前 言

适用范围

本手册所涉及的指令适用于以下版本编程软件。

PS501 V2.2.0 及以上中文版

PS501 V2.2.0 及以上英文版

面向的读者

本手册面向对 AC500 全系列 PLC 进行配置、编程和调试的技术人员。读者需要具备一定的自动化和 PLC 知识。

导读

如果已经熟练掌握 PS501 编程软件，可直接通过目录查找所需的指令。

如果刚刚开始学习 PS501 编程软件，建议阅读第 1 章指令概述，掌握指令的分类依据以及指令的调用方法。

第 2、3、4 章讲述 CoDeSys 编程环境标准提供的指令，如触发器指令、计数器指令、定时器指令、字符串处理指令等。

第 5 章至第 10 章讲述 AC500 功能块库提供的与硬件功能相关的指令，如通讯指令、实时时钟指令、高速计数指令等。

附录 A 介绍功能块故障代码的编码规则及一般故障信息。

附录 B 和 C 介绍 AC500 PLC 所支持的操作数与数据类型。

附录 D 简介 POU 的类型及特点。

相关手册

《PS501 Control Builder Plus_V2.2.0 软件手册》

产品和服务咨询

关于产品的任何咨询请寄往当地的 ABB 代表处，或登陆网站 www.abb.com/PLC 并选择销售或服务联系人。

网上资料库

登陆网站 www.abb.com/PLC 并进入下载专区获取您所需要的资料。

目录

第1章　　指令系统简述

可编程控制系统中令 CPU 完成某种操作或实现某种功能的命令称为指令，指令的集合称为指令系统。AC500 系列 PLC 为用户提供丰富的指令。

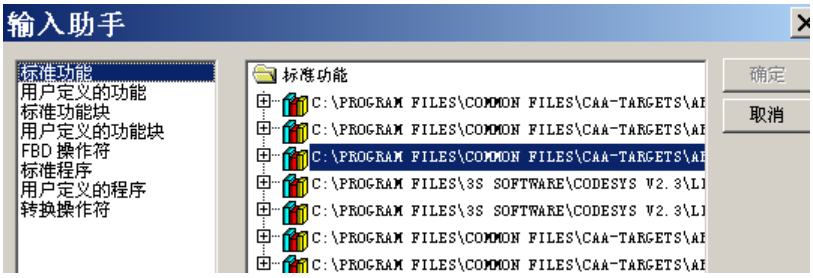
1.1 指令分类

1.1.1　POU 实现方式分类

AC500 指令按照 POU(程序组织单元,具体含义请参见附录 D)实现方式的不同分为功能和功能块两类。以功能方式实现的指令以 FUN 标注，在使用的时候无需声明实例。以功能块方式实现的指令以 FB 标注，在使用的时候需要声明实例。

AC500 还将某些指令进行打包，整体完成某项任务，以程序的方式提供给用户（以 PRG 标注）。

按照上述分类方法，用户调用时将指令分为标准功能指令、标准功能块指令、FBD 操作符（基本运算指令）、标准程序以及转换操作符（数据类型转换指令）等。如下图所示。这里的“标准”是指 PLC 系统标准提供的，与之对应的称为“用户定义的”，即用户自行编写的功能、程序或功能块。



1.1.2　指令库分类

库是若干指令的集合，所有的库文件均以“库名.lib”的方式命名。将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。按照库中指令功能的不同将其分为 CoDeSys 标准系统库和 AC500 库。

1.1.3　CoDeSys 标准系统库

CoDeSys 标准系统库提供常用指令，如触发器指令、计数器指令、定时器指令、字符串处理指令等。标准系统库与固件版本无关，可在 AC500 各系列 PLC 中使用。

CoDeSys 标准系统库主要包括以下三个库：

- Standard.lib（标准库，如触发器指令、计数器指令、定时器指令）
- Util.lib（应用库，如字符串处理指令、BCD 码转换指令）
- SysLibxxx.lib（系统库，如事件调用）

1. Standard.lib 标准库

Standard.lib 标准库在工程建立时自动添加，无需用户手动添加，包含以下指令。

组：Bistable Function Blocks（双稳态指令）	
指令	**功能**
RS	复位优先双稳态器
SR	置位优先双稳态器
SEMA	软件信号灯
组：Counter（计数器）	
指令	**功能**
CTU	递增计数器
CTD	递减计数器
CTUD	递增递减计数器
组：Timer（定时器）	
指令	**功能**
TON	延时导通定时器
TOF	延时断开定时器
TP	脉冲定时器
RTC	实时时钟
组：Trigger（触发器）	
指令	**功能**
R_TRIG	上升沿触发器
F_TRIG	下降沿触发器
组：String Function（字符串指令）	
指令	**功能**
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	左边获取字符串指令
RIGHT	右边获取字符串指令
LEN	取字符串长度指令
MID	获取字符串指令
REPLACE	替换字符串指令

2. Util.lib 应用库

Util.lib 应用库在工程建立时自动添加，无需用户手动添加，包含以下指令。

组：analog monitors（模拟量处理指令）	
指令	**功能**
HYSTERESIS	滞后指令
LIMITALARM	上下限报警指令
组：BCD conversions（BCD 码转换指令）	
指令	**功能**
BCD_TO_INT	BCD 码转整型指令
INT_TO_BCD	整型转 BCD 码指令

组：controller（PID 控制器指令）	
指令	**功能**
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
组：mathematical functions（高等数学运算指令）	
指令	**功能**
CHARCURVE	特征曲线指令
RAMP_INT	整型限速指令
RAMP_REAL	实型限速指令
组：bit/byte functions（位/字节转换指令）	
指令	**功能**
EXTRACT	位提取指令
PACK	位整合指令
PUTBIT	位赋值指令
UNPACK	位拆分指令
组：signals（信号生成指令）	
指令	**功能**
BLINK	脉冲信号发生器
GEN	典型周期信号发生器
FREQ_MEASURE	频率测量指令
组：function manipulators（函数操纵器指令）	
指令	**功能**
DERIVATIVE	微分指令
INTEGRAL	积分指令
LIN_TRAFO	线性转换指令
STATISTICS_INT	整型统计指令
STATISTICS_REAL	实型统计指令
VARIANCE	平方偏差指令
组：Version_Util（库版本查看指令）	
指令	**功能**
Version_Util	库版本查看指令

3. SysLibxxx.lib 系统库

- SysLibCallback.lib

指令	**功能**
SysCallbackRegister	事件调用
SysCallbackUnregiste	解除事件调用

- SysLibTIME.lib

指令	**功能**
CurTime	提供本地系统的实时时间
CurTimeEx	提供本地系统的实时时间的扩展信息

- SysLibMem.lib

指令	功能
SysMemCpy	拷贝内存
SysMemCmp	比较两个内存里的内容
SysMemFree	释放内存空间
SysMemMove	移动一个内存中的内容到令一个内存

- SysLibEvent.lib

指令	功能
SysEventCreate	产生一个事件
SysEventDelete	删除一个事件
SysEventSet	设置一个事件
SysEventWait	设置一个事件的超时时间

- Iecsfc.lib

指令	功能
SFCActionControl	SFC 语言中 IEC 步关联动作的控制

1.1.4 C500 库

AC500 库提供与硬件功能相关的指令，如通讯指令、实时时钟指令、高速计数指令等。用户进行硬件配置后与之对应的功能库将自动添加，无需用户手动添加。

1.1.4.1 AC500 库的命名规则

AC500 库名称包括功能描述 Funcx（例如 Ethernet）、target 目标系统（例如 AC500、S90）以及所要求的最低固件版本 Vxx，如“Ethernet_AC500_V10.lib”。统一格式如下。

<Funca><Funcb><Funcc>_AC500_V<VersInfo>.lib

1.1.4.2 AC500 库主要文件

AC500 库主要包括以下库文件：

- SysInt_AC500_V10.lib（内部系统库）
- SysExt_AC500_V10.lib（外部系统库）
- SysIntExt_AC500_V13.lib（内部系统扩展库）
- OnBoardIO_AC500_V13.lib（板载 IO 功能库）
- MODBUS_AC500_V10.lib（Modbus 通讯库）
- ASCII_AC500_V10.lib（ASCII 码通讯库）
- Ethernet_AC500_V10.lib（Ethernet 通讯库）
- EtherCAT_AC500_V13.lib（EtherCAT 通讯库）
- PROFIBUS_AC500_V10.lib（PROFIBUS DP 通讯库）
- Profinet_AC500_V13.lib（PROFINET 通讯库）
- CANopen_AC500_V11.lib（CANopen 通讯库）
- DC541_AC500_V11.lib（DC541 功能库）
- Counter_AC500_V20.lib（计数器库）

- Diag_AC500_V20.lib（诊断库）
- CD522_AC500_V13.lib（CD522 功能库）
- RCOM_AC500_V13.lib（CM574 功能库）
- IEC6870_AC500_V20.lib（ IEC6870 通讯库）
- SysLibABBCfg.lib（配置库）

1. SysInt_AC500_V10.lib（内部系统库）

SysInt_AC500_V10.lib 包含以下指令。

组：Data Storage（数据存储）	
子组：Configuration data（配置数据）	
指令	**功能**
CPU_CONFIG_READ	从 FLASH 中读取配置数据
CPU_CONFIG_WRITE	将配置数据写入 FLASH
子组：FLASH	
指令	**功能**
FLASH_DEL	从 FLASH 删除一个数据段
FLASH_READ	从 FLASH 读取一个数据组
FLASH_WRITE	向 FLASH 写入一个数据段
子组：Persistent data（Persistent 数据）	
指令	**功能**
PERSISTENT_CLEAR	删除 SRAM 中的持久型数据
PERSISTENT_EXPORT	将 RAM-DISC 中的持久型数据写入 SD 卡
PERSISTENT_IMPORT	将 SD 卡中的持久型数据写入 RAM-DISC
PERSISTENT_RESTORE	将 RAM-DISC 中的持久型数据写入 SRAM
PERSISTENT_SAVE	将 SRAM 中的持久型数据写入 RAM-DISC
子组：Production data（产品数据）	
指令	**功能**
CPU_PROD_ENTRY_READ	读取 FLASH 中的产品数据
子组：Retain data（Retain 数据）	
指令	**功能**
RETAIN_CLEAR	删除 SRAM 中的保持数据
RETAIN_EXPORT	将 RAM-DISC 中的保持数据写入 SD 卡
RETAIN_IMPORT	将 SD 卡中的保持数据写入 RAM-DISC
RETAIN_RESTORE	将 RAM-DISC 中的保持数据写入 SRAM
RETAIN_SAVE	将 SRAM 中的保持数据写入 RAM-DISC
子组：SD Card （SD 卡）	
指令	**功能**
SD_READ	从 SD 卡读取数据
SD_WRITE	向 SD 卡写入数据

组：Diagnosis（诊断）	
指令	**功能**
DIAG_ACK	确认故障信息
DIAG_ACK_ALL	确认同一等级的全部故障信息
DIAG_EVENT	生成故障事件
DIAG_GET	读取故障信息
DIAG_INFO	显示未读取信息的故障等级
组：DPRAM communication （DPRAM 通讯）	
指令	**功能**
DPRAM_CM5XX_REC	从 CM5xx 接收数据
DPRAM_CM5XX_SEND	将数据发送到 CM5xx
组：I/O Bus （I/O 总线）	
指令	**功能**
IO_DIAG	读取 I/O 总线上的诊断数据
IO_INFO	读取 I/O 总线上连接的模块数量
IO_MODULE_DIAG	读取 I/O 总线上的模块诊断数据
IO_VERSION	读取 I/O 总线驱动的版本号
组：Serial interface（串行接口）	
指令	**功能**
COM_SET_PROT	更改串行接口的协议
组：System information（系统命令）	
指令	**功能**
PLC_REBOOT	重新启动 PLC
组：System information（系统信息）	
指令	**功能**
CPU_INFO	读取 CPU 型号
FPU_EXCEPTION_INFO	读取 FPU 异常信息
RTS_INFO	读取 CPU 运行系统的版本号
SLOT_INFO	读取插槽信息
SYS_TIME	读取系统时间

2. SysExt_AC500_V10.lib（外部系统库）

SysExt_AC500_V10.lib 包含以下指令。

组：Battery（电池）	
指令	**功能**
BATT	读取电池状态
组：Data access（数据存取）	
CPUCheckUserAccess	预留给内部处理
CPUDevltfCmd	预留给内部处理
组：Realtime clock（实时时钟）	
指令	**功能**
CLOCK	设置/读取时钟和日期
CLOCK_DT	使用“DT”格式设置/读取时钟和日期

3. SysIntExt_AC500_V13.lib（内部系统扩展库）

SysIntExt_AC500_V13.lib 包含以下指令。

组：DPRAM communication（DPRAM 通讯）	
指令	**功能**
DPRAM_IO_COPY	复制 IO 数据区域
DPRAM_PM5XX_REC	从主机 CPU 接收数据
DPRAM_PM5XX_SEND	发送数据到主机 CPU
组：LED	
指令	**功能**
LED_SET	设置 CM574 LED 灯的模式
组：System information（系统信息）	
指令	**功能**
CPU_OWN_ADR	检测 CM574 旋转开关的地址设置

4. OnBoardIO_AC500_V13.lib（板载 IO 功能库）

OnBoardIO_AC500_V13.lib 包含以下指令。

组：Counter_OBIO（板载 IO 计数器）	
指令	**功能**
ONB_IO_CNT	板载 I/O 的高速计数指令
组：PWM_OBIO（板载 IO PWM 输出）	
指令	**功能**
ONB_IO_PWM_FREQ	板载 I/O 的 PWM 输出（频率方式）
ONB_IO_PWM_TIME	板载 I/O 的 PWM 输出（周期方式）
组：Interrupt_Input_BOIO（板载 IO 中断输入）	
指令	**功能**
CPU_IO_INT_IN	读取板载 I/O 的中断输入状态

5. MODBUS_AC500_V10.lib（Modbus 通讯库）

MODBUS_AC500_V10.lib 包含以下指令。

组：MODBUS	
指令	**功能**
COM_MOD_MAST	Modbus RTU 主站指令

6. ASCII_AC500_V10.lib（ASCII 码通讯库）

ASCII_AC500_V10.lib 包含以下指令。

组：ASCII	
指令	**功能**
COM_REC	串口自由协议接收数据
COM_SEND	串口自由协议发送数据

7. Ethernet_AC500_V10.lib（Ethernet 通讯库）

Ethernet_AC500_V10.lib 包含以下指令。

组：Diagnosis（常规）	
指令	**功能**
ETH_OWN_IP_INFO	输出 IP 设置信息
ETH_OWN_IP	输出 IP 地址

组：DPRAM communication （DPRAM 通讯）	
子组：含 AC31 报头	
指令	功能
ETH_UDP_INFO	读取 UDP/IP 处理的状态信息
ETH_UDP_REC	通过 UDP/IP 接收数据
ETH_UDP_SEND	通过 UDP/IP 发送数据
ETH_UDP_STO	读取 UDP/IP 超时数据包
子组：无 AC31 报头	
指令	功能
ETH_UDP_STD_INFO	读取标准 UDP/IP 处理的状态信息
ETH_UDP_STD_REC	通过标准 UDP/IP 接收数据
ETH_UDP_STD_SEND	通过标准 UDP/IP 发送数据
组：MODBUS_TCP （基于 TCP/IP 的 OpenModbus）	
指令	功能
ETH_MOD_MAST	Modbus TCP/IP 主站指令
组：IP conversions（IP 地址的格式转换）	
指令	功能
IP_ADR_DWORD_TO_STRING	IP 地址由 DWORD 格式转换为 STRING 格式
IP_ADR_STRING_TO_DWORD	IP 地址由 STRING 格式转换为 DWORD 格式

8. EtherCAT_AC500_V13.lib（EtherCAT®通讯库）

EtherCAT_AC500_V13.lib 包含以下指令。

组：CoE	
指令	功能
ECAT_COE_READ	读取从站的一个 CoE 对象
ECAT_COE_WRITE	向从站写入一个 CoE 对象
组：DIAG （诊断）	
指令	功能
ECAT_BUS_DIAG	ECAT 总线诊断信息
ECAT_SLV_DIAG	ECAT 从站诊断信息
ECAT_GET_DCLK_DEVI	读取分布式时钟的偏差
组：INFO （信息）	
指令	功能
ECAT_STATE	读取通讯模块的状态信息

9. PROFIBUS_AC500_V10.lib（PROFIBUS DP 通讯库）

PROFIBUS_AC500_V10.lib 包含以下指令。

组：Control（控制）	
指令	功能
DPM_CTRL	向从站发送全局控制命令
组：Data （数据）	
指令	功能
DPM_READ_INPUT	读取其他主站的从站的输入数据
DPM_READ_OUTPUT	读取其他主站的从站的输出数据

子组：DPV1	
指令	**功能**
DPV1_MSAC1_READ	读取 DPV1 从站的输入数据
DPV1_MSAC1_WRITE	向 DPV1 从站写入数据
组：Diagnosis （诊断）	
指令	**功能**
DPM_SLV_DIAG	读取从站诊断信息
DPM_STATE	读取通讯模块的状态信息
DPM_SYS_DIAG	读取所有从站的状态信息

10. Profinet_AC500_V13.lib（PROFINET 通讯库）

Profinet_AC500_V13.lib 包含以下指令。

组：Alarm（报警）	
指令	**功能**
PNIO_DEV_ALARM	读取 PROFINET IO 设备报警信息
组：Diagnosis （诊断）	
指令	**功能**
PNIO_DEV_DIAG	读取 PROFINET IO 设备诊断信息
组：INFO （信息）	
指令	**功能**
PNIO_DEV_INFO	读取 PROFINET IO 设备一般信息
PNIO_IM0	PROFINET IO 设备的标识和维护信息
PNIO_STATE	读取总线的一般信息
PNIO_SYS_DIAG	读取通讯模块的状态信息

11. CANopen_AC500_V11.lib（CANopen 通讯库）

CANopen_AC500_V11.lib 包含以下指令。

组：CAN 2.0A	
指令	**功能**
CAN2A_INFO	读取 CAN 2.0A 通讯信息
CAN2A_REC	接收 CAN 2.0A 报文
CAN2A_SEND	发送 CAN 2.0A 报文
组：CAN 2.0B	
指令	**功能**
CAN2B_INFO	读取 CAN 2.0B 通讯信息
CAN2B_REC	接收 CAN 2.0B 报文
CAN2B_SEND	发送 CAN 2.0B 报文
组：Control（控制）	
指令	**功能**
CANOM_NMT	控制 NMT 节点的状态
组：Parameter（参数）	
指令	**功能**
CANOM_SDO_READ	读取从站的 SDO
CANOM_SDO_WRITE	写从站的 SDO

组：Diagnosis （诊断）	
指令	**功能**
CANOM_NODE_DIAG	读取从站的诊断信息
CANOM_RES_ERR	复位通讯模块的错误
CANOM_STATE	读取通讯模块的诊断信息
CANOM_SYS_DIAG	显示所有从站的状态

12. DC541_AC500_V11.lib（DC541 功能库）

DC541_AC500_V11.lib 包含以下指令。

组：Counter（计数器）	
指令	**功能**
DC541_32BIT_CNT	DC541 32 位计数器指令
DC541_LIMIT	DC541 32 位计数器极限监视指令
DC541_FWD_CNT	DC541 32 位加计数器指令
DC541_FREQ_OUT	DC541 仅频率可调的 PWM 输出
组：Diagnosis （诊断）	
指令	**功能**
DC541_GET_CFG	读取 DC541 的配置信息
DC541_STATE	获取 DC541 的状态
组：IO	
指令	**功能**
DC541_INT_IN	读取中断源
DC541_IO	读/写 DC541 的输入和输出

13. CD522_AC500_V13.lib（CD522 功能库）

CD522_AC500_V13.lib 包含以下指令。

指令	功能
CD522_16BIT_2CNT	CD522 双 16 位计数器指令
CD522_16BIT_CNT	CD522 16 位计数器指令
CD522_32BIT_CNT	CD522 32 位计数器指令
CD522_32BIT_ENCODER	CD522 增量式编码器指令
CD522_SSI_CNT	CD522 绝对式编码器指令
CD522_FREQ_SCAN	CD522 时间频率计指令
CD522_PWM_OUT	CD522 频率及占空比可调的 PWM 输出
CD522_PULSE_OUT	CD522 脉冲输出
CD522_FREQ_OUT	CD522 仅频率可调的 PWM 输出

14. Counter_AC500_V20.lib（计数器库）

Counter_AC500_V20.lib 包含以下指令。

指令	功能
CNT_DC551	CS31 总线接口模块的高速计数指令
CNT_IO	S500 I/O 设备的高速计数指令

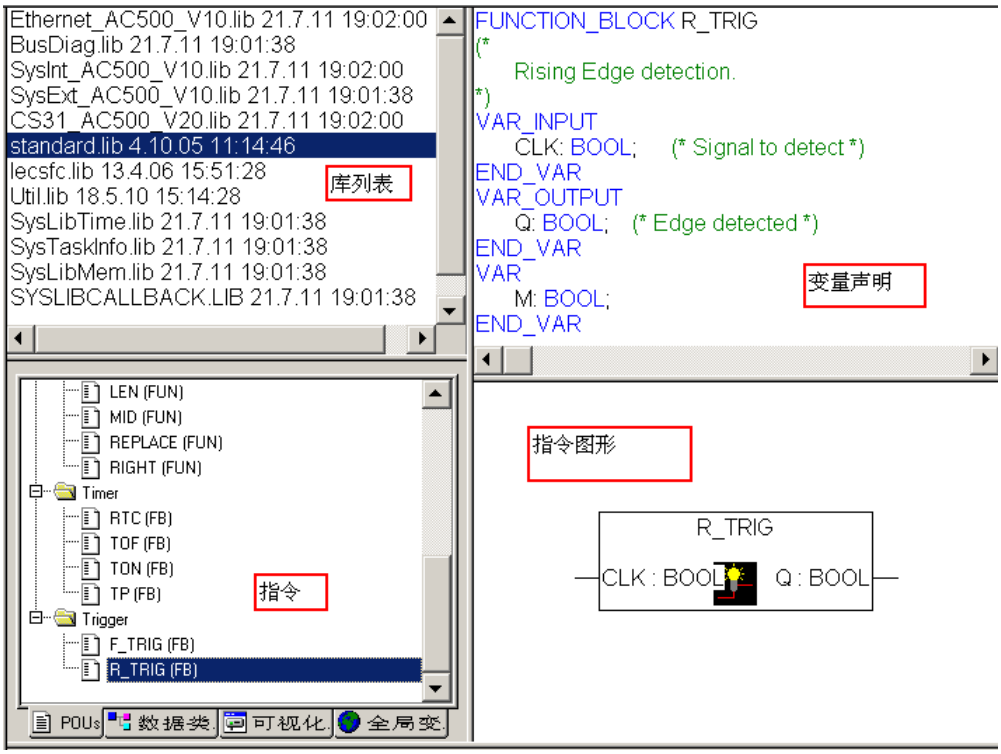
15. Diag_AC500_V20.lib（诊断库）

Diag_AC500_V20.lib 包含以下指令，各指令读取的诊断信息需要配合可视化界面进行显示。

组：CPU Diagnosis （CPU 诊断）	
指令	**功能**
CPU_DIAG	读取 CPU 诊断信息
CPU_LOAD	读取 CPU 负荷
组：CS31_BUS_Diagnosis （CS31 诊断）	
指令	**功能**
CS31_DIAG	CS31 总线诊断
CS31_DIAG_EXT	CS31 总线扩展诊断
组：FBP_Diagnosis（FBP 诊断）	
指令	**功能**
FBP_DIAG	诊断 FBP 从站的中断源

1.2 库文件管理器

指令库是指令的集合，调用某个指令，必需载入相应的库文件。CoDeSys 中通过库文件管理器来管理库文件，库文件管理器窗口的结构如下图所示。



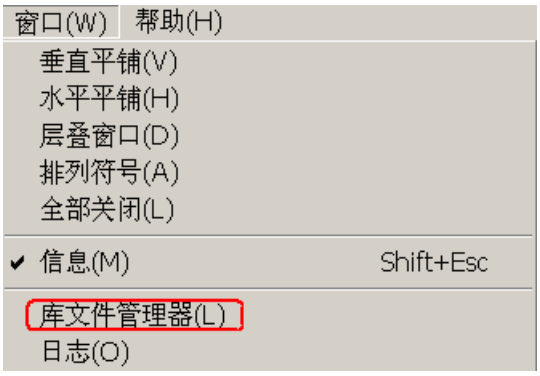
库文件管理器窗口分为“库列表”、“指令”、“变量声明”和“指令图形”这四部分。

- 库列表：列出当前工程中已添加的所有库文件。
- 指令：列出所选库文件中包含的指令并按类分别列出。
- 变量声明：列出所选指令的变量声明。
- 指令图形：显示所选指令的图形，左边为输入端，右边为输出端。

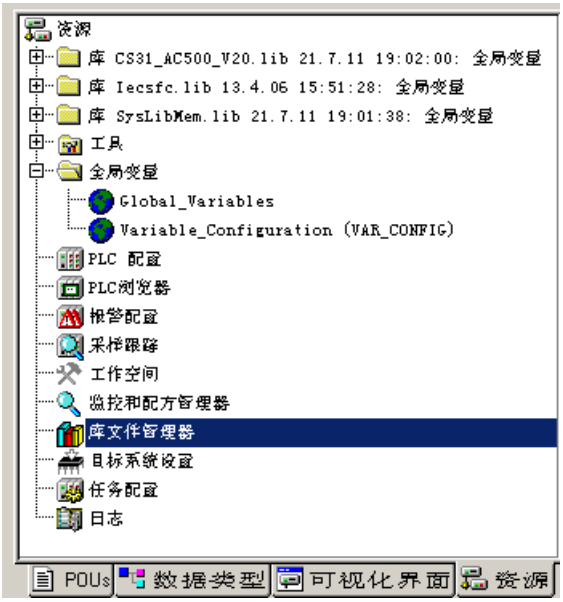
通过库文件管理器可实现以下功能：

1. 打开库管理器

- 点击编程界面菜单拦的“窗口（W）”，鼠标左键点击“库文件管理器（L）”，如下图。



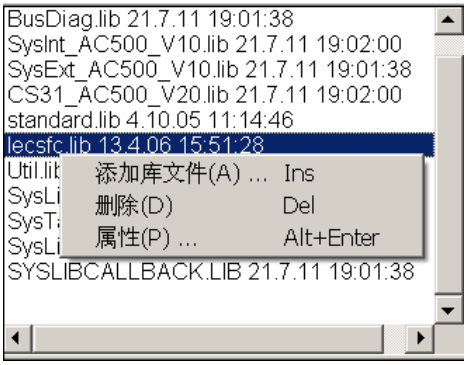
- 点击对象管理器中“资源”，打开资源窗口，再双击“库文件管理器”，如下图。



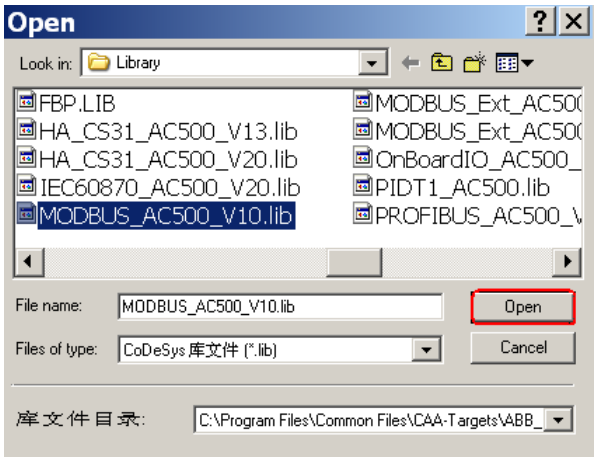
2. 指令库的添加

当库列表区的库文件已不能满足目前的编程需要时，则需要手动添加所需的库文件。

- 在库文件管理器窗口的库列表位置选择鼠标右键菜单命令“添加库文件”，如下图。



- 选择所需要的库文件，点击“Open”，打开对应的*.lib 文件。



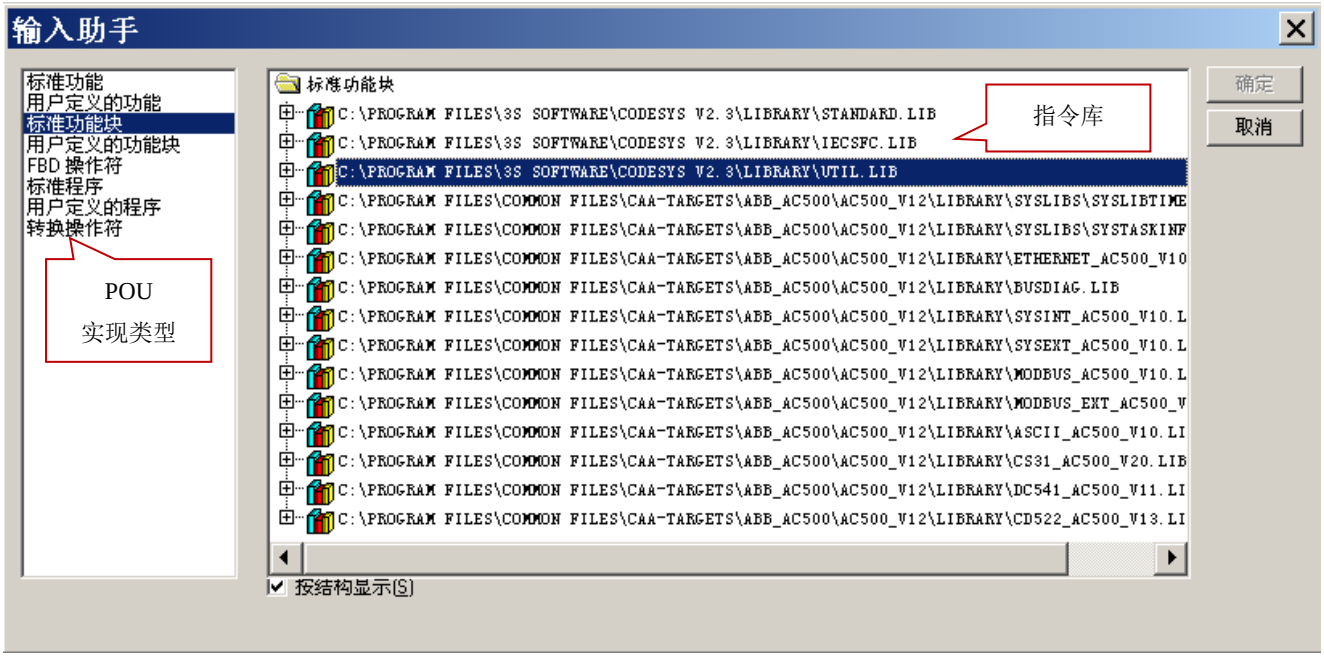
凡是添加到库文件管理器的库都会占用用户程序空间，所以建议用户只添加必须使用的库。

3. 删除库

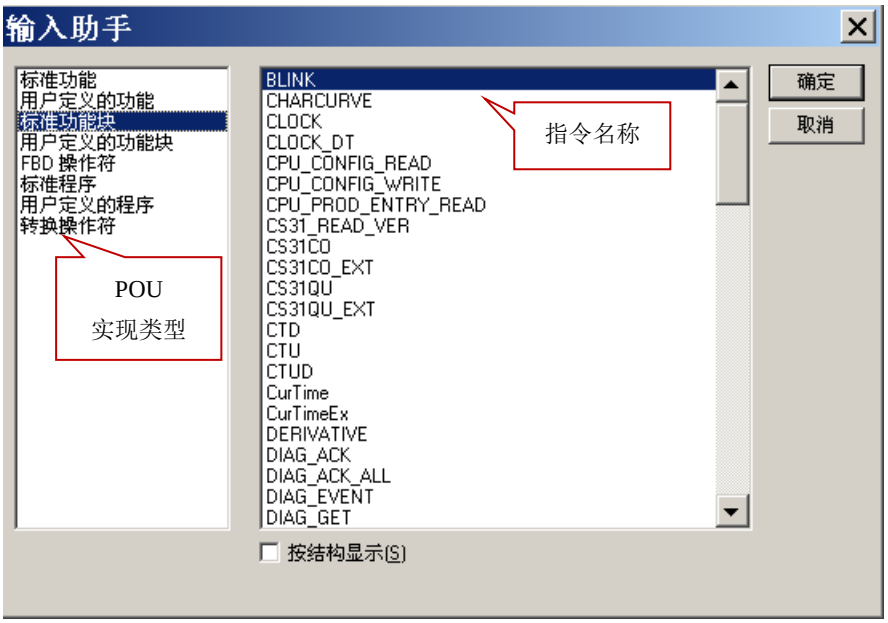
选中库文件，选择鼠标右键菜单命令“删除”，即可从工程和库文件管理器中删除已添加的库。

1.3 指令的调用

用户编写程序时，无论选用何种编程语言都可以借助输入助手（功能键<F2>）从中选择调用所需的指令，如下图。系统默认按照 POU 实现类型和指令库方式提供指令。



如果对指令所属的库文件名称不了解，可以去除“按结构显示”前的“√”，切换至指令名称显示形式，见下图。



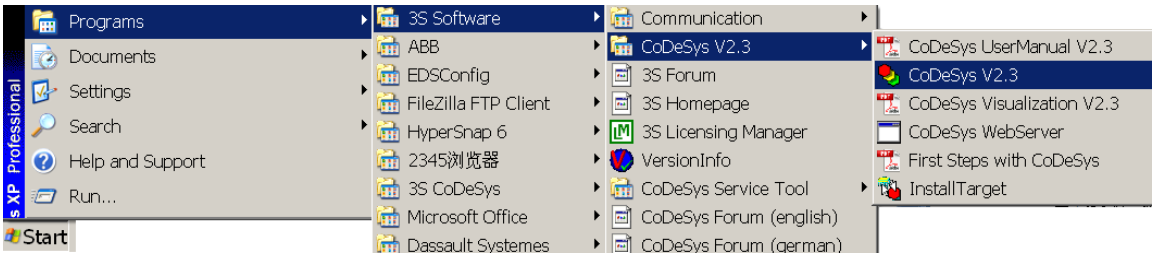
1.4 自定义库的制作

用户在工程实施和编写程序的过程中，如果有一些算法需要重复使用，而且可应用于多个工程，那么可将这部分算法制作成库，以便于程序调用和资源共享。

用户制作库或自定义库的步骤如下：

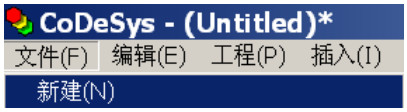
1. 直接打开 CoDeSys 编程平台

鼠标左键单击“开始/所有程序/3S Software/CoDeSys V2.3/CoDeSys V2.3”启动 CoDeSys V2.3 平台，如下图。启动后，便进入编程主界面。



2. 创建文件

鼠标左键点击编程界面菜单拦的“文件（F）”，再鼠标左键点击“新建（N）”

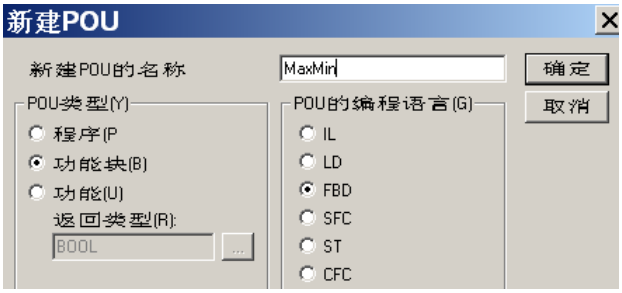


3. 选择目标系统

弹出“目标系统设置”窗口，其中“目标系统”项内选“None”，鼠标左键点击“确定”，弹出“创建POU”窗口。

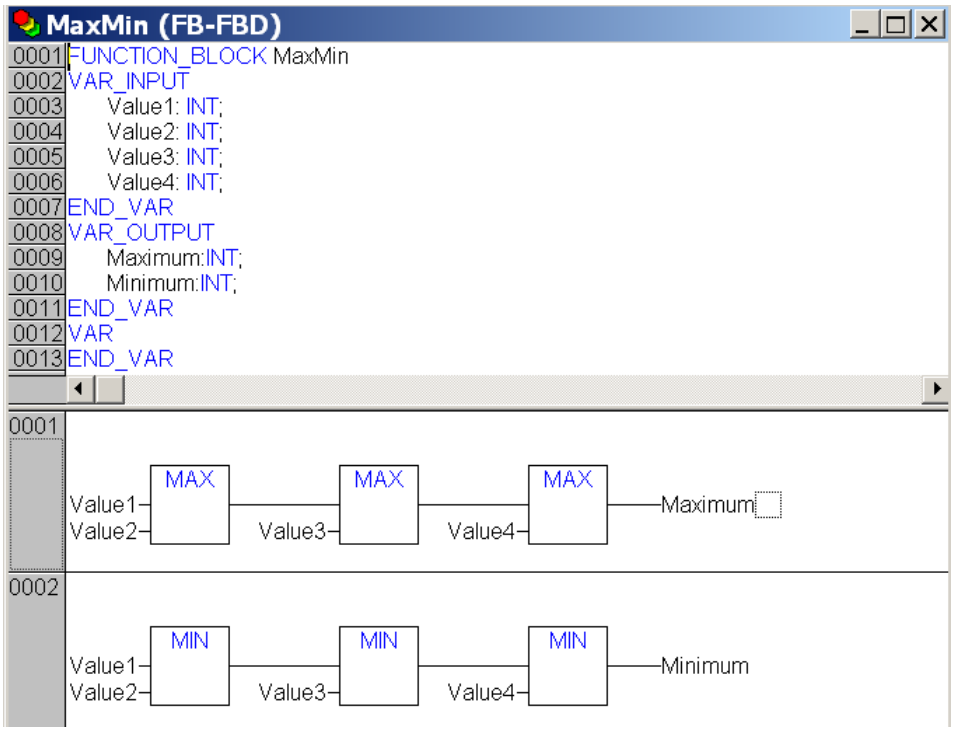
4. 新建 POU

在“新建 POU”窗口中，选则 POU 类型、POU 语言，填写新 POU 名称，例如本例中创建功能块“MaxMin”，实现计算四个输入整数的最大值及最小值。最后点击“确定”。



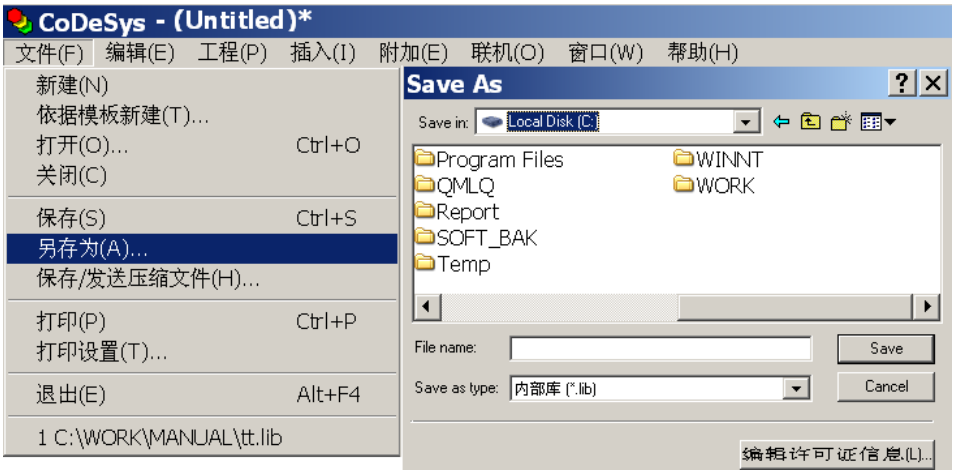
5. 完成指令算法

本例中的算法如下图。



6. 保存为内部库

保存时选择目录，文件名根据需要定义，保存类型需选择内部库（Internal Library），如下图。



7. 使用自定义库

编译通过后，就可以在其它的工程中使用此库。方法同使用 CoDeSys 标准库。使用前注意在库文件管理器中加入此库。一个库中可以包括多个功能块或功能。

第2章　　FBD 运算和数据类型转换指令

CoDeSys 编程环境标准提供 FBD 运算指令和数据类型转换指令，与库无关，用户可直接调用。

FBD 运算指令也称为 FBD 操作符。这些指令可对混合类型数据进行计算，但用户应确保选择的输出数据类型可以存储输出结果，否则可能引起数据错误，而且系统不会因此给出警告信息。

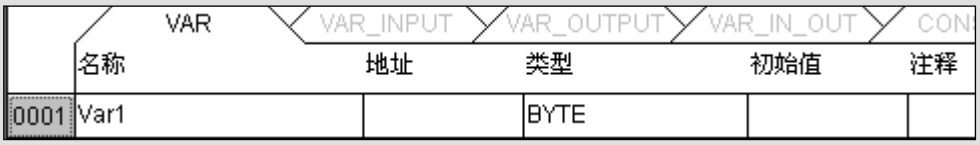
LD 编程语言环境中，FBD 运算指令、转换指令只能通过“带 EN 的框”调用。

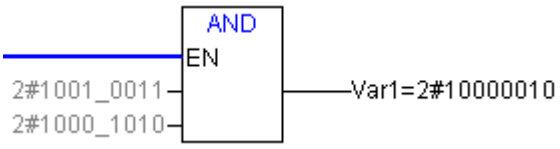
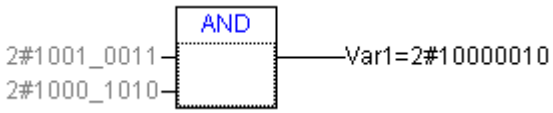
2.1 逻辑运算指令

2.1.1　AND（与指令）

1. 功能：变量或常量的相与运算。
2. 输入/输出变量类型：BOOL、BYTE、WORD、DWORD 和 LWORD。
3. 指令使用举例

变量定义

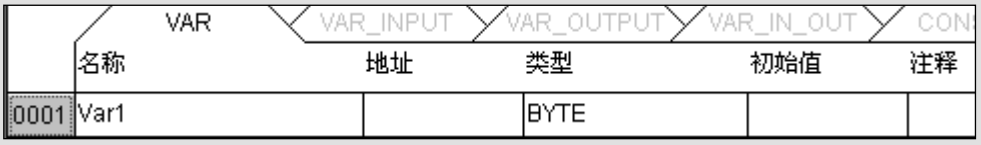


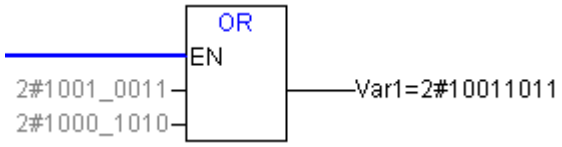
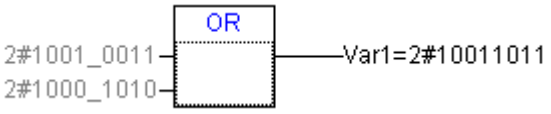
编程语言	程　序
梯形图（LD）	
结构化文本（ST）	Var1:=2#1001_0011 AND 2#1000_1010; （*结果 Var1 为 2#10000010*）
功能块（FBD）	

2.1.2　OR（或指令）

1. 功能：变量或常量的相或运算。
2. 输入/输出变量类型：BOOL、BYTE、WORD、DWORD 和 LWORD。
3. 指令使用举例

变量定义

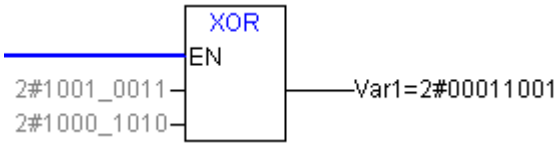
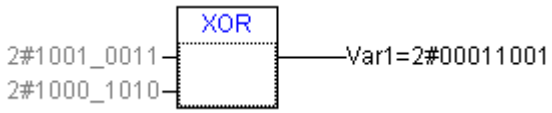


编程语言	程　序
梯形图（LD）	
结构化文本（ST）	Var1:=2#1001_0011 OR 2#1000_1010 ; （*结果 Var1 为 2#10011011*）
功能块（FBD）	

2.1.3　XOR（异或指令）

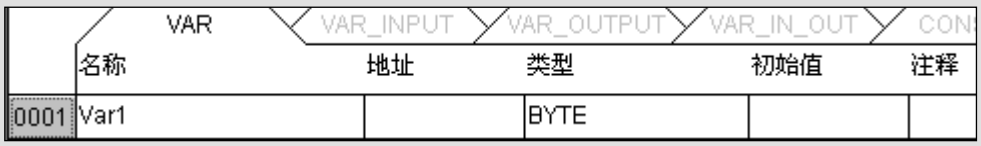
1. 功能：变量或常量的异或运算。
2. 输入/输出变量类型：BOOL、BYTE、WORD、DWORD 和 LWORD。
3. 指令使用举例

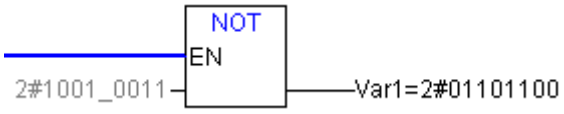
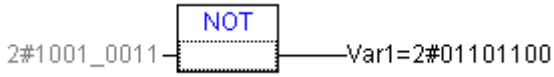
变量定义


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#0001_1001*）
功能块（FBD）	

2.1.4　NOT（取反指令）

1. 功能：变量或常量逐位取反。
2. 输入/输出变量类型：BOOL、BYTE、WORD、DWORD 和 LWORD。
3. 指令使用举例

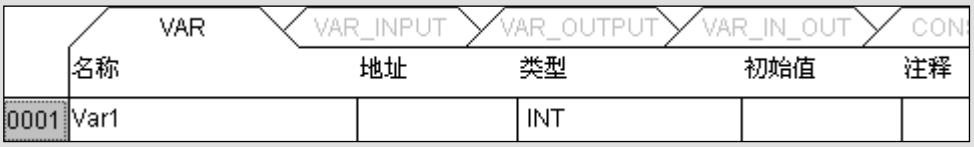
变量定义


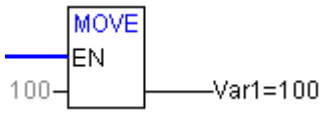
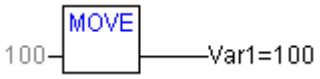
编程语言	程　序
梯形图（LD）	
结构化文本（ST）	Var1:=NOT 2#1001_0011; （*结果 Var1 为 2#0110_1100*）
功能块（FBD）	

2.2 赋值指令

1. MOVE：赋值指令
2. 功能：将一个常量或者变量的值赋给另外一个变量。
3. 输入/输出变量类型：任何类型。
4. 指令使用举例

变量定义



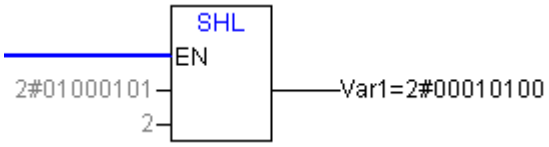
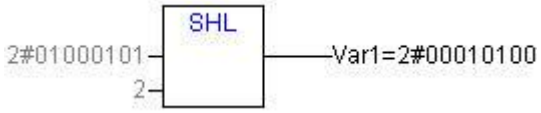
编程语言	程　序
梯形图（LD）	
结构化文本（ST）	Var1:=100;　　（*结果 Var1 为 100*）
功能块（FBD）	

2.3 移位和循环移位指令

2.3.1　SHL（左移指令）

1. 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。
2. 输入/输出变量类型：BYTE、INT、LINT、WORD、DWORD、LWORD、SINT、UINT、ULINT。
3. 指令使用举例

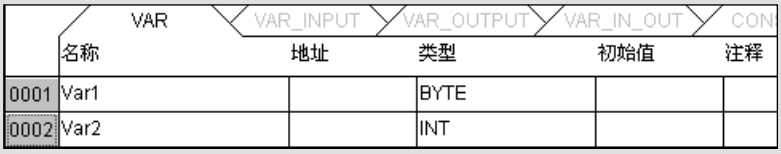
变量定义

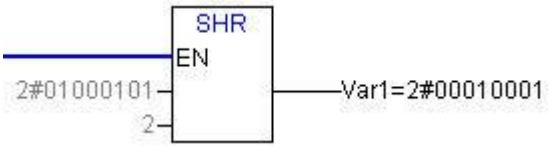
编程语言	程　序
梯形图（LD）	
结构化文本（ST）	Var1_BYTE:=SHL(16#45,2);　（*结果 Var1_BYTE 为 16#14*） Var2_WORD:=SHL(16#45,2);　（*结果 Var2_WORD 为 16#0114*）
功能块（FBD）	

2.3.2　SHR（右移指令）

1. 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。
2. 输入/输出变量类型：BYTE、INT、LINT、WORD、DWORD、LWORD、SINT、UINT、ULINT。
3. 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:=SHR(16#45,2);　（*结果 Var1 为 16#11*） Var2:=SHR(16#45,2);　（*结果 Var2 为 16#0011*）
功能块（FBD）	

对于移位的位数，若该变量是个常数则被视为最小数据类型。输入变量的数据类型对计算毫无影响，但输出变量的类型是不能忽视的。例如，上例中，虽然输入变量的值一样，但因为输出数据类型的大小不同，所以得到的结果不同。

2.3.3　ROL（左循环移位指令）

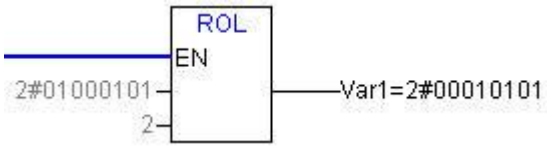
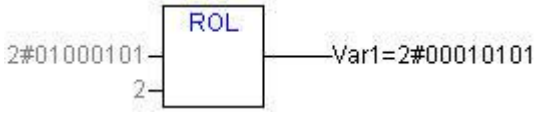
1. 功能：对操作数进行按位循环左移，左边移出的位直接补充到右边最低位。
2. 输入/输出变量类型：BYTE、INT、LINT、WORD、DWORD、LWORD、SINT、UINT、ULINT。

3. 指令使用举例

变量定义

VAR | VAR_INPUT | VAR_OUTPUT | VAR_IN_O

	名称	地址	类型	初值	注
0001	IN_BYTE		BYTE	16#45	
0002	Var1_BYTE		BYTE		
0003	Var2_WORD		WORD		

编程语言	程　序
梯形图（LD）	
结构化文本（ST）	Var1_BYTE:=ROL(16#45,2);　（*结果 Var1_BYTE 为 16#15*） Var2_WORD:=ROL(16#45,2);　（*结果 Var2_WORD 为 16#0114*）
功能块（FBD）	

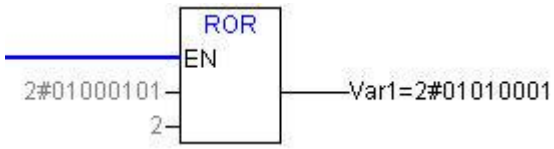
2.3.4　ROR（右循环移位指令）

1. 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。
2. 输入/输出变量类型：BYTE、INT、LINT、WORD、DWORD、LWORD、SINT、UINT、ULINT。
3. 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		

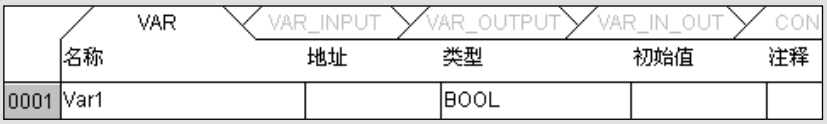
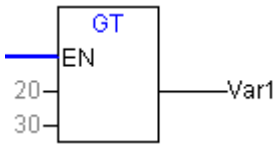
编程语言	程　序
梯形图（LD）	
结构化文本（ST）	Var1:=ROR(16#45,2);　（*结果 Var1 为 16#51*） Var2:=ROR(16#45,2);　（*结果 Var2 为 16#4011*）
功能块（FBD）	

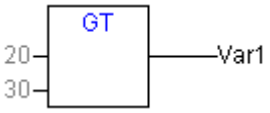
对于移位的位数，若该变量是个常数则被视为最小数据类型。输入变量的数据类型对计算毫无影响，但输出变量的类型是不能忽视的。例如，上例中，虽然输入变量的值一样，但因为输出数据类型的大小不同，所以得到的结果不同。

2.4 比较指令

2.4.1 GT（大于指令）

1. 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为 TRUE，否则输出为 FALSE。
2. 输入变量类型：除数组以外的任何标准类型。
3. 输出变量类型：BOOL。
4. 指令使用举例

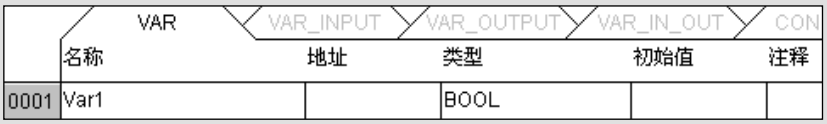
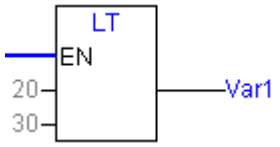
<table>
<tr><td colspan="2">变量定义</td></tr>
<tr><td colspan="2"> 
</td></tr>
<tr><td>编程语言</td><td>程　序</td></tr>
<tr><td>梯形图（LD）</td><td> 

（*结果 Var1 为 FALSE*）</td></tr>
<tr><td>结构化文本（ST）</td><td>Var1:=20>30;</td></tr>
<tr><td>功能块（FBD）</td><td> 

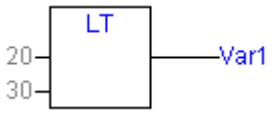
（*结果 Var1 为 FALSE*）</td></tr>
</table>

2.4.2 LT（小于指令）

1. 功能：判断两个操作数的大小，当第一个数小于第二个数时输出为 TRUE，否则输出为 FALSE。
2. 输入变量类型：除数组以外的任何标准类型。
3. 输出变量类型：BOOL。
4. 指令使用举例

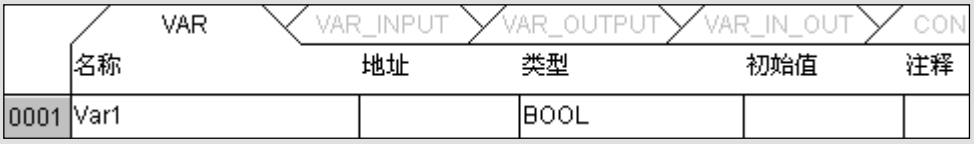
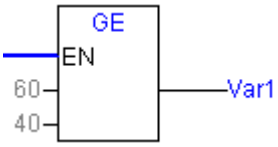
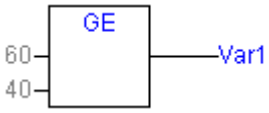
<table>
<tr><td colspan="2">变量定义</td></tr>
<tr><td colspan="2"> 
</td></tr>
<tr><td>编程语言</td><td>程　序</td></tr>
<tr><td>梯形图（LD）</td><td> 

（*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>
</table>

功能块（FBD）	 （*结果 Var1 为 TRUE*）

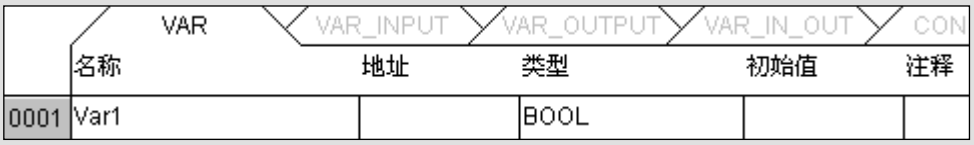
2.4.3 GE（大于等于指令）

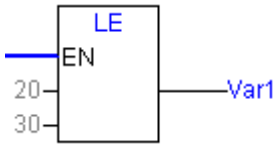
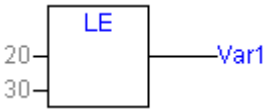
1. 功能：判断两个操作数的大小，当第一个数大于等于第二个数时返回 TRUE，否则结果为 FALSE。
2. 输入变量类型：除数组以外的任何标准类型。
3. 输出变量类型：BOOL。
4. 指令使用举例

5. 变量定义	
	
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*）
功能块（FBD）	 （*结果 Var1 为 TRUE*）

2.4.4 LE（小于等于指令）

1. 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。
2. 输入变量类型：除数组以外的任何标准类型。
3. 输出变量类型：BOOL。
4. 指令使用举例

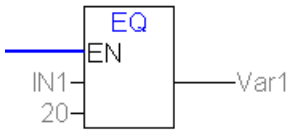
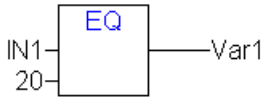
变量定义


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*）
功能块（FBD）	 （*结果 Var1 为 TRUE*）

2.4.5　EQ（等于指令）

1. 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。
2. 输入变量类型：除数组以外的任何标准类型。
3. 输出变量类型：BOOL。
4. 指令使用举例

变量定义				
VAR / VAR_INPUT / VAR_OUTPUT / VAR_IN				
	名称	地址	类型	初值
0001	Var1		BOOL	
0002	IN1		INT	

编程语言	程　序
梯形图（LD）	
结构化文本（ST）	Var1:=IN1=20;　　（*当 IN1 的值为 20 时，结果 Var1 为 TRUE *）
功能块（FBD）	

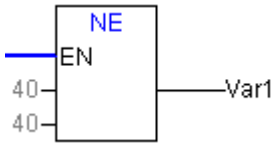
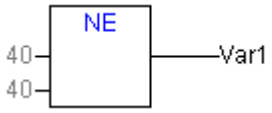
2.4.6　NE（不等于指令）

1. 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。
2. 输入变量类型：除数组以外的任何标准类型。
3. 输出变量类型：BOOL。

4. 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*）
功能块（FBD）	 （*结果 Var1 为 FALSE*）

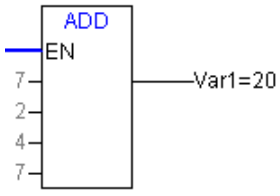
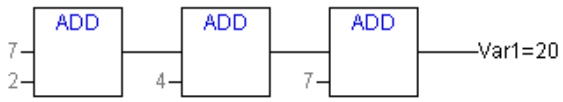
2.5 数学运算指令

2.5.1　ADD（加法指令）

1. 功能：两个（或者多个）变量或常量相加。
2. 输入/输出变量类型：BYTE、WORD、DWORD、LWORD、SINT、USINT、INT、LINT、UINT、DINT、UDINT、ULINT、REAL、LREAL、TIME。
3. 指令使用举例

变量定义

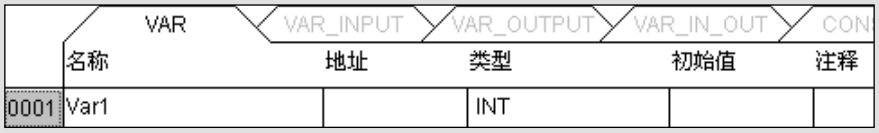
	VAR	VAR_INPUT	VAR_OUTPUT	VAR_IN_OUT	CON
	名称	地址	类型	初始值	注释
0001	Var1		INT		

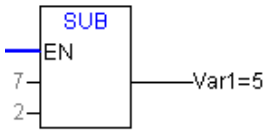
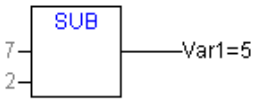
编程语言	程　序
梯形图（LD）	
结构化文本（ST）	Var1:=7+2+4+7;　（*结果 Var1 为 20*）
功能块（FBD）	

2.5.2 SUB（减法指令）

1. 功能：两个（或者多个）变量或常量相减。
2. 输入/输出变量类型：BYTE、WORD、DWORD、LWORD、SINT、USINT、INT、LINT、UINT、DINT、UDINT、ULINT、REAL、LREAL、TIME。
3. 指令使用举例

变量定义



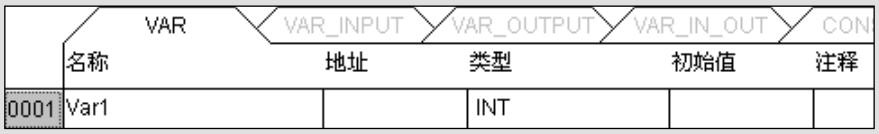
编程语言	程　序
梯形图（LD）	
结构化文本（ST）	Var1:=7-2;　　（*结果 Var1 为 5*）
功能块（FBD）	

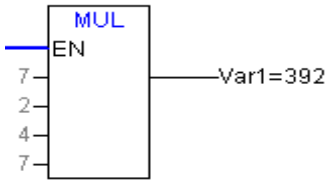
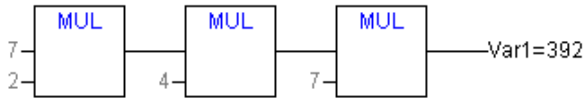
时间型变量也可以使用加法和减法指令，得到一个新的时间量。例如：t#1m35s - t#50s =t#45s，但时间结果不能有负值。

2.5.3 MUL（乘法指令）

1. 功能：两个（或者多个）变量或常量相乘。
2. 输入/输出变量类型：BYTE、WORD、DWORD、LWORD、SINT、USINT、INT、LINT、UINT、DINT、UDINT、ULINT、REAL、LREAL。
3. 指令使用举例

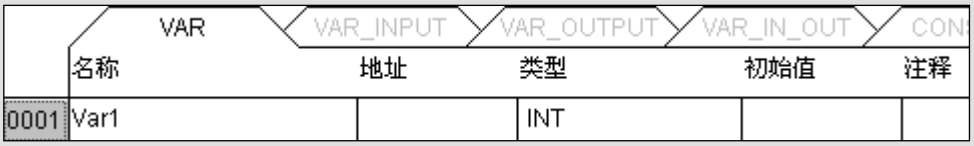
变量定义

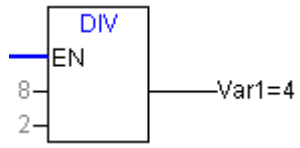
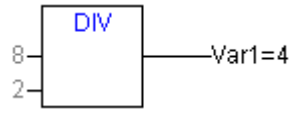


编程语言	程　序
梯形图（LD）	
结构化文本（ST）	Var1:=7*2*4*7;　（*结果 Var1 为 392*）
功能块（FBD）	

2.5.4　DIV（除法指令）

1. 功能：变量或常量相除。
2. 输入/输出变量类型：BYTE、WORD、DWORD、LWORD、SINT、USINT、INT、LINT、UINT、DINT、UDINT、ULINT、REAL、LREAL。
3. 指令使用举例

变量定义


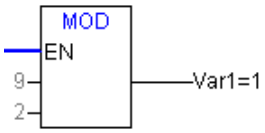
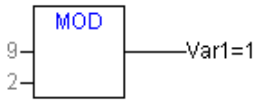
编程语言	程　序
梯形图（LD）	
结构化文本（ST）	Var1:=8/2;　（*结果 Var1 为 4*）
功能块（FBD）	

