

Allen-Bradley

# Logix5000™控制器
# 过程控制与驱动指令集

**1756-Lx, 1769-Lx, 1789-Lx,**
**1794-Lx, PowerFlex 700S**

参考手册

Rockwell Automation

## 重要用户信息

由于本出版物中所述产品使用方法不同，为了对产品的应用负责，在使用时必须按照所有必要步骤和要求操作，以确保每个实际应用都满足性能指标且符合安全要求，包括应用准则、规则、规范和标准。罗克韦尔自动化对使用此产品所造成的间接或因此而产生的损坏不负任何责任。

本手册中的图解、表格、程序实例和电路设计举例，完全是作为范例给出。因为各个应用不尽相同，且对于每个应用涉及到具体安装时又有许多不同要求，罗克韦尔自动化对以本出版物的范例为基础的实际应用所造成的后果不负任何责任(包括智能化产品的可靠性)。
艾伦—布拉德利出版物SGI-1.1，《固态控制的应用、安装和维护安全性指南》(可从当地罗克韦尔自动化办事处获取)，说明了固态设备与机电设备的重要差别，在应用产品时(如本手册所述的产品)必须考虑这些差异。

没有罗克韦尔自动化的书面授权许可，禁止全部或部分复制此具有版权资料的内容。
在本书中，我们使用以下的注释符号提醒您安全方面的注意事项。其中的注释和相关说明有助于用户去识别和避免潜在危险，并意识到潜在危险的后果：

**警　　告**



指明在危险环境下会发生爆炸进而导致人身伤害或死亡、财产毁坏或经济损失的行为和情况。

**注　　意**



指明可导致人身伤害或死亡，财产毁坏或经济损失的行为和环境。

**重　　要**

指明成功应用和理解产品的关键信息。

## 介 绍

该版本的文档包括了新增和更改的信息。为了帮助尽快找到这些内容，特将有变化的项目列于下段。

## 更新信息

该文档包括如下更新信息：

| 更新： | 查阅： |
|---|---|
| 指令索引包括每条指令名称 | 指令索引 |
| 关于SCRV功能的附加信息 | 章2 |
| 关于如何选择功能块元素，包括IREF、OREF、ICON和OCON | 附录A |
| 周期型定时方式下使用功能块指令的更新信息 | 附录A |
| 说明结构文本CASE结构与C/C++转换语句的区别 | 附录B |

**注释：**

## 快速查找指令

使用此索引可查找到Logix 指令的详细内容(灰色的指令可在其他手册查到)。该索引同时列出了该指令可使用的编程语言。

| 如果索引指示： | 指令所在文档： |
|---|---|
| 页码 | 本手册中 |
| 基本指令 | Logix5000 控制器基本指令参考手册，出版号1756-RM003 |
| 运动控制 | Logix5000控制器运动指令集参考手册，出版号1756-RM007 |

| 指令： | 位置： | 编程语言： |
|---|---|---|
| ABL<br>ASCII测试缓冲区行 | 基本指令 | 梯形图<br>结构文本 |
| ABS<br>绝对值 | 基本指令 | 梯形图<br>结构文本<br>功能块 |
| ACB<br>ASCII计数缓冲区中字符数 | 基本指令 | 梯形图<br>结构文本 |
| ACL<br>ASCII清零缓冲区 | 基本指令 | 梯形图<br>结构文本 |
| ACOS<br>余弦 | 基本指令 | 结构文本 |
| ACS<br>反余弦 | 基本指令 | 梯形图<br>功能块 |
| ADD<br>加 | 基本指令 | 梯形图<br>结构文本<br>功能块 |
| AFI<br>恒假 | 基本指令 | 梯形图 |
| AHL<br>ASCII 握手信号线 | 基本指令 | 梯形图<br>结构文本 |
| ALM | 1-2 | 结构文本<br>功能块 |
| AND<br>按位与 | 基本指令 | 梯形图<br>结构文本<br>功能块 |
| ARD<br>ASCII读 | 基本指令 | 梯形图<br>结构文本 |
| ARL<br>ASCII读行 | 基本指令 | 梯形图<br>结构文本 |
| ASIN<br>正弦 | 基本指令 | 结构文本 |

| 指令： | 位置： | 编程语言： |
|---|---|---|
| ASN<br>反正弦 | 基本指令 | 梯形图<br>功能块 |
| ATAN<br>正切 | 基本指令 | 结构文本 |
| ATN<br>反正切 | 基本指令 | 梯形图<br>功能块 |
| AVE<br>文件平均值 | 基本指令 | 梯形图 |
| AWA<br>ASCII写扩展 | 基本指令 | 梯形图<br>结构文本 |
| AWT<br>ASCII写 | 基本指令 | 梯形图<br>结构文本 |
| BAND<br>逻辑与 | 基本指令 | 结构文本<br>功能块 |
| BNOT<br>逻辑非 | 基本指令 | 结构文本<br>功能块 |
| BOR | 基本指令 | 结构文本<br>功能块 |
| BRK<br>终止 | 基本指令 | 梯形图 |
| BSL<br>位左移 | 基本指令 | 梯形图 |
| BSR<br>位右移 | 基本指令 | 梯形图 |
| BTD<br>位域分配 | 基本指令 | 梯形图 |
| BTDT<br>带目标的位域分配 | 基本指令 | 结构文本<br>功能块 |
| BTR<br>MESSAGE | 基本指令 | 梯形图<br>结构文本 |

| 指令： | 位置： | 编程语言： |
|---|---|---|
| BTW<br>MESSAGE | 基本指令 | 梯形图<br>结构文本 |
| BXOR<br>逻辑异或 | 过程控制 | 结构文本<br>功能块 |
| CLR<br>清零 | 基本指令 | 梯形图<br>结构文本 |
| CMP<br>综合比较 | 基本指令 | 梯形图 |
| CONCAT<br>连接字符串 | 基本指令 | 梯形图<br>结构文本 |
| COP<br>文件复制 | 基本指令 | 梯形图<br>结构文本 |
| COS<br>余弦 | 基本指令 | 梯形图<br>结构文本<br>功能块 |
| CPS<br>同步文件复制 | 基本指令 | 梯形图<br>结构文本 |
| CPT<br>综合计算 | 基本指令 | 梯形图 |
| CTD<br>减计数 | 基本指令 | 梯形图 |
| CTU<br>加计数 | 基本指令 | 梯形图 |
| CTUD<br>加/减计数 | 基本指令 | 结构文本<br>功能块 |
| D2SD<br>离散2态设备 | 1-6 | 结构文本<br>功能块 |
| D3SD<br>离散3态设备 | 1-15 | 结构文本<br>功能块 |
| DDT<br>诊断检测 | 基本指令 | 梯形图 |
| DEDT<br>死区时间 | 1-28 | 结构文本<br>功能块 |
| DEG<br>转换成角度 | 基本指令 | 梯形图<br>结构文本<br>功能块 |
| DELETE<br>删除字符串 | 基本指令 | 梯形图<br>结构文本 |
| DERV<br>微分 | 3-2 | 结构文本<br>功能块 |
| DFF<br>D触发器 | 6-2 | 结构文本<br>功能块 |

| 指令： | 位置： | 编程语言： |
|---|---|---|
| DIV<br>除法 | 基本指令 | 梯形图<br>结构文本<br>功能块 |
| DTOS<br>DINT转为字符串 | 基本指令 | 梯形图<br>结构文本 |
| DTR<br>数据传送 | 基本指令 | 梯形图<br>结构文本 |
| EOT<br>传送结束 | 基本指令 | 梯形图<br>结构文本 |
| EQU<br>等于 | 基本指令 | 梯形图<br>结构文本<br>功能块 |
| ESEL<br>增强型选择 | 4-2 | 结构文本<br>功能块 |
| EVENT<br>触发事件任务 | 基本指令 | 梯形图<br>结构文本 |
| FAL<br>文件算术与逻辑 | 基本指令 | 梯形图 |
| FBC<br>文件位比较 | 基本指令 | 梯形图 |
| FFL<br>FIFO装载 | 基本指令 | 梯形图 |
| FFU<br>FIFO卸载 | 基本指令 | 梯形图 |
| FGEN<br>函数发生器 | 1-33 | 结构文本<br>功能块 |
| FIND<br>查找字符串 | 基本指令 | 梯形图<br>结构文本 |
| FLL<br>文件填充 | 基本指令 | 梯形图 |
| FOR<br>循环 | 基本指令 | 梯形图 |
| FRD<br>转换成整数 | 基本指令 | 梯形图<br>功能块 |
| FSC<br>文件搜索与比较 | 基本指令 | 梯形图 |
| GEQ<br>大于等于 | 基本指令 | 梯形图<br>结构文本<br>功能块 |
| GRT<br>大于 | 基本指令 | 梯形图<br>结构文本<br>功能块 |
| GSV<br>获取系统值 | 基本指令 | 梯形图<br>结构文本 |

