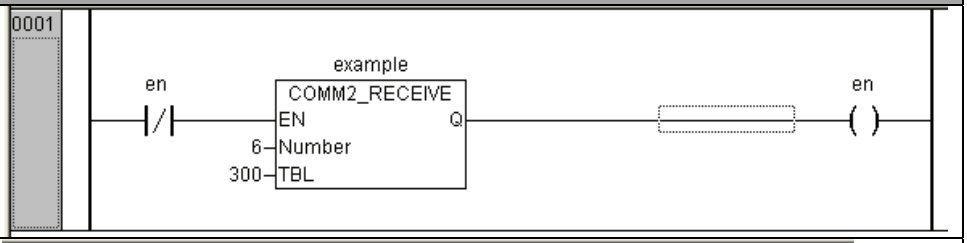
输入参数	数据类型	功能描述	参数值说明
EN	BOOL	使能	0：无效 1：上升沿使能
Number	BYTE	接收字节数	
TBL	INT	M 区接收缓存区首地址	例如：TBL=X，则首地址为%MBX
输出参数	**数据类型**	**功能描述**	**参数值说明**
Q	BOOL	接收完成标志	0：接收未完成 1：接收已完成

✧ 指令使用举例（梯形图和结构化文本）

变量定义

VAR | VAR_INPUT | VAR_OUTPUT | VAR_IN_OUT | CONSTANT | RETAIN

	名称	地址	类型	初始值	注释
0001	en		BOOL		
0002	example		COMM2_RECEIVE		

编程语言	程　序
梯形图（LD）	
结构化文本（ST）	0001 example(EN:=NOT(en),Number:=6,TBL:=300); 0002 en:=example.Q;

程序说明：

- 在 RS485 自由口模式下，上电 EN 等于 FALSE，当开始运行时第一个扫描周期接收 6 个字节数据分别保存到%MB300-%MB305 中，Q 等于 TRUE，从而 EN 等于 TRUE。第二个扫描周期时该指令不进行接收，Q 等于 FALSE，从而 EN 等于 FALSE。第三个扫描周期再次接收 6 个字节数据分别保存到%MB300-%MB305 中，如此循环接收数据。

5.3.4 Reset_COMM2_PRMT——RS485 恢复协议设置

✧ 指令示例

梯形图语言示例

✧ 参数说明

输入参数	数据类型	功能描述	参数值说明
EN	BOOL	使能	0：无效 1：上升沿使能
Mode	BYTE	通讯模式配置	Bits 7 6 5 4 3 2 1 0 校验 位数 波特率 未定义 详见 Mode 参数详细说明表
输出参数	**数据类型**	**功能描述**	**参数值说明**
Q	BOOL	设置完成标志	0：设置未完成 1：设置已完成

Mode 参数详细说明表

校验		位数	波特率			未定义		设置说明
7	6	5	4	3	2	1	0	
0	0							保留现所有通讯参数（恢复 G3 专有协议）
0	1							偶校验
1	0							无校验（不支持 7 位无校验，这时默认为 7 位偶校验）
1	1							奇校验
		0						8 位
		1						7 位
			0	0	0			38400bps
			0	0	1			19200bps
			0	1	0			9600bps
			0	1	1			4800bps
			1	0	0			2400bps
			1	0	1			1200bps
			1	1	0			600bps
			1	1	1			300bps

提示：

- RS485 口默认协议是 MODBUS 协议，使用 Set_COMM2_PRMT 指令将端口设置为自由口后，需要使用本指令恢复原协议。

✧ **指令使用举例（梯形图和结构化文本）**

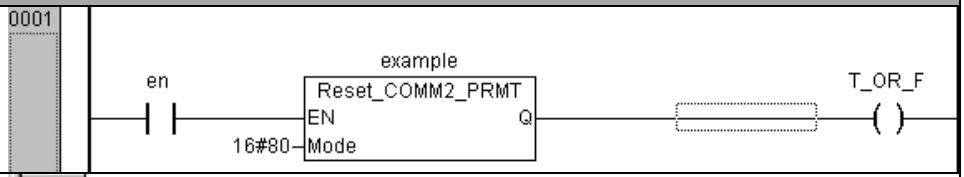
变量定义

VAR | VAR_INPUT | VAR_OUTPUT | VAR_IN_OUT | CONSTANT | RETAIN

	名称	地址	类型	初始值	注释
0001	en		BOOL		
0002	example		Reset_COMM2_PRI		
0003	T_OR_F		BOOL		

编程语言 | **程 序**

梯形图（LD）



结构化文本（ST）

```
0001 example(EN:= en,Mode:=16#80);
0002 T_OR_F:=example.Q;
```

如下表，Mode 十六进制为 80（二进制为 10000000），则端口通讯参数为无校验、8 位、波特率 38400bps。

Mode=16#80							
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
校验		位数	波特率			未定义	
1	0	0	0	0	0	0	0

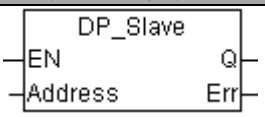
程序说明：

- EN 置位并保持时，将 RS485 口设置成无校验、8 位、波特率 38400bps 的 ModBus 协议模式。

5.4 Profibus-DP 指令（Hollysys_PLC_DPSlave.lib）

DP_Slave——Profibus-DP 从站模块（LM3401）调用

✧ **指令示例**

梯形图语言示例


✧ **参数说明**

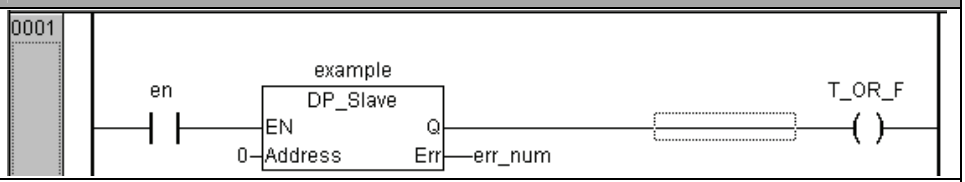
CPU 模块输入输出数据存放地址			
%IWxx %QWxx	PLC 配置中规定的 DP 从站模块所占用 I 和 Q 区地址		
输入参数	**数据类型**	**功能描述**	**参数值说明**
EN	BOOL	使能	0：无效，不对 DP 模块进行扫描 1：使能，对 DP 模块进行扫描
Address	BYTE	该模块节点 id	0-7（与 PLC 配置里的节点 id 一致）
输出参数	**数据类型**	**功能描述**	**参数值说明**
Q	BOOL	是否读到有效数据	0：未读到数据 1：读到有效数据
Err	BYTE	与从站模块通讯故障号	0：无故障 1：等待 DP 从站模块 Ready 错 2：读中断 DP 从站模块错 3：读应答错 4：读数据长度错 5：读数据错 6：写数据错 7：写中断 DP 从站模块错

✧ **指令使用举例（梯形图和结构化文本）**

变量定义

VAR | VAR_INPUT | VAR_OUTPUT | VAR_IN_OUT | CONSTANT | RETAIN

	名称	地址	类型	初始值	注释
0001	en		BOOL		
0002	example		DP_Slave		
0003	err_num		BYTE		
0004	T_OR_F		BOOL		

编程语言	**程 序**
梯形图（LD）	
结构化文本（ST）	0001 example(EN:= en,Address:=0); 0002 T_OR_F:=example.Q; 0003 err_num:=example.Err;

例子中 Address 处的输入值应与图 5-4-1 中 PLC 配置表中鼠标处的节点 id 一致。

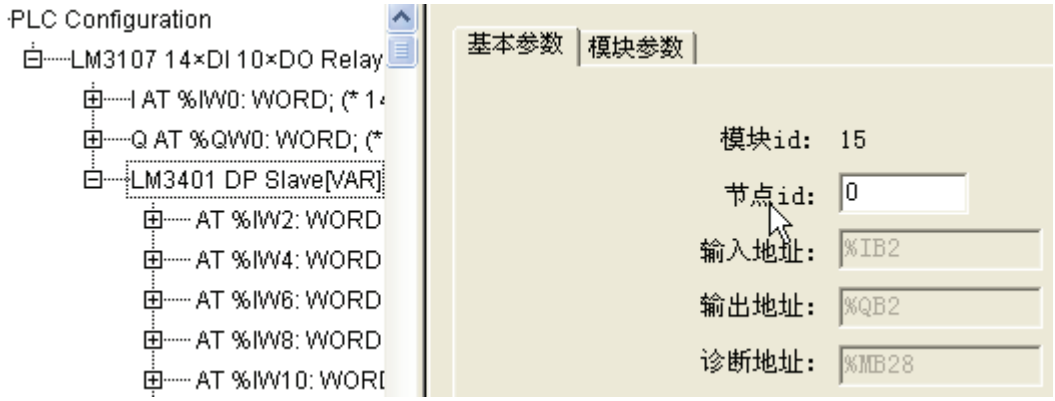
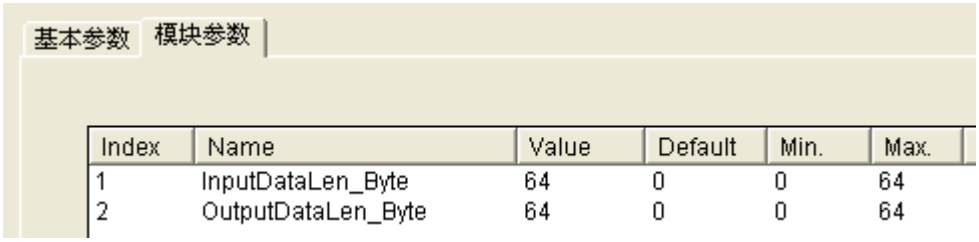


图 5-4-1

在图 5-4-2 中模块参数的 Value 列配置输入输出区的大小，最小为 0，最大为 64。



Index	Name	Value	Default	Min.	Max.
1	InputDataLen_Byte	64	0	0	64
2	OutputDataLen_Byte	64	0	0	64

图 5-4-2

程序说明：

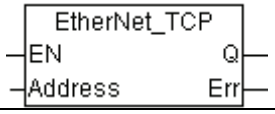
- EN 置位时，对 DP 模块进行扫描，EN 复位时，不对 DP 模块进行扫描。

> **提示：**
> - 关于 DP 模块的使用，请参考附录 B 例程。

5.5 以太网指令（Hollysys_PLC_EtherNet.lib）

EtherNet_TCP——以太网模块（LM3403）调用

✧ **指令示例**

梯形图语言示例


✧ **参数说明**

CPU 模块输入输出数据存放地址			
%IWxx %QWxx	PLC 配置中规定以太网模块所占用的 I 和 Q 区地址		
输入参数	**数据类型**	**功能描述**	**参数值说明**
EN	BOOL	使能	0：无效，不对以太网模块扫描 1：使能，对以太网模块扫描
Address	BYTE	该模块节点 id	0-7（与 PLC 配置里的节点 id 一致）

输出参数	数据类型	功能描述	参数值说明
Q	BOOL	是否读到有效数据	0：未读到数据 1：读到有效数据
Err	BYTE	与以太网模块 通讯故障号	0：无故障 1：等待以太网模块 Ready 错 2：读中断以太网模块错 3：读应答错 4：读数据长度错 5：读数据错 6：写数据错 7：写中断以太网模块错

✧ **指令使用举例（梯形图和结构化文本）**

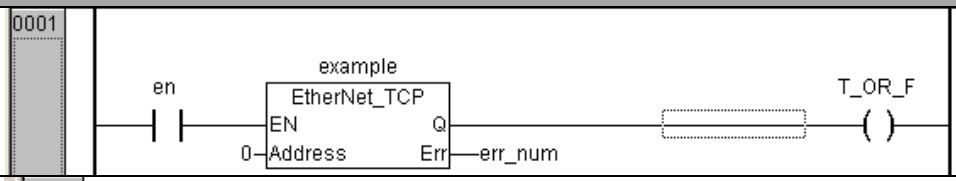
变量定义

VAR | VAR_INPUT | VAR_OUTPUT | VAR_IN_OUT | CONSTANT

	名称	地址	类型	初始值	注释
0001	en		BOOL		
0002	example		EtherNet_TCP		
0003	err_num		BYTE		
0004	T_OR_F		BOOL		

编程语言 | **程 序**

梯形图（LD）



结构化文本（ST）

```
0001 example(EN:= en,Address:=0);
0002 T_OR_F:=example.Q;
0003 err_num:=example.Err;
```

上例中 Address 处的输入值应与图 5-5-1 中 PLC 配置表中鼠标处的节点 id 一致。

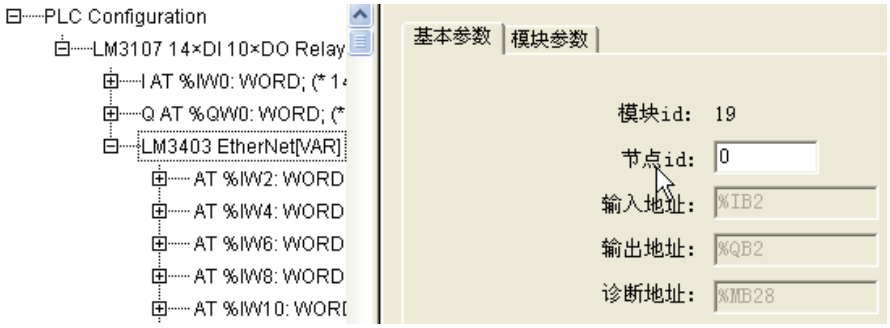


图 5-5-1

在图 5-5-2 中分别配置 IP 地址、子网掩码、网关、输入输出使用空间的大小。

基本参数 模块参数

Index	Name	Value	Default	Min.	Max.
1	IP_Address	172.20.45.160			
2	Subnet_Mask	255.255.252.0			
3	Gateway_Address	172.20.45.1			
4	MAC_Address				
5	ReadDataLen_Byte	200	0	0	200
6	WriteDataLen_Byte	200	0	0	200

图 5-5-2

程序说明：

- EN 置位时，对以太网模块进行扫描，EN 复位时，不对以太网模块进行扫描。

提示：

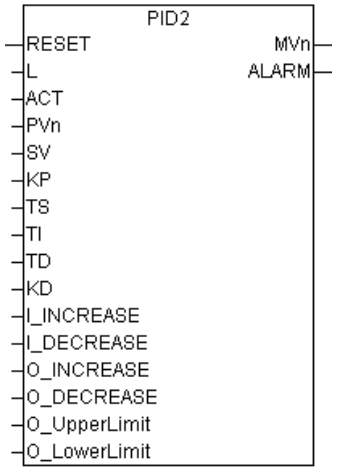
- 关于以太网模块的使用，请参考附录 B 例程。

5.6 正反动作 PID 控制器（Hollysys_PLC_Util.lib）

PID2——正反动作可选 PID 控制器

✧ **指令示例**

梯形图语言示例



✧ **参数说明**

输入参数	数据类型	功能描述	参数值说明
RESET	BOOL	复位，高电平有效	
L	REAL	滤波系数（0-1）	
ACT	UINT	动作方向/报警设置 （ACT 的.2 和.5 不能同时为 1）	.0 = 0—正动作，1—逆动作 .1 = 0—输入报警无效， 1—输入报警有效 .2 = 0—输出报警无效，

			1—输出报警有效 .5 = 0—输出上下限设定无效， 1—输出上下限设定有效
PVn	REAL	测定值	
SV	REAL	设定值	
KP	REAL	比例增益（KP>0）	
TS	REAL	采样周期（TS>0）	单位为秒（S）
TI	REAL	积分常数（TI≥0）	单位为秒（S）
TD	REAL	微分常数（TD≥0）	单位为秒（S）
KD	REAL	微分增益（0-1）	
I_INCREASE	REAL	输入变化量增侧报警设定值	
I_DECREASE	REAL	输入变化量减侧报警设定值	
O_INCREASE	REAL	输出变化量增侧报警设定值	
O_DECREASE	REAL	输出变化量减侧报警设定值	
O_UpperLimit	REAL	输出上限设定值	
O_LowerLimit	REAL	输出下限设定值	
输出参数	**数据类型**	**功能描述**	**参数值说明**
MVn	REAL	输出值	
ALARM	UINT	报警输出	.0 = 0—输入变化量增侧正常， 1—输入变化量增侧溢出 .1 = 0—输入变化量减侧正常， 1—输入变化量减侧溢出 .2 = 0—输出变化量增侧正常， 1—输出变化量增侧溢出 .3 = 0—输出变化量减侧正常， 1—输出变化量减侧溢出

✧ **PID 基本运算式**

表达式	含义		表达式	含义
EV_n	本次采样时的偏差		D_n	本次的微分项
EV_{n-1}	前一次采样时的偏差		D_{n-1}	前一次的微分项
SV	设定值		K_P	比例增益
PV_{nf}	本次采样时的测定值（滤波后）		T_S	采样周期
PV_{nf-1}	前一次采样时的测定值（滤波后）		T_I	积分时间常数
PV_{nf-2}	前二次采样时的测定值（滤波后）		T_D	微分时间常数
△MV	输出变化量		MV_n	本次的操作量
KD：	微分增益			

正动作：

$$\Delta MV = K_p\{(EV_n - EV_{n-1}) + \frac{T_S}{T_I} * EV_n + D_n\}$$

$$EV_n = PV_{nf} - SV$$

$$D_n = \frac{T_D}{T_S + KD * T_D}(-2PV_{nf-1} + PV_{nf} + PV_{nf-2}) + \frac{KD * T_D}{T_S + KD * T_D} * D_{n-1}$$

$$MV_n = \sum \Delta MV$$

逆动作：

$$\Delta MV = K_p\{(EV_n - EV_{n-1}) + \frac{T_S}{T_I} * EV_n + D_n\}$$

$$EV_n = SV - PV_{nf}$$

$$D_n = \frac{T_D}{T_S + KD * T_D}(2PV_{nf-1} - PV_{nf} - PV_{nf-2}) + \frac{KD * T_D}{T_S + KD * T_D} * D_{n-1}$$

$$MV_n = \sum \Delta MV$$

PV_{nf} 是根据读入的测定值由下列运算式求得的值。

滤波后测定值 $PV_{nf} = PV_n + \mathrm{L}(PV_{nf-1} - PV_n)$

PV_n：本次采样时的测定值。

L：滤波系数（0~1）。

PV_{nf-1}：1 个周期前的测定值（滤波后）。

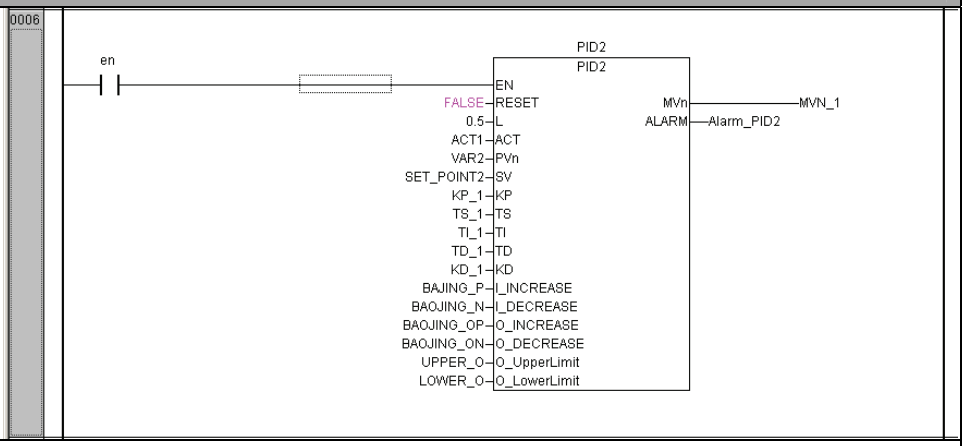
✧ **指令使用举例（梯形图和结构化文本）**

变量定义

	名称	地址	类型	初始值	注释
0001	PID2		PID2		
0002	en		BOOL		
0003	ACT1		UINT		
0004	VAR2		REAL		
0005	SET_POINT2		REAL	100	
0006	KP_1		REAL	0.1	
0007	TS_1		REAL	100	
0008	TI_1		REAL	100	
0009	TD_1		REAL	100	
0010	KD_1		REAL	0.2	
0011	BAJING_P		REAL	500	
0012	BAOJING_N		REAL	-500	
0013	BAOJING_OP		REAL	500	
0014	BAOJING_ON		REAL	-500	
0015	UPPER_O		REAL	1000	
0016	LOWER_O		REAL	-1000	
0017	MVN_1		REAL		
0018	Alarm_PID2		UINT		

编程语言 | **程　序**

梯形图（LD）



结构化文本（ST）

```
0001 IF en=TRUE THEN;
0002 PID2(RESET:=FALSE,L:=0.5,ACT:=ACT1,PVn:=VAR2,SV:=SET_POINT2,KP:=KP_1,TS:=TS_1,TI:=TI_1,TD:=TD_1,KD:=KD_1,I_INCREASE:=BAJING_P,
0003 I_DECREASE:=BAOJING_N,O_INCREASE:=BAOJING_OP,O_DECREASE:=BAOJING_ON,O_UpperLimit:=UPPER_O,O_LowerLimit:=LOWER_O);
0004 MVN_1:=PID2.MVn;
0005 Alarm_PID2:=PID2.ALARM;
0006 END_IF
```

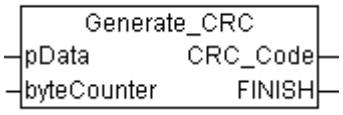
程序说明：

- EN 使能后，该指令执行，根据各个参数的值按照正动作或逆动作的公式完成 PID 运算；EN 复位时，不执行该指令。

5.7 Modbus 校验指令（Hollysys_PLC_Modbus_CRC.lib）

Generate_CRC——产生 Modbus CRC 校验码

✧ **指令示例**

梯形图语言示例


✧ **参数说明**

输入	数据类型	功能描述	参数值说明
pData	POINTER TO BYTE	需要校验数据的起始指针	
byteCounter	WORD	需要校验数据的字节数	
输出	**数据类型**	**功能描述**	**参数值说明**
CRC_Code	WORD	校验结果	
FINISH	BOOL	是否完成校验	0：未完成校验　1：完成校验

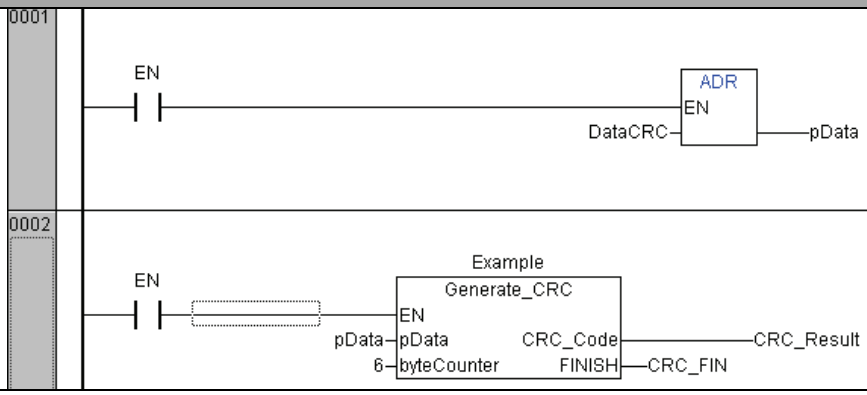
✧ **指令使用举例（变量定义与梯形图）**

变量定义

	名称	地址	类型	初始值	注释
0001	EN		BOOL		
0002	pData		POINTER TO BYTE		
0003	Example		Generate_CRC		
0004	CRC_Result	%MW206	WORD		
0005	CRC_FIN		BOOL		
0006	DataCRC	%MB200	ARRAY[1..6] OF BYTE	16#01,16#03,16#0C, 16#1C,16#00,16#01;	

编程语言：程　序

梯形图（LD）



结构化文本（ST）

```
0001 IF EN=TRUE THEN
0002    pData:=ADR(DataCRC);
0003    Example(pData:=pData,byteCounter:=6);
0004    CRC_Result:=Example.CRC_Code;
0005    CRC_FIN:=Example.FINISH;
0006 END_IF
```

程序说明：

- EN 置位时，先用 ADR 算出 DataCRC 数组的指针，然后赋值给 Generate_CRC 指令，在 byteCounter 处填入需要校验的字节数 6，则 Generate_CRC 指令可计算出 16#01、16#03、16#0C、16#1C、16#00、16#01 的 CRC 校验码值 CRC_Code,保存到变量 CRC_Result(地址为%MW206)处。

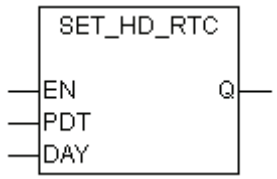
提示：

- 在调用该指令时，必须使用使能运算符调用该指令。

5.8 硬件实时时钟指令（Hollysys_PLC_HDRTC.lib）

5.8.1 Set_HD_RTC——设置实时时钟（DT 数据格式）

✧ 指令示例

梯形图语言示例


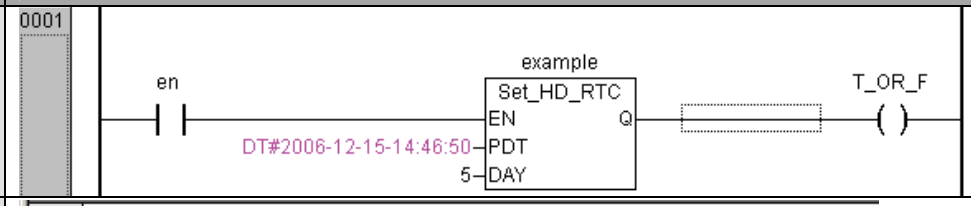
✧ 参数说明

输入参数	数据类型	功能描述	参数值说明
EN	BOOL	使能	0：无效 1：上升沿使能
PDT	DT	日期/时间	DT#2000-01-01-00:00:00 至 DT#2099-12-31-23:59:59
DAY	BYTE	星期值	1-7
输出参数	**数据类型**	**功能描述**	**参数值说明**
Q	BOOL	是否设定完成	0：未设定完成 1：设定完成

✧ 指令使用举例（梯形图和结构化文本）

变量定义

	名称	地址	类型	初始值	注释
0001	en		BOOL		
0002	T_OR_F		BOOL		
0003	example		Set_HD_RTC		

编程语言	程　序
梯形图（LD）	
结构化文本（ST）	0001 example(EN:= en,PDT:=DT#2006-12-15-14:46:50, DAY:=5); 0002 T_OR_F:=example.Q;

程序说明:

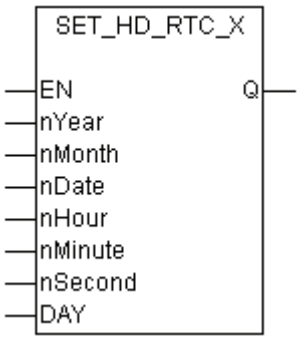
- EN 置位时，如图所示的日期、时间、星期将被设置到 PLC 硬件实时时钟之中，当前的实时时钟是：2006-12-15-14:46:50，星期五。

提示：

- 设置日期时间时不可以超出规定范围。

5.8.2 Set_HD_RTC_X——设置实时时钟（普通数据格式）

✧ **指令示例**

梯形图语言示例


✧ **参数说明**

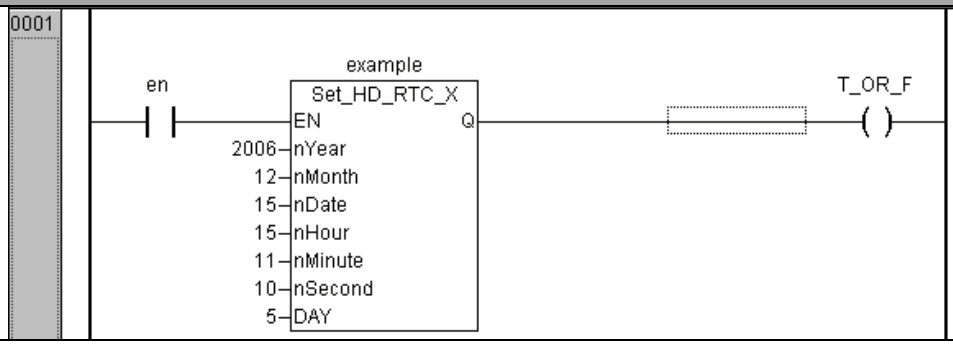
输入参数	数据类型	功能描述	参数值说明
EN	BOOL	使能	0：无效 1：上升沿使能
nYear	INT	年	2000-2099
nMonth	BYTE	月	1-12
nDate	BYTE	日	1-31
nHour	BYTE	时	0-24
nMinute	BYTE	分	0-60
nSecond	BYTE	秒	0-60
DAY	BYTE	星期	1-7
输出参数	**数据类型**	**功能描述**	**参数值说明**
Q	BOOL	是否设定完成	0：未设定完成 1：设定完成

✧ **Set_HD_RTC_X 指令举例（梯形图和结构化文本）**

变量定义					
0001	en		BOOL		
0002	T_OR_F		BOOL		
0003	example		Set_HD_RTC_X		

编程语言 | **程　序**

梯形图（LD）



结构化文本（ST）

```
0001 example(EN:= en,nYear:=2006,nMonth:=12,nDate:=15,nHour:=15, nMinute:=11,nSecond:=10,DAY:=5);
0002 T_OR_F:=example.Q;
```

程序说明:

- EN 置位时，如图所示的日期、时间、星期将被设置到 PLC 硬件实时时钟之中，当前的实时时钟是 2006 年 12 月 15 日 15 点 11 分 10 秒，星期五。

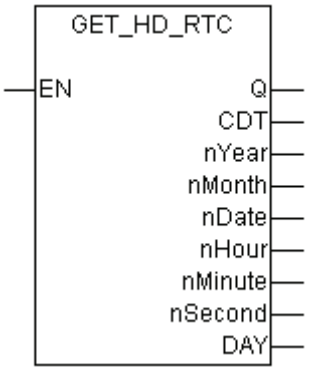
提示：

- 设置日期时间时，不可以超出规定范围。

5.8.3 Get_HD_RTC——读取实时时钟日期/时间/星期

✧ **指令示例**

梯形图语言示例



✧ 参数说明

输入参数	数据类型	功能描述	参数值说明
EN	BOOL	使能	0：无效 1：使能
输出参数	**数据类型**	**功能描述**	**参数值说明**
Q	BOOL	是否取出数据	0：未取出数据 1：正确取出数据
CDT	DT	日期/时间	DT#2000-01-01-00:00:00 至 DT#2099-12-31-23:59:59
nYear	INT	年	0-99（年份后两位）
nMonth	BYTE	月	1-12
nDate	BYTE	日	1-31
nHour	BYTE	时	0-24
nMinute	BYTE	分	0-60
nSecond	BYTE	秒	0-60
DAY	BYTE	星期	1-7