2.5.5　MOD（取余指令）

1. 功能：变量或常量相除取余，其结果将是一个整数。
2. 输入/输出变量类型：BYTE、WORD、DWORD、LWORD、SINT、USINT、INT、LINT、UINT、DINT、UDINT、ULINT。

3. 指令使用举例

变量定义

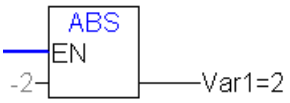
编程语言	程　序
梯形图（LD）	
结构化文本（ST）	Var1:=9 MOD 2;　（*结果 Var1 为 1*）
功能块（FBD）	

2.5.6 ABS（绝对值指令）

1. 功能：将输入数据的绝对值赋予输出变量。
2. 输入/输出变量类型：见下表

输入变量类型	输出变量类型
INT	INT、LINT、WORD、DWORD、LWORD、DINT、UINT、ULINT、REAL、LREAL
LINT	LINT、LWORD、REAL、LREAL
REAL	REAL、LREAL
LREAL	LREAL
BYTE	BYTE、INT、LINT、WORD、DWORD、LWORD、DINT、UINT、ULINT、REAL、LREAL
WORD	WORD、DWORD、LWORD、DINT、REAL、LREAL
DWORD	DWORD、DINT、REAL、LREAL
LWORD	LWORD、LREAL
SINT	INT、LINT、BYTE、WORD、DWORD、LWORD、DINT、UINT、ULINT、REAL、LREAL
USINT	INT、LINT、BYTE、WORD、DWORD、LWORD、DINT、UINT、ULINT、REAL、LREAL
UINT	WORD、DWORD、LWORD、DINT、UINT、ULINT、REAL、LREAL
DINT	DWORD、LWORD、DINT、ULINT、REAL、LREAL
UDINT	DWORD、LWORD、DINT、UDINT、ULINT、REAL、LREAL
ULINT	LWORD、ULINT、LREAL

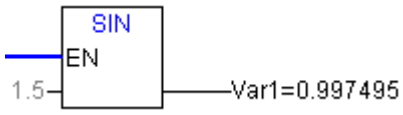
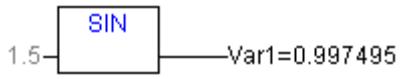
3. 指令使用举例

变量定义	
	
编程语言	**程　序**
梯形图（LD）	
结构化文本（ST）	Var1=ABS(-2);　　（*结果 Var1 为 2*）
功能块（FBD）	

2.5.7　SIN（正弦指令）

1. 功能：求输入数据的正弦值。输入数据以弧度表示。$\text{弧度(rad)} = \text{角度} * \frac{\pi}{180^o}$

2. 输入变量类型：BYTE、WORD、DWORD、INT、DINT、REAL、SINT、USINT、UINT 和 UDINT。
3. 输出变量类型：必须是 REAL 或 LREAL 型。
4. 指令使用举例

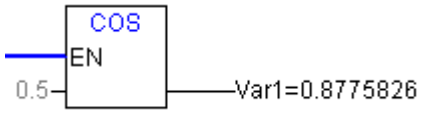
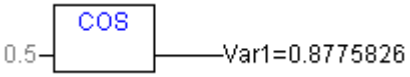
变量定义	
	
编程语言	**程　序**
梯形图（LD）	
结构化文本（ST）	Var1:=SIN(1.5);　　（*结果 Var1 为 0.997495 *）
功能块（FBD）	

2.5.8　COS（余弦指令）

1. 功能：求输入数据的余弦值。输入数据以弧度表示。$\text{弧度(rad)} = \text{角度} * \frac{\pi}{180^o}$

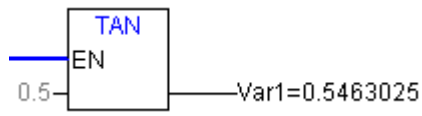
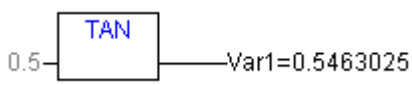
2. 输入变量类型：BYTE、WORD、DWORD、INT、DINT、REAL、SINT、USINT、UINT 和 UDINT。
3. 输出变量类型：必须是 REAL 或 LREAL 型。

4. 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 *）
功能块（FBD）	

2.5.9　TAN（正切指令）

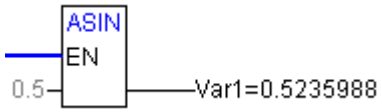
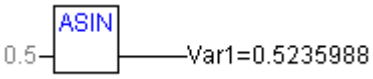
1. 功能：求输入数据的正切值。输入数据以弧度表示。弧度(rad) = 角度 * $\frac{\pi}{180^{o}}$
2. 输入变量类型：BYTE、WORD、DWORD、INT、DINT、REAL、SINT、USINT、UINT 和 UDINT。
3. 输出变量类型：必须是 REAL 或 LREAL 型。
4. 指令使用举例

变量定义	
	
编程语言	**程　序**
梯形图（LD）	
结构化文本（ST）	Var1:=TAN(0.5);　　（*结果 Var1 为 0.5463025 *）
功能块（FBD）	

2.5.10 ASIN（反正弦指令）

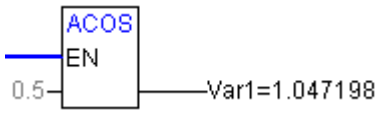
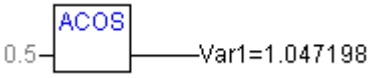
1. 功能：求输入数据的反正弦值。
2. 输入变量类型：BYTE、WORD、DWORD、INT、DINT、REAL、SINT、USINT、UINT 和 UDINT。
3. 输出变量类型：必须是 REAL 或 LREAL 型。

4. 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:=ASIN(0.5);　　（*结果 Var1 为 0.5235988 *）
功能块（FBD）	

2.5.11 ACOS（反余弦指令）

1. 功能：求输入数据的反余弦值。
2. 输入变量类型：BYTE、WORD、DWORD、INT、DINT、REAL、SINT、USINT、UINT 和 UDINT。
3. 输出变量类型：必须是 REAL 或 LREAL 型。
4. 指令使用举例

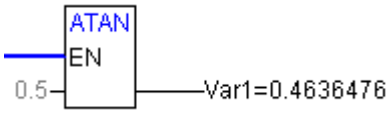
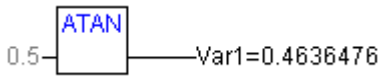
变量定义	
	
编程语言	**程　序**
梯形图（LD）	
结构化文本（ST）	Var1:=ACOS(0.5);　　（*结果 Var1 为 1.047198 *）
功能块（FBD）	

2.5.12 ATAN（反正切指令）

1. 功能：求输入数据的反正切值。
2. 输入变量类型：BYTE、WORD、DWORD、INT、DINT、REAL、SINT、USINT、UINT 和 UDINT。
3. 输出变量类型：必须是 REAL 或 LREAL 型。

4. 指令使用举例

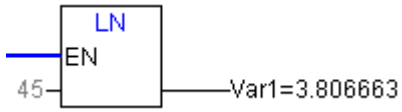
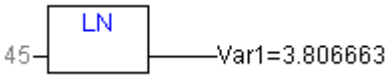
变量定义					
VAR	VAR_INPUT	VAR_OUTPUT	VAR_IN_OUT	CON	
名称	地址	类型	初始值	注释	
0001	Var1		REAL		

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

2.5.13 LN（自然对数指令）

1. 功能：对输入数据求自然对数，输入数据必须为正数。
2. 输入变量类型：BYTE、WORD、DWORD、INT、DINT、REAL、SINT、USINT、UINT 和 UDINT。
3. 输出变量类型：必须是 REAL 或 LREAL 型。
4. 指令使用举例

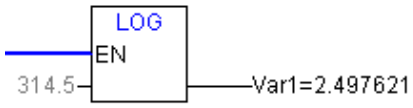
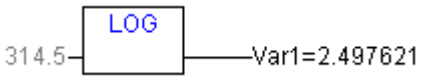
变量定义					
VAR	VAR_INPUT	VAR_OUTPUT	VAR_IN_OUT	CON	
名称	地址	类型	初始值	注释	
0001	Var1		REAL		

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

2.5.14 LOG（常用对数指令）

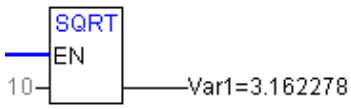
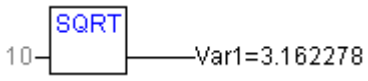
1. 功能：对输入数据求以 10 为底的对数，输入数据必须为正数。
2. 输入变量类型：BYTE、WORD、DWORD、INT、DINT、REAL、SINT、USINT、UINT 和 UDINT。
3. 输出变量类型：必须是 REAL 或 LREAL 型。

4. 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 *）
功能块（FBD）	

2.5.15 SQRT（平方根指令）

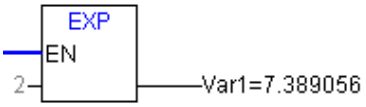
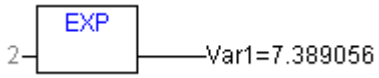
1. 功能：对输入数据求平方根，输入数据为非负数。
2. 输入变量类型：BYTE、WORD、DWORD、INT、DINT、REAL、SINT、USINT、UINT 和 UDINT。
3. 输出变量类型：必须是 REAL 或 LREAL 型。
4. 指令使用举例

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*）
功能块（FBD）	

2.5.16 EXP（指数指令）

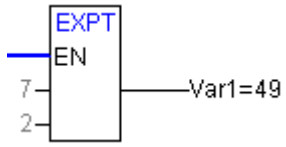
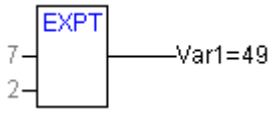
1. 功能：以输入数据为指数的幂计算，即 $y = e^x$，其中 x 为输入，y 为输出。
2. 输入变量类型：BYTE、WORD、DWORD、INT、DINT、REAL、SINT、USINT、UINT 和 UDINT。
3. 输出变量类型：必须是 REAL 或 LREAL 型。

4. 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*）
功能块（FBD）	

2.5.17 EXPT（幂指令）

1. 功能：对输入数据求幂，第一个输入数据为幂底数，第二个输入数据为幂指数。
2. 输入变量类型：BYTE、WORD、DWORD、INT、DINT、REAL、SINT、USINT、UINT 和 UDINT。
3. 输出变量类型：必须是 REAL 或 LREAL 型。
4. 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:=EXPT(7,2);　　（*结果 Var1 为 49 *）
功能块（FBD）	

2.6 选择指令

所有选择操作也都可以使用变量来操作。为了能够更加简明地说明问题各例只使用常量。输出数据的存储长度应大于被选择的输入数据的存储长度。

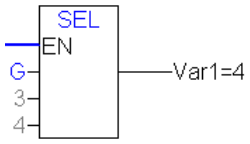
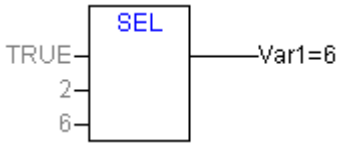
2.6.1 SEL（二选一指令）

1. 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。
2. 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。
3. 输入/输出变量类型：G 必须是 BOOL 类型，IN0、IN1 和输出变量可以是任意数据类型。
4. 指令使用举例

变量定义

VAR | VAR_INPUT | VAR_OUTPUT

	名称	地址	类型
0001	G		BOOL
0002	Var1		INT

编程语言	程　序
梯形图（LD）	
结构化文本（ST）	Var1:=SEL(G,3,4);　（*G 为 TRUE 时，Var1=4；G 为 FLASE 时，Var1=3 *）
功能块（FBD）	

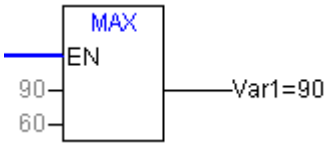
2.6.2 MAX（取最大值指令）

1. 功能：在两个输入变量中选择最大值作为输出。
2. 指令格式：OUT:=MAX(IN0, IN1)，其中 IN0 和 IN1 分别为第一个输入数据和第二个输入数据，OUT 是输出数据。
3. 输入/输出变量类型：IN0、IN1 和 OUT 可以是任意数据类型。
4. 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		

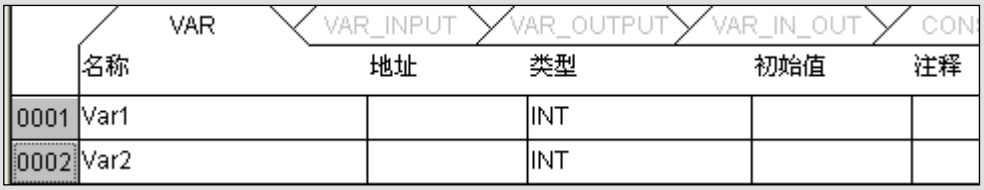
编程语言	程　序
梯形图（LD）	

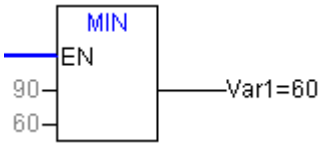
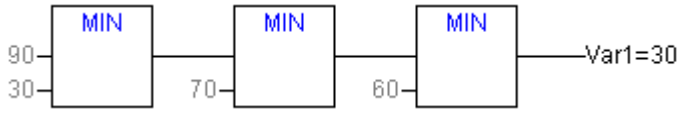
结构化文本（ST）	Var1:=MAX(30,40); （*结果 Var1 为 90 *） Var2:=MAX(40,MAX(90,30)); （*结果 Var2 为 90 *）
功能块（FBD）	

2.6.3 MIN（取最小值指令）

1. 功能：在两个输入数据中选择最小值作为输出。
2. 指令格式：OUT:=MIN(IN0, IN1)，其中 IN0 和 IN1 分别为第一个输入数据和第二个输入数据，OUT 是输出数据。
3. 输入/输出变量类型：IN0、IN1 和 OUT 可以是任意数据类型。
4. 指令使用举例

变量定义



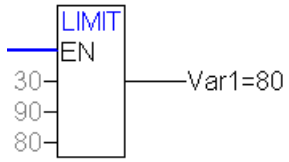
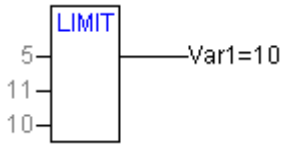
编程语言	**程　序**
梯形图（LD）	
结构化文本（ST）	Var1:=MIN(90,30); （*结果 Var1 为 30 *） Var2:=MIN(MIN(90,30),60); （*结果 Var2 为 30 *）
功能块（FBD）	

2.6.4 LIMIT（极限值指令）

1. 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，则直接将输入数据作为输出数据进行输出。若输入数据大于最大值，则将最大值作为输出值。若输入数据小于最小值，则将最小值作为输出值。
2. 指令格式：OUT:=LIMIT(Min, IN, Max)
3. 输入/输出变量类型：IN 和 OUT 可以是任意数据类型。

4. 指令使用举例

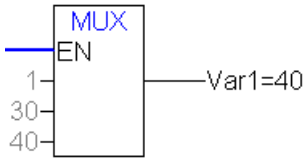
变量定义

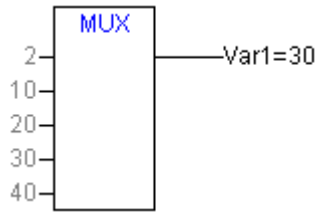
编程语言	程　序
梯形图（LD）	 （*90 为输入数据，30 为最小值，80 为最大值*）
结构化文本（ST）	Var1:=LIMIT(30,90,80);　　（*结果 Var1 为 80 *）
功能块（FBD）	 （*11 为输入数据，5 为最小值，10 为最大值*）

2.6.5　MUX（多选一指令）

1. 功能：通过控制数在多个输入数据中选择一个作为输出。
2. 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 K 为 x 时选择序列为 x+1 的输入数据作为输出。
3. 输入/输出变量类型：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。
4. 指令使用举例

变量定义

编程语言	程　序
梯形图（LD）	 （*控制数为 1，对应 40，所以输出 40 *）
结构化文本（ST）	Var1:=MUX(0,30,40,50,60,70,80);　　（*结果 Var1 为 30*）

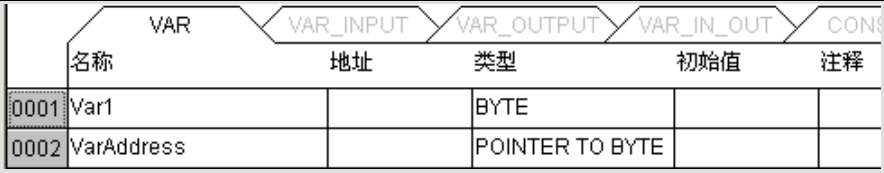
功能块（FBD）	 （*2 为控制数据，对应于 30，所以结果为 30*）

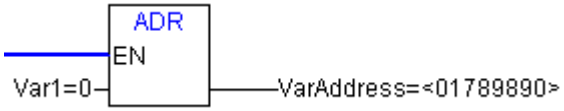
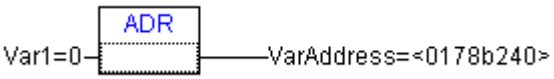
2.7 地址运算指令

2.7.1 ADR（取地址指令）

1. 功能：获取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其他 POU。
2. 输入变量类型：任何类型。
3. 输出变量类型：指针类型。
4. 指令使用举例

变量定义

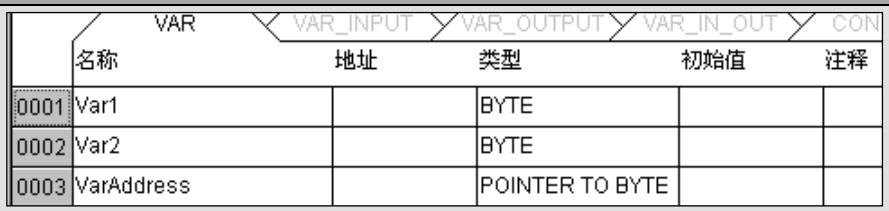


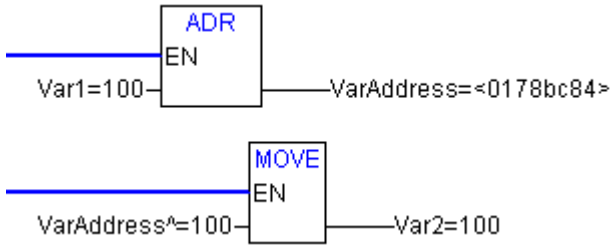
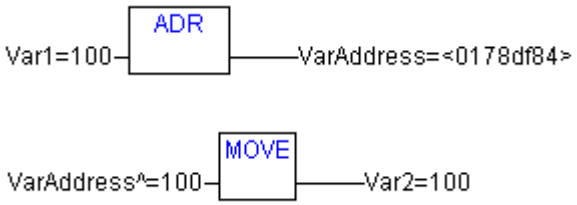
编程语言	**程　序**
梯形图（LD）	
结构化文本（ST）	VarAddress:=ADR(Var1);
功能块（FBD）	

2.7.2 ^（取地址内容指令）

1. 功能：在指针变量后增加“^”符号，以获取该指针所指地址的数据。
2. 指令使用举例

变量定义

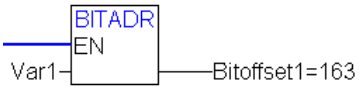
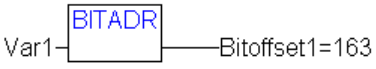


编程语言	程　序
梯形图（LD）	
结构化文本（ST）	VarAddress:=ADR(Var1); Var2:=VarAddress^;　　（*结果 Var2 为 100 *）
功能块（FBD）	

2.7.3　BITADR（位地址指令）

1. 功能：获取 BOOL 量的位偏移地址。
2. 输入变量类型：BOOL 类型。
3. 输出变量类型：DWORD 类型。
4. 指令使用举例

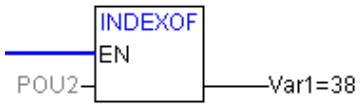
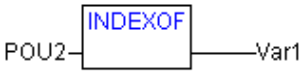
变量定义


编程语言	程　序
梯形图（LD）	
结构化文本（ST）	Bitoffset1:=BITADR(Var1);　（*Var1 的地址为%MX0.20.3，其偏移地址为 163，即 0*65536*8+20*8+3*）
功能块（FBD）	

2.7.4　INDEXOF（索引指令）

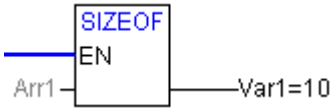
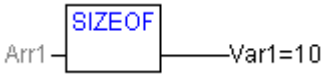
1. 功能：在 POU 中执行索引指令，可以寻找 POU 的索引号，
2. 输入变量类型：必须是 POU 的名称。

3. 指令使用举例

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

2.7.5　SIZEOF（获取数据类型大小指令）

1. 功能：获得输入数据占用的字节数，常用于指针运算。
2. 输入变量类型：无限制。
3. 输出变量类型：BYTE、WORD、DWORD、LWORD、SINT、USINT、INT、LINT、UINT、DINT、UDINT、ULINT。
4. 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:=SIZEOF(arr1);　　（*结果 Var1 为 10*）
功能块（FBD）	

2.8 初始化指令

1. INI：初始化操作指令
2. 功能：用于初始化程序中使用的功能块实例的内部保持型变量。

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

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

4. 指令使用举例

主程序 PLC_PRG 变量定义：

VAR | VAR_INPUT | VAR_OUTPUT | VAR_IN

	名称	地址	类型	初值
0001	FB1		FB_R	
0002	VarBOOL1		BOOL	
0003	VarBOOL2		BOOL	

FB_R（FB）变量定义：

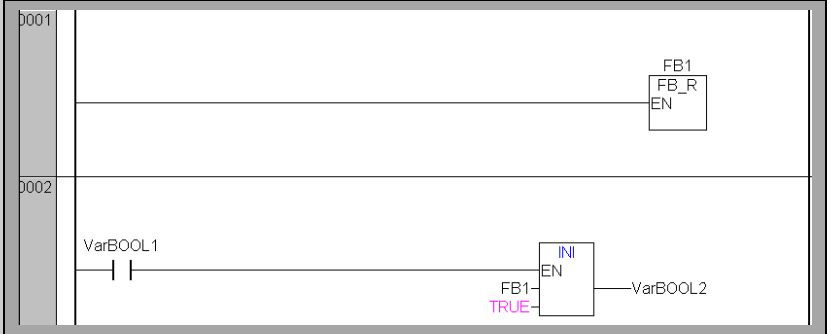
VAR | VAR_INPUT | VAR_OUTPUT | VAR_IN_OUT | CONSTANT | RETAIN

	名称	地址	类型	初值	注释
0001	R1		R_TRIG		
0002	varWord		WORD	2	
0003	VarByte		BYTE	1	

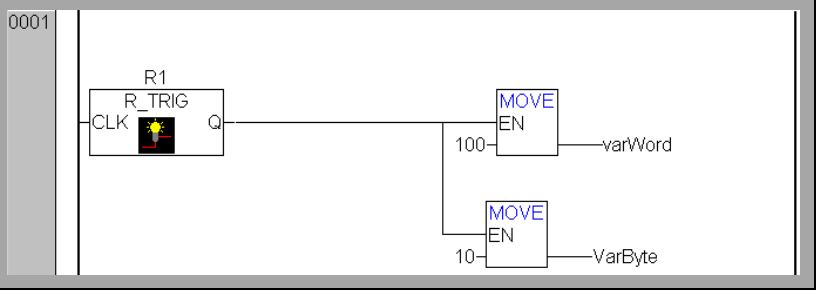
编程语言：梯形图（LD）

程　序

PLC_PRG



功能块 FB_R



5. 程序说明：

程序运行第一个扫描周期将变量 VarWord、VarByte 的值改为 100、10，强制变量 VarBOOL1 接通，INI 指令执行，将两个保持型变量恢复为初始值 2 和 1。

2.9 数据类型转换指令

数据类型转换指令也称为转换操作符，CodeSys 提供 380 条相关指令，用于各种数据类型之间相互转换。下表详细列出各条数据类型转换指令。

1. 数据类型转换指令列表

BOOL_TO_<TYPE>	BYTE_TO_<TYPE>	DATE_TO_<TYPE>	DINT/LINT_TO_<TYPE>
BOOL_TO_BYTE BOOL_TO_DATE BOOL_TO_DINT BOOL_TO_DT BOOL_TO_DWORD BOOL_TO_INT BOOL_TO_LINT BOOL_TO_LREAL BOOL_TO_LWORD BOOL_TO_REAL BOOL_TO_SINT BOOL_TO_STRING BOOL_TO_TIME BOOL_TO_TOD BOOL_TO_UDINT BOOL_TO_UINT BOOL_TO_ULINT BOOL_TO_USINT BOOL_TO_WORD	BYTE_TO_BOOL BYTE_TO_DATE BYTE_TO_DINT BYTE_TO_DT BYTE_TO_DWORD BYTE_TO_INT BYTE_TO_LINT BYTE_TO_LREAL BYTE_TO_LWORD BYTE_TO_REAL BYTE_TO_SINT BYTE_TO_STRING BYTE_TO_TIME BYTE_TO_TOD BYTE_TO_UDINT BYTE_TO_UINT BYTE_TO_ULINT BYTE_TO_USINT BYTE_TO_WORD	DATE_TO_BOOL DATE_TO_BYTE DATE_TO_DINT DATE_TO_DT DATE_TO_DWORD DATE_TO_INT DATE_TO_LINT DATE_TO_LREAL DATE_TO_LWORD DATE_TO_REAL DATE_TO_SINT DATE_TO_STRING DATE_TO_TIME DATE_TO_TOD DATE_TO_UDINT DATE_TO_UINT DATE_TO_ULINT DATE_TO_USINT DATE_TO_WORD	DINT_TO_BOOL DINT_TO_BYTE DINT_TO_DATE DINT_TO_DT DINT_TO_DWORD DINT_TO_INT DINT_TO_LINT DINT_TO_LREAL DINT_TO_LWORD DINT_TO_REAL DINT_TO_SINT DINT_TO_STRING DINT_TO_TIME DINT_TO_TOD DINT_TO_UDINT DINT_TO_UINT DINT_TO_ULINT DINT_TO_USINT DINT_TO_WORD
DT_TO_<TYPE>	**DWORD/LWORD_TO_<TYPE>**	**INT/LINT_TO_<TYPE>**	**WORD_TO_<TYPE>**
DT_TO_BOOL DT_TO_BYTE DT_TO_DATE DT_TO_DINT DT_TO_DWORD DT_TO_INT DT_TO_LINT DT_TO_LREAL DT_TO_LWORD DT_TO_REAL DT_TO_SINT DT_TO_STRING DT_TO_TIME DT_TO_TOD DT_TO_UDINT DT_TO_UINT DT_TO_ULINT DT_TO_USINT DT_TO_WORD	DWORD_TO_BOOL DWORD_TO_BYTE DWORD_TO_DATE DWORD_TO_DINT DWORD_TO_DT DWORD_TO_INT DWORD_TO_LINT DWORD_TO_LREAL DWORD_TO_LWORD DWORD_TO_REAL DWORD_TO_SINT DWORD_TO_STRING DWORD_TO_TIME DWORD_TO_TOD DWORD_TO_UDINT DWORD_TO_UINT DWORD_TO_ULINT DWORD_TO_USINT DWORD_TO_WORD	INT_TO_BOOL INT_TO_BYTE INT_TO_DATE INT_TO_DINT INT_TO_DT INT_TO_DWORD INT_TO_LINT INT_TO_LREAL INT_TO_LWORD INT_TO_REAL INT_TO_SINT INT_TO_STRING INT_TO_TIME INT_TO_TOD INT_TO_UDINT INT_TO_UINT INT_TO_ULINT INT_TO_USINT INT_TO_WORD	WORD_TO_BOOL WORD_TO_BYTE WORD_TO_DATE WORD_TO_DINT WORD_TO_DT WORD_TO_DWORD WORD_TO_INT WORD_TO_LINT WORD_TO_LREAL WORD_TO_LWORD WORD_TO_REAL WORD_TO_SINT WORD_TO_STRING WORD_TO_TIME WORD_TO_TOD WORD_TO_UDINT WORD_TO_UINT WORD_TO_ULINT WORD_TO_USINT
REAL/LREAL_TO_<TYPE>	**SINT_TO_<TYPE>**	**STRING_TO_<TYPE>**	**TIME_TO_<TYPE>**
REAL_TO_BOOL REAL_TO_BYTE REAL_TO_DATE REAL_TO_DINT REAL_TO_DT REAL_TO_DWORD REAL_TO_INT REAL_TO_LINT REAL_TO_LREAL REAL_TO_LWORD REAL_TO_SINT REAL_TO_STRING REAL_TO_TIME REAL_TO_TOD REAL_TO_UDINT REAL_TO_UINT REAL_TO_ULINT REAL_TO_USINT REAL_TO_WORD	SINT_TO_BOOL SINT_TO_BYTE SINT_TO_DATE SINT_TO_DINT SINT_TO_DT SINT_TO_DWORD SINT_TO_INT SINT_TO_LINT SINT_TO_LREAL SINT_TO_LWORD SINT_TO_REAL SINT_TO_STRING SINT_TO_TIME SINT_TO_TOD SINT_TO_UDINT SINT_TO_UINT SINT_TO_ULINT SINT_TO_USINT SINT_TO_WORD	STRING_TO_BOOL STRING_TO_BYTE STRING_TO_DATE STRING_TO_DINT STRING_TO_DT STRING_TO_DWORD STRING_TO_INT STRING_TO_LINT STRING_TO_LREAL STRING_TO_LWORD STRING_TO_REAL STRING_TO_SINT STRING_TO_TIME STRING_TO_TOD STRING_TO_UDINT STRING_TO_UINT STRING_TO_ULINT STRING_TO_USINT STRING_TO_WORD	TIME_TO_BOOL TIME_TO_BYTE TIME_TO_DATE TIME_TO_DINT TIME_TO_DT TIME_TO_DWORD TIME_TO_INT TIME_TO_LINT TIME_TO_LREAL TIME_TO_LWORD TIME_TO_REAL TIME_TO_SINT TIME_TO_STRING TIME_TO_TOD TIME_TO_UDINT TIME_TO_UINT TIME_TO_ULINT TIME_TO_USINT TIME_TO_WORD
TOD_TO_<TYPE>	**UDINT/ULINT_TO_<TYPE>**	**UINT_TO_<TYPE>**	**USINT_TO_<TYPE>**
TOD_TO_BOOL TOD_TO_BYTE TOD_TO_DATE TOD_TO_DINT TOD_TO_DT TOD_TO_DWORD TOD_TO_INT TOD_TO_LINT TOD_TO_LREAL TOD_TO_LWORD TOD_TO_REAL TOD_TO_SINT TOD_TO_STRING TOD_TO_TIME TOD_TO_UDINT TOD_TO_UINT TOD_TO_ULINT TOD_TO_USINT TOD_TO_WORD	UDINT_TO_BOOL UDINT_TO_BYTE UDINT_TO_DATE UDINT_TO_DINT UDINT_TO_DT UDINT_TO_DWORD UDINT_TO_INT UDINT_TO_LINT UDINT_TO_LREAL UDINT_TO_LWORD UDINT_TO_REAL UDINT_TO_SINT UDINT_TO_STRING UDINT_TO_TIME UDINT_TO_TOD UDINT_TO_UINT UDINT_TO_ULINT UDINT_TO_USINT UDINT_TO_WORD	UINT_TO_BOOL UINT_TO_BYTE UINT_TO_DATE UINT_TO_DINT UINT_TO_DT UINT_TO_DWORD UINT_TO_INT UINT_TO_LINT UINT_TO_LREAL UINT_TO_LWORD UINT_TO_REAL UINT_TO_SINT UINT_TO_STRING UINT_TO_TIME UINT_TO_TOD UINT_TO_UDINT UINT_TO_ULINT UINT_TO_USINT UINT_TO_WORD	USINT_TO_BOOL USINT_TO_BYTE USINT_TO_DATE USINT_TO_DINT USINT_TO_DT USINT_TO_DWORD USINT_TO_INT USINT_TO_LINT USINT_TO_LREAL USINT_TO_LWORD USINT_TO_REAL USINT_TO_SINT USINT_TO_STRING USINT_TO_TIME USINT_TO_TOD USINT_TO_UDINT USINT_TO_UINT USINT_TO_ULINT USINT_TO_WORD

2. 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，（例如从 INT 转变城 BYTE，或从 DINT 变成 WORD）。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，则该数据的高字节将被忽略。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，右边部分将被截去。

2.9.1 BOOL_TO_<TYPE>（布尔类型转换指令）

1. 功能：把布尔数据类型转换为其它数据类型。
2. 输入/输出变量类型：
   - 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'。
3. 指令使用举例

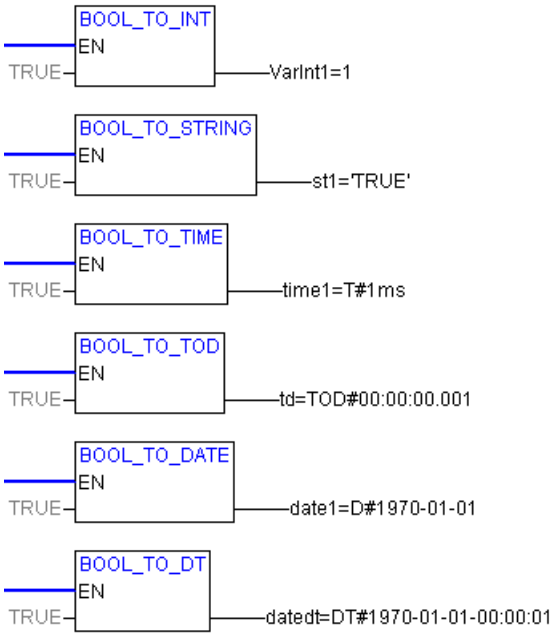
变量定义

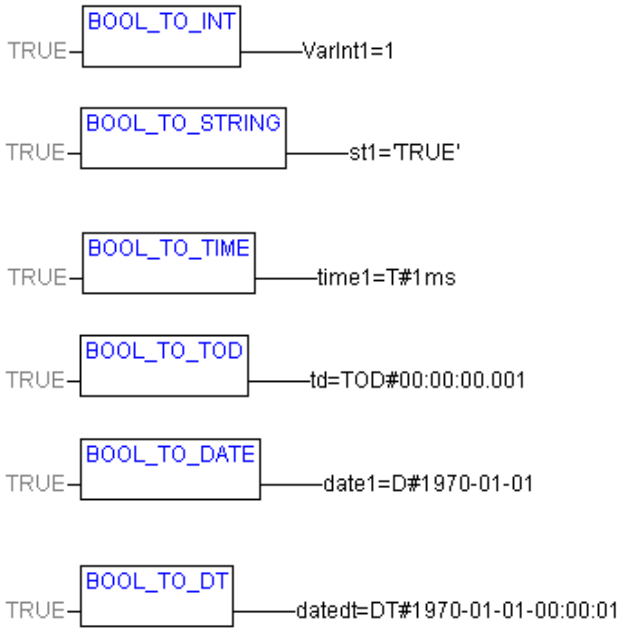
VAR | VAR_INPUT | VAR_OUTPUT | VAR_IN_OUT | CONSTANT

	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：梯形图（LD）

程　序



<table>
<tr><td>结构化文本（ST）</td><td>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*）</td></tr>
<tr><td>功 能 块（FBD）</td><td> 
</td></tr>
</table>

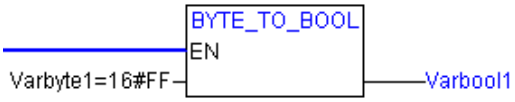
2.9.2 BYTE_TO_<TYPE>（字节类型转换指令）

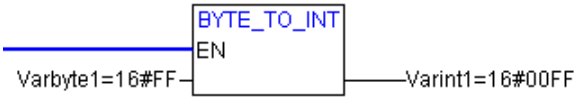
1. 功能：把字节类型转换为其他数据类型。
2. 输入/输出变量类型：
   - 当 BYTE_TO_BOOL 时，如果输入不等于 0 时输出为 TRUE，如果输入等于 0 时输出为 FALSE；
   - 当 BYTE_TO_TIME、BYTE_TO_TOD 时，输入将以毫秒值进行转换；
   - 当 BYTE_TO_DATE、BYTE_TO_DT 时，输入将以秒值进行转换。
3. 指令使用举例

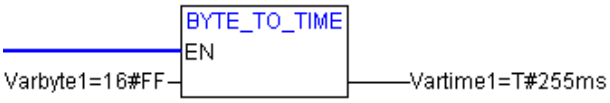
变量定义

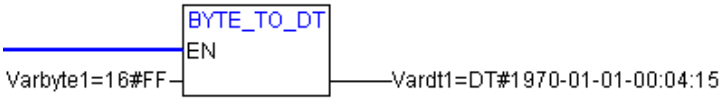
VAR | VAR_INPUT | VAR_OUTPUT | VAR_IN_OUT | CONSTANT

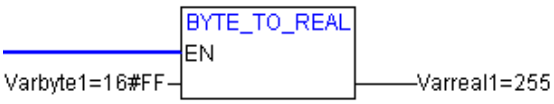
	名称	地址	类型	初始值	注释
0001	Varbool1		BOOL		
0002	Varbyte1		BYTE		
0003	Varint1		INT		
0004	Vartime1		TIME		
0005	Vardt1		DT		
0006	Varreal1		REAL		
0007	Varstring1		STRING		

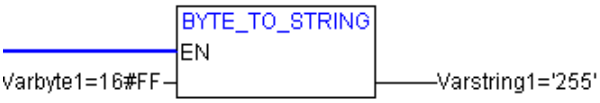
<table>
<tr><th>编程语言</th><th>程　序</th></tr>
<tr><td>梯形图（LD）</td><td>


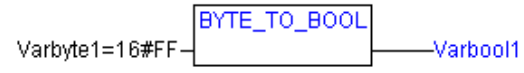
（*结果为 TRUE *）


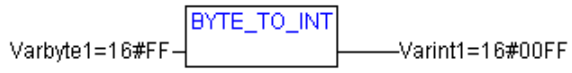
（*结果为 16# 00FF *）


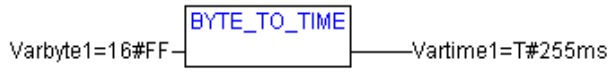
（*结果为 T#255ms *）


（*结果为 DT#1970-01-01-00:04:15 *）


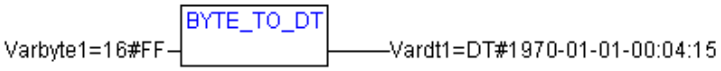
（*结果为 255 *）


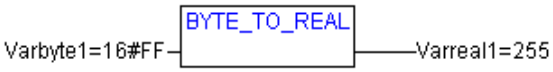
（*结果为字符串'255'*）</td></tr>
<tr><td>结 构 化 文 本（ST）</td><td>Varbyte1:=16#FF;　　（*Varbyte1 取值*）
Varbool1:=BYTE_TO_BOOL(Varbyte1);　　（*结果为 TRUE *）
Varint1:=BYTE_TO_INT(Varbyte1);　　（*结果为 16# FF *）
Vartime1:=BYTE_TO_TIME(Varbyte1);　　（*结果为 T#255ms *）
Vardt1:=BYTE_TO_DT(Varbyte1);　（*结果为 DT#1970-01-01-00:04:15 *）
Varreal1:=BYTE_TO_REAL(Varbyte1);　　（*结果为 255 *）
Varstring1:=BYTE_TO_STRING(Varbyte1);　　（*结果为字符串'255'*）</td></tr>
<tr><td>功能块（FBD）</td><td>


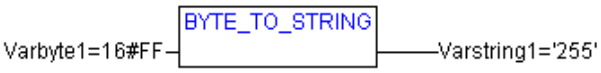
（*结果为 TRUE *）


（*结果为 16# 00FF *）


（*结果为 T#255ms *）</td></tr>
</table>

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


（*结果为 DT#1970-01-01-00:04:15 *）


（*结果为 255 *）


（*结果为字符串'255'*）
</td>
</tr>
</table>

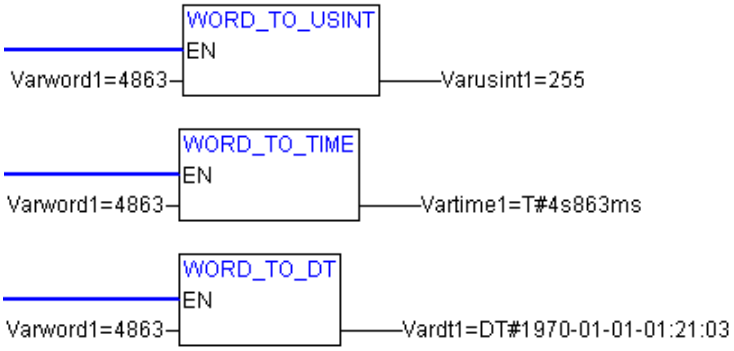
2.9.3 WORD_TO_<TYPE>（字类型转换指令）

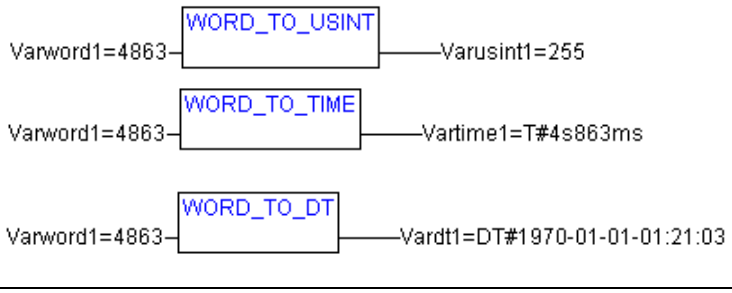
1. 功能：把字类型转换为其它数据类型。
2. 输入/输出变量类型：
   - 当 WORD_TO_BOOL 时，如果输入不等于 0 时输出为 TRUE，如果输入等于 0 时输出为 FALSE；
   - 当 WORD_TO_TIME、WORD_TO_TOD 时，输入将以毫秒值进行转换；
   - 当 WORD_TO_DATE、WORD_TO_DT 时，输入将以秒值进行转换。
3. 指令使用举例

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
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 *）

功能块（FBD）	

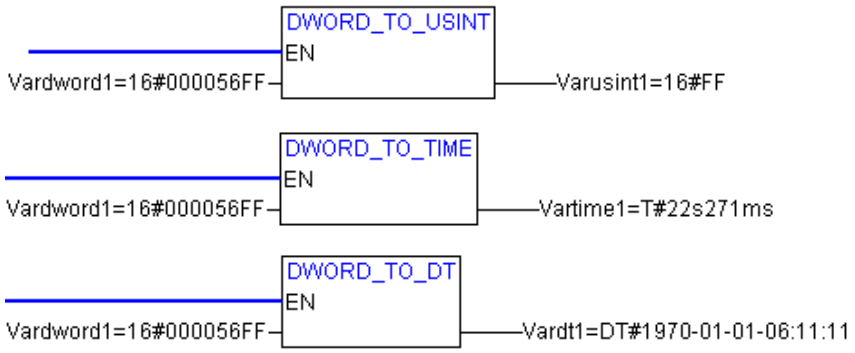
2.9.4 DWORD/LWORD_TO_<TYPE>（双字或长字类型转换指令）

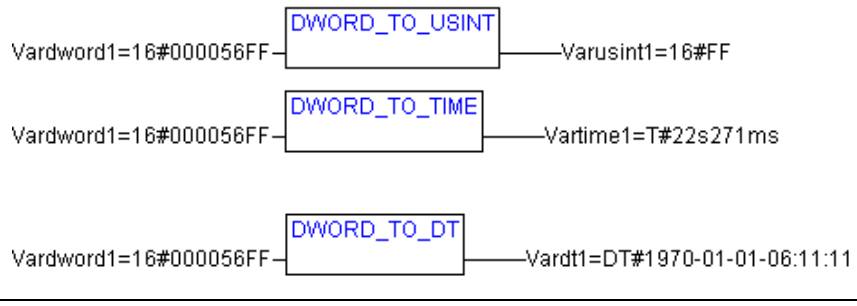
1. 功能：把双字或长字类型转换为其它数据类型。
2. 输入/输出变量类型：
   - 当 DWORD/LWORD_TO_BOOL 时，如果输入不等于 0 时输出为 TRUE，如果输入等于 0 时输出为 FALSE；
   - 当 DWORD/LWORD_TO_TIME、DWORD_TO_TOD 时，输入将以毫秒值进行转换；
   - 当 DWORD/LWORD_TO_DATE、DWORD_TO_DT 时，输入将以秒值进行转换。
3. 指令使用举例

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**
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 *）

功　能　块（FBD）	

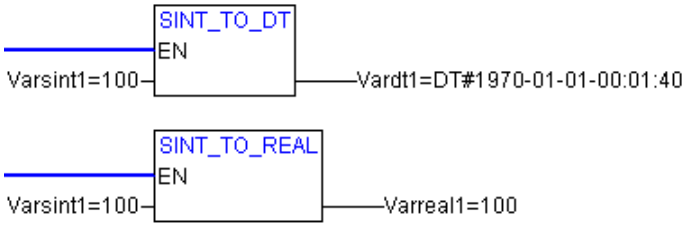
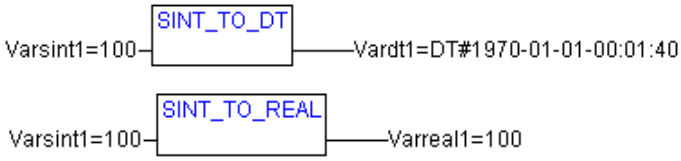
2.9.5 SINT_TO_<TYPE>（短整型转换指令）

1. 功能：把短整型转换为其它数据类型。
2. 输入/输出变量类型：
   - 当 SINT_TO_BOOL 时，如果输入不等于 0 时输出为 TRUE，如果输入等于 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。
3. 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
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*）
功能块（FBD）	

2.9.6 USINT_TO_<TYPE>（无符号短整型转换指令）

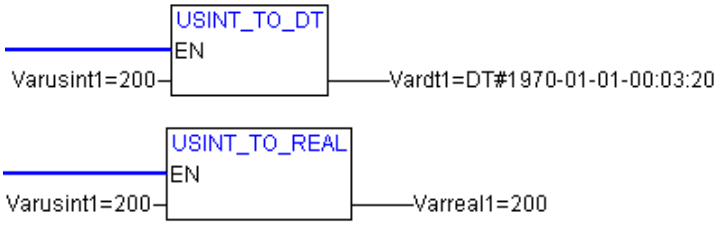
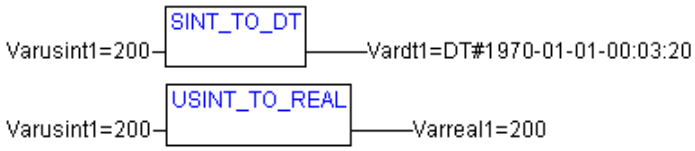
1. 功能：把无符号短整型转换为其它数据类型。
2. 输入/输出数据类型：
   - 当 USINT_TO_BOOL 时，如果输入不等于 0 时输出为 TRUE，如果输入等于 0 时输出为 FALSE；
   - 当 USINT_TO_TIME、USINT_TO_TOD 时，输入将以毫秒值进行转换；
   - 当 USINT_TO_DATE、USINT_TO_DT 时，输入将以秒值进行转换。

3. 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
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*）
功能块（FBD）	

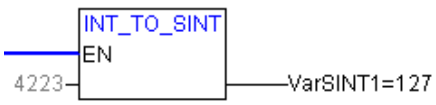
2.9.7 INT_TO_<TYPE>（整数类型转换指令）

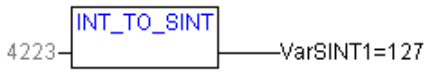
1. 功能：把整型数据类型转换为其它数据类型。
2. 输入/输出数据类型：
   - 当 INT_TO_BOOL 时，如果输入不等于 0 时输出为 TRUE，如果输入等于 0 时输出为 FALSE；
   - 当 INT_TO_TIME、INT_TO_TOD 时，输入将以毫秒值进行转换；
   - 当 INT_TO_DATE、INT_TO_DT 时，输入将以秒值进行转换。
3. 指令使用举例

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
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）。

功能块（FBD）	

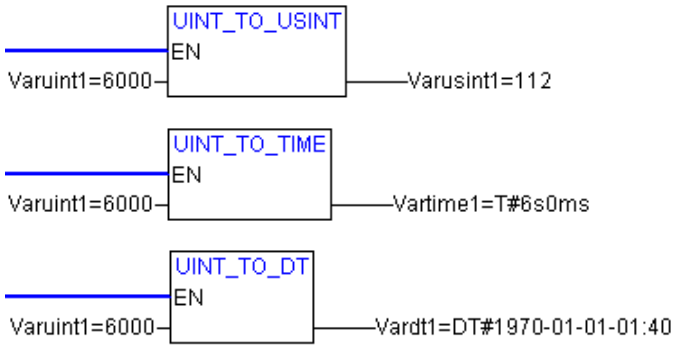
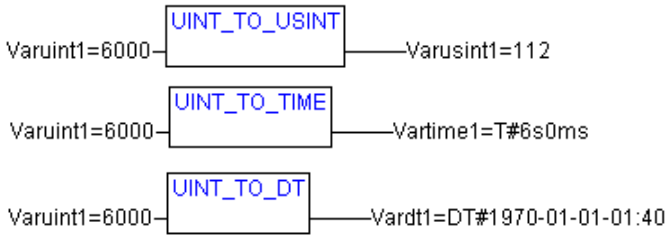
2.9.8 UINT_TO_<TYPE>（无符号整数类型转换指令）

1. 功能：无符号整数类型转换为其它数据类型。
2. 输入/输出数据类型：
   - 当 UINT_TO_BOOL 时，如果输入不等于 0 时输出为 TRUE，如果输入等于 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。
3. 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**
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);
功能块（FBD）	

2.9.9 DINT/LINT_TO<TYPE>（双整数或长整数类型转换指令）

1. 功能：双整数或长整数类型类型转换为其它数据类型。
2. 输入/输出数据类型：
   - 当 DINT/LINT_TO_BOOL 时，如果输入不等于 0 时输出为 TRUE，如果输入等于 0 时输出为 FALSE；
   - 当 DINT/LINT_TO_TIME、DINT_TO_TOD 时，输入将以毫秒值进行转换；
   - 当 DINT/LINT_TO_DATE、DINT_TO_DT 时，输入将以秒值进行转换。
3. 指令使用举例

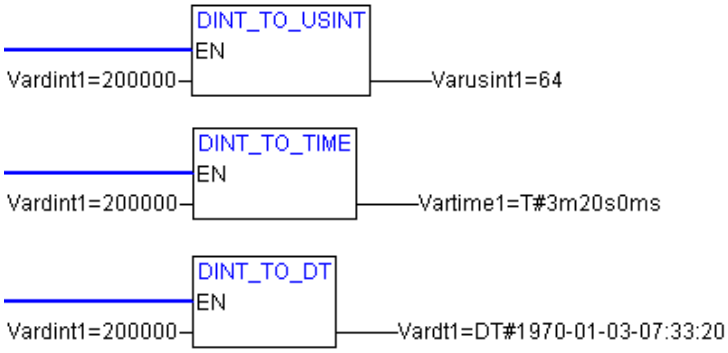
变量定义

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**

梯形图（LD）



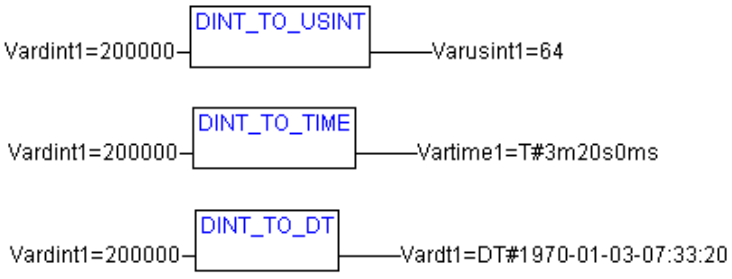
结构化文本（ST）

```
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 *）
```

功能块（FBD）



2.9.10 UDINT/ULINT_TO_<TYPE>（无符号双整数或长整数类型转换指令）

1. 功能：无符号双整数或无符号长整数类型转换为其它数据类型。
2. 输入/输出数据类型：
   - 当 UDINT/ULINT_TO_BOOL 时，如果输入不等于 0 时输出为 TRUE，如果输入等于 0 时输出为 FALSE；
   - 当 UDINT/ULINT_TO_TIME、UDINT_TO_TOD 时，输入将以毫秒值进行转换；
   - 当 UDINT/ULINT_TO_DATE、UDINT_TO_DT 时，输入将以秒值进行转换。
3. 指令使用举例

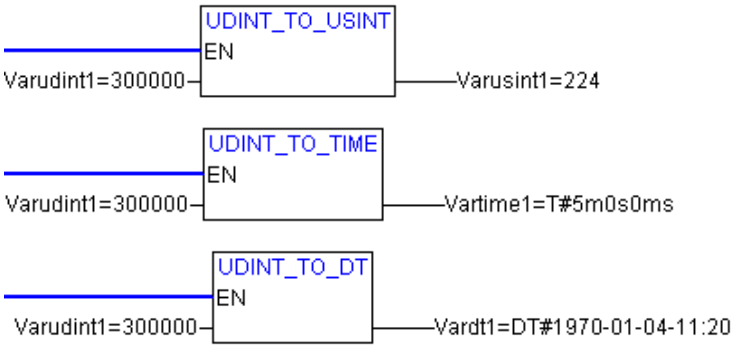
变量定义

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**

梯形图（LD）



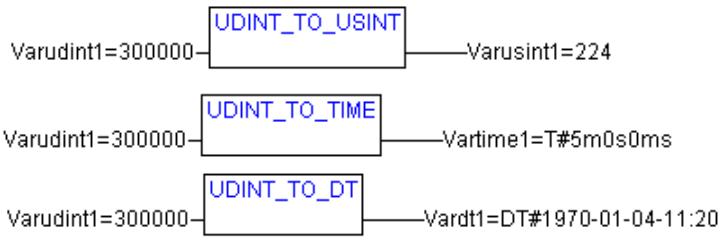
结构化文本（ST）

```
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 *）
```

功能块（FBD）



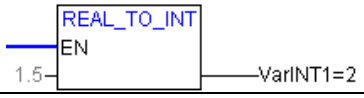
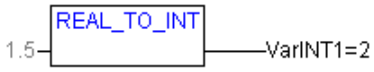
2.9.11 REAL/LREAL_TO_<TYPE>（实数类型转换指令）

1. 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。
2. 输入/输出数据类型：
   - 当 REAL/LREAL_TO_BOOL 时，如果输入不等于 0 时输出为 TRUE，如果输入等于 0 时输出为 FALSE；
   - 当 REAL/LREAL_TO_TIME、REAL_TO_TOD 时，输入将以毫秒值进行转换；
   - 当 REAL/LREAL_TO_DATE、REAL_TO_DT 时，输入将以秒值进行转换。
3. 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
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*）
功能块（FBD）	

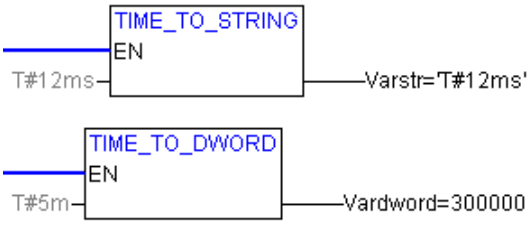
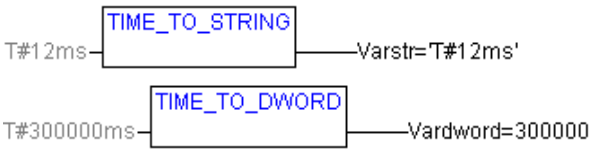
2.9.12 TIME_TO_<TYPE>（时间类型转换指令）

1. 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）。
2. 输入/输出数据类型：当 TIME_TO_BOOL 时，如果输入不等于 0 时输出为 TRUE，如果输入等于 0 时输出为 FALSE。
3. 指令使用举例

变量定义

VAR | VAR_INPUT | VAR_OUTPUT | VAR_IN_OUT | CON

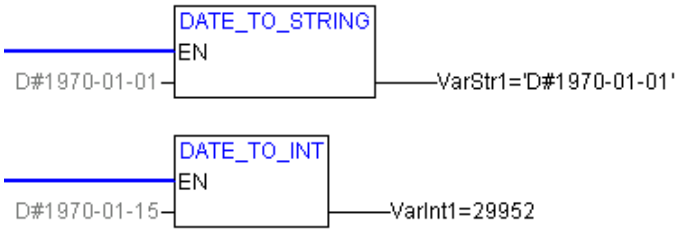
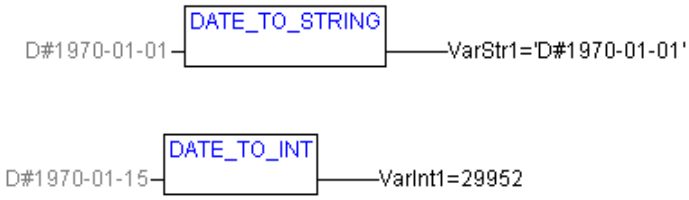
	名称	地址	类型	初始值	注释
0001	Varstr		STRING		
0002	Vardword		DWORD		

编程语言	程　序
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*）
功能块（FBD）	

2.9.13　DATE_TO_<TYPE>（日期类型转换指令）

1. 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。
2. 输入/输出数据类型：当 DATE_TO_BOOL 时，如果输入不等于 0 时输出为 TRUE，如果输入等于 0 时输出为 FALSE。
3. 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）	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。
功能块（FBD）	

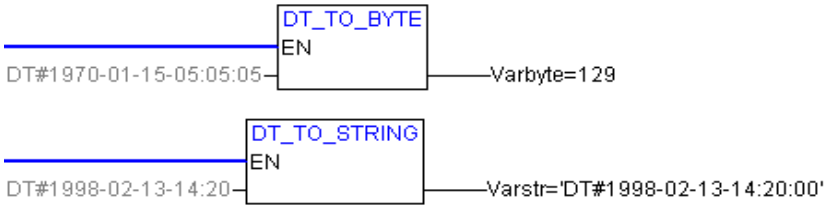
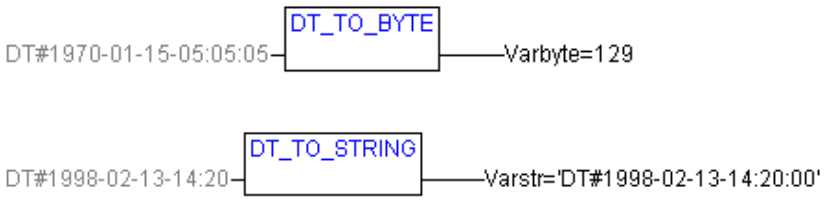
2.9.14 DT_TO_<TYPE>（日期时间类型转换指令）

1. 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。
2. 输入/输出数据类型：当 DT_TO_BOOL 时，如果输入不等于 0 时输出为 TRUE，如果输入等于 0 时输出为 FALSE。
3. 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	Varbyte		BYTE		
0002	VarStr		STRING		

编程语言	程　序
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’*）
功能块（FBD）	

2.9.15 TOD_TO_<TYPE>（时间类型转换指令）

1. 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。
2. 输入/输出数据类型：当 TOD_TO_BOOL 时，如果输入不等于 0 时输出为 TRUE，如果输入等于 0 时输出为 FALSE。