| 指令： | 位置： | 编程语言： |
|---|---|---|
| HLL<br>上/下限 | 4-9 | 结构文本<br>功能块 |
| HPF<br>高通滤波 | 3-6 | 结构文本<br>功能块 |
| ICON<br>输入线连接器 | A-1 | 功能块 |
| INSERT<br>插入字符串 | 基本指令 | 梯形图<br>结构文本 |
| INTG<br>积分器 | | 结构文本<br>功能块 |
| IOT<br>立即输出 | 基本指令 | 梯形图<br>结构文本 |
| IREF<br>输入参考 | A-1 | 功能块 |
| JKFF<br>JK触发器 | 6-4 | 结构文本<br>功能块 |
| JMP<br>跳转到标号 | 基本指令 | 梯形图 |
| JSR<br>跳转到子程序 | 基本指令 | 梯形图<br>结构文本<br>功能块 |
| JXR<br>跳转到外部程序 | 基本指令 | 梯形图 |
| LBL<br>标号 | 基本指令 | 梯形图 |
| LDL2<br>二阶超前-滞后 | 3-12 | 结构文本<br>功能块 |
| LDLG<br>超前-滞后 | 1-37 | 结构文本<br>功能块 |
| LEQ<br>小于或等于 | 基本指令 | 梯形图<br>结构文本<br>功能块 |
| LES<br>小于 | 基本指令 | 梯形图<br>结构文本<br>功能块 |
| LFL<br>LIFO 装载 | 基本指令 | 梯形图 |
| LFU<br>LIFO 卸载 | 基本指令 | 梯形图 |
| LIM<br>极限比较 | 基本指令 | 梯形图<br>功能块 |
| LN<br>自然对数 | 基本指令 | 梯形图<br>结构文本<br>功能块 |

| 指令： | 位置： | 编程语言： |
|---|---|---|
| LOG<br>以10为底的对数 | 基本指令 | 梯形图<br>结构文本<br>功能块 |
| LOWER<br>转换成小写 | 基本指令 | 梯形图<br>结构文本 |
| LPF<br>低通滤波 | 3-18 | 结构文本<br>功能块 |
| MAAT<br>运动应用轴调整 | 运动控制 | 梯形图<br>结构文本 |
| MAFR<br>传动轴故障复位 | 运动控制 | 梯形图<br>结构文本 |
| MAG<br>传动轴齿轮传动 | 运动控制 | 梯形图<br>结构文本 |
| MAH<br>传动轴复位指令 | 运动控制 | 梯形图<br>结构文本 |
| MAHD<br>运动应用接线诊断 | 运动控制 | r梯形图<br>结构文本 |
| MAJ<br>传动轴慢进给 | 运动控制 | 梯形图<br>结构文本 |
| MAM<br>传动轴传送 | 运动控制 | 梯形图<br>结构文本 |
| MAOC<br>运动提供凸轮输出 | 运动控制 | 梯形图<br>结构文本 |
| MAPC<br>运动轴位置凸轮 | 运动控制 | 梯形图<br>结构文本 |
| MAR<br>运动提供登录 | 运动控制 | 梯形图<br>结构文本 |
| MAS<br>传动轴停止 | 运动控制 | 梯形图<br>结构文本 |
| MASD<br>传动轴关机 | 运动控制 | 梯形图<br>结构文本 |
| MASR<br>传动轴关机复位 | 运动控制 | 梯形图<br>结构文本 |
| MATC<br>运动轴时间凸轮 | 运动控制 | 梯形图<br>结构文本 |
| MAVE<br>移动均值运算 | 5-2 | 结构文本<br>功能块 |
| MAW<br>运动提供监视 | 运动控制 | 梯形图<br>结构文本 |
| MAXC<br>捕捉最大值 | 5-6 | 结构文本<br>功能块 |

| 指令： | 位置： | 编程语言： |
|---|---|---|
| MCCP<br>运动计算凸轮轮廓 | 运动控制 | 梯形图<br>结构文本 |
| MCD<br>运动动态改变指令 | 运动控制 | 梯形图<br>结构文本 |
| MCR<br>主轴控制复位 | 基本指令 | 梯形图 |
| MDF<br>运动直接驱动关闭 | 运动控制 | 梯形图<br>结构文本 |
| MDO<br>运动直接驱动打开 | 运动控制 | 梯形图<br>结构文本 |
| MDOC<br>运动消除输出凸轮 | 运动控制 | 梯形图<br>结构文本 |
| MDR<br>运动消除登录 | 运动控制 | 梯形图<br>结构文本 |
| MDW<br>运动消除监视 | 运动控制 | 梯形图<br>结构文本 |
| MEQ<br>屏蔽等于 | 基本指令 | 梯形图<br>结构文本<br>功能块 |
| MGS<br>运动组停止 | 运动控制 | 梯形图<br>结构文本 |
| MGSD<br>运动组关机 | 运动控制 | 梯形图<br>结构文本 |
| MGSP<br>运动组选通位置 | 运动控制 | 梯形图<br>结构文本 |
| MGSR<br>运动组关机复位 | 运动控制 | 梯形图<br>结构文本 |
| MID<br>抽取字符串 | 基本指令 | 梯形图<br>结构文本 |
| MINC<br>捕捉最小值 | 5-8 | 功能块 |
| MOD<br>求模数 | 基本指令 | 梯形图<br>结构文本<br>功能块 |
| MOV<br>传送 | 基本指令 | 梯形图 |
| MRAT<br>运动运行轴调整 | 运动控制 | 梯形图<br>结构文本 |
| MRHD<br>运动运行接线诊断 | 运动控制 | 梯形图<br>结构文本 |

| 指令： | 位置： | 编程语言： |
|---|---|---|
| MRP<br>运动重定义位置 | 运动控制 | 梯形图<br>结构文本 |
| MSF<br>运动伺服关闭 | 运动控制 | 梯形图<br>结构文本 |
| MSG<br>通讯指令 | 基本指令 | 梯形图<br>结构文本 |
| MSO<br>运动伺服打开 | 运动控制 | 梯形图<br>结构文本 |
| MSTD<br>移动标准偏差 | 基本指令 | 结构文本<br>功能块 |
| MUL<br>乘 | 基本指令 | 梯形图<br>结构文本<br>功能块 |
| MUX<br>多路切换器 | 4-12 | 功能块 |
| MVM<br>屏蔽传送 | 基本指令 | 梯形图 |
| MVMT<br>带目标的屏蔽传送 | 基本指令 | 结构文本<br>功能块 |
| NEG<br>取反 | 基本指令 | 梯形图<br>结构文本<br>功能块 |
| NEQ<br>不等于 | 基本指令 | 梯形图<br>结构文本<br>功能块 |
| NOP<br>空操作 | 基本指令 | 梯形图 |
| NOT<br>按位非 | 基本指令 | 梯形图<br>结构文本<br>功能块 |
| NTCH<br>凹波滤波 | 过程控制 | 结构文本<br>功能块 |
| ONS<br>单脉冲触发 | 基本指令 | 梯形图 |
| OR<br>按位或 | 基本指令 | 梯形图<br>结构文本<br>功能块 |
| OREF<br>输出参考 | A-1 | 功能块 |
| OSF<br>下降沿触发 | 基本指令 | 梯形图 |