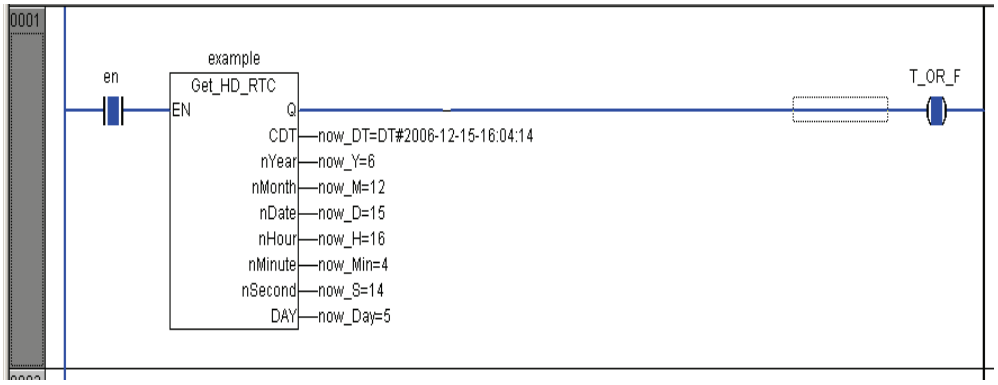
✧ Get_HD_RTC 指令举例（运行中梯形图和结构化文本）

变量定义

	名称	地址	类型	初始值
0001	en		BOOL	
0002	T_OR_F		BOOL	
0003	example		Get_HD_RTC	
0004	now_DT		DT	
0005	now_Y		INT	
0006	now_M		BYTE	
0007	now_D		BYTE	
0008	now_H		BYTE	
0009	now_Min		BYTE	
0010	now_S		BYTE	
0011	now_Day		BYTE	

编程语言 | **程 序**

梯形图（LD）



结构化文本（ST）

```
0001 example(EN:= en);
0002 T_OR_F:=example.Q;
0003 now_DT:=example.CDT;
0004 now_Y:=example.nYear;
0005 now_M:=example.nMonth;
0006 now_D:=example.nDate;
0007 now_H:=example.nHour;
0008 now_Min:=example.nMinute;
0009 now_S:=example.nSecond;
0010 now_Day:=example.DAY;
```

```
en = TRUE
T_OR_F = TRUE                 example.Q = TRUE
now_DT = DT#2006-12-15-16:06...  example.CDT = DT#2006-12-15-...
now_Y = 6                     example.nYear = 6
now_M = 12                    example.nMonth = 12
now_D = 15                    example.nDate = 15
now_H = 16                    example.nHour = 16
now_Min = 6                   example.nMinute = 6
now_S = 9                     example.nSecond = 9
now_Day = 5                   example.DAY = 5
```

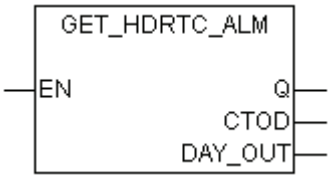
程序说明:

- EN 置位时，CDT 以 DT 型数据格式输出 PLC 实时时钟的日期、时间，nYear 以 INT 型数据格式输出年，nMonth、nDate、nHour、nMinute、nSecond、DAY 分别以 BYTE 型数据格式输出月、日、时、分、秒、星期，Q 等于 TRUE，当前的实时时钟是：2006-12-15-16:04:14，星期五。
- EN 复位时，Q 等于 FALSE，CDT、nYear、nMonth、nDate、nHour、nMinute、nSecond、DAY 保持最后一次输出值不变。

5.9 实时时钟报警指令（Hollysys_PLC_HDRTCALM.lib）

5.9.1 Get_HDRTC_ALM——读取硬件实时时钟报警时间/星期

✧ 指令示例

梯形图语言示例


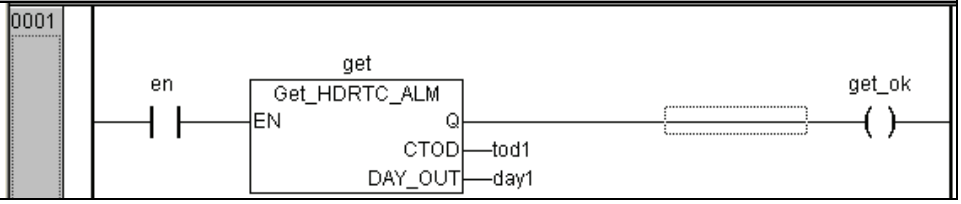
✧ 参数说明

输入参数	数据类型	功能描述	参数值说明
EN	BOOL	使能	0：无效 1：使能
输出参数	**数据类型**	**功能描述**	**参数值说明**
Q	BOOL	是否取出报警数据	0：未读出报警数据 1：正确取出报警数据
CTOD	TOD	当前报警时间	
DAY_OUT	BYTE	当前报警星期值	1-7

✧ 指令使用举例（梯形图和结构化文本）

变量定义

	名称	地址	类型	初始值	注释
0001	en		BOOL		
0002	tod1		TOD		
0003	day1		BYTE		
0004	get		Get_HDRTC_ALM		
0005	get_ok		BOOL		

编程语言	程　序
梯形图（LD）	

结构化文本（ST）	0001 get(EN:= en); 0002 get_ok:=get.Q; 0003 tod1:=get.CTOD; 0004 day1:=get.DAY_OUT;

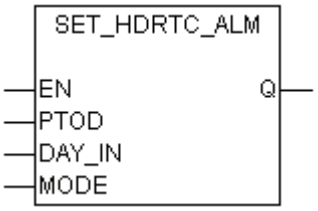
程序说明：

- EN 置位时，CTOD 输出 PLC 的实时时钟报警时间，DAY_OUT 输出 PLC 的实时时钟报警星期，Q 等于 TRUE，当前的报警时间是 15:21:22，星期是星期一。
- EN 复位时，Q 等于 FALSE，CTOD 和 DAY_OUT 保持最后一次输出值不变。

5.9.2 Set_HDRTC_ALM——设置硬件实时时钟报警时间/星期

✧ **指令示例**

梯形图语言示例



✧ **参数说明**

输入参数	数据类型	功能描述	参数值说明
EN	BOOL	使能	0：无效 1：上升沿使能
PTOD	TOD	时间	
DAY_IN	BYTE	星期值	1-7
MODE	BYTE	报警模式	0：触发报警事件关闭 1：当星期、小时、分钟、秒匹配时， 触发报警事件（每星期触发一次） 2：当小时、分钟、秒匹配时，触发报警事件（每天触发一次） 3：当分钟、秒匹配时，触发报警事件 （每小时触发一次） 4：当秒匹配时，触发报警事件 （每分钟触发一次） 5：每秒触发报警事件一次
输出参数	**数据类型**	**功能描述**	**参数值说明**
Q	BOOL	是否正确地 设定报警数据	0：未设定报警数据 1：正确设定报警数据

✧ **报警关联事件**

HD_RTC_ALM 0 interrupt。

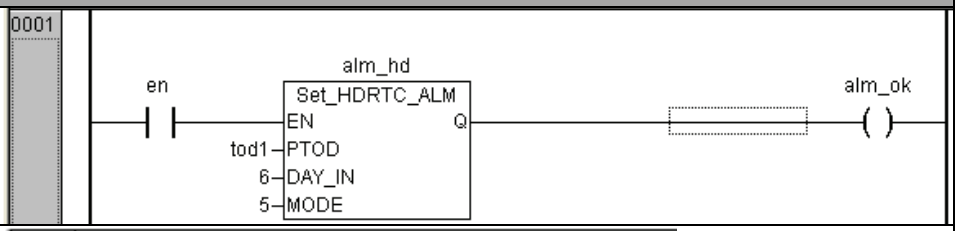
✧ **指令使用举例（梯形图和结构化文本）**

变量定义

	名称	地址	类型	初始值	注释
0001	en		BOOL		
0002	tod1		TOD		
0003	alm_hd		Set_HDRTC_ALM	T#12：05：0(	
0004	alm_ok		BOOL		

编程语言 | **程 序**

梯形图（LD）



结构化文本（ST）

```
0001 alm_hd(EN:= en,PTOD:=TOD1,DAY_IN:=6,MODE:=5);
0002 alm_ok:=alm_hd.Q;
```

程序说明:

- EN 置位时，如图所示设置 PLC 的实时报警时间/星期，Q 等于 TRUE，当前设置的报警时间是 12：05：00，星期是星期六。因为选择模式 5，所以每秒钟触发一次报警，时间和星期设置就不起作用了。
- EN 复位时，Q 等于 FALSE。

提示：

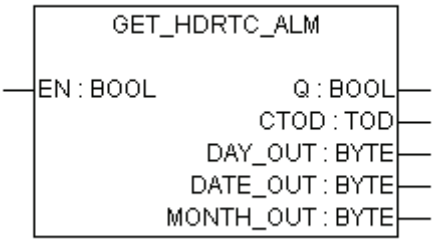
- 每次报警事件触发后应重新读取一次硬件实时时钟报警数据（即要调用一次 Get_HDRTC_ALM 指令），否则不能启动下一次报警。
- 中断关联事件的使用见附录 B。

5.10 LM3104/5 实时时钟报警指令（Hollysys_PLC_HDRTCALM_N.lib）

5.10.1 GET_HDRTC_ALM——获取实时时钟中断时间

✧ **指令示例**

梯形图语言示例



✧ 参数说明

输入参数	数据类型	功能描述	参数值说明
EN	BOOL	使能	0：无效 1：使能
输出参数	**数据类型**	**功能描述**	**参数值说明**
Q	BOOL	是否取出中断数据	0：未取出中断数据 1：正确取出中断数据
CTOD	TOD	当前中断时间	
DAY_OUT	BYTE	当前中断星期	1-7
DATE_OUT	BYTE	当前中断日期	1-31
MONTH_OUT	BYTE	当前中断月份	1-12

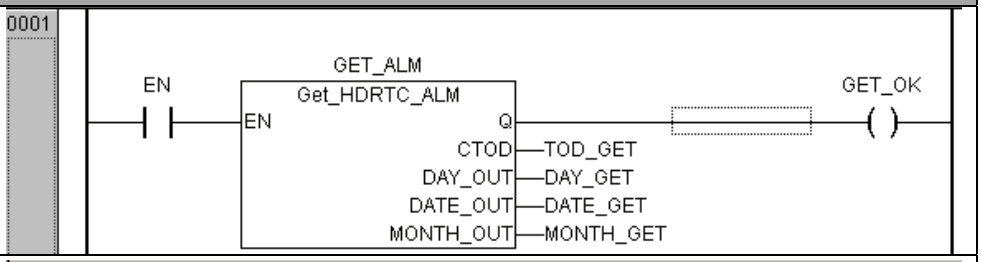
✧ 指令使用举例（运行中梯形图和结构化文本）

变量定义

	名称	地址	类型	初始值	注释
0001	EN		BOOL		
0002	GET_OK		BOOL		
0003	GET_ALM		Get_HDRTC_ALM		
0004	TOD_GET		BYTE		
0005	DAY_GET		BYTE		
0006	DATE_GET		BYTE		
0007	MONTH_GET		BYTE		

编程语言 | **程 序**

梯形图（LD）



结构化文本（ST）

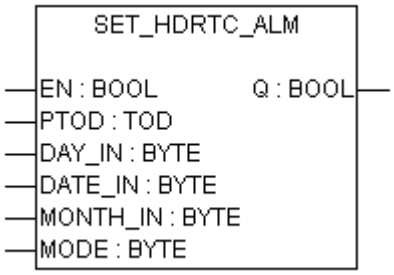
```
0001 GET_ALM(EN:= EN);
0002 GET_OK:=GET_ALM.Q;
0003 TOD_GET:=GET_ALM.CTOD;
0004 DAY_GET:=GET_ALM.DAY_OUT;
0005 DATE_GET:=GET_ALM.DATE_OUT;
0006 MONTH_GET:=GET_ALM.MONTH_OUT;
```

程序说明:

- EN 置位时，CTOD 以 TOD 型数据格式输出 PLC 实时时钟报警的时间，MONTH_OUT、DATE_OUT、DAY_OUT 分别以 BYTE 型数据格式输出实时时钟报警的月、日、星期，Q 等于 TRUE。
- EN 复位时，Q 等于 FALSE，CTOD、MONTH_OUT、DATE_OUT、DAY_OUT 保持最后一次输出值不变。

5.10.2 SET_HDRTC_ALM——设置实时时钟中断时间

✧ 指令示例

梯形图语言示例


✧ 参数说明

输入参数	数据类型	功能描述	参数值说明
EN	BOOL	使能	0：无效 1：上升沿使能
PTOD	TOD	时间	
DAY_IN	BYTE	星期值	1-7
DATE_IN	BYTE	日期	1-31
MONTH_IN	BYTE	月份	1~12
MODE	BYTE	中断模式	详见表 5-10-1
输出参数	**数据类型**	**功能描述**	**参数值说明**
Q	BOOL	是否正确地 设定中断数据	0：未设定中断数据 1：正确设定中断数据

表 5-10-1

7	6	5	4	3	2	1	0
无意义	无意义	0：禁止星期匹配 1：星期匹配	0：禁止月份匹配 1：月份匹配	0：禁止日期匹配 1：日期匹配	0：禁止小时匹配 1：小时匹配	0：禁止分匹配 1：分匹配	0：禁止秒匹配 1：秒匹配

如果设置中断模式为 2#00000011，则每次当前时钟和设定时钟的分和秒匹配时产生中断。

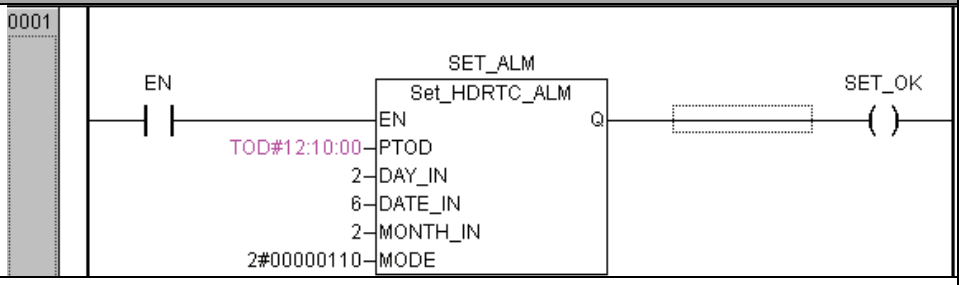
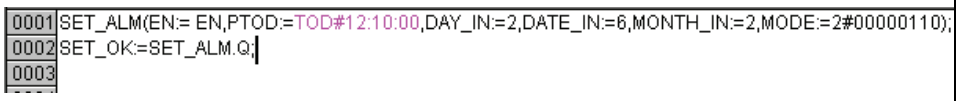
✧ 报警关联事件

- HD_RTC_ALM 0 interrupt。

✧ 指令使用举例（梯形图和结构化文本）

变量定义

	名称	地址	类型	初始值	注释
0001	EN		BOOL		
0002	SET_OK		BOOL		
0003	SET_ALM		Set_HDRTC_ALM		

编程语言	程　序
梯形图（LD）	
结构化文本（ST）	


```
0001 SET_ALM(EN:= EN,PTOD:=TOD#12:10:00,DAY_IN:=2,DATE_IN:=6,MONTH_IN:=2,MODE:=2#00000110);
0002 SET_OK:=SET_ALM.Q;
0003
```

程序说明:

- EN 置位时，如图所示设置 PLC 的实时报警时间，Q 等于 TRUE，当前设置的报警时间是 2 月 6 号的 12：10：00，星期是星期二。因为选择模式为 2#00000110，所以每天 12 点 10 分触发一次报警，月、日、秒和星期设置就不起作用了。
- EN 复位时，Q 等于 FALSE。

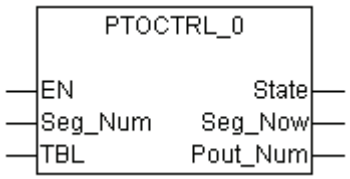
提示：

- 中断关联事件的使用见附录 B。

5.11 多段脉冲发送（Hollysys_PLC_PTOCtrl.lib）

5.11.1 PTOCtrl_0——通道 1.1 多段脉冲发送

✧ **指令示例**

梯形图语言示例


✧ **适用模块**

LM3106、LM3106A、LM3108。

✧ **功能描述**

该指令提供指定脉冲个数的方波（50%的占空比）脉冲串发生功能。周期以微秒为单位（范围是 20 到 335000），如果设定的周期值为奇数，占空比会有少许失真，脉冲数输出通道为 Q1.1，范围是 0 到 4294967295。

✧ 参数说明

外部端子输出	功能描述		
Q1.1	高速脉冲输出		
输入参数	**数据类型**	**功能描述**	**参数值说明**
EN	BOOL	使能	0：无效 1：使能
Seg_Num	BYTE	段数	1-255
TBL	INT	M 区内段表首地址	100-8000
输出参数	**数据类型**	**功能描述**	**参数值说明**
State	BOOL	脉冲发送状态	0：停止发送 1：正在发送
Seg_Now	BYTE	当前发送的段号	1-255
Pout_Num	DWORD	已发送的脉冲数	0-4294967295

✧ 关联冲突指令

使用指令 A	关联指令 B	关联硬件	可使用程度
PTOCtrl_0	PTO_PWM0	T12	不可使用指令 B
	PTO_PWM0_Run		

✧ **PTOCtrl_0 指令举例（梯形图和结构化文本）**

比如，Q1.1 按照表 5-11-1 所定义的参数发送脉冲，该表中共有 4 段，每段包含 1 个 32 位的周期起始值，1 个 16 位的周期增量，1 个 32 位的该段发送脉冲数，共计 10 个字节。周期的修改是在每个脉冲上进行的。

表 5-11-1 脉冲发送参数表

段号	起始周期（μs，32 位）	周期增量（μs，16 位）	脉冲数（32 位）
#1	%MD200	%MW204	%MD206
	500	-2	100
#2	%MD210	%MW214	%MD216
	300	-3	50
#3	%MD220	%MW224	%MD226
	150	0	500
#4	%MD230	%MW234	%MD236
	150	5	100

上表对应的时间-频率如图 5-11-1。

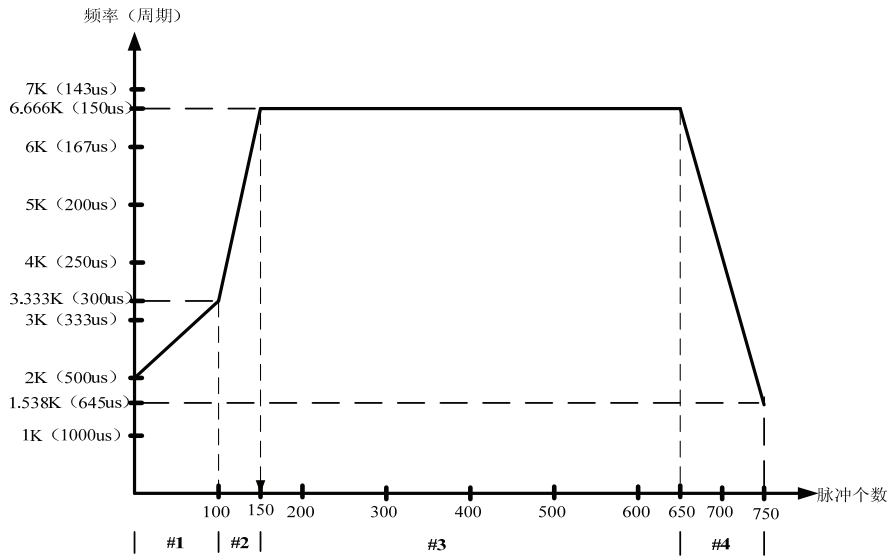


图 5-11-1 步进电机应用的时间-频率图

起始周期、周期增量、脉冲数对应关系：

给定段的周期增量=（ECT-ICT）/Q

ECT=该段结束的周期值（即下段开始的周期值）

ICT=该段初始的周期值

Q=该段上发送的脉冲数

即：# n 周期增量=（# n+1 起始周期—# n 起始周期）/# n 脉冲数

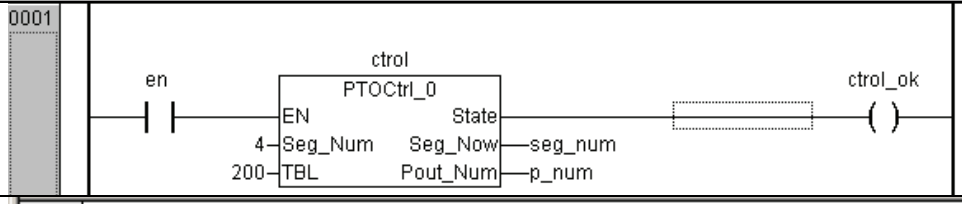
比如：#1 周期增量=（#2 起始周期—#1 起始周期）/#1 脉冲数。

变量定义

	名称	地址	类型	初始值	注释
0001	ctrol_ok		BOOL		
0002	seg_num		BYTE		
0003	p_num		DWORD		
0004	en		BOOL		
0005	q11	%QX1.1	BOOL		
0006	ctrol		PTOCtrl_0		
0007	PTOSeg	%MW200	ARRAY [1..4] OF PTOCtrlSet	(P:=500,I:=-2,N:=100), (P:=300,I:=-3,N:=50), (P:=150,I:=0,N:=500), (P:=150,I:=5,N:=100);	

编程语言 | **程 序**

梯形图（LD）



结构化文本（ST）

```
0001 ctrol(EN:= en,Seg_Num:=4,TBL:=200);
0002 ctrol_ok:=ctrol.State;
0003 seg_num:=ctrol.Seg_Now;
0004 p_num:=ctrol.Pout_Num;
```

程序说明：

- 当 EN 使能后，CPU 自动从 M 区指定的地址（比如上表中的起始地址为%MW200）开始读入段表中每段的特性，Q1.1 将以 500μs 的脉宽开始发送，每个脉冲的脉宽减少 2μs，直到发送完 100 个脉冲，第一段发送完毕，此时脉宽为 302μs，紧接着以 300μs 的脉宽开始进入第二段，每个脉冲的脉宽减少 3μs，发送完 50 个脉冲之后的脉宽为 153μs，此时以 150μs 的脉宽开始进入第三段，第三段的脉冲增量是 0，所以此段发送脉冲的脉宽均为 150μs，共计发送 500 个，然后以 150μs 的脉宽进入第四段，每个脉冲的脉宽增加 5μs，发送完 100 个脉冲，以 645μs 的脉宽停止。
- EN 使能后，该指令执行，按照步进电机应用的时间-频率图所示的曲线进行脉冲发送。Seg_Num 显示当前进行到的段号，Pout_Num 显示已经发送的脉冲数。
- EN 复位时，所有变量复位。

✧ **段表定义方法描述：**

M 区内段表的定义方法有 2 种，一种是利用 HOLLYSYS_PLC_PTOCtrl.lib 库中 PTOCtrlSet 的结构定义一个结构数组，或者自定义结构数组，对此结构数组进行赋值，另一种是依次给 M 区内的地址赋值。

HOLLYSYS_PLC_PTOCtrl.lib 库中的结构如下：

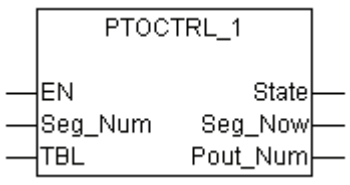
```
TYPE PTOCtrlSet :
STRUCT
    P: DWORD;
    I: INT;
    N: DWORD;
END_STRUCT
END_TYPE
```

在上表例子中，可以采用如下方式在%MW200 处定义一个结构数组，将表中的值依次填入即可。

```
PTOSeg AT %MW200: ARRAY [1..4] OF PTOCtrlSet:= (P:=500,I:=-2,N:=100),
(P:=300,I:=-3,N:=50),(P:=150,I:=0,N:=500),(P:=150,I:=5,N:=100);
```

5.11.2 PTOCtrl_1——通道 0.3 多段脉冲发送

✧ **指令示例**

梯形图语言示例


✧ **适用模块**

LM3104、LM3106、LM3106A、LM3108

✧ **功能描述**

与 PTOCtrl_0 的功能一样，区别在于周期以微秒为单位（范围是 20 到 5327），脉冲输出通道为 Q0.3。

✧ **参数说明**

外部端子输出	**功能**		
Q0.3	高速脉冲输出		
输入	**数据类型**	**功能描述**	**参数值说明**
EN	BOOL	使能	0：无效 1：使能
Seg_Num	BYTE	段数	1-255
TBL	INT	M 区内段表首地址	100-8000
输出	**数据类型**	**功能描述**	**参数值说明**
State	BOOL	脉冲发送状态	0：停止发送 1：正在发送
Seg_Now	BYTE	当前发送的段号	1-255
Pout_Num	DWORD	已发送的脉冲数	0-4294967295

✧ **关联冲突指令**

使用指令 A	**关联指令 B**	**关联硬件**	**可使用程度**
PTOCtrl_1	PTO_PWM1	T13	不可使用指令 B
	PTO_PWM1_Run		

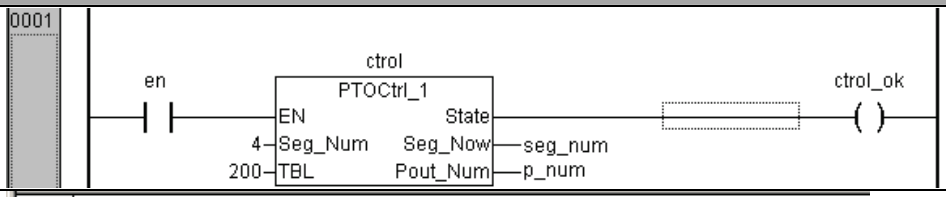
✧ **指令使用举例（梯形图和结构化文本）**

变量定义

	名称	地址	类型	初始值	注释
0001	ctrol_ok		BOOL		
0002	seg_num		BYTE		
0003	p_num		DWORD		
0004	en		BOOL		
0005	q03	%QX0.3	BOOL		
0006	ctrol		PTOCtrl_1		
0007	PTOSeg	%MW200	ARRAY [1..4] OF PTOCtrlSet	(P:=500,I:=-2,N:=100), (P:=300,I:=-3,N:=50), (P:=150,I:=0,N:=500), (P:=150,I:=5,N:=100);	

编程语言 | **程　序**

梯形图（LD）



结构化文本（ST）

```
0001 ctrol(EN:= en,Seg_Num:=4,TBL:=200);
0002 ctrol_ok:=ctrol.State;
0003 seg_num:=ctrol.Seg_Now;
0004 p_num:=ctrol.Pout_Num;
```

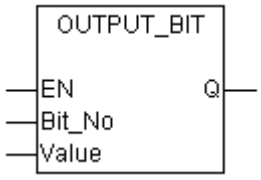
程序说明：

- EN 使能后，该指令执行，按照上图所示的曲线进行脉冲发送。Seg_Num 显示当前进行到的段号，P_Num 显示已经发送的脉冲数。
- EN 复位时，所有变量复位。
- M 区内段表定义方法描述与 PTOCtrl_0 一样。

5.12 立即输出指令（Hollysys_PLC_IO.lib）

5.12.1 OutPut_Bit——立即输出

✧ 指令示例

梯形图语言示例


✧ 功能描述

该指令在程序中立即输出通道的状态，而不需要等到一个扫描周期结束之后再刷新通道的状态。

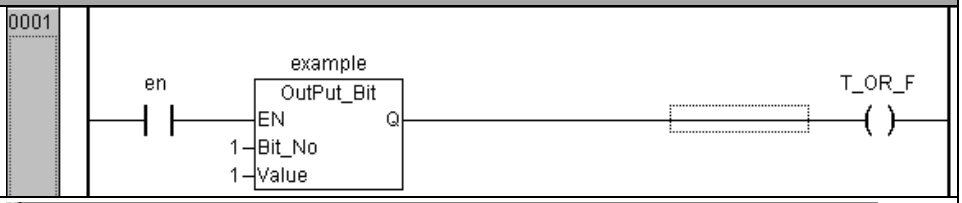
✧ 参数说明

输入参数	数据类型	功能描述	参数值说明
EN	BOOL	使能	0：无效 1：使能
Bit_No	BYTE	要立即输出的通道号	0-9（对应于 Q0.0-Q1.1）
Value	BOOL	通道值	0：Bit_No 对应的通道置 0 1：Bit_No 对应的通道置 1
输出参数	**数据类型**	**功能描述**	**参数值说明**
Q	BOOL	通道是否已输出	0：未输出 1：已输出

✧ 指令使用举例（梯形图和结构化文本）

变量定义

	名称	地址	类型	初始值	注释
0001	en		BOOL		
0002	example		OutPut_Bit		
0003	T_OR_F		BOOL		

编程语言	程　序
梯形图（LD）	
结构化文本（ST）	0001 example(EN:= en,Bit_No:=1,Value:=1); 0002 T_OR_F:=example.Q;

程序说明:

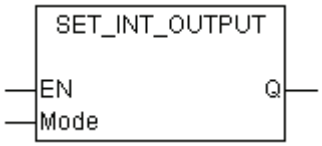
- EN 置位时，该指令使能，将 Q0.1 输出通道置 1。EN 复位时，该指令不起作用，扫描周期结束后输出该通道的值。

提示：

- 如果在中断中使用该指令立即输出某通道的值，需要使用 Set_INT_OutPut 指令在主程序中进行设置，使该通道在扫描周期结束后不输出，防止与中断中的立即输出发生冲突。

5.12.2 Set_INT_OutPut——设置中断立即输出

✧ **指令示例**

梯形图语言示例


✧ **功能描述**

该指令执行后，其 Mode 处设置的相应通道不会在扫描周期结束后更新，需使用 OutPut_Bit 指令更新通道值。

✧ **参数说明**

<table>
<tr><th>输入参数</th><th>数据类型</th><th>功能描述</th><th colspan="8">参数值说明</th></tr>
<tr><td>EN</td><td>BOOL</td><td>使能</td><td colspan="8">0：无效
1：使能</td></tr>
<tr><td rowspan="6">Mode</td><td rowspan="6">WORD</td><td rowspan="6">不需要更新的物理通道</td><td colspan="8">Bit 0-7</td></tr>
<tr><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td>Q0.7</td><td>Q0.6</td><td>Q0.5</td><td>Q0.4</td><td>Q0.3</td><td>Q0.2</td><td>Q0.1</td><td>Q0.0</td></tr>
<tr><td colspan="8">Bits 8-15</td></tr>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td></tr>
<tr><td colspan="6">无效</td><td>Q1.1</td><td>Q1.0</td></tr>
<tr><td colspan="8">当 Mode 中某位设为 1 时，其对应的通道不会在扫描周期结束后进行更新，需调用 OutPut_Bit 对其通道更新</td></tr>
<tr><th>输出参数</th><th>数据类型</th><th>功能描述</th><th colspan="8">参数值说明</th></tr>
<tr><td>Q</td><td>BOOL</td><td>是否设置完毕</td><td colspan="8">0：没有设置完毕　　1：设置完毕</td></tr>
</table>