3. 指令使用举例

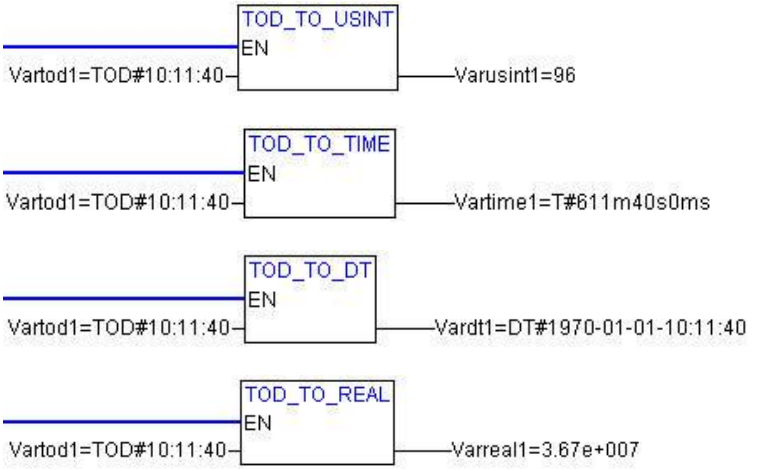
变量定义

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**

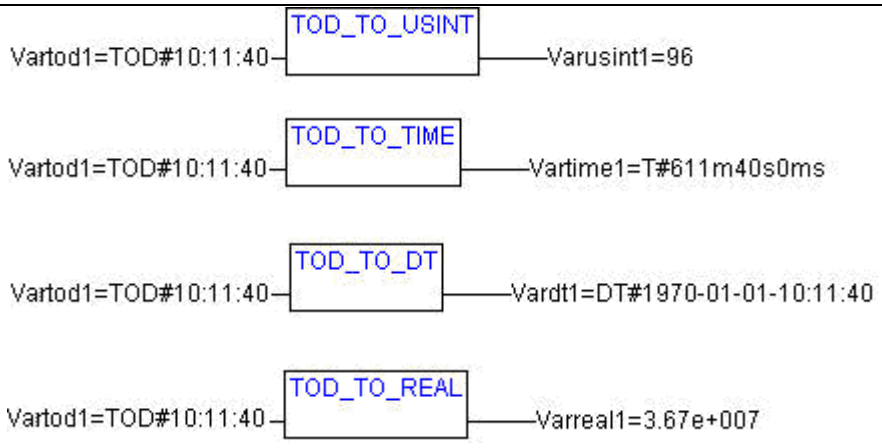
梯形图（LD）



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*）
```

功能块（FBD）



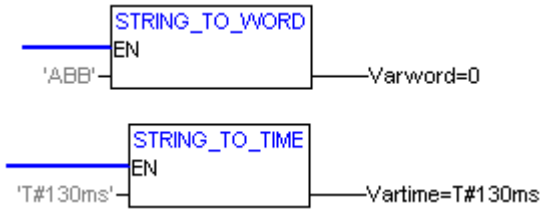
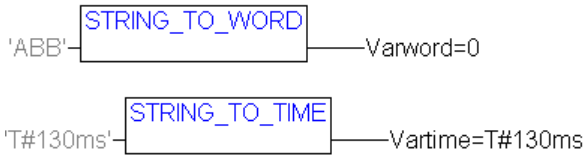
2.9.16 STRING_TO_<TYPE>（字符类型转换指令）

1. 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。
2. 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
梯形图（LD）	
结构化文本（ST）	Varword:=STRING_TO_WORD('ABB');　（*结果为 0*） Vartime:=STRING_TO_TIME('T#130ms');　（*结果为 T#130ms*）
功能块（FBD）	

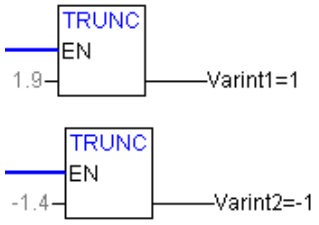
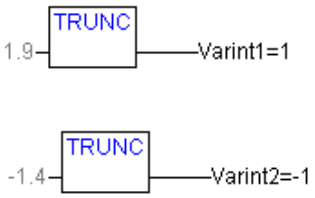
2.9.17 TRUNC（截短转换指令）

1. 功能：截去输入数据的小数部分，只保留整数部分。
2. 输入变量类型：REAL 型。
3. 输出变量类型：INT、WORD、DWORD 型。
4. 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		

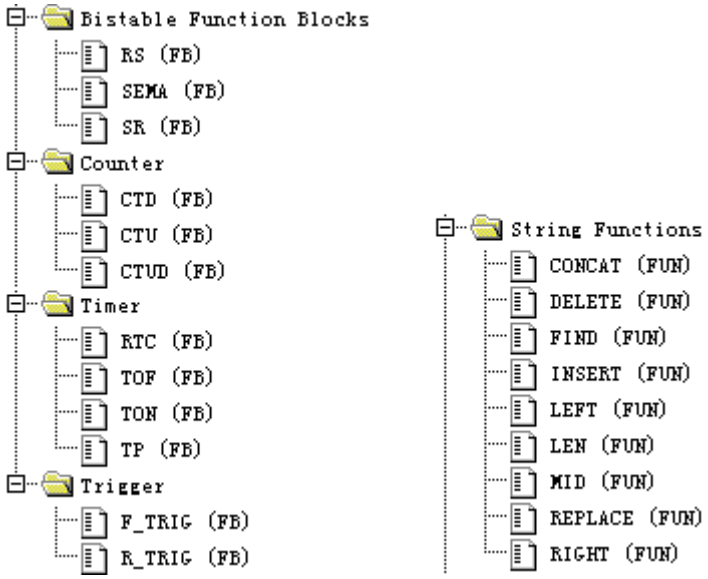
编程语言	程　序
梯形图（LD）	
结构化文本（ST）	Varint1:=TRUNC(1.9);　　（*结果 Varint1 为 1*） Varint2:=TRUNC(-1.4);　　（*结果 Varint2 为-1*）
功能块（FBD）	

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。

第3章　　CoDeSys 标准库指令

Standard.lib 称为 CoDeSys 标准库，在工程建立时自动添加，无需用户手动添加，主要包括字符串处理、触发器、计数器、定时器和字符串处理指令，如下图所示。



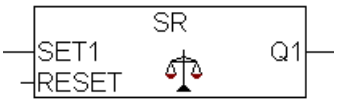
CoDeSys 标准库指令中字符串处理指令属于功能指令（以 FUN 标注），其他为功能块指令（以 FB 标注）。

> LD 编程语言环境中功能指令只能通过“带 EN 的框”进行调用。

3.1 Bistable FB（双稳态指令）

3.1.1　SR（置位优先双稳态器）

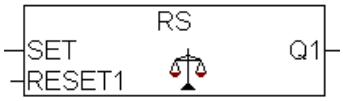
1. 功能：置位双稳态触发器，置位优先。
2. 类型：标准功能块（FB，含历史数据）。
3. 逻辑关系：Q1=(NOT RESET AND Q1) OR SET1
   SET1 为置位信号，RESET 为复位信号。
4. 输入/输出变量类型：均为 BOOL 型。
5. 真值表



SET1	RESET	Q1
0	0	保持原状态
1	0	1
0	1	0
1	1	1

3.1.2　RS（复位优先双稳态器）

1. 功能：复位双稳态触发器，复位优先。
2. 类型：标准功能块（FB，含历史数据）。



3. 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。
4. 输入/输出变量类型：均为 BOOL。
5. 真值表

SET	RESET1	Q1
0	0	保持原状态
1	0	1
0	1	0
1	1	0

3.1.3 SEMA（软件信号灯）

1. 功能：与 SR 相近，输入 CLAIM 相当于 S，输入 RELEASE 相当于 R。区别在于 SEMA 具有中间变量 X，含有历史数据。
2. 类型：标准功能块（FB，含历史数据）。
3. 逻辑关系：BUSY:=X;
   IF CLAIM THEN X:=TRUE;
   ELSE IF RELEASE THEN BUSY:=FALSE; X:=FALSE;

   X 为中间变量，初始值为 FALSE。
4. 输入/输出变量类型：均为 BOOL。
5. 真值表

CLAIM	RELEASE	BUSY
0	0	保持原状态
1	0	1
0	1	0
1	1	1

3.2 Counter（计数器）

3.2.1 CTU（递增计数器）

1. 功能：递增计数器。
2. 类型：标准功能块（FB，含历史数据）。
3. 参数说明

输入参数	数据类型	描述
CU	BOOL	计数输入。CU 每出现从 FALSE 到 TRUE 的上升沿，CV 加 1
RESET	BOOL	初始化。当 RESET 为 TRUE 时，CTU 被重新初始化为 0
PV	WORD	计数器设定值。0~65535
输出参数	**数据类型**	**描述**
Q	BOOL	计数标志输出。当 CV 等于或大于设定值 PV 时，Q 为 TRUE
CV	WORD	当前计数值

3.2.2 CTD（递减计数器）

1. 功能：递减计数器。
2. 类型：标准功能块（FB，含历史数据）。
3. 参数说明

输入参数	数据类型	描述
CD	BOOL	计数输入。CD 每出现从 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**	**描述**
Q	BOOL	计数标志输出。当 CV 等于 0 时，Q 为 TRUE
CV	WORD	当前计数值

3.2.3 CTUD（递增递减计数器）

1. 功能：递增递减计数器。
2. 类型：标准功能块（FB，含历史数据）。
3. 参数说明

输入参数	数据类型	描述
CU	BOOL	计数输入。CU 每出现从 FALSE 到 TRUE 的上升沿，CV 加 1
CD	BOOL	计数输入。CD 每出现从 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**	**描述**
QU	BOOL	计数标志输出。当 CV 等于或大于 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

3.3 Timer（定时器）

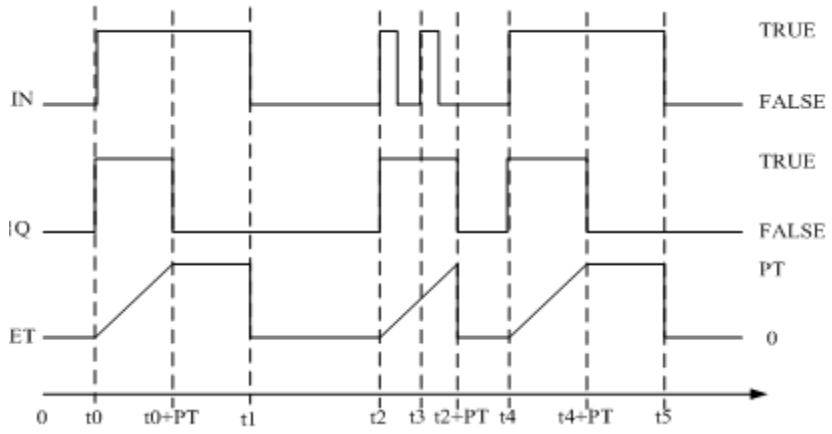
3.3.1 TP（脉冲定时器）

1. 功能：脉冲定时器。
2. 类型：标准功能块（FB，含历史数据）。

3. 参数说明

输入参数	数据类型	描述
IN	BOOL	若 IN 是 FALSE，Q 是 FALSE，ET 为 0。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**	**描述**
Q	BOOL	当 IN 为 TRUE 并且 ET 小于或等于 PT 时，Q 为 TRUE，反之 Q 是 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

4. 时序图

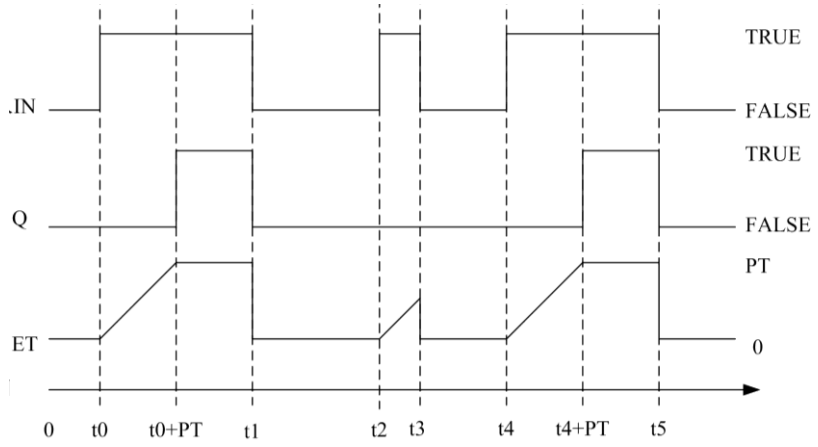


3.3.2 TON（延时导通定时器）

1. 功能：延时导通定时器。
2. 类型：标准功能块（FB，含历史数据）。
3. 参数说明

输入参数	数据类型	描述
IN	BOOL	若 IN 是 FALSE，Q 则是 FALSE，ET 则为 0。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**	**描述**
Q	BOOL	当 IN 为 TURE 并且 ET 等于 PT 时，Q 为 TRUE。反之 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。无论何时，当 IN 为 FALSE 时，ET 等于 0

4. 时序图

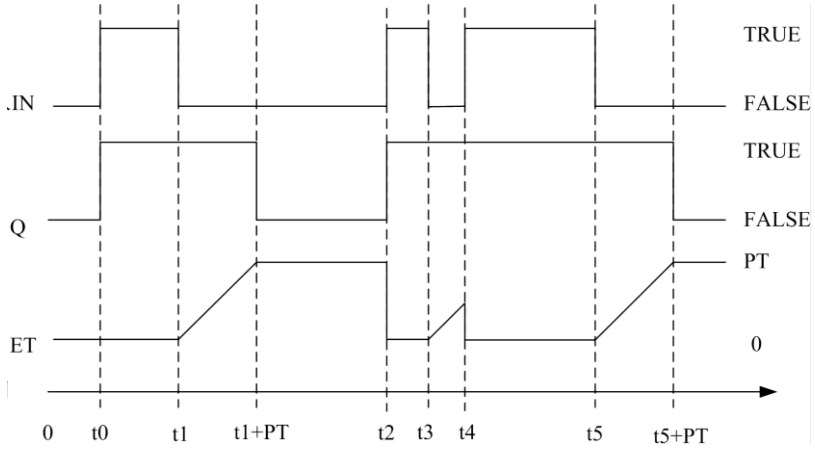


3.3.3 TOF（延时断开定时器）

1. 功能：延时断开定时器。
2. 类型：标准功能块（FB，含历史数据）。
3. 参数说明

输入参数	数据类型	描述
IN	BOOL	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**	**描述**
Q	BOOL	当 IN 为 FALSE 且 ET 等于 PT 时，Q 为 FALSE。反之 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

4. 时序图



3.3.4 RTC（实时时钟）

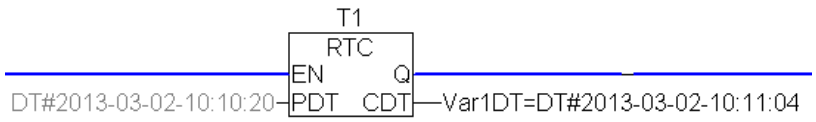
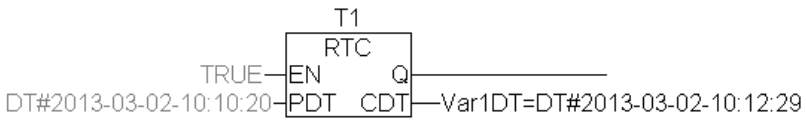
1. 功能：在给定时间启动，返回当前日期和时间。
2. 类型：标准功能块（FB，含历史数据）。

3. 参数说明

输入参数	数据类型	描述
EN	BOOL	使能功能块　FALSE：无效 TRUE：上升沿使能，高电平有效
PDT	DT	时间基值
输出参数	**数据类型**	**描述**
Q	BOOL	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 的时间开始计时，单位为秒。PDT 的时间仅在 EN 的上升沿设定

4. 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）	T1(PDT:=DT#2013-03-02-10:10:20); DT1:=RTCInst.CDT;
功能块（FBD）	

3.4 Trigger（触发器指令）

3.4.1　R_TRIG（上升沿触发器）

1. 功能：用于检测上升沿。
2. 类型：标准功能块（FB，含历史数据）。
3. 输入/输出变量类型：输入 CLK 和输出 Q 均为 BOOL 类型。
4. 逻辑关系：Q:=CLK AND NOT M;
   M:=CLK;

M 为中间变量，初始值为 FALSE。

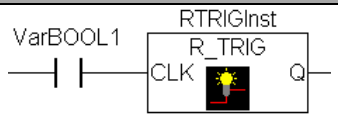
当 CLK 检测到上升沿时，Q 输出 TRUE，并维持一个扫描周期。

5. 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;

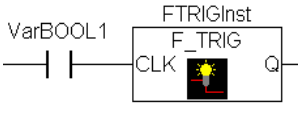
3.4.2　F_TRIG（下降沿触发器）

1. 功能：用于检测下降沿。
2. 类型：标准功能块（FB，含历史数据）。
3. 逻辑关系：Q:=NOT CLK AND NOT M;
   M:=NOT CLK;
   M 为中间变量，初始值为 FALSE。
   当 CLK 检测到下降沿时，Q 输出 TRUE，并维持一个扫描周期。
4. 输入/输出变量类型：输入 CLK 和输出 Q 均为 BOOL 类型。
5. 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;

3.5 String Functions（字符串处理指令）

3.5.1 CONCAT（合并字符串指令）

1. 功能：将两个字符串按前后顺序结合成一个字符串。
2. 类型：标准功能（FUN）。
3. 输入和返回值都是 STRING 型。

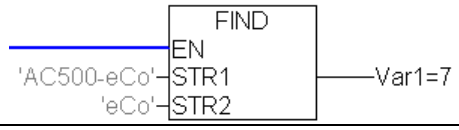
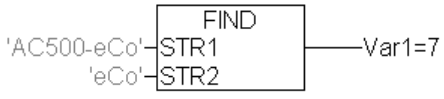
3.5.2 DELETE（删除字符指令）

1. 功能：从字符串中删除字符。
2. 类型：标准功能（FUN）。
3. 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 类型，该指令从输入字符的位置 POS 处开始向右删除 LEN 个字符。返回值为 STRING 类型。

3.5.3 FIND（查找字符串指令）

1. 功能：在一个字符串中查找与另一字符串完全相同的内容。
2. 类型：标准功能（FUN）。
3. 指令格式：FIND(STR1,STR2)。输入 STR1 和 STR2 都是 STRING 类型，为输入字符串。该指令在第一个字符串 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。返回值为 INT 类型。
4. 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:=FIND ('AC500-eCo', 'eCo');　（*结果为 7*）
功能块（FBD）	

3.5.4 INSERT（插入字符串指令）

1. 功能：将一个字符串插入到另一个字符串中。
2. 类型：标准功能（FUN）。
3. 指令格式：INSERT(STR1,STR2,POS)。输入 STR1 和 STR2 是 STRING 类型，POS 是 INT 类型，该指令将 STR2 插入到 STR1 的 POS 位置之后。返回值为 STRING 类型。

3.5.5 LEN（取字符串长度指令）

1. 功能：计算字符串的长度。
2. 类型：标准功能（FUN）。
3. 输入为 STRING 类型，返回值为 INT 类型。

3.5.6 LEFT（左边获取字符串指令）

1. 功能：从字符串左边获取字符串。
2. 类型：标准功能（FUN）。
3. 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 类型，为从输入字符串左边开始获取的字符个数。返回值为 STRING 类型。

3.5.7 MID（获取字符串指令）

1. 功能：从字符串指定位置获取字符串。
2. 类型：标准功能（FUN）。
3. 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 类型，该指令从 POS 开始从左往右获取 LEN 个字符，返回值为 STRING 类型。

3.5.8 REPLACE（替换字符串指令）

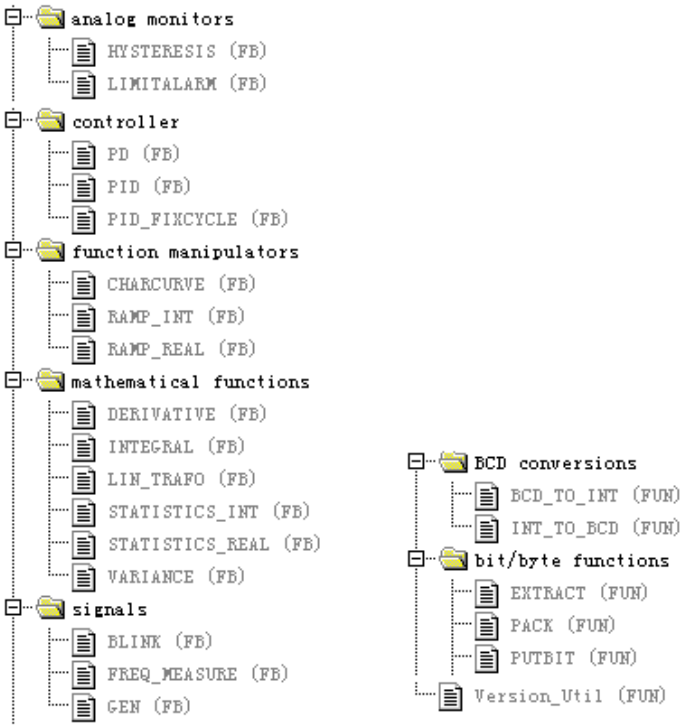
1. 功能：用一个字符串替代另一字符串中的部分内容。
2. 类型：标准功能（FUN）。
3. 指令格式：REPLACE(STR1,STR2,L,P)。输入 STR1 和 STR2 是 STRING 类型，为输入字符串。L 和 P 是 INT 类型，该指令用 STR2 代替 STR1 中从 P 位置开始向右的 L 个字符。返回值为 STRING 类型。

3.5.9 RIGHT（右边获取字符串指令）

1. 功能：从字符串右边获取字符串。
2. 类型：标准功能（FUN）。
3. 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 类型，为从输入字符串右边开始获取的字符个数。返回值为 STRING 类型。

第4章　　CoDeSys 应用库指令

Util.lib 称为 CoDeSys 应用库，主要包括 BCD 码转换、信号发生器、高等数学运算、控制器等指令，如下图所示。



CoDeSys 应用库指令中 BCD 码转换指令和位/字节操作指令属于功能指令，其他指令为功能块指令。

> LD 编程语言环境中功能指令只能通过“带 EN 的框”进行调用。

4.1 analog monitor（模拟量监视指令）

4.1.1　HYSTERESIS（滞后）

1. 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，并保持至 IN 大于上限值 HIGH 才变为 FALSE。OUT 保持 FALSE 至 IN 小于下限值 LOW，变为 TRUE，如此循环。
2. 类型：标准功能块（FB）。
3. 参数说明

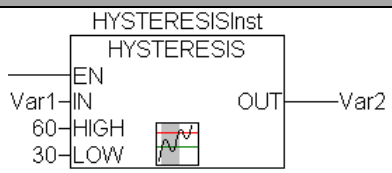
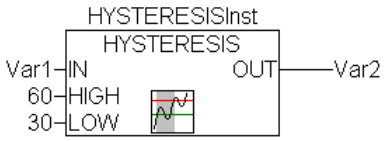
输入参数	数据类型	描述
IN	INT	输入值
HIGH	INT	上限值
LOW	INT	下限值
输出参数	**数据类型**	**描述**
OUT	BOOL	输出值。低于下限值后为 TRUE，直到上限值后为恢复 FALSE

4. 指令使用举例

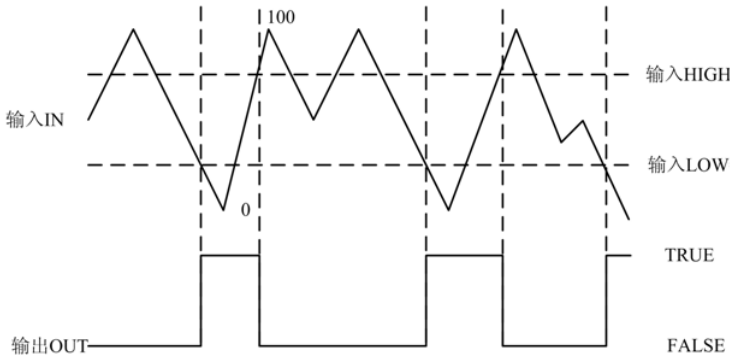
变量定义

VAR | VAR_INPUT | VAR_OUTPUT | VAR_IN_

	名称	地址	类型	初值
0001	Var1		INT	
0002	Var2		BOOL	
0003	HYSTERESISInst		HYSTERESIS	

编程语言	程　序
梯形图（LD）	
结构化文本（ST）	HYSTERESISInst(IN:=Var1, HIGH:=60, LOW:=30); Var2:=HYSTERESISInst.OUT;
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所示。



LD 语言环境中添加该指令应通过“带 EN 的框”调用。

4.1.2 LIMITALARM（上下限报警）

1. 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。
2. 类型：标准功能块（FB）。
3. 参数说明

输入参数	数据类型	描述
IN	INT	输入值
HIGH	INT	上限值
LOW	INT	下限值

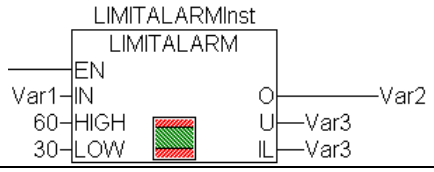
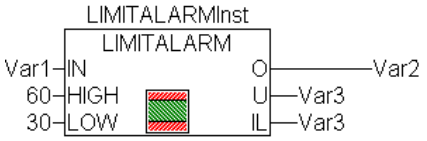
输出参数	数据类型	描述
O	BOOL	输出值
U	BOOL	输出值
IL	BOOL	输出值

4. 指令使用举例

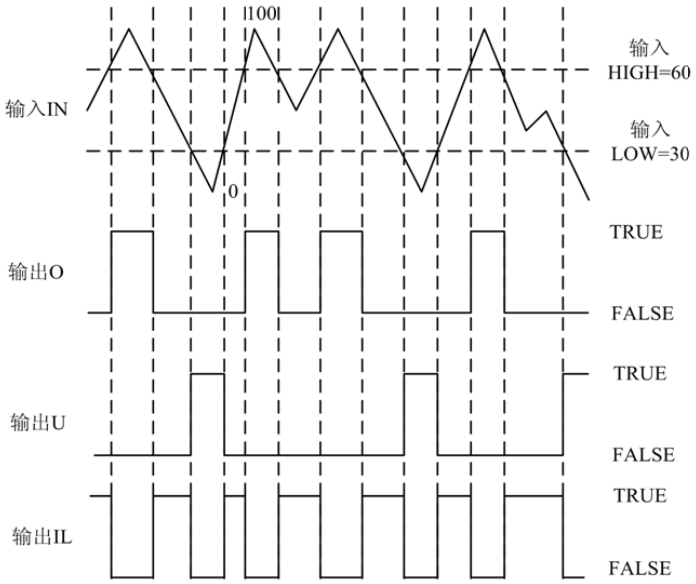
变量定义

VAR | VAR_INPUT | VAR_OUTPUT | VAR_IN_

	名称	地址	类型	初值
0001	LIMITALARMInst		LIMITALARM	
0002	Var1		INT	
0003	Var2		BOOL	
0004	Var3		BOOL	
0005	Var4		BOOL	

编程语言	程　序
梯形图（LD）	
结构化文本（ST）	LIMITALARMInst(IN:=Var1, HIGH:=60, LOW:=30); Var2:=LIMITALARMInst.U; Var3:=LIMITALARMInst.IL; Var1:=LIMITALARMInst.O;
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所示。



LD 语言环境中添加该指令应通过“带 EN 的框”进行调用。

4.2 BCD conversions （BCD 码转换指令）

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。

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。

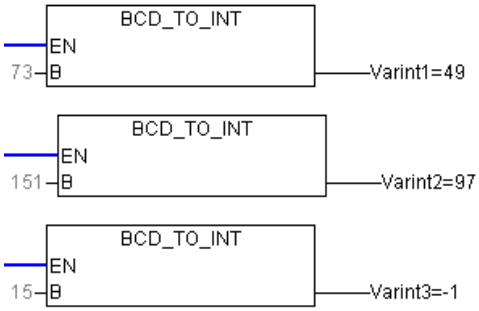
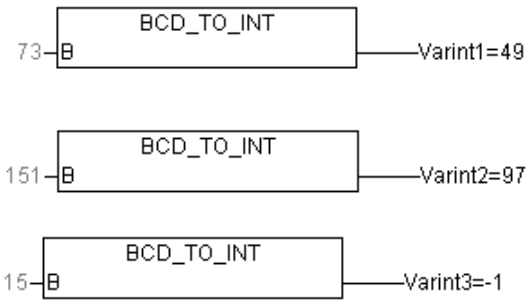
4.2.1 BCD_TO_INT（BCD 码转整型）

1. 功能：该指令将 BCD 码转为 INT 值。
2. 类型：标准功能（FUN）。
3. 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）。
4. 返回值：INT 型，该 BCD 码所代表的实际值，如果输入的字节不是 BCD 码，输出为-1。
5. 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*）
功能块（FBD）	

4.2.2 INT_TO_BCD（整型转 BCD 码）

1. 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，输出为 255。
2. 类型：标准功能（FUN）。
3. 输入 I：INT 型，如果整数值为 49，则此处输入整型数据 49。
4. 返回值：BYTE 型，转换完的 BCD 码的值。
5. 指令使用举例

变量定义

VAR | VAR_INPUT | VAR_OUTPUT | VAR_IN_OUT

	名称	地址	类型	初值	注释
0001	Var1		BYTE		
0002	Var2		BYTE		

编程语言	程　序
梯形图（LD）	INT_TO_BCD, EN, 49, Var1=73 INT_TO_BCD, EN, 100, Var2=255
结构化文本（ST）	Var1:=INT_TO_BCD(49);　（*结果为 73*） Varb2:=INT_TO_BCD(100);　（*错误！输出：255*）
功能块（FBD）	INT_TO_BCD, 49, Var1=73 INT_TO_BCD, 100, Var2=255

4.3 bit/byte functions（位/字节操作指令）

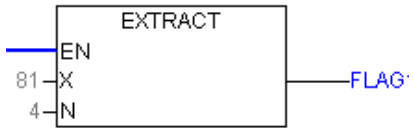
4.3.1 EXTRACT（位提取指令）

1. 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。N 从零位开始计数。
2. 类型：标准功能（FUN）。
3. 输入变量类型：X 是 DWORD 类型，N 是 BYTE 型。
4. 返回值类型：BOOL 类型。
5. 指令使用举例

变量定义

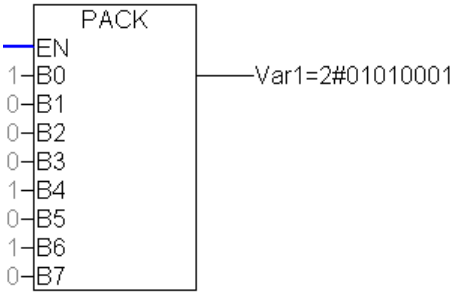
VAR | VAR_INPUT | VAR_OUTPUT | VAR_IN_OUT | CONSTANT

	名称	地址	类型	初始值	注释
0001	FLAG		BOOL		

编程语言	程　序
梯形图（LD）	
结构化文本（ST）	FLAG:=EXTRACT(X:=81, N:=4); （*结果：TRUE，因为 81 的二进制数是 1010001，所以第 4 位是 1*）

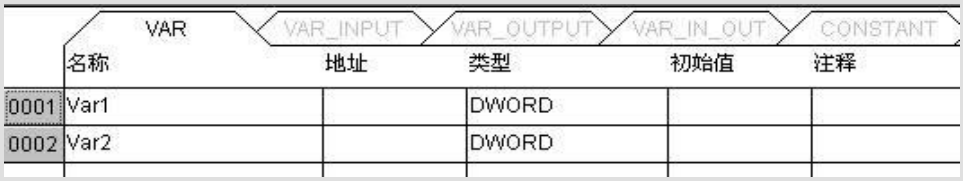
4.3.2　PACK（位整合指令）

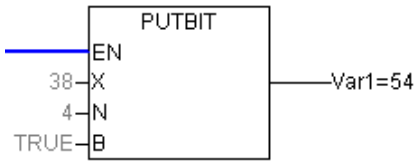
1. 功能：将输入 B0、B1、……、B7 合成为一个字节，与这个指令相对应的指令是 UNPACK。
2. 类型：标准功能（FUN）。
3. 输入变量类型：B0、B1、……、B7 均为 BOOL 类型。
4. 返回值类型：BYTE 类型。
5. 指令使用举例

变量定义	
	
编程语言	程　序
梯形图（LD）	
结构化文本（ST）	Var1:=PACK(1,0,0,0,1,0,1,0);　　（*结果为 2#01010001*）

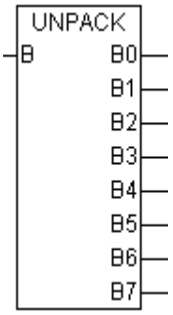
4.3.3　PUTBIT（位赋值指令）

1. 功能：将输入变量 X 值的第 N 位（N=0,1...）赋值为 B，并输出转变后的值。N 从零位开始计数。
2. 类型：标准功能（FUN）。
3. 输入变量类型：变量 X 为 DWORD 类型，N 为 BYTE 类型，B 为 BOOL 类型。
4. 返回值类型：DWORD 类型。
5. 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）	Var2:=38;　　（*二进制 100110*） Var1:=PUTBIT(Var2,4,TRUE);　　（*结果：54 =2#110110*）

4.3.4　UNPACK（位拆分）

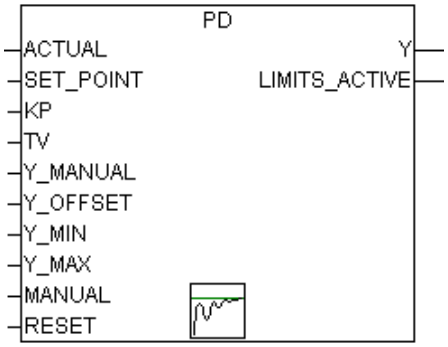


1. 功能：将字节型的输入 B 拆分转换成 8 个 BOOL 类型的输出变量 B0,…,B7，与 PACK 指令相反。
2. 类型：标准功能块（FB）。
3. 输入变量类型：BYTE 类型。
4. 输出变量类型：BOOL 类型。

LD 语言环境中添加该指令应通过“带 EN 的框”进行调用。

4.4 Controller（控制器指令）

4.4.1　PD（比例微分控制器）



1. 功能：该指令为比例微分控制器。
2. 类型：标准功能块（FB）。
3. 公式：$\Delta = \mathrm{SET\_}POINT - ACTUAL$

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

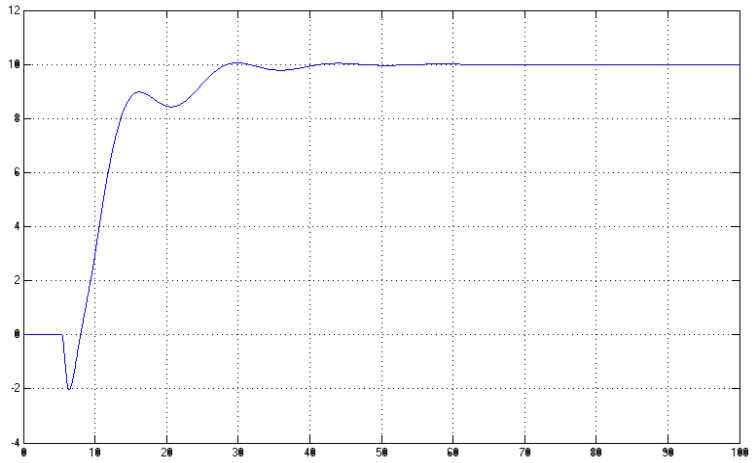
$\frac{\sigma\Delta}{\sigma t}$ 该指令会自动计算 ，用户无需考虑。

4. 参数说明

输入参数	数据类型	描述
ACTUAL	REAL	测量值
SET_POINT	REAL	设定值
KP	REAL	比例系数
TV	REAL	微分时间（s）
Y_MANUAL	REAL	手动输出值。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

输入参数	数据类型	描述
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	复位。TRUE 时重置该控制器，正常运行时应置 FALSE
输出参数	**数据类型**	**描述**
Y	REAL	输出值
LIMITS_ACTIVE	BOOL	输出超限标志。输出值超限时等于 TRUE

调整波形如下图所示。



LD 语言环境中添加该指令应通过“带 EN 的框”进行调用。

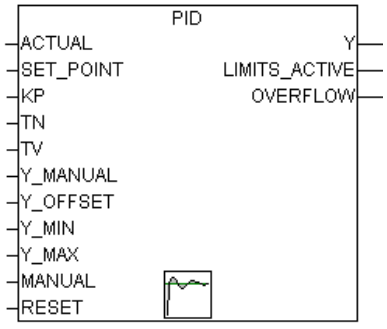
4.4.2 PID（比例积分微分控制器）

1. 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。
2. 类型：标准功能块（FB）。
3. 公式：

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

$$\Delta = SET\_POINT - ACTUAL$$

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



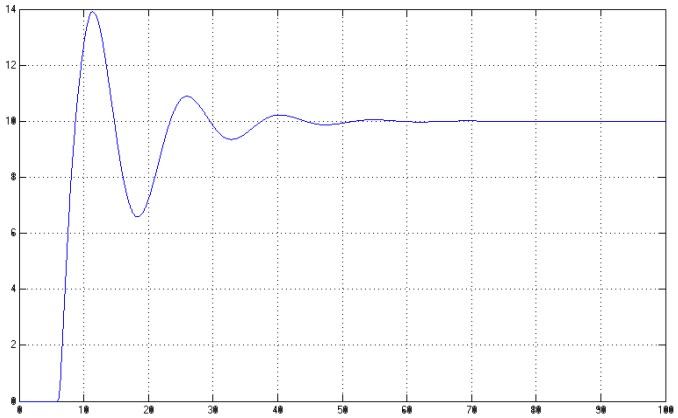
PID 的周期功能块内部自动测量。

4. 参数说明

输入参数	数据类型	描述
ACTUAL	REAL	测量值
SET_POINT	REAL	设定值
KP	REAL	比例系数
TN	REAL	积分时间（s）
TV	REAL	微分时间（s）
Y_MANUAL	REAL	手动输出值。MANUAL=TRUE 时，Y=Y_MANUAL

输入参数	数据类型	描述
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	复位。TRUE 时重置该控制器，正常运行时应置 FALSE
输出参数	**数据类型**	**描述**
Y	REAL	输出值
LIMITS_ACTIVE	BOOL	输出超限标志。输出值超限时等于 TRUE
OVERFLOW	BOOL	输出溢出标志。积分溢出时等于 TRUE，同时，控制器暂停，只有重新初始化时才会被再次激活

调整波形如下图所示。



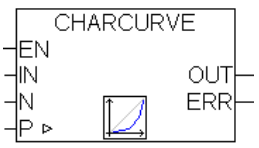
LD 语言环境中添加该指令应通过“带 EN 的框”进行调用。

4.4.3 PID_FIXCYCLE（比例积分微分控制器）

该指令与 PID 控制器对比，只是多一个采样周期的 REAL 型输入参数 CYCLE，用来设定积分和微分的时间步长，单位为秒（PID 的周期功能块内部自动测量）。控制方程、参数说明以及调整波形详见 PID 指令。

4.5 function mainpulators（操纵器指令）

4.5.1 CHARCURVE（特征曲线）



1. 功能：通过输入的 POINT 类型数组 P[0..N-1]，在 XY 坐标图上定义出一条曲线。输入 IN 指定坐标上的 X 轴上的点，输出值 OUT 为坐标上该曲线所对应的 Y 轴的值。
2. 类型：标准功能块（FB）。

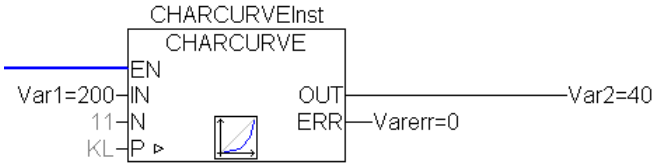
3. 参数说明

输入参数	数据类型	描述
IN	INT	输入坐标上 X 轴的值
N	BYTE	定义曲线所使用数组中的点数（2≤N≤11）
P	ARRAY[0..10] OF POINT	用来在 XY 坐标上定义特性曲线
输出参数	**数据类型**	**描述**
OUT	INT	输出值。坐标上曲线对应的 Y 轴的值
ERR	BYTE	显示信息 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，即超出数组定义曲线的 X 轴的范围 ERR=4：输入 N 小于 2，或者大于 11
P	ARRAY[0..10] OF POINT	N 输出值

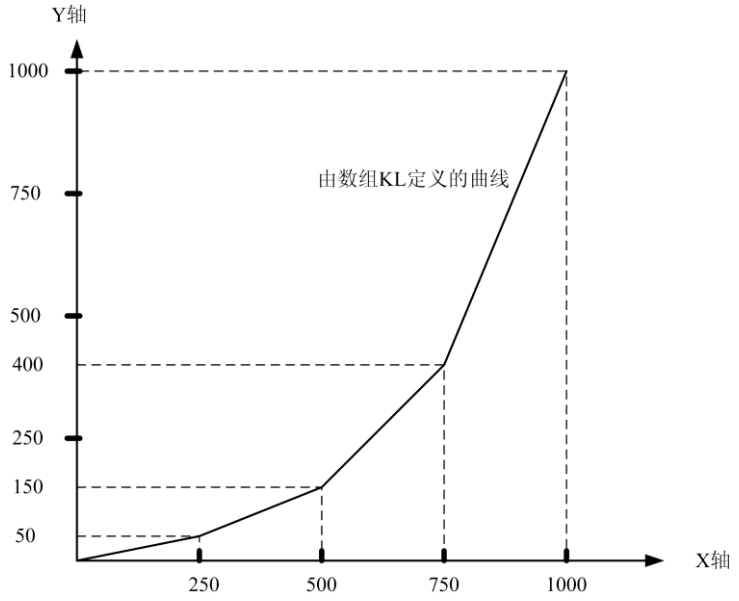
4. 指令使用举例

变量定义

```
CHARCURVEInst: CHARCURVE;
Var1:INT;
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));
Var2: INT;
Varerr: BYTE;
```

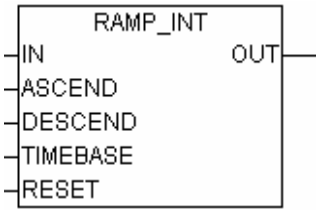
编程语言	程　序
梯形图（LD）	
结构化文本（ST）	CHARCURVEInst(IN:=Var1, N:=11, P:=KL); Varerr:=CHARCURVEInst.ERR; Var2:=CHARCURVEInst.OUT;　　（*当 Var1=200 时，Var2=40*）

上例中，根据数组 KL 确定的特性曲线如下图所示。输入 IN 为 X 轴上的值，输出 OUT 为该曲线对应的 Y 轴上的值。



LD 语言环境中添加该指令应通过“带 EN 的框”进行调用。

4.5.2 RAMP_INT（整型限速）



1. 功能：限制整型输入数据的升降速度。
2. 类型：标准功能块（FB）。
3. 参数说明

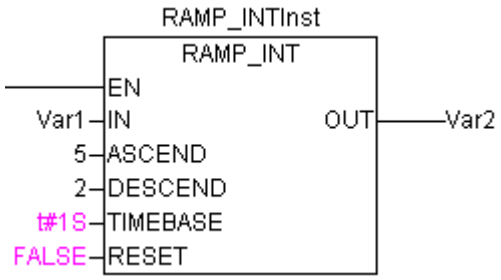
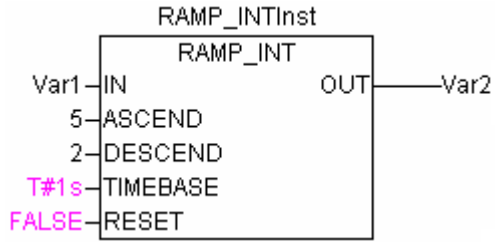
输入参数	数据类型	描述
IN	INT	输入值。若当前输入值大于先前值，则按照设定的上升速率进行加法运算，由 OUT 即时输出，若当前设定值小于先前值，则按照设定的下降速率进行减法运算，由 OUT 输出
ASCEND	INT	最大上升的增量。在规定时间（TIMEBASE）内最大上升的数量
DESCEND	INT	最大下升的减量。在规定时间（TIMEBASE）内最大下降的数量
TIMEBASE	TIME	时间基数
RESET	BOOL	复位。当 RESET=TRUE 时，重新启动指令
输出参数	**数据类型**	**描述**
OUT	INT	输出数据

4. 指令使用举例

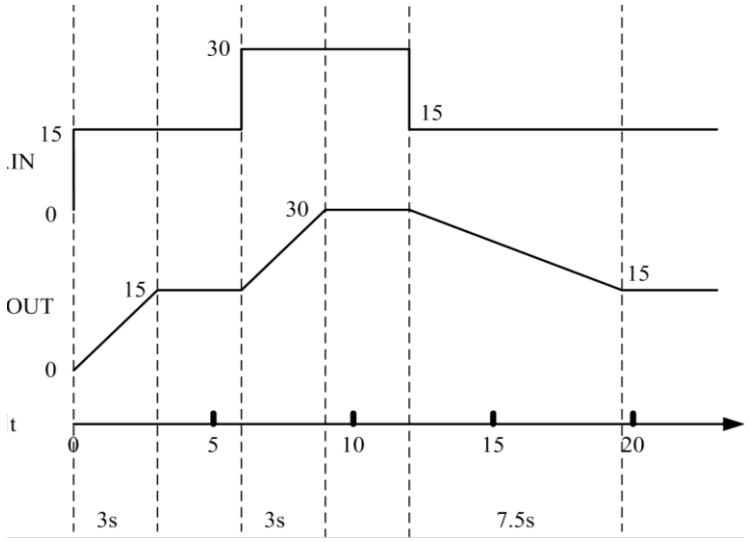
变量定义

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;
功能块（FBD）	

上例中，当指令执行时，根据输入值 IN 的变化，对应的输出值如下图所示。



LD 语言环境中添加该指令应通过“带 EN 的框”进行调用。

4.5.3　RAMP_REAL（实型限速）

1. 功能：限制实型输入数据的升降速度。
2. 类型：标准功能块（FB）。

3. 参数说明

输入参数	数据类型	描述
IN	REAL	输入值。若当前输入值大于先前值，则按照设定的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，则按照设定的下降速率进行减法运算，由 OUT 输出
ASCEND	REAL	最大上升的增量。在规定时间（TIMEBASE）内最大上升的数量
DESCEND	REAL	最大下升的减量。在规定时间（TIMEBASE）内最大下降的数量
TIMEBASE	TIME	时间基数
RESET	BOOL	复位。当 RESET=TRUE 时，重新启动指令
输出参数	**数据类型**	**描述**
OUT	REAL	输出数据

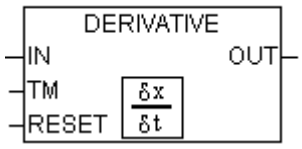
指令使用举例及输出图请参见 RAMP_INT 指令所述。

4.6 mathematical functions（高等数学运算指令）

4.6.1 DERIVATIVE（微分）

1. 功能：该指令对连续输入的变量进行微分运算。为了获得最好的结果，DERIVATIVE 指令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。
2. 类型：标准功能块（FB）。
3. 微分迭代公式：

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

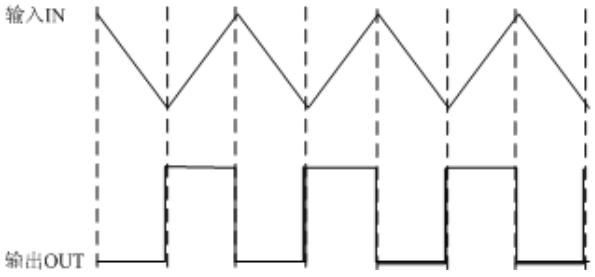


k-3、k-2、k-1、k 为连续四次输入值的标记。

4. 参数说明

输入参数	数据类型	描述
IN	REAL	输入连续信号
TM	DWORD	微分时间（ms）
RESET	BOOL	复位信号。当 RESET=TRUE 时，重新启动指令
输出参数	**数据类型**	**描述**
OUT	REAL	输出微分结果

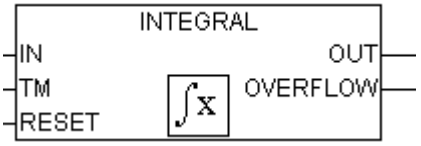
输入输出对应关系如下图所示。



LD 语言环境中添加该指令应通过“带 EN 的框”进行调用。

4.6.2 INTEGRAL（积分）

1. 功能：该指令对连续输入的变量进行积分运算。
2. 类型：标准功能块（FB）。
3. 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。

4. 参数说明

输入参数	数据类型	描述
IN	REAL	输入连续信号
TM	DWORD	积分时间（ms）
RESET	BOOL	复位信号。当 RESET=TRUE 时，重新启动指令
输出参数	**数据类型**	**描述**
OUT	REAL	输出积分结果
OVERFLOW	BOOL	溢出标志。其值是 TRUE 时，说明计算溢出

输入输出对应关系如下图所示。

LD 语言环境中添加该指令应通过“带 EN 的框”进行调用。

4.6.3 LIN_TRAFO（REAL 数据线性转换）

1. 功能：该指令根据设定的上下限范围将输入数据进行线性转换。
2. 类型：标准功能块（FB）。
3. 转换格式：(IN - IN_MIN):(IN_MAX – IN) =(OUT - OUT_MIN):(OUT_MAX – OUT)
4. 参数说明

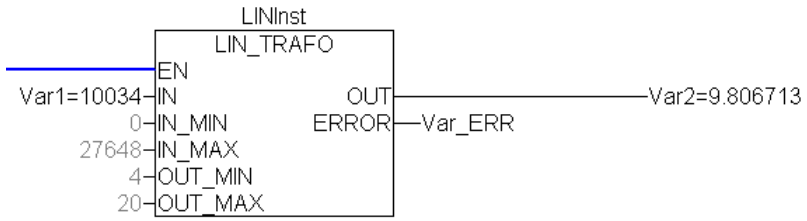
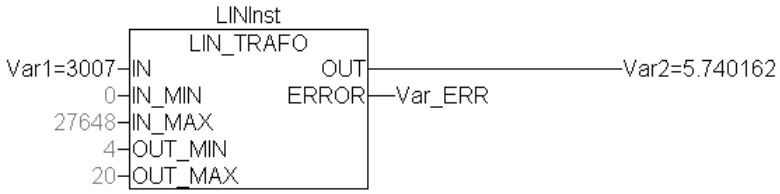
输入参数	数据类型	描述
IN	REAL	输入数据
IN_MIN	REAL	输入下限
IN_MAX	REAL	输入上限
OUT_MIN	REAL	输出下限
OUT_MAX	REAL	输出上限
输出参数	**数据类型**	**描述**
OUT	REAL	转换输出值
ERROR	BOOL	错误标志　　FALSE：无错误 TRUE：错误，如输入超限

5. 指令使用举例

变量定义

VAR | VAR_INPUT | VAR_OUTPUT | VAR_IN_OUT

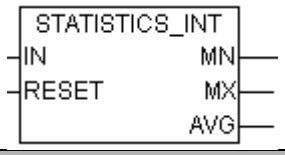
	名称	地址	类型	初值	注释
0001	LINInst		LIN_TRAFO		
0002	Var2		REAL		
0003	Var1		REAL		
0004	Var_ERR		BOOL		

编程语言	程　序
梯形图（LD）	
结构化文本（ST）	LINInst (in:=Var1,IN_MIN:=0,IN_MAX:=27648,OUT_MIN:=4,OUT_MAX:=20); Var2:=LINInst.OUT; Var_ERR:=LINInst.ERROR;
功能块（FBD）	

LD 语言环境中添加该指令应通过“带 EN 的框”进行调用。

4.6.4 STATISTICS_INT（整型统计）

1. 功能：统计整型输入数据的最小值、最大值和平均值。
2. 类型：标准功能块（FB）。
3. 参数说明

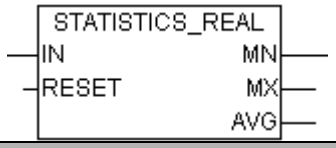


输入参数	数据类型	描述
IN	INT	输入
RESET	BOOL	复位信号。当 RESET=TRUE 时，重新启动指令
输出参数	**数据类型**	**描述**
MN	INT	最小值
MX	INT	最大值
AVG	INT	平均值

LD 语言环境中添加该指令应通过“带 EN 的框”进行调用。

4.6.5 STATISTICS_REAL（实型统计）

1. 功能：统计实型输入数据的最小值、最大值和平均值。该指令与 STATISTICS_INT 功能相同，只是输入和输出数据类型为 REAL 型。
2. 类型：标准功能块（FB）。
3. 参数说明

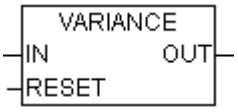


输入参数	数据类型	描述
IN	REAL	输入
RESET	BOOL	复位信号。当 RESET=TRUE 时，重新启动指令
输出参数	**数据类型**	**描述**
MN	INT	最小值
MX	INT	最大值
AVG	INT	平均值

LD 语言环境中添加该指令应通过“带 EN 的框”进行调用。

4.6.6 VARIANCE（平方偏差指令）

1. 功能：该指令计算输入数据的平方偏差。标准偏差可以由平方偏差的平方根得到。
2. 类型：标准功能块（FB）。
3. 参数说明



输入参数	数据类型	描述
IN	REAL	输入
RESET	BOOL	复位信号。当 RESET=TRUE 时，重新启动指令
输出参数	**数据类型**	**描述**
OUT	REAL	平方偏差

LD 语言环境中添加该指令应通过“带 EN 的框”进行调用。

4.7 Singals（信号发生器指令）

4.7.1 BLINK（脉冲信号发生器）

1. 功能：该指令用于产生脉冲信号。如果 ENABLE 被设为 TRUE，脉冲信号发生器开始工作，输出高电平时间由 TIMEHIGH 设定，输出低电平时间由 TIMELOW 设定。
2. 类型：标准功能块（FB）。

3. 参数说明

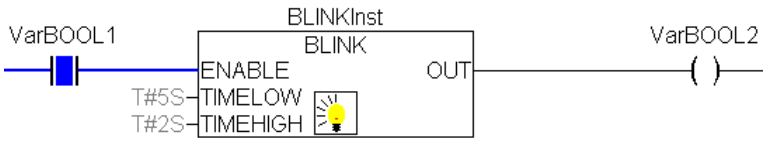
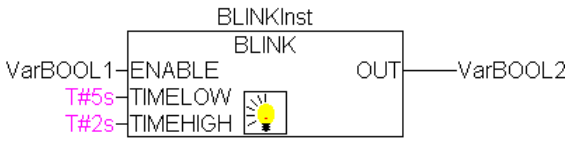
输入参数	数据类型	描述
ENABLE	BOOL	使能。当 ENABLE 为 TRUE 时，指令开始工作 当 ENABLE 为 FLASE 时，OUT 保持前一个周期的状态
TIMELOW	TIME	输出低电平时间
TIMEHIGH	TIME	输出高电平时间
输出参数	**数据类型**	**描述**
OUT	BOOL	脉冲信号输出值

4. 指令使用举例

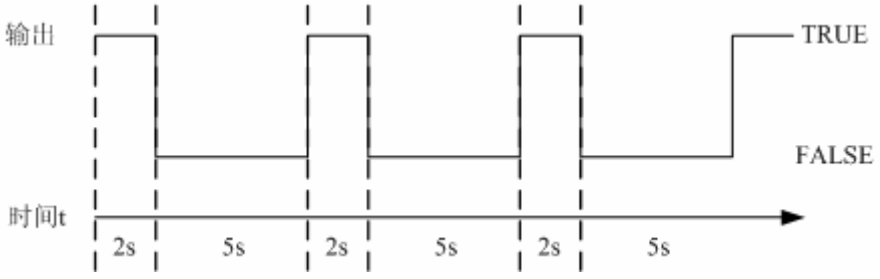
变量定义

VAR | VAR_INPUT | VAR_OUTPUT | VAR_IN_OUT

	名称	地址	类型	初值	注释
0001	VarBOOL1		BOOL		
0002	VarBOOL2		BOOL		
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;
功能块（FBD）	

当指令执行时，Varbool1 输出波形如下。



该指令输出的开始工作后首先固定输出高电平。

使能端断开，输出将保持。

4.7.2 GEN（典型周期信号发生器）

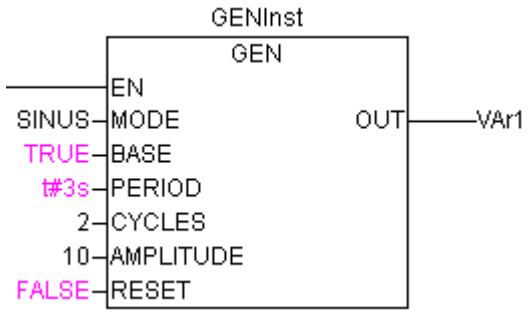
1. 功能：该指令用于生成典型的周期信号，如：三角波、零起点三角波、上升锯齿波、下降锯齿波、方波、正弦波和余弦波共七种。
2. 类型：标准功能块（FB）。

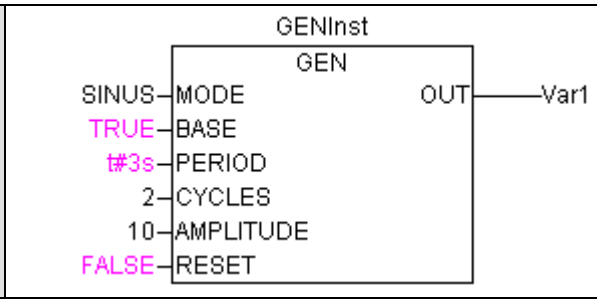
3. 参数说明

输入参数	数据类型	描述	
MODE	GEN_MODE	信号类型。可选择输入以下波形	
		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	复位信号。当 RESET=TRUE 时，信号发生器被重新设置为 0	
输出参数	**数据类型**	**描述**	
OUT	INT	波形信号输出值	

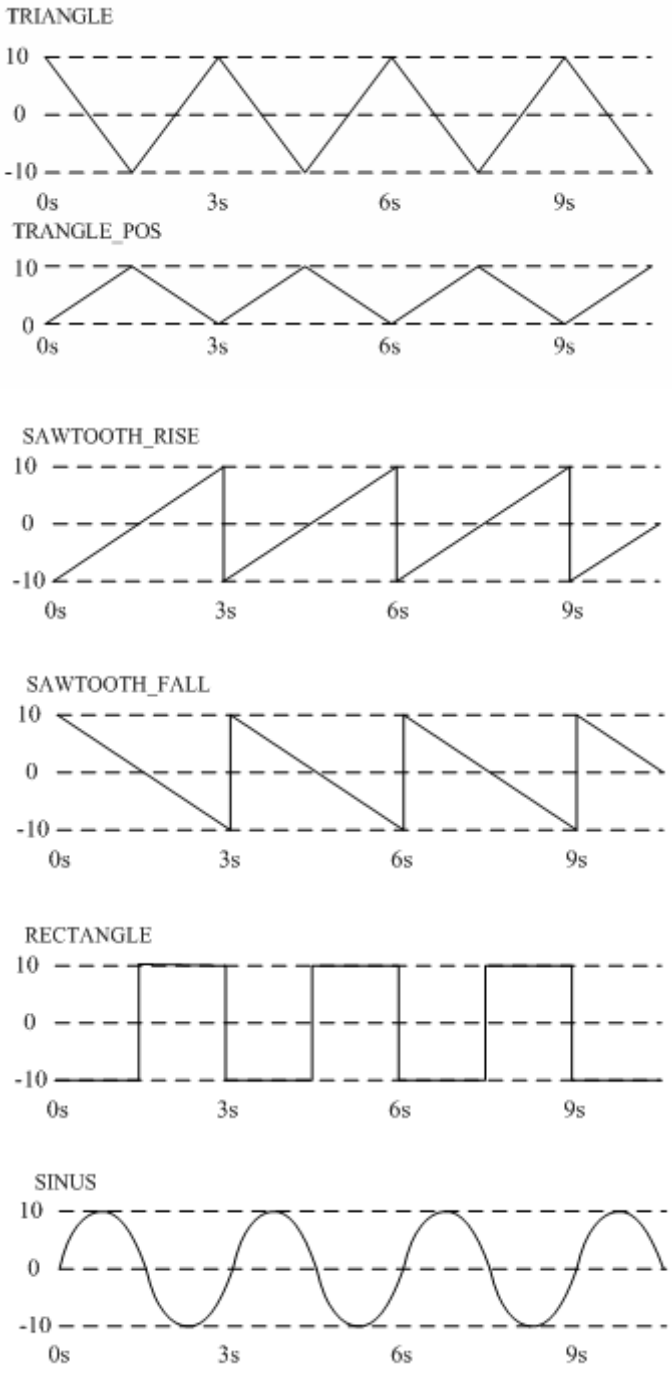
4. 指令使用举例

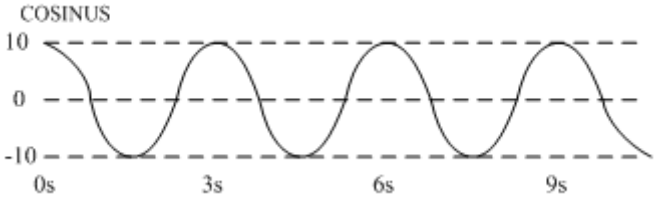
变量定义

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

功能块（FBD）	

根据 MODE 处输入不同，产生的波形如下所示。

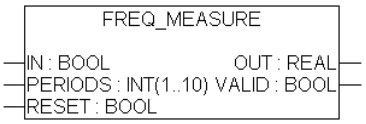




LD 语言环境中添加该指令应通过“带 EN 的框”进行调用。

4.7.3 FREQ_MEASURE（频率测量）

1. 功能：测量输入信号（BOOL 类型）的频率。
2. 类型：标准功能块（FB）。
3. 参数说明

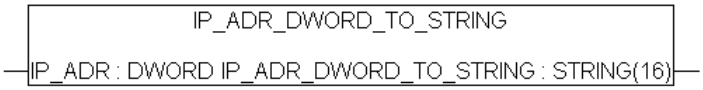


输入参数	数据类型	描述
IN	BOOL	输入信号
PERIODS	INT	测量时间间隔（s），允许范围：1~10
RESET	BOOL	复位，当 RESET=TRUE 时，所有输出清零
输出参数	**数据类型**	**描述**
OUT	REAL	测量的频率结果
VALID	BOOL	测量结果有效标志，有效时为 TRUE

第5章　　通讯指令

5.1 以太网通讯指令（Ethernet_AC500_V10.lib）

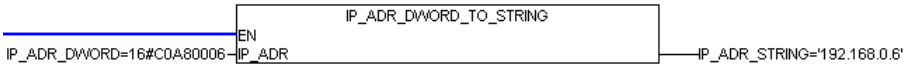
5.1.1　IP_ADR_DWORD_TO_STRING（IP 地址由 DWORD 格式转换为 STRING 格式）



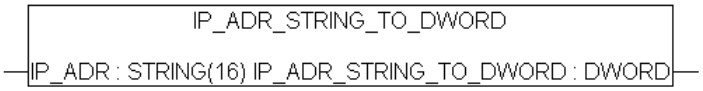
1. 功能：将以 DWORD 格式给定的 IP 地址转换成 STRING 格式的 IP 地址。STRING 格式 IP 地址由 4 个 0 到 255 之间的字段组成，以小数点分割。（例如‘192.15.24.2’）
2. 类型：标准功能（FUN）。
3. 输入变量类型：DWORD 类型。
4. 返回值类型：STRING 类型。
5. 指令使用举例

变量定义

```
0002 VAR
0003     IP_ADR_DWORD:DWORD:=16#C0A80006;
0004     IP_ADR_STRING:STRING;
0005 END_VAR
```

编程语言	程　序
梯形图（LD）	
结构化文本（ST）	IP_ADR_STRING:=IP_ADR_DWORD_TO_STRING(IP_ADR_DWORD);

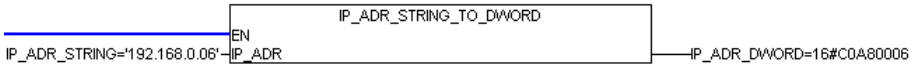
5.1.2　IP_ADR_STRING_TO_DWORD（IP 地址由 STRING 格式转换为 DWORD 格式）



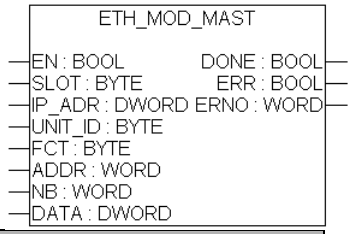
1. 功能：将以 STRING 格式给定的 IP 地址转换成 DWORD 格式的 IP 地址。IP 地址的每一字段用一个字节分别代表。（例如‘192.168.0.06’的转换结果为 16#C0A80006）
2. 类型：标准功能（FUN）。
3. 输入变量类型：STRING 类型。
4. 返回值类型：DWORD 类型。
5. 指令使用举例

变量定义