| 指令： | 位置： | 编程语言： |
|---|---|---|
| OSFI<br>带输入下降沿触发 | 基本指令 | 结构文本<br>功能块 |
| OSR<br>上升沿触发 | 基本指令 | 梯形图 |
| OSRI<br>带输入上升沿触发 | 基本指令 | 结构文本<br>功能块 |
| OTE<br>输出使能 | 基本指令 | 梯形图 |
| OTL<br>输出锁存 | 基本指令 | 梯形图 |
| OUT<br>输出解锁存 | 基本指令 | 梯形图 |
| PI<br>比例+积分 | 2-8 | 结构文本<br>功能块 |
| PID<br>比例积分微分控制 | 基本指令 | 梯形图<br>结构文本 |
| PIDE<br>增强型PID | 1-41 | 结构文本<br>功能块 |
| PMUL<br>多路脉冲切换器 | 2-20 | 结构文本<br>功能块 |
| POSP<br>位置比例 | 1-75 | 结构文本<br>功能块 |
| RAD<br>转换成弧度 | 基本指令 | 梯形图<br>结构文本<br>功能块 |
| RES<br>复位 | 基本指令 | 梯形图 |
| RESD<br>复位优先 | 6-6 | 结构文本<br>功能块 |
| RET<br>返回 | 基本指令 | 梯形图<br>结构文本<br>功能块 |
| RLIM<br>速率极限 | 4-15 | 功能块 |
| RMPS<br>斜坡/渗透 | 1-82 | 结构文本<br>功能块 |
| RTO<br>保持定时器 | 基本指令 | 梯形图 |
| RTOR<br>带复位的保持定时器 | 基本指令 | 结构文本<br>功能块 |
| RTOS<br>REAL 转换成字符串 | 基本指令 | 梯形图<br>结构文本 |

| 指令： | 位置： | 编程语言： |
|---|---|---|
| SBR<br>子程序 | 基本指令 | 梯形图<br>结构文本<br>功能块 |
| SCL<br>整定 | 1-96 | 功能块<br>结构文本 |
| SCRV<br>S_曲线 | 2-28 | 结构文本<br>功能块 |
| SEL<br>选择 | 4-19 | 功能块 |
| SETD<br>置位优先 | 6-8 | 结构文本<br>功能块 |
| SFP<br>SFC暂停 | 基本指令 | 梯形图<br>结构文本 |
| SFR<br>SFC复位 | 基本指令 | 梯形图<br>结构文本 |
| SIN<br>正弦 | 基本指令 | 梯形图<br>功能块 |
| SIZE<br>元素大小 | 基本指令 | 梯形图<br>结构文本 |
| SNEG<br>选择取反 | 4-21 | 结构文本<br>功能块 |
| SOC<br>二阶控制器 | 2-38 | 结构文本<br>功能块 |
| SQI<br>顺序器输入 | 基本指令 | 梯形图 |
| SQL<br>顺序器装载 | 基本指令 | 梯形图 |
| SQO<br>顺序器输出 | 基本指令 | 梯形图 |
| SQR<br>平方根 | 基本指令 | 梯形图<br>功能块 |
| SQRT<br>平方根 | 基本指令 | 结构文本 |
| SRT<br>文件排序 | 基本指令 | 梯形图<br>结构文本 |
| SRTP<br>脉宽调制 | 1-100 | 结构文本<br>功能块 |
| SSUM<br>选择求和 | 4-23 | 结构文本<br>功能块 |
| SSV<br>设置系统值 | 基本指令 | 梯形图<br>结构文本 |

| 指令： | 位置： | 编程语言： |
|---|---|---|
| STD<br>文件标准偏差 | 基本指令 | 梯形图 |
| STOD<br>字符串转换成DINT | 基本指令 | 梯形图<br>结构文本 |
| STOR<br>字符串转换成REAL | 基本指令 | 梯形图<br>结构文本 |
| SUB<br>减 | 基本指令 | 梯形图<br>结构文本<br>功能块 |
| SWPB<br>交换字节 | 基本指令 | 梯形图<br>结构文本 |
| TAN<br>正切 | 基本指令 | 梯形图<br>结构文本<br>功能块 |
| TND<br>暂停 | 基本指令 | 梯形图 |
| TOD<br>转换成BCD码 | 基本指令 | 梯形图<br>功能块 |
| TOF<br>延时断开计时器 | 基本指令 | 梯形图 |
| TOFR<br>带复位延时断开计时器 | 基本指令 | 结构文本<br>功能块 |
| TON<br>延时导通计时器 | 基本指令 | 梯形图 |
| TONR<br>带复位延时导通计时器 | 基本指令 | 结构文本<br>功能块 |
| TOT<br>累加器 | 1-106 | 结构文本<br>功能块 |
| TRN<br>截断 | 基本指令 | 梯形图<br>功能块 |
| TRNC | 基本指令 | 结构文本 |
| UID<br>禁止用户中断 | 基本指令 | 梯形图<br>结构文本 |
| UIE<br>使能用户中断 | 基本指令 | 梯形图<br>结构文本 |
| UPDN<br>增/减累加器 | 2-46 | 结构文本<br>功能块 |
| UPPER<br>转换成大写 | 基本指令 | 梯形图<br>结构文本 |
| XIC<br>检查是否闭和 | 基本指令 | 梯形图 |
| XIO<br>检查是否断开 | 基本指令 | 梯形图 |
| XOR<br>按位异或 | 基本指令 | 梯形图<br>结构文本<br>功能块 |
| XPY<br>X的Y次幂 | 基本指令 | 梯形图<br>结构文本<br>功能块 |

## 简介

本手册是Logix系列基本指令资料之一

| 任务/目标： | 参见资料： |
|---|---|
| 控制器顺序控制应用编程 | Logix5000控制器基本指令参考手册，<br>出版号：1756-RM003 |
| 控制器过程控制<br>驱动应用编程<br>本手册<br>**You are here**  | Logix5000控制器过程控制及驱动指令参考手册，<br>出版号：1756-RM006 |
| 运动系统应用编程 | Logix5000控制器运动指令参考手册，<br>出版号：1756-RM007 |
| 导入文本文件或标签到一个项目<br>导出一个项目或标签到文本文件 | Logix5500控制器导入/导出文件参考手册，<br>出版号：1756-RM084 |
| 转换PLC5或SLC500应用程序到Logix5000应用程序 | Logix5500控制器转换PLC-5或SLC500程序逻辑到Logix5000应用程序参考手册，<br>出版号：1756-RM085 |

以下是介绍Logix5000系列控制器的核心文档：

| 如果用户是： | 使用出版物： |
|---|---|
| Logix5000控制器的新用户<br>该快速入门为用户提供了完成控制器组态和运行所需每一基本步骤。 | Logix5000控制器快速入门<br>出版号：1756-QS00x |
| 有Logix5000控制器使用经验的用户<br>该系统参考手册提供了组态信息、控制器特性和指令(梯形图、功能块和结构文本)的高级索引。 | Logix5000控制器系统参考手册<br>出版号：1756-QR107 |
| Logix5000控制器的任一用户<br>通用规程手册说明了所有Logix5000控制器的通用特性和功能 | Logix5000控制器通用规程<br>出版号：1756-PM001 |

## 本手册使用对象

本手册为程序设计者提供Logix系列控制器所用指令的详细信息。用户首先应该已经熟悉Logix系列控制器存储并处理数据的原理。

初学者在使用一条指令之前，应该首先阅读与指令有关的详细资料。有经验的程序设计者可以只查阅有关信息来确认指令的详细内容。

## 本手册的用途

本手册以如下格式为每条指令提供说明：

| 该区域： | 提供以下信息： |
| --- | --- |
| 指令名称 | 标识指令<br>确定指令是输入指令或输出指令 |
| 操作数 | 列出指令所有操作数类型<br><br> 在梯形图中能够使用的操作数类型<br><br> 在结构文本中能够使用的操作数类型<br><br> 在功能块中能够使用的操作数类型<br><br>功能块所显示的引脚均为默认引脚，操作数列表显示功能块所有可能引脚的操作数。 |
| 指令结构体 | 列出指令的控制状态位及其数值 |
| 说明 | 说明指令的用法<br>如果需要，说明指令使能时和禁止时的区别 |
| 算术状态标志 | 指示指令是否影响算术状态标志<br>参阅附录通用属性 |
| 故障条件 | 确定指令是否产生次要或主要故障，如果产生，指明故障类型及故障代码 |
| 执行 | 详细描述该指令在特定条件下如何操作的 |
| 实例 | 每种可用编程语言至少提供一个实例，每个实例都包含解释说明 |