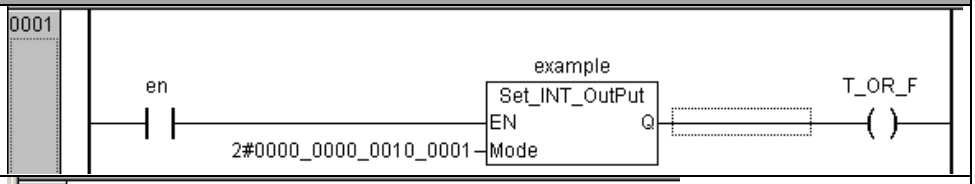
✧ 指令使用举例（梯形图和结构化文本）

变量定义

	名称	地址	类型	初始值	注释
0001	en		BOOL		
0002	example		Set_INT_OutPut		
0003	T_OR_F		BOOL		

编程语言 | **程 序**

梯形图（LD）



结构化文本（ST）

```
0001 example(EN:= en,Mode:=2#0000_0000_0010_0001);
0002 T_OR_F:=example.Q;
```

程序说明：

- EN 置位时，Mode 值为 2#0000000000100001，右边第 1 位与第 6 位为 1，则对应的 Q0.0 与 Q0.5 在每个扫描周期结束后不会自动更新，需在程序中调用 OutPut_Bit 指令对该通道进行更新。
- EN 复位时，该指令不执行，每个扫描周期结束后正常输出每个通道值。

提示：

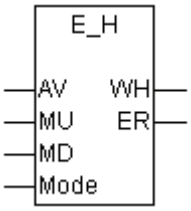
- 一般在中断中使用 OutPut_Bit 指令时，要在主程序中使用 Set_INT_OutPut 指令。如果主程序中不使用此指令对中断中涉及到的立即输出通道进行屏蔽，有可能产生冲突，比如一个扫描周期结束的时候，正在更新物理点，此时产生中断，调用 OutPut_Bit 功能块进行立即输出，会引起输出冲突。

5.13 工程量转换指令（Hollysys_PLC_Cnvt.lib）

5.13.1 E_H——工程量数据转换为 16 进制数

✧ 指令示例

梯形图语言示例



✧ 参数说明

输入参数	数据类型	功能描述	参数值说明
AV	REAL	工程量数据	
MU	REAL	工程量量程上限	
MD	REAL	工程量量程下限	
Mode	BYTE	转换模式	0：正常模式，MD-MU 对应 0-65535 1：电流 0-20mA/4-20mA 对应 0-3277（用于 LM3320） 2：电压 0-10V 对应 0-1638（用于 LM3320）
输出参数	**数据类型**	**功能描述**	**值**
WH	WORD	输出 16 进制数据	
ER	BOOL	错误代码	0：正确 1：错误

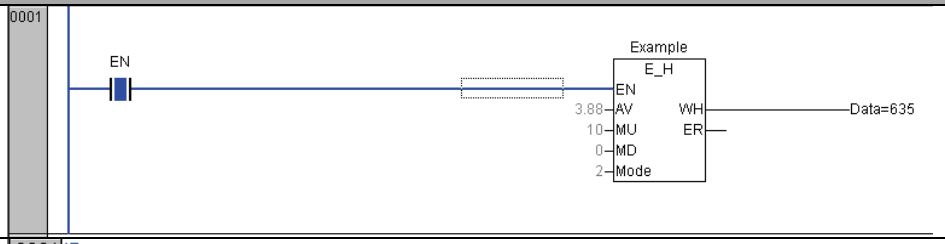
✧ 指令使用举例（梯形图和结构化文本）

变量定义

	名称	地址	类型	初始值	注释
0001	EN		BOOL		
0002	T_OR_F		BOOL		
0003	Example		E_H		
0004	Data		WORD		

编程语言 | **程 序**

梯形图（LD）



结构化文本（ST）

```
0001 IF
0002 en=1
0003 THEN
0004 example(av:=3.88,mu:=10,md:=0,mode:=2);
0005 data:=example.WH;
0006 ELSE
0007 aa:=1;
0008 END_IF
```

程序说明：

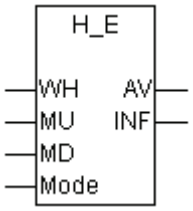
- EN 置位时，进行转换，工程量数据为 3.88V，输入上限是 10V，输入下限是 0V，选择模式 2，则对应于 0-1638（LM3320），此时输出值为 635。
- EN 复位时，输出值 635 保持不变。

提示：

- 必须通过使能运算符调用此指令。

5.13.2 H_E——16 进制数转换为工程量数据

✧ 指令示例

梯形图语言示例


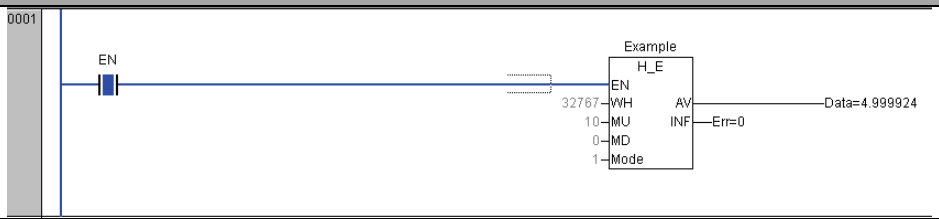
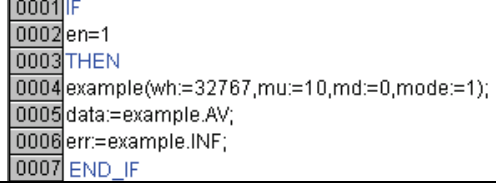
✧ 参数说明

输入参数	数据类型	功能描述	参数值说明
WH	WORD	输入 16 进制数据	
MU	REAL	工程量量程上限	
MD	REAL	工程量量程下限	
Mode	BOOL	电信号类型	0：电流，对应 0-65535 1：电压，对应 0-65535
输出参数	**数据类型**	**功能描述**	**值**
AV	REAL	输出工程量数据	
INF	BYTE	信息代码	0：正常 1：故障

✧ 指令使用举例（梯形图和结构化文本）

变量定义

	名称	地址	类型	初始值	注释
0001	EN		BOOL		
0002	Example		H_E		
0003	T_OR_F		BOOL		
0004	Data		REAL		
0005	Err		BYTE		

编程语言	程　序
梯形图（LD）	
结构化文本（ST）	


```
0001 IF
0002 en=1
0003 THEN
0004 example(wh:=32767,mu:=10,md:=0,mode:=1);
0005 data:=example.AV;
0006 err:=example.INF;
0007 END_IF
```

程序说明：

- EN 置位时，进行转换，输入数据为 32767，工程量上限是 10V，工程量下限是 0V，选择模式 1，则对应于 0-65535，此时输出值为 4.999924。
- EN 复位时，输出值 4.999924 保持不变。

提示：

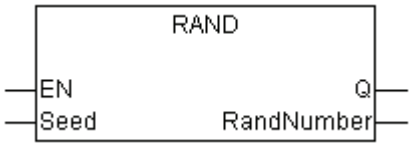
- 必须通过使能运算符调用此指令。

5.14 随机数发生指令（Hollysys_PLC_Math.lib）

Rand—随机数发生

✧ **指令示例**

梯形图语言示例



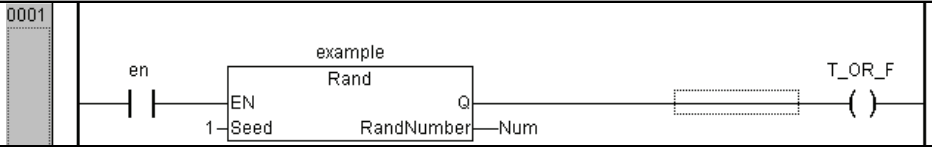
✧ **参数说明**

输入参数	数据类型	功能描述	参数值说明
EN	BOOL	使能	0：无效 1：使能
Seed	UINT	种子	0-65535
输出参数	**数据类型**	**功能描述**	**值**
Q	BOOL	是否有效数据	0：无效数据 1：有效数据
RandNumber	UINT	随机数	0-65535

✧ **指令使用举例（梯形图和结构化文本）**

变量定义

	名称	地址	类型	初始值	注释
0001	en		BOOL		
0002	example		RAND		
0003	T_OR_F		BOOL		
0004	Num		UINT		

编程语言	程　序
梯形图（LD）	

结构化文本（ST）	0001 example(en:=en,seed:=1); 0002 Num:=example.RandNumber; 0003 T_OR_F:=example.q;

程序说明:

- EN 置位时，产生随机数，每个扫描周期产生 1 个随机数。
- EN 复位时，停止产生随机数，RandNumber 保持最后一次产生的随机数不变。

5.15 Modbus 从站地址指令（Hollysys_PLC_Ex.lib）

5.15.1 SET_LOCAL_ADDRESS——设置 Modbus 从站通讯地址

✧ 指令示例

梯形图语言示例

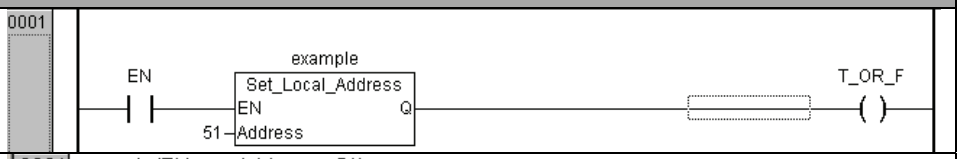
✧ 参数说明

输入参数	数据类型	功能描述	参数值说明
EN	BOOL	使能	0：无效 1：使能（上电或下装后首次置位指令有效）
Address	BYTE	Modbus 从站地址	1-247：从站号
输出参数	**数据类型**	**功能描述**	**参数值说明**
Q	BOOL	是否设置完毕	0：EN 复位 1：设置完毕，保持

✧ 指令使用举例（梯形图和结构化文本）

变量定义

	名称	地址	类型	初始值	注释
0001	EN		BOOL		
0002	example		Set_Local_Address		
0003	T_OR_F		BOOL		

编程语言	程　序
梯形图（LD）	
结构化文本（ST）	0001 example(EN:=en,Address:=51); 0002 T_OR_F:=example.q;

程序说明:

- 上电或者下装后，EN 首次置位时，该指令使能，从站设置完毕后，Q 等于 TRUE。
- EN 复位时，Q 等于 FALSE，但从站地址为先前设定值，不会改变。

提示：

- 如果所需通讯参数不是 38400bps、8 位、无校验，则需要使用 Reset_COMM_PRMT/Reset_COMM2_PRMT 指令设置相应的通讯参数，然后再使用 SET_LOCAL_ADDRESS 功能块设置站地址，建立通讯。此时，如果下装程序，需要将 RUN/STOP 开关拨到 STOP 位置，程序下装后再将 RUN/STOP 开关拨到 RUN 位置运行程序。

5.15.2 GET_LOCAL_ADDRESS——读取 Modbus 从站通讯地址

✧ 指令示例

梯形图语言示例

✧ 参数说明

输入参数	数据类型	功能描述	参数值说明
EN	BOOL	使能	0：无效 1：使能
输出参数	**数据类型**	**功能描述**	**值**
Q	BOOL	是否读出从站地址	0：EN 复位 1：读出 Modbus 从站地址，保持
Address	BYTE	Modbus 从站地址	1-247：从站号

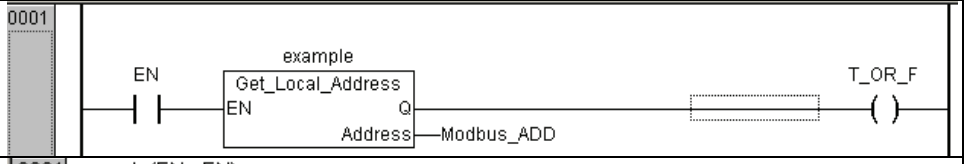
✧ 指令使用举例（梯形图和结构化文本）

变量定义

	名称	地址	类型	初始值	注释
0001	EN		BOOL		
0002	example		Get_Local_Address		
0003	T_OR_F		BOOL		
0004	Modbus_ADD		BYTE		

编程语言 | **程 序**

梯形图（LD）



结构化文本（ST）

```
0001 example(EN:=EN);
0002 T_OR_F:=example.Q;
0003 Modbus_ADD:=example.Address;
```

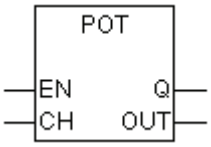
程序说明：

- EN 置位时，该指令使能，读取本机 Modbus 从站地址，Q 等于 TRUE。
- EN 复位时，Q 等于 FALSE，但 Address 输出值保持原读取值。

5.16 模拟电位器（Hollysys_PLC_Ex.lib）

POT—读取模拟电位器

✧ **指令示例**

梯形图语言示例


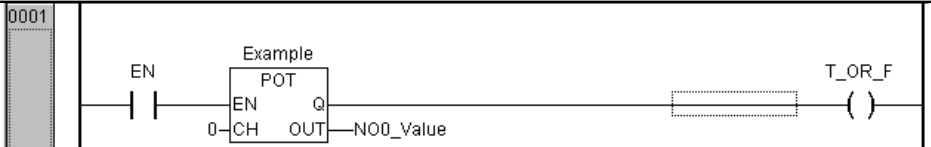
✧ **参数说明**

输入参数	数据类型	功能描述	参数值说明
EN	BOOL	使能	0：无效 1：使能
CH	BYTE	模拟电位器通道号	0：通道 0 1：通道 1
输出参数	**数据类型**	**功能描述**	**值**
Q	BOOL	是否已读入数据	0：EN 复位或数据未读入 1：读入数据，保持
OUT	BYTE	模拟电位器当前值	0-255

✧ **指令使用举例（梯形图和结构化文本）**

变量定义

	名称	地址	类型	初始值	注释
0001	EN		BOOL		
0002	Example		POT		
0003	T_OR_F		BOOL		
0004	NO0_value		BYTE		

编程语言	程　序
梯形图（LD）	
结构化文本（ST）	0001 Example(EN:=EN,CH:=0); 0002 NO0_Value:=Example.OUT; 0003 T_OR_F:=Example.Q;

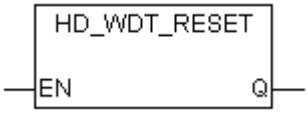
程序说明：

- EN 置位时，该指令使能，读到数值后，Q 输出 TRUE，并保持，OUT 输出值为第 0 通道电位器的值，当电位器右旋至最大位置，OUT 值等于 255。
- EN 复位时，Q 等于 FALSE，OUT 的输出保持最后一次读入的电位器值。

5.17 系统看门狗复位（Hollysys_PLC_Ex.lib）

HD_WDT_Reset—系统看门狗复位

✧ **指令示例**

梯形图语言示例


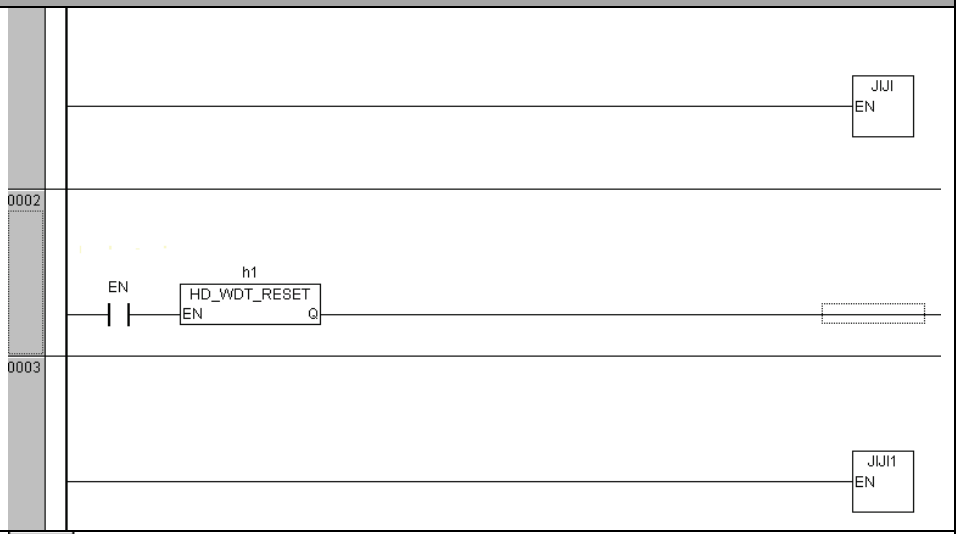
✧ **参数说明**

输入参数	数据类型	功能描述	参数值说明
EN	BOOL	使能	0：无效 1：使能
输出参数	**数据类型**	**功能描述**	**值**
Q	BOOL	是否复位完毕	0：EN 复位 1：复位完毕，保持

✧ **指令使用举例（梯形图和结构化文本）**

变量定义

	名称	地址	类型	初始值
0001	EN		BOOL	
0002	h1		HD_WDT_Reset	

编程语言	程　序
梯形图（LD）	
结构化文本（ST）	0001 jiji; 0002 h1(en:=1); 0003 jiji1;

程序说明:

- 本程序中调用两个子程序，两个程序都是执行几万次的 FOR 循环。如果两个子程序间没有看门狗指令，或者 EN 为 FALSE 时，PLC 就会因为程序扫描周期过长而死机。
- 在程序中间加看门狗复位的目的在于清掉 PLC 硬件定时器中记录的时间，这样系统看门狗就不会报警。

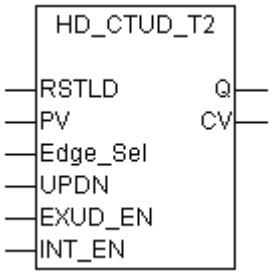
提示：

- 看门狗报警的现象是 PLC 错误灯闪亮，运行指示灯在错误灯闪亮的同时间隔性闪烁。不过这条指令一般用不到，因为一般程序的扫描周期都不会很长，在几十毫秒以内。PLC 硬件时钟记录到 300ms，系统看门狗就会报警，程序停止运行。

5.18 单相计数(Hollysys_PLC_Ex_CT.lib)

5.18.1 HD_CTUD_T2——T2 高速计数器

✧ **指令示例**

梯形图语言示例


✧ **参数说明**

通道说明			
I0.0	外部高速计数脉冲输入		
I0.1	外部高速方向控制输入		
输入参数	**数据类型**	**功能描述**	**参数值说明**
RSTLD	BOOL	使能	0：无效 1：上升沿使能
PV	UINT	计数设定值	0—65535
Edge_Sel	BYTE	I0.0 输入脉冲 触发边沿选择	0：禁止 1：上升沿 2：下降沿 3：双边沿
UPDN	BOOL	两个功能： 1.增/减计数选择（EXUD_EN=0 时有效） 2.初始化时复位/装载	0：减计数/装载 CV=PV 1：增计数/复位 CV=0
EXUD_EN	BOOL	外部 I0.1 方向控制使能 I0.1 高电平：增计数 I0.1 低电平：减计数	0：I0.1 方向控制禁止 1：I0.1 方向控制使能 （此时 UPDN 不控制增减计数）
INT_EN	BOOL	计数值到达中断使能 UPDN=0：CV=0 时产生中断 UPDN=1：CV=PV 时产生中断	0：T2 计数值到达中断禁止 1：T2 计数值到达中断使能

输出参数	数据类型	功能描述	值
Q	BOOL	计数值到达标志	0：当前计数值 CV≤PV（UPDN=1）或者 CV≥0（UPDN=0） 1：当前计数值 CV≥PV（UPDN=1）或者 CV≤0（UPDN=0）
CV	UINT	当前计数值	0—65535

✧ 关联冲突指令

使用指令 A	关联指令 B	关联硬件	可使用程度
HD_CTUD_T2	HD_DCTUD_T2	T2 内部计数器、 I0.0、I0.1 脉冲输入	不可使用指令 B

三个单相高速计数器占用端口说明

高速计数器	外部脉冲输入	外部方向控制输入
HD_CTUD_T2	I0.0	I0.1
HD_CTUD_T3	I0.2	I0.3
HD_CTUD_T7	I0.6	—

✧ 指令使用举例（梯形图和结构化文本）

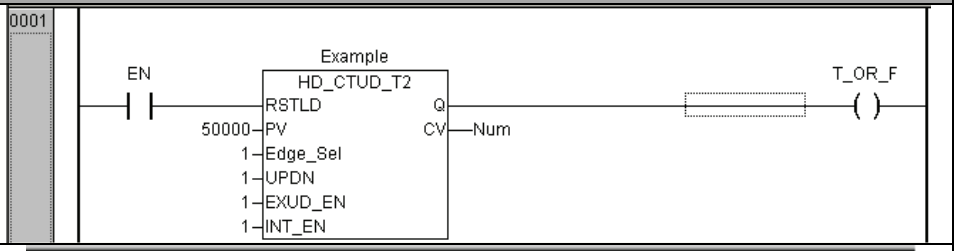
变量定义

VAR | VAR_INPUT | VAR_OUTPUT | VAR_IN_OUT | CONSTANT | RETAIN

	名称	地址	类型	初始值	注释
0001	EN		BOOL		
0002	T_OR_F		BOOL		
0003	Example		HD_CTUD_T2		
0004	Num		UINT		

编程语言：程 序

梯形图（LD）



结构化文本（ST）

```
0001 example(rstld:=en,pv:=50000,edge_sel:=1,updn:=1,exud_en:=1,int_en:=1);
0002 T_OR_F:=example.Q;
0003 Num:=example.CV;
```

程序说明：

- UPDN = 1，上电复位时 CV=0，因为此时 EXUD_EN = 1，所以 UPDN 不控制增减计数。
- EXUD_EN = 1，此时 UPDN 不控制增减计数，I0.1 方向控制使能，I0.1 高电平时增计数，I0.1 低电平时减计数。
- INT_EN = 1，因为 UPDN = 1，所以计数值 CV=PV 时触发中断，调用 HD_TC2 interrupt 事件。
- EN 置位并保持时，该指令执行，复位计数值 CV = 0，UPDN 不控制增减计数，此时 I0.0 每到达一个上升沿计数一个，Num 增加或者减少 1 个（与 I0.1 外部方向控制有关），当计数值等于 50000 时，触发 HD_TC2 interrupt 中断事件，Q 输出 1。

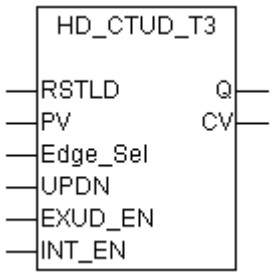
- EN 复位时，停止计数，Q 等于 0，CV 值保持先前值不变。

提示：

- 中断调用部分，请参见附录 B。

5.18.2 HD_CTUD_T3——T3 高速计数器

✧ 指令示例

梯形图语言示例


✧ 参数说明

通道说明			
I0.2	外部高速计数脉冲输入		
I0.3	外部高速方向控制输入		
输入参数	**数据类型**	**功能描述**	**参数值说明**
RSTLD	BOOL	使能	0：无效 1：上升沿使能
PV	UINT	计数设定值	0-65535
Edge_Sel	BYTE	I0.2 输入脉冲 触发边沿选择	0：禁止 1：上升沿 2：下降沿 3：双边沿
UPDN	BOOL	两个功能： 1.增/减计数选择（EXUD_EN=0 时有效） 2.初始化时复位/装载	0：减计数/装载 CV=PV 1：增计数/复位 CV=0
EXUD_EN	BOOL	外部 I0.3 方向控制使能 I0.3 高电平：增计数 I0.3 低电平：减计数	0：I0.3 方向控制禁止 1：I0.3 方向控制使能 （此时 UPDN 不控制增减计数）
INT_EN	BOOL	计数值到达中断使能 UPDN=0：CV=0 时产生中断 UPDN=1：CV=PV 时产生中断	0：T3 计数值到达中断禁止 1：T3 计数值到达中断使能
输出参数	**数据类型**	**功能描述**	**值**
Q	BOOL	计数值到达标志	0：当前计数值 CV≤PV（UPDN=1）或者 CV≥0（UPDN=0） 1：当前计数值 CV≥PV（UPDN=1）或者 CV≤0（UPDN=0）
CV	UINT	当前计数值	0-65535

✧ 关联冲突指令

使用指令 A	关联指令 B	关联硬件	可使用程度
HD_CTUD_T3	HD_DCTUD_T3	T3 内部计数器、 I0.2、I0.3 脉冲输入	不可使用指令 B
	HD_DCTUD32_T3		

✧ 指令使用举例（梯形图和结构化文本）

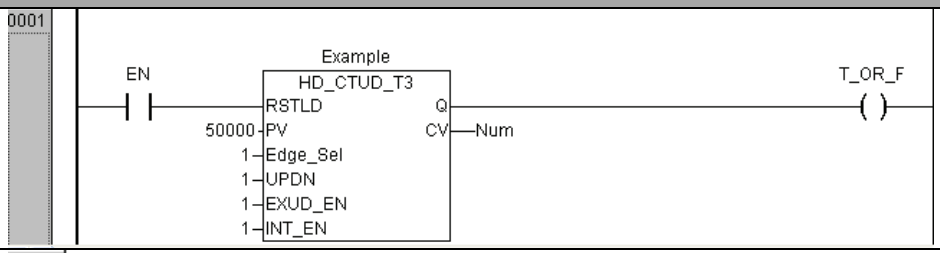
变量定义

VAR | VAR_INPUT | VAR_OUTPUT | VAR_IN_OUT | CONSTANT | RETAIN

	名称	地址	类型	初始值	注释
0001	EN		BOOL		
0002	T_OR_F		BOOL		
0003	Example		HD_CTUD_T3		
0004	Num		UINT		

编程语言 | **程 序**

梯形图（LD）



结构化文本（ST）

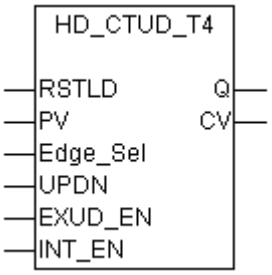
```
0001 Example(rstld:=en,pv:=50000,edge_sel:=1,updn:=1,exud_en:=1,int_en:=1);
0002 T_OR_F:=Example.Q;
0003 Num:=Example.CV;
```

程序说明：

- UPDN = 1，上电复位时 CV=0，因为此时 EXUD_EN = 1，所以 UPDN 不控制增减计数。
- EXUD_EN = 1，此时 UPDN 不控制增减计数，I0.3 方向控制使能，I0.3 高电平时增计数，I0.3 低电平时减计数。
- INT_EN = 1，因为 UPDN = 1，所以计数值 CV=PV 时触发中断，调用 HD_TC3 interrupt 事件。
- EN 置位并保持时，该指令执行，复位计数值 CV = 0，UPDN 不控制增减计数，此时 I0.2 每到达一个上升沿计数一个，Num 增加或者减少 1 个（与 I0.3 外部方向控制有关），当计数值等于 50000 时，触发 HD_TC3 interrupt 中断事件，Q 输出 1。
- EN 复位时，停止计数，Q 等于 0，CV 值保持先前值不变。

5.18.3 HD_CTUD_T4——T4 普通计数器

✧ 指令示例

梯形图语言示例


✧ 参数说明

通道说明			
I0.4	外部普通计数脉冲输入		
I0.5	外部普通方向控制输入		
输入参数	**数据类型**	**功能描述**	**参数值说明**
RSTLD	BOOL	使能	0：无效 1：上升沿使能
PV	UINT	计数设定值	0—65535
Edge_Sel	BYTE	I0.4 输入脉冲 触发边沿选择	0：禁止 1：上升沿 2：下降沿 3：双边沿
UPDN	BOOL	两个功能： 1.增/减计数选择（EXUD_EN=0 时有效） 2.初始化时复位/装载	0：减计数/装载 CV=PV 1：增计数/复位 CV=0
EXUD_EN	BOOL	外部 I0.5 方向控制使能 I0.5 高电平：增计数 I0.5 低电平：减计数	0：I0.5 方向控制禁止 1：I0.5 方向控制使能 （此时 UPDN 不控制增减计数）
INT_EN	BOOL	计数值到达中断使能 UPDN=0：CV=0 时产生中断 UPDN=1：CV=PV 时产生中断	0：T4 计数值到达中断禁止 1：T4 计数值到达中断使能
输出参数	**数据类型**	**功能描述**	**值**
Q	BOOL	计数值到达标志	0：当前计数值 CV≤PV（UPDN=1）或者 CV≥0（UPDN=0） 1：当前计数值 CV≥PV（UPDN=1）或者 CV≤0（UPDN=0）
CV	UINT	当前计数值	0-65535

✧ 关联冲突指令

使用指令 A	关联指令 B	关联硬件	可使用程度
HD_CTUD_T4	HD_DCTUD_T4	T4 内部计数器、 I0.4、I0.5 脉冲输入	不可使用指令 B
	HD_DCTUD32_T3	T4 内部计数器	

✧ 指令使用举例（梯形图和结构化文本）

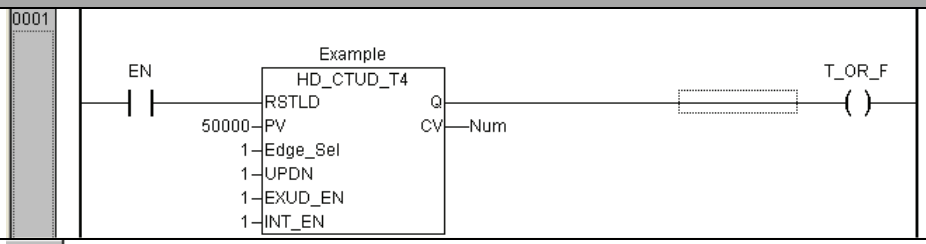
变量定义

VAR | VAR_INPUT | VAR_OUTPUT | VAR_IN_OUT | CONSTANT | RETAIN

	名称	地址	类型	初始值	注释
0001	EN		BOOL		
0002	T_OR_F		BOOL		
0003	Example		HD_CTUD_T4		
0004	Num		UINT		

编程语言 | **程 序**

梯形图（LD）



结构化文本（ST）

```
0001 Example(rstld:=en,pv:=50000,edge_sel:=1,updn:=1,exud_en:=1,int_en:=1);
0002 T_OR_F:=Example.Q;
0003 Num:=Example.CV;
```