```
0002 VAR
0003     IP_ADR_STRING:STRING:='192.168.0.06';
0004     IP_ADR_DWORD:DWORD;
0005 END_VAR
```

编程语言	程　序
梯形图（LD）	
结构化文本（ST）	IP_ADR_DWORD:=IP_ADR_STRING_TO_DWORD(IP_ADR_STRING);

5.1.3　ETH_MOD_MAST （Modbus TCP 主站指令）



1. 功能：通过以太网接口或模块发送一条 Modbus TCP/IP 报文到服务器（从站）并处理相关响应。多个 ETH_MOD_MAST 功能块实例可以并行使用。
2. 类型：标准功能块（FB，含历史数据）。
3. 参数说明

输入参数	数据类型	描述
EN	BOOL	使能功能块　FALSE：无效 TRUE：上升沿使能
SLOT	BYTE	通讯模块的插槽，CPU 集成接口为 0。扩展通讯模块从右向左计数，起始值为 1
IP_ADR	DWORD	服务器（从站）的 IP 地址
UNIT_ID	BYTE	从站子地址
FCT	BYTE	MODBUS 功能代码
ADDR	WORD	服务器操作数/寄存器地址
NB	WORD	读/写的数据的数量（根据功能吗的不同，单位可为位或字）
DATA	DWORD	地址指针，客户端（主站）中数据存储区的首地址（向服务器下发的数据或从服务器读取的数据将要保存的地址）
输出参数	**数据类型**	**描述**
DONE	BOOL	功能块完成标志，操作执行完毕为 TRUE，仅保持一个扫描周期
ERR	BOOL	功能块错误标志。该输出要与 DONE 一起综合判断。DONE 为 TRUE，且 ERR 为 TRUE，则说明操作完成但存在错误；DONE 为 TRUE，且 ERR 为 FALSE，则说明操作成功
ERNO	WORD	功能块操作存在错误的代码，代码含义请参见附录 A。只有 DONE 为 TRUE，且 ERR 也为 TRUE 时，ERNO 的输出值才有效

4. 描述

- EN （使能）

EN 出现上升沿时读取输入，如果输入的数值有效，将向指定的服务器（从站）发送一个请求报文。如果输入存在无效数值，将不产生报文，而是在输出 ERR 显示错误信息。请求处理过程中，输入 EN 的状态变化可被识别，但不会被处理。

- UNIT_ID （从站子地址）

如果 Modbus 从站是由 IP_ADR 定义的 Modbus 服务器串行连接的，则必须在输入 UNIT_ID 设定 Modbus 从站的地址。

- FCT （功能代码）

  主站支持以下 Modbus 功能代码。

功能代码		功能	长度限制
DEC	HEX		基于 TCP/IP 的 Modbus
01 或 02	01 或 02	读取多个位	1800 位（通讯模块固件版本 V01.041 以上）
03 或 04	03 或 04	读取多个字	100 字/50 双字
5	5	写 1 个位	1 位
6	6	写 1 个字	1 字
7	7	读取 CPU 的状态字节	8 位
15	0F	写多个位	1800 位（通讯模块固件版本 V01.041 以上）
16	10	写多个字	100 字/50 个双字

- ADDR （服务器操作数/寄存器地址）

只能访问标准 Modbus 地址。对于 AC500PLC 从站设备通过 Modbus 可以访问 CPU 的内部寄存器区（%M 区）的 0 和 1 段。输入和输出区（%I 和%Q 区）无法通过 Modbus 直接访问。对于 eCo CPU 只支持 0 段中的前 2KB，即%MB0.0~%MB0.2047。

按字和双字访问时地址分配如下表：

Modbus 地址		字	双字
HEX	DEC	INT / WORD	DINT / DWORD
0 段			
0000	0	%MW0.0	%MD0.0
0001	1	%MW0.1	
...	...	...	...
7FFE	32766	%MW0.32766	%MD0.16383
7FFF	32767	%MW0.32767	
1 段			
8000	32768	%MW1.0	%MD1.0
8001	32769	%MW1.1	
...	...	...	...
FFFE	65534	%MW1.32766	%MD1.16383
FFFF	65535	%MW1.32767	

按位访问时地址分配如下表：

Modbus 地址		字节	位
HEX	**DEC**	**BYTE**	**BOOL**
0 段			
0000	0	%MB0.0	%MX0.0.0
0001	1		%MX0.0.1
0002	2		%MX0.0.2
0003	3		%MX0.0.3
0004	4		%MX0.0.4
0005	5		%MX0.0.5
0006	6		%MX0.0.6
0007	7		%MX0.0.7
0008	8	%MB0.1	%MX0.1.0
0009	9		%MX0.1.1
000A	10		%MX0.1.2
000B	11		%MX0.1.3
000C	12		%MX0.1.4
000D	13		%MX0.1.5
000E	14		%MX0.1.6
000F	15		%MX0.1.7
...	...	...	...
0FFF	4095	%MB0.511	%MX0.511.7
1000	4096	%MB0.512	%MX0.512.0
...	...	...	...
7FFF	32767	%MB0.4095	%MX0.4095.7
8000	32768	%MB0.4096	%MX0.4096.0
...	...	...	...
FFFF	65535	%MB0.8191	%MX0.8191.7

位访问时的 Modbus 地址 ADDR（10 进制为例）与变量直接地址（%MX0.BYTE.BIT）的对应关系如下：

ADDR:=BTYE*8+BIT

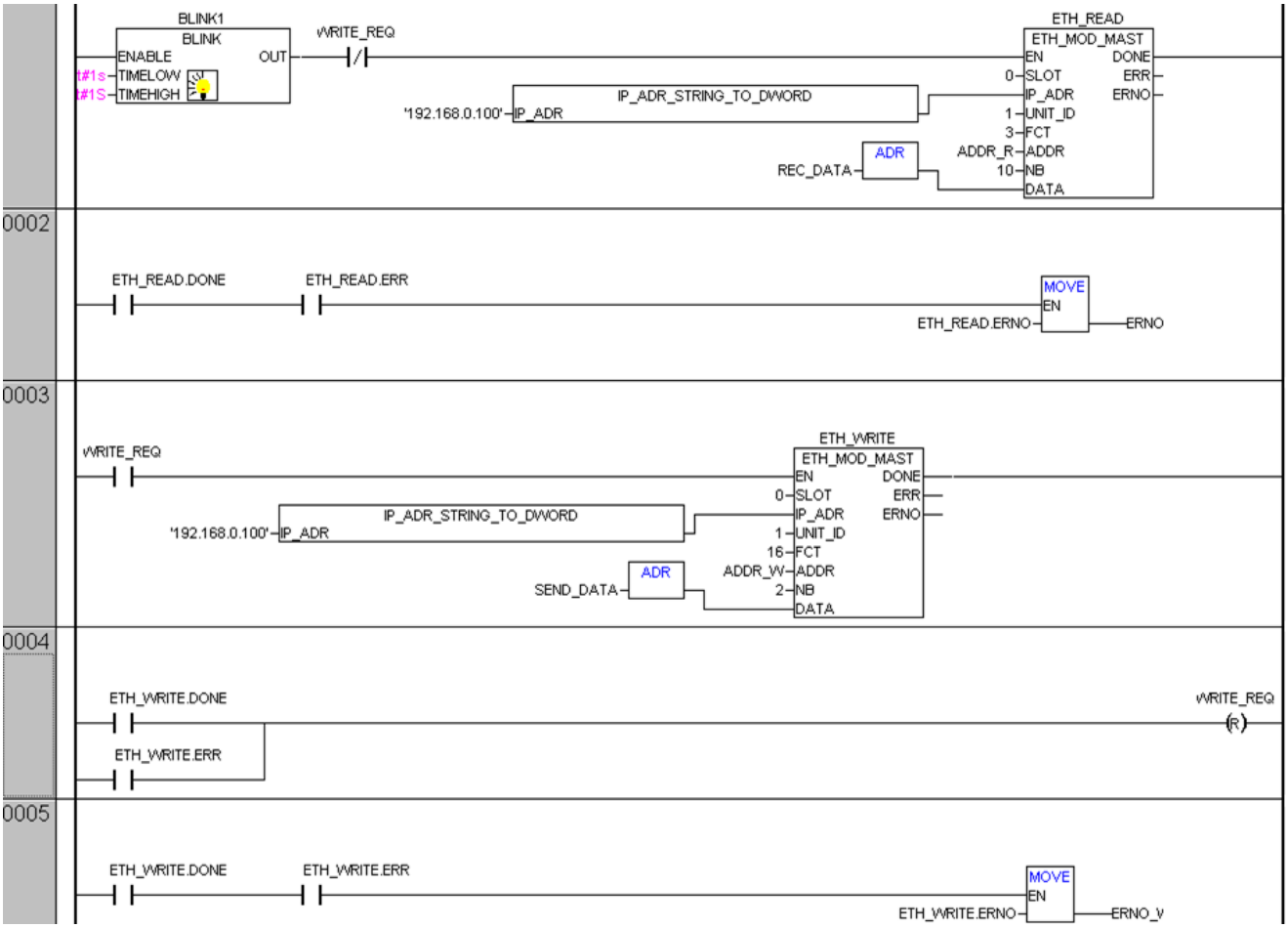
1. 按**字**访问的地址范围**%MW0.0 ..%MW1.32767**
   按**位**访问的地址范围**%MX0.0.0 ..%MX0.8191.7**
2. 不能对0段和1段中的读保护和写保护区域进行读写操作

- DATA （服务器操作数/寄存器地址）

输入 DATA 指定客户端（主站）中数据存储区的首地址（可以通过 ADR 指令得到）。数据从该首地址写入服务器（从站）或者以该首地址开始保存从服务器（从站）读取的数据。操作数的类型（位或字）必须与所选择的功能代码相匹配。

5. 指令使用举例

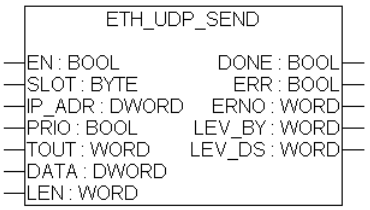
```
0002 VAR
0003     BLINK1: BLINK;
0004     ETH_READ: ETH_MOD_MAST;
0005     ADDR_R: WORD;
0006     REC_DATA AT %MW0.0: ARRAY [1..10] OF WORD;
0007     ERNO_R: WORD;
0008     WRITE_REQ: BOOL;
0009     ETH_WRITE: ETH_MOD_MAST;
0010     SEND_DATA AT %MW0.20: ARRAY [1..10] OF WORD;
0011     ADDR_W: WORD:=20;
0012     ERNO_W: WORD;
```



- 定时读取从站数据，数据长度 10 个字，保存的起始地址为%MW0.0。
- 如果读取操作出现错误，取出错误代码。
- 出现数据写入请求时（WRITE_RWQ 为 TRUE），主站向从站写入数据，数据长度 2 个字，数据的起始地址为%MW0.20。执行完毕后将 WRITE_RWQ 复位。
- 如果写入操作出现错误，取出错误代码。

5.1.4 ETH_UDP_SEND（通过 UDP/IP 发送数据）

1. 功能：通过以太网接口或模块采用 UDP/IP 协议发送数据包。
2. 类型：标准功能块（FB，含历史数据）。



3. 参数说明

输入参数	数据类型	描述
EN	BOOL	使能功能块　FALSE：无效 TRUE：上升沿使能
SLOT	BYTE	通讯模块的插槽，CPU 集成接口为 0。扩展通讯模块从右向左计数，起始值为 1
IP_ADR	DWORD	数据传输的目标站（接收设备）的 IP 地址
PRIO	BOOL	数据包传输的优先级　FALSE：低优先级 T RUE：高优先级
TOUT	WORD	数据包发送超时时间（ms）
DATA	DWORD	地址指针，指向发送数据变量的存储区首地址（可通过 ADR 指令获取该地址）。变量必须是 ARRAY 或 STRUCT 类型
LEN	WORD	发送数据的长度（以字节为单位，1≤LEN≤1464）
输出参数	**数据类型**	**描述**
DONE	BOOL	功能块完成标志，操作执行完毕为 TRUE，仅保持一个扫描周期
ERR	BOOL	功能块错误标志。该输出要与 DONE 一起综合判断。DONE 为 TRUE，且 ERR 为 TRUE，则说明操作完成但存在错误；DONE 为 TRUE，且 ERR 为 FALSE，则说明操作成功
ERNO	WORD	功能块操作存在错误的代码，代码含义请参见附录 A。只有 DONE 为 TRUE，且 ERR 也为 TRUE 时，ERNO 的输出值才有效
LEV_BY	WORD	低/高优先级的发送缓冲区的数据量（以字节为单位）（根据 PRIO 确定优先级）。只要 EN 为 TURE，显示值就会更新
LEV_DS	WORD	低/高优先级的发送缓冲区的数据量（以报文个数为单位）（根据 PRIO 确定优先级）。只要 EN 为 TURE，显示值就会更新

4. 描述

UDP/IP 数据交换在标准 UDP/IP 通讯报文基础上增加 AC31 报文，可用于 ABB PLC CPU 之间进行通讯。当 ABB PLC CPU 与第三方以太网设备进行 UDP 通讯时，应采用标准 UDP/IP 通讯。

- PRIO （优先级）

输入 PRIO 指定数据包传输的优先级。

PRIO =FALSE：指定的数据包为低优先级，数据包存储在低优先级发送缓冲区。

PRIO =TRUE：指定的数据包为高优先级，数据包存储在高优先级发送缓冲区。

发送缓冲区的大小通过 PLC 配置确定。

- TOUT （超时时间）

TOUT =0：数据交换不需要接收确认报文。

TOUT > 0：数据交换需要接收确认报文。如果在设定的超时时间内（ms）没有收到确认报文，则终止传输，并将数据包信息保存到超时缓冲区，可使用功能块 ETH_UDP_STO 进行读取。

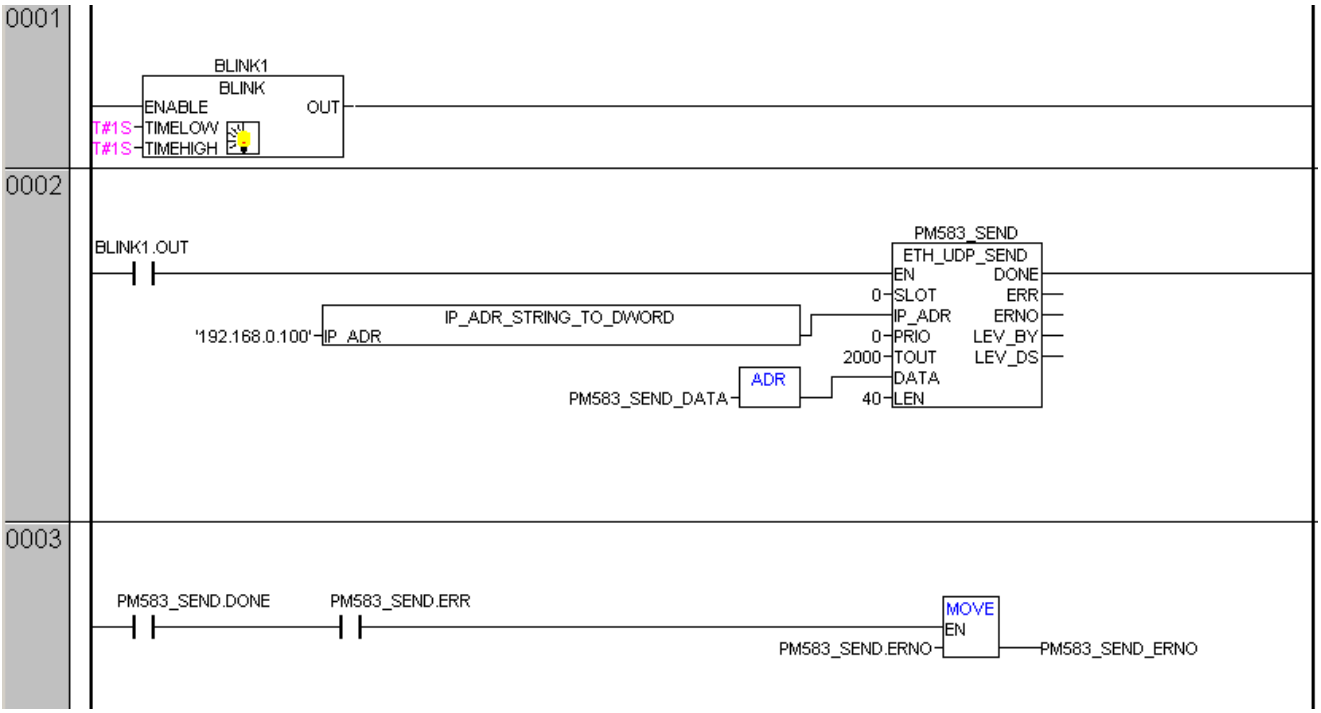
- LEV_BY （以字节为单位显示数据量）

LEV_BY 以字节为单位显示由输入 PRIO 选择的发送缓冲区中的数据量。只要 EN 为 TURE，显示值就会更新，传输完毕时读取最终数据。

在发送缓冲区中，一个报文占用 LEN+8 个字节（发送设备的 IP 地址占用 4 个字节，长度占用 2 个字节，超时时间占用 2 个字节）。

5. 指令使用举例

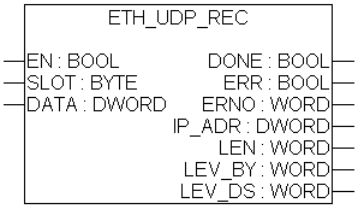
```
0002 VAR
0003     BLINK1: BLINK;
0004     PM583_SEND: ETH_UDP_SEND;
0005     PM583_SEND_DATA AT %MW0.50: ARRAY [1..20] OF WORD;
0006     PM583_SEND_ERNO: WORD;
```



- 定期向 IP 地址为 192.168.0.100 的 AC500 PLC 发送数据，数据长度 20 个字（40 个字节），数据区起始地址为%MW0.50。
- 如果发送出现错误，取出错误代码。如果发送期间发生错误，数据包不会自动重发，只做信息指示。

5.1.5 ETH_UDP_REC（通过 UDP/IP 接收数据）

1. 功能：读取通讯接口或模块接收的 UDP/IP 数据包。
2. 类型：标准功能块（FB，含历史数据）。
3. 参数说明



输入参数	数据类型	描述
EN	BOOL	使能功能块　FALSE：无效 TRUE：高电平有效
SLOT	BYTE	通讯模块的插槽，CPU 集成接口为 0。扩展通讯模块从右向左计数，起始值为 1
DATA	DWORD	地址指针，指向接收数据的存储区首地址（可通过 ADR 指令获取该地址）。变量必须是 ARRAY 或 STRUCT 类型

输出参数	数据类型	描述
DONE	BOOL	功能块完成标志，操作执行完毕为 TRUE，仅保持一个扫描周期
ERR	BOOL	功能块错误标志。该输出要与 DONE 一起综合判断。DONE 为 TRUE，且 ERR 为 TRUE，则说明操作完成但存在错误；DONE 为 TRUE，且 ERR 为 FALSE，则说明操作成功
ERNO	WORD	功能块操作存在错误的代码，代码含义请参见附录 A。只有 DONE 为 TRUE，且 ERR 也为 TRUE 时，ERNO 的输出值才有效
IP_ADR	DWORD	发送设备的 IP 地址
LEN	WORD	数据包的长度（以字节为单位，1≤LEN≤1464）
LEV_BY	WORD	接收缓冲区数据量（以字节为单位）
LEV_DS	WORD	接收缓冲区数据量（以报文个数为单位）

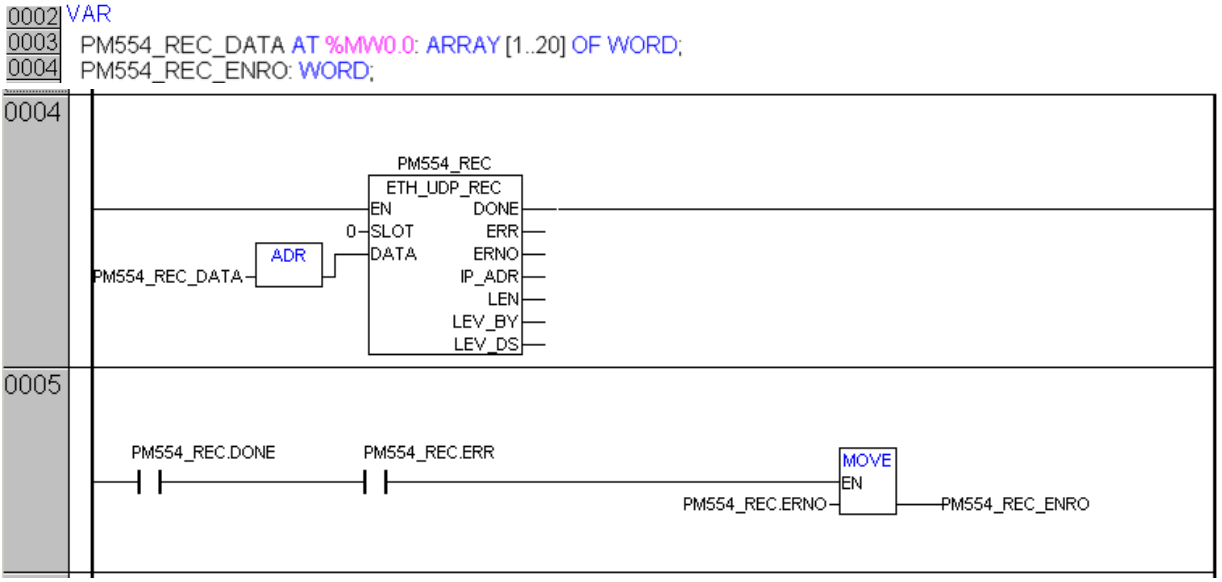
4. 描述

接收的数据首先保存到接收缓冲区内(缓冲区大小通过 PLC 配置确定)，再通过该指令读取每个数据包，并保存到指定的内存区域内（由 DATA 指定）。输出 IP_ADR 和 LEN 指示发送设备的地址和数据包长度。

LEV_BY 以字节为单位显示接收缓冲区中的数据量。只要 EN 为 TURE，显示值就会更新，传输完毕时读取最终数据。

在接收缓冲区中，一个报文占用 LEN+6 个字节(发送设备的 IP 地址占用 4 个字节，长度占用 2 个字节)。

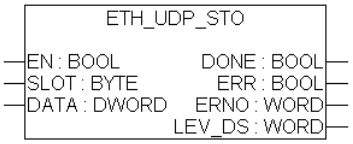
5. 指令使用举例



- 实时接收数据，数据保存的起始地址为%MW0.0。本例中接收数据长度为 20 个字，对应上例中（ETH_UDP_SEND）发送的数据长度。
- 如果出现错误，取出错误代码。

5.1.6 ETH_UDP_STO（读取 UDP/IP 超时数据包）

1. 功能：通过该指令从超时缓冲区读取丢失的数据包信息，并保存到设定的内存区域（由 DATA 指定）。
2. 类型：标准功能块（FB，含历史数据）。



3. 参数说明

输入参数	数据类型	描述
EN	BOOL	使能功能块　FALSE：无效 TRUE：高电平有效
SLOT	BYTE	通讯模块的插槽，CPU 集成接口为 0。扩展通讯模块从右向左计数，起始值为 1
DATA	DWORD	地址指针，指向保存超时数据包信息的存储区首地址（可通过 ADR 指令获取该地址）。变量必须是 ARRAY 或 STRUCT 类型
输出参数	**数据类型**	**描述**
DONE	BOOL	功能块完成标志，操作执行完毕为 TRUE，仅保持一个扫描周期
ERR	BOOL	功能块错误标志。该输出要与 DONE 一起综合判断。DONE 为 TRUE，且 ERR 为 TRUE，则说明操作完成但存在错误；DONE 为 TRUE，且 ERR 为 FALSE，则说明操作成功
ERNO	WORD	功能块操作存在错误的代码，代码含义请参见附录 A。只有 DONE 为 TRUE，且 ERR 也为 TRUE 时，ERNO 的输出值才有效
LEV_DS	WORD	超时缓冲区数据量（以报文个数为单位）

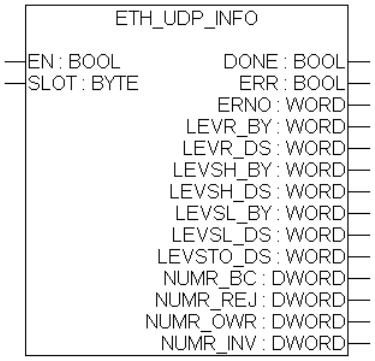
4. 描述

超时缓冲区结构类似循环缓冲区（FIFO）。如果缓冲区满，缓冲区中的最早的项将被覆盖。（超时缓冲区的大小通过 PLC 配置确定）。发送功能块 ETH_UDP_SEND 的超时时间 TOUT> 0 时，在发送数据包期间，将监视是否成功。当超过超时时间时，数据包的特殊信息被存储到超时缓冲区。

从超时缓冲区读取的数据包信息包括以下内容：

- 接收设备的 IP 地址（4 个字节）；
- 数据包的报头数据。

5.1.7 ETH_UDP_INFO（读取 UDP/IP 处理的状态信息）

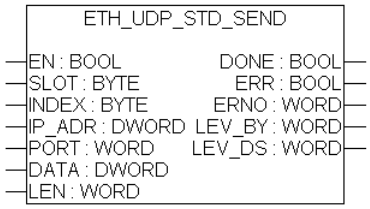


1. 功能：读取有关 UDP/IP 处理的各种状态信息。
2. 类型：标准功能块（FB，含历史数据）。
3. 参数说明

输入参数	数据类型	描述
EN	BOOL	使能功能块　FALSE：无效 TRUE：高电平有效
SLOT	BYTE	通讯模块的插槽，CPU 集成接口为 0。扩展通讯模块从右向左计数，起始值为 1
输出参数	**数据类型**	**描述**
DONE	BOOL	功能块完成标志，操作执行完毕为 TRUE，仅保持一个扫描周期

输出参数	数据类型	描述
ERR	BOOL	功能块错误标志。该输出要与 DONE 一起综合判断。DONE 为 TRUE，且 ERR 为 TRUE，则说明操作完成但存在错误；DONE 为 TRUE，且 ERR 为 FALSE，则说明操作成功
ERNO	WORD	功能块操作存在错误的代码，代码含义请参见附录 A。只有 DONE 为 TRUE，且 ERR 也为 TRUE 时，ERNO 的输出值才有效
LEVR_BY	WORD	接收缓冲区数据量（以字节为单位）
LEVR_DS	WORD	接收缓冲区数据量（以报文个数为单位）
LEVSH_BY	WORD	高优先级的发送缓冲区数据量（以字节为单位）
LEVSH_DS	WORD	高优先级的发送缓冲区数据量（以报文个数为单位）
LEVSL_BY	WORD	低优先级的发送缓冲区数据量（以字节为单位）
LEVSL_DS	WORD	低优先级的发送缓冲区数据量（以报文个数为单位）
LEVSTO_DS	WORD	超时缓冲区数据量（以报文个数为单位）
NUMR_BC	WORD	接收广播报文的数量
NUMR_REJ	WORD	由于接收缓冲区满载而丢弃的报文数量
NUMR_OWR	WORD	由于接收缓冲区满载而被覆盖的报文数量
NUMR_INV	WORD	接收故障的报文数量

5.1.8 ETH_UDP_STD_SEND（通过标准 UDP/IP 发送数据）



1. 功能：通过以太网接口或模块采用标准 UDP/IP 协议传输数据包。
2. 类型：标准功能块（FB，含历史数据）。
3. 参数说明

输入参数	数据类型	描述
EN	BOOL	使能功能块　FALSE：无效 TRUE：上升沿使能
SLOT	BYTE	通讯模块的插槽，CPU 集成接口为 0。扩展通讯模块从右向左计数，起始值为 1。
INDEX	BYTE	发送设备 PLC 配置中建立的标准 UDP/IP 连接的索引，从 0 开始
IP_ADR	DWORD	数据传输的目标站（接收设备）的 IP 地址
PORT	WORD	接收设备的 UDP 端口
DATA	DWORD	地址指针，指向发送数据变量的存储区首地址（可通过 ADR 指令获取该地址）。变量必须是 ARRAY 或 STRUCT 类型
LEN	WORD	发送数据的长度（以字节为单位，1≤LEN≤1464）
输出参数	**数据类型**	**描述**
DONE	BOOL	功能块完成标志，操作执行完毕为 TRUE，仅保持一个扫描周期
ERR	BOOL	功能块错误标志。该输出要与 DONE 一起综合判断。DONE 为 TRUE，且 ERR 为 TRUE，则说明操作完成但存在错误；DONE 为 TRUE，且 ERR 为 FALSE，则说明操作成功

输出参数	数据类型	描述
ERNO	WORD	功能块操作存在错误的代码，代码含义请参见附录 A。只有 DONE 为 TRUE，且 ERR 也为 TRUE 时，ERNO 的输出值才有效
LEV_BY	WORD	发送缓冲区的数据量（以字节为单位）只要 EN 为 TURE，显示值就会更新
LEV_DS	WORD	发送缓冲区的数据量（以报文个数为单位）只要 EN 为 TURE，显示值就会更新。

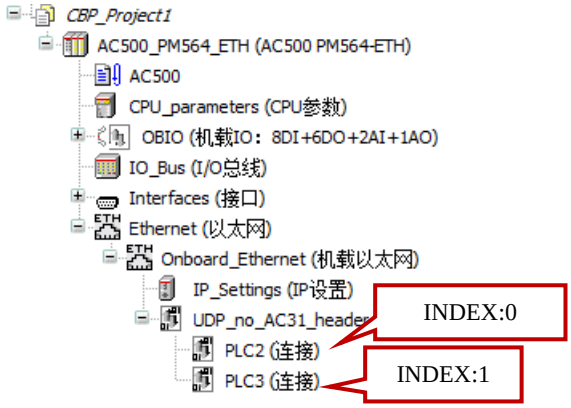
4. 描述

UDP/IP 数据交换在标准 UDP/IP 通讯报文基础上增加 AC31 报文，可用于 ABB PLC CPU 之间进行通讯。当 ABB PLC CPU 与第三方以太网设备进行 UDP 通讯时，应采用标准 UDP/IP 通讯。

指定的数据包存储到发送缓冲区，系统将数据包转交给以太网通讯模块（接口），以将其传输到输入 IP_ADR 设定的目标地址。如果处理期间发生错误，数据包不会自动重发，只做信息指示。发送缓冲区的大小通过 PLC 配置确定。

- INDEX （标准 UDP/IP 连接的索引）

INDEX 为发送设备配置中标准 UDP/IP 连接的索引，从 0 开始编号，见下图：



- PORT （接收设备的 UDP 端口）

CBP 中，有些端口具有特定的用途，如下表中的各端口：

应用	端口
Reserved（预留）	0
Echo	7
IEN 116	42
Domain Name Service	53
BootP	67/68
NetIdent Protocol	25383
UDP blocks	32768

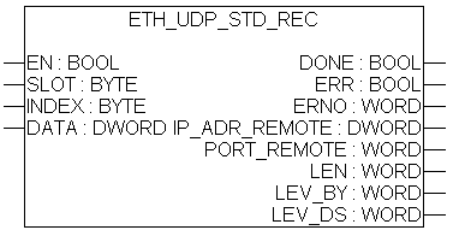
为了避免选择的端口与特定用途端口冲突，在设置端口时推荐使用 2000 到 65535 区段的端口。

- LEV_BY （以字节为单位显示数据量）

LEV_BY 以字节为单位显示发送缓冲区中的数据量。只要 EN 为 TURE，显示值就会更新，传输完毕时读取最终数据。

在发送缓冲区中，一个报文占用 LEN+8 个字节（发送设备的 IP 地址占用 4 个字节，长度占用 2 个字节，超时时间占用 2 个字节）。

5.1.9 ETH_UDP_STD_REC（通过标准 UDP/IP 接收数据）



1. 功能：读取从以太网接口或模块接收的标准 UDP/IP 数据包。
2. 类型：标准功能块（FB，含历史数据）。
3. 参数说明

输入参数	数据类型	描述
EN	BOOL	使能功能块　FALSE：无效 TRUE：高电平有效
SLOT	BYTE	通讯模块的插槽，CPU 集成接口为 0。扩展通讯模块从右向左计数，起始值为 1
INDEX	BYTE	接收设备 PLC 配置中建立的标准 UDP/IP 连接的索引，从 0 开始
DATA	DWORD	地址指针，指向接收数据的存储区首地址（可通过 ADR 指令获取该地址）。变量必须是 ARRAY 或 STRUCT 类型
输出参数	**数据类型**	**描述**
DONE	BOOL	功能块完成标志，操作执行完毕为 TRUE，仅保持一个扫描周期
ERR	BOOL	功能块错误标志。该输出要与 DONE 一起综合判断。DONE 为 TRUE，且 ERR 为 TRUE，则说明操作完成但存在错误；DONE 为 TRUE，且 ERR 为 FALSE，则说明操作成功
ERNO	WORD	功能块操作存在错误的代码，代码含义请参见附录 A。只有 DONE 为 TRUE，且 ERR 也为 TRUE 时，ERNO 的输出值才有效
IP_ADR_REMOTE	DWORD	发送设备的 IP 地址
PORT_REMOTE	WORD	发送设备的 UDP 源端口
LEN	WORD	数据包的长度（以字节为单位，1≤LEN≤1464）
LEV_BY	WORD	接收缓冲区数据量（以字节为单位）
LEV_DS	WORD	接收缓冲区数据量（以报文个数为单位）

4. 描述

系统读取从通讯接口接收的标准 UDP/IP 数据包，并将它们保存到接收缓冲区内。接收缓冲区的大小通过 PLC 配置确定。再通过该指令读取各数据包，保存到指定的内存区域内（由 DATA 指定）。

- INDEX （标准 UDP/IP 连接的索引）

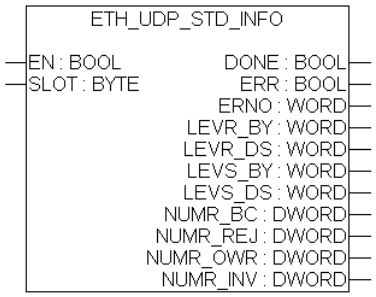
INDEX 为接收设备配置中标准 UDP/IP 连接的索引，从 0 开始编号，请参见 ETH_UDP_STD_SEND。

- LEV_BY （以字节为单位显示数据量）

以字节为单位显示接收缓冲区中的数据量。只要 EN 为 TURE，显示值就会更新，传输完毕时读取最终数据。接收缓冲区中，一个报文占用 LEN+6 个字节（发送设备的 IP 地址占用 4 个字节，长度占用 2 个字节）。

使用标准UDP/IP每次传输的数据报文长度最大为1464字节。

5.1.10 ETH_UDP_STD_INFO（读取标准 UDP/IP 处理的状态信息）

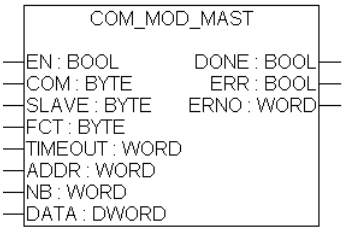


1. 功能：读取有关标准 UDP/IP 处理的各种状态信息。
2. 类型：标准功能块（FB，含历史数据）。
3. 参数说明

输入参数	数据类型	描述
EN	BOOL	使能功能块　FALSE：无效 TRUE：高电平有效
SLOT	BYTE	通讯模块的插槽，CPU 集成接口为 0。扩展通讯模块从右向左计数，起始值为 1
输出参数	**数据类型**	**描述**
DONE	BOOL	功能块完成标志，操作执行完毕为 TRUE，仅保持一个扫描周期
ERR	BOOL	功能块错误标志。该输出要与 DONE 一起综合判断。DONE 为 TRUE，且 ERR 为 TRUE，则说明操作完成但存在错误；DONE 为 TRUE，且 ERR 为 FALSE，则说明操作成功
ERNO	WORD	功能块操作存在错误的代码，代码含义请参见附录 A。只有 DONE 为 TRUE，且 ERR 也为 TRUE 时，ERNO 的输出值才有效
LEVR_BY	WORD	接收缓冲区数据量（以字节为单位）
LEVR_DS	WORD	接收缓冲区数据量（以报文个数为单位）
LEVS_BY	WORD	发送缓冲区数据量（以字节为单位）
LEVS_DS	WORD	发送缓冲区数据量（以报文个数为单位）
NUMR_BC	WORD	接收广播报文的数量
NUMR_REJ	WORD	由于接收缓冲区满载而丢弃的报文数量
NUMR_OWR	WORD	由于接收缓冲区满载而被覆盖的报文数量
NUMR_INV	WORD	接收故障的报文数量

5.2 Modbus RTU 通讯指令（Modbus_AC500_V10.lib）

CPU 集成串口进行 Modbus RTU 通讯，需要在主站编制数据读写程序，而从站部分无需编写程序，仅设置串口的基本通讯参数即可。



1. COM_MOD_MAST：集成串口的 Modbus RTU 主站指令
2. 功能：用于处理基于串口的 Modbus RTU 主站报文。
3. 类型：标准功能块（FB，含历史数据）。
4. 参数说明

输入参数	数据类型	描述
EN	BOOL	使能功能块　FALSE：无效 TRUE：上升沿使能
COM	BYTE	指定串行接口的编号　1 ：COM1 接口 2 ：COM2 接口
SLAVE	BYTE	从站地址
FCT	BYTE	功能代码
TIMEOUT	WORD	超时时间（ms）
ADDR	WORD	从站寄存器地址
NB	WORD	数据长度
DATA	DWORD	地址指针，指向主站数据区首地址。发送时将该区域的数据发送到从站；接收时将读自从站的数据保存在该区域。（可通过 ADR 指令获取该地址）
输出参数	**数据类型**	**描述**
DONE	BOOL	功能块完成标志，操作执行完毕为 TRUE，仅保持一个扫描周期
ERR	BOOL	功能块错误标志。该输出要与 DONE 一起综合判断。DONE 为 TRUE，且 ERR 为 TRUE，则说明操作完成但存在错误；DONE 为 TRUE，且 ERR 为 FALSE，则说明操作成功
ERNO	WORD	功能块操作存在错误的代码，代码含义请参见附录 A。只有 DONE 为 TRUE，且 ERR 也为 TRUE 时，ERNO 的输出值才有效

5. 描述

CPU 集成的串口 COM1 和 COM2 均支持 Modbus 主站功能。对于每个串口都应使用独立的 COM_MOD_MAST 功能块，通过输入 COM 具体指定。

- FCT　（功能代码）

主站支持以下 Modbus 功能代码。

功能代码		描述	长度限制
DEC	**HEX**		**基于串口的 Modbus**
01 或 02	01 或 02	读取多个个位	2000 位
03 或 04	03 或 04	读多个字	125 个字/62 个双字
5	5	写 1 个位	1 位
6	6	写 1 个字	1 个字
7	7	快速读取 CPU 状态字节	8 位
15	0F	写多个位	1968 位
16	10	写多个字	123 个字/61 个双字

- ADDR （从站寄存器地址）

只能访问标准 Modbus 地址。对于 AC500 PLC 从站设备通过 Modbus 可以访问 CPU 的内部寄存器区（%M 区）的 0 和 1 段。输入和输出区（%I 和%Q 区）无法通过 Modbus 直接访问。对于 eCo CPU 只支持 0 段中的前 2 KB，即%MB0.0~%MB0.2047。

按字和双字访问时地址分配如下表：

Modbus 地址		**字**	**双字**
HEX	**DEC**	**INT / WORD**	**DINT / DWORD**
0 段			
0	0	%MW0.0	%MD0.0
1	1	%MW0.1	
...	...	...	...
7FFE	32766	%MW0.32766	%MD0.16383
7FFF	32767	%MW0.32767	
1 段			
8000	32768	%MW1.0	%MD1.0
8001	32769	%MW1.1	
...	...	...	...
FFFE	65534	%MW1.32766	%MD1.16383
FFFF	65535	%MW1.32767	

按位访问时地址分配如下表：

Modbus 地址		**字节**	**位**
HEX	**DEC**	**BYTE**	**BOOL**
0 段			
0000	0	%MB0.0	%MX0.0.0
0001	1		%MX0.0.1
0002	2		%MX0.0.2
0003	3		%MX0.0.3
0004	4		%MX0.0.4
0005	5		%MX0.0.5
0006	6		%MX0.0.6
0007	7		%MX0.0.7
0008	8	%MB0.1	%MX0.1.0
0009	9		%MX0.1.1
000A	10		%MX0.1.2
000B	11		%MX0.1.3
000C	12		%MX0.1.4
000D	13		%MX0.1.5
000E	14		%MX0.1.6
000F	15		%MX0.1.7
...	...	...	...
0FFF	4095	%MB0.511	%MX0.511.7
1000	4096	%MB0.512	%MX0.512.0
...	...	...	...

0 段			
7FFF	32767	%MB0.4095	%MX0.4095.7
8000	32768	%MB0.4096	%MX0.4096.0
...	...	...	...
FFFF	65535	%MB0.8191	%MX0.8191.7

位访问时的 Modbus 地址 ADDR（10 进制为例）与变量直接地址（%MX0.BYTE.BIT）的对应关系如下：

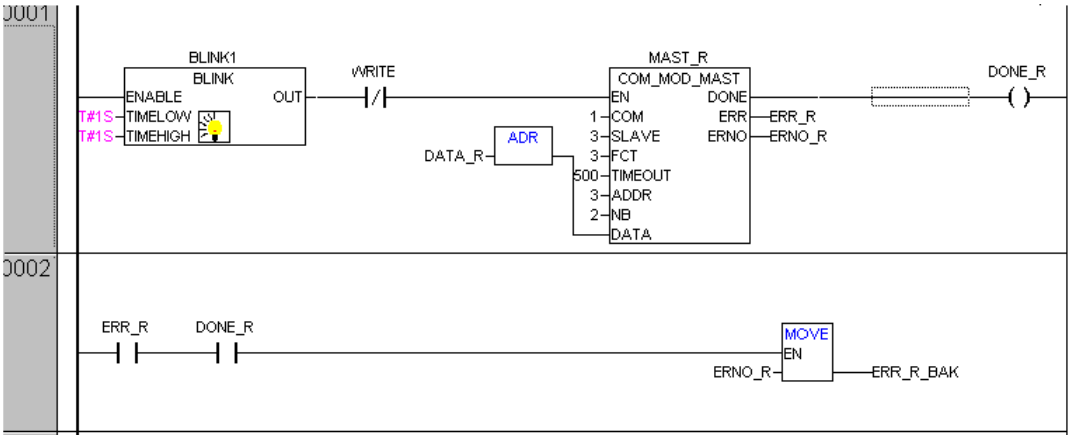
ADDR:=BTYE*8+BIT

1. 按**字**访问的地址范围**%MW0.0 ..%MW1.32767**
   按**位**访问的地址范围**%MX0.0.0 ..%MX0.0.8191.7**
2. 不能对0段和1段中的读保护和写保护区域进行读写操作

- DATA （主站数据区首地址）

输入 DATA 指定主站中数据存储区的首地址（可以通过 ADR 指令得到）。数据从该首地址写入从站或者以该首地址开始保存读取的从站数据。操作数的类型（位或字）必须与所选择的功能代码相匹配。

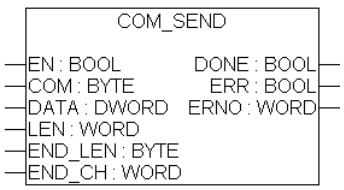
6. 指令使用举例



- 周期通过 COM1 读取 3 号从站的 0003 和 0004 两个寄存器的数据，数据发送时不进行读取。
- 数据存放在 DATA_R 数组中。
- 读取错误时取出错误代码。

5.3 ASCII 通讯指令（ASCII_AC500_V10.lib）

5.3.1 COM_SEND（串口在自由模式下发送数据）



1. 功能：CPU 集成串口在自由模式下发送数据。
2. 类型：标准功能块（FB，含历史数据）。

3. 参数说明

输入参数	数据类型	描述
EN	BOOL	使能功能块　FALSE：无效 TRUE：上升沿使能
COM	BYTE	指定串行接口的编号　1 ：COM1 接口 2 ：COM2 接口
DATA	DWORD	地址指针，指向发送数据的存储区首地址（可通过 ADR 指令获取该地址）
LEN	WORD	发送有效数据的长度（以字节为单位，从 DATA 地址开始，不包括校验及和终止字符）
END_LEN	BYTE	要在数据尾端添加的终止字符个数（0，1，2）
END_CH	WORD	要在数据尾端添加的终止字符，不得超过 2 个字符（例如：16#0D0A 占用 2 个字符）
输出参数	**数据类型**	**描述**
DONE	BOOL	功能块完成标志，操作执行完毕为 TRUE，仅保持一个扫描周期
ERR	BOOL	功能块错误标志。该输出要与 DONE 一起综合判断。DONE 为 TRUE，且 ERR 为 TRUE，则说明操作完成但存在错误；DONE 为 TRUE，且 ERR 为 FALSE，则说明操作成功
ERNO	WORD	功能块操作存在错误的代码，代码含义请参见附录 A。只有 DONE 为 TRUE，且 ERR 也为 TRUE 时，ERNO 的输出值才有效

4. 描述

在一个项目中，COM_SEND 功能块的数量以及其分布情况对用户程序来说是不受限制的。输入端 EN 的上升沿触发传送操作。

要发送的数据的长度不受限制。但是，当发送缓冲器具有足够的剩余空间时才能发送成功，因此推荐在发送缓冲器中写入的数据块长度不超过 256 字节。如果需要，较长的报文可以通过使用多个 COM_SEND 功能块实例逐个发送。

CBP 配置中 ASCII 协议的参数“校验和”，如果选择的不为“无”，发送时会附加所选择的校验和。其字符数量由输入 END_LEN 来决定。

发送报文包括以下数据：

数据 0	数据 1	数据 2	..	数据 n	校验和 1	[校验和 2]	结束 1	[结束 2]

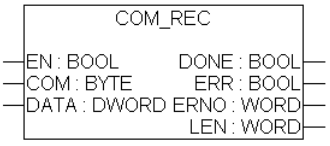
[ ]标示可选，仅适用于 CRC16 校验和或 2 个结束字符的报文。

对于以上报文 DATA 指向的区域包含以下数据：

数据 0	数据 1	数据 2	..	数据 n

5.3.2 COM_REC（串口在自由模式下接收数据）

1. 功能：CPU 集成串口在自由模式下接收数据。接收报文的结束条件通过 CBP 中的 ASCII 参数设置。
2. 类型：标准功能块（FB，含历史数据）。
3. 参数说明



输入参数	数据类型	描述
EN	BOOL	使能功能块　FALSE：无效 TRUE：上升沿使能，高电平有效
COM	BYTE	指定串行接口的编号　1 ：COM1 接口 2 ：COM2 接口
DATA	DWORD	地址指针，指向存储接收数据的存储区首地址（可通过 ADR 指令获取该地址）

输出参数	LEN	描述
DONE	BOOL	功能块完成标志，操作执行完毕为 TRUE，仅保持一个扫描周期
ERR	BOOL	功能块错误标志。该输出要与 DONE 一起综合判断。DONE 为 TRUE，且 ERR 为 TRUE，则说明操作完成但存在错误；DONE 为 TRUE，且 ERR 为 FALSE，则说明操作成功
ERNO	WORD	功能块操作存在错误的代码，代码含义请参见附录 A。只有 DONE 为 TRUE，且 ERR 也为 TRUE 时，ERNO 的输出值才有效。
LEN	WORD	接收有效数据的长度（以字节为单位，从 DATA 地址开始，不包括终止字符）

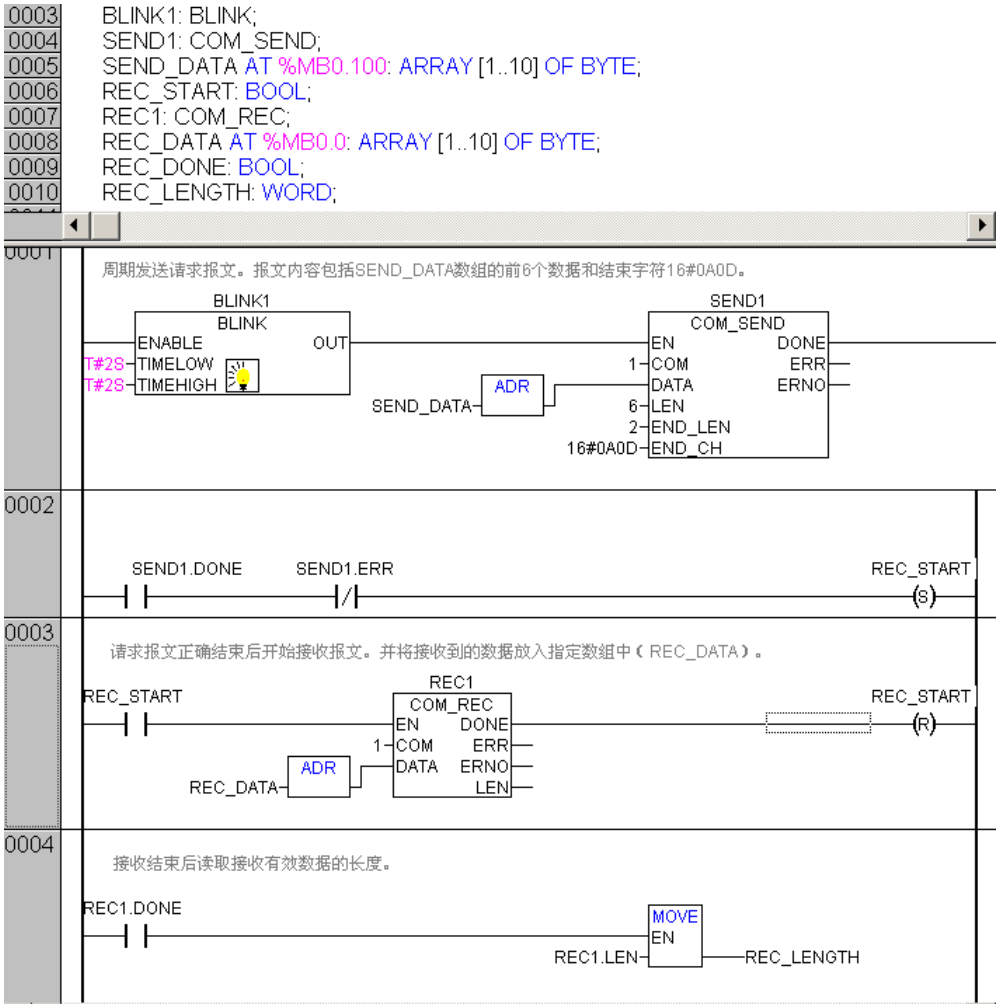
4. 描述

建议在一个项目中仅使用一个 COM_REC 功能块。

接收的数据长度最大限制为 256 字节，并在 LEN 输出。LEN 仅在 DONE =TRUE 时有效且保持一个扫描周期，应及时读取。

CBP 配置中 ASCII 协议的参数“校验和”，如果选择的不为“无”，则接收时会自动检查校验和。如果参数“报文尾缀选择”设置为“字符串”，则根据字符判断报文是否结束。校验和及结束字符均不输出，即它们不包含在接收数据中（DATA 指向的区域）。

5. 指令使用举例



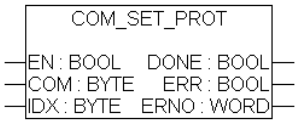
- 周期通过 COM1 发送请求报文。报文内容包括 SEND_DATA 数组的前 6 个数据和结束字符 16#0A0D。
- 请求报文正确结束后开始接收报文。并将接收到的数据放入指定数组中（REC_DATA）。
- 接收结束后读取接收有效数据的长度。（假设本例的结束条件在 CBP 中的设置为 1 个结束字符 16#0D）

COM1 - ASCII 配置

参数	类型	值	缺...	单位	描述
允许登入	Enumeration of BYTE	禁止	禁止		检查CoDeSys登入
RTS控制器	Enumeration of BYTE	报文	无		RTS控制器必须为RS485设定为'报文'!
TLS	WORD(0..65535)	0	0		设定以ms计的TLS(载体超前时间)(TLS > CDLY)
CDLY	WORD(0..65535)	0	0		设定以ms 计的CDLY(载体滞后时间)(CDLY <= TLS)
字符超时	WORD(0..65535)	0	0		设定以字符计的字符超时 当报文尾缀为字符超时时，必须为0
报文尾缀选择	Enumeration of BYTE	字符串(检查接收)	无		报文尾缀选择
报文尾缀字符	BYTE(0..255)	1	0		设定报文尾缀字符
报文尾缀值	WORD(0..65535)	16#0D	0		设定以ms或字符计的报文尾缀值
检查和	Enumeration of BYTE	无	无		选择检查和类型
波特率	Enumeration of DWORD	19200	19200	bits/s	设定以每秒位数计的波特率
寄偶性	Enumeration of BYTE	无	无		设定寄偶位类型
数据位	Enumeration of BYTE	8	8	位数/...	设定字符长度
停止位	Enumeration of BYTE	1	1		设定每个字符的停止位数 2表示当字符长度为5个位时1.5
配置错误时启...	Enumeration of BYTE	否	否		配置错误时启动PLC程序

5.3.3 COM_SET_PROT（更改串行接口的协议）

1. 功能：程序运行状态下，使串口在预定义的两个协议之间切换。
2. 属性：标准程序（PRG）。
3. 参数说明

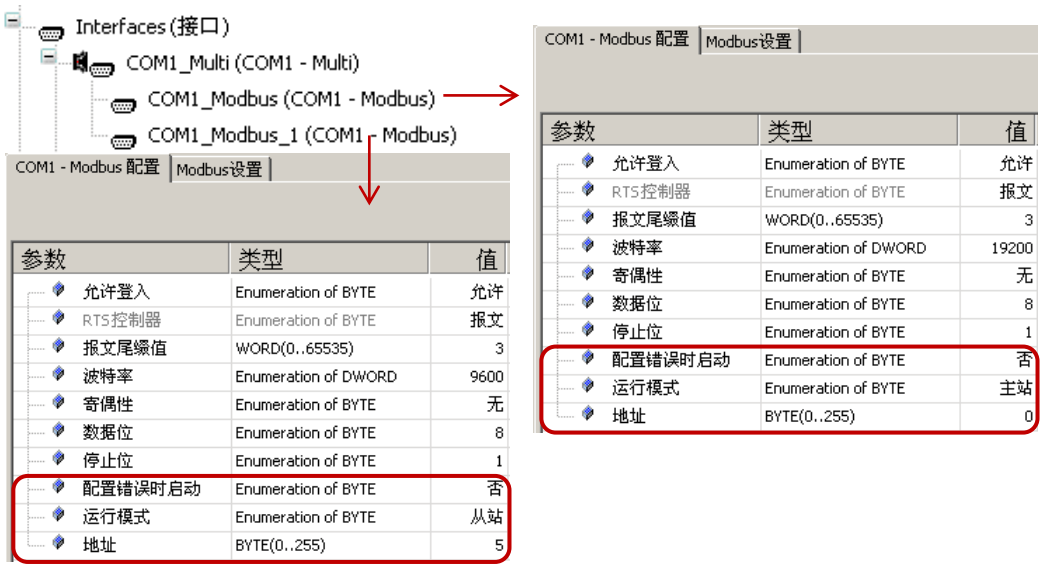


输入参数	数据类型	描述
EN	BOOL	使能功能块 FALSE：无效 TRUE：上升沿有效
COM	BYTE	指定串行接口的编号 1 ：COM1 接口 2 ：COM2 接口
IDX	BYTE	预定义协议的索引编号，从 0 起始
输出参数	**数据类型**	**描述**
DONE	BOOL	功能块完成标志，操作执行完毕为 TRUE，仅保持一个扫描周期
ERR	BOOL	功能块错误标志。该输出要与 DONE 一起综合判断。DONE 为 TRUE，且 ERR 为 TRUE，则说明操作完成但存在错误；DONE 为 TRUE，且 ERR 为 FALSE，则说明操作成功
ERNO	WORD	功能块操作存在错误的代码，代码含义请参见附录 A。只有 DONE 为 TRUE，且 ERR 也为 TRUE 时，ERNO 的输出值才有效

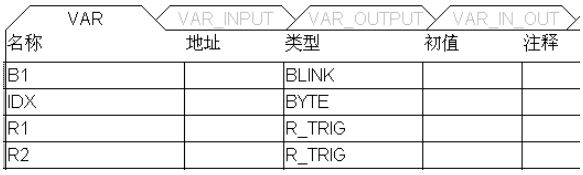
4. 描述

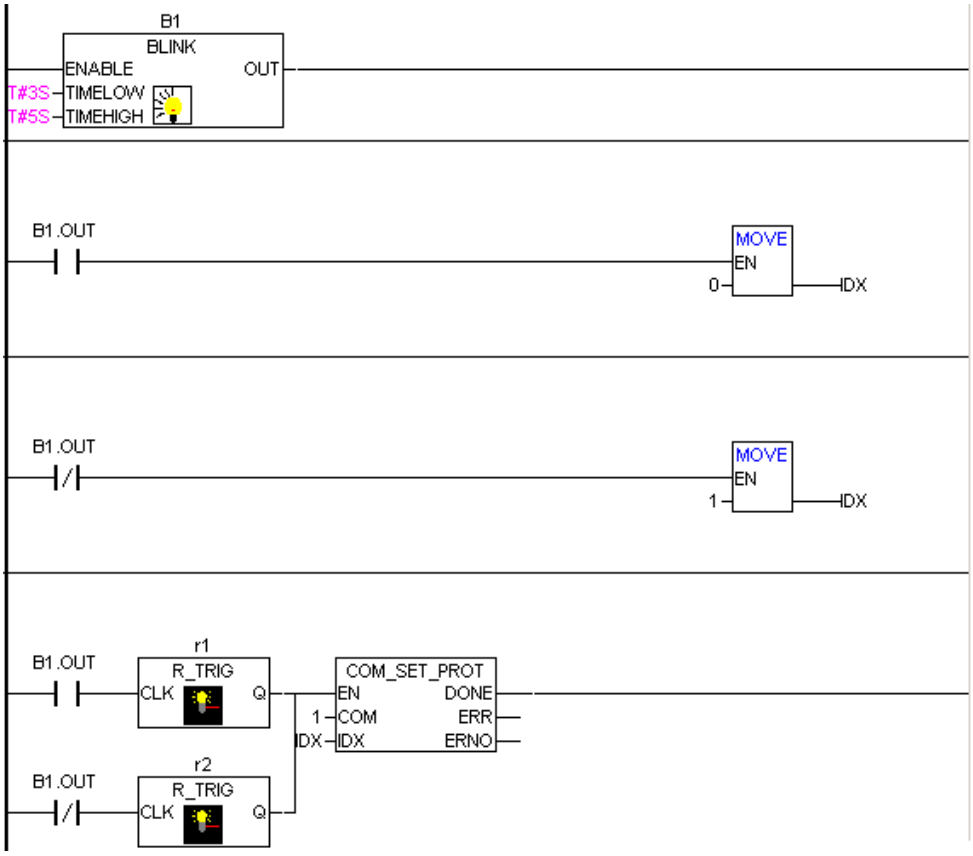
- INX （预定义协议的索引编号）

用户通过 CBP 将某串行接口设置为多协议（COMx - Multi），该接口便最多可支持两种协议模式。应对每一种协议的参数进行配置，如下例图。IDX=0 对应第一个协议，本例中为 Modbus 主站；IDX=1 对应第二个协议，本例中为 Modbus 从站。如果串口仅设置了一个协议，必须使 IDX =0。



5. 指令使用举例

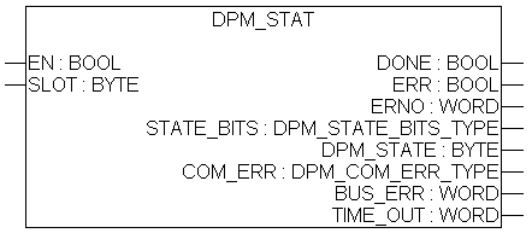




- 周期性修改 COM1 的协议参数。
- Modbus 主站持续 5S（B1 高电平），Modbus 从站持续 3S（B1 低电平）。

5.4 Profibus-DP 通讯指令（ASCII_AC500_V10.lib）

5.4.1 DPM_STAT（读取 DP 主站的诊断信息）



1. 功能：读取 DP 主站的诊断信息。
2. 类型：标准功能块（FB，含历史数据）。
3. 参数说明

输入参数	数据类型	描述
EN	BOOL	使能功能块　FALSE：无效 TRUE：高电平有效
SLOT	BYTE	通讯模块的插槽，CPU 集成接口为 0。扩展通讯模块从右向左计数，起始值为 1
输出参数	**数据类型**	**描述**
DONE	BOOL	功能块完成标志，操作执行完毕为 TRUE，仅保持一个扫描周期
ERR	BOOL	功能块错误标志。该输出要与 DONE 一起综合判断。DONE 为 TRUE，且 ERR 为 TRUE，则说明操作完成但存在错误；DONE 为 TRUE，且 ERR 为 FALSE，则说明操作成功
ERNO	WORD	功能块操作存在错误的代码，代码含义请参见附录 A。只有 DONE 为 TRUE，且 ERR 也为 TRUE 时，ERNO 的输出值才有效
STAT_BITS	DPM_STATE_BITS_TYPE	非典型通讯状态
DPM_STATE	BYTE	一般状态
COM_ERR	DPM_COM_ERR_TYPE	通讯故障信息
BUS_ERR	WORD	总线严重故障数量，如总线短路
TIME_OUT	WORD	超时故障数量

4. 描述

- STAT_BITS （非典型通讯状态）

DP 主站非典型通讯状态诊断数据，通过结构体 DPM_STATE_BITS_TYPE 描述，各成员含义如下：

```
TYPE DPM_STATE_BITS_TYPE:
STRUCT
        CTRL:BOOL;
        AUTO_CLR:BOOL;
        NO_EXCH:BOOL;
        FATAL:BOOL;
        EVENT:BOOL;
        TIMEOUT:BOOL;
```

```
END_STRUCT
END_TYPE
```

成员	数据类型	描述
CTRL	BOOL	TRUE：参数设置错误
AUTO_CLR	BOOL	TRUE：主站配置中激活了“自动清除模式”（Auto Clear Mode），该位才有效。如果 AUTO_CLR 为 TRUE，任意一个 DP 从站通讯期间故障，主站将停止与所有从站的数据交换，并切换回 CLEAR 状态
NO_EXCH	BOOL	TRUE：不能与一个或多个 DP 从站进行数据交换
FATAL	BOOL	TRUE：发生严重内部故障（例如总线短路）
EVENT	BOOL	TRUE：发生总线短路故障
TIMEOUT	BOOL	TRUE：发生超时故障

- DPM_STATE （一般状态）

输出 DPM_STATE，输出 DP 主站以下状态：

状态		含义
十进制 Dec	十六进制 Hex	
0	0	离线（OFFLINE）
64	40	停止（STOP）
128	80	清除（CLEAR）
192	C0	操作（OPERATE）

- DPM_STATE =OFFLINE

如果 DPM_STATE 的值为 0（OFFLINE），主站模块进入初始化状态。初始化完成后进入 STOP 状态。

- DPM_STATE =STOP

如果 DPM_STATE 为 64（STOP），说明主站模块已完成初始化。此时准备接收配置数据，与 DP 从站不进行数据交换。如果没有运行用户程序，主站模块将保持此状态。

- DPM_STATE =CLEAR

启动用户程序时，主站模块从 STOP 切换到 CLEAR 状态，通过总线开始对分配的 DP 从站进行参数化。成功参数化后，主站模块切换到 OPERATE 状态。如果参数化期间发生故障，主站模块将切换回 STOP 状态。

- DPM_STATE =OPERATE

通常情况下，用户程序运行后，主站模块为 OPERATE 状态。该状态下，DP 主站与 DP 从站交换 I/O 数据。如果该过程期间发生故障且配置时激活了“自动清除模式”（Auto Clear Mode），主站模块切换回 CLEAR 状态，尝试重新对 DP 从站进行参数化；如果未激活“自动清除模式”，主站模块保持 OPERATE 状态。如果用户程序停止，主站模块也切换回 STOP 状态。

- COM_ERR （通讯故障信息）

COM_ERR 通过结构体 DPM_COM_ERR_TYPE 详细定位通讯故障，各成员含义如下：