以下图标用于帮助定义编程语言的特殊信息

| 图标 | 表示以下编程语言 |
| --- | --- |
|  | 梯形图 |
|  | 结构文本 |
|  | 功能块 |

## 所有指令通用信息

Logix5000指令集具有以下通用属性：

| 信息： | 参考附录： |
|---|---|
| 通用属性 | 在通用属性附录中说明：<br>· 算术状态标志<br>· 数据类型<br>· 关键字 |
| 数组 | 在数组概念附录中定义了数组的概念并解释了控制器如何处理数组。 |
| 功能块属性 | 在功能块属性附录中说明了：<br>· 编程和操作员控制<br>· 定时模式 |

## 约定和相关术语

### 置位（设置）和清零

本手册使用置位/设置和清零来定义位(布尔型)和数值(非布尔型)的状态：

| 术语： | 含义： |
|---|---|
| 置位 | 置位—位为1(ON)<br>置位—数值为非零数值 |
| 清零 | 清零—位为0(OFF)<br>清零—数值所有位 |

假如操作数或参数支持多种数据类型，则黑体字数据类型为最优数据类型。指令的所有操作数的使用同种最优数据类型，则指令执行速度最快且占用内存最小，典型的最优数据类型是DINT和REAL。

## 梯级运行条件

控制器根据指令前面的梯级条件(梯级输入条件)判断梯形图指令。然后根据梯级条件和指令，控制器设置下一指令的梯级条件(梯级输出条件)，这样交替影响后面的指令。




如果一个输入指令的梯级输入条件为真，则控制器执行该指令，并根据指令执行的结果，设置梯级输出条件。如果指令执行结果为真，则输出梯级条件为真；如果该指令执行结果为假，则输出梯级条件为假。

控制器同时预扫描梯级。预扫描是控制器对所有梯级的一种特殊扫描方式，控制器在预扫描期间扫描所有主例程和子例程，但会忽略跳过执行的跳转指令。控制器执行所有For循环和子程序调用。如果一个子程序被多次调用，则每次调用都执行。控制器通过预扫描梯形图指令来重置非保持型I/O和内部数据。

在预扫描期间输入数据不是当前数据，输出数据也不写入。以下条件产生预扫描：

- 将程序从编程状态转换到运行状态；
- 从上电状态自动转换到运行状态。

当程序处于下列条件时不进行预扫描：

- 在控制器运行时预定程序；
- 在程序进入运行模式时非预定程序。

## 功能块状态

控制器根据不同的状态条件执行功能块指令。

| 可能的条件： | 描述： |
|---|---|
| 预扫描 | 功能块例程的预扫描与梯形图的预扫描相同。唯一的不同就是每个功能块使能参数在预扫描期间都被清零。 |
| 指令首次扫描 | 指令首次扫描是指指令在预扫描后首次执行。控制器使用首次指令扫描来读当前输入并确定输入的状态。 |
| 指令首次执行 | 指令首次执行是指指令在新的数据结构体的情况下的首次运行。控制器使用首次运行来产生系数和其它数据存储，但不改变初始下载的功能块 |

每个功能块指令包括输入使能和输出使能参数：

- 当输入使能位置位时功能块正常执行；
- 当输入使能位清零时功能块只执行预扫描逻辑、后期扫描逻辑或跳过正常算法执行。
- 输出使能位为输入使能位映象，但是当功能块执行时检测到一个溢出条件，则输出使能位也被清零。
- 当输入使能位从清零变为置位时，功能块从断点继续执行。然而有些功能块指令包含特殊功能函数，如重新初始化，当使能位由清零转换到置位状态，则重新初始化。对于有时间基参数的功能块指令且定时模式为过采样方式，当输入使能位从清零转换为置位时，功能块总是从断点继续执行。

当输入使能位不连接时，指令总是将输入使能位置位条件正常执行。如果用户清零输入使能位，则下次指令执行时置位。

**重　　要**

由于其内部浮点数计算使用单精度浮点数，在功能块编程时，工程量范围严格限制在+/-10+/-15。如果工程单位超出此范围但接近单精度浮点数极限(+/-10+/-38)会导致计算结果精度降低。

注释：

# 目录

**过程控制指令(ALM, D2SD, D3SD, DEDT, FGEN, LDLG, PIDE, POSP, RMPS, SCL, SRTP, TOT)**

## 第一章


概述 ……1-1
报警(ALM) ……1-2
监控报警指令 ……1-4
离散式2态设备(D2SD) ……1-6
监控D2SD指令 ……1-9
程序和操作员控制切换 ……1-11
程序控制中的命令状态 ……1-11
操作员控制中的命令状态 ……1-12
手动方式或自动方式 ……1-12
输出状态 ……1-13
故障报警条件 ……1-13
方式报警条件 ……1-14
离散式3态设备(D3SD) ……1-15
监控D3SD指令 ……1-20
程序和操作员控制切换 ……1-23
程序控制中的命令状态 ……1-23
操作员控制中的命令状态 ……1-24
手动方式或自动方式 ……1-24
输出状态 ……1-26
故障报警条件 ……1-26
方式报警条件 ……1-27
死区时间(DEDT) ……1-28
死区时间缓冲器的维护 ……1-30
故障过渡指令动作 ……1-31
函数发生器(FGEN) ……1-33
超前-滞后补偿(LDLG) ……1-37
增强型PID(PIDE) ……1-41
计算CV值 ……1-53
监控PIDE指令 ……1-53
程序和操作员控制切换 ……1-60
操作方式 ……1-61
选择设定点 ……1-62
PV值上/下限报警 ……1-64
转化PV值和SP值为百分数 ……1-66
偏差上/下限报警 ……1-67
零点交叉死区控制 ……1-68
选择控制变量 ……1-69
主回路控制 ……1-73
故障处理 ……1-74

位置比例(POSP) ……1-75
位置及设定点数值整定 ……1-77
POSP 指令如何使用内部周期定时器 ……1-77
产生输出脉冲 ……1-78
脉宽计算 ……1-79
斜坡/渗透(RMPS) ……1-82
监控RMPS 指令 ……1-86
指令首次扫描应用的初始方式 ……1-87
程序控制和操作员控制切换 ……1-89
程序控制 ……1-91
操作员控制 ……1-92
执行斜坡/渗透功能 ……1-93
整定 (SCL) ……1-96
报警 ……1-98
极限 ……1-98
脉宽调制(SRTP) ……1-100
使用内部周期定时器 ……1-102
计算加热及制冷时间 ……1-102
累加器 (TOT) ……1-106
监控 TOT指令 ……1-110
检测低输入截止 ……1-111
操作方式 ……1-112
复位TOT 指令 ……1-113
计算累加值 ……1-113
判断是否达到目标值 ……1-113

## 第二章

## 驱动指令(INTG, PI, PMUL, SCRV, SOC, UPDN)

概述 ……2-1
积分器(INTG) ……2-2
极限 ……2-4
比例+积分算法 (PI) ……2-8
线性方式下的运算 ……2-12
非线性方式下的运算 ……2-12
极限 ……2-15
脉冲倍频器 (PMUL) ……2-20
计算输出值和余数值 ……2-22
S-曲线(SCRV) ……2-28
计算输出值和速率值 ……2-33
二阶控制器 (SOC) ……2-38
参数极限 ……2-41
极限 ……2-41
增/减累加器 (UPDN) ……2-46

## 第三章

**滤波指令(DERV, HPF, LDL2, LPF, NTCH)**

概述 ……3-1
微分 (DERV) ……3-2
高通滤波器 (HPF) ……3-6
二阶超前_滞后滤波器(LDL2) ……3-12
低通滤波器(LPF) ……3-18
凹波滤波器(NTCH) ……3-24