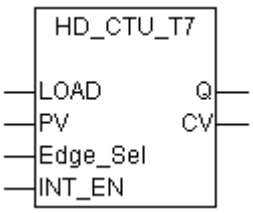
程序说明：

- UPDN = 1，上电复位时 CV=0，因为此时 EXUD_EN = 1，所以 UPDN 不控制增减计数。
- EXUD_EN = 1，此时 UPDN 不控制增减计数，I0.5 方向控制使能，I0.5 高电平时增计数，I0.5 低电平时减计数。
- INT_EN = 1，因为 UPDN = 1，所以计数值 CV=PV 时触发中断，调用 HD_TC4 interrupt 事件。
- EN 置位并保持时，该指令执行，复位计数值 CV = 0，UPDN 不控制增减计数，此时 I0.4 每到达一个上升沿计数一个，Num 增加或者减少 1 个（与 I0.5 外部方向控制有关），当计数值等于 50000 时，触发 HD_TC4 interrupt 中断事件，Q 输出 1。
- EN 复位时，停止计数，Q 等于 0，CV 值保持先前值不变。

5.18.4 HD _CTU_T7——T7 高速计数器

✧ 指令示例

梯形图语言示例



✧ **参数说明**

通道说明			
I0.6	外部高速计数脉冲输入		
输入参数	**数据类型**	**功能描述**	**参数值说明**
LOAD	BOOL	使能	0：无效 1：上升沿使能
PV	UINT	计数设定值	0-65535
Edge_Sel	BYTE	I0.6 输入脉冲 触发边沿选择	0：禁止 1：上升沿 2：下降沿 3：双边沿
INT_EN	BOOL	计数值到达中断使能	0：T7 计数设定值到达中断禁止 1：T7 计数设定值到达中断使能
输出参数	**数据类型**	**功能描述**	**值**
Q	BOOL	计数值到达标志	0：当前计数值 CV≤PV（UPDN=1）或者 CV≥0（UPDN=0） 1：当前计数值 CV≥PV（UPDN=1）或者 CV≤0（UPDN=0）
CV	UINT	当前计数值	0-65535

✧ **关联冲突指令**

使用指令 A	**关联指令 B**	**关联硬件**	**可使用程度**
HD_CTU_T7	HD_DCTUD_T2	I0.6 脉冲输入与 I0.6 外部清零脉冲输入	不可使用 B 外部清零脉冲输入功能
	Fast_ExINT	I0.6 脉冲输入与 I0.6 快速外部中断 3 脉冲输入	不可使用 B 快速外部中断 3 脉冲输入通道
	Fast_ExINT_E		
	HD_TIMER_T7	T7	不可使用指令 B

✧ **HD_DCTUD_T7 指令举例（梯形图和结构化文本）**

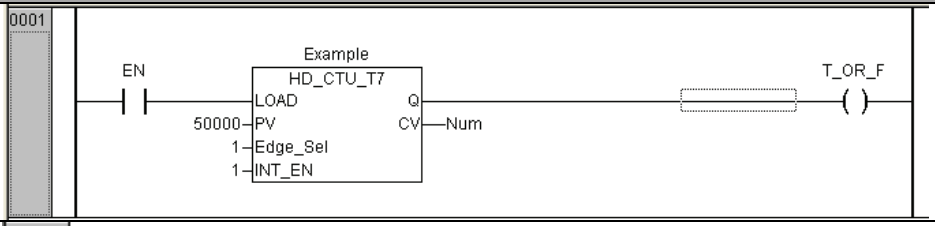
变量定义

VAR | VAR_INPUT | VAR_OUTPUT | VAR_IN_OUT | CONSTANT | RETAIN

	名称	地址	类型	初始值	注释
0001	EN		BOOL		
0002	T_OR_F		BOOL		
0003	Example		HD_CTU_T7		
0004	Num		UINT		

编程语言 | **程　序**

梯形图（LD）



结构化文本（ST）

```
0001 example(load:=en,pv:=50000,edge_sel:=1,int_en:=1);
0002 T_OR_F:=example.Q;
0003 NUM:=example.CV;
```

程序说明：

- Edge_Sel = 1，I0.6 输入脉冲为上升沿时，计数一个。
- INT_EN = 1，计数值到达标志时触发 HD_TC7 interrupt 中断事件。
- EN 置位并保持时，该指令执行，复位计数值 CV = 0，此时 I0.6 每到达一个上升沿计

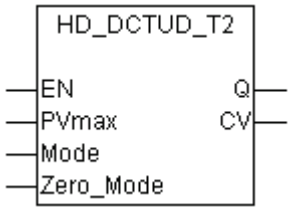
数一个，Num 增加 1 个，当计数值大于等于 50000 时，触发中断，调用 HD_TC7 interrupt 事件，Q 输出 1。

- EN 复位时，停止计数，Q 等于 0，CV 值保持先前值不变。

5.19 两相计数（Hollysys_PLC_Ex_DCT.lib）

5.19.1 HD_DCTUD_T2——T2 两相高速计数器

✧ **指令示例**

梯形图语言示例


✧ **参数说明**

通道说明			
I0.0	外部高速计数脉冲输入 A		
I0.1	外部高速计数脉冲输入 B		
I0.6	外部清零脉冲输入		
输入参数	**数据类型**	**功能描述**	**参数值说明**
EN	BOOL	使能	0：停止计数，CV 清零 1：开始计数，从 CV=0 开始，增计数 CV 值增加，减计数 CV 值减小
Mode	BYTE	模式选择	详见表 5-19-1
Zero_Mode	BYTE	清零脉冲使能	详见表 5-19-1
PVmax	UINT	设定值	范围 0-65535
输出参数	**数据类型**	**功能描述**	**值**
Q	BOOL	计数完成标志	0：增计数 CV＜PV 或 减计数 CV＞PV 时； 1：增计数 CV≥PV 或 减计数 CV≤PV 时；
CV	UINT	当前计数值	0-65535

表 5-19-1 Mode 和 Zero_Mode 参数详细说明

参数	参数说明
Mode	0 0：禁止 0 1：A 相和 B 相的上升沿计数 1 0：A 相和 B 相的下降沿计数 1 1：I0.0 或 I0.1 的上升或下降沿计数
Zero_Mode	0 0：禁止。PV 有效，当 CV≥PV，CV 复位（I0.6 无效） 0 1：I0.6 清零脉冲上升沿触发 CV 复位（PVmax 无效） 1 0：I0.6 清零脉冲下降沿触发 CV 复位（PVmax 无效） 1 1：I0.6 清零脉冲上升或下降沿触发 CV 复位（PVmax 无效）

✧ 关联冲突指令

使用指令 A	关联指令 B	关联硬件	可使用程度
HD_DCTUD_T2	HD_CTUD_T2	T2 内部计数器、I0.0、I0.1 脉冲输入	不可使用指令 B
	HD_CTU_T7	I0.6 外部清零脉冲输入与 I0.6 脉冲输入	使用 A 的外部清零脉冲输入则不可以使用 B
	Fast_ExINT	I0.6 外部清零脉冲输入与 I0.6 快速外部中断 3 脉冲输入	使用 A 的外部清零脉冲输入，则不可使用 B 快速外部中断 3 脉冲输入通道
	Fast_ExINT_E		

两个两相高速计数器占用端口说明

高速计数器	外部脉冲输入 A	外部脉冲输入 B	外部清零脉冲输入
HD_DCTUD_T2	I0.0	I0.1	I0.6
HD_DCTUD_T3	I0.2	I0.3	I0.7

✧ 指令使用举例（梯形图和结构化文本）

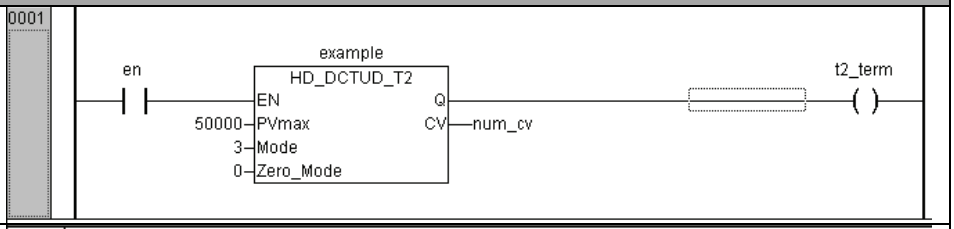
变量定义

VAR | VAR_INPUT | VAR_OUTPUT | VAR_IN_OUT | CONSTANT | RETAIN

	名称	地址	类型	初始值	注释
0001	en		BOOL		
0002	example		HD_DCTUD_T2		
0003	num_cv		UINT		
0004	t2_term		BOOL		

编程语言 | **程 序**

梯形图（LD）



结构化文本（ST）

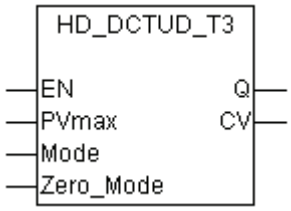
```
0001 example(EN:=en,PVmax:=5000,Mode:=3,Zero_Mode:=0);
0002 t2_term:=example.Q;
0003 num_cv:=example.CV;
0004
```

程序说明：

- Mode = 3，I0.0 或 I0.1 的上升或下降沿触发计数。
- Zero_Mode = 0，I0.6 外部清零脉冲输入不起作用，当 CV≥PVmax 时，CV 值为 0。
- EN 置位并保持时（上升沿），该指令执行，复位计数值 CV = 0，I0.0 或 I0.1 的上升或下降沿触发计数。当 CV≥PVmax，CV 值为 0，重新开始计数。
- EN 复位时，停止计数，Q 等于 0，CV 值保持先前值不变。

5.19.2 HD_DCTUD_T3——T3 两相高速计数器

✧ 指令示例

梯形图语言示例


✧ 参数说明

通道说明			
I0.2	外部高速计数脉冲输入 A		
I0.3	外部高速计数脉冲输入 B		
I0.7	外部清零脉冲输入		
输入参数	**数据类型**	**功能描述**	**参数值说明**
EN	BOOL	使能	0：停止计数，CV 清零 1：开始计数，从 CV=0 开始，增计数 CV 值增加，减计数 CV 值减小
Mode	BYTE	模式选择	详见表 5-19-2
Zero_Mode	BYTE	清零脉冲使能	详见表 5-19-2
PVma	UINT	设定值	范围 0-65535
输出参数	**数据类型**	**功能描述**	**值**
Q	BOOL	计数完成标志	0：增计数 CV＜PV 或减计数 CV＞PV 时； 1：增计数 CV≥PV 或减计数 CV≤PV 时；
CV	UINT	当前计数值	0-65535

表 5-19-2 Mode 和 Zero_Mode 参数详细说明

参数	参数说明
Mode	0 0：禁止　　0 1：A 相和 B 相的上升沿计数 1 0：A 相和 B 相的下降沿计数　　1 1：I0.0 或 I0.1 的上升或下降沿计数
Zero_Mode	0 0：禁止。PV 有效，当 CV≥PV，CV 复位（I0.7 无效） 0 1：I0.7 清零脉冲上升沿触发 CV 复位（PVmax 无效） 1 0：I0.7 清零脉冲下降沿触发 CV 复位（PVmax 无效） 1 1：I0.7 清零脉冲上升或下降沿触发 CV 复位（PVmax 无效）

✧ 关联冲突指令

使用指令 A	关联指令 B	关联硬件	可使用程度
HD_DCTUD_T2	HD_CTUD_T2	T2 内部计数器、I0.0、I0.1 脉冲输入	不可使用指令 B
	HD_CTU_T7	I0.6 外部清零脉冲输入与 I0.6 脉冲输入	使用 A 的外部清零脉冲输入，则不可以使用 B
	Fast_ExINT	I0.6 外部清零脉冲输入与 I0.6 快速外部中断 3 脉冲输入	使用 A 的外部清零脉冲输入，则不可使用 B 快速外部中断 3 脉冲输入通道
	Fast_ExINT_E		

两相高速计数器占用端口说明

高速计数器	外部脉冲输入 A	外部脉冲输入 B	外部清零脉冲输入
HD_DCTUD_T2	I0.0	I0.1	I0.6
HD_DCTUD_T3	I0.2	I0.3	I0.7

✧ 指令使用举例（梯形图和结构化文本）

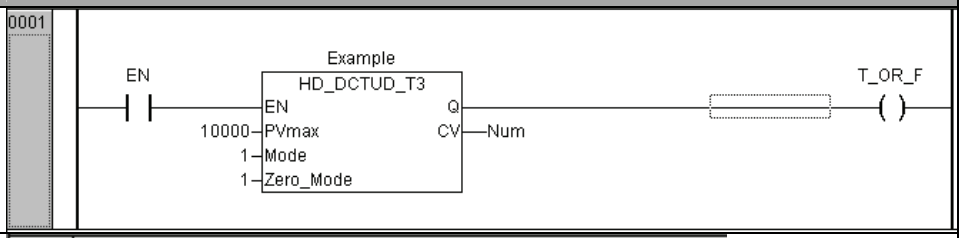
变量定义

VAR | VAR_INPUT | VAR_OUTPUT | VAR_IN_OUT | CONSTANT | RETAIN

	名称	地址	类型	初始值	注释
0001	EN		BOOL		
0002	T_OR_F		BOOL		
0003	Example		HD_DCTUD_T3		
0004	Num		UINT		

编程语言 | **程 序**

梯形图（LD）



结构化文本（ST）

```
0001 example(EN:=en,pvmax:=10000,mode:=1,zero_mode:=1);
0002 T_OR_F:=example.Q;
0003 NUM:=example.CV;
```

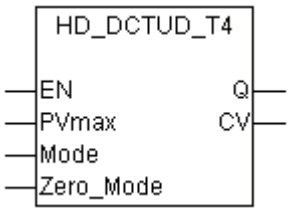
程序说明:

- Mode = 1，I0.3 的上升或下降沿触发计数。
- Zero_Mode = 1，PVmax 不起作用，当 I0.7 上升沿到达触发 CV 复位。
- EN 置位并保持时（上升沿），该指令执行，复位计数值 CV = 0，I0.3 的上升或下降沿触发计数。当 I0.7 上升沿到达触发 CV 复位，重新开始计数。
- EN 复位时，停止计数，Q 等于 0，CV 值保持先前值不变。

5.19.3 HD_DCTUD_T4——T4 两相普通计数器

✧ 指令示例

梯形图语言示例



✧ 参数说明

通道说明			
I0.4	外部普通计数脉冲输入 A		
I0.5	外部普通计数脉冲输入 B		
I1.0	外部清零脉冲输入		
输入参数	**数据类型**	**功能描述**	**参数值说明**
EN	BOOL	使能	0：停止计数，CV 清零 1：开始计数，从 CV=0 开始，增计数 CV 值增加，减计数 CV 值减小
Mode	BYTE	模式选择	详见表 5-19-3
Zero_Mode	BYTE	清零脉冲使能	详见表 5-19-3
PVma	UINT	设定值	范围 0-65535
输出参数	**数据类型**	**功能描述**	**值**
Q	BOOL	计数完成标志	0：增计数 CV＜PV 或减计数 CV＞PV 时； 1：增计数 CV≥PV 或减计数 CV≤PV 时；
CV	UINT	当前计数值	0-65535

表 5-19-3 Mode 和 Zero_Mode 参数详细说明

参数	参数说明
Mode	0 0：禁止 0 1：A 相和 B 相的上升沿计数 1 0：A 相和 B 相的下降沿计数 1 1：I0.4 或 I0.5 的上升或下降沿计数
Zero_Mode	0 0：禁止。PV 有效，当 CV≥PV，CV 复位（I1.0 无效） 0 1：I1.0 清零脉冲上升沿触发 CV 复位（PVmax 无效） 1 0：I1.0 清零脉冲下降沿触发 CV 复位（PVmax 无效） 1 1：I1.0 清零脉冲上升或下降沿触发 CV 复位（PVmax 无效）

✧ 关联冲突指令

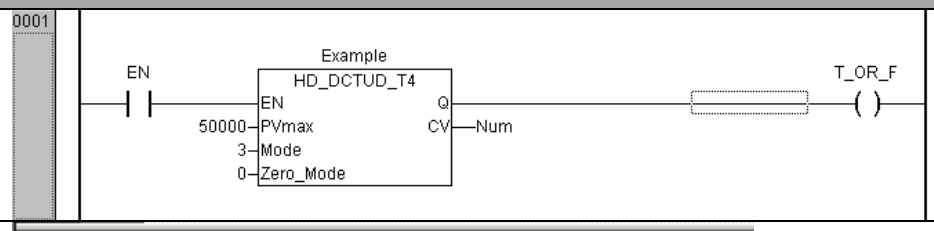
使用指令 A	关联指令 B	关联硬件	可使用程度
HD_DCTUD_T4	HD_CTUD_T4	T4 内部计数器、I0.4、I0.5 脉冲输入	不可使用指令 B
	HD_DCTUD32_T3	T4 内部计数器	
	Fast_ExINT	I1.0 外部清零脉冲输入与 I1.0 快速外部中断 1 脉冲输入	使用 A 的外部清零脉冲输入，则不可使用 B 快速外部中断 1 脉冲输入通道
	Fast_ExINT_E		

✧ 指令使用举例（梯形图和结构化文本）

变量定义

VAR | VAR_INPUT | VAR_OUTPUT | VAR_IN_OUT | CONSTANT | RETAIN

	名称	地址	类型	初始值	注释
0001	EN		BOOL		
0002	T_OR_F		BOOL		
0003	Example		HD_DCTUD_T4		
0004	Num		UINT		

编程语言	程　序
梯形图（LD）	
结构化文本（ST）	0001 example(EN:=en,pvmax:=50000,mode:=3,zero_mode:=0); 0002 T_OR_F:=example.Q; 0003 NUM:=example.CV;

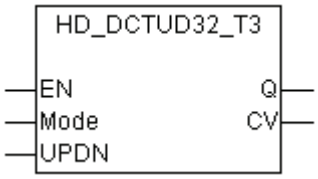
程序说明：

- Mode = 3，I0.4 或 I0.5 的上升或下降沿触发计数。
- Zero_Mode = 0，I1.0 外部清零脉冲输入无效，当 CV≥PVmax 时，CV 值为 0。
- EN 置位并保持时，该指令执行，复位计数值 CV = 0，I0.4 或 I0.5 的上升或下降沿触发计数。当 CV≥PVmax，CV 值为 0，重新开始计数。
- EN 复位时，停止计数，Q 等于 0，CV 值保持先前值不变。

5.20 两相 32 位计数（Hollysys_PLC_Ex_DCT32.lib）

HD_DCTUD32_T3—32 位高速计数器

✧ **指令示例**

梯形图语言示例


✧ **参数说明**

通道说明			
I0.2	外部普通计数脉冲输入 A		
I0.3	外部高速计数脉冲输入 B		
输入参数	**数据类型**	**功能描述**	**参数值说明**
EN	BOOL	使能	0：无效 1：上升沿使能
Mode	BYTE	计数模式选择	0：禁止 1：I0.3 的上升或下降沿 2：I0.2 的上升或下降沿 3：I0.2 或 I0.3 的上升或下降沿
UPDN	BOOL	进位方向（低 16 位向高 16 位）（计数值-32767-32768 之间不需要设置进位方向）	0:向下进位 1:向上进位

输出参数	数据类型	功能描述	值
Q	BOOL	计数启动标志	0：计数禁止 1：计数已启动
CV	DWORD	当前计数器值	-2147483648-2147483647

✧ 关联冲突指令

使用指令 A	关联指令 B	关联硬件	可使用程度
HD_DCTUD32_T3	HD_CTUD_T3	T3 内部计数器、I0.2、I0.3 脉冲输入	不可使用指令 B
	HD_CTUD_T4	T4 内部计数器	
	HD_DCTUD_T3	T3 内部计数器、I0.2、I0.3 脉冲输入	
	HD_DCTUD_T4	T4 内部计数器	

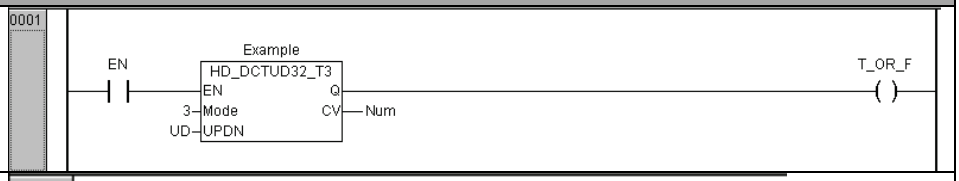
✧ 指令使用举例（梯形图和结构化文本）

变量定义

	名称	地址	类型	初始值
0001	EN		BOOL	
0002	T_OR_F		BOOL	
0003	Example		HD_DCTUD32_T3	
0004	UD		BOOL	
0005	Num		DWORD	

编程语言 | **程 序**

梯形图（LD）



结构化文本（ST）

```
0001 Example(EN:=EN,mode:=3,UPDN:=UD);
0002 Num:=Example.CV;
0003 T_OR_F:=Example.Q;
```

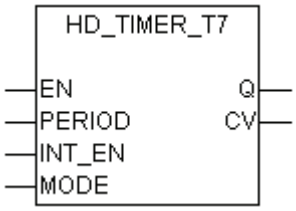
程序说明：

- Mode = 3，I0.2 或 I0.3 的上升或下降沿触发计数一个。
- UD 控制低 16 位向高 16 位进位方向。
- EN 置位并保持时，该指令执行，复位计数值 CV = 0，I0.2 或 I0.3 的上升或下降沿触发计数一个，当 CV 到达 32767 时，如果再增计数需要保证此时 UD 等于 1，否则计数出错。
- EN 复位时，停止计数，Q 等于 0，CV 值保持先前值不变。

5.21 中断定时器（Hollysys_PLC_Ex_TIMER.lib）

5.21.1 HD_TIMER_T7——中断定时器

✧ 指令示例

梯形图语言示例


✧ 参数说明

输入参数	数据类型	功能描述	参数值说明
EN	BOOL	使能	0：无效 1：上升沿使能
PERIOD	DWORD	设定时间（us）	100-2680000
INT_EN	BOOL	T7 中断使能	0：设定时间到不产生中断 1：设定时间到产生中断
MODE	BOOL	触发模式选择	0：上升沿触发 1：电平触发
输出参数	**数据类型**	**功能描述**	**值**
Q	BOOL	是否设置完毕	0：EN 复位 1：设置完毕，保持
CV	DWORD	已过时间（ns）	0-2680000000

✧ 关联冲突指令

使用指令 A	关联指令 B	关联硬件	可使用程度
HD_TIMER_T7	HD_CTU_T7	T7	不可使用指令 B

✧ **HD_TIMER_T7 指令举例（梯形图和结构化文本）**

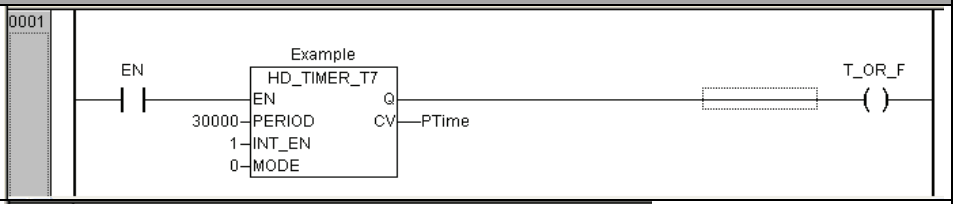
变量定义

VAR | VAR_INPUT | VAR_OUTPUT | VAR_IN_OUT | CONSTANT | RETAIN

	名称	地址	类型	初始值	注释
0001	EN		BOOL		
0002	Example		HD_TIMER_T7		
0003	PTime		DWORD		
0004	T_OR_F		BOOL		

编程语言 | **程　序**

梯形图（LD）



结构化文本（ST）

```
0001 example(en:=en,period:=30000,int_en:=1,mode:=0);
0002 T_OR_F:=example.q;
0003 PTIME:=example.CV;
```

程序说明：

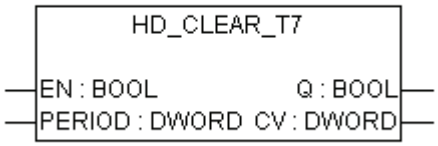
- EN 置位并保持时（上升沿），Q 等于 1，因为 INT_EN 等于 1，所以此后每隔 30000us 就触发一次 HD_TC7 interrupt 中断事件，通过 PowerPro 中的系统事件来调用 HD_TC7

interrupt 中断事件执行程序，CV 显示当前已过时间值，单位为 ns。

- EN 复位时，停止产生 HD_TC7 interrupt 中断事件，Q 等于 0。

5.21.2 HD_CLEAR_T7——重载定时器

✧ 指令示例

梯形图语言示例


✧ 参数说明

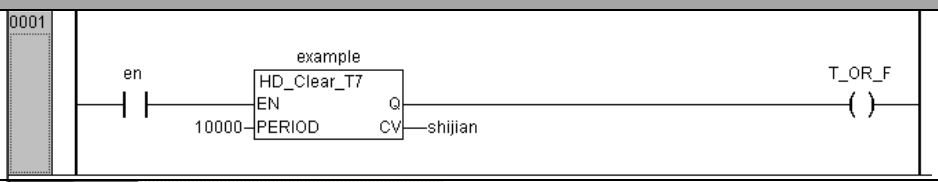
输入参数	数据类型	功能描述	参数值说明
EN	BOOL	使能	0：无效 1：使能
PERIOD	DWORD	最大定时时间（us）	100-2680000
输出参数	**数据类型**	**功能描述**	**值**
Q	BOOL	是否已重载	0：未重载 1：已重载
CV	DWORD	重载前已过时间（ns）	0-2680000000

✧ HD_CLEAR_T7 指令举例（梯形图和结构化文本）

变量定义

VAR | VAR_INPUT | VAR_OUTPUT | VAR_IN_OUT | CONSTANT | RETAIN

	名称	地址	类型	初始值	注释
0001	en		BOOL		
0002	example		HD_Clear_T7		
0003	T_OR_F		BOOL		
0004	shijian		DWORD		

编程语言	程　序
梯形图（LD）	
结构化文本（ST）	0001 example(en:=en,period:=10000); 0002 T_OR_F:=example.Q; 0003 shijian:=example.CV;

程序说明：

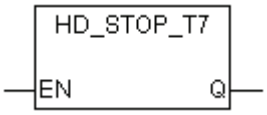
- 此指令配合 HD_TIMER_T7 与快速外部中断使用，可以用来测量脉冲的宽度，在主程序中启动快速外部中断与 HD_TIMER_T7 定时器，比如定义 I0.6 捕捉快速外部中

断的上升沿与下降沿，在中断程序中调用 HD_CLEAR_T7。当 I0.6 上升沿到来时产生中断，调用该指令将 T7 的值初始化，当 I0.6 下降沿到来时再次产生中断，此时将脉宽值存入 CV 中（单位为 ns），同时重载 T7。

- 需要注意的是 HD_CLEAR_T7 与 HD_TIMER_T7 的 PERIOD 的值应该保持一致，否则 CV 处的值会出现错误。

5.21.3 HD_STOP_T7——停止定时器

✧ **指令示例**

梯形图语言示例


✧ **参数说明**

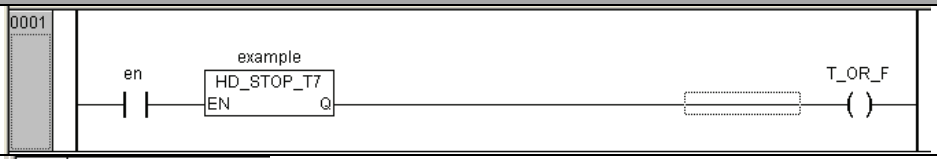
输入参数	数据类型	功能描述	参数值说明
EN	BOOL	使能	0：无效 1：使能
输出参数	**数据类型**	**功能描述**	**值**
Q	BOOL	是否停止	0：未停止或者使能断开 1：已停止

✧ **HD_STOP_T7 指令举例（梯形图和结构化文本）**

变量定义

VAR | VAR_INPUT | VAR_OUTPUT | VAR_IN_OUT | CONSTANT | RETAIN

	名称	地址	类型	初始值	注释
0001	en		BOOL		
0002	example		HD_STOP_T7		
0003	T_OR_F		BOOL		

编程语言	程　序
梯形图（LD）	
结构化文本（ST）	0001 example(en:=en); 0002 T_OR_F:=example.Q;

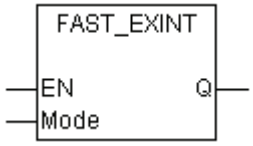
程序说明：

- 此指令的作用是使 T7 定时器停止运行，配合 HD_TIMER_T7 使用。
- 使用 HD_TIMER_T7 启动 T7 定时器，当 HD_STOP_T7 运行时 T7 停止。

5.22 外部中断（Hollysys_PLC_Ex_ExINT.lib）

5.22.1 Fast_ExINT——快速外部中断

✧ 指令示例

梯形图语言示例


✧ 参数说明

通道说明			
I1.0	快速外部中断 1 脉冲输入（不适用于 LM3104、LM3105）		
I0.7	快速外部中断 2 脉冲输入		
I0.6	快速外部中断 3 脉冲输入		
输入参数	**数据类型**	**功能描述**	**参数值说明**
EN	BOOL	使能	0：无效 1：上升沿使能
Mode	BYTE	中断模式设置	详见表 5-22-1
输出参数	**数据类型**	**功能描述**	**值**
Q	BOOL	当正确地设定参数时为 1	0：设置不正确 1：设置正确

表 5-22-1 Mode 参数详细说明

Bits							
7	6	5	4	3	2	1	0
中断 3		中断 2		中断 1		无	
中断 n 设置（n=1、2、3）							
禁止：　　0 0 上升沿触发：　　0 1 下降沿触发：　　1 0 上升或下降沿触发：1 1							

✧ 关联冲突指令

使用指令 A	关联指令 B	关联硬件	可使用程度
Fast_ExINT	HD_CTU_T7	I0.6 快速外部中断 3 脉冲输入与 I0.6 脉冲输入	使用 A 快速外部中断 3 脉冲输入，则不可使用 B
	HD_DCTUD_T2	I0.6 快速外部中断 3 脉冲输入与 I0.6 外部清零脉冲输入	使用 A 快速外部中断 3 脉冲输入，则不可使用 B 外部清零脉冲输入通道
	HD_DCTUD_T3	I0.7 快速外部中断 2 脉冲输入与 I0.7 外部清零脉冲输入	使用 A 快速外部中断 2 脉冲输入，则不可使用 B 外部清零脉冲输入通道
	HD_DCTUD_T4	I1.0 快速外部中断 1 脉冲输入通道、I1.0 外部清零脉冲输入	使用 A 快速外部中断 1 脉冲输入，则不可使用 B 外部清零脉冲输入通道