```
TYPE DPM_COM_ERR_TYPE:
STRUCT
        ADDRESS:BYTE;
        EVENT:BYTE;
END_STRUCT
END_TYPE
```

成员	数据类型	描述
ADDRESS	BYTE	故障设备的 DP 总线地址（0~125，255）。如果 ADDRESS 值为 255，说明错误存在于主站模块本身。
EVENT	BYTE	故障原因

- DDRESS =255（主站故障）

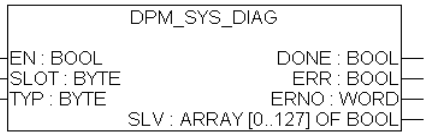
EVENT	含义	故障原因	修复方法
50	没有找到 USR_INTF 任务	DP 主站	联系 ABB 技术支持
51	没有全局数据域	DP 主站	联系 ABB 技术支持
52	没有找到 FDL 任务	DP 主站	联系 ABB 技术支持
53	没有找到 PLC 任务	DP 主站	联系 ABB 技术支持
54	没有主站参数记录	配置	创建工程配置数据，控制器重新加载程序
55	主站参数记录中有错误	配置	检查 DP 主站模块的配置，控制器重新加载程序
56	没有从站参数记录	配置	添加 DP 从站到配置数据中，控制器重新加载程序
57	从站参数记录中有错误	配置	检查所属的 DP 从站的配置，控制器重新加载程序
58	重复的从站地址	配置	检查所属的 DP 从站的总线地址，控制器重新加载程序
59	输出数据地址偏移错误	配置	检查所属的 DP 从站的的 IEC 地址，控制器重新加载程序
60	输入数据地址偏移错误	配置	检查所属的 DP 从站的的 IEC 地址，控制器重新加载程序
61	输出数据范围重叠	配置	检查所属的 DP 从站的的 IEC 地址，控制器重新加载程序
62	输入数据范围重叠	配置	检查所属的 DP 从站的的 IEC 地址，控制器重新加载程序
63	未知过程数据同步交换	CPU/ DP 主站	重新上电,或可联系 ABB
64	内存不够	DP 主站	联系 ABB 技术支持
65	错误的从站参数记录	配置	检查所属的 DP 从站的配置，控制器重新加载程序
202	无网段	DP 主站	联系 ABB 技术支持
212	数据库读取错误	配置/ DP 主站	控制器重新加载程序
213	操作系统传输结构错误	DP 主站	联系 ABB 技术支持

- DDRESS =0~125（从站故障）

EVENT	含义	故障原因	修复方法
2	用户报告溢出	DP 主站报文	检查所属的 DP 从站的配置，控制器重新加载程序
3	用户不支持请求的功能	DP 主站报文	根据 PROFIBUS 标准检查 DP 从站的符合性
9	响应报文中无数据	DP 从站	比较所属的 DP 从站的配置与硬件实际是否相同，控制器重新加载程序
17	用户无响应	DP 从站	检查 DP 从站总线地址
18	DP 主站不在逻辑令牌中	DP 主站	检查 DP 主站总线地址、其他 DP 主站系统中的最高站地址（HSA）、总线短路
21	请求报文存在参数错误	DP 主站报文	联系 ABB 技术支持

5.4.2 DPM_SYS_DIAG（读取所有 DP 从站状态）

1. 功能：检测所有 DP 从站的状态。
2. 类型：标准功能块（FB，含历史数据）。



3. 参数说明

输入参数	数据类型	描述
EN	BOOL	使能功能块　FALSE：无效 TRUE：高电平有效
SLOT	BYTE	通讯模块的插槽，CPU 集成接口为 0。扩展通讯模块从右向左计数，起始值为 1
TYP	BYTE	检测类型
输出参数	**数据类型**	**描述**
DONE	BOOL	功能块完成标志，操作执行完毕为 TRUE，仅保持一个扫描周期
ERR	BOOL	功能块错误标志。该输出要与 DONE 一起综合判断。DONE 为 TRUE，且 ERR 为 TRUE，则说明操作完成但存在错误；DONE 为 TRUE，且 ERR 为 FALSE，则说明操作成功
ERNO	WORD	功能块操作存在错误的代码，代码含义请参见附录 A。只有 DONE 为 TRUE，且 ERR 也为 TRUE 时，ERNO 的输出值才有效
SLV	ARRAY[0..127] OF BOOL	从站检测结果

4. 描述

- TYP （检测类型）

输入 TYP 用于选择对 DP 从站进行的检测类型。

➢ TYP =1（配置检测）

输出 SLV 显示哪些 DP 从站被 DP 主站已成功配置。请注意，当生成配置数据时，DP 主站只对分配给该主站的 DP 从站进行设置，令其进入运行状态。

➢ TYP =2（数据交换检测）

输出 SLV 显示哪些 DP 从站与主站交换数据。DP 主站只与被其配置的从站进行数据交换。只有主站为 OPERATE 状态时，才能请求数据交换检测。

➢ TYP =3（诊断信息检测）

输出 SLV 指示哪些 DP 从站的诊断信息已发生变化。只有主站为 OPERATE 状态时，才能请求诊断信息检测。

- SLV （DP 从站检测结果）

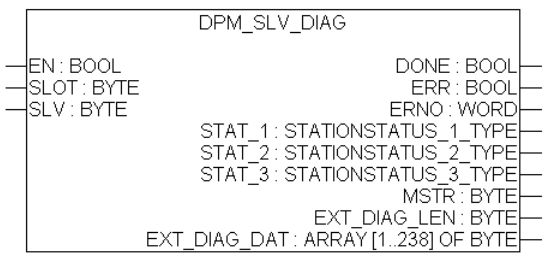
SLV 以位形式输出检测结果。数组中的每个位代表一个 DP 从站，索引号码对应于 DP 从站的总线地址。当某一位为 TRUE 时，则说明其相应的 DP 从站符合 TYP 选择的检测条件。

例如，如果 TYP=1 且 SLV[2]=TRUE，则说明总线地址为 2 的 DP 从站被主站成功配置。如果 SLV[2]=FALSE，则说明此 DP 从站的配置还未完成或此从站不属于该 DP 主站。

如果 TYP=2 且 SLV[2]=TRUE，则说明总线地址为 2 的 DP 从站与主站交换数据。

如果 TYP=3，且 SLV[2]=TRU，则说明总线地址为 2 的 DP 从站的诊断信息已发生变化，可通过功能块 DNM_SLV_DIAG 读取详细的诊断结果。

5.4.3 DPM_SLV_DIAG（读取 DP 从站的诊断信息）



1. 功能：读取指定 DP 从站的详细诊断信息。
2. 类型：标准功能块（FB，含历史数据）。
3. 参数说明

输入参数	数据类型	描述
EN	BOOL	使能功能块　FALSE：无效 TRUE：上升沿有效
SLOT	BYTE	通讯模块的插槽，CPU 集成接口为 0。扩展通讯模块从右向左计数，起始值为 1
SLV	BYTE	需要诊断的 DP 从站地址
输出参数	**数据类型**	**描述**
DONE	BOOL	功能块完成标志，操作执行完毕为 TRUE，仅保持一个扫描周期
ERR	BOOL	功能块错误标志。该输出要与 DONE 一起综合判断。DONE 为 TRUE，且 ERR 为 TRUE，则说明操作完成但存在错误；DONE 为 TRUE，且 ERR 为 FALSE，则说明操作成功
输出参数	**数据类型**	**描述**
ERNO	WORD	功能块操作存在错误的代码，代码含义请参见附录 A。只有 DONE 为 TRUE，且 ERR 也为 TRUE 时，ERNO 的输出值才有效
STAT_1	STATIONSTATUS_1_TYPE	诊断数据_第 1 字节。DONE=TRUE 且 ERR=FALSE，STAT_1 才有效
STAT_2	STATIONSTATUS_2_TYPE	诊断数据_第 2 字节。DONE=TRUE 且 ERR=FALSE，STAT_1 才有效
STAT_3	STATIONSTATUS_3_TYPE	诊断数据_第 3 字节。DONE=TRUE 且 ERR=FALSE，STAT_1 才有效
MSTR	BYTE	相关 DP 主站的总线地址
EXT_DIAG_LEN	BYTE	扩展诊断数据的长度。如果 EXT_DIAG_LEN=0，说明没有扩展的诊断数据
EXT_DIAG_DAT	ARRAY[1..238] OF BYTE	扩展诊断数据。DONE=TRUE、ERR=FALSE 且 EXT_DIAG_LEN>0 时，EXT_DIAG_DAT 中的数据才有效

4. 描述

- STAT_1 （诊断数据_第 1 字节）

诊断数据的第一个字节通过结构体 STATIONSTATUS_1_TYPE 描述，各成员含义如下：

```
TYPE STATIONSTATUS_1_TYPE:
STRUCT
        NON_EXISTENT:BOOL;
        NOT_READY:BOOL;
        CFG_FAULT:BOOL;
        EXT_DIAG:BOOL;
        NOT_SUPPORTED:BOOL;
        INVALID_RESPONSE:BOOL;
        PRM_FAULT:BOOL;
        MASTER_LOCK:BOOL;
END_STRUCT
END_TYPE
```

成员	数据类型	描述
NON_EXISTENT	BOOL	TRUE：该从站不存在
NOT_READY	BOOL	TRUE：该从站未准备好进行 I/O 数据交换
CFG_FAULT	BOOL	TRUE：配置数据与 DP 从站的实际配置不匹配
EXT_DIAG	BOOL	TRUE：EXT_DIAG_DAT 中有该从站的扩展诊断信息
NOT_SUPPORTED	BOOL	TRUE：该从站不支持所请求的功能
INVALID_RESPONSE	BOOL	TRUE：无效的从站响应。由主站设置
PRM_FAULT	BOOL	TRUE：参数设置错误
MASTER_LOCK	BOOL	TRUE：该从站被其他主站锁定，MSTR 输出此主站的总线地址

- STAT_2 （诊断数据_第 2 字节）

诊断数据的第二个字节通过结构体 STATIONSTATUS_2_TYPE 描述，各成员含义如下：

```
TYPE STATIONSTATUS_2_TYPE:
STRUCT
        PRM_REQ:BOOL;
        STAT_DIAG:BOOL;
        DP_Slave:BOOL;
        WD_ON:BOOL;
        FREEZE_MODE:BOOL;
        SYNC_MODE:BOOL;
        reserved:BOOL;
        DEACTIVATED:BOOL;
END_STRUCT
END_TYPE
```

成员	数据类型	描述
PRM_REQ	BOOL	TRUE：该从站必须重新设置参数
STAT_DIAG	BOOL	TRUE：该从站为静态诊断
DP_Slave	BOOL	始终为 TRUE
WD_ON	BOOL	TRUE：监视狗（watchdog）功能激活
FREEZE_MODE	BOOL	TRUE：该从站处于冻结（Freeze）模式
SYNC_MODE	BOOL	TRUE：该从站处于同步（Sync）模式
reserved	BOOL	保留
DEACTIVATED	BOOL	TRUE：该从站未被主站激活，不进行循环 I/O 数据交换。由主站设置

- STAT_3 （诊断数据_第 3 字节）

诊断数据的第三个字节通过结构体 STATIONSTATUS_3_TYPE 描述，各成员含义如下：

```
TYPE STATIONSTATUS_3_TYPE:
STRUCT
        reserved0:BOOL;
        reserved1:BOOL;
        reserved2:BOOL;
        reserved3:BOOL;
```

```
        reserved4:BOOL;
        reserved5:BOOL;
        reserved6:BOOL;
        EXT_DIAG_OVERFLOW:BOOL;
END_STRUCT
END_TYPE
```

成员	数据类型	描述
Reserved0	BOOL	保留
Reserved1	BOOL	保留
Reserved2	BOOL	保留
Reserved3	BOOL	保留
Reserved4	BOOL	保留
Reserved5	BOOL	保留
Reserved6	BOOL	保留
DEACTIVATED	BOOL	TRUE：诊断数据溢出，从站诊断数据过多

- EXT_DIAG_DAT （扩展诊断数据）

每个 DP 从站必须支持以上的标准诊断数据，某些从站还可能提供扩展的诊断数据。Profibus-DP 标准对扩展诊断数据的格式进行了定义，用户依此获得所需的诊断信息。但是由于扩展诊断数据包含供应商专用项，因此功能块 DM_SLV_DIAG 并不能进行任何自定义数据的处理。

输出 EXT_DIAG_LEN 指示完整的扩展诊断数据的有效长度。当处理诊断时，只考虑包含在范围 EXT_DIAG_DAT[1]到 EXT_DIAG_DAT[EXT_DIAG_LEN]的数据。

扩展诊断数据由设备诊断信息、模块诊断信息和通道诊断信息 3 部分组成。

- 设备诊断信息

设备诊断部分包含了一般的诊断信息，如失电。数据由头字节和诊断数据组成。头字节的最高两位固定为 00，较低的 6 个位说明设备诊断数据的字节长度，该长度包括头字节本身。

头字节	设备相关的诊断数据字节
00XXXXXX	1~62 字节（包括头字节有最多 63 个字节）

设备相关的诊断数据由供应商定义，其含义的详细信息可以参见各设备的说明文档。

- 模块诊断信息

模块诊断部分包含了直接分配给设备特定 I/O 模块的诊断信息。数据由头字节和诊断数据组成。头字节的最高两位固定为 01，较低的 6 个位说明模块诊断数据的字节长度，该长度包括头字节本身。

头字节	模块相关的诊断数据字节
01XXXXXX	1~62 字节（包括头字节有最多 63 个字节）

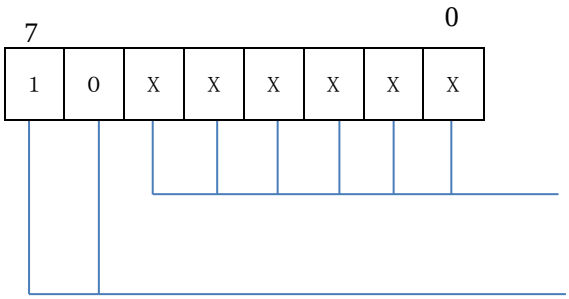
模块相关的诊断数据字节中的每一个位分配给一个模块。模块索引通过数据中的位偏移来表示，某 I/O 模块有诊断数据其对应位被置为 TRUE。

- 通道诊断信息

每个通道诊断信息由 3 个字节组成，整个信息可能包含多个这样的通道诊断信息，一个通道诊断信息由头字节和诊断数据组成。

头字节	通道相关的诊断数据字节
10XXXXXX	2 字节（包括头字节有 3 个字节）

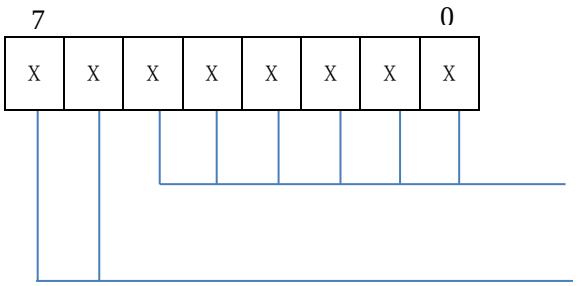
头字节的最高两位固定为 10，较低的 6 位说明具有诊断数据的模块号。



故障模块号，范围 000000~111111。
例如 000011 说明模块 3 有故障。

固定为 10，指明是通道诊断信息

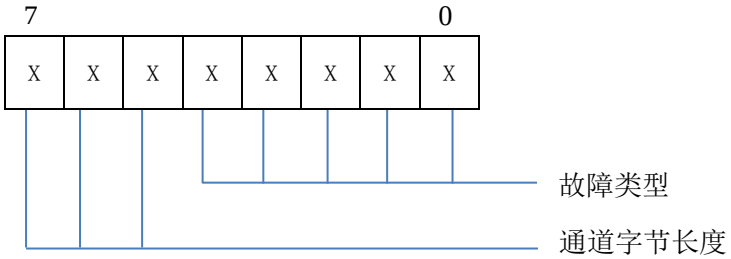
第 2 个字节最高两位说明通道字节类型，较低的 6 位说明故障通道号。



故障通道号，范围 000000~111111。
例如 000101 说明通道 5 有故障。

01：输入通道
10：输出通道
11：输入/输出通道

第 3 个字节较高的 3 位说明通道字节长度，较低的 5 位说明故障类型。



较高 3 位通道字节长度的具体含义如下：

0	0	0	保留
0	0	1	1 位（Bit）
0	1	0	2 位（2 Bit）
0	1	1	4 位（4 Bit）
1	0	0	字节（Byte）
1	0	1	字（Word）
1	1	0	双字（2 Word）
1	1	1	保留

较低 5 位故障类型的编码如下：

0	保留	0	0	0	0	0
1	短路	0	0	0	0	1
2	欠压	0	0	0	1	0
3	过压	0	0	0	1	1
4	过载	0	0	1	0	0
5	过温	0	0	1	0	1
6	电缆断	0	0	1	1	0
7	超过上限	0	0	1	1	1
8	低于下限	0	1	0	0	0
9	一般故障	0	1	0	0	1
10	保留	0	1	0	1	0
:	:	0	1	x	x	x
15	保留	0	1	1	1	1
16	供应商专用	1	0	0	0	0
:	:	1	x	x	x	x
31	供应商专用	1	1	1	1	1

- 扩展诊断数据分析举例

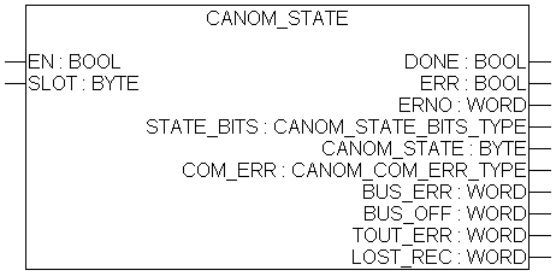
例如：EXT_DIAG_LEN =15

分析诊断数据时，只考虑包含在范围EXT_DIAG_DAT[1]到EXT_DIAG_DAT[EXT_DIAG_LEN]的数据。

字节	7	6	5	4	3	2	1	0	含义
EXT_DIAG_DAT[1]	0	0	0	0	0	1	0	0	设备相关的诊断；长度为 4 个字节，包括头字节
EXT_DIAG_DAT[2]	X	x	x	X	X	x	X	X	设备诊断的 3 个字节
EXT_DIAG_DAT[3]	X	x	x	X	X	x	X	X	
EXT_DIAG_DAT[4]	X	x	x	X	X	x	X	X	
EXT_DIAG_DAT[5]	0	1	0	0	0	1	0	1	模块相关的诊断；长度为 5 个字节，包括头字节
EXT_DIAG_DAT[6]	0（7）	0（6）	0（5）	0（4）	0（3）	0（2）	0（1）	1（0）	模块 0 有诊断信息
EXT_DIAG_DAT[7]	0（15）	0（14）	0(13)	0(12)	0(11)	0(10)	0（9）	0（8）	
EXT_DIAG_DAT[8]	0（23）	0（22）	0(21)	0(20)	0(19)	0(18)	0（17）	0(16)	
EXT_DIAG_DAT[9]	0（31）	1（30）	0(29)	0(28)	0(27)	0(26)	0（25）	0(24)	模块 30 有诊断信息
EXT_DIAG_DAT[10]	1	0	0	0	0	0	0	0	模块 0 通道相关的诊断
EXT_DIAG_DAT[11]	0	1	0	0	0	0	1	0	通道 2，输入
EXT_DIAG_DAT[12]	0	0	1	0	0	1	0	0	过载，位通道
EXT_DIAG_DAT[13]	1	0	0	1	1	1	1	0	模块 30 通道相关的诊断
EXT_DIAG_DAT[14]	1	0	0	0	0	1	1	0	通道 6，输出
EXT_DIAG_DAT[15]	1	0	1	0	0	1	1	1	超过上限，字通道

5.5 CANopen 通讯指令（CANopen_AC500_V11.lib）

5.5.1 CANOM_STATE（读取 CANopen 主站的诊断信息）



1. 功能：读取 CANopen 主站的诊断信息。
2. 类型：标准功能块（FB，含历史数据）。
3. 参数说明

输入参数	数据类型	描述
EN	BOOL	使能功能块　FALSE：无效 TRUE：高电平有效
SLOT	BYTE	通讯模块的插槽，CPU 集成接口为 0。扩展通讯模块从右向左计数，起始值为 1
输出参数	**数据类型**	**描述**
DONE	BOOL	功能块完成标志，操作执行完毕为 TRUE，仅保持一个扫描周期
ERR	BOOL	功能块错误标志。该输出要与 DONE 一起综合判断。DONE 为 TRUE，且 ERR 为 TRUE，则说明操作完成但存在错误；DONE 为 TRUE，且 ERR 为 FALSE，则说明操作成功
ERNO	WORD	功能块操作存在错误的代码，代码含义请参见附录 A。只有 DONE 为 TRUE，且 ERR 也为 TRUE 时，ERNO 的输出值才有效
输出参数	**数据类型**	**描述**
STAT_BITS	CANOM_STATE_BITS_TYPE	非典型通讯状态
CANOM_STATE	BYTE	一般状态
CANOM_ERR	CANOM_COM_ERR_TYPE	通讯故障信息
BUS_ERR	WORD	总线故障数量。内部错误帧计数器超过了特定的值，将出现总线错误。可通过功能块 CANOM_RES_ERR 复位
BUS_OFF	WORD	总线脱离数量。可通过功能块 CANOM_RES_ERR 复位
TOUT_ERR	WORD	发送丢失的 CAN 消息的数量，如果报文不能在 20ms 内完成发送，则发送任务失败。可通过功能块 CANOM_RES_ERR 复位
LOST_REC	WORD	接收丢失的 CAN 消息的数量。可通过功能块 CANOM_RES_ERR 复位

4. 描述

- STAT_BITS （非典型通讯状态）

CANopen 主站非典型通讯状态诊断数据，通过结构体 CANOM_STATE_BITS_TYPE 描述，各成员含义如下：

```
TYPE CANOM_STATE_BITS_TYPE:
STRUCT
    CTRL:BOOL;
    AUTO_CLR:BOOL;
    NO_EXCH:BOOL;
    FATAL:BOOL;
    EVENT:BOOL;
    reserved1:BOOL;
    TIMEOUT:BOOL;
    reserved2:BOOL;
END_STRUCT
END_TYPE
```

成员	数据类型	描述
CTRL	BOOL	TRUE：参数设置错误
AUTO_CLR	BOOL	TRUE：主站配置中激活了“自动清除模式”（Auto Clear Mode），该位才有效。如果 AUTO_CLR 为 TRUE，任意一个从站通讯期间故障，主站将停止与所有从站的数据交换，并切换回 CLEAR 状态
NO_EXCH	BOOL	TRUE：不能与一个或多个从站进行数据交换
FATAL	BOOL	TRUE：发生严重内部故障
EVENT	BOOL	TRUE：发生传输故障，在相应的输出 BUS_ERR 和 BUS_OFF 显示发生传输故障的数量。如果 EVENT 位为 TRUE，只能通过功能块 CANOM_RES_ERR 进行复位
reserved1	BOOL	保留
TIMEOUT	BOOL	TRUE: 发生超时故障，在相应的输出 TOUT_ERR 显示发生超时错误的数量。如果 TIMEOUT 位为 TRUE，只能通过功能块 CANOM_RES_ERR 进行复位
reserved2	BOOL	保留

- CANOM_STATE （一般状态）

输出 CANOM_STATE，输出 CANopen 主站以下状态：

O 状态		含义
十进制 Dec	十六进制 Hex	
0	0	离线（OFFLINE）
64	40	停止（STOP）
128	80	清除（CLEAR）
192	C0	操作（OPERATE）

- CANOM_STATE =OFFLINE

如果 DPM_STATE 的值为 0（OFFLINE），主站模块进入初始化状态。初始化完成后进入 STOP 状态。

- CANOM_STATE =STOP

如果 DPM_STATE 为 64（STOP），说明主站模块已完成初始化。此时准备接收配置数据，与从站不进行数据交换。如果没有运行用户程序，主站模块将保持此状态。

- CANOM_STATE =CLEAR

启动用户程序时，主站模块从 STOP 切换到 CLEAR 状态，通过总线开始对分配的从站进行参数化。成功参数化后，主站模块切换到 OPERATE 状态。如果参数化期间发生错误，主站模块将切换回 STOP 状态。

- CANOM_STATE =OPERATE

通常情况下，用户程序运行后，主站模块为 OPERATE 状态。该状态下，主站与从站交换 I/O 数据。如果该过程期间发生错误且配置时激活了“自动清除模式”（Auto Clear Mode），主站模块切换回 CLEAR 状态，尝试重新对从站进行参数化；如果未激活“自动清除模式”，主站模块保持 OPERATE 状态。如果用户程序停止，主站模块也切换回 STOP 状态。

- COM_ERR （通讯故障信息）

COM_ERR 通过结构体 CANOM_COM_ERR_TYPE 详细定位通讯故障，各成员含义如下：

```
TYPE CANOM_COM_ERR_TYPE:
STRUCT
        ADDRESS:BYTE;
        EVENT:BYTE;
END_STRUCT
END_TYPE
```

成员	数据类型	描述
ADDRESS	BYTE	故障设备的节点地址。如果 ADDRESS 值为 255，说明错误存在于主站模块本身
EVENT	BYTE	故障原因

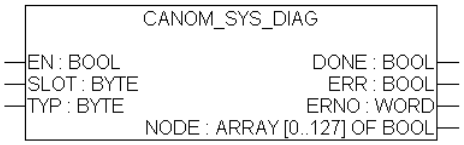
- DDRESS =255（主站故障）

EVENT	含义	故障原因	修复方法
52	未知的握手模式过程数据	配置	联系 ABB 技术支持
56	无效的数据传输速率	配置	联系 ABB 技术支持
60	节点地址配置重复	配置	检查所有设备配置的节点地址
210	无配置数据	配置/主站	加载配置数据到主站模块
212	数据库读取错误	配置/主站	重新加载配置数据到主站模块，联系 ABB 技术支持
220	看门狗错误	主站/CPU	联系 ABB 技术支持

- DDRESS≠255（节点故障）

EVENT	含义	故障原因	修复方法
30	从站监视超时	从站	检查从站的连接和节点地址
31	从站终止运行模式	从站	复位从站
32	节点保护协议中顺序错误	从站	复位从站
33	配置远程 PDO 无响应	从站	检查从站是否支持远程帧
34	配置期间从站无响应	从站	检查从站是否连接并准备好运行
35	从站的配置与硬件实际不同	配置	检查从站的配置
36	从站的配置与设备类型不同	配置	检查从站支持的服务
37	收到未知的 SDO 响应	从站	从站不满足 CiA 协议规范
38	接收的 SDO 响应的长度不等于 8	从站	从站不满足 CiA 协议规范
39	从站没有运行	配置/主站	取消“自动清除模式”（Auto Clear Mode）

5.5.2 CANOM_SYS_DIAG（读取所有 CANopen 从站状态）



1. 功能：检测所有 CANopen 从站（节点）的状态。
2. 类型：标准功能块（FB，含历史数据）。
3. 参数说明

输入参数	数据类型	描述
EN	BOOL	使能功能块　FALSE：无效 TRUE：高电平有效
SLOT	BYTE	通讯模块的插槽，CPU 集成接口为 0。扩展通讯模块从右向左计数，起始值为 1
TYP	BYTE	检测类型
输出参数	**数据类型**	**描述**
DONE	BOOL	功能块完成标志，操作执行完毕为 TRUE，仅保持一个扫描周期
ERR	BOOL	功能块错误标志。该输出要与 DONE 一起综合判断。DONE 为 TRUE，且 ERR 为 TRUE，则说明操作完成但存在错误；DONE 为 TRUE，且 ERR 为 FALSE，则说明操作成功
ERNO	WORD	功能块操作存在错误的代码，代码含义请参见附录 A。只有 DONE 为 TRUE，且 ERR 也为 TRUE 时，ERNO 的输出值才有效
NODE	ARRAY[0..127] OF BOOL	从站检测结果

4. 描述

- TYP （检测类型）

输入 TYP 用于选择对从站进行的检测类型。

- TYP =1（配置检测）

输出 SLV 显示哪些从站已被主站成功配置。主站只与由该主站配置的从站建立连接。

- TYP =2（运行检测）

输出 SLV 显示哪些从站没有错误，并正在运行。只有主站为 OPERATE 状态时，才能请求运行检测。

- TYP =3（诊断信息检测）

输出 SLV 指示哪些从站的诊断信息已发生变化。只有主站为 OPERATE 状态时，才能请求诊断信息检测。

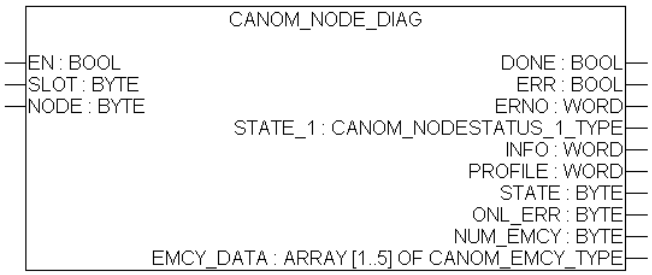
- NODE （从站检测结果）

NODE 以位形式输出检测结果。数组中的每个位代表一个从站（节点），索引号码对应于从站的节点地址。当某一位为 TRUE 时，则说明其相应的从站符合 TYP 选择的检测条件。

例如，如果 TYP=1 且 SLV[2]=TRUE，则说明节点地址为 2 的从站被主站成功配置。如果 SLV[2]=FALSE，则说明此从站的配置还未完成或此从站不是被该主站配置的。

如果 TYP=3，且 SLV[2]=TRUE，则说明节点地址为 2 的从站的诊断信息已发生变化，可通过功能块 CANOM_NODE_DIAG 读取详细的诊断结果。

5.5.3 CANOM_NODE_DIAG（读取 CANopen 从站的诊断信息）



1. 功能：读取指定 CANopen 从站（节点）的详细诊断信息。
2. 类型：标准功能块（FB，含历史数据）。
3. 参数说明

输入参数	数据类型	描述
EN	BOOL	使能功能块　FALSE：无效 TRUE：上升沿有效
SLOT	BYTE	通讯模块的插槽，CPU 集成接口为 0。扩展通讯模块从右向左计数，起始值为 1
NODE	BYTE	需要诊断的 CANopen 从站（节点）
输出参数	**数据类型**	**描述**
DONE	BOOL	功能块完成标志，操作执行完毕为 TRUE，仅保持一个扫描周期
ERR	BOOL	功能块错误标志。该输出要与 DONE 一起综合判断。DONE 为 TRUE，且 ERR 为 TRUE，则说明操作完成但存在错误；DONE 为 TRUE，且 ERR 为 FALSE，则说明操作成功
ERNO	WORD	功能块操作存在错误的代码，代码含义请参见附录 A。只有 DONE 为 TRUE，且 ERR 也为 TRUE 时，ERNO 的输出值才有效
STAT_1	CANOM_NODESTATUS_1_TYPE	诊断数据
INFO	WORD	根据 CiA 规范对象 16#1000 的附加信息
PROFILE	WORD	根据 CiA 规范对象 16#1000 的配置文件编号
STATE	BYTE	从站的运行状态
ONL_ERR	BYTE	从站的联机错误
NUM_EMCY	BYTE	紧急消息的数量
EMCY_DATA	ARRAY[1...5] OF CANOM_EMCY_TYPE	紧急消息的内容

4. 描述

- STAT_1 （诊断数据）

诊断数据结构体 CANOM_NODESTATUS_1_TYPE 各成员含义如下：

```
TYPE CANOM_NODESTATUS_1_TYPE:
STRUCT
        NO_RESPONSE:BOOL;
        EMCY_OVF:BOOL;
        PRM_FAULT:BOOL;
        GUARD_ACT:BOOL;
```

```
        reserved1:BOOL;
        reserved2:BOOL;
        reserved3:BOOL;
        DEACTIVATED:BOOL;
END_STRUCT
END_TYPE
```

成员	数据类型	描述
NO_RESPONSE	BOOL	TRUE：该节点无响应
EMCY_OVF	BOOL	TRUE：该节点紧急消息的数量超过最大缓冲数量
PRM_FAULT	BOOL	TRUE：该节点配置数据与从站的实际配置不匹配
GUARD_ACT	BOOL	TRUE：该节点的保护服务已激活
reserved1	BOOL	保留
reserved2	BOOL	保留
reserved3	BOOL	保留
DEACTIVATED	BOOL	TRUE：该节点在配置中未激活，不进行处理

- EMCY_DATA （紧急消息的内容）

EMCY_DATA 输出从站节点最后 5 个（参见 NUM_EMCY）紧急消息的内容。EMCY_DATA 由结构体数组 CANOM_EMCY_TYPE ARRAY [1..5]组成。CANOM_EMCY_TYPE 各成员含义如下：