## 第四章

**选择/限制指令(ESEL, HLL, MUX, RLIM, SEL, SNEG, SSUM)**

概述 ……4-1
增强型选择 (ESEL) ……4-2
监控ESEL 指令 ……4-6
程序控制与操作员控制间切换 ……4-8
上/下限限幅(HLL) ……4-9
多路切换器 (MUX) ……4-12
速率限幅器 (RLIM) ……4-15
选择 (SEL) ……4-19
选择取反 (SNEG) ……4-21
选择求和(SSUM) ……4-23

## 第五章

**统计指令(MAVE, MAXC, MINC, MSTD)**

概述 ……5-1
移动均值运算 (MAVE) ……5-2
初始化均值算法 ……5-4
获取最大值 (MAXC) ……5-6
获取最小值(MINC) ……5-8
移动标准偏差(MSTD) ……5-10
初始化标准偏差算法 ……5-12

## 第六章

**移动/逻辑指令(DFF, JKFF, RESD, SETD)**

概述 ……6-1
D触发器 (DFF) ……6-2
JK 触发器(JKFF) ……6-4
复位优先(RESD) ……6-6
置位优先(SETD) ……6-8

功能块属性

## 附录 A

概述 ……A-1
选择功能块单元 ……A-1
数据锁存 ……A-2
执行顺序 ……A-4
解析回路 ……A-5
解析两功能块间数据流 ……A-6
创建一个扫描周期的延迟 ……A-7
总结 ……A-7
功能块对溢出条件的响应 ……A-7
定时方式 ……A-9
用于定时方式的通用指令参数 ……A-11
定时方式概述 ……A-13
程序/操作员控制 ……A-14

结构文本编程

## 附录 B

概述 ……B-1
结构文本语法 ……B-1
赋值语句 ……B-3
指定非保持赋值语句 ……B-4
为字符串赋值ASCII码 ……B-5
表达式 ……B-5
使用算术运算符和功能 ……B-7
使用关系运算符 ……B-8
使用逻辑运算符 ……B-10
使用位运算符 ……B-11
确定运算顺序 ……B-11
指令 ……B-12
结构 ……B-13
IF...THEN ……B-14
CASE...OF ……B-17
FOR...DO ……B-19
WHILE...DO ……B-22
REPEAT...UNTIL ……B-25
注释 B-28

通用属性

## 附录C

概述 ……C-1
立即数 ……C-1
数据转换 ……C-1
SINT 或INT转换为DINT ……C-3
Integer 转换为REAL ……C-5
DINT转换为SINT 或 INT ……C-5
REAL 转换为Integer ……C-6

## 功能块面板控制

### 附录D

概述 ……D-1
组态基本属性 ……D-2
组态显示属性 ……D-3
组态服务器属性 ……D-4
组态字体属性 ……D-5
ALM 控制 ……D-6
ESEL 控制 ……D-8
TOT 控制 ……D-9
RMPS 控制 ……D-11
D2SD 控制 ……D-14
D3SD 控制 ……D-16
PIDE 控制 ……D-18

注释：

# 过程控制指令(ALM, D2SD, D3SD, DEDT, FGEN, LDLG, PIDE, POSP, RMPS, SCL, SRTP, TOT)

**概述：**

以下过程控制指令可用：

| 如果用户需要： | 使用指令： | 可用语言： | 参考页码： |
|---|---|---|---|
| 模拟量信号报警 | 报警(ALM) | 结构文本<br>功能块 | 1-2 |
| 控制只有两个状态的离散型设备，例如开/关、打开/闭合等。 | 离散型-2态设备(D2SD) | 结构文本<br>功能块 | 1-6 |
| 控制具有三种可能状态的离散型设备，例如快/慢/关、前进/停止/后退等。 | 离散型-3态设备(D3SD) | 结构文本<br>功能块 | 1-15 |
| 单输入信号延时。用户可选择延时时间量。 | 死区时间(DEDT) | 结构文本<br>功能块 | 1-28 |
| 输入信号分段线性转换功能。 | 函数发生器(FGEN) | 结构文本<br>功能块 | 1-33 |
| 对输入信号进行相位超前/滞后补偿。 | 超前-滞后(LDLG) | 结构文本<br>功能块 | 1-37 |
| 利用PID算法调节模拟量输出保持在给定的设定点。 | 增强型PID(PIDE) | 结构文本<br>功能块 | 1-41 |
| 通过控制开或关脉冲来提升或下放设备 | 位置比例(POSP) | 结构文本<br>功能块 | 1-75 |
| 提供交互的斜坡和渗透周期。 | 斜坡/渗透(RMPS) | 结构文本<br>功能块 | 1-82 |
| 将一个非整定的输入量整定为一个浮点型的工程量。 | 整定(SCL) | 结构文本<br>功能块 | 1-96 |
| 用一个周期性脉冲表示PID回路和驱动器加热及冷却的数字量输出的0-100%百分数。 | 脉宽调制(SRTP) | 结构文本<br>功能块 | 1-100 |
| 模拟量输入在一个时间段内的累计值。 | 累加器(TOT) | 结构文本<br>功能块 | 1-106 |

# 报警(ALM)

报警指令为任一个模拟量信号提供报警功能。

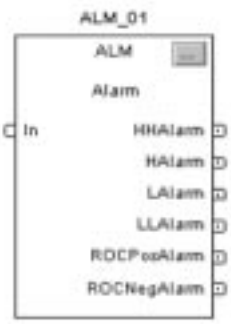
## 操作数：

```
ALM(ALM_tag);
```

ALM(ALM_tag); 结构文本

| 操作数： | 类型： | 格式： | 说明： |
|---|---|---|---|
| ALM 标签 | ALARM | 结构体 | ALM 结构体 |

## 功能块




| 操作数： | 类型： | 格式： | 说明： |
|---|---|---|---|
| ALM 标签 | ALARM | 结构体 | ALM 结构体 |

## 报警结构体

| 输入参数： | 数据类型： | 说明： |
|---|---|---|
| EnableIn | BOOL | 功能块：<br>如果该位清零，则指令不执行且输出不更新。<br>如果置位，指令执行。<br>缺省状态为置位。<br>结构文本：<br>无影响。指令一直执行。 |
| In | REAL | 模拟量信号输入。<br>有效值＝浮点数<br>缺省值＝0.0 |
| HHLimit | REAL | 输入最高报警上限。<br>有效值＝实数<br>缺省值＝最大正数 |
| HLimit | REAL | 输入报警上限<br>有效值＝实数<br>缺省值＝最大正数 |
| LLimit | REAL | 输入报警下限<br>有效值＝实数<br>缺省值＝最小负数 |
| LLLimit | REAL | 输入最低报警下限<br>有效值＝实数<br>缺省值＝最小负数 |
| DeadBand | REAL | 最高上限与最低下限报警的死区<br>有效值＝大于等于零的实数<br>缺省值＝0.0 |

| 输入参数： | 数据类型： | 说明： |
|---|---|---|
| ROCPosLimit | REAL | 每秒内输入信号正向(增加)变化报警频率限制。<br>设ROCPosLimit=0可禁止ROC正跳变报警。无效时，<br>该指令设定其为0.0，并置位状态字中的相应位。<br>有效值＝大于等于0的实数<br>缺省值＝0.0 |
| ROCNegLimit | REAL | 每秒内输入信号负向(减小)变化报警频率限制。<br>设ROCPNegLimit=0可禁止ROC负跳变报警。无效时，该指令设定其为0，并置位状态字中的相应位。<br>有效值＝大于等于0的实数<br>缺省值＝0.0 |
| ROCPeriod | REAL | 用于计算变化频率的时间周期。设ROCPeriod=0可禁止ROC报警并置ROC输出为零。<br>无效时，该指令设定值为0，并置位状态字中的相应位。<br>有效值＝大于等于0的实数<br>缺省值＝0.0 |

| 输出参数： | 数据类型： | 说明： |
|---|---|---|
| EnableOut | BOOL | 输出使能。 |
| HHAlarm | BOOL | 最高上限报警指示。 |
| HAlarm | BOOL | 上限报警指示。<br>缺省＝false |
| LAlarm | BOOL | 下限报警指示。<br>缺省＝false |
| LLAlarm | BOOL | 最低下限报警指示。<br>缺省＝false |
| ROCPosAlarm | BOOL | 正跳变频率报警指示。<br>缺省＝false |
| ROCNegAlarm | BOOL | 负跳变频率报警指示<br>缺省＝false |
| ROC | REAL | 跳变频率输出。该输出影响算术状态标志。 |
| Status | DINT | 功能块状态字 |
| InstructFault (Status.0) | BOOL | 指令检测如下的运行错误。该错误不使控制器产生次要或主要错误。检测其它状态位以确定发生故障的类型。 |
| DeadbandInv (Status.1) | BOOL | Deadband数值无效 |
| ROCPosLimitInv (Status.2) | BOOL | ROCPosLimit数值无效 |
| ROCNegLimitInv (Status.3) | BOOL | ROCNegLimit 数值无效 |
| ROCPeriodInv (Status.4) | BOOL | ROCPeriod数值无效 |