✧ **FAST_ExINT 指令举例（梯形图和结构化文本）**

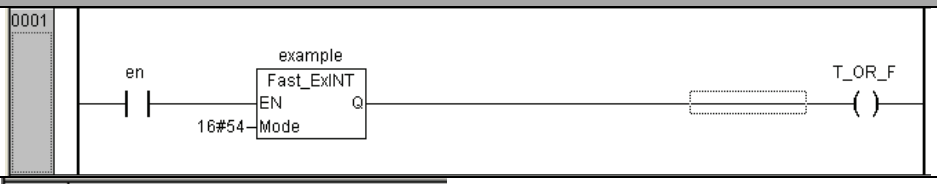
变量定义

VAR | VAR_INPUT | VAR_OUTPUT | VAR_IN_OUT | CONSTANT | RETAIN

	名称	地址	类型	初始值	注释
0001	en		BOOL		
0002	example		Fast_ExINT		
0003	T_OR_F		BOOL		

编程语言 | **程 序**

梯形图（LD）



结构化文本（ST）

```
0001 example(en:=en,mode:=16#54);
0002 T_OR_F:=example.q;
```

程序说明：

- EN 置位并保持时，Q 等于 1，I0.6、I0.7、I1.0 每到达一个上升沿，见下表，设定 Mode 值十六进制为 54（二进制为 01 01 01 00），三个快速外部中断全部设定为上升沿触发。

Mode = 16#54							
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
中断 3		中断 2		中断 1		中断 0	
0	1	0	1	0	1	无	

- 触发相应的快速外部中断，通过 PowerPro 的系统事件来调用相应的中断执行程序。
- EN 复位时，I0.6、I0.7、I1.0 停止接收中断脉冲，Q 等于 0。

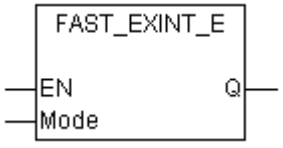
提示：

- FAST_ExINT 指令适用于程序存储容量为 28K 的模块。

5.22.2 Fast_ExINT_E——快速外部中断

✧ **指令示例**

梯形图语言示例



✧ **参数说明**

通道说明			
I1.1	快速外部中断 0 脉冲输入（不适用于 LM3104、LM3105）		
I1.0	快速外部中断 1 脉冲输入（不适用于 LM3104、LM3105）		
I0.7	快速外部中断 2 脉冲输入		
I0.6	快速外部中断 3 脉冲输入		
输入参数	**数据类型**	**功能描述**	**参数值说明**
EN	BOOL	使能	0：无效　　1：上升沿使能
Mode	BYTE	中断模式设置	详见表 5-22-2
输出参数	**数据类型**	**功能描述**	**值**
Q	BOOL	当正确地设定参数时为 1	0：设置不正确　　1：设置正确

表 5-22-2 Mode 参数详细说明

Bits							
7	6	5	4	3	2	1	0
中断 3		中断 2		中断 1		中断 0	
中断 n 设置（n=1、2、3）							
禁止：　　0 0 上升沿触发：　　0 1 下降沿触发：　　1 0 上升或下降沿触发：1 1							

✧ 关联冲突指令

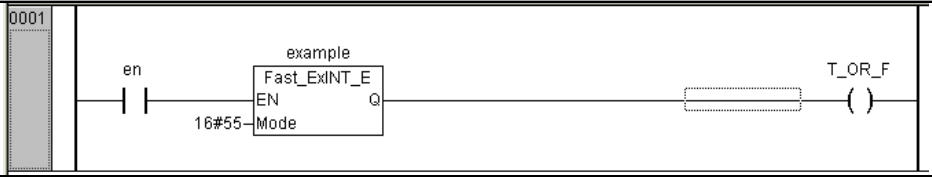
使用指令 A	关联指令 B	关联硬件	可使用程度
Fast_ExINT_E	HD_CTU_T7	I0.6 快速外部中断 3 脉冲输入与 I0.6 脉冲输入	使用 A 快速外部中断 3 脉冲输入，则不可使用 B
	HD_DCTUD_T2	I0.6 快速外部中断 3 脉冲输入与 I0.6 外部清零脉冲输入	使用 A 快速外部中断 3 脉冲输入，则不可使用 B 外部清零脉冲输入通道
	HD_DCTUD_T3	I0.7 快速外部中断 2 脉冲输入与 I0.7 外部清零脉冲输入	使用 A 快速外部中断 2 脉冲输入，则不可使用 B 外部清零脉冲输入通道
	HD_DCTUD_T4	I1.0 快速外部中断 1 脉冲输入通道、I1.0 外部清零脉冲输入	使用 A 快速外部中断 1 脉冲输入，则不可使用 B 外部清零脉冲输入通道

✧ 指令使用举例（梯形图和结构化文本）

变量定义

VAR | VAR_INPUT | VAR_OUTPUT | VAR_IN_OUT | CONSTANT | RETAIN

	名称	地址	类型	初始值	注释
0001	en		BOOL		
0002	example		Fast_ExINT_E		
0003	T_OR_F		BOOL		

编程语言	程　序
梯形图（LD）	
结构化文本（ST）	example(en:=en,mode:=16#55); T_OR_F:=example.Q;

程序说明:

- 设定 Mode 值十六进制为 55（二进制为 01 01 01 01）（见下表），四个快速外部中断全部设定为上升沿触发。

Mode = 16#55							
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
中断 3		中断 2		中断 1		中断 0	
0	1	0	1	0	1	0	1

- EN 置位并保持时（上升沿），Q 等于 1，I0.6、I0.7、I1.0、I1.1 每到达一个上升沿，就触发相应的快速外部中断，通过 PowerPro 的系统事件来调用相应的中断执行程序。
- EN 复位时，I0.6、I0.7、I1.0、I1.1 停止接收中断脉冲，Q 等于 0。

提示：

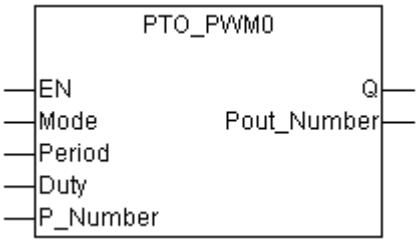
- Fast_ExINT_E 指令适用于程序存储容量为 120K 的模块。

5.23 脉冲输出指令（Hollysys_PLC_Ex_PT.lib）

5.23.1 PTO_PWM0——PTO/PWM 脉冲输出

✧ **指令示例**

梯形图语言示例



✧ **参数说明**

通道说明			
Q1.1	高速脉冲输出（Mode=0，1，2，3）		
Q1.0	高速脉冲互补输出（Mode=2，3）		
输入参数	**数据类型**	**功能描述**	**参数值说明**
EN	BOOL	使能	0：无效 1：上升沿使能
Mode	BYTE	脉冲输出模式选择（Mode=1、3 时 P_Number 无效，Pout_Number 输出 0）	0：Q1.1 输出 PTO 1：Q1.1 输出 PWM 2：Q1.1 与 Q1.0 互补输出 PTO 3：Q1.1 与 Q1.0 互补输出 PWM
Period	DWORD	周期（μs）设置	详见表 5-23-1
Duty	BYTE	占空比	0-100（PTO 模式恒为 50）

P_Number	DWORD	设定脉冲个数	0-4294967295
输出参数	**数据类型**	**功能描述**	**值**
Q	BOOL	脉冲发送标志	0：停止发送 1：正在发送
Pout_Number	DWORD	已发出的脉冲个数	0-4294967295

表 5-23-1 Period 参数详细说明

模块类型	PTO	PWM
LM3106	50-335000	50-335000
LM3106A	20-335000	10-335000
LM3108	50-335000	50-335000

✧ **关联冲突指令**

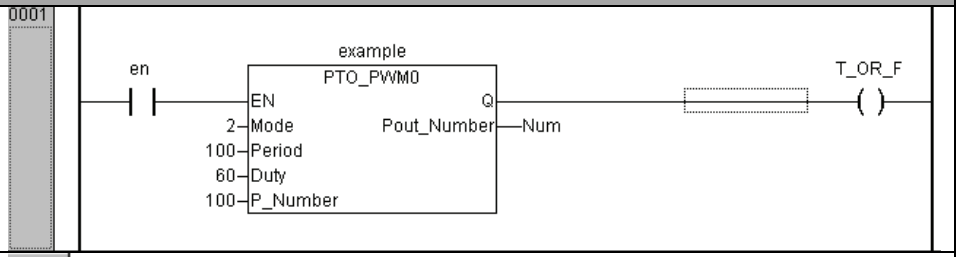
使用指令 A	关联指令 B	关联硬件	可使用程度
PTO_PWM0	PTO_PWM0_Run	T12	不可使用指令 B
	PTOCtrl_0		

✧ **PTO_PWM0 指令举例（梯形图和结构化文本）**

变量定义

0001	en		BOOL		
0002	example		PTO_PWM0		
0003	Num		DWORD		
0004	T_OR_F		BOOL		

编程语言：梯形图（LD）



编程语言：结构化文本（ST）

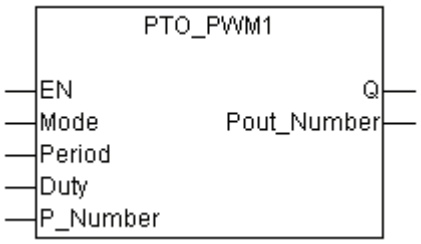
```
0001 example(en:=en,mode:=2,period:=100,duty:=60,p_number:=100);
0002 T_OR_F:=example.Q;
0003 Num:=example.Pout_Number;
```

程序说明：

- EN 置位并保持时，Q1.1 与 Q1.0 以相反的电平开始发送脉冲（Mode=2），脉冲的频率为 1s / 100μs = 10K（Period=100），Num 显示当前已经发送的脉冲数量，Q 等于 1 并保持，共发送 10000 个脉冲，发送完毕后停止，Q 等于 0。
- 发送过程中，若 EN 复位，Q1.1 与 Q1.0 停止发送脉冲，Q 等于 0，Num 保持当前值不变。
- 因为选择 Mode=2，输出 PTO，占空比为 50（此时不论指令上 Duty 输入多少，占空比恒为 50）。

5.23.2 PTO_PWM1——PTO/PWM 脉冲输出

✧ 指令示例

梯形图语言示例


✧ 参数说明

通道说明			
Q0.3	高速脉冲输出（Mode=0，1）		
输入参数	**数据类型**	**功能描述**	**参数值说明**
EN	BOOL	使能	0：无效 1：上升沿使能
Mode	BYTE	脉冲输出模式选择（Mode=1 时，P_Number 无效，Pout_Number 输出 0）	0：Q0.3 输出 PTO 1：Q0.3 输出 PWM
Period	DWORD	周期（μs）设置	详见表 5-23-2
Duty	BYTE	占空比	0-100（PTO 模式恒为 50）
P_Number	DWORD	设定脉冲个数	0-4294967295
输出参数	**数据类型**	**功能描述**	**值**
Q	BOOL	脉冲发送标志	0：停止发送　　1：正在发送
Pout_Number	DWORD	已发出的脉冲个数	0-4294967295

表 5-23-2 Period 参数详细说明

模块类型	PTO	PWM
LM3104	50-2630000	50-2630000
LM3106	50-2630000	50-2630000
LM3106A	20-2630000	10-2630000
LM3108	50-2630000	50-2630000

✧ 关联冲突指令

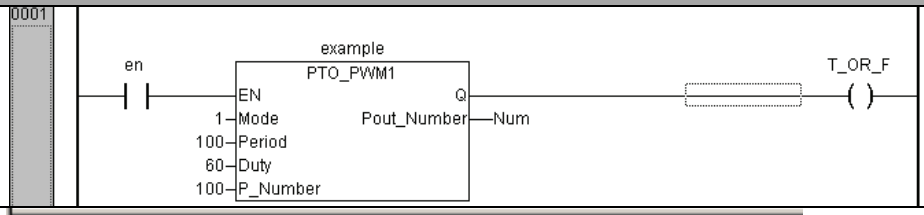
使用指令 A	关联指令 B	关联硬件	可使用程度
PTO_PWM1	PTO_PWM1_Run	T8、T13	不可使用指令 B
	PTOCtrl_1	T13	

✧ **PTO_PWM1 指令举例（梯形图和结构化文本）**

变量定义

VAR | VAR_INPUT | VAR_OUTPUT | VAR_IN_OUT | CONSTANT | RETAIN

	名称	地址	类型	初始值	注释
0001	en		BOOL		
0002	example		PTO_PWM1		
0003	T_OR_F		BOOL		
0004	Num		DWORD		

编程语言	程　序
梯形图（LD）	
结构化文本（ST）	0001 example(en:=en,mode:=1,period:=100,duty:=60,p_number:=100); 0002 T_OR_F:=example.Q; 0003 Num:=example.Pout_Number;

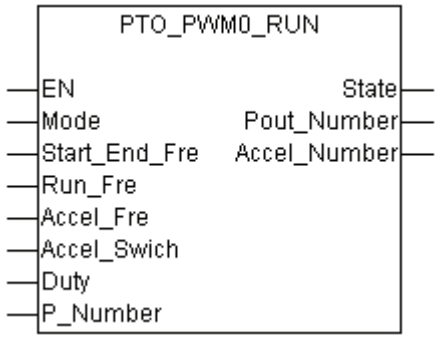
程序说明：

- EN 置位并保持时，Q0.3 开始发送脉冲，脉冲的频率为 1s / 100μs = 10K，占空比为 60，Num 显示当前已经发送的脉冲数量，Q 等于 1 并保持。
- EN 复位时，Q0.3 停止发送脉冲，Q 等于 0，Num 保持当前值不变。
- 因为选择 Mode = 1，输出 PWM 脉冲，占空比为 60。

5.24 脉冲加减速输出指令（Hollysys_PLC_Ex_PTRun.lib）

5.24.1 PTO_PWM0_RUN——PTO_PWM 脉冲输出（加减速）

✧ 指令示例

梯形图语言示例


✧ 功能描述

- ✓ 若 Start_End_Fre>Run_Fre，为减速运行功能，在任何模式下，则按减速—匀速运行。
- ✓ 若 Start_End_Fre<Run_Fre，为加速运行功能，在 PTO 模式下，按加速—匀速—减速运行（加速和减速过程是对称的），在 PWM 模式，按加速—匀速运行。
- ✓ 如果 Start_End_Fre=Run_Fre，则为匀速运行功能，此时 Accel_Fre、Accel_Swich 无效。

✧ 参数说明

通道说明	
Q1.1	高速脉冲输出（Mode=0，1，2，3）
Q1.0	高速脉冲互补输出（Mode=2，3）

输入参数	数据类型	功能描述	参数值说明
EN	BOOL	使能	0：无效　　1：上升沿使能
Mode	BYTE	脉冲输出模式选择（Mode=1、3 时 P_Number 无效，Pout_Number 输出 0）	0：Q1.1 输出 PTO 1：Q1.1 输出 PWM 2：Q1.1 与 Q1.0 互补输出 PTO 3：Q1.1 与 Q1.0 互补输出 PWM
Start_End_Fre	DWORD	起始频率&停止频率（Hz）	详见表 5-24-1
Run_Fre	DWORD	运行频率（Hz）	详见表 5-24-1
Accel_Fre	DWORD	频率增量（Hz，正值）	
Accel_Swich	BOOL	频率改变开关	0:不改变　　1：改变
Duty	BYTE	占空比	0-100（PTO 模式恒为 50）
P_Number	DWORD	要发送的脉冲数	0-4294967295
输出参数	**数据类型**	**功能描述**	**值**
State	BOOL	脉冲发送标志	0：停止发送　　1：正在发送
Pout_Number	DWORD	已发送的脉冲数（模式 0、2 下有效）	0-4294967295
Accel_Number	DWORD	变速阶段发送的脉冲数	0-4294967295

表 5-24-1 Start_End_Fre 和 Run_Fre 参数详细说明

模块类型	PTO	PWM
LM3106	3-20000	3-20000
LM3106A	3-50000	3-100000
LM3108	3-20000	3-20000

✧ 关联冲突指令

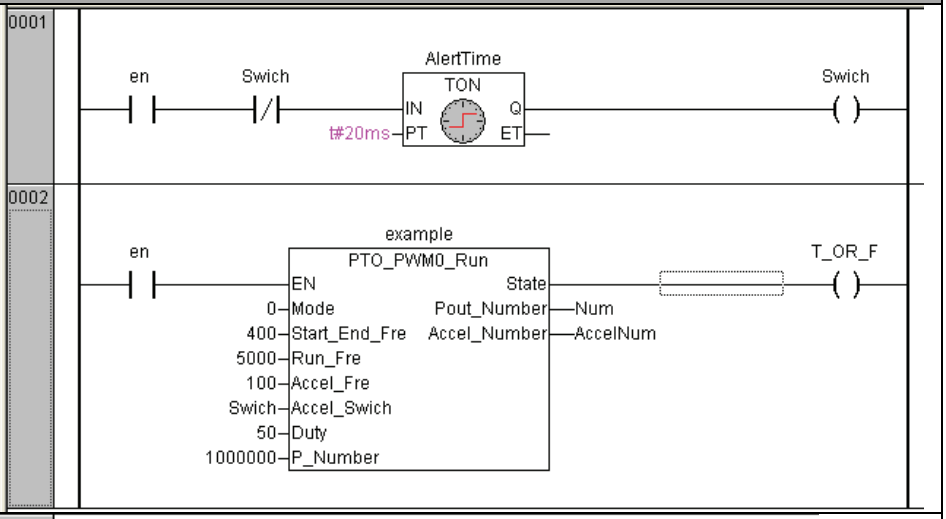
使用指令 A	关联指令 B	关联硬件	可使用程度
PTO_PWM0_Run	PTO_PWM0	T12	不可使用指令 B
	PTOCtrl_0		

✧ 指令使用举例（梯形图和结构化文本）

变量定义

VAR | VAR_INPUT | VAR_OUTPUT | VAR_IN_OUT | CONSTANT | RETAIN

	名称	地址	类型	初始值	注释
0001	en		BOOL		
0002	T_OR_F		BOOL		
0003	AlertTime		TON		
0004	Num		DWORD		
0005	Swich		BOOL		
0006	example		PTO_PWM0_Run		
0007	AccelNum		DWORD		

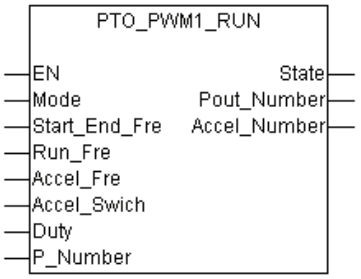
编程语言	程　序
梯形图（LD）	
结构化文本（ST）	0001 AlertTime(in:=en AND NOT swich,pt:=t#20ms); 0002 SWICH:=AlertTime.Q; 0003 0004 example(en:=en,mode:=0,start_End_Fre:=400,Run_Fre:=5000,Accel_Fre:=100, 0005 Accel_Swich:=Swich,duty:=50,P_Number:=1000000); 0006 T_OR_F:=example.State; 0007 NUM:=example.Pout_Number; 0008 AccelNum:=example.Accel_Number;

程序说明：

- EN 置位并保持时，Q1.1 开始发送脉冲，脉冲的频率为 400Hz，Num 显示当前已发送脉冲数，AccelNum 显示 0，此后每过 20ms 脉冲频率增加 100Hz，直到脉冲频率增加到 5000Hz，AccelNum 显示加速阶段发送的脉冲数（受扫描周期影响），然后开始以 5000Hz 匀速运行，因为选择 Mode=0，所以当剩余待发送脉冲数等于加速阶段发送的脉冲数时开始减速，直到发送完 1000000 个脉冲后，停止发送，在下一个扫描周期到达时， AccelNum 清 0，Num 一直保持直到 EN 再次到达上升沿。
- EN 复位时，Q1.1 停止发送脉冲，Q 等于 0。
- 因为选择 Mode=0，输出 PTO，占空比为 50（此时不论指令上 Duty 等于多少，占空比恒为 50）。

5.24.2 PTO_PWM1_RUN——PTO_PWM 脉冲输出（加减速）

✧ **指令示例**

梯形图语言示例


✧ 功能描述

- ✓ 若 Start_End_Fre>Run_Fre，为减速运行功能，在任何模式下，则按减速—匀速运行。
- ✓ 若 Start_End_Fre<Run_Fre，为加速运行功能，在 PTO 模式下，按加速—匀速—减速运行（加速和减速过程是对称的），在 PWM 模式，按加速—匀速运行。
- ✓ 如果 Start_End_Fre=Run_Fre，则为匀速运行功能，此时 Accel_Fre、Accel_Swich 无效。

✧ 参数说明

通道说明			
Q0.3	高速脉冲输出（Mode=0，1）		
输入参数	**数据类型**	**功能描述**	**参数值说明**
EN	BOOL	使能	0：无效 1：上升沿使能
Mode	BYTE	脉冲输出模式选择（Mode=1 时，P_Number 无效，Pout_Number 输出 0）	0：Q0.3 输出 PTO 1：Q0.3 输出 PWM
Start_End_Fre	DWORD	起始频率&停止频率（Hz）	详见表 5-24-2
Run_Fre	DWORD	运行频率（Hz）	详见表 5-24-2
Accel_Fre	DWORD	频率增量（Hz，正值）	
Accel_Swich	BOOL	频率改变开关	0:不改变 1：改变
Duty	BYTE	占空比	0-100（PTO 模式恒为 50）
P_Number	DWORD	要发送的脉冲数	0-4294967295
输出参数	**数据类型**	**功能描述**	**值**
State	BOOL	脉冲发送标志	0：停止发送 1：正在发送
Pout_Number	DWORD	已发送的脉冲数（模式 0、2 下有效）	0-4294967295
Accel_Number	DWORD	变速阶段 发送的脉冲数	0-4294967295

Start_End_Fre 和 Run_Fre 参数详细说明

模块类型	PTO	PWM
LM3104	1-20000	1-20000
LM3106	1-20000	1-20000
LM3106A	1-50000	1-100000
LM3108	1-20000	1-20000

✧ 关联冲突指令

使用指令 A	关联指令 B	关联硬件	可使用程度
PTO_PWM1_Run	PTO_PWM1	T8、T13	不可使用指令 B
	PTOCtrl_1	T13	

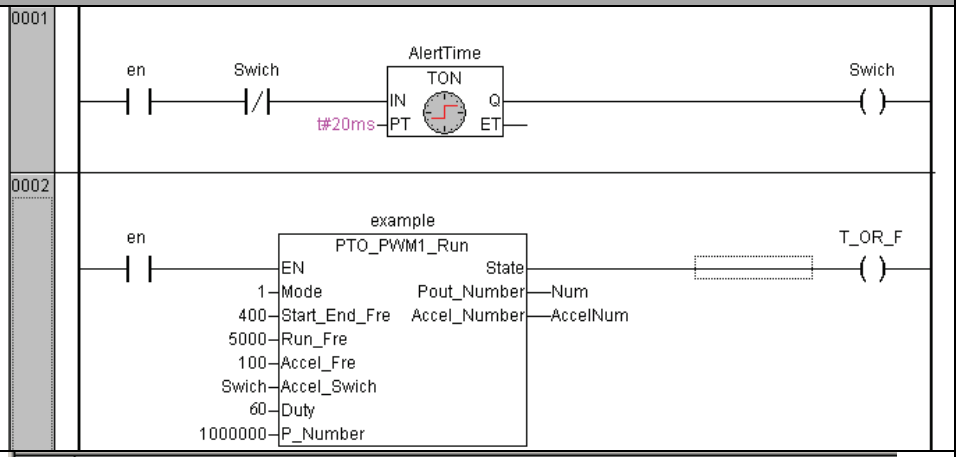
✧ 指令使用举例（梯形图和结构化文本）

变量定义

VAR | VAR_INPUT | VAR_OUTPUT | VAR_IN_OUT | CONSTANT | RETAIN

	名称	地址	类型	初始值	注释
0001	en		BOOL		
0002	T_OR_F		BOOL		
0003	AlertTime		TON		
0004	Num		DWORD		
0005	Swich		BOOL		
0006	example		PTO_PWM1_Run		
0007	AccelNum		DWORD		

编程语言：梯形图（LD）



结构化文本（ST）

```
0001 AlertTime(in:=en AND NOT swich,pt:=t#20ms);
0002 SWICH:=AlertTime.Q;
0003
0004 example(en:=en,mode:=1,start_End_Fre:=400,Run_Fre:=5000,Accel_Fre:=100,
0005 Accel_Swich:=Swich,duty:=60,P_Number:=1000000);
0006 T_OR_F:=example.State;
0007 NUM:=example.Pout_Number;
0008 AccelNum:=example.Accel_Number;
```

程序说明：

- EN 置位并保持时（上升沿），Q0.3 开始发送脉冲，脉冲的频率为 400Hz，Num 显示 0，AccelNum 显示 0。此后每过 20ms 脉冲频率增加 100Hz，直到脉冲频率增加到 5000Hz，AccelNum 显示加速阶段发送的脉冲数（受扫描周期影响），然后开始以 5000Hz 匀速运行。
- EN 复位时，Q0.3 停止发送脉冲，Q 等于 0。
- 因为选择 Mode=1，所以 P_Number=1000000 无效，输出 PWM，占空比为设置的 60，Q0.3 将一直发送脉冲，直到 EN 等于 0。

✧ 步进电机升降速曲线控制方法

步进电机可以开环方式控制而无需反馈就能对位置和速度进行控制。但也正是因为负载位置对控制电路没有反馈，步进电机就必须正确响应每次励磁变化。如果励磁频率选择不当，电机不能够移到新的位置，那么实际的负载位置相对控制器所期待的位置出现永久误差，即发生

失步现象或过冲现象。因此步进电机开环控制系统中，应该防止失步和过冲。

失步和过冲现象分别出现在步进电机启动和停止的时候。一般情况下，系统的极限启动频率比较低，而要求的运行速度往往比较高。如果系统以要求的运行速度直接启动，因为该速度已超过极限启动频率而不能正常启动，轻则可能发生丢步，重则根本不能启动，产生堵转。系统运行起来以后，如果达到终点时立即停止发送脉冲串，令其立即停止，则由于系统惯性作用，电机转子会转过平衡位置。如果负载的惯性很大，会使步进电机转子转到接近终点平衡位置的下一个平衡位置，并在该位置停下。

为了克服失步和过冲现象，应在步进电机启停时进行如图 5-24-1 所示的升降速控制，其中 ls 为启动频率，hs 为运行频率。

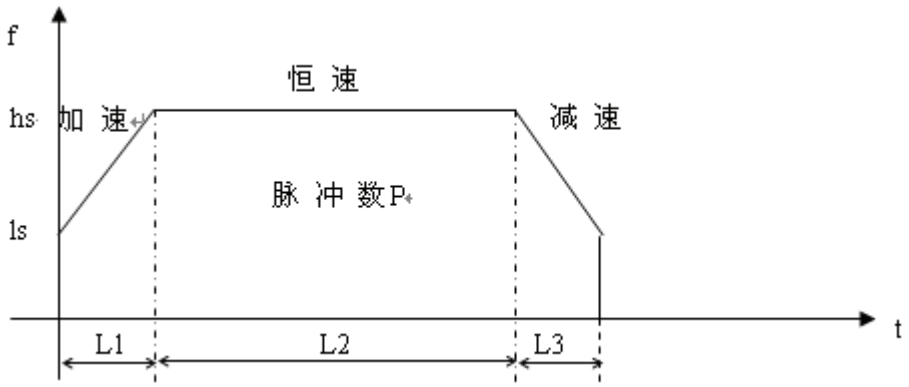


图 5-24-1 步进电机升降运行曲线

从图 5-24-1 可以看出，L2 段为恒速运行，L1 段为升频，L3 段为降频。按照“失步”的定义，如果在 L1 及 L3 段上升及下降的控制频率变化大于步进电机的响应频率变化，步进电机就会失步，失步会导致步进电机停转，经常会影响系统的正常工作。因此，在步进电机变速运行中，必须进行正确的升降速控制。

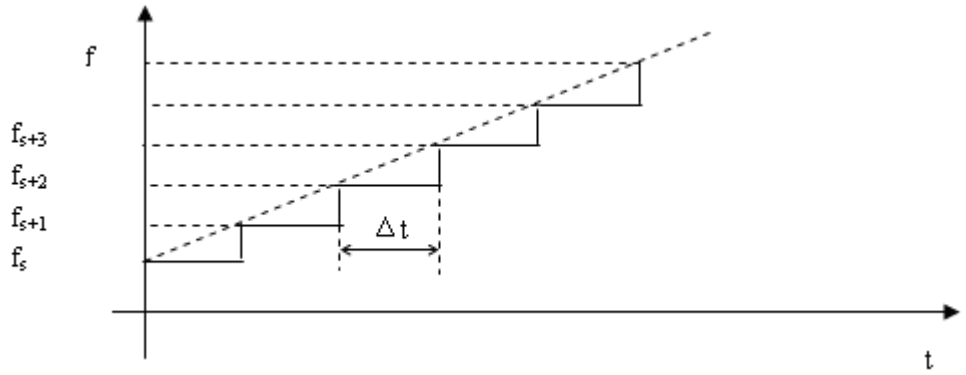


图 5-24-2 速度的离散化过程

PLC 在控制步进电机加减速的过程中，一般用离散方法逼近理想的升降速曲线。加减速的斜率在直线加速过程中，速度不是连续变化，而是按分档阶段变化，为与要求的升速斜率相逼近，必须确定每个台阶上的运行时间，见图 5-24-2。时间 Δt 越小，升速越快，反之越慢。Δt 的大小可由理论或实验确定，以升速最快而又不失步为原则。

附录 A

A.1 G3 指令速查表

基 本 指 令			
指令类型	指令说明	指令名	所在库
算术运算指令	加法指令	ADD（FUN）	无
	乘法指令	MUL（FUN）	
	减法指令	SUB（FUN）	
	除法指令	DIV（FUN）	
	取余指令	MOD（FUN）	
赋值指令	赋值指令	MOVE（FUN）	无
逻辑运算指令	与指令	AND（FUN）	无
	或指令	OR（FUN）	
	异或指令	XOR（FUN）	
	取非指令	NOT（FUN）	
移位指令	左移指令	SHL（FUN）	无
	右移指令	SHR（FUN）	
	循环左移指令	ROL（FUN）	
	循环右移指令	ROR（FUN）	
选择指令	二选一指令	SEL（FUN）	无
	取最大值指令	MAX（FUN）	
	取最小值指令	MIN（FUN）	
	极限值指令	LIMIT（FUN）	
	多选一指令	MUX（FUN）	
比较指令	大于指令	GT（FUN）	无
	小于指令	LT（FUN）	
	大于等于指令	GE（FUN）	
	小于等于指令	LE（FUN）	
	等于指令	EQ（FUN）	
	不等于指令	NE（FUN）	
数据类型转换指令	布尔类型转换指令	BOOL_TO_<TYPE>(FUN)	无
	字节类型转换指令	BYTE_TO_<TYPE>(FUN)	
	日期类型转换指令	DATE_TO_<TYPE>(FUN)	
	长整型转换指令	DINT_TO_<TYPE>(FUN)	
	日期时间类型转换指令	DT_TO_<TYPE>(FUN)	
	双字类型转换指令	DWORD_TO_<TYPE>(FUN)	
	整数类型转换指令	INT_TO_<TYPE>(FUN)	