```
TYPE CANOM_EMCY_TYPE:
STRUCT
        ERROR_CODE:WORD;
        ERROR_REG:BYTE;
        ERROR_DATA:ARRAY[1..5] OF BYTE;
END_STRUCT
END_TYPE
```

- ERROR_CODE （故障代码）

紧急消息包含以下设备规范中定义的故障代码：

故障代码		故障原因
十进制 Dec	十六进制 Hex	
00000...00255	0000...00FF	故障已复位或无错误
04096...04351	1000...10FF	一般故障
08192...08447	2000...20FF	电流错误
08448...08703	2100...21FF	设备输入侧故障
08704...08959	2200...22FF	设备内部故障
08960...09215	2300...23FF	设备输出侧故障
12288...12543	3000...30FF	电压错误
12544...12799	3100...31FF	电源电压故障
12800...13055	3200...32FF	设备内部故障
13056...13311	3300...33FF	设备输出侧故障
16384...16639	4000...40FF	温度错误

故障代码		故障原因
16640...16895	4100...41FF	环境温度
16896...17151	4200...42FF	设备内部温度
20480...20735	5000...50FF	设备硬件故障
24576...24831	6000...60FF	设备软件故障
24832...25087	6100...61FF	设备内部软件故障
25088...25343	6200...62FF	应用软件故障
25344...25599	6300...63FF	数据错误
28672...28927	7000...70FF	附加的模块故障
32768...33023	8000...80FF	监视故障
33024...33279	8100...81FF	通讯故障
36864...37119	9000...90FF	外部故障
61440...61695	F000...F0FF	附加的功能错误
65280...65535	FF00...FFFF	设备规格错误

- ERROR_REG （故障寄存器对象）

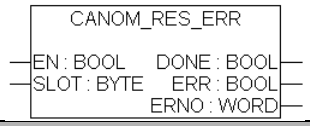
显示从站故障寄存器的内容（对象 1001 hex），该值作为紧急消息的一部分由从站传输。

- ERROR_DATA （故障信息）

应用时，ERROR_DATA 用于输出供应商定义的故障信息，作为紧急消息的一部分由从站传输。关于数据的详细信息可以参见特殊设备的文档。

5.5.4 CANOM_RES_ERR （复位 CANopen 模块）

1. 功能：复位通讯模块的内部故障和故障计数器。
2. 类型：标准功能块（FB，含历史数据）。
3. 参数说明



输入参数	数据类型	描述
EN	BOOL	使能功能块 FALSE：无效 TRUE：上升沿有效
SLOT	BYTE	通讯模块的插槽，CPU 集成接口为 0。扩展通讯模块从右向左计数，起始值为 1。
输出参数	**数据类型**	**描述**
DONE	BOOL	功能块完成标志，操作执行完毕为 TRUE，仅保持一个扫描周期
ERR	BOOL	功能块错误标志。该输出要与 DONE 一起综合判断。DONE 为 TRUE，且 ERR 为 TRUE，则说明操作完成但存在错误；DONE 为 TRUE，且 ERR 为 FALSE，则说明操作成功
ERNO	WORD	功能块操作存在错误的代码，代码含义请参见附录 A。只有 DONE 为 TRUE，且 ERR 也为 TRUE 时，ERNO 的输出值才有效

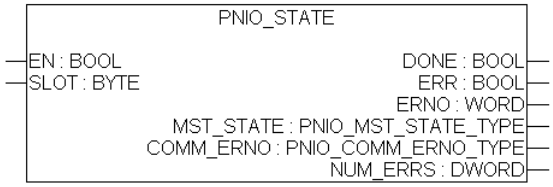
4. 描述

通过功能块 CANOM_RES_ERR，可以复位 CANopen 主站模块的内部故障和故障计数器，同时复位功能块 CANOM_STAT 的以下故障输出指示：

- STATE_BITS.EVENT
- STATE_BITS.TIMEOUT
- BUS_ERR
- BUS_OFF
- TOUT_ERR
- LOST_REC

5.6 PROFINET 通讯指令（PROFINET_AC500_V13.lib）

5.6.1 PNIO_STAT（读取 PROFINET 主站及总线的诊断信息）



1. 功能：读取 PROFINET 主站模块及总线的一般状态。
2. 类型：标准功能块（FB，含历史数据）。
3. 参数说明

输入参数	数据类型	描述
EN	BOOL	使能功能块　FALSE：无效 TRUE：高电平有效
SLOT	BYTE	通讯模块的插槽，CPU 集成接口为 0。扩展通讯模块从右向左计数，起始值为 1
输出参数	**数据类型**	**描述**
DONE	BOOL	功能块完成标志，操作执行完毕为 TRUE，仅保持一个扫描周期
ERR	BOOL	功能块错误标志。该输出要与 DONE 一起综合判断。DONE 为 TRUE，且 ERR 为 TRUE，则说明操作完成但存在错误；DONE 为 TRUE，且 ERR 为 FALSE，则说明操作成功
ERNO	WORD	功能块操作存在错误的代码，代码含义请参见附录 A。只有 DONE 为 TRUE，且 ERR 也为 TRUE 时，ERNO 的输出值才有效
MST_STATE	PNIO_MST_STATE_TYPE	PROFINET 主站模块的状态
COMM_ERNO	PNIO_COMM_ERNO_TYPE	PROFINET 总线状态
NUM_ERRS	DWORD	电源接通后的错误数量

4. 描述

- MST_STATE （PROFINET 主站模块的状态）

输出 MST_STATE 指示 PROFINET 主站模块的状态，借助枚举 PNIO_MST_STATE_TYPE 以纯文本形式显示。

值	含义
PNIO_MST_STATE_UNKOWN	未知状态
PNIO_MST_STATE_NOT_CONFIGURED	预配置
PNIO_MST_STATE_STOP	已配置但没有运行
PNIO_MST_STATE_IDLE	空闲
PNIO_MST_STATE_OPERATE	运行状态

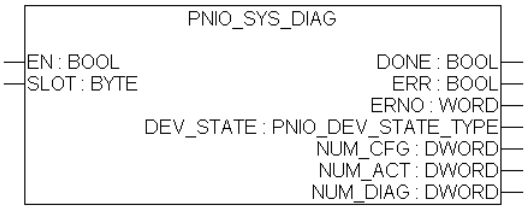
- COMM_ERNO （PROFINET 总线状态）

输出 COMM_ERNO 指示当前 PROFINET 总线的状态，借助枚举 PNIO_COMM_ERNO_TYPE.，以纯文本形式显示。

值	含义
PNIO_COMM_ERNO_NONE	无错误
PNIO_CONIFGURATION_FAULT	配置失败
PNIO_PARAMETER_ERROR	参数错误
PNIO_INV_NETWORK_ERROR	无效的网络地址
PNIO_NETWORK_FAULT	网络故障
PNIO_CONNECTION_CLOSED	连接关闭
PNIO_CONNECTION_TIMEOUT	连接超时
PNIO_LONELY_NETWORK	单个网络
PNIO_DUBLICATE_NODE	重复节点
PNIO_CABLE_DISCONNECTED	电缆断开
PNIO_COMM_ERNO_UNKNOWN	未知的通讯错误

5.6.2 PNIO_SYS_DIAG（读取所有 PROFINET IO 设备的状态）

1. 功能：读取所有 PROFINET IO 设备的状态。
2. 类型：标准功能块（FB，含历史数据）。
3. 参数说明



输入参数	数据类型	描述
EN	BOOL	使能功能块　FALSE：无效 TRUE：高电平有效
SLOT	BYTE	通讯模块的插槽，CPU 集成接口为 0。扩展通讯模块从右向左计数，起始值为 1
输出参数	**数据类型**	**描述**
DONE	BOOL	功能块完成标志，操作执行完毕为 TRUE，仅保持一个扫描周期
ERR	BOOL	功能块错误标志。该输出要与 DONE 一起综合判断。DONE 为 TRUE，且 ERR 为 TRUE，则说明操作完成但存在错误；DONE 为 TRUE，且 ERR 为 FALSE，则说明操作成功
ERNO	WORD	功能块操作存在错误的代码，代码含义请参见附录 A。只有 DONE 为 TRUE，且 ERR 也为 TRUE 时，ERNO 的输出值才有效
DEV_STATE	PNIO_DEV_STATE_TYPE	所有 PROFINET IO 设备的一般状态
NUM_CFG	DWORD	已配置的 PROFINET IO 设备的数量
NUM_ACTIVE	DWORD	已激活的 PROFINET IO 设备的数量
NUM_DIAG	DWORD	诊断信息变化的 PROFINET IO 设备的数量

4. 描述

- DEV_STATE （所有设备的状态）

输出 DEV_STATE 指示与 IO 设备的数据交换是否正在工作，并指出有无设备存在需要处理的诊断信息。

任何 IO 设备消失或出现诊断信息时，输出将切换为 FAILED。

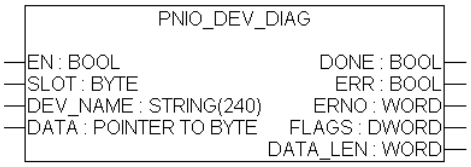
所有的从站回到正常运行状态时，DEV_STATE 将改变为 OK。

借助枚举 PNIO_DEV_STATE_TYPE，以纯文本形式显示 DEV_STATE。

值	含义
PNIO_DEV_STATE_UNDEFINED	未定义
PNIO_DEV_STATE_OK	OK
PNIO_DEV_STATE_FAILED	故障或警告（至少一个 IO 设备）

5.6.3 PNIO_DEV_DIAG（读取 PROFINET IO 设备诊断信息）

1. 功能：读取指定 PROFINET IO 设备的状态诊断信息。
2. 类型：标准功能块（FB，含历史数据）。
3. 参数说明



输入参数	数据类型	描述
EN	BOOL	使能功能块　FALSE：无效 TRUE：上升沿有效
SLOT	BYTE	通讯模块的插槽，CPU 集成接口为 0。扩展通讯模块从右向左计数，起始值为 1
DEV_NAME	STRING [240]	字符串，指明 PROFINET IO 设备的名称。
DATA	POINTER TO ARRAY [0...1560] OF BYTE	指向保存诊断数据的数组的指针
输出参数	**数据类型**	**描述**
DONE	BOOL	功能块完成标志，操作执行完毕为 TRUE，仅保持一个扫描周期
ERR	BOOL	功能块错误标志。该输出要与 DONE 一起综合判断。DONE 为 TRUE，且 ERR 为 TRUE，则说明操作完成但存在错误；DONE 为 TRUE，且 ERR 为 FALSE，则说明操作成功
ERNO	WORD	功能块操作存在错误的代码，代码含义请参见附录 A。只有 DONE 为 TRUE，且 ERR 也为 TRUE 时，ERNO 的输出值才有效
FLAGS	DWORD	所选择设备的 MAC 地址的状态标志
DATA_LEN	DWORD	诊断数据的字节长度

4. 描述

- DEV_NAME （设备名称）

输入 DEV_NAME 指定需要诊断的 PROFINET IO 设备。字符串的最大长度可达 240 个字符，其中包括结束字符’0’。

书写规范为 ciXXX-pn-##（对于 ABB 设备，##代表设备前端旋转开关设置的地址）。例如：‘ci501-pn-##’和‘ci502-pn-##’

- FLAGS （状态标志）

输出 FLAGS 显示指定 PROFINET IO 设备的状态。DWORD 中每个位的含义如下：

位	含义
0	IO 设备不存在或 IO 设备对于 DCP 空闲请求无响应
1	IO 设备没有就绪
2	IO 设备存在配置错误（例如在一个网络上，使用了多次 NameOfStation 或 IP）
3	IO 设备返回了一个无效的响应（即 DCP Set IP 没有成功）
4	IO 设备中存在参数错误（例如错误的模块 ID）
5	IO 设备被禁止
6	IO 设备存在诊断数据
7	IO 设备发送了一个“诊断消失”（Diagnosis disappeared）信息
8	对于 IO 设备发送的诊断数据，通讯模块的诊断缓冲区太小
9	之前的数据未读取，通讯模块的诊断缓冲区被 IO 设备的新的诊断数据覆盖
10	请求存放诊断数据的数组太小，以至于不能传送 IO 设备的诊断数据
11	建立连接期间，IO 设备报告了一个 ModuleDiffBlock 信息
12~31	保留

第6章　　高速计数指令

6.1 高速计数功能概述

AC500 支持高速计数功能，并提供多种实现途径。用户可根据需求灵活选择。

- eCo CPU PM5x4 的板载 I/O，输入信号的频率最高可达 50KHz
- S500 开关量 I/O 模块，输入信号的频率最高可达 50KHz
- CS31 总线接口模块，输入信号的频率最高可达 50KHz
- 专用编码器模块 CD522，输入信号的频率最高可达 300KHz
- 专用中断和高速计数模块 DC541，输入信号的频率最高可达 50KHz

6.2 集成于 PM5x4 CPU 板载 I/O 上的高数计数器

eCo CPU PM5x4 的板载 I/O 支持高速计数功能，根据操作模式，每个 CPU 最多提供两个高速计数器。（通道 0 和通道 1），输入信号的频率最高可达 50KHz。

下表说明高速计数器输入通道和输出通道的分配情况。

CPU	是否高速计数器	分配的输入		分配的输出	注释
		通道 A	通道 B	通道 C	
PM554	集成	输入通道 0（%IX4000.0）	输入通道 1（%IX4000.1）	输出通道 0（%QX4000.0）	模块最多提供两个高速计数器。输入通道 0 是高速计数器的默认通道。根据高速计数器的模式，输入通道 1 的功能有所不同。
PM564	集成	输入通道 0（%IX4000.0）	输入通道 1（%IX4000.1）	输出通道 0（%QX4000.0）	

6.2.1　PM5x4 CPU 板载 I/O 高速计数器的操作模式

PM5x4 CPU 板载 I/O 高速计数器提供 9 种操作模式。在 CBP PLC 配置中，通过配置参数选择所需的操作模式。

操作模式的选择决定使能、设置及方向功能的控制方式。计数器未使用的输入或输出通道可用于其他功能。

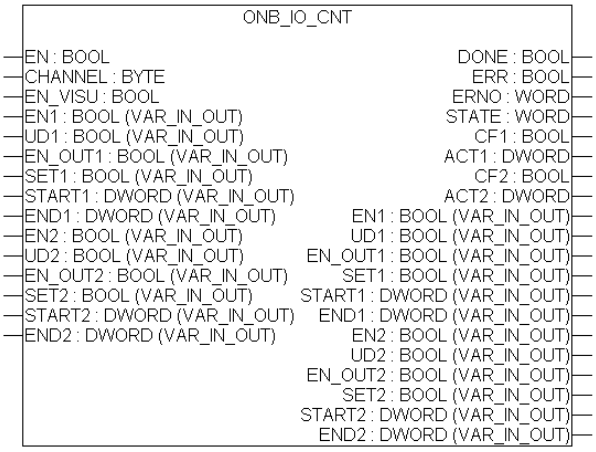
高速计数器即支持普通的计数器模式也支持 A/B 轨迹计数器模式。A/B 轨迹计数器用于对运动传感器的信号进行计数，且支持倍频功能。

下表对操作模式进行了详细的描述，其中 A 指输入通道 A，B 指输入通道 B，C 指输出通道 C。

操作模式	功能	占用的通道	描述	最大计数频率
0	无计数器	无	不使用集成的高速计数器选择此种模式	-
1	一个加计数器	A =计数输入	对 A 的上升沿进行加计数，范围从 0 到 0xFFFFFFFF	50 KHz
		C =达到结束值	达到结束值时，C 被置为高电平	
2	一个通过端子使能输入的加计数器	A =计数输入	对 A 的上升沿进行加计数，范围从 0 到 0xFFFFFFFF	50 KHz
		B =使能输入	只有 B 为高电平，计数器才被激活	
		C =达到结束值	当已达结束值时，C 将被置为高电平	
3	两个加/减计数器	A =计数器输入 1	两个独立的计数器 通过用户程序控制使能、设置及方向功能	50 KHz
		B =计数器输入 2		

操作模式	功能	占用的通道	描述	最大计数频率
4	两个加/减计数器	A =计数器输入 1	两个独立的计数器	50 KHz
		B =计数器输入 2	与模式 3 相似，但输入 2 是对 B 的下降沿进行计数	
5	一个通过端子上升沿设置的加/减计数器	A =计数器输入	对 A 的上升沿进行计数。使能和方向功能通过用户程序控制。设置功能由端子输入 B 控制	50 KHz
		B =动态设置输入	“动态”是指设置功能由 B 的上升沿触发完成	
6	一个通过端子下降沿设置的加/减计数器	A =计数器输入	与模式 5 相似	50 KHz
		B =动态设置输入	“动态”是指设置功能由 B 的下降沿触发完成	
7	一个加/减计数器（增量式编码器）	A =A 相	在该模式下，可使用增量式编码器，计数信号的 A 相和 B 相的相位差为 90°	35 KHz
		B =B 相	依靠 A 和 B 两个信号的顺序决定加或减计数。使能和设置功能通过用户程序控制。无脉冲倍频功能。传感器必须提供 24V 信号，5V 传感器信号必须转换	
8	保留	-	-	-
9	一个加/减计数器（2 倍频增量式编码器）	A =A 相 B =B 相	与模式 7 相似，但对计数输入端的脉冲进行 2 倍频。这意味着计数器对 A 相的上升沿和下降沿都进行计数。精度也相应增加	30 KHz
10	一个加/减计数器（4 倍频增量式编码器）	A =A 相 B =B 相	与模式 7 相似，但对计数输入端的脉冲进行 4 倍频。这意味着计数器对 A 相和 B 相的上升沿和下降沿都进行计数。精度也相应增加	15 KHz

6.2.2 ONB_IO_CNT（板载 I/O 高速计数指令）



1. 功能：控制 eCo CPU PM5x4 集成的高速计数器。
2. 类型：标准功能块（FB，含历史数据）。
3. 所属库：OnBoardIO_AC500_V13.lib。
4. 参数说明

输入参数	数据类型	描述
EN	BOOL	使能功能块　FALSE：无效 TRUE：高电平有效
CHANNEL	BYTE	高速计数的通道号（0）。0 是唯一有效的值，占用输入 I0 和 I1

输入参数	数据类型	描述
EN_VISU	BOOL	使能可视化界面 visuONB_IO_CNT 的控制功能 FALSE：禁用可视化界面控制功能 TRUE：使能可视化界面控制功能
输入输出参数	**数据类型**	**描述**
EN1	BOOL	使能计数器 1　　FALSE：禁用计数器 1 TRUE：高电平使能计数器 1
UD1	BOOL	计数器 1 的计数方向　　FALSE：加计数 TRUE：减计数
EN_OUT1	BOOL	输出 C 的控制来源（仅适用于模式 1 和模式 2） FALSE：输出 C 根据模式依计数器状态动作 TRUE：通过 PLC 程序控制输出 C
SET1	BOOL	设置输入计数器 1 上升沿激活起始值及方向控制功能。功能变化时需要重新激活。 保持为 TRUE 时，计数器的当前值保持不变
START1	DWORD	计数器 1 的起始值
END1	DWORD	计数器 1 的结束值
EN2	BOOL	使能计数器 2（仅适用于模式 3 和模式 4）
UD2	BOOL	计数器 2 的计数方向　　FALSE：加计数 TRUE：减计数
EN_OUT2	BOOL	保留
SET2	BOOL	设置输入计数器 2 上升沿激活起始值及方向控制功能。功能变化时需要重新激活。 保持为 TRUE 时，计数器的当前值保持不变
START2	DWORD	计数器 2 的起始值
END2	DWORD	计数器 2 的结束值
输出参数	**数据类型**	**描述**
DONE	BOOL	功能块完成标志，操作执行完毕为 TRUE，仅保持一个扫描周期
ERR	BOOL	功能块错误标志。该输出要与 DONE 一起综合判断。DONE 为 TRUE，且 ERR 为 TRUE，则说明操作完成但存在错误；DONE 为 TRUE，且 ERR 为 FALSE，则说明操作成功
ERNO	WORD	功能块操作存在错误的代码，代码含义请参见附录 A。只有 DONE 为 TRUE，且 ERR 也为 TRUE 时，ERNO 的输出值才有效
STATE	WORD	功能块状态显示
CF1	BOOL	计数器 1 达到结束值指示
ACT1	DWORD	计数器 1 的当前值
CF2	BOOL	计数器 2 达到结束值指示
ACT2	DWORD	计数器 2 的当前值

5. 描述

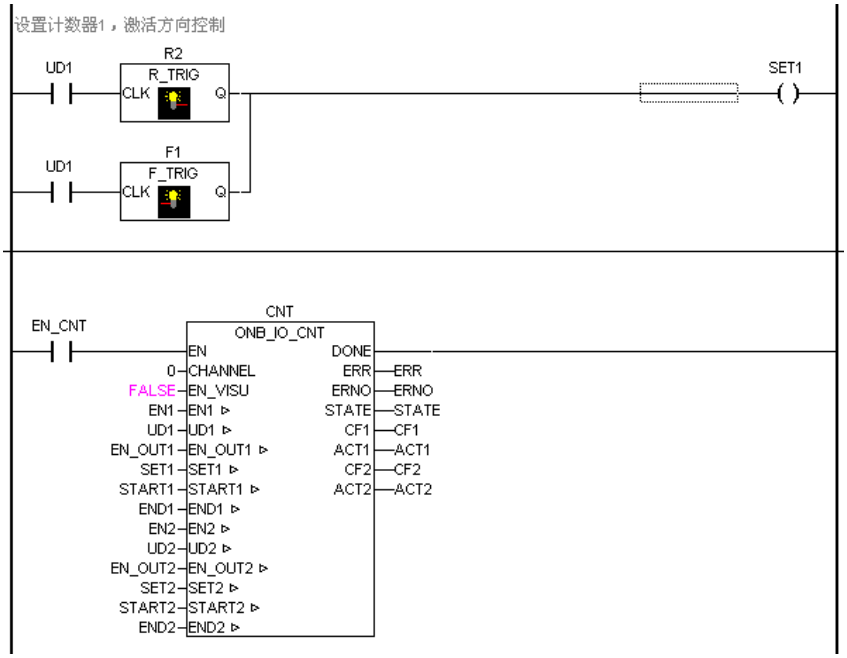
- 调用该功能块完成对计数器的操作，用户仅进行逻辑编写无需考虑计数器与用户程序之间如何进行数据交换。
- 计数范围 0 ~4,294,967,295（16 进制 00 00 00 00~ FF FF FF FF）。
- 可以为每个计数器设置一个起始值。加计数器的默认起始值为 0，减计数器的默认起始值为 4,294,967,295。根据操作模式的不同，可通过用户程序或端子使计数器的当前值等于起始值。

- 应该为每个计数器设置一个结束值。计数器不断地将其与当前值进行比较，判断是否相等。当计数器（当前值）达到设定的结束值时，输出 CF 被置位。根据操作模式的不同，可通过用户程序或端子将结束值存储到模块中。
- 在所有的操作模式下都可以进行加计数。从起始值开始计数，直到结束值。到达极限值 4,294,967,295 后，再收到一个脉冲计数器当前值变为 0。
- UPx：方向控制。某些操作模式下，计数器也可以进行减计数。如果需要减计数，UP/DOWN 位必须置为 TRUE。这时，计数器从起始值开始减计数，直到结束值。到达极限值 0 后，再收到一个脉冲计数器当前值变为 4,294,967,295。
- ENx：使能控制。EN 必须为 TRUE，计数器才被激活。
- SETx：设置控制。SET 的上升沿激活起始值及方向控制功能。功能变化时需要重新激活。SET 为 TRUE 时，计数器的当前值保持不变。
- CFx：当计数器到达程序设定的结束值时，计数器的输出保持，CF=TRUE（到达结束值）。只有当再次被设定时（起始值），CF 复位为 FALSE。

6. 程序举例

VAR | VAR_INPUT | VAR_OUTPUT | VAR_IN_OUT | CONSTANT

	名称	地址	类型	初值	注释
0001	CNT		ONB_IO_CNT		
0002	EN_CNT		BOOL	TRUE	使能功能块
0003	EN1		BOOL	TRUE	使能计数器1
0004	EN2		BOOL	TRUE	使能计数器2
0005	UD1		BOOL		计数器1默认为加计数
0006	UD2		BOOL		计数器2默认为加计数
0007	EN_OUT1		BOOL		模式3中无效
0008	SET1		BOOL		设置计数器1
0009	START1		DWORD	0	计数器1起始值为0
0010	END1		DWORD	20000	计数器1结束值为20000
0011	SET2		BOOL		设置计数器2
0012	START2		DWORD	0	计数器2起始值为0
0013	END2		DWORD	10000	计数器2结束值为10000
0014	EN_OUT2		BOOL		模式3中无效



- 方向控制发生变化时需激活计数器设置变量“SETx”。
- 本例中计数器始终处于激活状态。

6.3 集成于 S500 I/O 模块中的高速计数器

部分 S500 开关量 I/O 模块集成高速计数功能，且计数器功能只能在 I/O 总线本地扩展的模块上正常工作。根据操作模式，每个 I/O 模块最多提供两个高速计数器。计数器占用 2 个开关量输入和 1 个开关量输出（与操作模式相关），计数器不使用的输入或输出通道可用于其他功能。

下表说明哪些 S500 模块集成高速计数器，哪些开关量输入和输出被高速计数器占用。

I/O 模块	是否集成高速计数器	分配的输入		分配的输出
		通道 A	通道 B	通道 C
DC522	是	C8	C9	C10
DC523	是	C16	C17	C18
DC532	是	C24	C25	C26
DI524	是	I24	I25	无硬件输出点
DX522	是	I0	I1	高速计数器不能使用继电器通道
DC561	否	-	-	-
DI561, DI562, DI571	否	-	-	-
DO561, DO571, DO572	否	-	-	-

6.3.1 S500 I/O 模块高速计数器的操作模式

S500 I/O 模块高速计数器的操作模式与 PM5x4 CPU 板载 I/O 高速计数器的操作模式相同，请参见 6.2.1 章节。

6.4 集成于 CS31 总线接口模块中的高速计数器

CS31 总线接口模块 DC551-CS31，CI590-CS31-HA 和 CI592-CS31 集成高速计数功能，根据操作模式，每个 I/O 模块最多提供两个高速计数器。计数器占用 2 个开关量输入和 1 个开关量输出（与操作模式相关），计数器不使用的输入或输出通道可用于其他功能。

下表说明高速计数器输入通道和输出通道的分配情况。

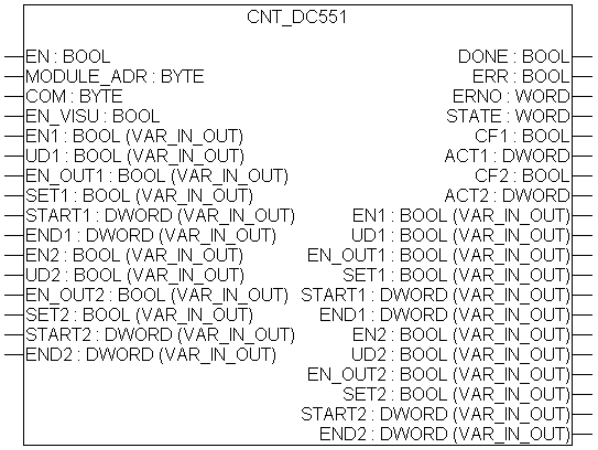
I/O 模块	是否集成高速计数器	分配的输入		分配的输出
		通道 A	通道 B	通道 C
DC551-CS31	是	C16	C17	C18
CI590-CS31-HA	是	C8	C9	C10
CI592-CS31	是	DC8	DC9	DC10

只有当模块上的地址旋转开关设置的总线地址大于 70 时，CS31 总线模块的计数器功能才被激活。这时，有效的总线地址等于设置的地址减去 70。例如，如果总线地址为 83，其有效总线地址为（83-70）=13，并且集成的高速计数器有效。

6.4.1 CS31 总线接口模块高速计数器的操作模式

CS31 总线接口模块高速计数器的操作模式与 PM5x4 CPU 板载 I/O 高速计数器的操作模式相同，请参见 6.2.1 章节。

6.4.2 CNT_DC551（CS31 总线接口模块高速计数指令）



1. 功能：控制 CS31 总线接口模块集成的高速计数器（DC551-CS31、CI590-CS31 和 CI592-CS31）。
2. 类型：标准功能块（FB，含历史数据）。
3. 所属库：Counter_AC500_V20.lib。
4. 参数说明

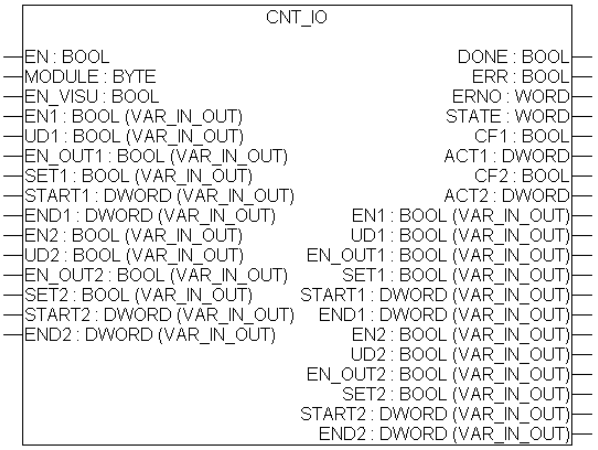
输入参数	数据类型	描述
EN	BOOL	使能功能块　FALSE：无效 TRUE：高电平有效
MODULE_ADR	BYTE	CS31 总线接口模块的地址（70～99）
COM	BYTE	CS31 总线主站的接口编号（1 ：COM1；2 ：COM2）
EN_VISU	BOOL	使能可视化界面 visuCNT_DC551 的控制功能 FALSE：禁用可视化界面控制功能 TRUE：使能可视化界面控制功能
输入输出参数	**数据类型**	**描述**
EN1	BOOL	使能计数器 1　FALSE：禁用计数器 1 TRUE：高电使能计数器 1
UD1	BOOL	计数器 1 的计数方向　FALSE：加计数 TRUE：减计数
EN_OUT1	BOOL	输出 C 的控制来源（仅适用于模式 1 和模式 2） FALSE：输出 C 根据模式依计数器状态动作 TRUE：通过 PLC 程序控制输出 C
SET1	BOOL	设置输入计数器 1 上升沿激活起始值及方向控制功能。功能变化时需要重新激活。保持为 TRUE 时，计数器的当前值保持不变
START1	DWORD	计数器 1 的起始值
END1	DWORD	计数器 1 的结束值
EN2	BOOL	使能计数器 2（仅适用于双模式 3 和模式 4）
UD2	BOOL	计数器 2 的计数方向　FALSE：加计数 TRUE：减计数
EN_OUT2	BOOL	保留

输入输出参数	数据类型	描述
SET2	BOOL	设置输入计数器 2 上升沿激活起始值及方向控制功能。功能变化时需要重新激活。保持为 TRUE 时，计数器的当前值保持不变
START2	DWORD	计数器 2 的起始值
END2	DWORD	计数器 2 的结束值
输出参数	**数据类型**	**描述**
DONE	BOOL	功能块完成标志，操作执行完毕为 TRUE，仅保持一个扫描周期
ERR	BOOL	功能块错误标志。该输出要与 DONE 一起综合判断。DONE 为 TRUE，且 ERR 为 TRUE，则说明操作完成但存在错误；DONE 为 TRUE，且 ERR 为 FALSE，则说明操作成功
ERNO	WORD	功能块操作存在错误的代码，代码含义请参见附录 A。只有 DONE 为 TRUE，且 ERR 也为 TRUE 时，ERNO 的输出值才有效
STATE	WORD	指示高速计数器的当前模式
CF1	BOOL	计数器 1 达到结束值指示
ACT1	DWORD	计数器 1 的当前值
CF2	BOOL	计数器 2 达到结束值指示
ACT2	DWORD	计数器 2 的当前值

5. 描述

- 调用该功能块完成对计数器的操作，用户仅进行逻辑编写无需考虑计数器与用户程序之间如何进行数据交换。
- 计数范围 0 ~4,294,967,295（16 进制 00 00 00 00~ FF FF FF FF）。
- 可以为每个计数器设置一个起始值。加计数器的默认起始值为 0，减计数器的默认起始值为 4,294,967,295。根据操作模式的不同，可通过用户程序或端子使计数器的当前值等于起始值。
- 应该为每个计数器设置一个结束值。计数器不断地将其与当前值进行比较，判断是否相等。当计数器（当前值）达到设定的结束值时，输出 CF 被置位。根据操作模式的不同，可通过用户程序或端子将结束值存储到模块中。
- 在所有的操作模式下都可以进行加计数。从起始值开始计数，直到结束值。到达极限值 4,294,967,295 后，再收到一个脉冲计数器当前值变为 0。
- UPx：方向控制。某些操作模式下，计数器也可以进行减计数。如果需要减计数，UP/DOWN 位必须置为 TRUE。这时，计数器从起始值开始减计数，直到结束值。到达极限值 0 后，再收到一个脉冲计数器当前值变为 4,294,967,295。
- ENx：使能控制。EN 必须为 TRUE，计数器才被激活。
- SETx：设置控制。SET 的上升沿激活起始值及方向控制功能。功能变化时需要重新激活。SET 为 TRUE 时，计数器的当前值保持不变。
- CFx：当计数器到达程序设定的结束值时，计数器的输出保持，CF=TRUE（到达结束值）。只有当再次被设定时（起始值），CF 复位为 FALSE。

6.4.3 CNT_IO（S500 I/O 模块高速计数指令）



1. 功能：控制 S500 开关量 I/O 模块集成的高速计数器。
2. 类型：标准功能块（FB，含历史数据）。
3. 所属库：Counter_AC500_V20.lib。
4. 参数说明

输入参数	数据类型	描述
EN	BOOL	使能功能块　FALSE：无效 TRUE：高电平有效
MODULE	BYTE	I/O 总线的模块位置编号（1~10），CPU 右侧第一个模块的编号为 1
EN_VISU	BOOL	使能可视化界面 visuCNT_IO 的控制功能 FALSE：禁用可视化界面控制功能 TRUE：使能可视化界面控制功能
输入输出参数	**数据类型**	**描述**
EN1	BOOL	使能计数器 1　FALSE：禁用计数器 1 TRUE：高电平使能计数器 1
UD1	BOOL	计数器 1 的计数方向　FALSE：加计数 TRUE：减计数
EN_OUT1	BOOL	输出 C 的控制来源（仅适用于模式 1 和模式 2） FALSE：输出 C 根据模式依计数器状态动作 TRUE：通过 PLC 程序控制输出 C
SET1	BOOL	设置输入计数器 1 上升沿激活起始值及方向控制功能。功能变化时需要重新激活。保持为 TRUE 时，计数器的当前值保持不变
START1	DWORD	计数器 1 的起始值
END1	DWORD	计数器 1 的结束值
EN2	BOOL	使能计数器 2（仅适用于模式 3 和模式 4）
UD2	BOOL	计数器 2 的计数方向　FALSE：加计数 TRUE：减计数
EN_OUT2	BOOL	保留
SET2	BOOL	设置输入计数器 2 上升沿激活起始值及方向控制功能。功能变化时需要重新激活。保持为 TRUE 时，计数器的当前值保持不变

输入输出参数	数据类型	描述
START2	DWORD	计数器 2 的起始值
END2	DWORD	计数器 2 的结束值
输出参数	**数据类型**	**描述**
DONE	BOOL	功能块完成标志，操作执行完毕为 TRUE，仅保持一个扫描周期
ERR	BOOL	功能块错误标志。该输出要与 DONE 一起综合判断。DONE 为 TRUE，且 ERR 为 TRUE，则说明操作完成但存在错误；DONE 为 TRUE，且 ERR 为 FALSE，则说明操作成功
ERNO	WORD	功能块操作存在错误的代码，代码含义请参见附录 A。只有 DONE 为 TRUE，且 ERR 也为 TRUE 时，ERNO 的输出值才有效
STATE	WORD	指示高速计数器的当前模式
CF1	BOOL	计数器 1 达到结束值指示
ACT1	DWORD	计数器 1 的当前值
CF2	BOOL	计数器 2 达到结束值指示
ACT2	DWORD	计数器 2 的当前值

5. 描述

- 调用该功能块完成对计数器的操作，用户仅进行逻辑编写无需考虑计数器与用户程序之间如何进行数据交换。
- 计数范围 0 ~4,294,967,295（16 进制 00 00 00 00~ FF FF FF FF）。
- 可以为每个计数器设置一个起始值。加计数器的默认起始值为 0，减计数器的默认起始值为 4,294,967,295。根据操作模式的不同，可通过用户程序或端子使计数器的当前值等于起始值。
- 应该为每个计数器设置一个结束值。计数器不断地将其与当前值进行比较，判断是否相等。当计数器（当前值）达到设定的结束值时，输出 CF 被置位。根据操作模式的不同，可通过用户程序或端子将结束值存储到模块中。
- 在所有的操作模式下都可以进行加计数。从起始值开始计数，直到结束值。到达极限值 4,294,967,295 后，再收到一个脉冲计数器当前值变为 0。
- UPx：方向控制。某些操作模式下，计数器也可以进行减计数。如果需要减计数，UP/DOWN 位必须置为 TRUE。这时，计数器从起始值开始减计数，直到结束值。到达极限值 0 后，再收到一个脉冲计数器当前值变为 4,294,967,295。
- ENx：使能控制。EN 必须为 TRUE，计数器才被激活。
- SETx：设置控制。SET 的上升沿激活起始值及方向控制功能。功能变化时需要重新激活。SET 为 TRUE 时，计数器的当前值保持不变。
- CFx：当计数器到达程序设定的结束值时，计数器的输出保持，CF=TRUE（到达结束值）。只有当再次被设定时（起始值），CF 复位为 FALSE。

6.5 编码器模块 CD522 的高速计数器

CD522 为专用编码器和脉冲输出功能模块。CD522 集成两路独立的高速计数器，每一路均具有如下特点：

- 支持 12 种操作模式
- 最高计数频率达 300KHz
- 最高测频频率 5KHz
- 支持+24V，5VDC，差分输入，1Vpp 正弦输入
- 支持复位、设置、触发/锁存和参考点初始化（RPI）功能
- 支持数字输入滤波功能
- 具有到达结束值和溢出指示

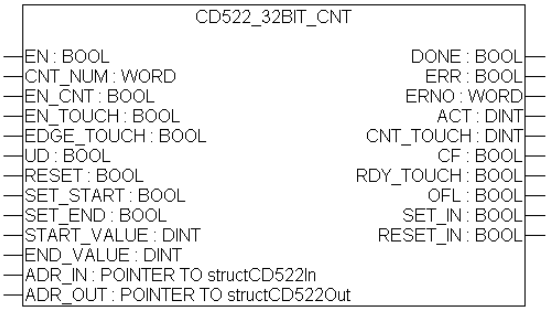
6.5.1 CD522 高速计数器的操作模式

CD522 高速计数器支持 12 种操作模式。在 CBP PLC 配置中，通过配置参数选择所需的操作模式。操作模式的选择决定使能、复位、设置、以及触发/锁存等功能的控制方式。

操作模式	功能	占用通道	描述	对应功能块
0	无计数器	无	不使用高速计数器	-
1	一个 32 位加/减计数器	A =计数输入	对 A 的上升沿进行加/减计数。支持复位、设置、触发/锁存功能。具有到达结束值和溢出指示	CD522_32BIT_CNT
2	一个通过端子使能输入的 32 位加/减计数器	A =计数输入 B =使能输入	对 A 的上升沿进行加/减计数。支持复位、设置、触发/锁存功能。具有到达结束值和溢出指示。 只有 B 为高电平，计数器才被激活	CD522_32BIT_CNT
3	两个 16 位加/减计数器	A =计数输入 1 B =计数输入 2	两个独立的计数器 支持复位、触发/锁存功能。具有溢出指示	CD522_16BIT_2CNT
4	两个 16 位加/减计数器	A =计数输入 1 B =计数输入 2	两个独立的计数器 与模式 3 相似，但输入 2 是对 B 的下降沿进行计数	CD522_16BIT_2CNT
5	一个可通过端子上升沿设置的 32 位加/减计数器	A =计数输入 B =动态设置输入	对 A 的上升沿进行加/减计数。支持复位、设置、触发/锁存功能。具有到达最终值和溢出指示。设置功能可通过端子输入 B 完成。 “动态”是指设置功能由 B 的上升沿触发完成	CD522_32BIT_CNT
6	一个可通过端子下降沿设置的 32 位加/减计数器	A =计数输入 B =动态设置输入	与模式 5 相似 “动态”是指设置功能由 B 的下降沿触发完成	CD522_32BIT_CNT
7	保留	-	-	-
8	一个通过端子使能输入的 16 位加/减计数器，具有过 0 检测功能	A =计数输入 B =使能输入	对 A 的上升沿进行加/减计数（范围 -32768~32767）。支持复位、设置、触发/锁存功能。具有到达最终值、溢出及过 0 指示。 只有 B 为高电平，计数器才被激活	CD522_16BIT_CNT
9	保留	-	-	-
10	保留	-	-	-
11	增量式编码器	A =A 相 B =B 相 Z =Z 相	计数信号 A 相和 B 相的相位差为 90°。依靠 A 和 B 信号的顺序决定加或减计数。无脉冲倍频功能。支持复位、设置、触发/锁存功能。具有到达最终值和溢出指示	CD522_32BIT_ENCODER
12	2 倍频增量式编码器	A =A 相 B =B 相 Z =Z 相	与模式 7 相似，但对计数输入端的脉冲进行 2 倍频。这意味着计数器对 A 相的上升沿和下降沿都进行计数，精度也相应增加	CD522_32BIT_ENCODER

操作模式	功能	占用通道	描述	对应功能块
13	4 倍频增量式编码器	A =A 相 B =B 相 Z =Z 相	与模式 7 相似，但对计数输入端的脉冲进行 4 倍频。这意味着计数器对 A 相和 B 相的上升沿和下降沿都进行计数，精度也相应增加	CD522_32BIT_ENCODER
14	绝对式 SSI 编码器	A =数据信号 B =时钟信号	使用 SSI 接口的绝对编码器	CD522_SSI_CNT
15	时间频率测量	Z =输入信号	通过对 Z 信号上升沿、下降沿的时间测量，计算每分钟的旋转速度和频率	CD522_FREQ_SCAN

6.5.2 CD522_32BIT_CNT（CD522 32 位计数器指令）



1. 功能：用于控制 CD522 模块的 32 位加/减计数器。CD522 工作于模式 1、2、5 和模式 6 时，可提供 2 路 32 位加/减计数器，范围-2,147,483,648~2,147,483,647。脉冲计数对于计数器 0 为 A0，对于计数器 1 为 A1。使能或动态设置信号，对于计数器 0 为 B0，对于计数器 1 为 B1。
2. 类型：标准功能块（FB，含历史数据）。
3. 所属库：CD522_AC500_V13.lib
4. 参数说明

输入参数	数据类型	描述
EN	BOOL	使能功能块　　FALSE：无效 TRUE：高电平有效
CNT_NUM	WORD	计数器选择（0 或 1）
EN_CNT	BOOL	使能计数器　　FALSE：计数器不工作 TRUE：高电平使能
EN_TOUCH	BOOL	使能触发/锁存功能　　FALSE：功能无效 TRUE：高电平有效，上升沿使能一次
EDGE_TOUCH	BOOL	触发边沿选择　　FALSE：下降沿激活触发/锁存功能 TRUE ：上升沿激活触发/锁存功能
UD	BOOL	加/减计数方向选择　　FALSE：加计数 TRUE：减计数
RESET	BOOL	复位计数器，当前值清零　　FALSE：无效 TRUE：高电平有效。保持为 TRUE 时，计数器的当前值始终为 0（ACT=0）
SET_START	BOOL	设置计数器的起始值　　FALSE：无效 TRUE：高电平有效。保持为 TRUE 时，计数器的当前值保持不变（ACT=START_VALUE）

输入参数	数据类型	描述
SET_END	BOOL	设置计数器的结束值　　FALSE：无效 TRUE：高电平有效。为计数器设置一个目标值。计数器不断地将其与当前值进行比较，判断是否相等
START_VALUE	DINT	计数器的起始值
END_VALUE	DINT	计数器的结束值
ADR_IN	POINTER TO structCD522counterIn	指向 CD522 模块输入首地址的指针
ADR_OUT	POINTER TO structCD522counterOut	指向 CD522 模块输出首地址的指针
输出参数	**数据类型**	**描述**
DONE	BOOL	功能块完成标志，操作执行完毕为 TRUE，仅保持一个扫描周期
ERR	BOOL	功能块错误标志。该输出要与 DONE 一起综合判断。DONE 为 TRUE，且 ERR 为 TRUE，则说明操作完成但存在错误；DONE 为 TRUE，且 ERR 为 FALSE，则说明操作成功
ERNO	WORD	功能块操作存在错误的代码，代码含义请参见附录 A。只有 DONE 为 TRUE，且 ERR 也为 TRUE 时，ERNO 的输出值才有效
ACT	DINT	计数器的当前值
CNT_TOUCH	DINT	计数器锁存的计数值
CF	BOOL	计数器达到结束值指示
RDY_TOUCH	BOOL	锁存值更新标志，有新的锁存值时为 TRUE
OFL	BOOL	计数器溢出标志，溢出时为 TRUE
SET_IN	BOOL	“SET”功能有效标志，有效时 TRUE
RESET_IN	BOOL	“RESET”功能有效标志，有效时 TRUE

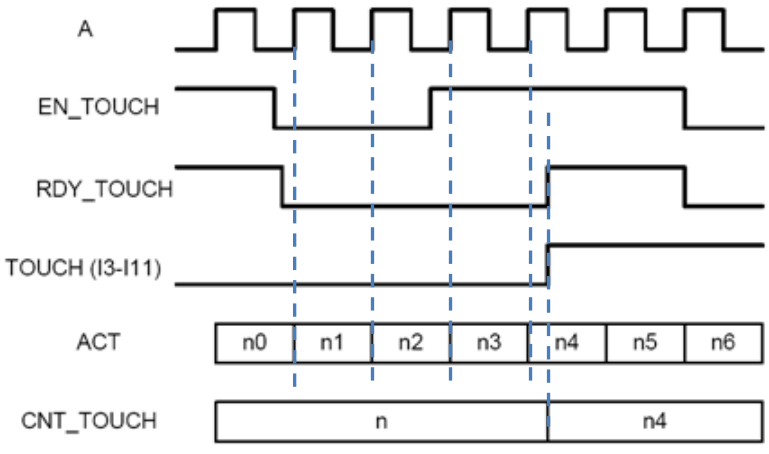
5. 描述

- 触发/锁存功能

触发/锁存功能利用外部硬件信号同步获得计数位置，除去了所有 I/O 总线和网络的延迟时间。如果使用相同的硬件用于触发/锁存输入信号，该功能可实现两个不同编码器设备的同步。

当 TOUCH 外部信号边沿出现时，当前的计数器值（ACT）被存储到一个专用的双字（CNT_TOUCH）中，并通过锁存值更新标志（RDY_TOUCH）进行标示。

EDGE_TOUCH 的值决定该功能是由上升沿还是下降沿触发，下图以上升沿触发为例说明各参数的关系。



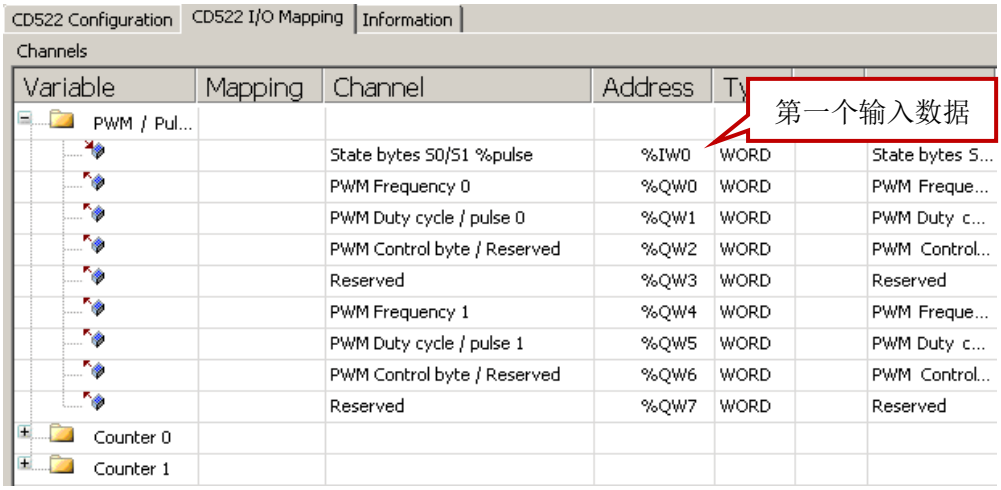
- EN_TOUCH （使能触发/锁存）

EN_TOUCH 输入出现上升沿时，激活触发/锁存功能。如果 EN_TOUCH=TRUE，输入 I3（对于计数器 0）或者 I11（对于计数器 1）出现上升沿时，功能块保存当前计数值 ACT，并在 CNT_TOUCH 中输出。

如果输入 EN_TOUCH 为 FALSE，输出 RDY_TOUCH 复位为 FALSE。

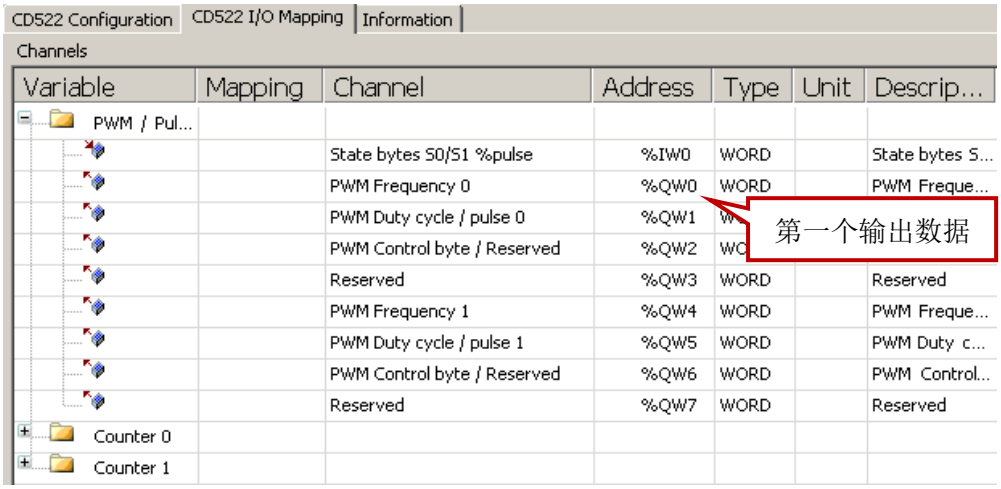
- ADR_IN （输入地址指针）

该输入指定 CD522 模块输入结构的第一个输入数据的地址，需要使用 ADR 运算符获得。如果输入 ADR_IN 没有连接，输出 ERR=TRUE。



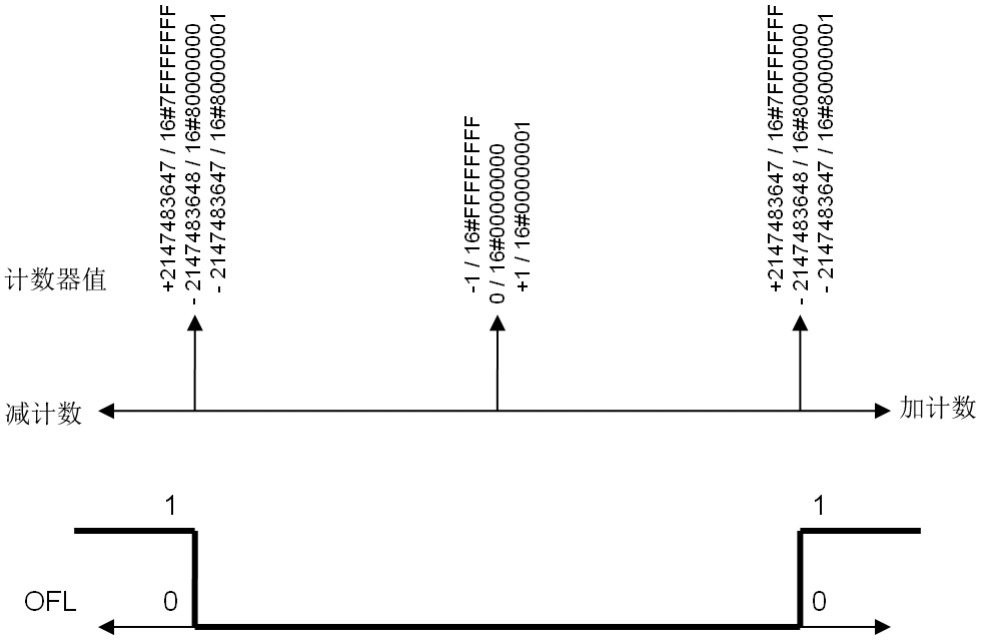
- ADR_OUT　POINTER TO structCD522counterOut（输出地址指针）

该输入指定 CD522 模块输出结构的第一个输出数据的地址，需要使用 ADR 运算符获得。如果输入 ADR_OUT 没有连接，输出 ERR=TRUE。

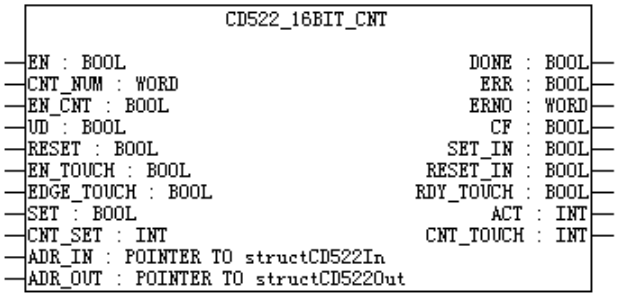


- OFL （溢出标志）

计数器采用无限循环方式进行计数，当 32 位测量值达到 16#80000000 =-2,147,483,648 时，则发生溢出，超过或低于该值（取决于使用的是增计数还是减计数）时，OFL 将被置为 TRUE。如下图所示。即对于加计数器，当前值 ACT 从正数变为负数时，OFL=TRUE；对于减计数器，当前值 ACT 从负数变为正数时，OFL=TRUE。



6.5.3 CD522_16BIT_CNT（CD522 16 位计数器指令）



1. 功能：CD522 模块工作于模式 8 时，可提供 2 路 16 位且具有过 0 检测功能的加/减计数器，范围 -32768~32767。脉冲计数对于计数器 0 为 A0，对于计数器 1 为 A1。使能或动态设置信号，对于计数器 0 为 B0，对于计数器 1 为 B1。可通过此功能块对其进行管理。
2. 类型：标准功能块（FB，含历史数据）。
3. 所属库：CD522_AC500_V13.lib。

4. 参数说明

输入参数	数据类型	描述
EN	BOOL	使能功能块 FALSE：无效 TRUE：高电平有效
CNT_NUM	WORD	计数器选择（0 或 1）
EN_CNT	BOOL	使能计数器 FALSE：计数器不工作 TRUE：高电平使能
UD	BOOL	加/减计数方向选择 FALSE：加计数 TRUE：减计数
RESET	BOOL	复位计数器，当前值清零 FALSE：无效 TRUE：高电平有效。保持为 TRUE 时，计数器的当前值始终为 0（ACT=0）
EN_TOUCH	BOOL	使能触发/锁存功能 FALSE：功能无效 TRUE：高电平有效，上升沿使能一次
EDGE_TOUCH	BOOL	触发边沿选择 FALSE：下降沿激活触发/锁存功能 TRUE ：上升沿激活触发/锁存功能
SET	BOOL	设置计数器的起始值 FALSE：无效 TRUE：高电平有效。保持为 TRUE 时，计数器的当前值保持不变（ACT=CNT_SET）
CNT_SET	INT	计数器的起始值（-32768~32767）
ADR_IN	POINTER TO structCD522counterIn	指向 CD522 模块输入首地址的指针
ADR_OUT	POINTER TO structCD522counterOut	指向 CD522 模块输出首地址的指针
输出参数	**数据类型**	**描述**
DONE	BOOL	功能块完成标志，操作执行完毕为 TRUE，仅保持一个扫描周期
ERR	BOOL	功能块错误标志。该输出要与 DONE 一起综合判断。DONE 为 TRUE，且 ERR 为 TRUE，则说明操作完成但存在错误；DONE 为 TRUE，且 ERR 为 FALSE，则说明操作成功
ERNO	WORD	功能块操作存在错误的代码，代码含义请参见附录 A。只有 DONE 为 TRUE，且 ERR 也为 TRUE 时，ERNO 的输出值才有效
CF	BOOL	计数器过 0 标志，ACT>0 时为 TRUE
SET_IN	BOOL	“SET”功能有效标志，有效时为 TRUE
RESET_IN	BOOL	“RESET”功能有效标志，有效时为 TRUE
RDY_TOUCH	BOOL	锁存值更新标志，有新的锁存值时为 TRUE
ACT	INT	计数器的当前值
CNT_TOUCH	INT	计数器锁存的计数值

5. 描述

- CF （过 0 标志）

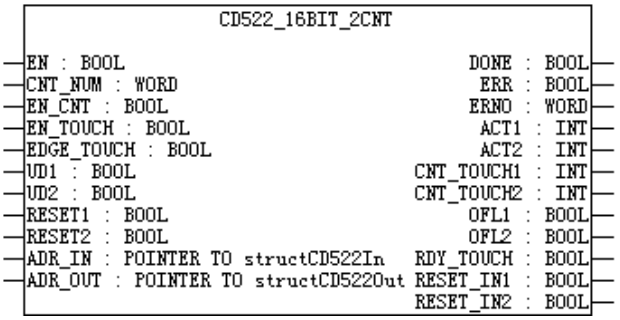
计数器当前值 ACT 小于或等于 0 时，CF 为 FALSE；ACT 大于 0 时，CF 为 TRUE。

当输入 B 为 TRUE 时，该信息有效。

- 其他参数的功能描述请参见 6.5.2 章节 CD522_32BIT_CNT 指令。

6.5.4 CD522_16BIT_2CNT（CD522 双 16 位计数器指令）

1. 功能：CD522 模块的计数器工作于模式 3 或 4 时，每一个计数器可提供 2 路 16 位加/减计数器（计数器 A 和 B），范围-32768~32767。因此 CD522 最多支持 4 路计数器。可通过此功能块进行管理。
2. 类型：标准功能块（FB，含历史数据）。
3. 所属库：CD522_AC500_V13.lib。
4. 参数说明



输入参数	数据类型	描述
EN	BOOL	使能功能块　　FALSE：无效 TRUE：高电平有效
CNT_NUM	WORD	计数器选择（0 或 1）
EN_CNT	BOOL	使能计数器　　FALSE：计数器不工作 TRUE：高电平使能
EN_TOUCH	BOOL	使能触发/锁存功能　　FALSE：功能无效 TRUE：高电平有效，上升沿使能一次
EDGE_TOUCH	BOOL	触发边沿选择　　FALSE：下降沿激活触发//锁存功能 TRUE ：上升沿激活触发/锁存功能
UD1	BOOL	计数器 A 加/减计数方向选择　　FALSE：加计数 TRUE：减计数
UD2	BOOL	计数器 B 加/减计数方向选择　　FALSE：加计数 TRUE：减计数
RESET1	BOOL	复位计数器 A，当前值清零　　FALSE：无效 TRUE：高电平有效。保持为 TRUE 时，计数器 A 的当前值始终为 0（ACT1=0）
RESET2	BOOL	复位计数器 B，当前值清零　　FALSE：无效 TRUE：高电平有效。保持为 TRUE 时，计数器 B 的当前值始终为 0（ACT2=0）
ADR_IN	POINTER TO structCD522counterIn	指向 CD522 模块输入首地址的指针
ADR_OUT	POINTER TO structCD522counterOut	指向 CD522 模块输出首地址的指针
输出参数	**数据类型**	**描述**
DONE	BOOL	功能块完成标志，操作执行完毕为 TRUE，仅保持一个扫描周期
ERR	BOOL	功能块错误标志。该输出要与 DONE 一起综合判断。DONE 为 TRUE，且 ERR 为 TRUE，则说明操作完成但存在错误；DONE 为 TRUE，且 ERR 为 FALSE，则说明操作成功
ERNO	WORD	功能块操作存在错误的代码，代码含义请参见附录 A。只有 DONE 为 TRUE，且 ERR 也为 TRUE 时，ERNO 的输出值才有效

输出参数	数据类型	描述
ACT1	INT	计数器 A 的当前值
ACT2	INT	计数器 B 的当前值
CNT_TOUCH1	INT	计数器 A 锁存的计数值
CNT_TOUCH2	INT	计数器 B 锁存的计数值
OFL1	BOOL	计数器 A 的溢出标志
OFL2	BOOL	计数器 B 的溢出标志
RDY_TOUCH	BOOL	锁存值更新标志，有新的锁存值时为 TRUE
RESET_IN1	BOOL	计数器 A 的“RESET”功能有效标志，有效时为 TRUE
RESET_IN2	BOOL	计数器 B 的“RESET”功能有效标志，有效时为 TRUE

5. 描述

- OFL1 （计数器 A 溢出标志）

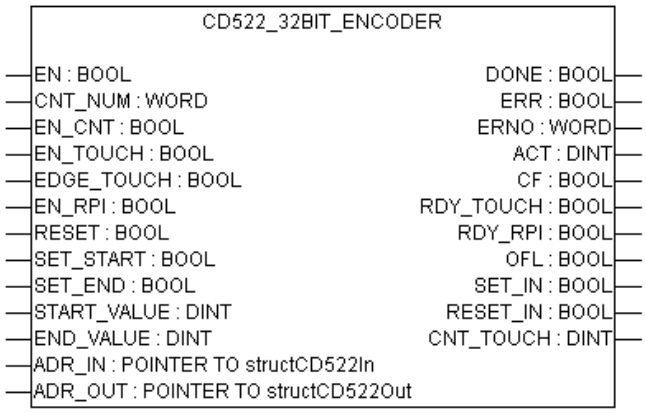
计数器采用无限循环方式进行计数，对于加计数器，当计数器当前值 ACT1 从-1 变为 0 时，OFL1=TRUE；对于减计数器，当计数器当前值 ACT1 从 0 变为-1 时，OFL1=TRUE。加减计数均以“0”为初始值。

- OFL2 （计数器 B 溢出标志）

计数器采用无限循环方式进行计数，对于加计数器，当计数器当前值 ACT2 从-1 变为 0 时，OFL2=TRUE；对于减计数器，当计数器当前值 ACT2 从 0 变为-1 时，OFL2=TRUE。加减计数均以“0”为初始值。

- 其他参数的功能描述请参见 6.5.2 章节 CD522_32BIT_CNT 指令。

6.5.5 CD522_32BIT_ENCODER（CD522 增量式编码器指令）



1. 功能：CD522 模块的计数器工作于模式 11、12 或 13 时，每一个计数器可连接一个增量式编码器，范围-2,147,483,648~2,147,483,647。可通过此功能块进行管理。
2. 类型：标准功能块（FB，含历史数据）。
3. 所属库：CD522_AC500_V13.lib。

4. 参数说明

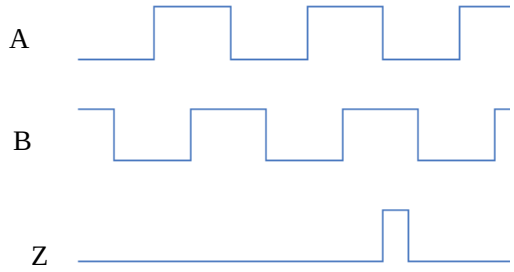
输入参数	数据类型	描述
EN	BOOL	使能功能块　　FALSE：无效 TRUE：高电平有效
CNT_NUM	WORD	计数器选择（0 或 1）
EN_CNT	BOOL	使能计数器　　FALSE：计数器不工作 TRUE：高电平使能
EN_TOUCH	BOOL	使能触发/锁存功能　　FALSE：功能无效 TRUE：高电平有效，上升沿使能一次
EDGE_TOUCH	BOOL	触发边沿选择　　FALSE：下降沿激活触发/锁存功能 TRUE ：上升沿激活触发/锁存功能
EN_RPI	BOOL	使能 RPI 功能　　FALSE：RPI 功能无效 TRUE：高电平使能，上升沿使能一次
RESET	BOOL	复位计数器，当前值清零　　FALSE：无效 TRUE：高电平有效。保持为 TRUE 时，计数器的当前值始终为 0（ACT=0）
SET_START	BOOL	设置计数器的起始值　　FALSE：无效 TRUE：高电平有效。保持为 TRUE 时，计数器的当前值保持不变（ACT=START_VALUE）
SET_END	BOOL	设置计数器的结束值　　FALSE：无效 TRUE：高电平有效。为计数器设置一个目标值。计数器不断地将其与当前值进行比较，判断是否相等
START_VALUE	DINT	计数器的起始值
END_VALUE	DINT	计数器的结束值
ADR_IN	POINTER TO structCD522counterIn	指向 CD522 模块输入首地址的指针
ADR_OUT	POINTER TO structCD522counterOut	指向 CD522 模块输出首地址的指针
输出参数	**数据类型**	**描述**
DONE	BOOL	功能块就绪信息
ERR	BOOL	功能块错误标志
ERNO	WORD	功能块故障代码
ACT	DINT	计数器的当前值
CF	BOOL	计数器达到结束值指示
RDY_TOUCH	BOOL	锁存值更新标志，有新的锁存值时为 TRUE
RDY_RPI	BOOL	RPI 功能有效标志，有效时为 TRUE
OFL	BOOL	计数器溢出标志，溢出时为 TRUE
SET_IN	BOOL	“SET”功能有效标志，有效时 TRUE
RESET_IN	BOOL	“RESET”功能有效标志，有效时 TRUE
CNT_TOUCH	DINT	计数器锁存的计数值

5. 描述

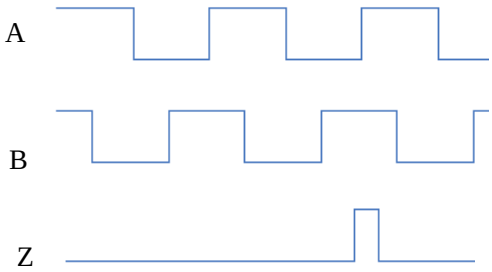
CD522 模块可连接两个增量式编码器。通过测量 A、B、Z 三个信号以确定相对位置。A 和 B 两个信号用于识别旋转方向和脉冲数量，对于计数器 0 为 A0 和 B0，对于计数器 1 为 A1 和 B1。Z 信号用于多圈检测，

对旋转的圈数（机械 0 位）进行计数，对于计数器 0 是 Z0，对于计数器 1 是 Z1。

旋转方向通过 A 和 B 信号之间的 90°相位差进行识别。在模块 CD522 中，信号 A 超前 B 时为顺时针旋转方向，即加计数（见下图）。



顺时针旋转，加计数

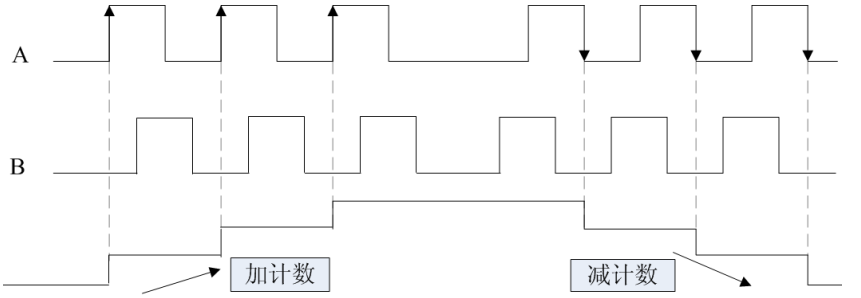


逆时针旋转，减计数

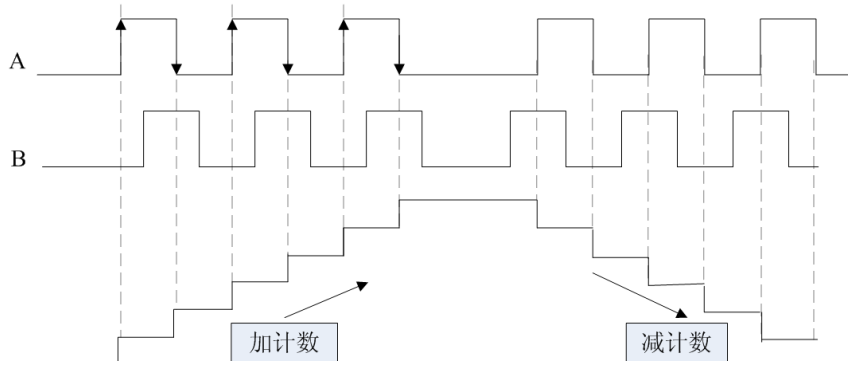
根据操作模式的不同，计数过程的分辨率有所不同，分为一倍频，二倍频和四倍频。一倍频计数模式（模式 11），编码器模块对 A 号的上升沿进行计数。

为了增加分辨率，可以设定为二倍频计数模式（模式 12）。二倍频计数模式在 A 信号的上升沿与下降沿进行计数。

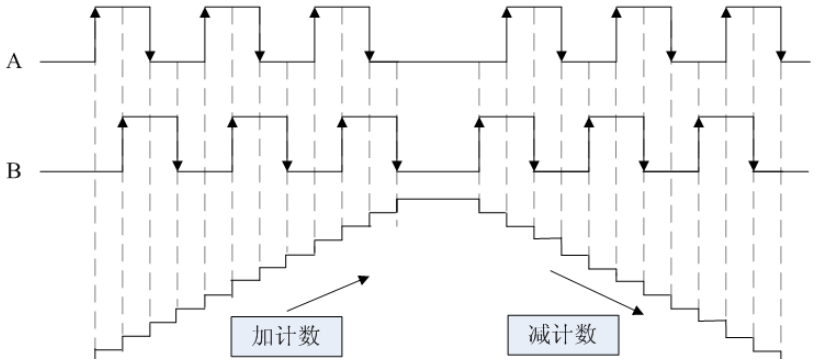
使用四倍频计数模式，分辨率可提高四倍（模式 13）。编码器模块对 A 和 B 信号的上升沿和下降沿均进行计数。



一倍频



二倍频



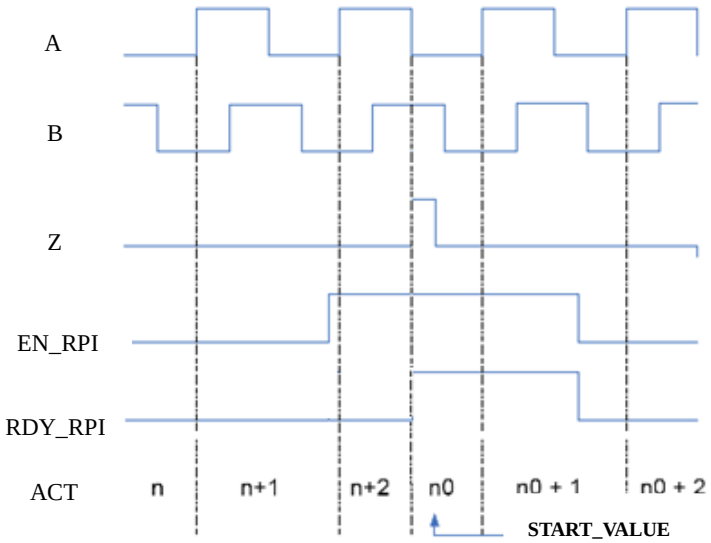
四倍频

- RPI 功能

RPI 功能（参考点初始化）可使编码器与自身的“零点”信号同步。

通过 EN_RPI（控制位）使能 RPI 功能。如果 EN_RPI 为 TRUE，RPI 功能有效。此时，输入信号 RPI 上升沿有效后，在检测到 Z 信号的上升沿时，使 ACT=START_VALUE（当前值=设置的起始值），同时 RDY_RPI 被置位。如果输入 EN_RPI 为 FALSE，输出 RDY_RPI 复位为 FALSE。如下图所示。

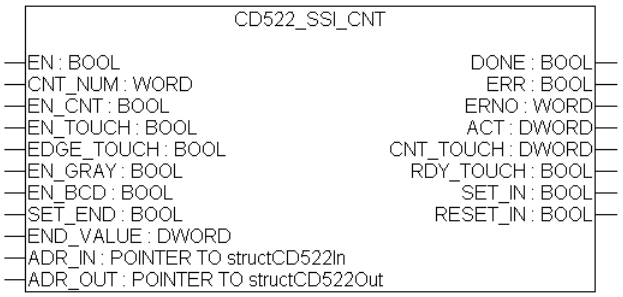
RPI 信号对于计数器 0 在 I3、I4、I5、I6、I7 中配置选择，对于计数器 1 在 I11、I12、I13、I14、I15 中配置选择。



需要通过外部同步信号为编码器设置新的参考值时，该操作方式可用作“触发/锁存”操作。当这两个功能同时激活，或者在其中一个未完成时又激活了另一个功能，则 RPI 功能比触发/锁存功能享有更高的优先级。

- 其他参数的功能描述请参见 6.5.2 章节 CD522_32BIT_CNT 指令。

6.5.6 CD522_SSI_CNT（CD522 绝对式编码器指令）



1. 功能：CD522 模块的计数器工作于模式 14 时，每一个计数器可连接一个 SSI 绝对式编码器。可通过此功能块进行管理。
2. 类型：标准功能块（FB，含历史数据）。
3. 所属库：CD522_AC500_V13.lib。

4. 参数说明

输入参数	数据类型	描述
EN	BOOL	使能功能块　　FALSE：无效 TRUE：高电平有效
CNT_NUM	WORD	计数器选择（0 或 1）
EN_CNT	BOOL	使能计数器　　FALSE：计数器不工作 TRUE：高电平使能
EN_TOUCH	BOOL	使能触发/锁存功能　　FALSE：功能无效 TRUE：高电平有效，上升沿使能一次
EDGE_TOUCH	BOOL	触发边沿选择　　FALSE：下降沿激活触发/锁存功能 TRUE ：上升沿激活触发/锁存功能
EN_GRAY	BOOL	使能 GRAY 码（格林码）模式　　FALSE：无效 TRUE：高电平使能，通过 GRAY 码计算出实际值
EN_BCD	BOOL	使能 BCD 码模式　　FALSE：无效 TRUE：高电平使能，通过 BCD 码计算出实际值
SET_END	BOOL	设置计数器的结束值　　FALSE：无效 TRUE：高电平有效。为计数器设置一个目标值。计数器不断地将其与当前值进行比较，判断是否相等
END_VALUE	DWORD	计数器的结束值
ADR_IN	POINTER TO structCD522counterIn	指向 CD522 模块输入首地址的指针
ADR_OUT	POINTER TO structCD522counterOut	指向 CD522 模块输出首地址的指针
输出参数	**数据类型**	**描述**
DONE	BOOL	功能块就绪信息
ERR	BOOL	功能块错误标志
ERNO	WORD	功能块故障代码
ACT	DWORD	计数器的当前值
CNT_TOUCH	DWORD	计数器锁存的计数值
RDY_TOUCH	BOOL	锁存值更新标志，有新的锁存值时为 TRUE
SET_IN	BOOL	“SET”功能有效标志，有效时 TRUE
RESET_IN	BOOL	“RESET”功能有效标志，有效时 TRUE

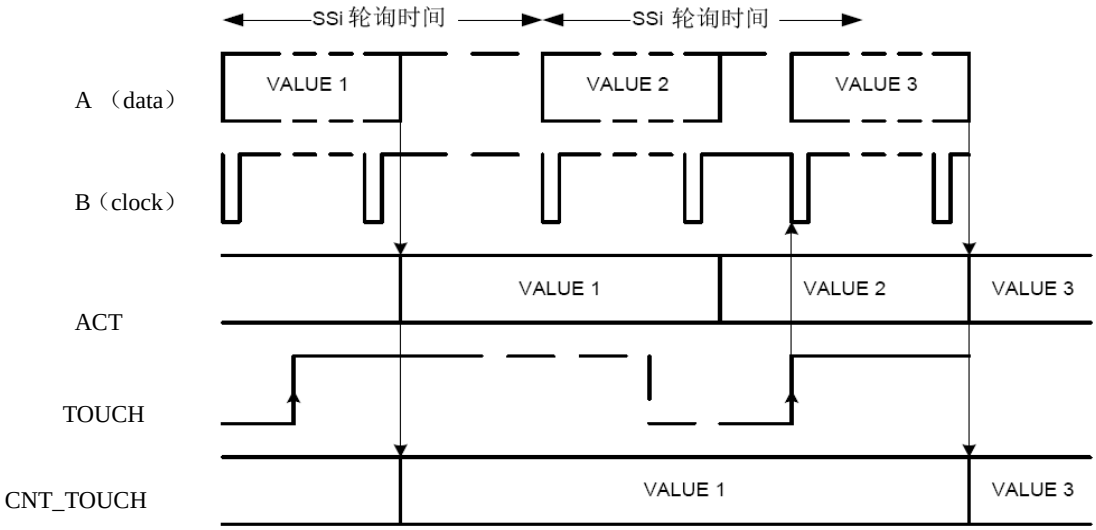
5. 描述

- SSI 的“触发/锁存”功能

SSI 传感器支持触发/锁存功能。通过触发/锁存实现传感器的同步。读取结果根据读取过程是否结束（轮询过程）进行不同的处理。

如果触发信号激活时，读取过程正常结束，最后读取的值存储到触发锁存寄存器中（如下例中的 VALUE1）。

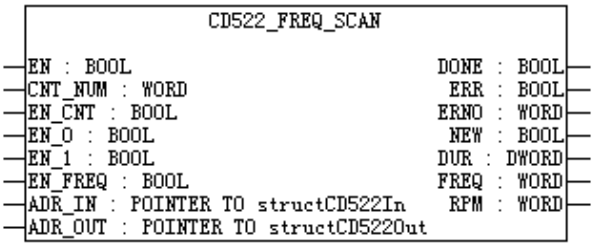
如果触发信号激活时，读取过程还未结束处于两次测量之间，则需再次启动读取过程，最后读取的结果存储到触发锁存寄存器中（如下例中的 VALUE3）。



- 其他参数的功能描述请参见 6.5.2 章节 CD522_32BIT_CNT 指令。

6.5.7 CD522_FREQ_SCAN（CD522 时间频率计指令）

1. 功能：CD522 模块的计数器工作于模式 15 时，可对输入信号 Z（对于计数器 0 为输入 I3，对于计数器 1 为输入 I11）进行时间、频率和旋转速度的测量。分辨率为 1 μs。可通过此功能块进行管理。
2. 类型：标准功能块（FB，含历史数据）。
3. 所属库：CD522_AC500_V13.lib。
4. 参数说明



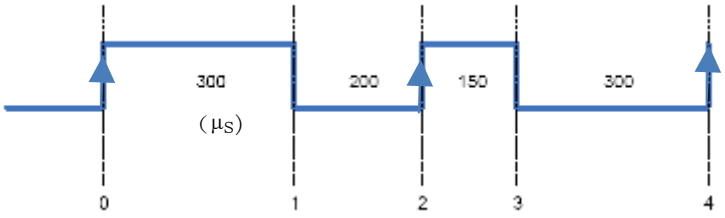
输入参数	数据类型	描述
EN	BOOL	使能功能块　　FALSE：无效 TRUE：高电平有效
CNT_NUM	WORD	计数器选择（0 或 1）
EN_CNT	BOOL	使能频率测量功能　　FALSE：不工作 TRUE：高电平使能
EN_0	BOOL	下降沿捕捉选择　　FALSE：功能无效 TRUE：捕捉信号的下降沿进行测量
EN_1	BOOL	上升沿捕捉选择　　FALSE：功能无效 TRUE：捕捉信号的上升沿进行测量

输入参数	数据类型	描述
EN_FREQ	BOOL	频率模式选择　　FALSE：时间模式，输出通过 DUR（μs）显示 TRUE：频率转速模式，输出通过 FREQ（Hz）和 RPM（rpm）显示
ADR_IN	POINTER TO structCD522counterIn	指向 CD522 模块输入地址的指针
ADR_OUT	POINTER TO structCD522counterOut	指向 CD522 模块输出地址的指针
输出参数	**数据类型**	**描述**
DONE	BOOL	功能块就绪信息
ERR	BOOL	功能块错误标志
ERNO	WORD	功能块故障代码
NEW	BOOL	时间测量值更新标志，有新的测量值时为 TRUE（仅时间模式时有效）
DUR	DWORD	信号持续时间（μs）
FREQ	WORD	信号频率（Hz）
RPM	WORD	信号旋转速度（RPM）

5. 描述

功能块 CD522_FREQ_SCAN 通过输入 EN_CNT 进行控制，通过输入 EN_0 或输入 EN_1 配置下降沿或上升沿捕捉，通过输入 EN_FREQ 配置时间或频率、转速测量（见下图和下表）。

下表显示根据配置的输入参数和时间举例所得到的测量值，其他参数的功能描述请参见 6.5.2 章节 CD522_32BIT_CNT 指令。



EN_0	EN_1	EN_FREQ	类型	1	2	3	4
FALSE	FALSE	TRUE	不进行测量	0	0	0	0
FALSE	TRUE	TRUE	2 个上升沿之间	不测量	2000 （Hz）/ 120000（rpm）	不测量	2222 （Hz）/ 133333（rpm）
TRUE	FALSE	TRUE	2 个下降沿之间	不测量	不测量	2850 （Hz）/ 171000（rpm）	不测量
TRUE	TRUE	TRUE	任意 2 个沿之间	不确定	不确定	不确定	不确定
FALSE	FALSE	FALSE	不进行测量	0	0	0	0
TRUE	FALSE	FALSE	低电平持续时间（脉宽）	不测量	200（μs）	不测量	300（μs)
FALSE	TRUE	FALSE	高电平持续时间（脉宽）	300（μs）	不测量	150（μs）	不测量
TRUE	TRUE	FALSE	任意 2 个沿之间	不确定	不确定	不确定	不确定

6.6 中断和高速计数模块 DC541 的高速计数器

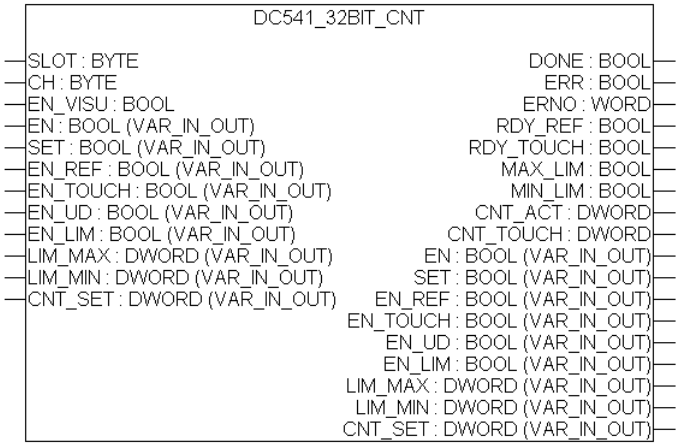
DC541 是一款多功能开关量 I/O 模块。与其他 I/O 模块不同，DC541 安装在 AC500 CPU 模块的左侧。DC541 具有 8 路可配置通道 C0...C7，可以根据需要配置为普通 DI/DO、中断输入或高速计数器。工作在高速计数器，支持以下功能：

- 32 位编码器
- 32 位正向计数器（频率 50KHz 或 5KHz）
- 时间和频率测量

6.6.1 DC541 高速计数器的操作模式

模式	占用的通道	描述	对应功能块
32 位编码器	C0：增量式编码器输出的 A 相 C1：增量式编码器输出的 B 相 C2、C3：REF 或 TOUCH 输入点	可连接一个增量式编码器。支持参考点逼近、触发/锁存功能。	DC541_32BIT_CNT
极限监视（32 位计数器）	C4~C7：其中的某个通道为输出通道	监视 32 位计数器的当前值与限制值的对比关系，并通过配置的输出点进行指示。	DC541_LIMIT
正向计数器	C0~C7：脉冲信号输入端	C0~C7 各通道均可以配置为正向计数器，对信号的上升沿进行加计数。	DC541_FWD_CNT

6.6.2 DC541_32BIT_CNT（DC541 32 位编码器指令）



1. 功能：DC541 模块的计数器工作于 32 位计数器模式时，可连接一个增量式编码器。通过此功能块进行管理。
2. 类型：标准功能块（FB，含历史数据）。
3. 所属库：DC541_AC500_V11.lib。
4. 参数说明

输入参数	数据类型	描述
SLOT	BYTE	DC541 所在插槽的编号。CPU 左侧第一个插槽的编号为 1。
CH	BYTE	高速计数的通道号（0）。0 是唯一有效的值。

输入参数	数据类型	描述
EN_VISU	BOOL	使能可视化界面 visuDC541_32BIT_CNT 的控制功能 FALSE：禁用可视化界面控制功能 TRUE：使能可视化界面控制功能
输入输出参数	**数据类型**	**描述**
EN	BOOL	使能功能块　　FALSE：无效 TRUE：高电平有效
SET	BOOL	设置计数器的设置值　　FALSE：无效 TRUE：高电平有效。保持为 TRUE 时，计数器的当前值保持不变（CNT_ACT=CNT_SET）
EN_TOUCH	BOOL	使能触发/锁存功能　　FALSE：功能无效 TRUE：高电平有效，上升沿使能一次
EN_REF	BOOL	使能参考点逼近功能　　FALSE：功能无效 TRUE：高电平有效，上升沿使能一次
EN_UD	BOOL	选择计数方向识别方式 FALSE：依靠 A 和 B 两个信号的顺序决定加或减计数。 TRUE：C1 通过的信号（B）加计数/C0 通过的信号（A）减计数
EN_LIM	BOOL	使能限制值　　FALSE：无效 TRUE：高电平有效
LIM_MAX	DWORD	计数器的上限值
LIM_MIN	DWORD	计数器的下限值
CNT_SET	DWORD	计数器设置值
输出参数	**数据类型**	**描述**
DONE	BOOL	功能块就绪信息
ERR	BOOL	功能块错误标志
ERNO	WORD	功能块故障代码
RDY_REF	BOOL	参考点归零功能有效标志，有效时为 TRUE
RDY_TOUCH	BOOL	锁存值更新标志，有新的锁存值时为 TRUE
MAX_LIM	BOOL	EN_LIM=FALSE 且 CNT_ACT> LIM_MAX 时 MAX_LIM=TRUE
MIN_LIM	BOOL	EN_LIM=FALSE 且 CNT_ACT< LIM_MIN 时 MIN_LIM=TRUE
CNT_ACT	DWORD	计数器的当前值
CNY_TOUCH	DWORD	计数器锁存的计数值

5. 描述

- EN_UD （选择计数方向识别方式）

EN_UD=FALSE：依靠 A 和 B 两个信号的相位差决定计数方向（C0 连接编码器 A 相，C1 连接编码器 B 相）。计数信号最高频率 50KHz。

EN_UD=TRUE：C1 连接的信号（B）加计数/C0 连接的信号（A）减计数。计数信号最高频率 50KHz。

- EN_LIM （使能限制值）

EN_LIM=FALSE：计数器作为无限计数器（无终止模式）。计数范围为 0~4,294,967,295（0~16#FFFFFFFF）。当 CNT_ACT> LIM_MAX 时，MAX_LIM=TRUE；当 CNT_ACT< LIM_MIN 时，MIN_LIM=TRUE。

EN_LIM=TRUE：计数器为限制模式。计数范围为 LIM_MIN~LIM_MAX。当 CNT_ACT 到达 LIM_MAX 时，计数器从 LIM_MIN 重新开始。

LIM_MAX 必须高于下限值 LIM_MIN，反之，计数器无法工作，同时输出 ERR/ERNO 显示错误信息。LIM_MAX 和 LIM_MIN 之间的差值应大于 DC541 检测循环时间内产生脉冲数量的两倍。

例如：

脉冲频率=40KHz =40000 个脉冲/s =40 个脉冲/ms

DC541 的循环时间=100 μs

因此，IM_MAX 和 LIM_MIN 之间的最小差值为 8。

- EN_TOUCH （使能触发/锁存功能）

如果 EN_TOUCH 使能，当在通道 C2 或 C3 出现上升沿时，功能块保存当前计数值 CNT_ACT，并在 CNT_TOUCH 中输出。CNT_TOUCH 的有效性通过输出 RDY_TOUCH 加以说明。

- EN_REF （使能参考点逼近功能）

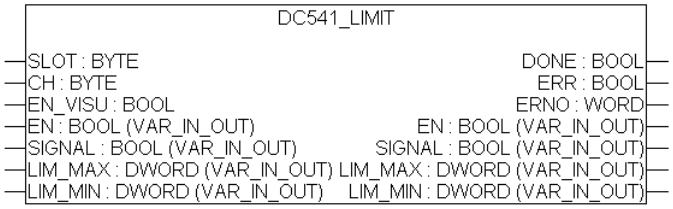
如果 EN_REF 使能，当在通道 C2 或 C3 出现上升沿时，CNT_ACT =CNT_SET。C2 或 C3 保持为 TRUE 时，计数器的当前值保持不变。

同一时刻，REF 和 TOUCH 只能激活一个功能，如果同时激活或者在其中一个功能未完成时激活了另一个功能，将在输出 ERR/ERNO 显示相应的错误信息。

- 其他参数的功能描述请参见 6.5.2 章节 CD522_32BIT_CNT 指令。

LD 语言环境中添加该指令应通过“带 EN 的框”调用。

6. 6. 3 DC541_LIMIT（DC541 32 位计数器极限监视指令）



1. 功能：监视 32 位计数器的当前值与限制值的对比关系。
2. 类型：标准功能块（FB，含历史数据）。
3. 所属库：DC541_AC500_V11.lib。
4. 参数说明

输入参数	数据类型	描述
SLOT	BYTE	DC541 所在插槽的编号。CPU 左侧第一个插槽的编号为 1
CH	BYTE	选择监视输出的通道。对于输出 C4~C7，有效值为 4~7
EN_VISU	BOOL	使能可视化界面 visuDC541_LIMIT 的控制功能 FALSE：禁用可视化界面控制功能 TRUE：使能可视化界面控制功能
输入输出参数	**数据类型**	**描述**
EN	BOOL	使能功能块　　FALSE：无效 TRUE：高电平有效
SIGNAL	BOOL	确定输出通道（CH）的状态
LIM_MAX	DWORD	计数器的上限值，应确保 LIM_MAX> LIM_MIN
LIM_MIN	DWORD	计数器的下限值

输出参数	数据类型	描述
DONE	BOOL	功能块完成标志，操作执行完毕为 TRUE，仅保持一个扫描周期
ERR	BOOL	功能块错误标志。该输出要与 DONE 一起综合判断。DONE 为 TRUE，且 ERR 为 TRUE，则说明操作完成但存在错误；DONE 为 TRUE，且 ERR 为 FALSE，则说明操作成功
ERNO	WORD	功能块操作存在错误的代码，代码含义请参见附录 A。只有 DONE 为 TRUE，且 ERR 也为 TRUE 时，ERNO 的输出值才有效

5. 描述

功能块的时间分辨率<100 μs，即可对频率 10 KHz 以下信号的测量结果进行监视。

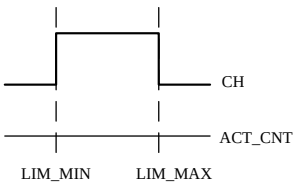
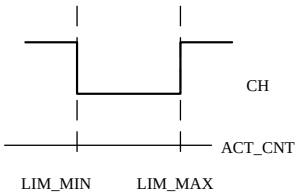
设备 DC541 必须被配置为计数设备，通道 C0 配置为 32 位计数器模式，输出 C4~C7 中的某个通道必须配置为限制通道 0。

- CH （输出通道）

输入 CH 用于选择监视输出的通道。对于输出 C4~C7，有效值为 4~7，并确保与“限制通道 0”的配置相一致。

- SIGNAL （确定输出通道的状态）

输入 SIGNAL，与 32 位计数器的当前值、限制值及 CH 的关系如下表所示：

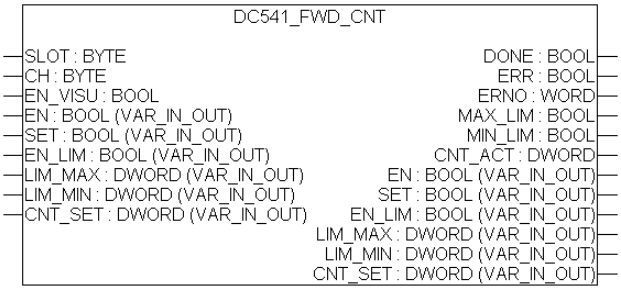
DC541_32BIT_CNT / ACT_CNT	当 SIGNAL=TRUE 时，输出 CH	当 SIGNAL=FALSE 时，输出 CH
ACT_CNT < LIM_MIN	FALSE	TRUE
LIM_MIN <=ACT_CNT <=LIM_MAX	TRUE	FALSE
ACT_CNT > LIM_MAX	FALSE	TRUE
		

当 SIGNAL 为 TRUE，如果 32 位计数器的当前值在限制值 LIM_MIN 和 LIM_MAX 给定的范围内，输出通道（CH）为 TRUE；如果计数器的当前值超过了此范围，输出通道（CH）为 FALSE。

当 SIGNAL 为 FALSE，如果 32 位计数器的当前值在限制值 LIM_MIN 和 LIM_MAX 给定的范围内，输出通道（CH）为 FALSE；如果计数器的当前值超过了此范围，输出通道（CH）为 TRUE。

LD 语言环境中添加该指令应通过“带 EN 的框”调用。

6.6.1 DC541_FWD_CNT（DC541 32 位加计数器指令）



1. 功能：DC541 模块的各个通道均可以配置为正向计数器，对信号的上升沿进行加计数，彼此独立。通过此功能块进行管理。
2. 类型：标准功能块（FB，含历史数据）。
3. 所属库：DC541_AC500_V11.lib。
4. 参数说明

输入参数	数据类型	描述
SLOT	BYTE	DC541 所在插槽的编号。CPU 左侧第一个插槽的编号为 1
CH	BYTE	DC541 的通道号。通道 C0~C7，对应的有效的值为 0~7
EN_VISU	BOOL	使能可视化界面 visuDC541_FWD_CNT 的控制功能 FALSE：禁用可视化界面控制功能 TRUE：使能可视化界面控制功能
输入输出参数	**数据类型**	**描述**
EN	BOOL	使能功能块　　FALSE：无效 TRUE：高电平有效
SET	BOOL	设置计数器的设置值　　FALSE：无效 TRUE：高电平有效。保持为 TRUE 时，计数器的当前值保持不变（CNT_ACT=CNT_SET）。
EN_LIM	BOOL	使能限制值　　FALSE：无效 TRUE：高电平有效
LIM_MAX	DWORD	计数器的上限值
LIM_MIN	DWORD	计数器的下限值
CNT_SET	DWORD	计数器设置值
输出参数	**数据类型**	**描述**
DONE	BOOL	功能块完成标志，操作执行完毕为 TRUE，仅保持一个扫描周期
ERR	BOOL	功能块错误标志。该输出要与 DONE 一起综合判断。DONE 为 TRUE，且 ERR 为 TRUE，则说明操作完成但存在错误；DONE 为 TRUE，且 ERR 为 FALSE，则说明操作成功
ERNO	WORD	功能块操作存在错误的代码，代码含义请参见附录 A。只有 DONE 为 TRUE，且 ERR 也为 TRUE 时，ERNO 的输出值才有效
MAX_LIM	BOOL	EN_LIM=FALSE 且 CNT_ACT> LIM_MAX 时 MAX_LIM=TRUE
MIN_LIM	BOOL	EN_LIM=FALSE 且 CNT_ACT< LIM_MIN 时 MIN_LIM=TRUE
CNT_ACT	DWORD	计数器的当前值

5. 描述

DC541 模块的各个通道均可以配置为一个 32 位的正向计数器，输入 C0 和 C1 的最大的计数频率为 50 KHz，输入 C2~C7 的计数频率为 5 KHz。DC541 内部为 16 位计数器，通过累加每个周期的计数差值，在功能块内部生成实际的当前值 ACT_CNT。为了避免错过脉冲信号的遗漏，应减小功能块的调用周期。各通道的最大调用周期按照以下方法计算：

通道 0、1：最大 50 KHz-> 32767 / 50 =655 ms

通道 2 ~ 7：最大 5 KHz-> 32767 / 5 =6550 ms

因此，使用循环时间小于 100ms 的任务调用该功能块，便可以防止任何计数脉冲的丢失。

- EN_LIM （使能限制值）

EN_LIM=FALSE：计数作为无限计数器（无终止模式）。计数范围为 0~4,294,967,295（0~16#FFFFFFFF）。当 CNT_ACT> LIM_MAX 时，MAX_LIM=TRUE；当 CNT_ACT< LIM_MIN 时，MIN_LIM=TRUE。

EN_LIM=TRUE：计数器为限制模式。计数范围为 LIM_MIN ~ LIM_MAX。当 CNT_ACT 到达 LIM_MAX 时，计数器从 LIM_MIN 重新开始。

LIM_MAX 必须高于下限值 LIM_MIN，反之，计数器无法工作，同时输出 ERR/ERNO 显示错误信息。LIM_MAX 和 LIM_MIN 之间的差值应大于 DC541 检测循环时间内产生脉冲数量的两倍。

例如：

脉冲频率=40 KHz =40000 个脉冲/s =40 个脉冲/ms

DC541 的循环时间=100 μs

因此，IM_MAX 和 LIM_MIN 之间的最小差值为 8。

- 其他参数的功能描述请参见 6.6.2 DC541_32BIT_CNT32 指令。

LD 语言环境中添加该指令应通过“带 EN 的框”调用。

第7章　　PWM 控制功能指令

7.1 PWM 控制概述

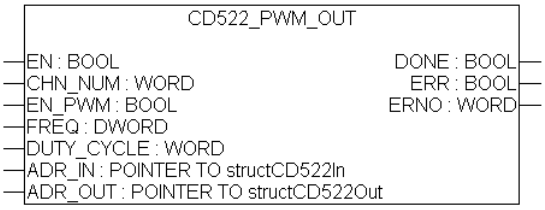
AC500 PLC 支持脉宽调制（PWM）方式和脉冲输出的运动控制功能，最高频率为 100KHz，提供多种实现途径。用户可根据需求灵活选择。

- 专用编码器模块 CD522，支持 PWM 输出或脉冲输出，频率最高可达 100KHz。
- 专用中断和高速计数模块 DC541，支持 PWM 输出或脉冲输出，频率最高可达 2.5KHz。
- eCo CPU PM5x4 的板载 I/O，PWM 输出的频率最高可达 20KHz。

7.2 CD522 模块及 DC541 模块的 PWM 输出指令

模式	占用的通道	描述	对应功能块
PWM 输出	CD522 模块的 O0、O1：两路均支持，彼此独立，特性相同。	频率及占空比均可调，频率最高可达 100KHz。	CD522_PWM_OUT
		仅频率可调，占空比固定为 50%，频率最高可达 100KHz。	CD522_FREQ_OUT
脉冲输出	CD522 模块的 O0、O1：两路均支持，彼此独立，特性相同。	频率以及脉冲个数可调，占空比固定为 50%，频率最高可达 15KHz。	CD522_PULSE_OUT
脉冲输出	DC541 模块的 C0~C7：各个通过均支持，彼此独立，特性相同。	频率以及脉冲个数可调，占空比固定为 50%，频率最高可达 2.5KHz。	DC541_FREQ_OUT

7.2.1　CD522_PWM_OUT（CD522 频率及占空比可调的 PWM 输出）



1. 功能：CD522 模块在 PWM 模式下提供两路独立的 PWM 输出，输出通道 O0 和通道 O1。通过此功能块完成对频率（1Hz ~100KHz）以及占空比的控制。
2. 类型：标准功能块（FB，含历史数据）。
3. 所属库：CD522_AC500_V13.lib。
4. 参数说明

输入参数	数据类型	描述
EN	BOOL	使能功能块　　FALSE：无效 TRUE：高电平有效
CNT_NUM	WORD	计数器选择。输出通道 O0、O1 对应的有效值为 0、1
EN_PWM	BOOL	使能 PWM 输出　　FALSE：不工作 TRUE：高电平有效
FREQ	DWORD	PWM 的频率。1Hz ~100000Hz（100KHz ）可调
DUTY_CYCLE	WORD	占空比。0 ~1000（0~100.0%）可调，不包括小数点。例如要设置 75.8%，则应写为 758。

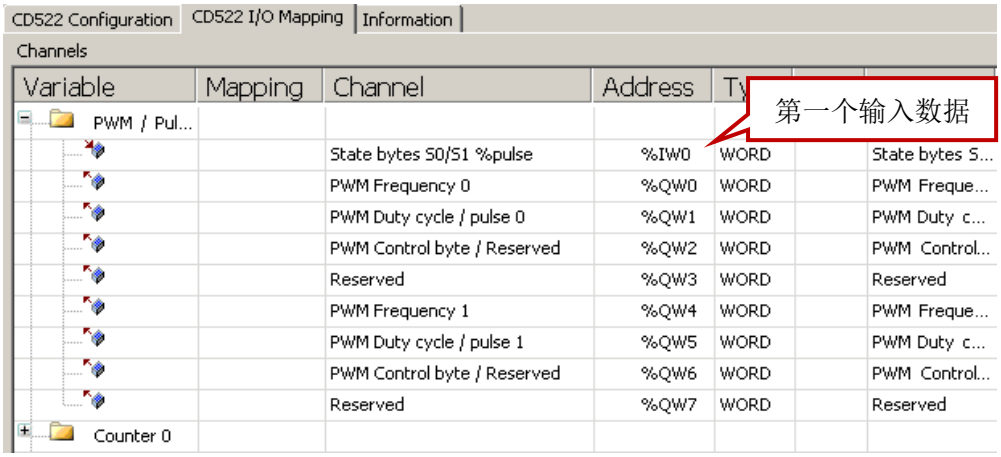
输入参数	数据类型	描述
ADR_IN	POINTER TO structCD522counterIn	指向 CD522 模块输入地址的指针
ADR_OUT	POINTER TO structCD522counterOut	指向 CD522 模块输出地址的指针
输出参数	**数据类型**	**描述**
DONE	BOOL	功能块完成标志，操作执行完毕为 TRUE，仅保持一个扫描周期
ERR	BOOL	功能块错误标志。该输出要与 DONE 一起综合判断。DONE 为 TRUE，且 ERR 为 TRUE，则说明操作完成但存在错误；DONE 为 TRUE，且 ERR 为 FALSE，则说明操作成功
ERNO	WORD	功能块操作存在错误的代码，代码含义请参见附录 A。只有 DONE 为 TRUE，且 ERR 也为 TRUE 时，ERNO 的输出值才有效

5. 描述

CD522 模块的输出通道 O0 和 O1 均支持 PWM 输出方式。且两通道的特性相同，彼此可以独立工作。

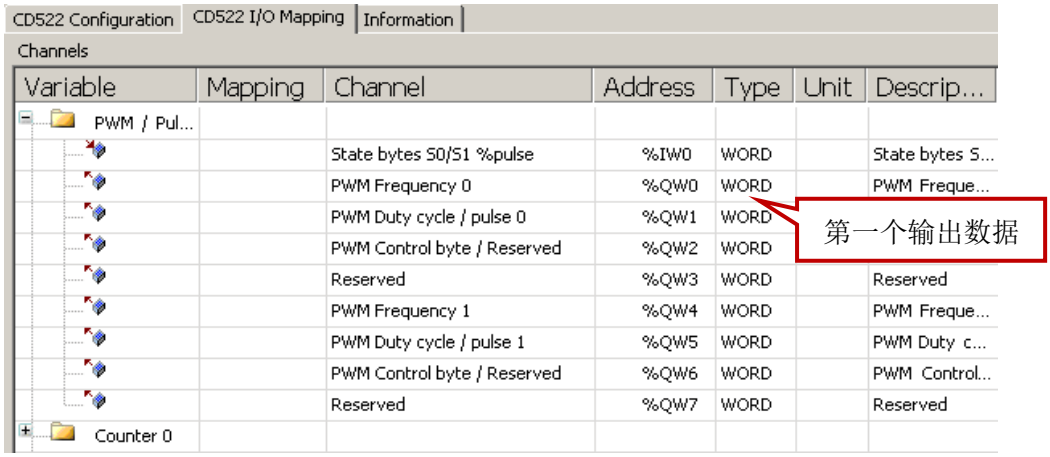
- ADR_IN （输入地址指针）

该输入指定 CD522 模块输入结构的第一个输入数据的地址，需要使用 ADR 运算符获得。如果输入 ADR_IN 没有连接，输出 ERR=TRUE。

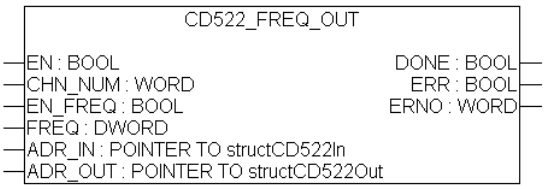


- ADR_OUT （输出地址指针）

该输入指定 CD522 模块输出结构的第一个输出数据的地址，需要使用 ADR 运算符获得。如果输入 ADR_OUT 没有连接，输出 ERR=TRUE。



7.2.2 CD522_FREQ_OUT（CD522 仅频率可调的 PWM 输出）



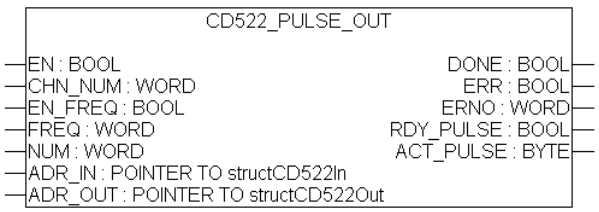
1. 功能：CD522 模块在 PWM 模式下提供两路独立的 PWM 输出，输出通道 O0 和通道 O1。通过此功能块完成对频率（1Hz ~100KHz）的控制，占空比固定为 50%。
2. 类型：标准功能块（FB，含历史数据）。
3. 所属库：CD522_AC500_V13.lib。
4. 参数说明

输入参数	数据类型	描述
EN	BOOL	使能功能块　　FALSE：无效 TRUE：高电平有效
CNT_NUM	WORD	计数器选择。输出通道 O0、O1 对应的有效值为 0、1
EN_FREQ	BOOL	使能 PWM 输出　　FALSE：不工作 TRUE：高电平有效
FREQ	DWORD	PWM 的频率。1Hz ~100000Hz（100KHz ）可调
ADR_IN	POINTER TO structCD522counterIn	指向 CD522 模块输入地址的指针
ADR_OUT	POINTER TO structCD522counterOut	指向 CD522 模块输出地址的指针
输出参数	**数据类型**	**描述**
DONE	BOOL	功能块完成标志，操作执行完毕为 TRUE，仅保持一个扫描周期
ERR	BOOL	功能块错误标志。该输出要与 DONE 一起综合判断。DONE 为 TRUE，且 ERR 为 TRUE，则说明操作完成但存在错误；DONE 为 TRUE，且 ERR 为 FALSE，则说明操作成功
ERNO	WORD	功能块操作存在错误的代码，代码含义请参见附录 A。只有 DONE 为 TRUE，且 ERR 也为 TRUE 时，ERNO 的输出值才有效

5. 描述

参数功能描述请参见 7.2.1 章节 CD522_PWM_OUT 指令。

7.2.3 CD522_PULSE_OUT（CD522 脉冲输出）



1. 功能：CD522 模块在脉冲输出模式下提供两路独立的方波（占空比固定为 50%）输出，输出通道 O0 和通道 O1。通过此功能块完成对频率以及脉冲个数的控制。

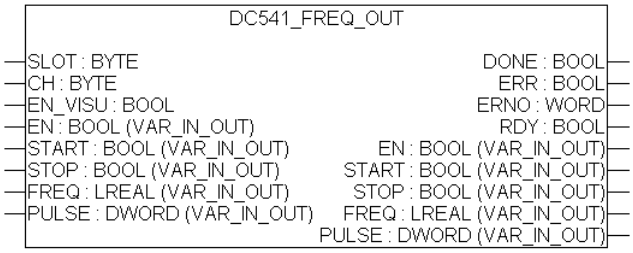
2. 类型：标准功能块（FB，含历史数据）。
3. 所属库：CD522_AC500_V13.lib。
4. 参数说明

输入参数	数据类型	描述
EN	BOOL	使能功能块　　FALSE：无效 TRUE：高电平有效
CNT_NUM	WORD	计数器选择。输出通道 O0、O1 对应的有效值为 0、1
EN_FREQ	BOOL	使能 PULSE 输出　　FALSE：不工作 TRUE：高电平有效
FREQ	WORD	脉冲的频率。1Hz ~15000Hz（15KHz）可调
NUM	WORD	设定的脉冲个数，有效范围 1~65535
ADR_IN	POINTER TO structCD522counterIn	指向 CD522 模块输入地址的指针
ADR_OUT	POINTER TO structCD522counterOut	指向 CD522 模块输出地址的指针
输出参数	**数据类型**	**描述**
DONE	BOOL	功能块完成标志，操作执行完毕为 TRUE，仅保持一个扫描周期
ERR	BOOL	功能块错误标志。该输出要与 DONE 一起综合判断。DONE 为 TRUE，且 ERR 为 TRUE，则说明操作完成但存在错误；DONE 为 TRUE，且 ERR 为 FALSE，则说明操作成功
ERNO	WORD	功能块操作存在错误的代码，代码含义请参见附录 A。只有 DONE 为 TRUE，且 ERR 也为 TRUE 时，ERNO 的输出值才有效
RDY_PULSE	BOOL	脉冲发送完成标志，设定的脉冲数发送完毕后置为 TRUE
ACT_PULSE	BYTE	已发出脉冲的脉冲个数，以百分比形式显示（0~100%）

5. 描述

参数功能描述请参见 7.2.1 章节 CD522_PWM_OUT 指令。

7.2.4 DC541_FREQ_OUT（DC541 仅频率可调的 PWM 输出）



1. 功能：DC541 模块的各个通道均可以配置为频率输出模式，彼此独立。通过此功能块完成对频率（0.5Hz ~2500Hz）以及脉冲个数的控制，占空比固定为 50%。
2. 类型：标准功能块（FB，含历史数据）。
3. 所属库：DC541_AC500_V11.lib。

4. 参数说明

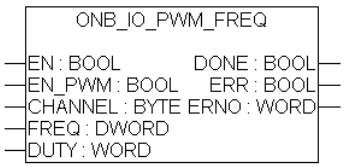
输入参数	数据类型	描述
SLOT	BYTE	DC541 所在插槽的编号。CPU 左侧第一个插槽的编号为 1
CH	BYTE	DC541 的通道号。通道 C0~C7，对应的有效的值为 0~7
EN_VISU	BOOL	使能可视化界面 visuDC541_FREQ_OUT 的控制功能 FALSE：禁用可视化界面控制功能 TRUE：使能可视化界面控制功能
输入输出参数	**数据类型**	**描述**
EN	BOOL	使能功能块　　FALSE：无效 TRUE：高电平有效
START	BOOL	启动输出　　FALSE：不进行输出 TRUE：启动输出，高电平有效。上升沿时从第一个脉冲开始输出
STOP	BOOL	终止输出　　FALSE：无效 TRUE：终止输出，高电平有效。优先级高于 START。
FREQ	LREAL	脉冲频率。0.5Hz ~2500Hz（2.5KHz ）可调
PULSE	DWORD	脉冲个数 =0：无限输出 >0：设定脉冲个数
输出参数	**数据类型**	**描述**
DONE	BOOL	功能块完成标志，操作执行完毕为 TRUE，仅保持一个扫描周期
ERR	BOOL	功能块错误标志。该输出要与 DONE 一起综合判断。DONE 为 TRUE，且 ERR 为 TRUE，则说明操作完成但存在错误；DONE 为 TRUE，且 ERR 为 FALSE，则说明操作成功
ERNO	WORD	功能块操作存在错误的代码，代码含义请参见附录 A。只有 DONE 为 TRUE，且 ERR 也为 TRUE 时，ERNO 的输出值才有效
RDY	BOOL	脉冲发送完成标志，设定的脉冲数发送完毕后置为 TRUE

7.3 板载 I/O 的 PWM 输出指令

eCo CPU 板载 I/O 中的晶体管输出最多提供两路 PWM 输出，输出通道 2（%QX4000.2）和输出通道 3（%QX4000.3）。支持频率和周期两种控制方式，如果输出通道 2 和 3 的 PWM 输出都需要使用，PWM 功能均必须激活且必须配置为相同的控制方式。

控制方式	占用的通道	描述	对应功能块
频率控制	%QX4000.2 和 QX4000.3	设置频率和占空比，最高频率 20KHz。	ONB_IO_PWM_FREQ
周期控制	%QX4000.2 和 QX4000.3	设置周期时间和占空比，最高频率 20KHz。	ONB_IO_PWM_TIME

7.3.1 ONB_IO_PWM_FREQ（板载 IO 的 PWM 频率方式输出）

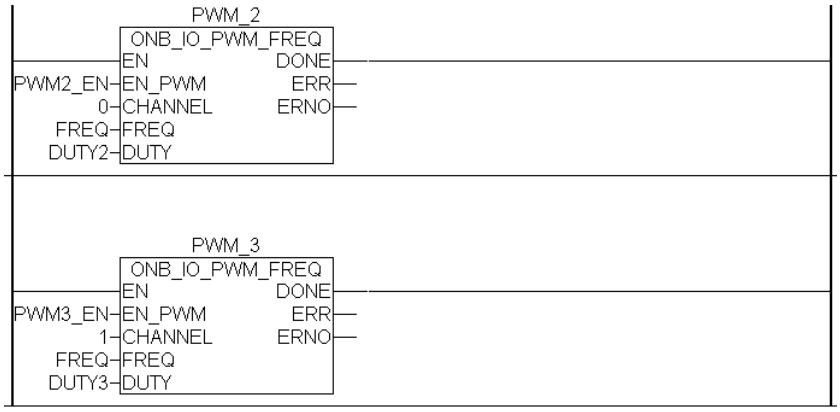


1. 功能：通过设置频率和占空比，产生频率 125Hz~20KHz，占空比 0~100%的脉冲。
2. 类型：标准功能块（FB，含历史数据）。
3. 所属库：OnBoardIO_AC500_V13.lib。
4. 参数说明

输入参数	数据类型	描述
EN	BOOL	使能功能块　FALSE：无效 TRUE：高电平有效
EN_PWM	BOOL	使能 PWM 输出　FALSE：禁用输出 TRUE：使能输出
CHANNEL	BYTE	板载 I/O 上 PWM 输出的通道 0：选择输出通道 2（%QX4000.2） 1：选择输出通道 3（%QX4000.3）
FREQ	DWORD	PWM 的频率。125Hz ~20000Hz（20KHz）可调
DUTY	WORD	占空比。0 ~1000（0~100.0%）可调，不包括小数点。例如要设置 75.8%，则应写为 758
输出参数	**数据类型**	**描述**
DONE	BOOL	功能块完成标志，操作执行完毕为 TRUE，仅保持一个扫描周期
ERR	BOOL	功能块错误标志。该输出要与 DONE 一起综合判断。DONE 为 TRUE，且 ERR 为 TRUE，则说明操作完成但存在错误；DONE 为 TRUE，且 ERR 为 FALSE，则说明操作成功
ERNO	WORD	功能块操作存在错误的代码，代码含义请参见附录 A。只有 DONE 为 TRUE，且 ERR 也为 TRUE 时，ERNO 的输出值才有效

5. 指令使用举例

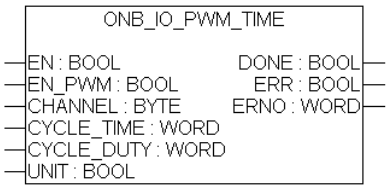
```
PWM_2: ONB_IO_PWM_FREQ;
PWM2_EN: BOOL;                  (* 使能PWM2输出 *)
DUTY2: WORD:=500 ;              (* PWM2脉冲的占空比，初值为50% *)
FREQ: DWORD := 1000;            (* PWM脉冲的频率，两个通道的频率必须相同。初值为1000Hz *)
PWM_3: ONB_IO_PWM_FREQ;
PWM3_EN: BOOL;                  (* 使能PWM3输出 *)
DUTY3:WORD:=600 ;               (* PWM3脉冲的占空比，初值为60% *)
```



6. 描述

可为输出通道 2 和通过 3 声明不同的功能块实例。两路脉冲的占空比可以不同，但频率必须相同。

7.3.2 ONB_IO_PWM_TIME（板载 IO 的 PWM 周期方式输出）



1. 功能：通过设置周期时间和占空比，同样可产生频率 125Hz~20KHz，占空比 0~100%的脉冲。
2. 类型：标准功能块（FB，含历史数据）。
3. 所属库：OnBoardIO_AC500_V13.lib。
4. 参数说明

输入参数	数据类型	描述
EN	BOOL	使能功能块　FALSE：无效 TRUE：高电平有效
EN_PWM	BOOL	使能 PWM 输出　FALSE：禁用输出 TRUE：使能输出
CHANNEL	BYTE	板载 I/O 上 PWM 输出的通道 0：选择输出通道 2（%QX4000.2） 1：选择输出通道 3（%QX4000.3）
CYCLE_TIME	WORD	PWM 的周期时间。周期时间从 8000 μs（8ms）至 50 μs 可调，即 125Hz ~20 KHz。当输入变量 UNIT=TRUE 时，时间单位为“ms”
CYCLE_DUTY	WORD	占空比。0 ~1000（0~100.0%）可调，不包括小数点。例如要设置 75.8%，则应写为 758
UNIT	BOOL	周期时间单位。　FALSE：μs TRUE：ms
输出参数	**数据类型**	**描述**
DONE	BOOL	功能块完成标志，操作执行完毕为 TRUE，仅保持一个扫描周期
ERR	BOOL	功能块错误标志。该输出要与 DONE 一起综合判断。DONE 为 TRUE，且 ERR 为 TRUE，则说明操作完成但存在错误；DONE 为 TRUE，且 ERR 为 FALSE，则说明操作成功
ERNO	WORD	功能块操作存在错误的代码，代码含义请参见附录 A。只有 DONE 为 TRUE，且 ERR 也为 TRUE 时，ERNO 的输出值才有效

5. 描述

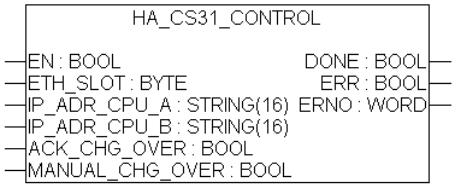
ONB_IO_PWM_TIME 与 ONB_IO_PWM_FREQ 的使用方法相同，可以声明多个实例，但要求周期时间相等。

第8章　　高可靠性系统 HA CS31 指令

AC500 高可靠性系统是根据高可靠性自动化系统的要求设计的，简称 HA 系统。HA CS31 库（HA_CS31_AC500_V20.lib）提供多条指令，用于管理和控制 HA 系统。

> HA_CS31_AC500_V20.lib 需要用户在库管理器中手动添加。

8.1 HA_CS31_CONTROL（HA 系统控制指令）



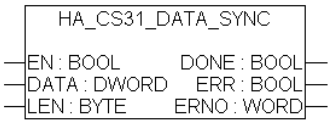
1. 功能：该功能块处理 AC500HA 系统操作，例如：一旦主 CPU 故障，控制命令需切换到备用 CPU，同时在 HA CPU 之间传输相关的诊断信息及数据。
2. 类型：标准功能块（FB，含历史数据）。
3. 参数说明

输入参数	数据类型	描述
EN	BOOL	使能功能块　FALSE：无效 TRUE：高电平有效
ETH_SLOT	BYTE	以太网槽号。集成通讯模块的编号为 0，所有扩展通讯模块从右向左计数，起始值为 1
IP_ADR_CPU_A	STRING[16]	连接到总线 A 的 AC500 CPU 的 IP 地址
IP_ADR_CPU_B	STRING[16]	连接到总线 B 的 AC500 CPU 的 IP 地址
ACK_CHG_OVER	BOOL	切换事件确认输入
MANUAL_CHG_OVER	BOOL	手动切换输入。在主 CPU 中，如果出现上升沿时，将强制主 CPU 切换为热备 CPU
输出参数	**数据类型**	**描述**
DONE	BOOL	功能块完成标志，操作执行完毕为 TRUE，仅保持一个扫描周期
ERR	BOOL	功能块错误标志。该输出要与 DONE 一起综合判断。 DONE 为 TRUE，且 ERR 为 TRUE，则说明操作完成但存在错误；DONE 为 TRUE，且 ERR 为 FALSE，则说明操作成功
ERNO	WORD	功能块操作存在错误的代码，代码含义请参见附录 A。只有 DONE 为 TRUE，且 ERR 也为 TRUE 时，ERNO 的输出值才有效

4. 描述

- 该功能块负责管理 HA 系统，因此必须被调用。
- 建议创建冗余处理程序，程序中调用该功能块。

8.2 HA_CS31_DATA_SYNC（HA 系统数据同步指令）



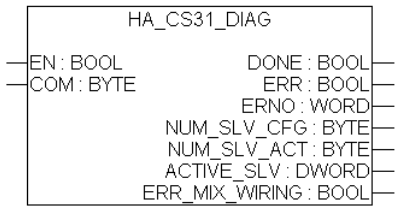
1. 功能：该功能块用于在 HA 系统中同步指定数据块。
2. 类型：标准功能块（FB，含历史数据）。
3. 参数说明

输入参数	数据类型	描述
EN	BOOL	使能功能块　FALSE：无效 TRUE：上升沿触发，高电平有效
DATA	DWORD	地址指针，指向需要同步的数据（可通过 ADR 指令获取该地址）
LEN	BYTE	数据长度，以字节为单位（可通过 SIZEOF 指令获取长度）
输出参数	**数据类型**	**描述**
DONE	BOOL	功能块完成标志，操作执行完毕为 TRUE，仅保持一个扫描周期
ERR	BOOL	功能块错误标志。该输出要与 DONE 一起综合判断。DONE 为 TRUE，且 ERR 为 TRUE，则说明操作完成但存在错误；DONE 为 TRUE，且 ERR 为 FALSE，则说明操作成功
ERNO	WORD	功能块操作存在错误的代码，代码含义请参见附录 A。只有 DONE 为 TRUE，且 ERR 也为 TRUE 时，ERNO 的输出值才有效

4. 描述

- DATA 读取需要同步的数据的首地址，LEN 读取长度信息。每个功能块可同步的最大数据量为 256 字节，如果同步数据较多，可声明多个实例。
- 交换数据最大 1024 个字节。
- 建议创建冗余处理程序，程序中调用该功能块。

8.3 HA_CS31_DIAG（读取 HA 系统诊断信息）



1. 功能：该功能块读取 CS31 总线上各 CI590-CS31-HA 模块的状态字节。
2. 类型：标准功能块（FB，含历史数据）。
3. 参数说明

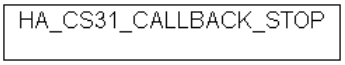
输入参数	数据类型	描述
EN	BOOL	使能功能块　FALSE：无效 TRUE：上升沿触发，高电平有效
COM	BYTE	指定串行接口的编号　1 ：COM1 接口 2 ：COM2 接口
输出参数	**数据类型**	**描述**
DONE	BOOL	功能块完成标志，操作执行完毕为 TRUE，仅保持一个扫描周期

输出参数	数据类型	描述
ERR	BOOL	功能块错误标志。该输出要与 DONE 一起综合判断。DONE 为 TRUE，且 ERR 为 TRUE，则说明操作完成但存在错误；DONE 为 TRUE，且 ERR 为 FALSE，则说明操作成功
ERNO	WORD	功能块操作存在错误的代码，代码含义请参见附录 A。只有 DONE 为 TRUE，且 ERR 也为 TRUE 时，ERNO 的输出值才有效
NUM_SLV_CFG	BYTE	PLC 中配置的 CI590 从站数量
NUM_SLV_ACT	BYTE	CS31 总线上激活的 CI590 从站数量
ACTIVE_SLV	DWORD	双字中的每个位代表一个 CS31 总线上激活的 CI590 设备
ERR_MIX_WIRING	BOOL	CS31 从站的总线 A 与总线 B 之间存在交叉接线错误标志

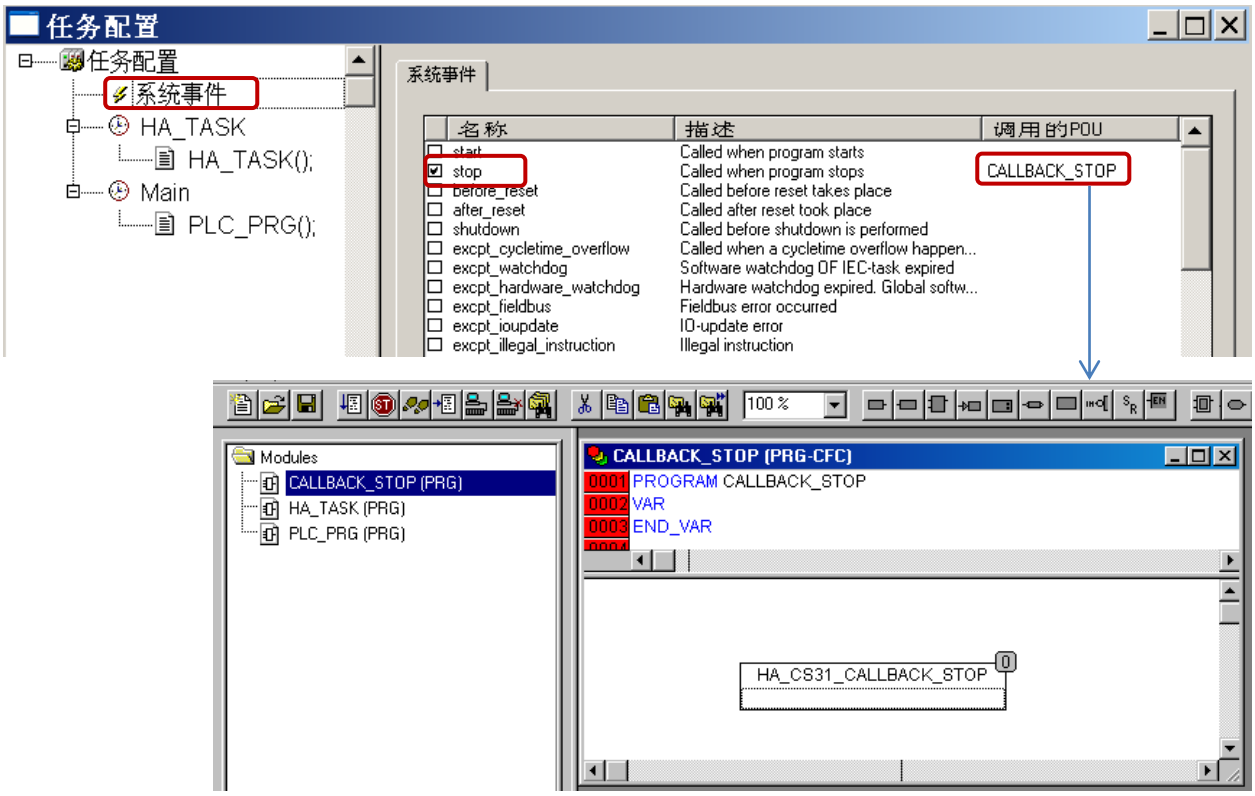
4. 描述

   建议创建冗余处理程序，程序中调用该功能块。

8.4 HA_CS31_CALLBACK_STOP （HA 系统 CPU 停止事件调用程序）



1. 功能：处理 AC500 HA 工程在 CPU STOP 模式时的相关逻辑。
2. 类型：标准程序（PRG）。
3. 指令使用举例



- 通过 CPU 停止系统事件调用 CALLBACK_STOP；
- CALLBACK_STOP 中调用该程序。
- HA 工程必须调用该程序。

CPU 停止事件调用的程序强制命名为：“CALLBACK_STOP”（不区分大小写）

8.5 冗余数据标准功能块指令

HA 高可靠性系统中与时间相关的数据应该包含在 2 个 CPU 中，即成为冗余数据。冗余数据包括以下三种：

- 积分值
- 计数值
- 时序值

用户程序中如果需要对冗余数据进行操作，应选择以下标准功能块，以便实现数据的同步。各参数的说明请参见对应的非冗余标准功能块。

- HA_CS31_PID
- HA_CS31_PID_FIXCYCLE
- HA_CS31_INTEGRAL
- HA_CS31_RAMP_REAL
- HA_CS31_RAMP_INT
- HA_CS31_CTUD
- HA_CS31_CTU
- HA_CS31_CTD
- HA_CS31_TON
- HA_CS31_TOF

第9章　　附件操作指令

9.1 实时时钟指令 Realtime clock（SysExt_AC500_V10.LIB）

9.1.1　CLOCK（设置/读取时钟和日期）

1. 功能：通过普通数据格式设置和显示当前时间和日期。当输入变量 SET 由 FALSE 变为 TRUE 时，设定的时间和日期值生效。只要 EN 输入为 TRUE，功能块输出便显示当前的日期和时间。
2. 类型：标准功能块（FB，含历史数据）。
3. 参数说明

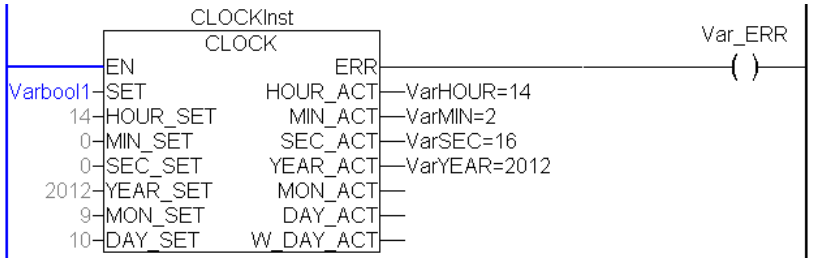
输入参数	数据类型	描述
EN	BOOL	使能功能块　FALSE：无效 TRUE：使能，读取当前时间和日期
SET	BOOL	TRUE：高电平有效。出现上升沿时设定的时间和日期值生效
HOUR_SET	BYTE	设置小时（有效数值范围：0...23）
MIN_SET	BYTE	设置分钟（有效数值范围：0...59）
SEC_SET	BYTE	设置秒数（有效数值范围：0...59）
YEAR_SET	WORD	设定年份（必须输入 4 位数字，如 2012）
MON_SET	BYTE	设定月份（有效数值范围：1...12）
DAY_SET	BYTE	设定日期（有效数值范围：1...31）
输出参数	**数据类型**	**描述**
ERR	BOOL	功能块错误标志　TRUE：未取出数据 FALSE：正确取出数据
HOUR_ACT	BYTE	当前时间——小时
MIN_ACT	BYTE	当前时间——分钟
SEC_ACT	BYTE	当前时间——秒
YEAR_ACT	WORD	当前时间——年
MON_ACT	BYTE	当前时间——月
DAY_YCT	BYTE	当前时间——日
W_DAY_ACT	BYTE	当前时间——星期

4. 指令使用举例

变量定义

VAR | VAR_INPUT | VAR_OUTPUT | VAR_IN_OUT

	名称	地址	类型	初值	注释
0001	CLOCKInst		CLOCK		
0002	Varbool1		BOOL		
0003	Var_ERR		BOOL		
0004	VarHOUR		BYTE		
0005	VarMIN		BYTE		
0006	VarSEC		BYTE		
0007	VarYEAR		WORD		

编程语言	程　序
梯形图（LD）	
结构化文本（ST）	ClockInst (EN:=true,SET:=Varbool1,HOUR_SET:=14, YEAR_SET:=2012, MON_SET :=9,DAY_SET :=10); Var_ERR:=ClockInst.ERR; VarHOUR:=ClockInst.HOUR_ACT; VarMIN:=ClockInst.MIN_ACT; VarSEC:=ClockInst.SEC_ACT; VarYEAR:=ClockInst.YEAR_ACT;

9.1.2　CLOCK_DT（使用 DT 格式设置/读取时钟和日期）

1. 功能：通过标准“DT”格式设置和显示当前时间和日期。通过 DT_SET 设置时钟。当输入变量 SET 由 FALSE 变为 TRUE 时，设定的时间和日期值生效。只要 EN 输入为 TRUE，输出“DT_ACT”便以“DT”格式显示当前的日期和时间。
2. 类型：标准功能块（FB，含历史数据）。
3. 参数说明

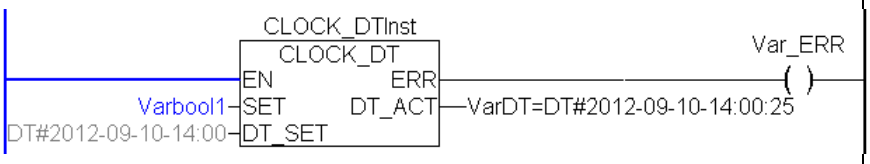
输入参数	数据类型	描述
EN	BOOL	使能功能块　FALSE：无效 TRUE：使能，读取当前时间和日期
SET	BOOL	TRUE：高电平有效。出现上升沿时设定的时间和日期值生效
DT_SET	DT	以“DT”格式设置时间和日期
输出参数	**数据类型**	**描述**
ERR	BOOL	功能块错误标志　TRUE：未取出数据 FALSE：正确取出数据
DT_ACT	DT	以“DT”格式表示的当前的时间和日期

4. 指令使用举例

变量定义

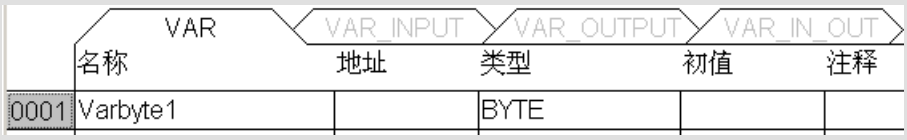
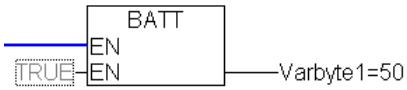
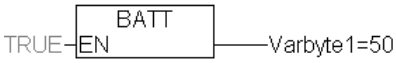
VAR | VAR_INPUT | VAR_OUTPUT | VAR_IN_OUT

	名称	地址	类型	初值	注释
0001	CLOCK_DTInst		CLOCK_DT		
0002	Varbool1		BOOL		
0003	Var_ERR		BOOL		
0004	VarDT		DT		

编程语言	程　序
梯形图（LD）	
结构化文本（ST）	Clock_DTInst(EN:=TRUE,SET:=Varbool1,DT_SET:=dt#2012-09-10-14:0:0); Var_ERR:=CLOCK_DTInst.ERR; VarDT:=CLOCK_DTInst.DT_ACT;

9.2 电池维护指令（SysExt_AC500_V10.LIB）

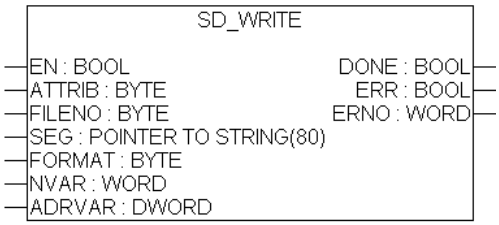
1. BATT：读取电池的电量状态
2. 类型：标准功能（FUN）。
3. 输入变量类型：BOOL 类型。
4. 返回值类型：BYTE 类型。
5. 指令使用举例

变量定义	
	
编程语言	**程　序**
梯形图（LD）	
结构化文本（ST）	Varbyte1:=BATT(TRUE);　（*结果 Varbyte1 为 50*）
功能块（FBD）	

9.3 SD 卡操作指令（SysInt_AC500_V10.LIB）

系统库 SysInt_AC500_V10.lib 中，提供两个功能块用于直接对 SD 卡的文件进行操作，SD_WRITE 用于向 SD 卡中写数据，SD_READ 用于从 SD 卡读取数据。所操作的目录为 USERDATA/PM5xx/USERDAT，保存的文件名格式统一为 USRDATxx.DAT（0≤xx≤99）。使用 SD_WRITE 或 SD_READ 最多操作 100 个文件。

9.3.1 SD_WRITE（向 SD 卡写入一个数据段）



1. 功能：按照固定的格式写入数据，支持以下预定义格式：第一种不具有段标签（Section Lable），这种格式每条数据记录无段标签，每条数据记录占用一行。第二种具有段标签（Section Lable），这种格式可以通过段标签来区分不同组的数据记录，在文件中段标签占用一行，数据记录占用一行。
2. 类型：标准功能块（FB，含历史数据）。
3. 参数说明

输入参数	数据类型	描述
EN	BOOL	使能功能块　FALSE：无效 TRUE：高电平有效。上升沿时执行一次操作
ATTRIB	BYTE	写入功能块的属性。 1：删除文件 （删除） 2：写入数据集 3：写入段标签
FILENO	BYTE	SD 卡存储 USRDATXX.DAT 文件的编号。设置范围 0~99
SEG	DWORD	段标签地址，段标签用户可自定义（可通过 ADR 指令获取该地址）
FORMAT	BYTE	数据记录的格式 00 十六进制（0 十进制）：BYTE 01 十六进制（1 十进制）：CHAR 10 十六进制（16 十进制）：WORD 11 十六进制（17 十进制）：INT 20 十六进制（32 十进制）：DWORD 21 十六进制（33 十进制）：DINT
NVAR	WORD	每个数据集的元素个数
ADRVAR	DWORD	地址指针，指向被记录数据的存储区首地址（可通过 ADR 指令获取该地址）
输出参数	**数据类型**	**描述**
DONE	BOOL	功能块完成标志，操作执行完毕为 TRUE，仅保持一个扫描周期
ERR	BOOL	功能块错误标志。该输出要与 DONE 一起综合判断。 DONE 为 TRUE，且 ERR 为 TRUE，则说明操作完成但存在错误；DONE 为 TRUE，且 ERR 为 FALSE，则说明操作成功
ERNO	WORD	功能块操作存在错误的代码，代码含义请参见附录 A。只有 DONE 为 TRUE，且 ERR 也为 TRUE 时，ERNO 的输出值才有效

4. 描述

SD_WRITE 功能块将数据集写入 SD 卡固定路径的以下文件：

...\UserData\PM5x1\UserDat\USRDATxx.DAT

这些文件与 EXCEL 兼容且可读，每个数据以 ASCII 码格式存储，并自动地用分号分隔。最后一个数据以 <CR><LF>结束。

启动一个数据集的写入操作时（通过输入 EN 的上升沿触发），数据集的数据必须保持不变，直至完成（DONE =TRUE）。向 SD 卡存储一个数据集需要几个 PLC 周期。数据写入期间系统不再对输入变量 EN 进行处理。

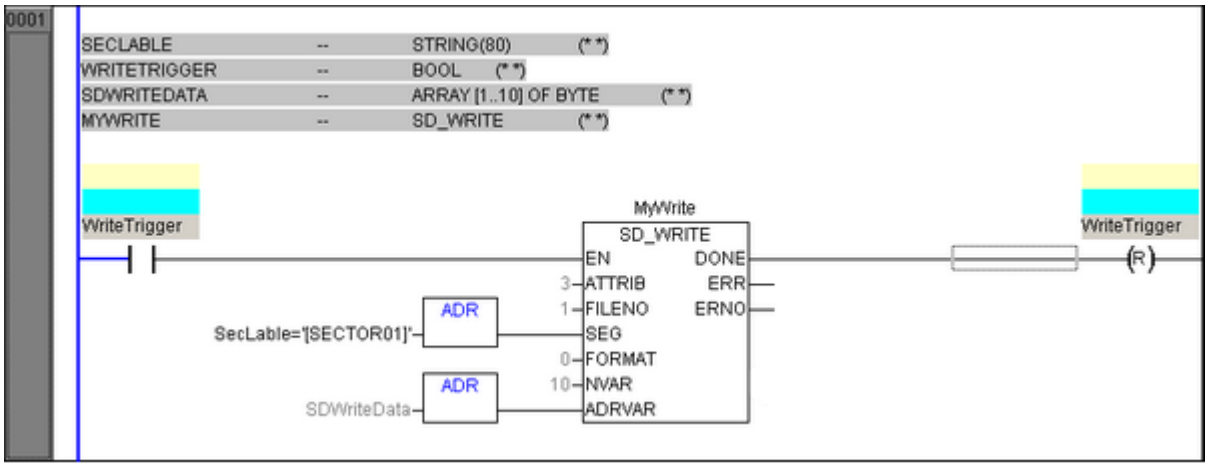
程序中如果存在该功能块的多个实例时，应在逻辑上进行互锁，即：一定要保证在某一时间内只有一个功能块实例处于使能状态。

如果在写访问过程中突然断电，文件 USRDATxx.DAT 将会被损坏。建议对已经存储的数据进行备份。

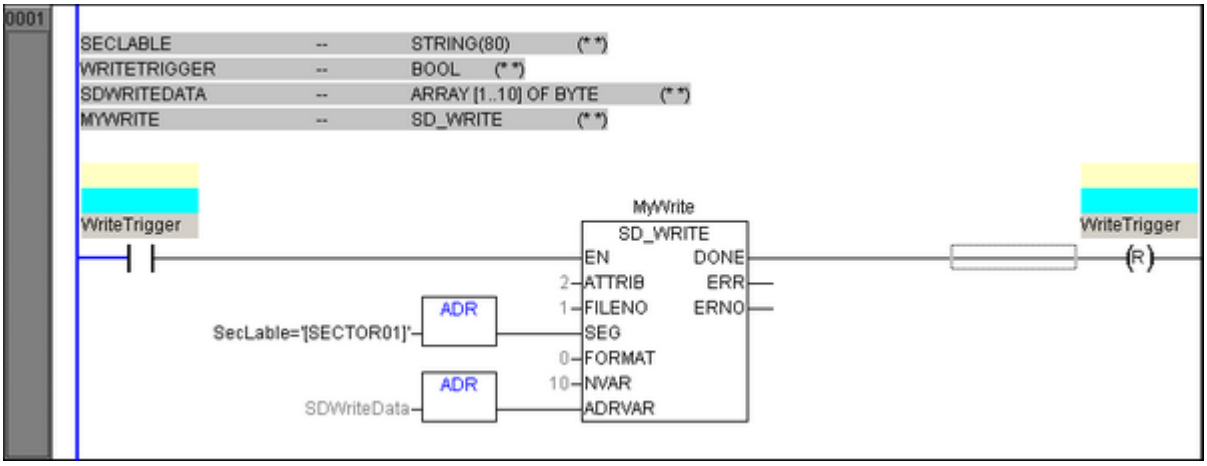
5. 指令使用举例

下例中向 SD 卡写入具有段标签的数据。