**说明：** 报警指令提供最高上限、上限、下限、最低下限、正跳变频率、负跳变频率报警指示。报警死区用于最高上限与最低下限报警。用户可自定义跳变频率的周期。

## 监控报警(ALM)指令

报警指令具有操作员面板。需要了解更多信息请参阅附录：功能块面板控制
算术状态标志：算术状态标志用于置位ROC输出。

**故障条件：** 无

**执行：**

| 条件: | 功能块动作: | 结构文本动作： |
|---|---|---|
| 预扫描 | 无动作。 | 无动作。 |
| 指令首次扫描 | 清除所有报警输出。<br>累计消耗时间清零。 | 清除所有报警输出。<br>累计消耗时间清零。 |
| 指令首次执行 | 清除所有报警输出。<br>累计消耗时间清零。 | 清除所有报警输出。<br>累计消耗时间清零。 |
| 输入使能(EnableIn)清零 | 输出使能(EnableOut)清零。<br>指令不执行，输出不更新。 | 不影响 |
| 输入(EnableIn)置位 | 执行指令。<br>输出使能(EnableOut)置位。 | 输入(EnableIn)一直置位。指令执行。 |
| 后期扫描 | 无动作。 | 无动作。 |

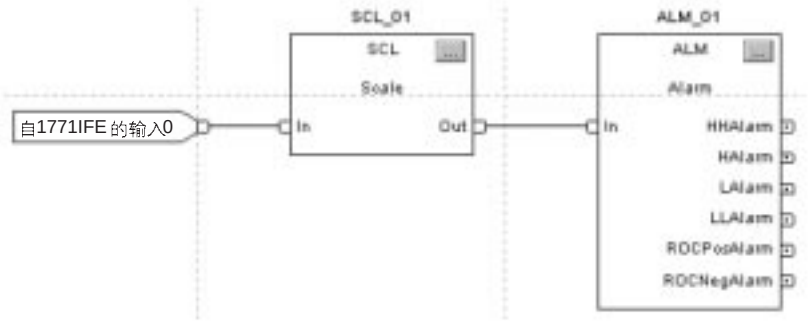
**功能块实例：** 报警指令典型应用与不支持在线报警或无法对计算的变量产生报警的模拟量输入模块配合使用(例如1771 I/O模块)。在该实例中，由1771-IFE模块输入的模拟量信号先用整定(SCL)指令变换为工程量。SCL指令的输出作为ALM指令的一个输入，决定是否置位报警。报警输出的参数就可以在用于程序中和(/或)在操作员界面中显示。

### 结构文本

```
SCL_01.In := Input0From1771IFE;
SCL(SCL_01);

ALM_01.In := SCL_01.Out;
ALARM(ALM_01);
```

### 功能块：



## 离散型2态设备(D2SD)

D2SD指令是用于控制只有两种状态的离散型设备，例如只有开/关，打开/闭合等状态的设备。

### 操作数：




### 结构文本：

| 操作数： | 类型： | 格式： | 说明： |
|---|---|---|---|
| D2SD标签 | DISCRETE_2STATE | 结构体 | D2SD结构体 |




### 功能块操作数：

| 操作数： | 类型： | 格式： | 说明： |
|---|---|---|---|
| D2SD标签 | DISCRETE_2STATE | 结构体 | D2SD结构体 |

### DISCRETE_2STATE结构体

| 输入参数： | 数据类型： | 说明： |
|---|---|---|
| EnableIn | BOOL | **功能块**：<br>如果该位清零，则指令不执行且输出不更新。<br>如果置位，指令执行。<br>缺省状态为置位。<br>**结构文本**：<br>无影响。指令执行。 |
| ProgCommand | BOOL | 在程序控制方式中用于确定设备的命令状态。置位时，设备被设置为1状态；清零时，设备被设置为0状态。<br>缺省状态为清零。 |
| Oper0Req | BOOL | 请求操作员状态0。在操作员界面控制时，设置该位使设备处于0状态。<br>缺省状态为清零 |
| Oper1Req | BOOL | 请求操作状态1。在操作员界面控制时，设置该位使设备处于1状态。<br>缺省状态为清零。 |
| State0Perm | BOOL | 允许状态0。除非在手动或自动方式下，此输入必须置位，设备才能进入0状态。如设备已处于0状态，则此位不影响。<br>缺省状态为置位。 |

| 输入参数： | 数据类型： | 说明： |
|---|---|---|
| State1Perm | BOOL | 允许状态1 。除非在手动或自动方式下，此输入必须置位，设备才能进入状态1。如设备已处于状态1，则该位不影响。<br>缺省状态为置位。 |
| FB0 | BOOL | D2SD指令第一可用反馈通道。<br>缺省状态为清零。 |
| FB1 | BOOL | D2SD指令第二可用反馈通道。<br>缺省状态为清零。 |
| HandFB | BOOL | 手动反馈输入。该输入来自手动/停止/自动区域，显示设备的状态请求。置位时，请求设备进入1状态；清零时，请求设备进入0状态。<br>缺省状态为清零。 |
| FaultTime | REAL | 故障计时值。配置允许设备转换到新命令状态的时间值(以秒计)。置FaultTime = 0以禁止故障计数。该参数数值无效时，指令设定其值为零，并置位状态字中的相应位。<br>有效值=大于等于0.0的浮点数<br>缺省值=0.0 |
| FaultAlarmLatch | BOOL | 故障报警锁存输入。该输入置位且出现故障报警时，锁存故障报警，置位FaultAlmUnlatch或清零FaultAlarmLatch可清除故障报警锁存。<br>缺省状态为清零 |
| FaultAlmUnLatch | BOOL | 故障报警解锁输入。故障报警锁存时，置位该位清除锁存。指令清零该输入位。<br>缺省状态为清零。 |
| OverrideOnInit | BOOL | 自动初始化请求。如果该位置位，则在指令首次扫描期间，2态设备处于操作员控制状态，Override置位，Hand清零。如果ProgHandReq置位，则Override清零，Hand置位。<br>缺省状态值为清零。 |
| OverrideOnFault | BOOL | 自动故障请求。当故障报警时设备进入Override方式和OverrideState时，OverrideOnFault置位。当故障报警清除时，2-态设备进入操作员控制状态。<br>缺省状态值为清零。 |
| OutReverse | BOOL | 缺省输出状态取反。当命令为状态0时，缺省输出状态清零，命令为状态1时输出置位。当OutReverse置位时，命令为状态0时输出置位，命令为状态1时输出清零。<br>缺省状态为清零 |
| OverrideState | BOOL | Override状态输入。 在Override方式下配置该位以指定设备的状态。 当设备转化的状态1时置位指示，当设备转化到状态0时，清零指示。<br>缺省状态为清零 |
| FB0State0 | BOOL | 0通道反馈状态0输入。配置设备为状态0时FB0状态。<br>缺省状态为清零 |
| FB0State1 | BOOL | 0通道反馈状态1输入。配置设备为状态1时FB0状态。<br>缺省状态为清零 |
| FB1State0 | BOOL | 1通道反馈状态0输入。配置设备为状态0时FB1状态。<br>缺省状态为清零 |
| FB1State1 | BOOL | 1通道反馈状态1输入。配置设备为状态1时FB1状态。<br>缺省状态为清零 |