指令类型	指令说明	指令名	所在库
数据类型转换指令	字类型转换指令	WORD_TO_<TYPE>(FUN)	无
	实数类型转换指令	REAL_TO_<TYPE>(FUN)	
	短整型转换指令	SINT_TO_<TYPE>(FUN)	
	字符类型转换指令	STRING_TO_<TYPE>(FUN)	
	时钟类型转换指令	TIME_TO_<TYPE>(FUN)	
	时间类型转换指令	TOD_TO_<TYPE>(FUN)	
	无符号长整型转换指令	UDINT_TO_<TYPE>(FUN)	
	无符号整型转换指令	UINT_TO_<TYPE>(FUN)	
	无符号短整型转换指令	USINT_TO_<TYPE>(FUN)	
	截短转换指令	TRUNC(FUN)	
初等数学运算指令	绝对值指令	ABS(FUN)	无
	平方根指令	SQRT(FUN)	
	自然对数指令	LN(FUN)	
	常用对数指令	LOG(FUN)	
	指数指令	EXP(FUN)	
	正弦指令	SIN(FUN)	
	余弦指令	COS(FUN)	
	正切指令	TAN(FUN)	
	反正弦指令	ASIN(FUN)	
	反余弦指令	ACOS(FUN)	
	反正切指令	ATAN(FUN)	
	幂指令	EXPT(FUN)	
地址运算指令	取地址指令	ADR(FUN)	无
	取地址内容指令	^(FUN)	
	位地址指令	BITADR(FUN)	
	索引指令	INDEXOF（FUN）	
	数据类型大小指令	SIZEOF（FUN）	
调用指令	调用运算指令	CAL(FUN)	无
初始化操作指令	初始化操作指令	INI(FUN)	无
字符串指令	结合字符串指令	CONCAT(FUN)	Standard.lib
	删除字符指令	DELETE(FUN)	
	查找字符串指令	FIND(FUN)	
	插入字符串指令	INSERT(FUN)	
	左边取字符串指令	LEFT(FUN)	
	字符串长度指令	LEN(FUN)	
	中间取字符串指令	MID(FUN)	
	替换字符串指令	REPLACE(FUN)	
	右边取字符串指令	RIGHT(FUN)	
库版本信息指令	读取库版本查看	Version_Util(FUN)	Util.lib
软件版本信息指令	读取软件版本信息	SyslibGetVersion2300(FUN)	SysLibC16x.lib

指令类型	指令说明	指令名	所在库
事件使用指令	事件调用	SysCallbackRegister(FUN)	SysLibCallback.lib
	解除事件调用	SysCallbackUnregistrer(FUN)	
检查指令	数组边界检查	CheckBounds(FUN)	Check.lib
	字节型除数为零检查	CheckDivByte(FUN)	
	字型除数为零检查	CheckDivWord(FUN)	
	双字形型除数为零检查	CheckDivDword(FUN)	
	实型除数为零检查	CheckDivReal(FUN)	
	整型边界检查	CheckRangeSigned(FUN)	
	无符号整型边界检查	CheckRangeUnsigned(FUN)	
BCD 转换指令	BCD 码转整型指令	BCD_TO_INT(FUN)	Util.lib
	整型转 BCD 码指令	INT_TO_BCD(FUN)	
位/字节操作指令	位提取指令	EXTRACT(FUN)	Util.lib
	位整合指令	PACK(FUN)	
	位赋值指令	PUTBIT(FUN)	
	位拆分指令	UNPACK(FB)	
高等数学运算指令	微分	DERIVATIVE(FB)	Util.lib
	积分	INTEGRAL(FB)	
	整型统计	STATISTICS_INT(FB)	
	实型统计	STATISTICS_REAL(FB)	
	平方偏差	VARIANCE(FB)	
控制器指令	比例控制器	P(FB)	Util.lib
	比例微分控制器	PD(FB)	
	比例积分微分控制器	PID(FB)	
	比例积分微分控制器	PID_FIXCYCLE(FB)	
信号发生器指令	脉冲信号发生器	BLINK(FB)	Util.lib
	典型周期信号发生器	GEN(FB)	
SFC 动作控制指令	SFC 动作控制	SFCActionControl(FB)	Iecsfc.lib
函数操纵器指令	特征曲线	CHARCURVE(FB)	Util.lib
	整型限速	RAMP_INT(FB)	
	实型限速	RAMP_REAL(FB)	
模拟量处理指令	滞后	HYSTERESIS(FB)	Util.lib
	上下限报警	LIMITALARM(FB)	
双稳态指令	置位优先双稳态器	SR(FB)	Standard.lib
	复位优先双稳态器	RS(FB)	
触发器	上升沿检测触发器	R_TRIG(FB)	Standard.lib
	下降沿检测触发器	F_TRIG(FB)	
计数器	递增计数器	CTU(FB)	Standard.lib
	递减计数器	CTD(FB)	
	递增递减计数器	CTUD(FB)	
定时器	普通定时器	TP(FB)	Standard.lib
	通电延时定时器	TON(FB)	
	断电延时定时器	TOF(FB)	
	实时时钟	RTC(FB)	

扩 展 指 令			
指令类型	**指令说明**	**指令名**	**所在库**
模拟量模块处理指令	模拟量输入	Analog_IN(FB)	Hollysys_PLC_Analog.lib
	模拟量输出	Analog_OUT(FB)	
RS232 口通讯指令	RS232 自由口通讯参数设置	Set_COMM_PRMT(FB)	Hollysys_PLC_Comm.lib
	RS232 自由口通讯数据发送	COMM_SEND(FB)	
	RS232 自由口通讯数据接收	COMM_RECEIVE(FB)	
	RS 232 恢复协议设置	Reset_COMM_PRMT(FB)	
RS485 口通讯指令	RS485 自由口通讯参数设置	Set_COMM2_PRMT(FB)	Hollysys_PLC_Comm2.lib
	RS485 自由口通讯数据发送	COMM2_SEND(FB)	
	RS485 自由口通讯数据接收	COMM2_RECEIVE(FB)	
	RS485 恢复协议设置	Reset_COMM2_PRMT	
Profibus-DP 模块调用指令	Profibus-DP 从站模块调用	DP_Slave(FB)	Hollysys_PLC_DPSlave.lib
以太网模块调用指令	以太网模块调用	EtherNet_TCP(FB)	Hollysys_PLC_EtherNet.lib
正反动作PID控制器	比例积分微分控制器	PID2(FB)	Hollysys_PLC_Util.lib
Modbus 校验指令	产生 Modbus CRC 校验	Generate_CRC(FB)	Hollysys_PLC_Modbus_CRC.lib
硬件实时时钟指令	设置实时时钟（DT 数据格式）	Set_HD_RTC(FB)	Hollysys_PLC_HdRtc.lib
	设置实时时钟（普通数据格式）	Set_HD_RTC_X(FB)	
	读取实时时钟日期/时间/星期	Get_HD_RTC(FB)	
实时时钟报警指令	读取硬件实时时钟报警时间/星期	Get_HDRTC_ALM(FB)	Hollysys_PLC_HdRtcALM.lib
	设置硬件实时时钟报警时间/星期	Set_HDRTC_ALM(FB)	
LM3104/5 实时时钟报警指令	读取硬件实时时钟报警时间/星期	Get_HDRTC_ALM(FB)	Hollysys_PLC_HdRtcALM_N.lib
	设置硬件实时时钟报警时间/星期	Set_HDRTC_ALM(FB)	
多段脉冲发送指令	通道 1.1 脉冲发送	PTOCtrl_0(FB)	Hollysys_PLC_PTOCtrl.lib
	通道 0.3 脉冲发送	PTOCtrl_1(FB)	
立即输出指令	立即输出	OutPut_Bit(FB)	Hollysys_PLC_IO.lib
	设定中断立即输出	Set_INT_OutPut(FB)	
工程量转换指令	工程量转换为 16 进制数	E_H(FB)	Hollysys_PLC_Cnvt.lib
	16 进制数转换工程量数据	H_E(FB)	
随机数发生指令	随机数发生	Rand(FB)	Hollysys_PLC_Math.lib
Modubs 从站地址指令	设置本机 Modbus 从站通讯地址	SET_LOCAL_ADDRESS (FB)	Hollysys_PLC_Ex.lib

	读取本机 Modbus 从站通讯地址	GET_LOCAL_ADDRESS (FB)	
模拟电位器	读取模拟电位器值	POT(FB)	Hollysys_PLC_Ex.lib
指令类型	**指令说明**	**指令名**	**所在库**
系统看门狗复位	系统看门狗复位	HD_WDT_Reset(FB)	Hollysys_PLC_Ex.lib
单相计数	T2 高速计数器	HD_CTUD_T2(FB)	Hollysys_PLC_Ex_CT.lib
	T3 高速计数器	HD_CTUD_T3(FB)	
	T4 普通计数器	HD_CTUD_T4(FB)	
	T7 高速计数器	HD_CTU_T7(FB)	
两相计数	T2 两相高速计数器	HD_DCTUD_T2(FB)	Hollysys_PLC_Ex_DCT.lib
	T3 两相高速计数器	HD_DCTUD_T3(FB)	
	T4 两相普通计数器	HD_DCTUD_T4(FB)	
两相 32 位计数	32 位高速计数器	HD_DCTUD32_T3(FB)	Hollysys_PLC_Ex_DCT32.lib
中断定时器	中断定时器	HD_TIMER_T7(FB)	Hollysys_PLC_Ex_TIMER.lib
	重载定时器	HD_CLEAR_T7(FB)	
	停止定时器	HD_STOP_T7(FB)	
外部中断	快速外部中断（适用于存储容量为 28K）	Fast_ExINT(FB)	Hollysys_PLC_Ex_ExINT.lib
	快速外部中断（适用于存储容量为 120K）	Fast_ExINT_E(FB)	
脉冲输出指令	PTO/PWM 脉冲输出	PTO_PWM0(FB)	Hollysys_PLC_Ex_PT.lib
		PTO_PWM1(FB)	
脉冲加减速输出指令	PTO/PWM 脉冲输出（加减速）	PTO_PWM0_Run(FB)	Hollysys_PLC__EX_PTRun.lib
		PTO_PWM1_Run(FB)	

A.2 IEC 标准指令表

G3 PLC 编程软件 PowerPro 遵循国际电工技术委员会（IEC）的 IEC61131-3 标准，在这个标准中明确规定了作为可编程控制器必须具有的指令，即标准指令。标准指令包括类型转换指令、数字运算指令、位移位指令、比较指令、类型转换指令、定时器、计数器等，所有 IEC61131-3 标准规定指令，见下表。

IEC 标 准 指 令		
指令类型	**指令说明**	**指令名**
算术运算指令	加法指令	ADD
	乘法指令	MUL
	减法指令	SUB
	除法指令	DIV
	取余指令	MOD
逻辑指令	与指令	AND
	或指令	OR
	异或指令	XOR
	取非指令	NOT
移位指令	左移指令	SHL

	右移指令	SHR
	循环左移指令	ROL
	循环右移指令	ROR
选择指令	二选一指令	SEL
	取最大值指令	MAX
	取最小值指令	MIN
	极限值指令	LIMIT
	多选一指令	MUX
比较指令	大于指令	GT
	小于指令	LT
	大于等于指令	GE
	小于等于指令	LE
	等于指令	EQ
	不等于指令	NE
类型转换指令	布尔类型转换指令	BOOL_TO_<TYPE>
	字节类型转换指令	BYTE_TO_<TYPE>
	日期类型转换指令	DATE_TO_<TYPE>
	长整型转换指令	DINT_TO_<TYPE>
	日期时间类型转换指令	DT_TO_<TYPE>
	双字类型转换指令	DWORD_TO_<TYPE>
	整数类型转换指令	INT_TO_<TYPE>
	字类型转换指令	WORD_TO_<TYPE>
	实数类型转换指令	REAL_TO_<TYPE>
	短整型转换指令	SINT_TO_<TYPE>
	字符类型转换指令	STRING_TO_<TYPE>
	时钟类型转换指令	TIME_TO_<TYPE>
	时间类型转换指令	TOD_TO_<TYPE>
	无符号长整型转换指令	UDINT_TO_<TYPE>
	无符号整型转换指令	UINT_TO_<TYPE>
	无符号短整型转换指令	USINT_TO_<TYPE>
	截短转换指令	TRUNC
初等数学运算指令	绝对值指令	ABS
	平方根指令	SQRT
	自然对数指令	LN
	常用对数指令	LOG
	指数指令	EXP
	正弦指令	SIN
	余弦指令	COS
	正切指令	TAN
	反正弦指令	ASIN
	反余弦指令	ACOS

	反正切指令	ATAN
	幂指令	EXPT
地址运算指令	取地址指令	ADR
	取地址内容指令	^
	位地址指令	BITADR
	索引指令	INDEXOF
	数据类型大小指令	SIZEOF
调用指令	调用运算指令	CAL
初始化操作指令	初始化操作指令	INI
字符串指令	结合字符串指令	CONCAT
	删除字符指令	DELETE
	查找字符串指令	FIND
	插入字符串指令	INSERT
	左边取字符串指令	LEFT
	字符串长度指令	LEN
	中间取字符串指令	MID
	替换字符串指令	REPLACE
	右边取字符串指令	RIGHT
BCD 转换指令	BCD 码转整型指令	BCD_TO_INT
	整型转 BCD 码指令	INT_TO_BCD
双稳态指令	置位优先双稳态器	SR
	复位优先双稳态器	RS
触发器	上升沿检测触发器	R_TRIG
	下降沿检测触发器	F_TRIG
计数器	递增计数器	CTU
	递减计数器	CTD
	递增递减计数器	CTUD
定时器	普通定时器	TP
	通电延时定时器	TON
	断电延时定时器	TOF
	实时时钟	RTC

A.3 指令关联冲突速查表

部分指令因为使用了同一内部硬件，或者使用了相同的输入/输出端口，所以不可以同时使用。下表列出使用指令 A 时，与其相关联的所有指令 B，并列出指令 B 的可使用程度。

使用指令 A	关联指令 B	关联硬件	可使用程度
HD_CTUD_T2	HD_DCTUD_T2	T2 内部计数器、I0.0、I0.1 脉冲输入	不可使用指令 B
HD_CTUD_T3	HD_DCTUD_T3	T3 内部计数器、I0.2、I0.3 脉冲输入	不可使用指令 B
	HD_DCTUD32_T3		

使用指令 A	关联指令 B	关联硬件	可使用程度
HD_CTUD_T4	HD_DCTUD_T4	T4 内部计数器、I0.4、I0.5 脉冲输入	不可使用指令 B
	HD_DCTUD32_T3		
HD_CTU_T7	HD_DCTUD_T2	I0.6 脉冲输入与 I0.6 外部清零脉冲输入	不可使用 B 外部清零脉冲输入功能
	Fast_ExINT	I0.6 脉冲输入与 I0.6 快速外部中断 3 脉冲输入	不可使用 B 快速外部中断 3 脉冲输入通道
	Fast_ExINT_E		
	HD_TIMER_T7	T7	不可使用指令 B
HD_DCTUD_T2	HD_CTUD_T2	T2 内部计数器、I0.0、I0.1 脉冲输入	不可使用指令 B
	HD_CTU_T7	I0.6 外部清零脉冲输入与 I0.6 脉冲输入	使用 A 的外部清零脉冲输入则不可以使用 B
	Fast_ExINT	I0.6 外部清零脉冲输入与 I0.6 快速外部中断 3 脉冲输入	使用 A 的外部清零脉冲输入，则不可使用 B 快速外部中断 3 脉冲输入通道
	Fast_ExINT_E		
HD_DCTUD_T3	HD_CTUD_T3	T3 内部计数器、I0.2、I0.3 脉冲输入	不可使用指令 B
	HD_DCTUD32_T3		
	Fast_ExINT	I0.7 外部清零脉冲输入与 I0.7 快速外部中断 2 脉冲输入	使用 A 的外部清零脉冲输入，则不可使用 B 快速外部中断 2 脉冲输入
	Fast_ExINT_E		
HD_DCTUD_T4	HD_CTUD_T4	T4 内部计数器、I0.4、I0.5 脉冲输入	不可使用指令 B
	HD_DCTUD32_T3		
	Fast_ExINT	I1.0 外部清零脉冲输入与 I1.0 快速外部中断 1 脉冲输入	使用 A 的外部清零脉冲输入，则不可使用 B 快速外部中断 1 脉冲输入通道
	Fast_ExINT_E		
Fast_ExINT	HD_CTU_T7	I0.6 快速外部中断 3 脉冲输入与 I0.6 脉冲输入	使用 A 快速外部中断 3 脉冲输入，则不可使用 B
	HD_DCTUD_T2	I0.6 快速外部中断 3 脉冲输入与 I0.6 外部清零脉冲输入	使用 A 快速外部中断 3 脉冲输入，则不可使用 B 外部清零脉冲输入通道
	HD_DCTUD_T3	I0.7 快速外部中断 2 脉冲输入与 I0.7 外部清零脉冲输入	使用 A 快速外部中断 2 脉冲输入，则不可使用 B 外部清零脉冲输入通道
	HD_DCTUD_T4	I1.0 快速外部中断 1 脉冲输入通道、I1.0 外部清零脉冲输入	使用 A 快速外部中断 1 脉冲输入，则不可使用 B 外部清零脉冲输入通道
Fast_ExINT_E	HD_CTU_T7	I0.6 快速外部中断 3 脉冲输入与 I0.6 脉冲输入	使用 A 快速外部中断 3 脉冲输入，则不可使用 B
	HD_DCTUD_T2	I0.6 快速外部中断 3 脉冲输入与 I0.6 外部清零脉冲输入	使用 A 快速外部中断 3 脉冲输入，则不可使用 B 外部清零脉冲输入通道
	HD_DCTUD_T3	I0.7 快速外部中断 2 脉冲输入通道与 I0.7 外部清零脉冲输入	使用 A 快速外部中断 2 脉冲输入，则不可使用 B 外部清零脉冲输入通道
	HD_DCTUD_T4	I1.0 快速外部中断 1 脉冲输入通道与 I1.0 外部清零脉冲输入	使用 A 快速外部中断 1 脉冲输入，则不可使用 B 外部清零脉冲输入通道
PTO_PWM0	PTO_PWM0_Run	T12	不可使用指令 B
	PTOCtrl_0		
PTO_PWM1	PTO_PWM1_Run	T8、T13	不可使用指令 B
	PTOCtrl_1	T13	

使用指令 A	关联指令 B	关联硬件	可使用程度
PTO_PWM0_Run	PTO_PWM0	T12	不可使用指令 B
	PTOCtrl_0		
PTO_PWM1_Run	PTO_PWM1	T8、T13	不可使用指令 B
	PTOCtrl_1	T13	
PTOCtrl_0	PTO_PWM0	T12	不可使用指令 B
	PTO_PWM0_Run		
PTOCtrl_1	PTO_PWM1	T13	不可使用指令 B
	PTO_PWM1_Run		
HD_DCTUD32_T3	HD_CTUD_T3	T3 内部计数器、I0.2、I0.3 脉冲输入	不可使用指令 B
	HD_CTUD_T4	T4 内部计数器	
	HD_DCTUD_T3	T3 内部计数器、I0.2、I0.3 脉冲输入	
	HD_DCTUD_T4	T4 内部计数器	
HD_TIMER_T7	HD_CTU_T7	T7	不可使用指令 B

A.4 硬件模块状态信息

A.4.1 专用寄存器区

%MB0~%MB9 的 10 个字节，为专用寄存器，其中%MB0、%MB1 为错误标志，%MB2、%MB3 表示错误数据，%MB2 为数据低 8 位，%MB3 为数据高 8 位。

%MB0：

高位 低位

Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0

Bit0=1：扩展地址 0 模块通讯 CRC 校验错，错误数据无定义。
Bit1=1：扩展地址 1 模块通讯 CRC 校验错，错误数据无定义。
Bit2=1：扩展地址 2 模块通讯 CRC 校验错，错误数据无定义。
Bit3=1：扩展地址 3 模块通讯 CRC 校验错，错误数据无定义。
Bit4=1：扩展地址 4 模块通讯 CRC 校验错，错误数据无定义。
Bit5=1：扩展地址 5 模块通讯 CRC 校验错，错误数据无定义。
Bit6=1：扩展地址 6 模块通讯 CRC 校验错，错误数据无定义。

%MB1：

高位 低位

Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0

Bit0=1：SSC 硬件操作错，系统故障，错误数据为当前 SSCCON。
Bit1=1：ASC 硬件操作错，错误数据为当前 S0CON 值。
Bit2=1：ADC 硬件操作错，错误数据为当前 ADEIC 值。
Bit3=1：B 类硬件错，错误数据为当前 TFR 值。
Bit4=1：系统栈下溢错，错误数据为当前 TFR 值。
Bit5=1：系统栈上溢错，错误数据为当前 TFR 值。
Bit6=1，系统 NMI 错，错误数据为当前 TFR 值。

%MB2、%MB3：错误数据。

高字节 低字节

%MB3	%MB2

A.4.2 模块诊断区

采用 PowerPro 软件编程进行硬件模块配置时，为每个模块（包括 CPU 模块）在 M 区分配了 10 个字节的诊断区。CPU 模块的诊断区从%MB20 开始，扩展模块的诊断区从%MB30 开始，每个模块占 10 个字节。每个模块的起始地址可以从 PLC 配置中模块的基本参数项中读取。如图 A-4-1 所示。

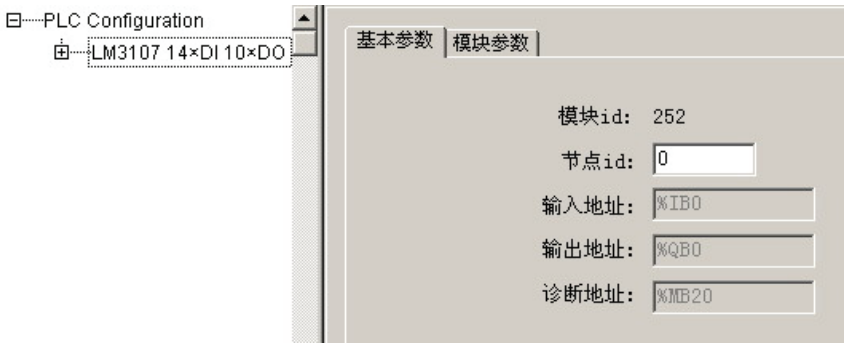


图 A-4-1

模块诊断字节的第 1 字节表示模块状态，不同的模块有不同的模块状态；对于模拟量输入模块从第 2 字节开始表示通道状态，每个通道占用 4bit， 低通道占用低 4 位，高通道占用高 4 位。

下面详细说明每种模块的模块状态定义：

✧ **CPU 模块诊断区**

- CPU 模块：只有 1 个字节为模块状态。

Bit7							Bit0
×	×	×	×	×	×	×	ID

- Bit0=1，模块 ID 配置正确； Bit0=0，模块 ID 配置错。
- Bit1~Bit7：保留。
- 第 2~10 字节保留。

✧ **开关量模块诊断区**

- 第 1 字节：

Bit7							Bit0
×	RD	×	×	×	×	×	ID

- Bit0=1，模块 ID 配置正确；Bit0=0，模块 ID 配置错。
- Bit6=1，CPU 模块成功读取扩展模块 ID；Bit6=0，CPU 模块未成功读取扩展模块 ID。
- Bit1~Bit5、Bit7：保留。
- 第 2 字节为第 1 字节中 ID 正确的情况下，读取到的扩展模块 ID，否则为 0。
- 第 3~10 字节保留。

✧ **模拟量输入模块诊断区**

第 1 字节：模块状态。

Bit7	6	5	4	3	2	1	Bit0
CRC	×	×	AD_OK	READY	CHANNEL	PWR	ID

- Bit0=1，模块 ID 配置正确；Bit0=0，模块 ID 配置错。
- Bit1=1，外供电 DC 24V 电源正常，Bit1=0，无外供 DC 24V 电源。
- Bit2=1，无通道错误，Bit2=0，有通道错误。
- Bit3=1，模块已经准备好，Bit3=0，模块没有准备好。
- Bit4=1，AD 芯片正常，Bit4=0，AD 芯片异常。
- Bit5-Bit6：保留。
- Bit7=1，扩展模块读取到的数据 CRC 错误，Bit7=0，扩展模块读取到的数据 CRC 正确。

第 2 字节以后为通道状态：

每 4 位表示一个通道的状态信息，不同的模块类型有不同的通道状态信息。

Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
第 2 通道（高通道）				第 1 通道（低通道）			

LM3310 通道状态定义，4 通道，占用 2、3 字节，其余字节保留：

- 0x1：断线（仅仅对 4-20mA 量程时有效）。
- 0x2：通道没有准备好。
- 0xF：通道无错误。
- 其他：保留。

LM3311 通道状态定义，4 通道，占用 2、3 字节，其余字节保留：

- 0x1：断偶。
- 0x2：通道没有准备好。
- 0xF：通道无错误。
- 其他：保留。

LM3312 通道状态定义，4 通道，占用 2、3 字节，其余字节保留：

- 0x0：超量程，电阻值过小（<39Ω）。
- 0x2：通道没有准备好。
- 0xF：通道无错误。
- 其它：保留。

LM3313 通道状态定义，8 通道，占用 2-5 字节，其余字节保留：

- 0x2: 通道没有准备好。
- 0xF: 通道无错误。
- 其他：保留。

LM3314 通道状态定义，8 通道，占用 2~5 字节，其余字节保留：

- 0x0: 超量程，电阻值过小（<600Ω）。
- 0x1: 超量程，电阻值过大（>100KΩ）。
- 0x2: 通道没有准备好。
- 0xF: 通道无错误。
- 其它：保留。

✧ **模拟量输出模块诊断区**

第 1 字节：模块状态。

Bit7	6	5	4	3	2	1	Bit0
CRC	×	×	×	×	×	PWR	ID

- Bit0=1，模块 ID 配置正确；Bit0=0，模块 ID 配置错。
- Bit1=1，外供电 DC 24V 电源正常，Bit1=0，无外供 DC 24V 电源。
- Bit2~6：保留
- Bit7=1，扩展模块读取到的数据 CRC 错误，Bit7=0，扩展模块读取到的数据 CRC 正确。

模拟量输出模块没有通道状态，字节 2~10 保留。

✧ **DP 模块诊断区**

第 1 字节：模块状态。

Bit7	6	5	4	3	2	1	Bit0
CRC	×	×	SPC3_STATE	OUTDATA_LEN	INDATA_LEN	×	ID

- Bit0=1，模块 ID 配置正确；Bit0=0，模块 ID 配置错。
- Bit1：保留。
- Bit2=1，PowerPro 中对 DP 模块配置的输入数据字节数与 DP-Master 组态软件中对 DP 模块配置的输出数据字节数匹配；Bit2=0，上述字节数不匹配。
- Bit3=1，PowerPro 中对 DP 模块配置的输出数据字节数与 DP-Master 组态软件中对 DP 模块配置的输入数据字节数匹配；Bit3=0，上述字节数不匹配。
- Bit4=1，SPC3 通讯正常；Bit4=0，SPC3 通讯错误。
- Bit5-Bit6：保留。
- Bit7=1，扩展模块读取到的数据 CRC 错误，Bit7=0，扩展模块读取到的数据 CRC 正确。

DP 模块无通道状态，字节 2~10 保留。

✧ **以太网模块诊断区**

第 1 字节：模块状态。

Bit7	6	5	4	3	2	1	Bit0
CRC	×	×	×	×	×	SWP	ID

- Bit0=1，模块 ID 配置正确；Bit0=0，模块 ID 配置错。
- Bit1=1，有网络数据交换，Bit1=0 无网络数据交换。
- Bit2-Bit6：保留。
- Bit7=1，扩展模块读取到的数据 CRC 错误，Bit7=0，扩展模块读取到的数据 CRC 正确。

以太网模块没有通道状态，字节 2~10 保留。

附录 B

B.1 电机循环启动发送脉冲举例

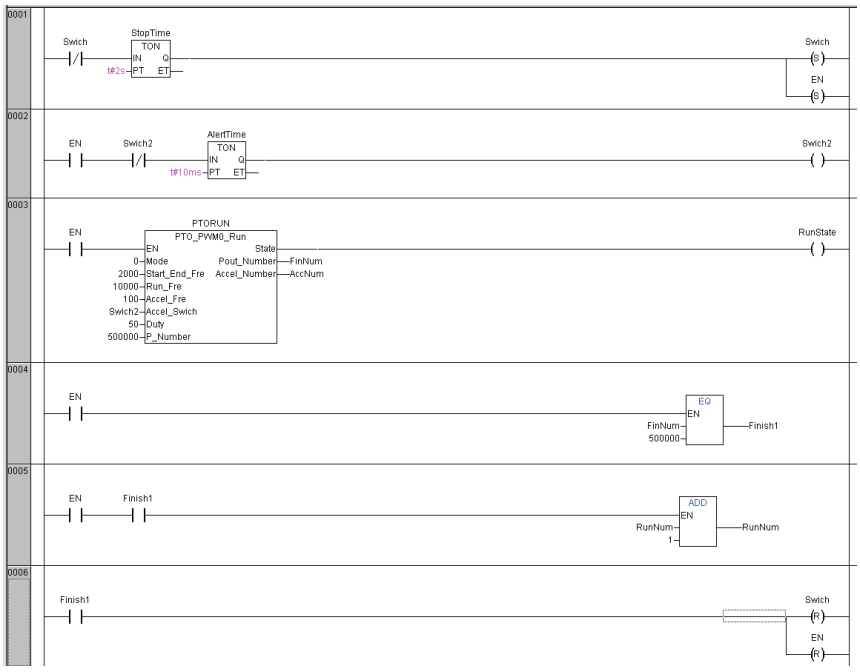
✧ 要求

步进电机以 2KHz 的速度启动，以 10KHz 的速度运行，发送 500000 个脉冲，然后停止 2 秒钟，再以 2KHz 的速度启动，以 10KHz 的速度运行，发送 500000 个脉冲，如此重复下去。

✧ 定义变量

```
PROGRAM PLC_PRG
VAR
    Swich: BOOL;
    Swich2: BOOL;
    EN: BOOL;
    Finish1: BOOL;
    RunNum: WORD;
    RunState: BOOL;
    FinNum: DWORD;
    AccNum: DWORD;
    AlertTime: TON;
    PTORUN: PTO_PWM0_Run;
    StopTime: TON;
END_VAR
```