```
0001 PROGRAM PLC_PRG
0002 VAR
0003     WriteTrigger:BOOL;
0004     SDWriteData:ARRAY[1..10] OF BYTE:=1,2,3,4,5,6,7,8,9,10;
0005     SecLable:STRING:='[SECTOR01]';
0006     MyWrite:SD_WRITE;
0007 END_VAR
```



- 将 ATTRIB 设置为 3，WriteTrigger 出现上升沿时首先向 SD 卡中写入段标签名（本例为 SECTOR01）。
- 写入过程 CPU 的指示灯“RUN”以慢闪状态进行提示，结束后（DONE =TRUE）“RUN”常亮，表示段标签写入完成。
- 从 CPU 中取出 SD 卡，插入电脑中的读卡器，查看 USERDATA/PM5xx/USERDAT 写入的文件（本例中 FILENO 为 1 对应 USRDAT1.DAT）如下：



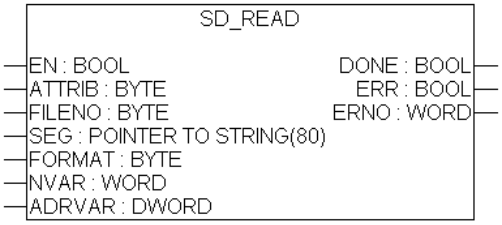
- 将 ATTRIB 改为 2，WriteTrigger 再出现上升沿时，向 SD 卡中写入 BYTE 格式（FORMAT 设为 3）的 10 个数据（NVAR 为 10）。
- 写入过程 CPU 的“RUN”灯以慢闪状态进行提示，结束后（DONE =TRUE）“RUN”灯常亮，表示数据写入完成。
- 再次查看 USERDATA/PM5xx/USERDAT 写入的文件（本例 USRDAT1.DAT）如下：