| 输入参数 | 数据类型 | 说明 |
|---|---|---|
| ProgProgReq | BOOL | 程序请求进入程序控制方式。用户程序置位该位请求程序控制。当ProgOperReq置位时该请求被忽略。保持置位对ProgOperReq 清零可锁定指令进入程序控制状态。<br>缺省状态为清零。 |
| ProgOperReq | BOOL | 程序请求进入操作员控制方式。用户程序置位该位请求操作员控制。保持该位可将指令锁定为操作员控制。<br>缺省状态为清零 |
| ProgOverrideReq | BOOL | 程序请求进入自动方式。用户程序置位该位请求设备进入自动方式。如果ProgHandReq 被置位则忽略该请求。<br>缺省状态为清零 |
| ProgHandReq | BOOL | 程序请求进入手动控制方式。 用户程序置位该位请求设备进入手动方式。<br>缺省状态为清零 |
| OperProgReq | BOOL | 操作员请求进入程序控制方式。通过操作员界面请求程序控制。指令清零该输入。<br>缺省状态为清零 |
| OperOperReq | BOOL | 操作员进入操作员控制方式请求。通过操作员界面请求操作员控制。指令清零该输入。<br>缺省状态为清零 |
| ProgValueReset | BOOL | 复位程序控制值。置位时，每次执行指令行时所有程序请求全部清零。<br>缺省状态为清零 |

| 输出参数： | 数据类型： | 说明： |
|---|---|---|
| EnableOut | BOOL | 输出使能。 |
| Out | BOOL | 2态指令的输出。 |
| Device0State | BOOL | 设备状态0输出。当命令为状态0且反馈指示确实为状态0时该位置位。 |
| Device1State | BOOL | 设备状态1输出。当命令为状态1且反馈指示确实为状态1时该位置位。 |
| CommandStatus | BOOL | 命令状态输出。命令为状态1时置位，命令为状态0时清零。 |
| FaultAlarm | BOOL | 故障报警输出。当命令为新状态时，已经达到出错次数限制且没有完成新状态转换的反馈指示，则该位置位。已经达到命令状态后，反馈突然指示设备不处于命令状态时，该位也置位。 |
| ModeAlarm | BOOL | 方式报警输出。如果设备处于操作员控制状态，而程序命令改变到与当前操作员命令不同的状态时，则该位置位。该报警用于表明设备处于操作员控制状态。 |
| ProgOper | BOOL | 程序/操作员控制指示。程序控制置位，操作员控制清零 |
| Override | BOOL | 自动方式。当设备处于自动方式时该位置位。 |
| Hand | BOOL | 手动方式。当设备处于手动方式状态时该位置位 |
| Status | DINT | 功能块的状态字。 |

| 输出参数： | 数据类型： | 说明： |
|---|---|---|
| InstructFault(Status.0) | BOOL | 指令检测以下的运行错误。该错误不使控制器产生次要或主要错误。检测其它状态位以确定发生故障的类型。 |
| FaultTimeInv (Status.1) | BOOL | 无效的错误次数值。该指令设置FaultTime = 0。 |
| OperReqInv (Status.2) | BOOL | 两个操作员状态请求位置位。 |

**说明：** D2SD指令控制2态设备，例如开/关、打开/闭合等。典型设备包括电动机、泵、电磁阀。

## 监控D2SD指令

D2SD指令具有操作员界面。需要了解更多信息请参阅附录：功能块界面控制。

**算术状态标志：** 不影响算术状态标志。

**故障条件：** 无

**执行：**

| 条件： | 功能块动作： | 结构文本动作： |
|---|---|---|
| 预扫描 | 无动作。 | 无动作。 |
| 指令首次扫描 | 清零出错计时。<br>清零方式报警。<br>清零所有操作员请求输入。<br>如果ProgValueReset置位，清零所有程序请求输入。<br>当OverrideOnInit置位，则ProgOper清零(操作员控制)<br>如果ProgHandReq清零且OverrideOnInit置位，则HA置位ND清零，Override置位(自动方式)。<br>如果ProgHandReq置位，则HAND置位，Override清零(手动方式) | |
| 指令首次执行 | ProgOper和CommandStatus清零。 | ProgOper和CommandStatus清零。 |
| 输入使能(EnableIn)清零 | 输出使能(EnableOut)清零。 | 不影响。 |
| 输入(EnableIn)置位 | 执行指令。<br>输出使能(EnableOut)置位 | 输入一直置位。<br>执行指令。 |
| 后期扫描 | 无动作。 | 无动作。 |

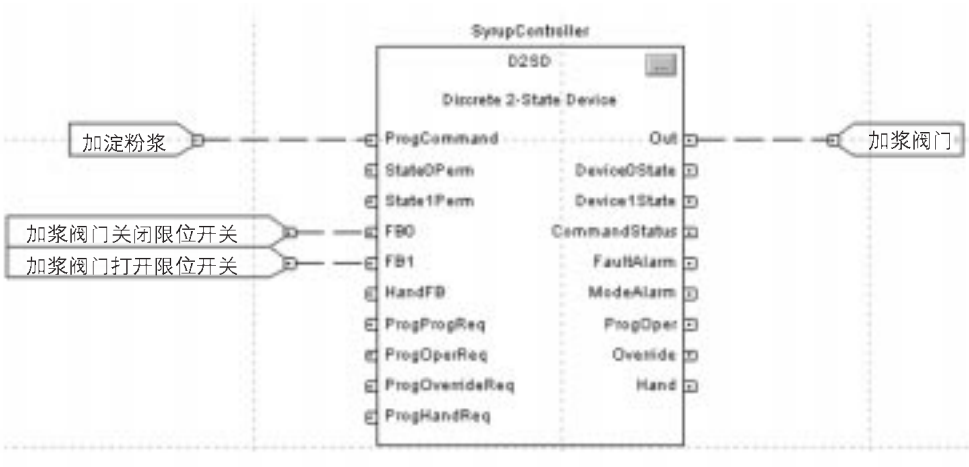
**实例：** D2SD指令典型应用于控制开/关、打开/闭合的设备，例如泵或者电磁阀。在此例中，D2SD指令用于控制一个电磁阀，向一批容器中加淀粉浆。D2SD指令用于程序控制中，当AddSyrup输入置位时，阀门打开。如果需要操作员也可通过操作员控制方式控制阀门开或关。此例中，电磁阀具有限位开关，作为阀门完全闭合或打开的指示。这两个开关连接到FB0和FB1反馈输入。这使D2SD指令在给定的故障时间内，如果电磁阀没有到达所指定的位置时产生一个故障报警。

### 结构文本

```
SyrupController.ProgCommand := AddSyrup;
SyrupController.FB0 := SyrupValveClosedLimitSwitch;
SyrupController.FB1 := SyrupValveOpenedLimitSwitch;
D2SD(SyrupController);

SyrupValve := SyrupController.Out
```