✧ 梯形图程序



✧ **描述**

当程序开始运行时，2 秒后电机以 2KHz 的频率启动，以后每隔 10ms 增加 100Hz，当增加至 10KHz 后开始匀速运行，当剩余未发送脉冲数等于加速阶段发送的脉冲数时，开始减速运行，减速曲线与加速曲线对应，当 500000 个脉冲发送完毕，电机停止运行 2 秒钟，然后重新开始重复上述过程。

B.2 Profibus-DP 模块使用举例

✧ **要求**

PLC 通过 DP 从模块向 DP 主模块发送 8 个字节的数据，同时从 DP 主模块接收 8 个字节的数据。

✧ **变量定义**

```
PROGRAM PLC_PRG
VAR
    EN: BOOL;
    Example: DP_Slave;
    T_OR_F: BOOL;
    SendDataA: WORD;    PLC 向 DP 主站发送的数据 A
    SendDataB: WORD;    PLC 向 DP 主站发送的数据 B
    SendDataC: WORD;    PLC 向 DP 主站发送的数据 C
    SendDataD: WORD;    PLC 向 DP 主站发送的数据 D
    RecDataA: WORD;     DP 主站向 PLC 发送的数据 A
    RecDataB: WORD;     DP 主站向 PLC 发送的数据 B
    RecDataC: WORD;     DP 主站向 PLC 发送的数据 C
    RecDataD: WORD;     DP 主站向 PLC 发送的数据 D
END_VAR
```

✧ **软件配置**

➢ 配置 3401DP 从模块如图 B-2-1 中鼠标位置处所示。

InputDataLen_Byte 为 DP 主站向 PLC 发送的数据长度，输入接收的字节数 8；

OutputDataLen_Byte 为 PLC 向 DP 主站发送的数据长度，输入发送的字节数 8。

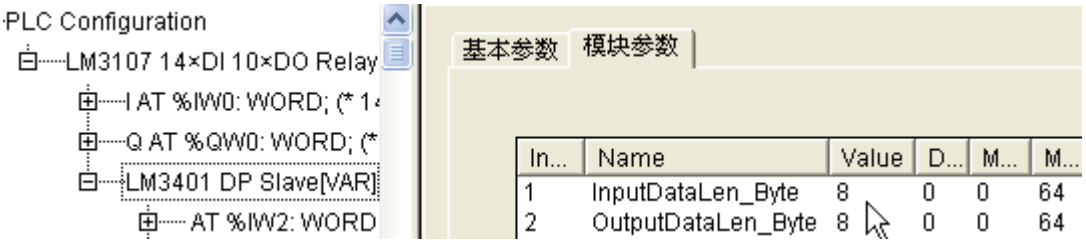


图 B-2-1

➢ DP_Slave 指令中的 Address 与图 B-2-2 所示的节点 id 一致，DP 主站向 PLC 发送的数

据 A、B、C、D 分别存放在下图所示的%IW2、%IW4、%IW6、%IW8 之中。

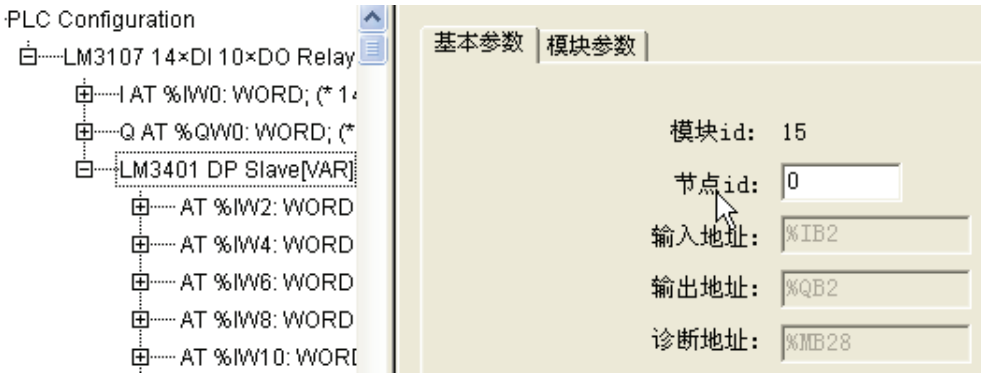


图 B-2-2

- PLC 向 DP 主站发送的数据 A、B、C、D 分别存放在图 B-2-3 所示的%QW2、%QW4、%QW6、%QW8 之中。

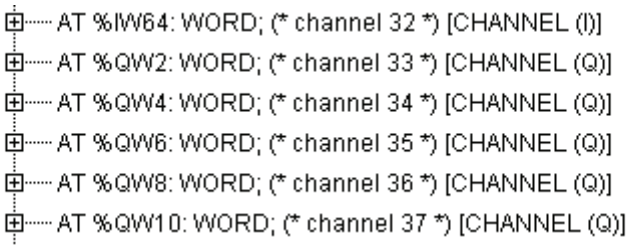


图 B-2-3

- 配置 DP 主站的接收和发送区，DP 从站的 QW 区数据会自动传送到 DP 主站的接收区，DP 主站的发送区数据会自动传送到 DP 从站的 IW 区。

✧ 梯形图程序

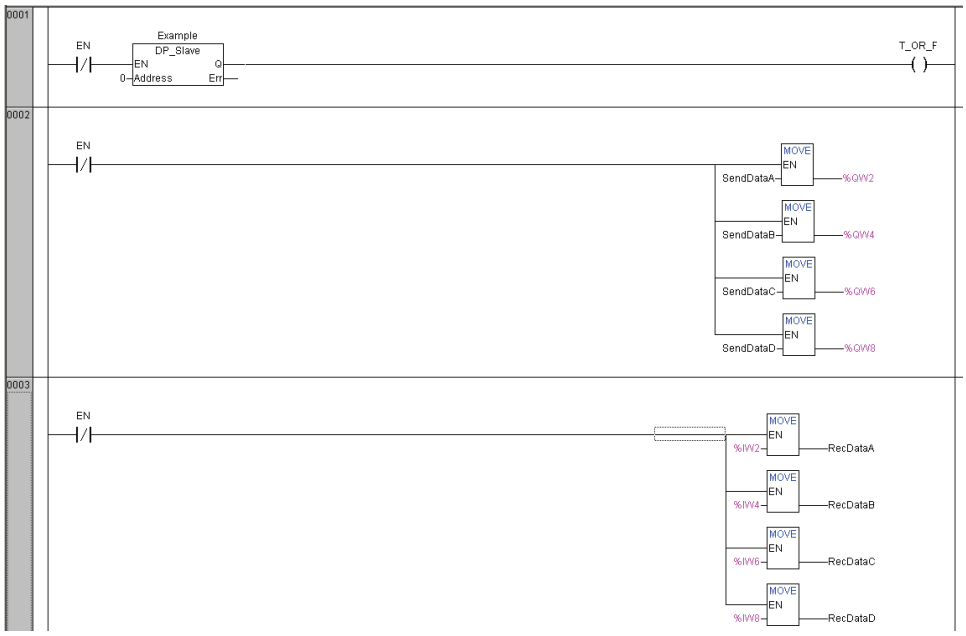


图 B-2-4

✧ **描述**

上电后，程序不停地将 SendDataA、SendDataB、SendDataC、SendDataD 的数据拷贝到%QW2、%QW4、%QW6、%QW8 之中，%QW2、%QW4、%QW6、%QW8 之中的数据会自动发送到 DP 主站的接收区。

DP 主站发送的数据会自动存放在%IW2、%IW4、%IW6、%IW8 之中，程序不断的把%IW2、%IW4、%IW6、%IW8 之中的数据分别拷贝到 RecDataA、RecDataB、RecDataC、RecDataD 之中。

B.3 以太网指令使用举例

本例程以组态王 V6.5 为例，PC 机（MODBUS/TCP 主站）通过以太网模块向 PLC 输入区发送 2 个字的数据，从 PLC 输出区接收 2 个字的数据，同时 PC 机按位操作向 PLC 输入区发送 1 个位，从 PLC 输出区接收 1 个位。我们首先介绍在 PowePro 中如何通过组态对 LM3403 进行操作，然后简单介绍一下 MODBUS/TCP 主站的组态方法（以组态王 V6.5 为例）。

1、PowerPro 程序

✧ **变量定义**

```
PROGRAM PLC_PRG
VAR
    EN: BOOL;
    Example: EtherNet_TCP;
    T_OR_F: BOOL;
    SendDataA: WORD;    (*PLC 向 MODBUS/TCP 主站发送的数据 A*)
    SendDataB: WORD;    (*PLC 向 MODBUS/TCP 主站发送的数据 B*)
    SendBitC: BOOL;     (*PLC 向 MODBUS/TCP 主站发送的位 C*)
    RecDataA: WORD;     (*MODBUS/TCP 主站向 PLC 发送的数据 A*)
    RecDataB: WORD;     (*MODBUS/TCP 主站向 PLC 发送的数据 B*)
    RecBitC: BOOL;      (*MODBUS/TCP 主站向 PLC 发送的位 C*)
END_VAR
```

✧ **软件配置**

➢ 配置 LM3403 以太网模块如图 B-3-1。

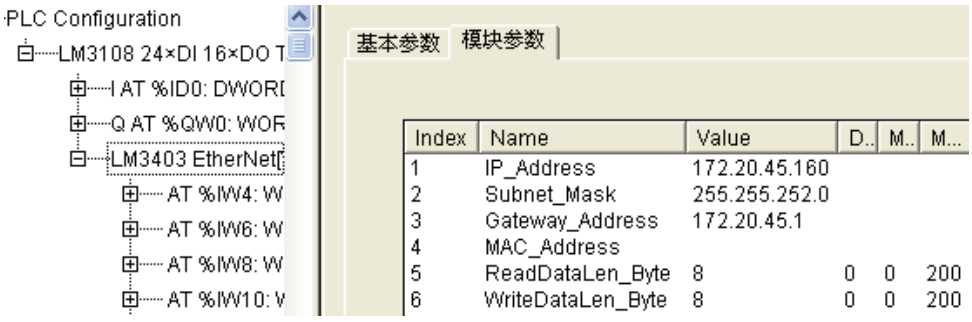


图 B-3-1

- IP_Address 为本以太网模块 IP 地址（必须与 PC 机在同一网段，且无冲突的 IP 地址）。
- Subnet_Mask 为子网掩码，与 PC 机的子网掩码一致。
- GateWay_Addres 为网关地址。
- MAC_Address 不填。
- ReadDataLen_Byte 为 PC 机向 PLC 发送的数据长度，输入接收的字节数 8（必须大于实际用到的长度，且最大 200）。
- WriteDataLen_Byte 为 PLC 向 PC 机发送的数据长度，输入发送的字节数 8（必须大于实际用到的长度，且最大 200）。

➢ 以太网指令中的 Address 与图 B-3-2 所示的节点 id 一致，PC 机向 PLC 发送的数据 A、B 分别存放在下图所示的%IW4、%IW6，位 C 存放在%IX8.0。

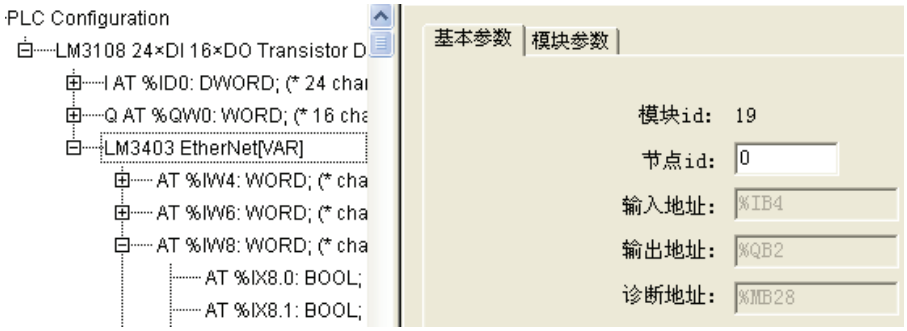


图 B-3-2

➢ PLC 向 PC 机发送的数据 A、B 分别存放在图 B-3-3 所示的%QW2、%QW4 之中， 位 C 存放在%QX6.0。

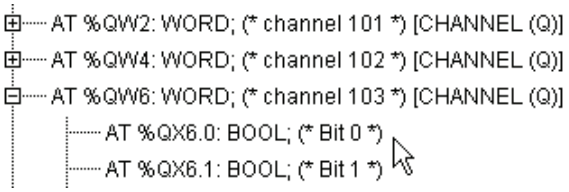


图 B-3-3

✧ **梯形图程序**

图 B-3-4

2、组态王程序

这里给出新建设备和数据词典中关键几步，其他画面按默认或查看组态软件相关帮助。

✧ **新建设备**

（1）在工程浏览器中新建设备如图 B-3-5，选中 Modbus（以太网）-网卡。

图 B-3-5

（2）选择“串口号”，选择一个不存在且与其他不冲突的串口号，如图 B-3-6：

图 B-3-6

（3）如图 B-3-7 输入设备地址，首先是从站 IP 地址与 PowerPro 中的一致，然后是空格，输入从站编号，如果不清楚可以点击地址帮助查看。

图 B-3-7

✧ **新建数据词典**

（1）如图 B-3-8，变量类型选择 I/O 整数，寄存器 30001 对应 PLC 输出区第 1 个字（整型即上面组态的%QW2）。

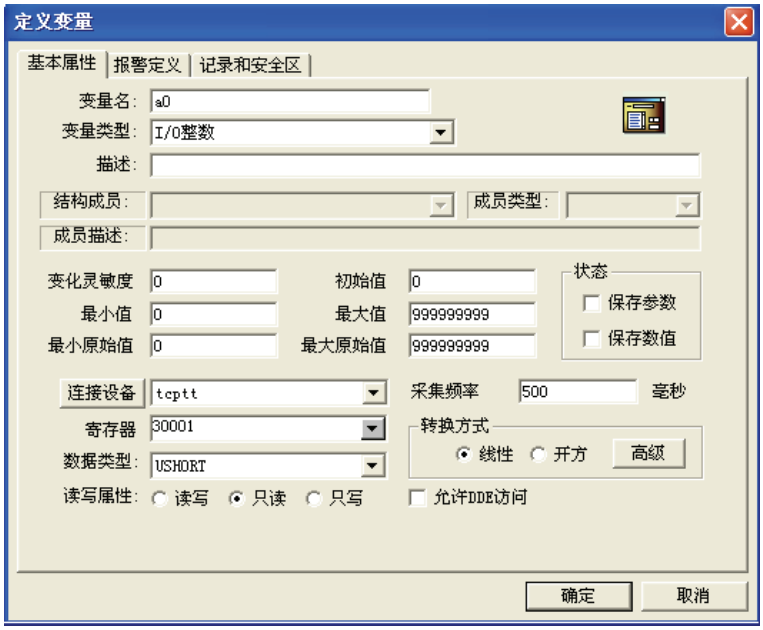


图 B-3-8

（2）如图 B-3-9，变量类型选择 I/O 整型，寄存器 30002 对应 PLC 输出区第 2 个字（整型即上面组态的%QW4，因为 PowerPro 的序号是以字节来标记的）。同理，寄存器 30003 对应 PLC 输出区第 3 个字（整型即上面组态的%QW6），以此类推。

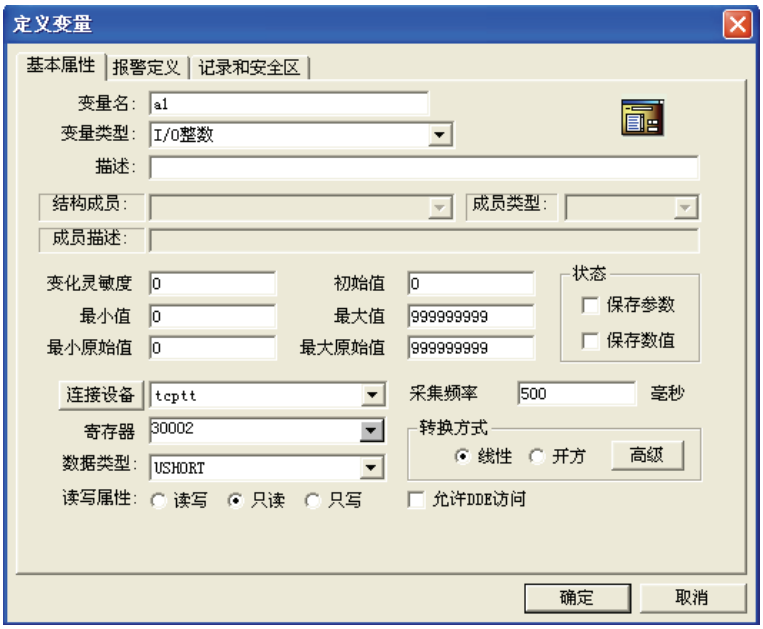


图 B-3-9

（3）如图 B-3-10，变量类型选择 I/O 整数，寄存器 40001 对应 PLC 输入区第 1 个字（整型即上面组态的%IW4）。

图 B-3-10

（4）如图 B-3-11，变量类型选择 I/O 整数，寄存器 40002 对应 PLC 输入区第 2 个字（整型即上面组态的%IW6）。同理，寄存器 40003 对应 PLC 输入区第 3 个字（整型即上面组态的%IW8），以此类推。

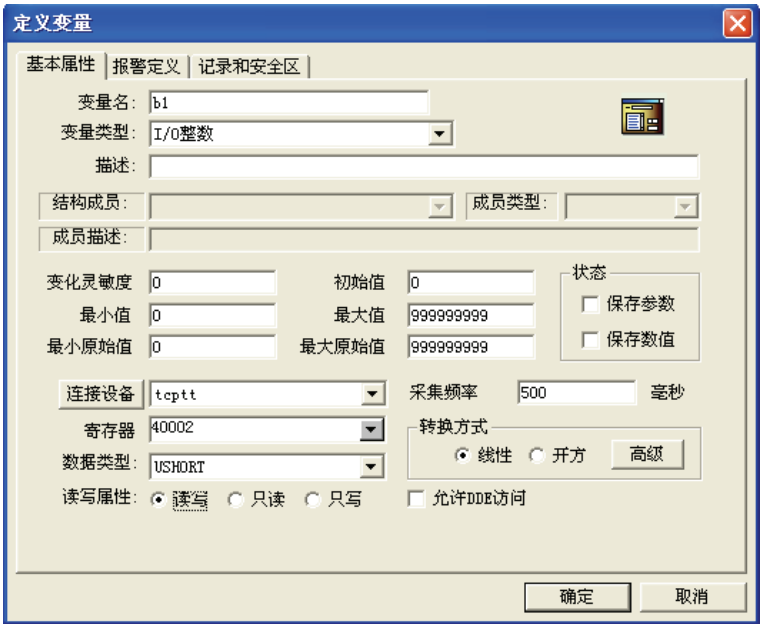


图 B-3-11

（5）如图 B-3-12，变量类型选择 I/O 离散型，寄存器 10033 对应 PLC 输出区第 33 个位（即上面组态的%QX6.0）。

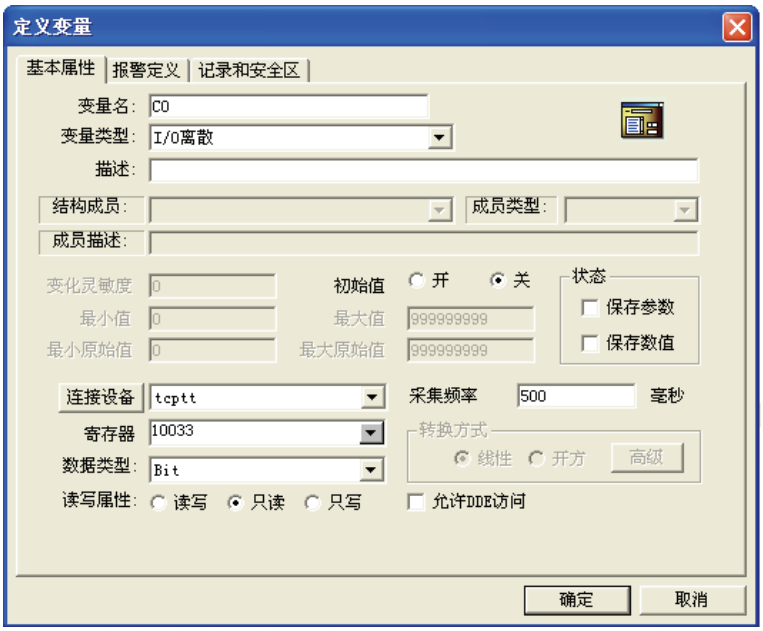


图 B-3-12

（6）如图 B-3-13，变量类型选择 I/O 离散型，寄存器 00033 对应 PLC 输入区第 33 个位（即上面组态的%IX8.0）。

图 B-3-13

B.4 中断关联事件使用举例

✧ **中断事件优先级顺序**

中断事件	优先级顺序
Fast External 0 interrupt	低 ↓ 高
Fast External 1 interrupt	
Fast External 2 interrupt	
Fast External 3 interrupt	
HD_TC7 interrupt	
HD_TC2 interrupt	
HD_TC3 interrupt	
HD_TC4 interrupt	
HD_RTC_ALM 0 interrupt	

✧ **要求**

不受 PLC 扫描周期的影响，I0.6 每到达一个上升沿，PLC 立刻响应该脉冲，产生一个中断，%MW100 中的值增加 10。I0.7 每到达一个上升沿，PLC 立刻响应该脉冲，产生一个中断，%MW102 中的值增加 15。

✧ **程序分析**

按上述要求，硬件可使用 LM3106 CPU 模块，软件需要使用以下指令：

- Fast_ExINT（快速外部中断）

程序分为如下 3 部分：

- 主程序——定义 3106 的快速外部中断模式
- INT3PRO——I0.6 脉冲到达执行的中断程序——%MW100 中的值增加 10
- INT2PRO——I0.7 脉冲到达执行的中断程序——%MW102 中的值增加 15

✧ 程序

主程序：

（1）首先要将程序所用到的库 Hollysys_PLC_Ex_ExINT.lib 添加到库管理器，如何添加库请参见 1.5 节。

（2）按照要求，I0.6、I0.7 每到达一个上升沿产生中断，I1.0 不使用，则 Fast_ExINT 指令的 Mode=16#50，主程序变量定义与梯形图如图 B-4-1。

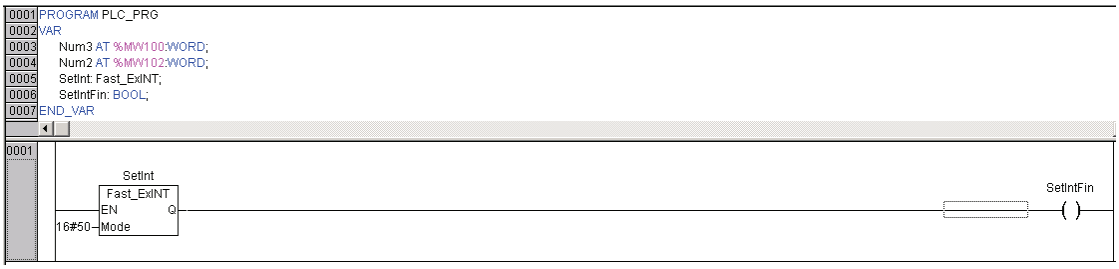


图 B-4-1

（3）由于我们使用了快速外部中断 3（I0.6）、快速外部中断 2（I0.7），所以在图 B-4-2 鼠标所示位置前打勾。

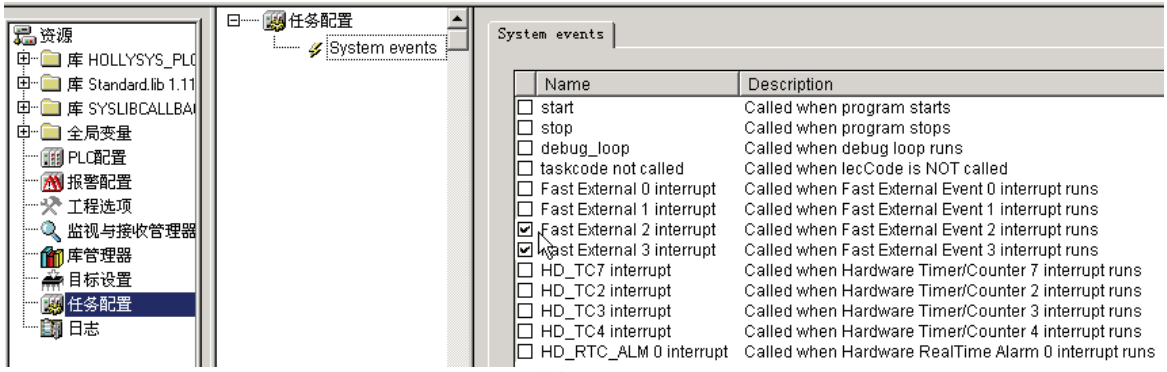


图 B-4-2

（4）在 Fast Extennal 2 interrupt 和 Fast Extennal 3 interrupt 后面分别创建一个子程序 INT2Pro、INT3Pro，分别点击 Create POU，则 2 个中断程序被创建成功，如图 B-4-3。

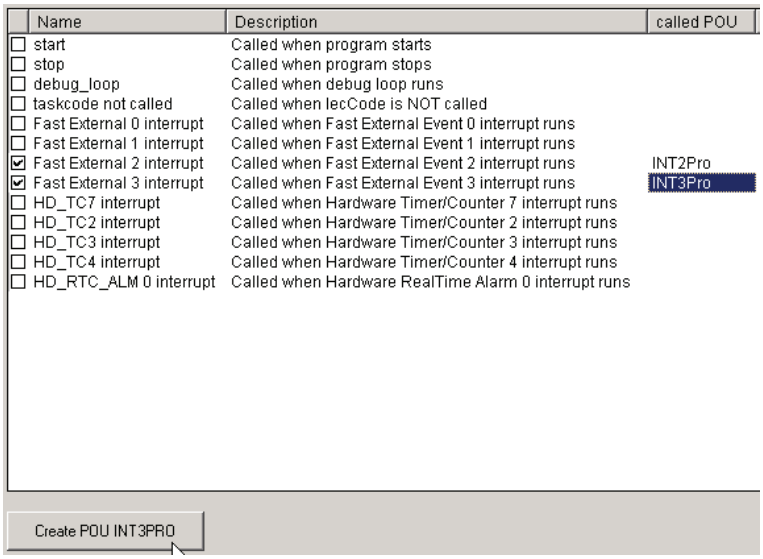


图 B-4-3

（5）创建的中断程序，默认为 ST 语言，可以选择转换为 LD，转换前需要编译通过，如图 B-4-4。

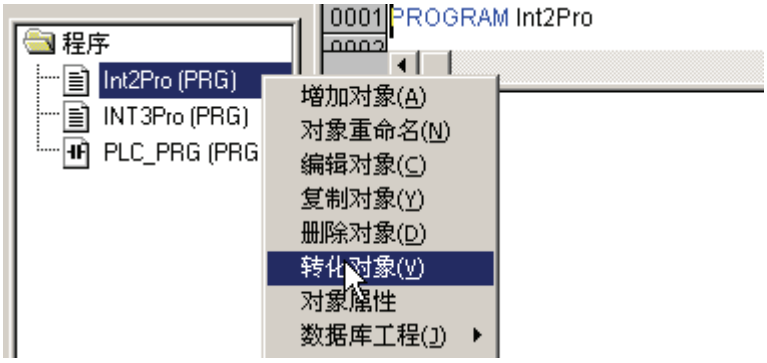


图 B-4-4

（6）INT3Pro 中断程序梯形如图 B-4-5。

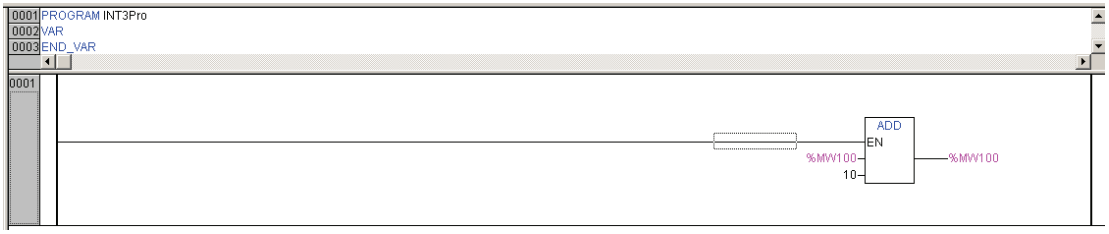


图 B-4-5

（7）INT2Pro 中断程序梯形图如图 B-4-6。

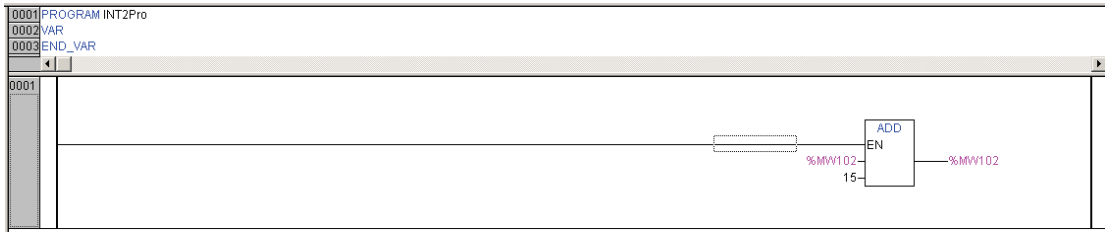


图 B-4-6

B.5 自由口通讯使用举例

✧ 要求

一检测仪表与 LM3108 模块进行 485 自由口通讯，仪表发出的协议格式如下：

自由协议格式（共 18 个字节）																	
STX	A	B	C	X	X	X	X	X	X	Y	Y	Y	Y	Y	Y	CR	CKS
起始符（02H）	状态字 ABC			重量						皮重						回车 0DH	校验

状态字 A			
Bit2	Bit1	Bit0	小数点位置
0	0	0	XXXX00
0	0	1	XXXXX0
0	1	0	XXXXXX
0	1	1	XXXXX.X
1	0	0	XXXX.XX
1	0	1	XXX.XXX
1	1	0	XX.XXXX
1	1	1	X.XXXXX

LM3108 需要解析的部分是状态字 A 中的前三位 bit 0、bit 1、bit 2，表示后面“重量”处数据的小数点位置，另外，就是“重量”所对应的具体的数字。仪表发送 0～10 之间的数值，LM3108 进行解析后输出 0～10V 的模拟量信号。