```
1 [SECTOR01]
2 1;2;3;4;5;6;7;8;9;10
```

- 重复上述步骤可以写入多个具有段标签和数据记录。

> 在同一个数据记录内的所有数据类型必须一致，具有段标签的数据记录，每个标签内只能有一条数据记录，且数据记录内的所有数据类型必须一致，但段之间可以不一致。

9.3.2 SD_READ（从 SD 卡读取一个数据段）



1. 功能：功能块 SD_READ 可从 SD 卡的文件中读取一个数据集，并存储所读取的数据。
2. 类型：标准功能块（FB，含历史数据）。
3. 参数说明

输入参数	数据类型	描述
EN	BOOL	使能功能块 FALSE：无效 TRUE：高电平有效。上升沿时执行一次操作
ATTRIB	BYTE	写入功能块的属性： 1：打开文件，并定位段标签号，且读取一个数据集 2：打开文件，并读取无段标签的数据集 3：继续读取下一个数据集 4：继续读取下一个数据集并关闭文件 5：关闭文件

输入参数	数据类型	描述
FILENO	BYTE	SD 卡存储 USRDATXX.DAT 文件的编号。设置范围 0~99
SEG	DWORD	段标签地址，段标签用户可自定义（可通过 ADR 指令获取该地址）
FORMAT	BYTE	数据记录的格式 00 十六进制（0 十进制）：BYTE 01 十六进制（1 十进制）：CHAR 10 十六进制（16 十进制）：WORD 11 十六进制（17 十进制）：INT 20 十六进制（32 十进制）：DWORD 21 十六进制（33 十进制）：DINT
NVAR	WORD	每个数据集的元素个数
ADRVAR	DWORD	地址指针，指向被记录数据的存储区首地址（可通过 ADR 指令获取该地址）
输出参数	**数据类型**	**描述**
DONE	BOOL	功能块完成标志，操作执行完毕为 TRUE，仅保持一个扫描周期
ERR	BOOL	功能块错误标志。该输出要与 DONE 一起综合判断。 DONE 为 TRUE，且 ERR 为 TRUE，则说明操作完成但存在错误；DONE 为 TRUE，且 ERR 为 FALSE，则说明操作成功
ERNO	WORD	功能块操作存在错误的代码，代码含义请参见附录 A。只有 DONE 为 TRUE，且 ERR 也为 TRUE 时，ERNO 的输出值才有效

4. 描述

SD_READ 功能块可读取 SD 卡固定路径文件的数据集：

...\UserData\PM5x1\UserDat\USRDATxx.DAT

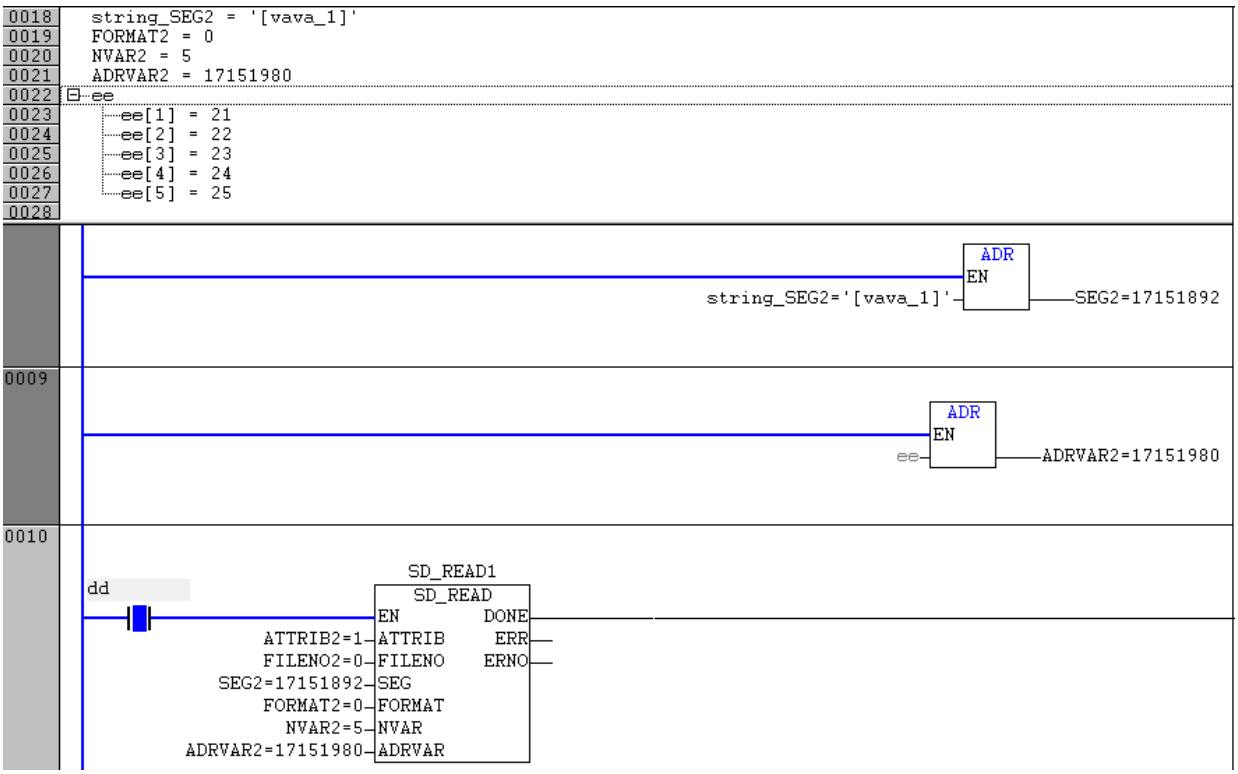
通过输入 EN 的上升沿触发数据集的读取操作。读取操作需要几个 PLC 周期，过程中系统不再对输入变量 EN 进行处理。

程序中如果存在该功能块的多个实例时，应在逻辑上进行互锁，即：一定要保证在某一时间内只有一个功能块实例处于使能状态。

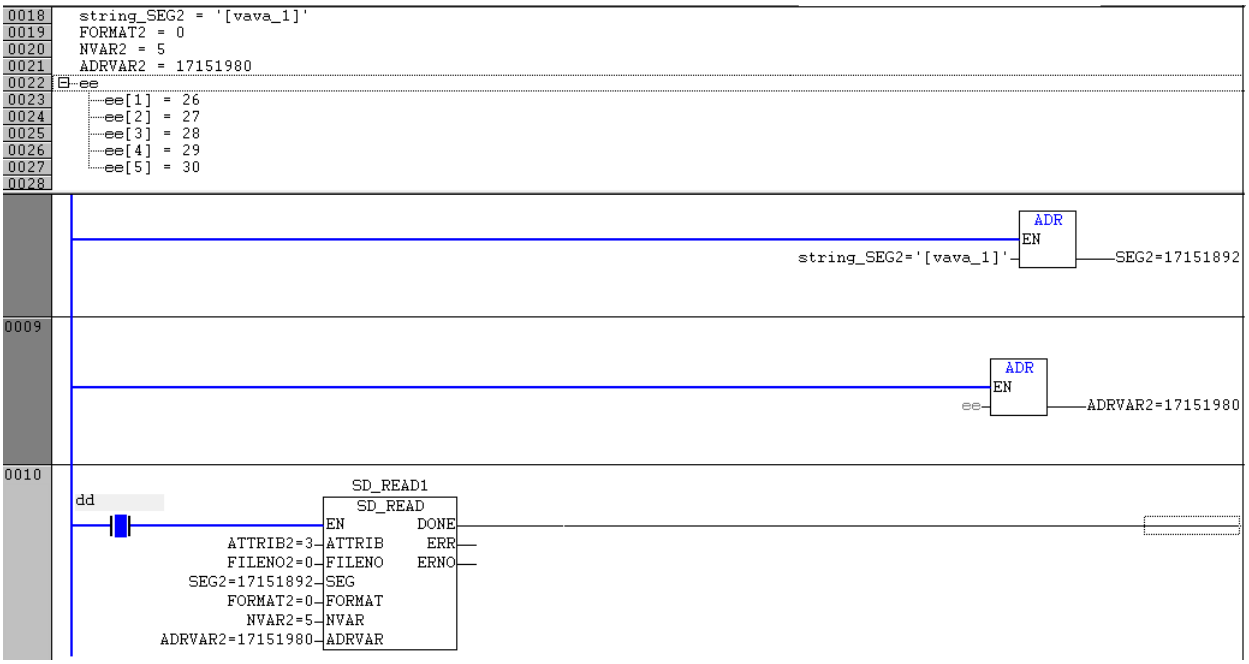
5. 指令使用举例

以下举例说明如何从 SD 卡读取带段标签的数据。

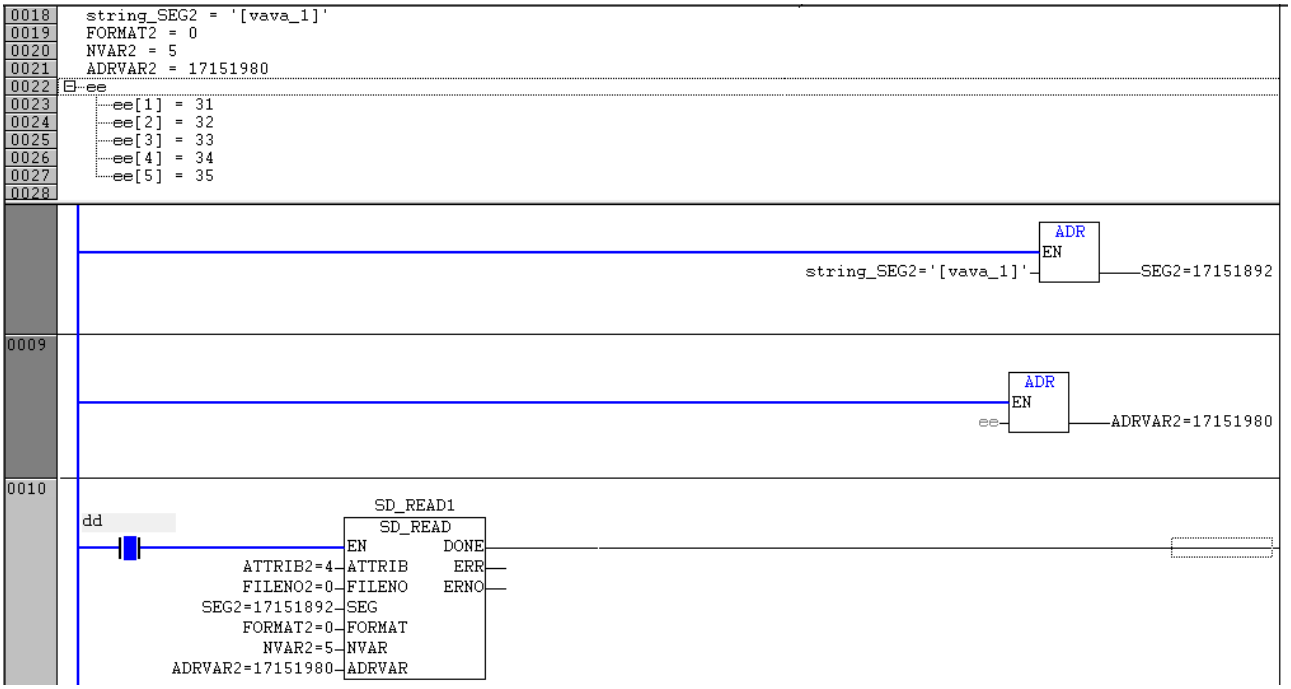
```
0017    dd: BOOL;
0018    SD_READ1: SD_READ;
0019    string_SEG2:STRING;
0020    ee:ARRAY[1..5] OF BYTE;
0021    ATTRIB2: BYTE;
0022    FILENO2: BYTE;
0023    SEG2: DWORD;
0024    FORMAT2: BYTE;
0025    NVAR2: BYTE;
0026    ADRVAR2: DWORD;
```



- 将 ATTRIB2 设为 1，表示打开文件，查找段标签号，并读取数据集。
- 将 FILENO2 设为 1，表示读取 SD 卡中的 USRDAT0.DAT 文件
- SEG2 对应段标签的地址（本例为 vava_1）
- FORMAT2 设为 0，表示读取字节类型的数据记录
- NVAR2 设为 5，表示读取数据的个数
- ADRVAR2 对应存储记录数据的变量地址（本例定义为 ee）
- dd 的上升沿触发操作，读取带 vava_1 标签的一条数据集，数据保存在变量 ee 中。在读取过程中 CPU 的指示灯“RUN”慢闪，待“RUN”常亮，表示数据读取完成。



- 将 ATTRIB2 设为 3，表示读取下一条数据集，继续读取 vava_1 之后的数据。dd 的上升沿触发读取操作。



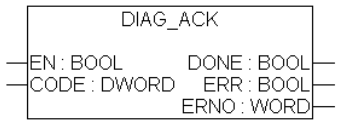
- 再将 ATTRIB2 设为 4，读取下一条数据集并关闭。可继续读取 vava_1 下的数据并关闭。dd 的上升沿触发。

第10章　　PLC 故障处理指令

内部系统库中（SysInt_AC500_V10.lib）提供用于处理 PLC 故障的指令，如确认故障信息、读取故障信息、复位诊断系统等。

10.1 DIAG_ACK（确认故障信息）

1. 功能：确认故障信息。
2. 类型：标准功能块（FB，含历史数据）。
3. 参数说明



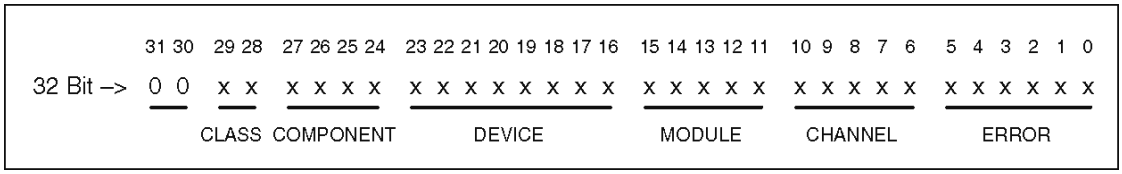
输入参数	数据类型	描述
EN	BOOL	使能功能块　FALSE：无效 TRUE：上升沿触发
CODE	DWORD	待确认的错误代码
输出参数	**数据类型**	**描述**
DONE	BOOL	功能块完成标志，操作执行完毕为 TRUE，仅保持一个扫描周期
ERR	BOOL	功能块错误标志。该输出要与 DONE 一起综合判断。DONE 为 TRUE，且 ERR 为 TRUE，则说明操作完成但存在错误；DONE 为 TRUE，且 ERR 为 FALSE，则说明操作成功
ERNO	WORD	功能块操作存在错误的代码，代码含义请参见附录 A。只有 DONE 为 TRUE，且 ERR 也为 TRUE 时，ERNO 的输出值才有效

4. 描述

- CODE　DWORD　（待确认的错误代码）

通过 CODE 输入待确认的错误代码。现存错误代码可通过 DIAG_GET 功能块读取，或者手动查询。

- 错误代码的结构



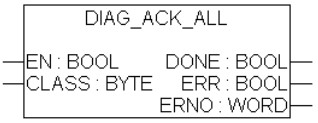
位数	含义	有效值
位 0 到 5	错误标识（ERROR）	0~63
位 6 至 10	通道编号（CHANNEL）	0~31
位 11 到 15	模块编号（MODULE）	0~31
位 16 到 23	设备编号（DEVICE）	0~255
位 24 到 27	组件编号（COMPONENT）	0~15
位 28 到 29	故障等级（CLASS）	1~4
位 30 到 31	保留	0

- 错误标识（ERROR）的含义

错误标识（ERROR）	含义	错误标识（ERROR）	含义
0	通常错误	27	无配置
1	错误值	28	实际/配置不同
2	无效值	29	写配置时出错
3	超时	30	读配置时出错
4	最高级别错误	31	错误的类型/模块
5	高级别错误	32	未知的类型/模块
6	低级别错误	33	等待复位
7	最低级别错误	34	等待就绪
8	空或丢失	35	等待运行
9	溢出	36	等待通讯
10	太大	37	循环时间错误
11	太小	38	例外
12	读取时出错	39	未知的 POU
13	写入时出错	40	版本错误
14	删除时出错	41	发送错误
15	分配内存时出错	42	接收错误
16	释放内存时出错	43	内部错误
17	访问时出错	44	无可调值
18	测试错误	45	断路
19	校验和错误	46	过载
20	信息错误	47	短路
21	发出信息时出错	48	过载/断路
22	收到信息错误	49	短路/断路
23	等待消息错误	50	过载/短路
24	删除信息错误	51	过载/短路/断路
25	等待应答错误	52	丢失值
26	配置错误	63	其他

10.2 DIAG_ACK_ALL（确认同一等级的全部故障信息）

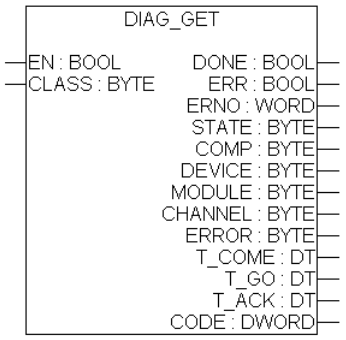
1. 功能：确认某一等级的全部故障信息。
2. 类型：标准功能块（FB，含历史数据）。
3. 参数说明



输入参数	数据类型	描述
EN	BOOL	使能功能块　FALSE：无效 TRUE：上升沿触发
CLASS	BYTE	待确认的故障等级，有效范围 1~4

输出参数	数据类型	描述
DONE	BOOL	功能块完成标志，操作执行完毕为 TRUE，仅保持一个扫描周期
ERR	BOOL	功能块错误标志。该输出要与 DONE 一起综合判断。DONE 为 TRUE，且 ERR 为 TRUE，则说明操作完成但存在错误；DONE 为 TRUE，且 ERR 为 FALSE，则说明操作成功
ERNO	WORD	功能块操作存在错误的代码，代码含义请参见附录 A。只有 DONE 为 TRUE，且 ERR 也为 TRUE 时，ERNO 的输出值才有效

10.3 DIAG_GET（读取故障信息）



1. 功能：读取任意故障等级中未被读取的故障信息。
2. 类型：标准功能块（FB，含历史数据）。
3. 参数说明

输入参数	数据类型	描述
EN	BOOL	使能功能块　FALSE：无效 TRUE：上升沿触发
CLASS	BYTE	设定故障等级，有效范围 1~4
输出参数	**数据类型**	**描述**
DONE	BOOL	功能块完成标志，操作执行完毕为 TRUE，仅保持一个扫描周期
ERR	BOOL	功能块错误标志。该输出要与 DONE 一起综合判断。DONE 为 TRUE，且 ERR 为 TRUE，则说明操作完成但存在错误；DONE 为 TRUE，且 ERR 为 FALSE，则说明操作成功
ERNO	WORD	功能块操作存在错误的代码，代码含义请参见附录 A。只有 DONE 为 TRUE，且 ERR 也为 TRUE 时，ERNO 的输出值才有效
STATE	BYTE	故障的状态
COMP	BYTE	故障的组件编号
DEVICE	BYTE	故障的设备编号
MODULE	BYTE	设定故障的模块编号
CHANNEL	BYTE	设定故障的通道
ERROR	BYTE	设定故障的错误标识
T_COME	DT	故障发生时间。无有效输出时保留默认时间 DT#1970-01-01-00:00
T_GO	DT	故障消失时间，无有效输出时保留默认时间 DT#1970-01-01-00:00
T_ACK	DT	故障确认时间，无有效输出时保留默认时间 DT#1970-01-01-00:00
CODE	DWORD	错误代码

4. 描述

该功能块用于读取任何等级的故障信息。每条信息只能读取一次，对于某设定的故障等级，若本功能块的使用次数超过一次，下次的输出则是未读取的最早发生的故障信息。如果所有故障信息都已读取，或不存在，则通过输出变量 STATE 指出。

- STATE （故障的状态）

STATE 读取故障的当前状态。故障状态包括“发生”、“消失”及“已确认”。具体状态编号如下：

State	发生	消失	确认
16#02	**x**[1]	-[2]	-
16#04	-	**x**	-
16#06	**x**	**x**	
16#08	-	-	**x**
16#0A	**x**	-	**x**
16#0C	-	**x**	**x**
16#0E	**x**	**x**	**x**
16#F0	信息已被全部读取或无信息		

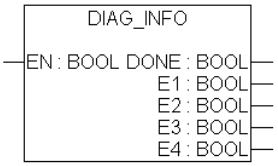
- ERROR （错误标识）

ERROR 输出错误标识，错误标识的含义请参见 10.1 章节。

- CODE （错误代码）

CODE 输出错误代码，错误代码的结构请参见 10.1 章节。

10.4 DIAG_INFO（显示未读取信息的故障等级）

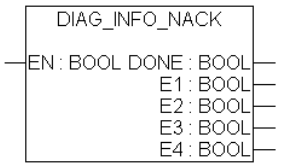


1. 功能：显示未读取信息的故障等级。
2. 类型：标准功能块（FB，含历史数据）。
3. 参数说明

输入参数	数据类型	描述
EN	BOOL	使能功能块　FALSE：无效 TRUE：高电平有效
输出参数	**数据类型**	**描述**
DONE	BOOL	功能块就绪信息
E1	BOOL	为 TRUE 时说明 E1 等级故障中存在未读取信息
E2	BOOL	为 TRUE 时说明 E2 等级故障中存在未读取信息
E3	BOOL	为 TRUE 时说明 E3 等级故障中存在未读取信息
E4	BOOL	为 TRUE 时说明 E4 等级故障中存在未读取信息

[1] X 表示有效
[2] - 表示无效

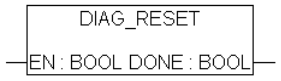
10.5 DIAG_INFO_NACK（显示未确认信息的故障等级）



1. 功能：显示未确认信息的故障等级。
2. 类型：标准功能块（FB，含历史数据）。
3. 参数说明

输入参数	数据类型	描述
EN	BOOL	使能功能块　FALSE：无效 TRUE：高电平有效
输出参数	**数据类型**	**描述**
DONE	BOOL	功能块就绪信息
E1	BOOL	为 TRUE 时说明 E1 等级故障中存在未确认信息
E2	BOOL	为 TRUE 时说明 E2 等级故障中存在未确认信息
E3	BOOL	为 TRUE 时说明 E3 等级故障中存在未确认信息
E4	BOOL	为 TRUE 时说明 E4 等级故障中存在未确认信息

10.6 DIAG_RESET（复位诊断系统）



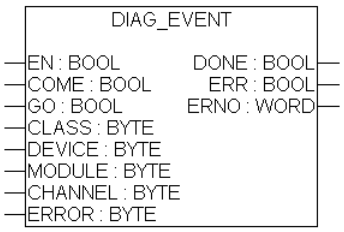
1. 功能：复位诊断系统。
2. 类型：标准功能块（FB，含历史数据）。
3. 参数说明

输入参数	数据类型	描述
EN	BOOL	使能功能块　FALSE：无效 TRUE：上升沿触发
输出参数	**数据类型**	**描述**
DONE	BOOL	功能块就绪信息

4. 描述

该功能块用于复位诊断系统，清除之前发生的全部故障记录。

10.7 DIAG_EVENT （生成故障信息）



1. 功能：产生一个可编程的故障信息。
2. 类型：标准功能块（FB，含历史数据）。

3. 参数说明

输入参数	数据类型	描述
EN	BOOL	使能功能块　FALSE：无效 TRUE：上升沿触发
COME	BOOL	设定故障状态，COME：TRUE 表示“故障已发生”。本输入可与输入 GO 连用
GO	BOOL	设定故障状态，COME：TRUE 表示“故障已消失”。本输入可与输入 COME 连用
CLASS	BYTE	设定故障等级，有效范围 1~4
DEVICE	BYTE	设定故障的设备编号，有效范围 0~255
MODULE	BYTE	设定故障的模块编号，有效范围 0~31
CHANNEL	BYTE	设定故障的通道，有效范围 0~31
ERROR	BYTE	设定故障的错误标识，有效范围 0~63
输出参数	**数据类型**	**描述**
DONE	BOOL	功能块完成标志，操作执行完毕为 TRUE，仅保持一个扫描周期
ERR	BOOL	功能块错误标志。该输出要与 DONE 一起综合判断。 DONE 为 TRUE，且 ERR 为 TRUE，则说明操作完成但存在错误；DONE 为 TRUE，且 ERR 为 FALSE，则说明操作成功
ERNO	WORD	功能块操作存在错误的代码，代码含义请参见附录 A。只有 DONE 为 TRUE，且 ERR 也为 TRUE 时，ERNO 的输出值才有效

4. 描述

- 故障状态

每个故障有三种状态：1：故障发生（COME）；2：故障消失（GO）；3：故障已确认。 使用 DIAG_EVENT 功能块，可生成状态为 1 和 2 的故障。确认故障需要使用功能块 DIAG_ACK 和 DIAG_ACK_ALL 或通过其他诊断方法来实现。

- ERROR （错误标识）

ERROR 输入端设定要产生的错误标识。为了使后期的查找方便，推荐使用已经规定的错误标识。有关错误标识的详细信息，请参见 10.1 章节。如果某条故障没有对应的故障标识，可自由定义。

附录

A. 功能块的错误代码

AC500功能块库中的各条功能块指令均提供一个输出参数“ENRO”，显示功能块操作存在的错误信息。代码遵循统一的编码规则，便于用户查找原因，排除错误。只有输出“DONE”为TRUE，且“ERR”也为TRUE时，ERNO的输出值才有效。

- 0000hex...0FFFhex - 报文错误
- 1000hex...1FFFhex - 设备错误
- 2000hex...2FFFhex - 接口错误
- 3000hex...3FFFhex - 协议错误
- 4000hex...4FFFhex - 输入错误
- 5000hex...5FFFhex - 请求错误
- 6000hex...6FFFhex - 通讯模块错误

i. 0000hex...0FFFhex - 报文错误

DEC	HEX	错误描述
0	0000	无错误
1	0001	COM_MOD_MAST：从机接收到的功能代码是不允许的（ILLEGAL FUNCTION）
		ETH_MOD_MAST：从机接收到的功能代码是不允许的（ILLEGAL FUNCTION）
2	0002	COM_MOD_MAST：查询中的数据地址对于从机来说是非法的（ILLEGAL DATA ADDRESS）
		ETH_MOD_MAST：查询中的数据地址对于从机来说是非法的（ILLEGAL DATA ADDRESS）
3	0003	COM_MOD_MAST：查询中的数据域的数值不是从机所允许的（ILLEGAL DATA VALUE）
		ETH_MOD_MAST：查询中的数据域的数值不是从机所允许的（ILLEGAL DATA VALUE）
4	0004	COM_MOD_MAST：当从站试图进行所要求的动作时，发生了不可恢复的错误（SLAVE DEVICE FAILURE）
		ETH_MOD_MAST：当从站试图进行所要求的动作时，发生了不可恢复的错误（SLAVE DEVICE FAILURE）
5	0005	COM_MOD_MAST：从站已接收查询正在处理，需要持续较长时间（ACKNOWLEDGE）
		ETH_MOD_MAST:从站已接收查询正在处理,需要持续较长时间(ACKNOWLEDGE)
6	0006	COM_MOD_MAST：从站正在处理，请主站稍候（SLAVE DEVICE BUSY）
		ETH_MOD_MAST：从站正在处理，请主站稍候（SLAVE DEVICE BUSY）

DEC	HEX	错误描述
8	0008	COM_MOD_MAST：从站在查询寄存器时发现奇偶校验错误（PARITY ERROR）
		ETH_MOD_MAST：从站在查询寄存器时发现奇偶校验错误（PARITY ERROR）
10	000A	COM_MOD_MAST：从站返回 GATEWAY PATH UNAVAILABLE 错误信息
		ETH_MOD_MAST：从站返回 GATEWAY PATH UNAVAILABLE 错误信息
11	000B	COM_MOD_MAST：从站返回 GATEWAY TARGET DEVICE FAILED TO RESPOND 错误信息
		ETH_MOD_MAST：从站返回 GATEWAY TARGET DEVICE FAILED TO RESPOND 错误信息

ii. 0000hex...0FFFhex – 设备错误

DEC	HEX	错误描述
4097	1001	设备不存在
4098	1002	设备：设备固件/硬件不支持此功能块；库版本高于设备固件版本
		FC..[3]：使用的模块不支持高速计数器功能
4100	1004	错误的操作模式
		FC..：PLC 配置中操作模式设定为“0”（不使用计数器）
4101	1005	无效状态
		FLASH_READ：数据块还未写
		FLASH_WRITE：数据块已写成功
		RETAIN..[4]：未加载
4117	1015	格式错误
		SD..[5]：格式无效，不能读取数据或不能完整读取数据。
4119	1017	长度不正确
		PERSISTENT..[6]：数据长度不正确
		RETAIN..[7]：数据长度不正确
4120	1018	校验和错误
4122	101A	FC..：内部错误（没有找到 CS31 地址 Adr 参数；或 CS31 地址 Adr 错误等）
		HA..[8]：内部错误（收到无效指针；功能内部的返回值错误；配置中没有找到对应项等）
4123	101B	设备访问错误
		Flash..：资源不可用
		HA..：备用 CPU 故障
		PERSISTENT..：不能复制数据，访问错误或数据不存在
		RETAIN..：不能复制数据，访问错误或数据不存在
		SD..：不能访问 SD 卡（内存耗尽、文件已经打开等）
4124	101C	数量不正确

[3]FC..为 FAST COUNTER 的缩写
[4]RETAIN..代表针对 RETAIN 数据的操作
[5]SD..代表针对 SD 卡地操作
[6]PERSISTENT..代表针对 PERSISTENT 数据的操作
[7]RETAIN..代表针对 RETAIN 数据的操作
[8]HA..为 HA 高可靠系统的缩写

DEC	HEX	错误描述
		PERSISTENT..：只加载了部分参数和数据。
4127	101F	访问保护
		SD..：SD 卡被写保护
4128	1020	打开时错误
		SD..：打开 SD 卡中存储的文件时错误
4129	1021	没有找到
		FC..：未找到 CS31 总线模块
		HA..：CS31 从站故障
		SD..：文件中找不到所查询的数据或文件
		TASK_INFO：未知的任务
4130	1022	到达终点
		SD..：到达结束部分或文件结尾
		PERSISTENT..：因为没有加载 CPU 参数和数据
4131	1023	读错误
		FLASH..：数据段读错误；校验和不正确
4132	1024	写错误
		FLASH..：数据块不能编辑
		SD..：文件不能删除或覆盖
4137	1029	FC..：配置错误
		HA..：远程 CS31 从站故障
8191	1FFF	未就绪
		Flash..：命令已经被其他实例执行
		SD..：其他实例已被激活，该命令不能被执行

iii. 2000hex...2FFFhex - 接口错误

DEC	HEX	错误描述
8193	2001	无效的接口，通讯模块编号或插槽号（Slot）错误
		COM..[9]：接口没有配置为“自由模式”
8194	2002	接口不支持的命令。设备固件不支持此功能块；库版本高于设备固件版本
8195	2003	无效的接口或通讯模块类型；功能块不适用于此类型
8197	2005	HA..：CS31 总线故障
8211	2013	超时
		COM_MOD_MAST：在设定的时间内从站没有响应
		HA..：无以太网连接
8212	2014	帧错误（不正确的波特率、停止位数量或每个字符的位）
8213	2015	奇偶错误
8214	2016	空闲错误
		COM..：发生字符超时
8215	2017	无效的长度
		COM_MOD_MAST：接收到无效的数据长度

[9]COM..代表针对串行接口的操作

DEC	HEX	错误描述
		COM_REC：接收的数据超过期望的数据
8216	2018	校验和错误
8217	2019	同步交换错误
8218	201A	服务故障
		COM_REC：接口未知的错误信息
		COM_SET_PROT：不能访问硬件接口，系统启动期间初始化失败
		COM_MOD_SLV_SET_ADDR：内部错误（收到无效指针；功能内部的返回值错误）
		CS31：内部错误（例如 CS31 通讯失败）
		FC..：内部错误（收到无效指针；功能内部的返回值错误；配置中没有找到对应项等）
		HA..：内部错误（收到无效指针；功能内部的返回值错误；配置中没有找到对应项等）
8219	201B	访问错误
		HA..：备用 CPU CS31 总线故障
		IO..：模块号码不存在
8220	201C	不正确的号码（数量）
		COM_SET_PROT：无效的协议索引；设备不支持这个索引
		COM_MOD_SLV_SET_ADDR..：无效的协议索引
		HA..：总线 A 和总线 B 中 CS31 配置的模块数量不同
		IO..：无效的模块号码
8223	201F	拒绝访问
		COM..：当前不能访问该接口（CBP、SYCON.net、OPC 或其他程序已占用接口）
8226	2022	COM_MOD_SLV_SET_ADDR..：错误的配置（未配置任何协议）
		FC..：错误的配置
		HA..：同步表已满或数据不同
8299	2029	HA..：错误的配置（CS31 未配置；没有发现 CS31 模块，CS31 总线 A 和总线 B 的切换配置不正确）

iv. 3000hex...3FFFhex – 协议错误

DEC	HEX	错误描述
12289	3001	未知的协议
12290	3002	协议不支持该命令。设备固件不支持此功能块；库版本高于设备固件版本
12291	3003	协议配置错误
		FC_DC551：选择的接口（COMx）未设置为 CS31 协议
		COM_MOD_SLV_SET_ADDR：选择的接口（COMx）未设置为多协议
		COM_MOD_SLV_SET_ADDR：错误的协议索引（设定的索引中没有 Modbus 协议）
12292	3004	操作模式错误
		COM_MOD_MAST：无效的操作模式（主/从）
12293	3005	协议状态错误
		现场总线通讯模块 .._SYS_DIAG：主站不在 OPERATE 状态
12307	3013	IEC..[10]：ACTCON 超时

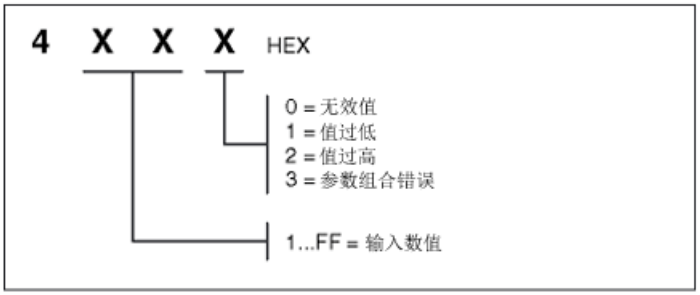
[10]IEC.. .代表有关 IEC60870-5-104 协议通讯的操作

DEC	HEX	错误描述
12308	3014	IEC..：接收到 NACK
12310	3016	IEC..：超时
12311	3017	长度不正确
		ARC..[11]：缓冲区满
		CAN2..[12]：信息长度超限
		IEC..：队列超限
		ETH_UDP..[13]：缓冲区满
12313	3019	IEC..：ACTERM 超时
12314	301A	执行失败
		COM_SET_PROT：协议初始化失败
		IEC..：由于队列已删除发送失败
		COM_MOD_SLV_SET_ADDR：内部错误（收到无效指针；功能内部的返回值错误）
12315	301B	访问错误
		ARC..：缓冲区不存在或没有设定
		CAN2..：缓冲区不存在或没有设定
		ETH_UDP..：缓冲区不存在或没有设定
		IEC..：协议访问错误
		IO..：选择的槽中未发现模块
12316	301C	错误号码
		IO..：无效的模块号码，超出范围
12319	301F	拒绝访问
		COM..：当前不能访问该接口（CBP、SYCON.net、OPC 或其他程序已占用接口）
12320	3020	打开时错误
		ARC..：协议初始化时错误；议未就绪
		CAN2..：协议初始化时错误；协议未就绪
		IEC..：数据到达时数据类型与该 PIN 不匹配
		ETH_UDP..：协议初始化时错误；协议未就绪
12325	3025	地址错误
		COM_MOD_MAST：接收报文中的寄存器地址错误
12326	3026	功能错误
		COM_MOD_MAST：接收的 FCT 与发送的 FCT 不对应
12327	3027	无效的值
		COM_MOD_MAST：接收的报文包含意外信息
12329	3029	HA..：CS31 从站配置不完整
12333	302D	没有连接。连接已关闭或尚未建立
16383	3FFF	未就绪。当前资源不可用
		COM_MOD_MAST：当前不能传输。另一个功能块实例正在传输
		COM_SEND：当前不能传输。另一个功能块的实例正在传输

[11]ARC.. 代表有关 ARCNET 协议通讯的操作
[12]CAN2.. 代表有关 CAN2.0 协议通讯的操作
[13]ETH_UDP.. 代表有关 Ethernet UDP 通讯的操作

v. 4000hex...4FFFhex - 功能块输入错误

当检测到功能块输入参数错误时，出现错误 4xxxhex。 这个错误的结构如下：



vi. 5000hex...5FFFhex - 请求错误

DEC	HEX	错误描述
20482	5002	不支持的请求（CBP 处于仿真模式下）
20503	5017	长度不正确
		ARC..：接收到无效的数据长度
		CAN2..：信息中无效的 DLC
		ETH_UDP..：无效的数据长度
20507	501B	访问错误
		COM_MOD_MAST：无效的内存地址 DATA 或 DATA+NB（至少一个数据超过了用户程序的访问范围）
		ETH_MOD_MAST：无效的内存地址 DATA 或 DATA+NB（至少一个数据超过了用户程序的访问范围）
20508	501C	无效编号（数量）
		CAN2A_SEND：输入的 NUM 为无效的信息号码
		COM_MOD_MAST：NB（数据个数）无效（0 或超过允许值）
		ETH_MOD_MAST：NB（数据个数）无效（0 或超过允许值）
20517	5025	地址错误
		ARC..：无效的 IP 地址
		CAN2..：报文中含有无效的标识符
		COM_MOD_MAST：无效的从站地址；选择的功能码不支持广播
		ETH_MOD_MAST：无效的从站地址；选择的功能码不支持广播
		ETH_UDP..：无效的 IP 地址
20518	5026	功能错误
		COM_MOD_MAST：无效的功能代码 FCT
		ETH_MOD_MAST：无效的功能代码 FCT
20735	50FF	DIAG..：仿真模式

vii. 6000hex...6FFFhex - 通讯模块错误

DEC	HEX	错误描述
24577	6001	EtherCAT®..[14]：收到无效的命令
		PNIO..[15]：由于内存不足，在 RPC 任务中创建 TLR 定时数据包失败
24578	6002	EtherCAT®..：连接不存在或看门狗超时
		CAN..：SDO 终止，节点拒绝服务；索引/子索引无效或无权访问指定的节
		DNM..[16]：资源不可用或无效的类别 ID
		DPM../DPV1..[17]：资源不可用（对于请求的服务，从站中没有足够的缓冲内存）
		PNIO：一般 RPC 错误代码或没有足够的内存
24579	6003	EtherCAT®..：在读取从站配置期间错误或请求的看门狗时间过短
		DPM../DPV1..：从站中没有激活所请求的服务（例如 DPV1）
24580	6004	EtherCAT®..：在处理总线配置期间发生错误，或请求的看门狗时间过长
24581	6005	EtherCAT®..：现有总线与配置的总线不匹配或复位时错误（看门狗复位）
24582	6006	EtherCAT®..：不是所有的从站都可用或复位期间错误（清除动态资源）
24583	6007	EtherCAT®..：复位时错误（停止主站）或主站在临界错误状态，必须复位
24584	6008	EtherCAT®..：复位时错误（初始化主站）或激活看门狗错误
		DNM..：模块中的服务不可用。选择的类别不支持读/写功能
24585	6009	EtherCAT®..：复位时错误（清除动态资源）或配置的输入大小大于循环 DPM 输入的大小
		DNM..：无效或不支持的属性
		DPM../DPV1..：未接收到从站数据
24586	600A	EtherCAT®..：主站在临界错误状态。必须复位或配置的输出数据大小大于 DPM 数据输出的大小
24587	600B	EtherCAT®..：请求的总线循环时间无效
		DNM..：请求已经在进行中
24588	600C	EtherCAT®..：从站离线，参数无效
		DNM..：对象状态冲突
24589	600D	EtherCAT®..：主站状态为内部错误
24590	600E	EtherCAT®..：看门狗超时
		DNM..：不能设置属性或不允许写
24591	600F	EtherCAT®..：CoE 使用了无效的从站 ID
		DNM..：允许检查失败或拒绝访问
24592	6010	EtherCAT®..：对于 CoE 传输没有可用的资源
		DNM..：状态冲突，禁止执行
24593	6011	EtherCAT®..：使用 CoE 时内部错误
		CAN..：选择的节点没有响应
		DNM..：选择的设备没有响应
		DPM0/DPV1..：从站没有响应。

[14]EtherCAT®..代表有关 EtherCAT 通讯的操作
[15]PNIO..代表有关 PROFINET IO 通讯的操作
[16]DNM.. 代表有关 DeviceNet 通讯的操作
[17]DPM../DPV1.. 代表有关 PROFIBUS DP 通讯的操作

DEC	HEX	错误描述
24594	6012	EtherCAT®..：请求从站的索引无效
		DPM../DPV1..：DP 主站不在逻辑令牌中
		PNIO..：对于这次请求，内存不足
24595	6013	EtherCAT®..：使用 CoE 时无效的总线通讯状态
		CAN.. ：选择的节点没有准备好
		DNM..：接收的数据不足
24596	6014	EtherCAT®..：CoE 数据帧丢失
		CAN..：本地的资源不可用。请求的总线参数不可用；没有配置通讯模块
		DNM..：本地的资源不可用。 请求的总线参数不可用；没有配置通讯模块
		PNIO..：在当前的 CMCTL 状态下，不能服务该请求
24597	6015	EtherCAT®..：CoE 服务时超时
		CAN..：参数错误
		DNM..：参数错误
24598	6016	EtherCAT®..：未找到从站（不在总线上或电源关闭）
		DNM..：对象不存在
24599	6017	EtherCAT®..：请求的列表类型无效
		其他通讯模块：接收的数据长度过大；内部缓冲区太小
24600	6018	EtherCAT®..：从站中的数据响应对于确认包来说太大
		PNIO..：发送数据包给其他任务时错误
24601	6019	EtherCAT®..：选择了无效的访问任务（在 GetEntryDesc 期间）
		DPM../DPV1..：意外的从站行为或行为不满足标准
24602	601A	EtherCAT®..：CoE 服务期间，从站的工作计数器错误
24603	601B	EtherCAT®..：服务已经在使用
24604	601C	EtherCAT®..：该通讯状态下，通讯不可用
24605	601D	EtherCAT®..：必须为该命令激活分布式时钟
24606	601E	EtherCAT®..：扫描已经在运行，同时不能启动两次
24607	601F	EtherCAT®..：总线扫描超时，但至少建立了一个连接
24608	6020	EtherCAT®..：之前没有启动总线扫描或还没有结束
24609	6021	EtherCAT®..：请求的从站无效
24610	6022	EtherCAT®..：使用 CoE 时内部错误
24612	6024	PNIO..：请求的信息 ID 不正确，超出顺序
24624	6030	CAN..：功能超时
		DNM..：没有配置设备
		PNIO..：对该设备进行过多非标准 RPC 请求或请求数据包中含有无效的报警优先级
24625	6031	PNIO..：发送内部消息给其他任务时错误
24626	6032	DNM..：接收的数据格式错误
		ETH_UDP..：TCP/UDP 任务不可用或 IP 任务没有就绪
		PNIO..：用于 IO 设备的 ulHandle 句柄错误
24627	6033	CAN..：接收的数据超过了最大的缓冲区
		ETH_UDP..：内部任务配置数据不可用
		PNIO..：对应于请求数据包中索引的 ALPMR 协议无效

DEC	HEX	错误描述
24628	6034	CAN..：功能不可用；未知代码
		DNM..：未知代码
		ETH_MOD..：服务连接（ServerConnection）的参数无效
		ETH_UDP..：没有可用的 MAC 地址
		PNIO..：对于当前的请求 ALPMR 协议无效
24629	6035	CAN..：未知区域；超过缓冲区
		DNM..：缓冲区长度溢出
		ETH_MOD..：任务超时（Task Timeout）的参数无效
		ETH_UDP..：等待应用程序进行热启动
24630	6036	CAN..：在 HOST 信息中有未知的功能或功能仍处于激活状态
		DNM..：有其他激活的服务
		ETH_MOD..：OBM 超时（OBM Timeout）的参数无效
		ETH_UDP..：启动参数中有未知的标志
		DPM../DPV1..：从站拒绝访问请求的数据
24631	6037	CAN..：参数错误
		DNM..：参数错误或 MAC ID 超过有效的范围
		ETH_MOD..：模式（mode）的参数无效
		ETH_UDP..：启动参数中有无效的 IP 地址
		PNIO..：ALPMR 协议机的索引无效
24632	6038	ETH_MOD..：发送超时（Send Timeout）的参数无效
		ETH_UDP..：启动参数中有无效的子网掩码
24633	6039	CAN..：顺序错误
		DNM..：顺序错误或一个 MAC ID 在网络中使用了多次
		ETH_MOD..：连接超时（Connect Timeout）的参数无效
		ETH_UDP..：启动参数中有无效的网关 IP
24634	603A	ETH_MOD..：关闭超时（Close Timeout）的参数无效
24635	603B	CAN..：数据错误
		DNM..：数据错误
		ETH_MOD..：Swab 参数无效
		ETH_UDP..：未知的设备类型
24636	603C	CAN..：节点地址配置重复
		DNM..：数据集的总数量显示不正确
		ETH_MOD..：TCP 任务没有就绪
		ETH_UDP..：访问设定的源 IP 地址失败
24637	603D	CAN..：添加表不正确
		DNM..：添加表不正确
		ETH_MOD..：PLC 任务没有就绪
		ETH_UDP..：初始化驱动层失败
24638	603E	CAN..：节点参数的总长度不正确
		DNM..：I/O 配置表的大小不正确
		ETH_MOD..：初始化期间错误
		ETH_UDP..：没有 IP 地址设置源（BOOTP, DHCP, IP 地址参数）

DEC	HEX	错误描述
24639	603F	CAN..：未知的传输类型
		DNM..：I/O 配置与 ADD 表不匹配
24640	6040	CAN..：PDO-cfg 文件的长度太大
		DNM..：参数大小不正确或通道/句柄已经在使用
		PNIO：索引的 CMCTL 协议机无效
24641	6041	CAN..：未知的波特率
		DNM..：在 ADD 表中定义的输入的数量与 I/O 配置不匹配
		PNIO..：索引对应的 CMTCL 协议机无效
24642	6042	CAN..：COB-ID SYNC 超过了有效的范围
		DNM..：在 ADD 表中定义输出的数量与 I/O 不匹配
24643	6043	CAN..：同步定时器值无效
		DNM..：I/O 配置中的数据类型未知
24644	6044	CAN..：PDO 的输入偏移太大
		DNM..：定义的 I/O 模块的数据类型与定义的数据大小不匹配
24645	6045	CAN..：PDO 的输出偏移太大
		DNM..：I/O 模块配置的输出地址超出范围，3584 个字节
24646	6046	CAN..：PDO 和 ADD 表矛盾
		DNM..：I/O 模块配置的输入地址超出范围，3584 个字节
24647	6047	CAN..：ADD 表的长度矛盾
		DNM..：未知的连接类型
24648	6048	CAN..：总长度矛盾
		DNM..：定义了多个相同的连接
24649	6049	CAN..：紧急消息 COB-ID 超过了有效的范围
		DNM..：连接 Exp_Packet_Rate 配置的值小于 Prod_Inhibit_Time 的值
24650	604A	CAN..：COM-ID 节点保护超过了有效的范围
		DNM..：参数域 DNM_PRED_MSTSL_CFG_DATA 矛盾
24651	604B	CAN..：配置的 PDO 长度大于 8
		DNM..：设备不能执行 MAC-ID 重复检查（Duplicate_MAC-ID check）。 波特率不正确或不能建立与设备的连接
24652	604C	CAN..：SDO 数据中定义的对象数量太大
		DNM..：usRecFragTimer 的值超过了允许范围
24657	6051	PNIO..：当前的总线状态是 OFF，不能发送帧
24659	6053	PNIO..：当前请求 ALPMR 协议机的状态无效
24660	6054	PNIO..：APMS 不能得到 Edd_Frame 缓冲区，发送失败
24661	6055	PNIO..：当 APMS 尝试发送一个 Edd_Frame 时发生错误
24662	6056	PNIO..：不能到达设备（没有在工程中设置 DEV_NAME）
24663	6057	PNIO..：没有足够的空间提供 APMS_send_req_Date（分配定时指示数据包）
24686	606E	ETH_UDP..：超时
24687	606F	ETH_MOD..：未知的发送或接收报文
		ETH_UDP..：无效超时参数
24688	6070	ETH_MOD..：TCP 响应错误
		ETH_UDP..：无效的 socket

DEC	HEX	错误描述
24689	6071	ETH_MOD..：没有找到对应的 socket
		ETH_UDP..：当前 socket 不能执行命令
24690	6072	ETH_MOD..：命令值无效
24691	6073	ETH_MOD..：TCP 任务状态错误
		ETH_UDP..：不能访问目标 IP 地址
24692	6074	ETH_UDP..：无效的参数
24693	6075	ETH_MOD..：没有找到空闲的 socket
		ETH_UDP..：无效的命令参数
24694	6076	ETH_MOD..：未知的 socket
		ETH_UDP..：无效的 IP 地址或不能访问地址
		HA..：IP 地址配置错误
24695	6077	ETH_MOD..：客户端工作的时间过期
		ETH_UDP..：无效的端口号或端口号不可用
24696	6078	ETH_MOD..：socket 意外被关闭
		ETH_UDP..：连接关闭
24697	6079	ETH_MOD..：用户设置了未就绪标志
		ETH_UDP..：连接复位
24698	607A	ETH_MOD..：OMB 任务不能打开 socket
		ETH_UDP..：无效的协议
24699	607B	ETH_MOD..：PLC 任务中的看门狗事件，只在 I/O 模式中
		ETH_UDP..：无空闲 socket
24700	607C	ETH_MOD..：TCP 任务处于配置状态
		ETH_UDP..：无效的 MAC 地址
24701	607D	ETH_MOD..：没有初始化 PLC 任务
24702	607E	ETH_MOD..：服务器 socket 关闭，设备没有响应
24705	6081	DPM../DPV1..：DPV1 未在“打开”（OPEN）状态
24706	6082	ETH_UDP..：无效的模式参数
		DPM../DPV1..：收到从站无效的参数
24707	6083	ETH_UDP..：数据长度过长或 ARP 缓存满
		DPM../DPV1..：服务仍处于激活状态，不能进行并行操作
24708	6084	ETH_UDP..：超过了信息的最大数量
		DPM../DPV1..：接收缓冲区的数据长度过长
24709	6085	ETH_UDP..：超过了多路 IP 的最大数量
		DPM../DPV1..：错误的参数
24710	6086	ETH_UDP..：在 ARP 缓存中没有找到 ARP 输入
24725	6095	ETH_UDP..：收到无效的信息响应
24727	6097	ETH_MOD..：无效的信息长度
		ETH_UDP..：无效的信息长度
24728	6098	CAN..：未知的信息命令
		DNM..：未知的信息命令
		ETH_MOD..：未知的信息命令
		ETH_UDP..：未知的信息命令

DEC	HEX	错误描述
24730	609A	DPM../DPV1..：无效的信息命令
24732	609C	ETH_UDP..：在顺序信息模式下传输顺序错误
24734	609E	ETH_UDP..：不能执行命令
24736	60A0	ETH_MOD..：报头错误
24737	60A1	CAN..：节点地址超过了允许的范围
		DNM..：设备地址超过了允许的范围
		ETH_MOD..：在报文中检测到无效地址
		DPM../DPV1..：无效的从站地址
24738	60A2	CAN..：无效地址范围
		DNM..：无效地址范围
24739	60A3	CAN..：数据缓冲区溢出
		DNM..：数据缓冲区溢出
		ETH_MOD..：无效的数据地址
24741	60A5	CAN..：数据计数器不正确
		DNM..：数据计数器不正确
		ETH_MOD..：无效的数据计数器
24742	60A6	CAN..：未知的数据类型
		DNM..：未知的数据类型
24743	60A7	CAN..：未知的功能
		DNM..：未知的功能
		ETH_MOD..：在 TCP 任务响应中 OBM 任务接收错误
24776	60C8	CAN..：没有配置通讯模块
		DNM..：没有配置通讯模块
		ETH_UDP..：任务没有初始化
24778	60CA	ETH_MOD..：OBM 任务没有来自 RCS 的段
24779	60CB	ETH_MOD..：命令设置了未知或无效的发送者
24786	60D2	ETH_UDP..：无可用的配置数据
24788	60D4	ETH_UDP..：当读取配置数据时错误
24789	60D5	ETH_UDP..：当创建诊断结构时错误
24794	60DA	ETH_UDP..：没有足够的内存可用
24832	6100	PNIO..：一般 RPC 错误
25088	6200	PNIO..：发送内部消息给其他任务时错误
25089	6201	PNIO..：由于内存不足，在 RPC 任务中创建 TLR 定时数据包失败
26117	6605	PNIO..：RPC 客户端实例的句柄无效
26118	6606	PNIO..：未完成的 RPC 客户端请求达到最大数量
26119	6607	PNIO..：没有未完成的 RPC 请求，RPC 客户端示例才能连接到 I/O 设备

缩写	
RPC	远程过程调用
CMCTL	控制器内存管理
APMS	非循环协议机发送者
APMR	非循环协议机接收者

B. 操作数

操作数是指令执行的参与者，也就是各种操作的对象。AC500 系列 PLC 中常量、变量、PLC 直接地址和功能的返回值均属于操作数。

i.常量

在 PLC 编程的时候，会用到一些数值不变的参数，诸如定时器的时间、换算的比例等，这些数值不变的参数称为常量。CoDeSys 支持多种数据类型的常量，常见的常量包括布尔型、时间型、数字型等，如下表。

常量类型	使用方法	
布尔型	描述	布尔常量只有两个：TRUE 和 FALSE（或表示为 1 和 0），TRUE 等价于 1，FALSE 等价于 0
	示例	TRUE、0
整数型	描述	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
	示例	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*）
实数型	描述	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
	示例	7.4 （*实数 7.4*） 1.64e+009 （*实数 1.64e+009*）
时间型	描述	时间常量一般用来操作时间，由“T#”（或“t#”）加上“时间值”构成，时间值的单位包括天（d）、小时（h）、分（m）、秒（s）和毫秒（ms）。注意，它们的正确顺序为 d、h、m、s、ms
	示例	T#18ms （*18 毫秒的一个时间常量*） T#100s12ms （*100 秒 12 毫秒的一个时间常量，高单位允许超限*） t#12h34m15s （*12 小时 34 分 15 秒的一个常量*） 下面是错误的时间常量： t#5m68s （*低单位不允许超限*） 15ms （*没有 T# *） t#4ms13d （*顺序错误*）
时刻型	描述	时刻常量用于存储当前时刻，由“TOD#”（“tod#”、“TIME_OF_DAY#”或“time_of_day#”）加上“时刻值”构成。时刻值的格式为：小时:分钟:秒（可以用实数形式输入秒）
	示例	TOD#00:00:00 （*时刻常量为 0 点 0 时 0 分*） TIME_OF_DAY#15:36:30.123 （*时刻常量为 15 点 36 分 30.123 秒*）
日期型	描述	日期常量由“D#”（“d#”、“DATE#”或“date#”）加上“日期值”构成
	示例	DATE#2005-05-06 （*日期常量 2005 年 5 月 6 日*） d#1980-09-22 （*日期常量 1980 年 9 月 22 日*）
日期时刻型	描述	日期常量和时刻常量合并起来称为日期时刻常量，由“DT#”（“dt#”、“DATE_AND_TIME#”或“date_and_time#”）加上“日期时刻值”构成
	示例	DT#1980-09-22-15:45:18 （*时刻日期常量为 1980 年 9 月 22 日 15 点 45 分 18 秒*） date_and_time#2001-03-09-00:00:00 （*时刻日期常量为 2001 年 3 月 9 日 0 点 0 分 0 秒*）

常量类型	使用方法	
字符串型	描述	字符串常量在两个单引号之间，可以包含空格和特殊字符
	示例	'Abby and Craig' （*字符串 Abby and Craig *） ':-)' （*字符串:-)*）

CoDeSys 中字母不区分大小写，诸如 T#18ms 和 t#18ms 为同一常数。

ii.变量

所谓变量，即变化的数据，是用字母、数字和下划线组成的一个标识符。在 POU 的声明部分定义变量（也可以在全局列变量表中定义全局变量）。

按照数据类型的不同，变量可以分为标准类型和用户自定义类型。其中标准类型包括布尔型（BOOL）、整型（INT）、实型（REAL）、字符串型（STRING）以及时间型（TIME）等。自定义类型包括结构体（STRUCT）和枚举（ENUM）。

按照使用范围的不同，变量可以分为全局变量和局部变量。局部变量只在对其进行声明的程序中有效，其它程序不能引用。全局变量则可以被整个工程的任意程序引用，在整个工程中均有效。

按照类别的不同，变量分为中间变量、输入型变量、输出型变量、输入输出型变量等。

按照能否掉电保护，变量分为保持型变量和非保持型变量。

1. 变量命名规则

- 必须以一个字母或者单一的下划线开始，随后是一定数量的字母、数字或下划线。。
- 字母与大小写无关，ABC 和 abc 被认为是同一个变量。
- 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）
- 关键字不能用于变量名。关键字是标准的标识符，其作用和命名已在系统中自动声明。

2. 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>

3. 变量的索引

- 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、LINT、ULINT、BYTE、WORD、DWORD 和 LWORD。
- 定义在 VAR_IN_OUT 数据区的变量，不能访问变量的数据位。

iii.直接地址

PLC 的数据存储区分为输入区（I 区）、输出区（Q 区）、内部寄存器区（M 区）和可寻址掉电保持区（R 区）。通过特殊字符的排列直接显示各个存储器的位置称为 PLC 直接地址。该排列包括 “%”、范围前缀、大小前缀等。

1. 寻址方式

AC500 PLC 直接地址操作数采用 Motorola（摩托罗拉）字节顺序存储。

以输入区（%I 区）为例，说明位（BOOL），字节（BYTE），字（WORD）和双字（DWORD）的关系：

地址	Addr	Addr + 1	Addr +2	Addr +3
BYTE	%IB0	%IB1	%IB2	%IB3
BOOL	7 ... 0	7 ... 0	7 ... 0	7 ... 0
	%IX0.7...%IX0.0	%IX1.7...%IX1.0	%IX2.7...%IX2.0	%IX3.7...%IX3.0
WORD	%IW0		%IW1	
	15 ... 8	7 ... 0	15 ... 8	7 ... 0
DWORD	%ID0			
	31 ... 24	23 ... 16	15 ... 8	7 ... 0

例如某数据的对应关系如下：

%IX0.0	:=TRUE		
	%IB0:=1	:=16#01	
	%IW0:=256	:=16#0100	（Bit 8 =TRUE）
	%ID0:=16777216	:=16#01000000	（Bit 24 =TRUE）
%IX3.0	:=TRUE		
	%IB3:=1	:=16#01	
	%IW1:=1	:=16#0001	
	%ID0:=1	:=16#00000001	

2. 寻址格式

按 IEC61131-3 标准，所有的直接地址都从“%”开始。以 M 区为例，不同数据的寻址格式如下表所示。

位寻址	格式	%MXl.m.n
	描述	X 表示是按位寻址 l 表示在%M 存储区中的段号，（%I 和%Q 区不含此项） m 表示在%M 存储区某段中的字节编号 n 表示位于该字节的第几位，范围为 0~7
	数据类型	BOOL
	示例	%MX0.1.3、%MX0.6.5
字节寻址	格式	%MBl.m
	描述	B 表示按字节寻址 l 表示在%M 存储区中的段号，（%I 和%Q 区不含此项）； m 表示在%M 存储区某段中的字节编号；
	数据类型	BYTE
	示例	%MB0.103、%MB0.2000
字寻址	格式	%MWl.m
	描述	W 表示按字寻址 l 表示在%M 存储区中的段号，（%I 和%Q 区不含此项） m 表示在%M 存储区某段中的字编号
	数据类型	WORD
	示例	%MW0.150
双字寻址	格式	%MDl.m
	描述	D 表示按双字寻址 l 表示在%M 存储区中的段号，（%I 和%Q 区不含此项） m 表示在%M 存储区某段中的双字编号
	数据类型	DWORD
	示例	%MD0.10

C. 数据类型

i.标准类型

AC500 PLC 支持标准类型和用户自定义类型。标准数据类型及范围，如下表所示。

类型	类型名称	数据下限	数据上限	数据宽度	备注
BOOL	布尔型	0	1	1bit	
BYTE	字节型	0	255	8 Bit	
WORD	字型	0	65535	16 Bit	
DWORD	双字型	0	4294967295	32Bit	
LWORD	长字型	0	18446744073709551615	64Bit	
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	
REAL	实数型	-3.402823E+38	3.402823E+38	32Bit	
LREAL	长实数型	2.2250738585072014e-308	1.7976931348623158e+308	64Bit	
TIME	时间型			32Bit	示例：Time1:TIME:=t#3s;
TOD	时刻型				示例：Tod1:TOD:=TOD#00:00:00;
DATE	日期型				示例：Date1:DATE:=D#2008-8-8;
DT	日期时刻型				示例：DT1:DT:=dt#2008-08-08-20:08:08;
STRING	字符串型				示例：Str:STRING(35):='hi';
ARRAY	数组				示例：Arr1:ARRAY[1..5]OF BYTE:=1,2,3,4,5;

CodeSys 仿真模式下不支持 LWORD 类型

ii.数组

数组（ARRAY）是一组具有相同类型的数据。使用数组，可便于用户的定义及引用。数组不但支持多种数据类型，还支持多维数组，最多支持 9 维数组嵌套。

编程时，可以根据基本数据类型来定义一维、二维和三维数组。数组可以采用自动定义，也可以在变量声明区手动定义。

1. 定义数组

定义一维、二维和三维数组的格式如下：

<数组名>:ARRAY [<Ll1>..<Ul1>,<Ll2>..<Ul2>] OF <数据类型>

- 数组名
- 数组关键字“ARRAY”
- 数组维数和数组元素多少。例如二维数组为[L1..U1,L2..U2]，其中 L 为每一维首元素的编号，U 为每一维末元素的编号，L 和 U 必须是正整数，每维之间用逗号“,”隔开。
- 数组元素的数据类型“OF 数据类型”，例如“OF INT ”、“OF REAL”。
- 数组元素初值，每个值之间用逗号“,”隔开。

例如：Array_V1:ARRAY[1..5] OF INT;（*一维数组*）

Array_V21:ARRAY[1..3,1..3] OF INT:=111,0,0,210,222,0,310,0,333;（*二维数组*）

2. 初始化数组元素

定义数组时，可选择给数组中的元素赋初始值。赋初始值有以下两种方式：

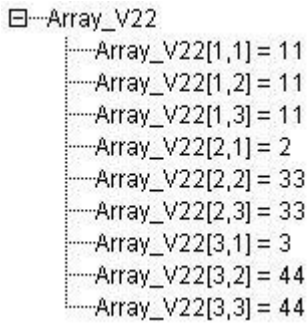
- 逐个写出每个元素的初始值，例如

Array_V21:ARRAY[1..3,1..3] OF INT:=111,0,0,210,222,0,310,0,333;（*二维数组*）

- 用缩写形式写出每个元素的初始值，例如

Array_V22:ARRAY[1..3,1..3] OF INT:=3(11),2,2(33),3,2(44);（*二维数组*）

其中 3(11)是“11,11,11”的缩写形式，2,2(33)是“2,33,33”的缩写形式；3,2（44)是“3,44,44”的缩写形式。3(11)表示圆括号外 3 为重复次数，圆括号内 11 为重复数值，其他类似。下图列出程序运行时上述数组 Array_V22 元素的初值。



3. 引用数组元素

程序中可引用数组元素，格式如下：

〈数组名〉[Index1,Index2]

例如

引用一维数组元素 Array_V1[2]

引用二维数组元素 Array_V21[1,2]

iii.结构体

结构体（STRUCT）可以将不同类型的数据组合成一个有机的整体，以便于引用。

结构体在 CoDeSys 中属于用户自定义变量类型。

1. 定义结构体的格式

结构体由多个成员（或分量）组成，每个成员有成员名及类型，类型可不同或相同。

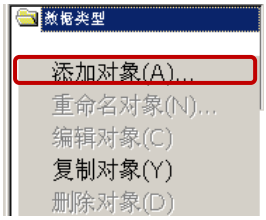
例如，下例中定义结构体“STRUCT_V1”，内含 3 个成员（或分量），分别为 V1:INT、V2:UINT、V3:REAL。如图：

```
STRUCT_V1
0001 TYPE STRUCT_V1 :
0002 STRUCT
0003     V1:INT;(*整型*)
0004     V2:UINT;(*无符号整型*)
0005     V3:REAL;(*实型*)
0006 END_STRUCT
0007 END_TYPE
```

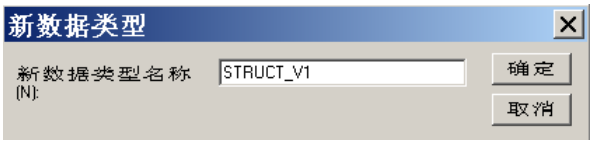
2. 定义结构体的方法

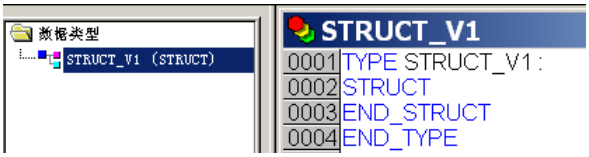
在编程界面“对象组织器”的“数据类型”下添加结构体，例如添加结构体“STRUCT_V1”，主要步骤如下：

右击“对象组织器”的“数据类型”，弹出对话框，如图所示。



鼠标左键单击“添加对象（A）”，弹出“新数据类型”窗口。在“新建数据类型名称（N）”框内填写结构体名“STRUCT_V1”，鼠标左键单击“确认”。添加一个结构体“STRUCT_V1”，如图所示。





编写结构体的成员（或分量）名及类型，分别为 V1:INT、V2:UINT、V3:REAL，如图所示。

```
STRUCT_V1
0001 TYPE STRUCT_V1 :
0002 STRUCT
0003     V1:INT;(*整型*)
0004     V2:UINT;(*无符号整型*)
0005     V3:REAL;(*实型*)
0006 END_STRUCT
0007 END_TYPE
```

3. 引用结构体的方法

工程编译后才可引用结构体。

首先应定义结构体的实例，然后程序中可引用结构体的成员（或分量）。如下图，其中“ST_V1”为结构体 STRUCT_V1 的“实例名”。

```
PLC_PRG (PRG-ST)
0001 PROGRAM PLC_PRG
0002 VAR
0003     ST_V1:STRUCT_V1;(*定义结构体实例*)
0004     RV1:REAL;
0005     END_VAR

0001 RV1:=ST_V1.V3+7;(*引用结构变量*)
```

结构体进行实例定义后，可通过“实例名.成员名称”调用结构体中的成员。例如，上例中“ST_V1”为结构体 STRUCT_V1 的“实例名”，“V1、V2、V3”为结构体 STRUCT_V1 的“成员名称”。

iv.指针

指针变量是一类特殊的变量。AC500 系列 PLC 中，所有存储区都占用 CPU 的地址，这种地址被称为绝对地址。%I 区、%Q 区、%M 区以及%R 区可以认为是相对地址。指针是指数据区的绝对地址而不是相对地址。

1. 指针的结构

作为特殊的变量，在使用过程中需明确指针的如下四方面内容：指针的类型，指针所指向的类型，指针的地址，及指针的内容。指针定义的语法格式为：

<指针名> :POINTER TO <数据类型/功能块>;

- 指针的类型

指针本身是一种特殊的变量，具有自己的数据类型。在指针声明语句中，指针名后面的部分就是该指针变量的类型。

```
piIndex:POINTER TO INT;          (*指针的类型是 POINTER TO INT，即整形变量指针）

pt:POINTER TO ARRAY [0..7] OF BYTE;  (*指针的类型是 POINTER TO ARRAY [0..7] OF BYTE，
                                       即 8 个元素的字节型数组的指针*）
```

- 指针所指向的类型

在指针声明语句中，<数据类型>字段就是指针所指向的类型。

```
piIndex:POINTER TO INT;          (*指针所指向的类型是 INT 型*）

pt:POINTER TO ARRAY [0..7] OF BYTE;  (*指针所指向的类型是字节型数组*）
```

- 指针的地址

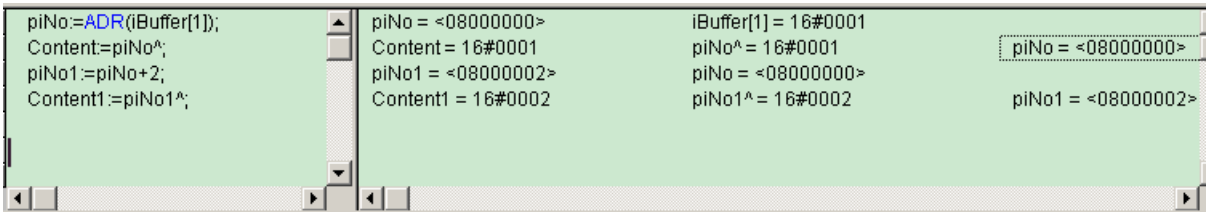
作为变量的一种，指针变量本身也占用内存区，也有指向自己的指针。在 CoDeSys 平台里，指针本身占据了 4 个字节的长度。

相邻的两个相对地址，其绝对地址也是相邻的。因为 AC500 系列 PLC 的存储区是按字节存储的，因此：访问字节时，指针值加 1；访问字或整形或定时器或计数器的当前值时，指针值加 2；访问双字时，指针值加 4。下图示例中相邻整形数据指针的运算。

定义：

```
VAR
    iBuffer AT %MW0.0:ARRAY[1..10]OF INT:=1,2,3,4,5,6,7,8,9,0;
    piNo:POINTER TO INT;
    piNo1:POINTER TO INT;
    Content:INT;
    Content1:INT;
END_VAR
```

程序及运行结果：



- 指针的内容

指针的内容指的是该指针所指地址的数据内容。在 CoDeSys 中，通过在指针变量后增加^符号获得指针的内容，如上例程序中第 2 行和第 4 行。

2. 指针的应用

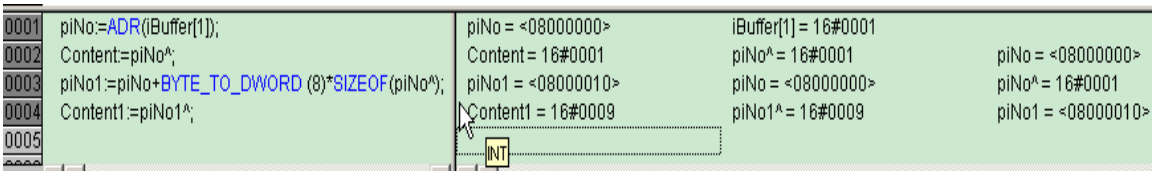
- 取地址指令 ADR

CoDeSys 中提供 ADR 指令取得输入变量的内存地址并输出。该地址可在程序内当作指针使用，也可作为指针传送给功能。如上例程序中第 1 行。

- 地址运算

作为一种特殊的变量，指针也可以进行加减运算，但是含义有所不同。指针加减运算的主要目的是指向前一个或后一个存储区。

CoDeSys 中可以通过使用 SIZEOF（指针名^）来计算指针所指向的数据类型长度，用于指针计算。如上例程序还可以如下编写。



程序中 piNo 指向数组第一个元素，初始化数值为 1，通过第 3 行的指令，数组向后移动 8 个 INT 型数据区，指向数组第 9 个元素，初始化为 9。

v.枚举

列举为自定义数据类型，由一长串的数字常量组成，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  数据类型里，通过在数据类型上添加对象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

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
	（*每个颜色的初始值分别为 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。

D.　POU（程序组织单元）

POU 是程序组织单元（Program Organization Unit）的简称。 POU 按其功能性的递增顺序，可分为功能（Function），功能块（Function Block）和程序（Program）三种类型。它们在某些特性方面有所不同。POU 彼此间能够带有或不带有参数地相互调用。

i.功能（Function）

可以赋予参数但没有记忆的 POU。功能本身不能单独执行，只能由程序或功能块调用执行。函数在执行时，会针对一系列特定的输入，产生一个输出结果，这个输出结果被赋给函数本身，称为返回值。对于相同的输入参数，功能总是返回相同的结果（没有内部存储空间）。

使用功能时，应遵循以下规则：

- 功能内部不能读写全局变量；
- 功能内部不能读写直接地址；
- 功能内部不能调用功能块。

ii.功能块（Function Block）

可以赋予参数并具有记忆的 POU。当以相同的输入参数调用时，功能块（例如定时器或计数器）的输出取决于其内部变量和外部接口变量的状态，这些状态存储在功能块的实例中。功能块的输入量可以是一个或多个，输出量也可以是一个或多个。

- 有历史数据的功能块

对于有历史数据的功能块，如果要需要得到不同的数据结果，必须为每一个实例定义不同的实例名称。

- 无历史数据的功能块

对于无历史数据的功能块，只能有一个实例被定义为该功能块类型。使用不同的 I/O 值，同一实例名可多次使用。

iii.程序（Program）

程序是唯一可执行的用户程序的主体，程序可以调用功能块、功能和其他程序。“主程序”是该类型 POU 的代表。程序是最高层的 POU，能存取赋予 PLC 直接地址的变量，并使它们能为其它 POU 所存取。程序可以通过任务组态来激活，也可以通过其他程序来调用。

POU 类型	缩写	含义
Program	PRG	例如主程序，可支配全局变量
Function block	FB	带输入和输出变量的块，最常用的 POU 类型
Function	FUN	具有功能值的块，用于扩展 PLC 的基本运算符

联系我们

北京**ABB**电气传动系统有限公司
地址：北京朝阳区酒仙桥北路甲十号D区1号
邮编：100015
总机：（+8610）58217788
传真：（+8610）58217618
24小时服务热线：（+86）400 810 8885
网址:www.abb.com.cn/PLC