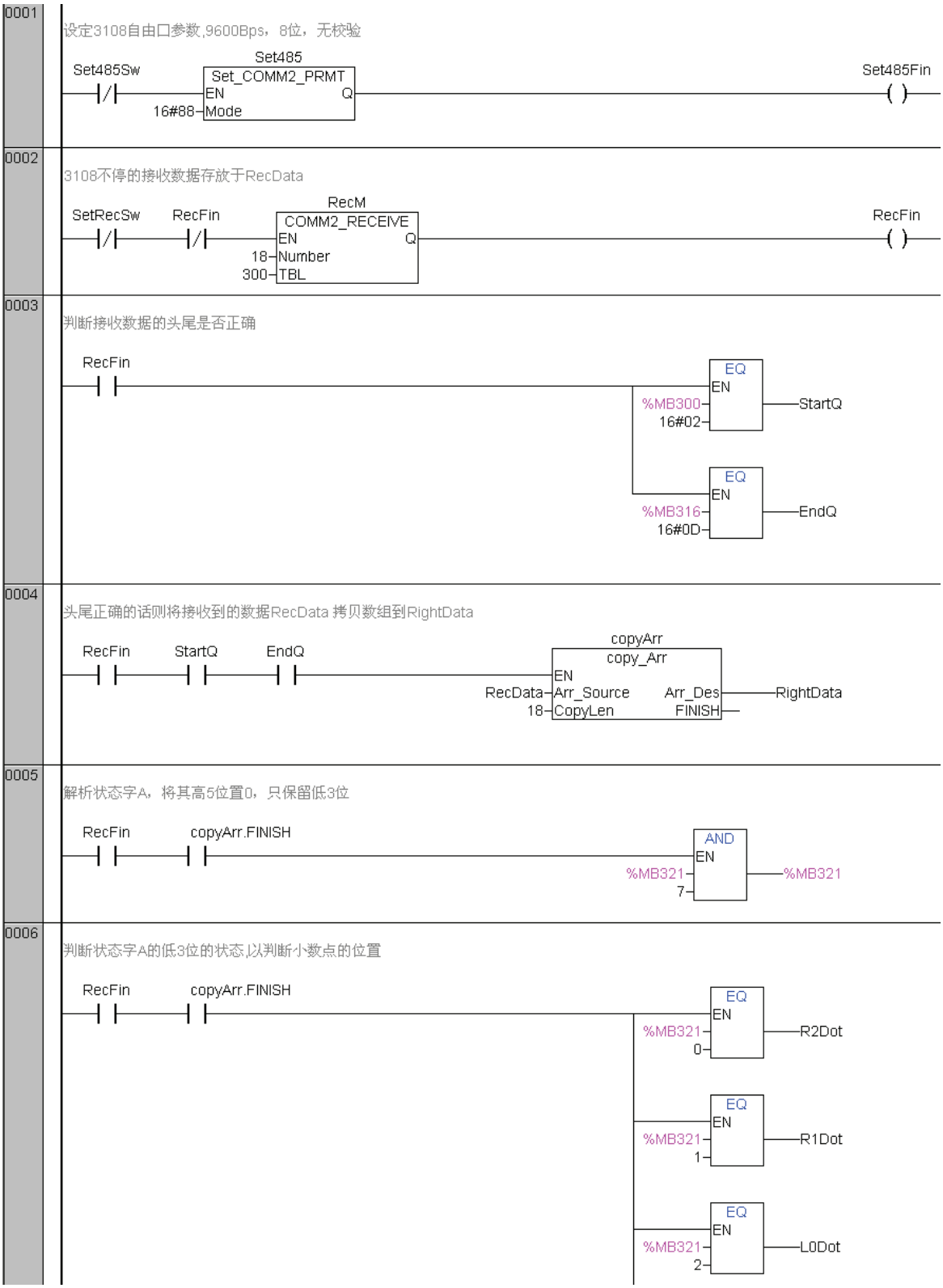
通讯参数为：9600Bps，8 位，无校验。

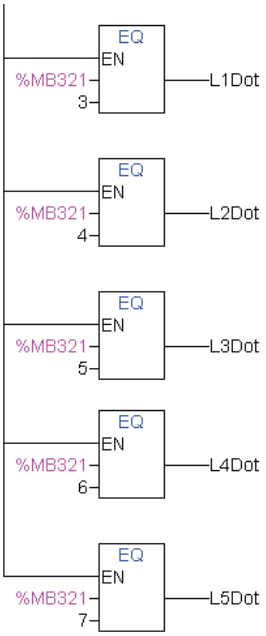
✧ 程序

变量定义

```
PROGRAM PLC_PRG
VAR
    RecFin: BOOL;
    Set485: Set_COMM2_PRMT;
    RecData AT %MB300 : ARRAY [1..18] OF BYTE;
    RightData AT %MB320 : ARRAY [1..18] OF BYTE;
    Set485Sw: BOOL;
    Set485Fin: BOOL;
    SetRecSw: BOOL;
    RecM: COMM2_RECEIVE;
    StartQ: BOOL;
    EndQ: BOOL;
    copyArr: copy_Arr;
    L5Dot: BOOL;
    L4Dot: BOOL;
    L3Dot: BOOL;
    L2Dot: BOOL;
    L1Dot: BOOL;
    L0Dot: BOOL;
    R1Dot: BOOL;
    R2Dot: BOOL;
    ResNum: DWORD;
    AddAllNum: AddNum;
    ResNumTemp: DWORD;
    DIVNUM: DWORD;
    OUTNUMH AT %MW202 : WORD;
    OUTNUML AT %MW200 : WORD;
    OUTNUM AT %MD200 : DWORD;
    AnOutInit: BOOL;
    AnOut: Analog_OUT;
    AnOutFin: BOOL;
END_VAR
```

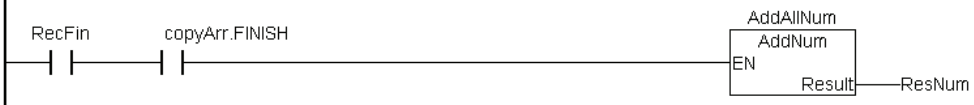
✧ 主程序梯形图程序


0001
设定3108自由口参数,9600Bps，8位，无校验
Set485Sw
Set485
Set_COMM2_PRMT
EN
Q
16#88
Mode
Set485Fin
0002
3108不停的接收数据存放于RecData
SetRecSw
RecFin
RecM
COMM2_RECEIVE
EN
Q
18
Number
300
TBL
RecFin
0003
判断接收数据的头尾是否正确
RecFin
EQ
EN
%MB300
16#02
StartQ
EQ
EN
%MB316
16#0D
EndQ
0004
头尾正确的话则将接收到的数据RecData 拷贝数组到RightData
RecFin
StartQ
EndQ
copyArr
copy_Arr
EN
RecData
Arr_Source
Arr_Des
RightData
18
CopyLen
FINISH
0005
解析状态字A，将其高5位置0，只保留低3位
RecFin
copyArr.FINISH
AND
EN
%MB321
7
%MB321
0006
判断状态字A的低3位的状态,以判断小数点的位置
RecFin
copyArr.FINISH
EQ
EN
%MB321
0
R2Dot
EQ
EN
%MB321
1
R1Dot
EQ
EN
%MB321
2
L0Dot



0007

将"重量"处的每个字节的数拼成一个DWORD型的数据



0008

转换成模拟量输出模块需要的数据0-1638,小数点位置在下面程序标定

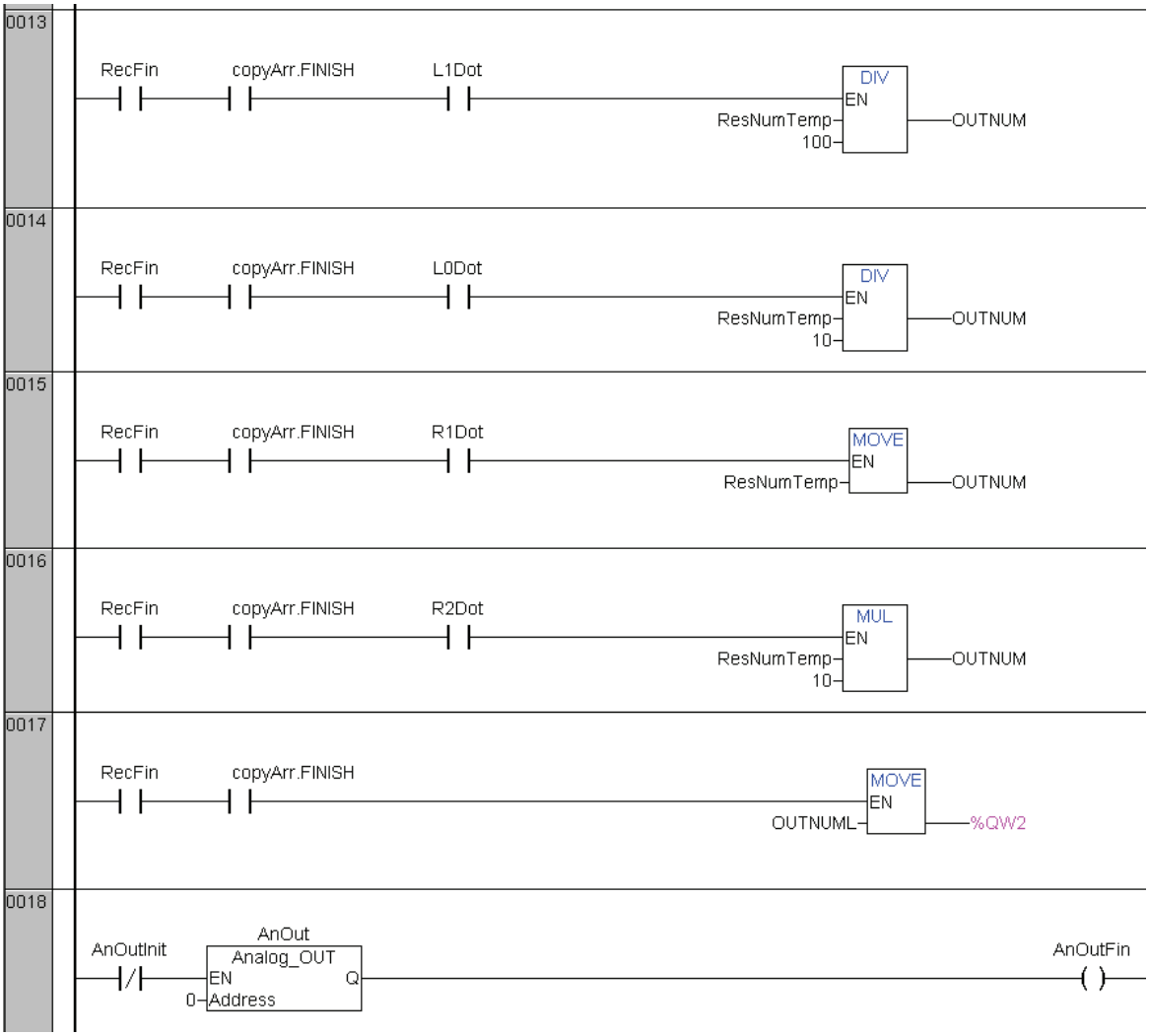
0009

判断小数点的位置,输出数据

0010

0011

0012



✧ **AddNum 自定义功能块程序**

```
0001 FUNCTION_BLOCK AddNum
0002 VAR_INPUT
0003 END_VAR
0004 VAR_OUTPUT
0005     Result: DWORD;
0006 END_VAR
0007 VAR
0008 END_VAR
```

```
0001
0002 Result := %MB324*100000+%MB325*10000+%MB326*1000+%MB327*100+%MB328*10+%MB329;
```

✧ **copy_Arr 自定义功能块程序**

```
0001 FUNCTION_BLOCK copy_Arr
0002 VAR
0003     i: WORD;
0004 END_VAR
0005 VAR_INPUT
0006     Arr_Source: ARRAY [1..18] OF BYTE;
0007     CopyLen: WORD;
0008 END_VAR
0009 VAR_OUTPUT
0010     Arr_Des: ARRAY [1..18] OF BYTE;
0011     FINISH: BOOL := FALSE;
0012 END_VAR
```

```
0001 FOR i:=1 TO CopyLen BY 1 DO
0002     Arr_Des[i]:=Arr_Source[i];
0003 END_FOR;
0004 FINISH:=TRUE;
```

B.6 PID 控制器使用举例

✧ **要求**

PLC 通过对信号发生器产生信号运算，模拟 PID 控制器的测量值，通过 PID 调节后的输出值经过 E_H 转换给输出模块通道。

✧ **变量定义**

```
PROGRAM PLC_PRG
VAR
    GEN1: GEN;                  信号发生器，用来模拟采集信号
    VAR1: INT;                  信号发生器的输出值
    KP1: REAL := 0.5;           比例系数，初始值设为 0.5
    TV1: DWORD := 4;            微分时间，初始值设为 4
    Tn1: DWORD := 100;          积分时间，初始值设为 100
    pid1: PID;                  PID 控制器
    en: BOOL;
    cyc_num: INT := 10000;      信号的个数，初始值设为 10000
    perio_num: TIME := T#4000ms;  信号周期，设为 4 秒
    VAR_ACTUAL: REAL;           PID 测量值
    E_H1: E_H;                  工程量转换数字量指令，结果给模拟量输出端口
    VAR_Y: REAL;                PID 控制器输出值
    ANA_IN: Analog_IN;          模拟量输入指令
    ANA_OUT: Analog_OUT;        模拟量输出指令
END_VAR
```

✧ **PLC 配置**

如图 B-6-1 中所示，配置 CPU 模块 LM3107、一个模拟量输入模块 LM3310 和一个模拟量输出模块 LM3320，PID 控制器必定需要一个被控制量的当前值，也需要 PID 控制器输出值转换

为执行对象动作值，配置中分别对应模拟量输入和模拟量输出模块。在本例中，为了便于试验将被控制量当前值用信号发生器产生的正弦波代替。

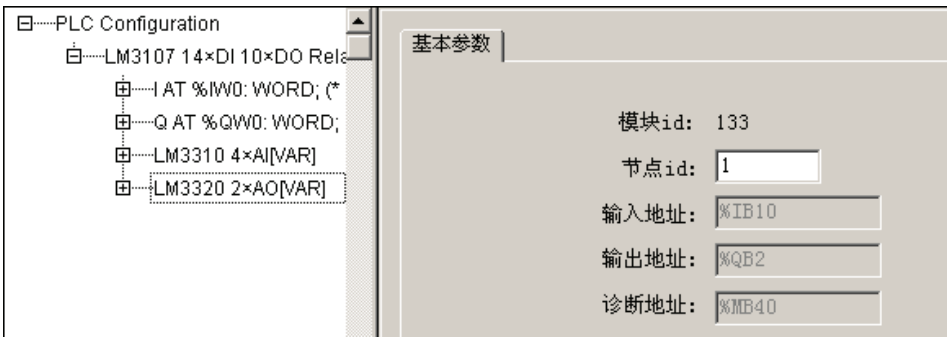


图 B-6-1

✧ 视图程序

PID 控制器参数的优化是 PID 控制的关键，由于每个系统固有属性不同，系统 PID 控制器参数都要试验确定，为了便于确定具体的微分时间、积分时间和比例系数参数，建议采用视图功能调节。本例中利用软件的视图功能（关于视图功能使用帮助，请参看软件手册）创建了一个视图界面，见图 B-6-2，其中上面为 PID 控制器测量值曲线图，下面是 PID 控制器输出曲线图，右边有微分时间、积分时间、比例系数调节界面，通过改变参数值可以优化参数。

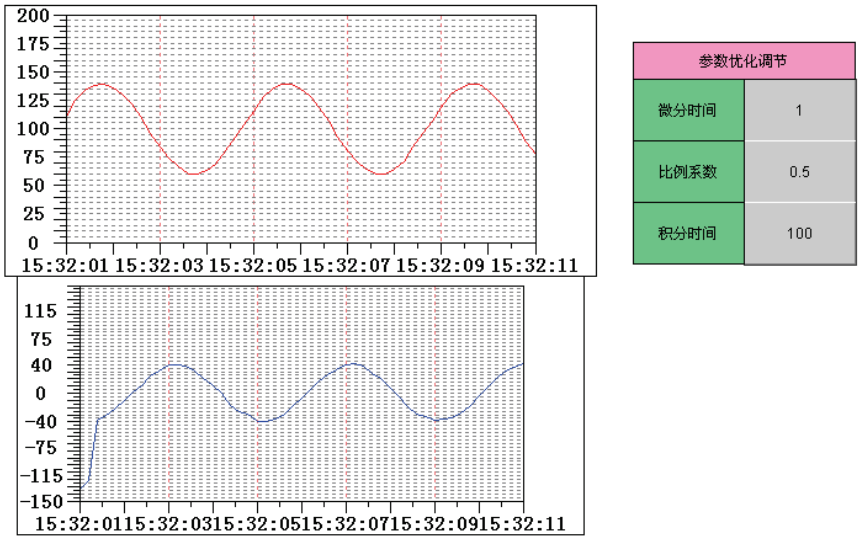


图 B-6-2

✧ 梯形图程序

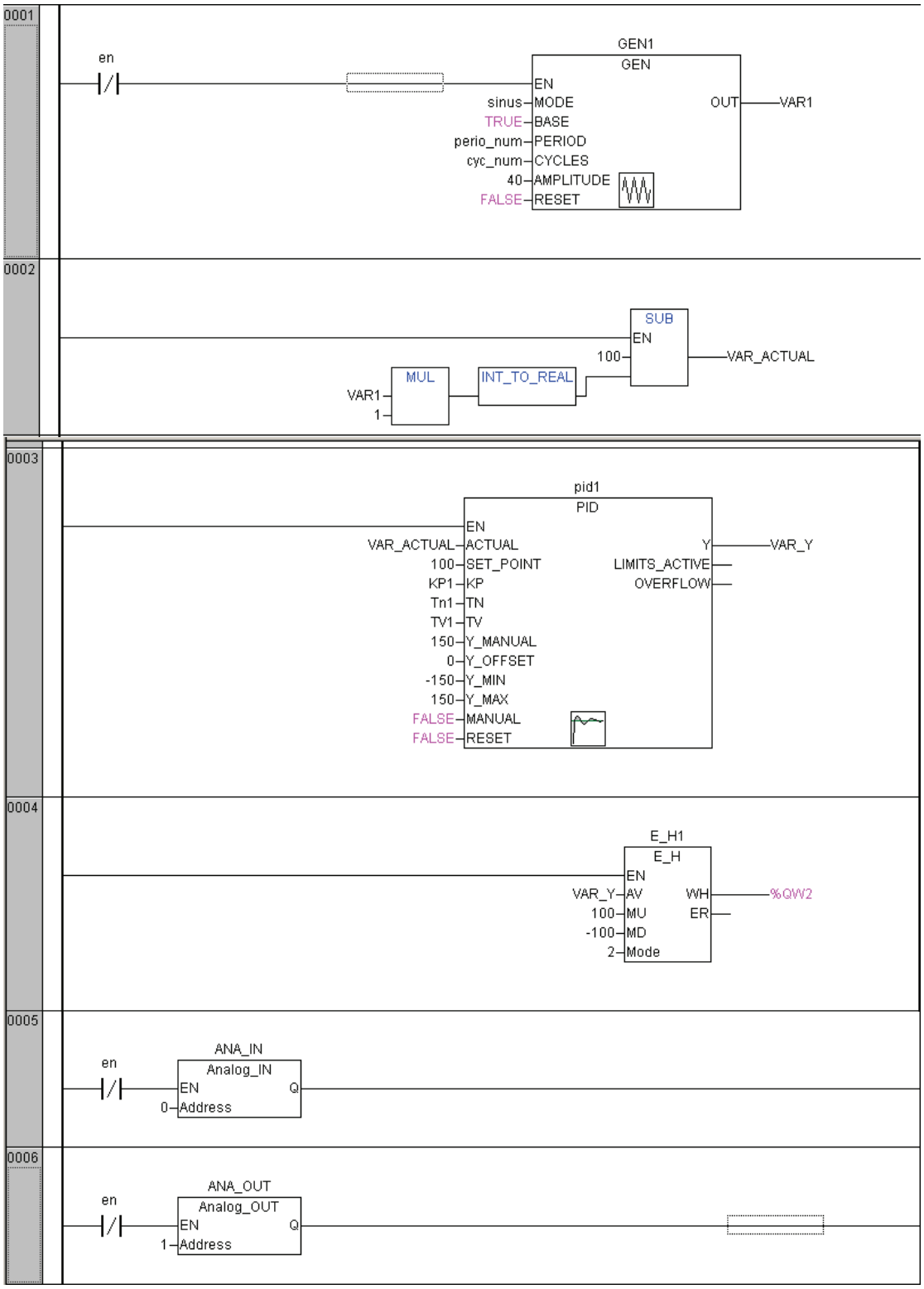


图 B-6-2

✧ 描述

GEN1 产生正弦信号赋给变量 VAR1，相当于现实模拟量输入量，VAR1 经过处理后转化为 PID 控制器的测量值，PID 控制器输出值经过工程量转换赋给模拟量输出通道。

B.7 高速计数器使用举例

✧ **要求**

本例叙述 G3 PLC 高速计数器（HD_CTUD_T2）的一种使用功能，高速计数器对来自 PLC 输出的高速脉冲信号进行处理，脉冲输出指令（PTO_PWM0）提供高速脉冲输出信号。

✧ **接线例图**

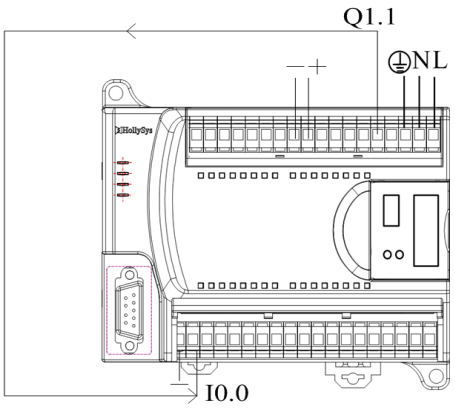


图 B-7-1 接线图（模块为 LM3106A，脉冲从 Q1.1 端口输出到端口 I0.0）

✧ **变量定义**

```
PROGRAM PLC_PRG
VAR
    EN: BOOL;
    PTO_INS: PTO_PWM0;          脉冲发送指令
    PT_OVER: BOOL;
    CTUD_INS: HD_CTUD_T2;       高速计数器
    P_NUM: DWORD;               已发送的脉冲数
    CV_NUM: UINT;               已计数的个数
    CTUD_OVER: BOOL;
END_VAR
```

✧ **PLC 配置**

本例中配置 CPU 模块为 LM3106，试验时可以采用 LM3106A 模块和 LM3106 模块，这两个模块点数都是一样的，不过 LM3106A 模块是专为高速运动控制而设计的模块，高速输出可以达到 100KHz，主要用来控制步进电机或伺服电机实现定位控制。

✧ **梯形图程序**

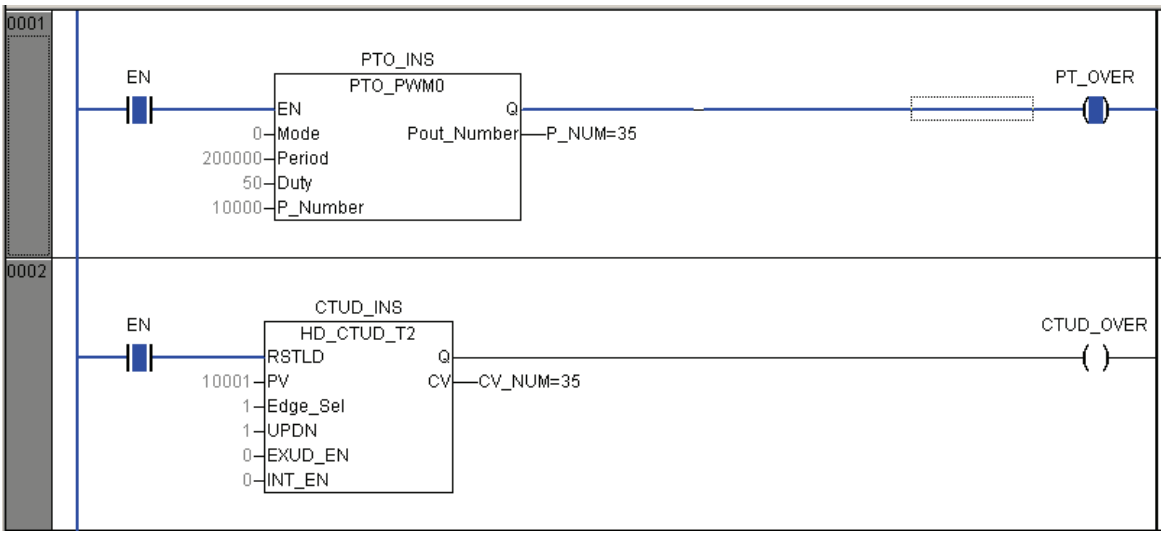


图 B-7-2

✧ **描述**

当 EN 被使能后，脉冲发送指令以 PTO 的方式通过端口 Q1.1 发送脉冲，脉冲的数量是 10000 个，周期为 200ms，P_NUM 显示已经发送的脉冲数，高速计数器通过端口 I0.0 接受脉冲信号，计脉冲信号的个数，每当脉冲的上升沿到达时就计一个数，CV_NUM 显示当前计数的值。

B.8 模拟电位器使用举例

✧ **要求**

HOLLiAS-LEC G3 PLC 的 CPU 模块都有两个模拟电位器，本例中用一个电位器模拟温度信号，用另一个电位器设置定时器的延时时间，当温度大于设定值，并超过延时时间后，模块的一个输出通道点亮，表示超温报警。

✧ **模拟电位器位置图**

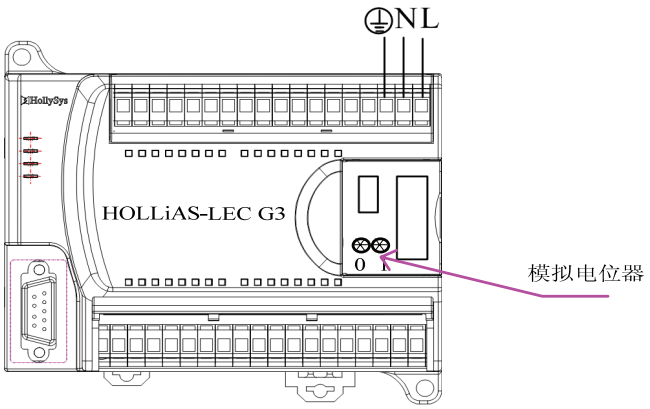


图 B-8-1 接线图（模块为 LM3106）

✧ **变量定义**

```
PROGRAM PLC_PRG
VAR
    EN: BOOL;
    POT1: POT;                  模拟电位器实例，读通道 1 的值
    TEMPERATURE: BYTE;          温度变量，通道 0 的值
    POT0: POT;                  模拟电位器实例，读通道 0 的值
    POT0_TERM: BOOL;
    TIME_DELAY: BYTE;           延时时间变量，通道 1 的值
    POT1_TERM: BOOL;
    M1: BOOL;                   中间变量，温度大约设定值置 1
    TON1: TON;                  定时器
    TIM_ET: TIME;               定时器已走时间
END_VAR
```

✧ **PLC 配置**

本试验采用任意一款 G3 PLC 的 CPU 模块都可以，在本例中配置 CPU 模块 LM3106。

✧ **梯形图程序**

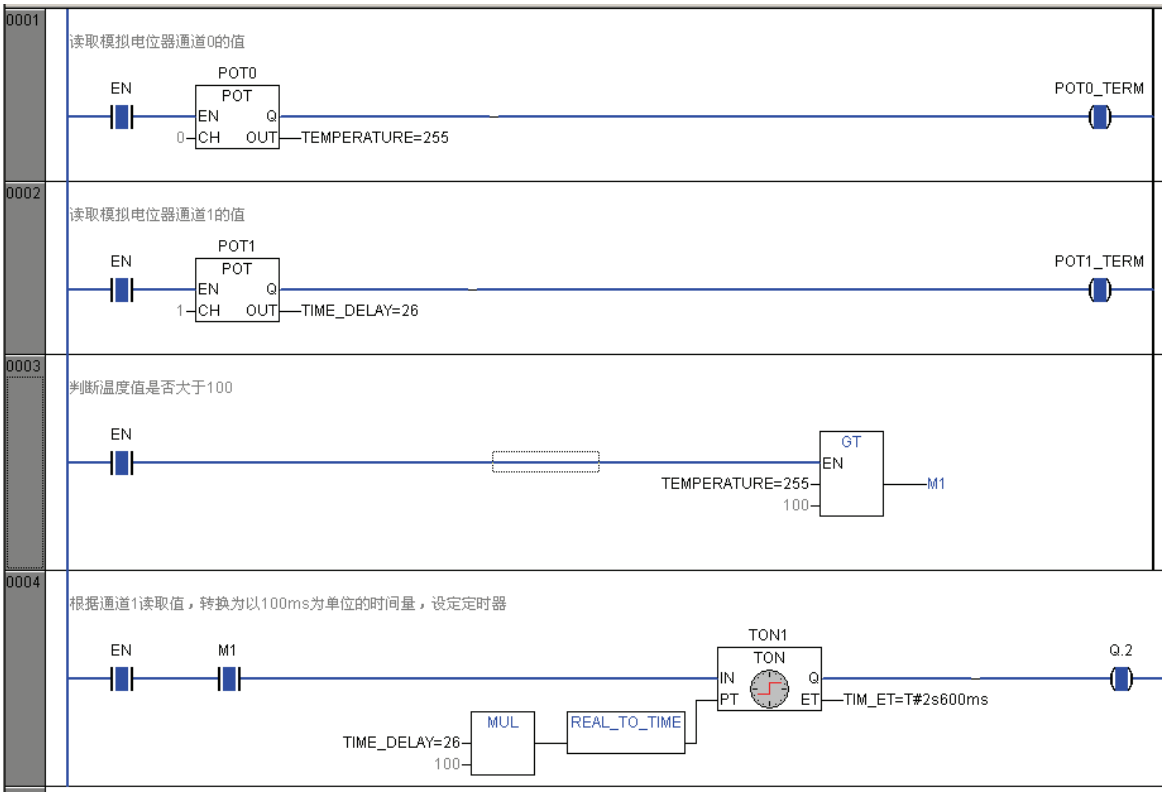


图 B-8-2

✧ **描述**

当 EN 被使能后，模拟电位器读取通道 0 的值赋给变量 TEMPERATURE，读取通道 1 的值赋给变量 TIME_DELAY，由于 TEMPERATURE 的值大于 100，M1 被置位，所以定时器开始计时，定时器的设定值必须是时间型的变量，中间加了一个类型转换指令，延时时间到后模块输出通道 Q0.2 接通。