### 功能块



## 程序和操作员控制切换

以下图表演示了D2SD指令如何在程序控制与操作员控制之间切换




(1)当ProgOperReq置位时，指令仍处于操作员控制方式

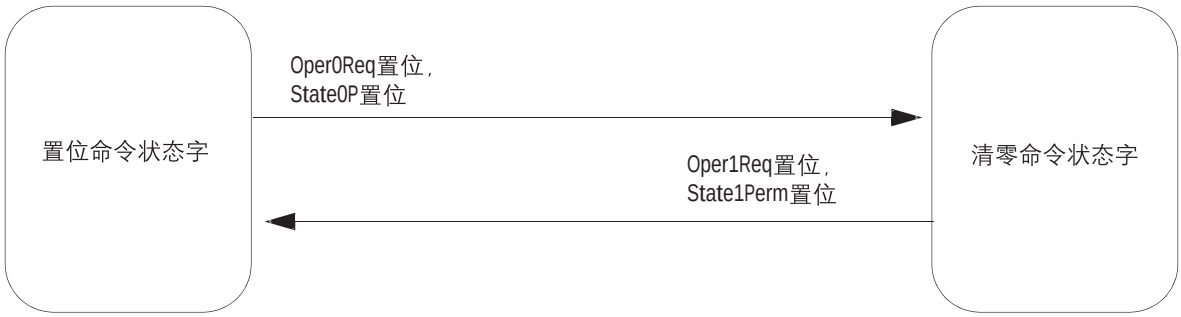
## 程序控制中的命令状态

以下图表阐明了D2SD指令在程序控制方式下如何工作。



操作员控制中的命令状态

以下图表阐明了D2SD指令在操作员控制方式下如何工作。




如果Oper0Req和Oper1Req都置位：

- 指令将相应的状态位置位
- 如果Override和Hand都为清零，该指令保持原有状态

每次指令执行后，指令：

- 清零所有操作请求输入
- 如果ProgValueReset置位，则所有程序请求输入被清零

## 手动或自动方式

以下表格说明了D2SD指令如何确定手动或自动操作方式

| ProgHandReq: | ProgOverrideReq: | FaultAlarm and OverrideOnFault: | 说明： |
|---|---|---|---|
| 置位 | 任一 | 任一 | 手动方式<br>Hand置位<br>Override清零 |
| 清零 | 置位 | 任一 | 自动方式<br>Hand 清零<br>Override 置位 |
| 清零 | 任一 | 置位 | 自动方式<br>Hand 清零<br>Override 置位 |

当指令为自动方式，CommandStatus = OverrideState

当指令为手动方式，CommandStatus = HandFB

## 输出状态

D2SD指令的输出状态是基于以下命令状态的。

| 命令状态字： | 输出状态： |
|---|---|
| 清零 | 如果OutReverse清零，则输出清零<br>如果OutReverse置位，则输出置位 |
| 置位 | 如果OutReverse清零，则输出置位<br>如果OutReverse置位，则输出清零 |
| 清零且<br>FB0 = FB0State0且<br>FB1 = FB1State0 | 故障计时器停止并且清零<br>Device0State置位 |
| 置位且<br>FB0 = FB0State1且<br>FB1 = FB1State1 | 故障计时器停止并且清零<br>Device1State置位 |

### 故障报警条件

D2SD指令检测以下故障报警条件：

| 故障报警条件源于： | 规则： |
|---|---|
| 设备被命令改变状态，但在故障计时时间内反馈指示设备没有到达预定状态。 | 在CommandStatus n ≠ CommandStatus n-1时，启动故障计时器。<br>当出错计时完成且FaultTime > 0.0时，置位FaultAlarm。 |
| 设备在没有被命令的情况下，意外地离开某个状态(根据反馈指示)。 | 置位FaultAlarm，当故障计时器不计时且满足以下条件之一时：<br>CommandStatus被清零且Device0State被清零；<br>CommandStatus被置位且Device1State被清零。 |

出现以下条件时故障报警(FaultAlarm)被清除：

- CommandStatus被清零且Device0State被置位
- CommandStatus被置位且Device1State被置位
- FaultTime ≤ 0

故障报警锁存(FaultAlarmLatch)置位时，故障报警(FaultAlarm)不能被清除，除非故障报警解锁(FaultAlmUnlatch)被置位且当前无故障状态。

## 方式报警条件

方式报警提醒操作员有设备处于操作员控制方式。只有在操作员控制方式下，程序试图从操作员控制状态改变设备的状态时才产生方式报警。当操作员将设备置为操作员方式并改变状态时，不产生方式报警。D2SD指令采用这些规则检测方式报警条件。

| 方式报警： | 条件： |
| --- | --- |
| 置位 | ProgCommandn ≠ ProgCommandn-1 且 ProgCommandn ≠ CommandStatus |
| 清零 | ProgCommand = CommandStatus 或设备处于自动、手动或程序控制方式。 |

## 离散量3态设备 (D3SD)

D3SD指令控制具有三态的设备，例如快/慢/停,正转/停止/反转等。

### 操作数：



```
D3SD(D3SD_tag);
```

### 结构文本：

| 操作数： | 类型 | 格式 | 说明 |
|---|---|---|---|
| D3SD标签 | DISCRETE_3STATE | 结构体 | D3SD结构体 |



### 功能块：




| 操作数： | 类型 | 格式 | 说明 |
|---|---|---|---|
| D3SD块标签 | DISCRETE_3STATE | 结构体 | D3SD结构体 |

### DISCRETE_3STATE结构体：

| 输入参数： | 数据类型 | 说明 |
|---|---|---|
| EnableIn | BOOL | 功能块：<br>如果该位清零，则指令不执行输出不更新。<br>如果置位，指令执行。<br>缺省状态为置位。<br>结构文本：<br>无影响。指令执行。 |
| Prog0Command | BOOL | 程序状态0命令。当设备处于程序控制状态时，该输入位决定设备状态。置位时，设备被命令为状态0。<br>缺省状态为清零 |
| Prog1Command | BOOL | 程序状态1命令。当设备处于程序控制状态时，该输入位决定设备状态。置位时，设备被命令为状态1。<br>缺省状态为清零 |
| Prog2Command | BOOL | 程序状态2命令。当设备处于程序控制状态时，该输入位决定设备状态。置位时，设备被命令为状态2。<br>缺省状态为清零 |

| 输入参数: | 数据类型 | 说明 |
|---|---|---|
| Oper0Req | BOOL | 操作员状态0请求。在操作员界面中控制时，设置该位使设备处于状态0。<br>缺省状态为清零 |
| Oper1Req | BOOL | 操作员状态1请求。在操作员界面中控制时，设置该位使设备处于状态1。<br>缺省状态为清零 |
| Oper2Req | BOOL | 操作状态2请求。在操作员界面中控制时，设置该位使设备处于状态2。<br>缺省状态为清零 |
| State0Perm | BOOL | 状态0允许。除非在手动或自动方式下，此输入必须置位，设备才能进入状态0。如设备已处于状态0，则此位不影响。<br>缺省状态为置位 |
| State1Perm | BOOL | 状态1允许。除非在手动或自动方式下，此输入必须置位，设备才能进入状态1。如设备已处于状态1，则此位不影响。<br>缺省状态为置位 |
| State2Perm | BOOL | 状态2允许。除非在手动或自动方式下，此输入必须置位，设备才能进入状态2。如设备已处于状态2，则此位不影响。<br>缺省状态为置位 |
| FB0 | BOOL | D2SD指令第一反馈通道。<br>缺省为清零。 |
| FB1 | BOOL | D2SD指令第二反馈通道。<br>缺省为清零。 |
| FB2 | BOOL | D2SD指令第三反馈通道。<br>缺省为清零。 |
| FB3 | BOOL | D2SD指令第四反馈通道。<br>缺省为清零。 |
| HandFB0 | BOOL | 手动反馈状态0。该输入来自手动/停止/自动旋钮，显示设备的状态请求。设备请求进入状态0时，置位指示。请求设备进入其它状态时，清除指示位。<br>缺省状态为清零 |
| HandFB1 | BOOL | 手动反馈状态1。该输入来自手动/停止/自动旋钮，显示设备的状态请求。设备请求进入状态1时，置位指示。设备请求进入其它状态时，清除指示位。<br>缺省状态为清零 |
| HandFB2 | BOOL | 手动反馈状态2。该输入来自手动/停止/自动旋钮，显示设备的状态请求。设备请求进入状态2时，置位指示。设备请求进入其它状态时，清除指示位。<br>缺省状态为清零 |
| FaultTime | REAL | 故障次数值。配置允许设备每秒钟转换到新命令状态的次数。设置FaultTime = 0禁止故障计数。该参数数值无效时，指令设定其值为零，并置相应状态位。<br>有效值=大于等于0.0的浮点数<br>缺省值=0.0 |
| FaultAlarmLatch | BOOL | 故障报警锁存输入。该输入置位且出现故障报警时，锁存故障报警，置位FaultAlmUnlatch或清零FaultAlarmLatch可清除故障报警锁存。<br>缺省状态为清零 |

| 输入参数： | 数据类型： | 说明： |
|---|---|---|
| FaultAlmUnLatch | BOOL | 故障报警解锁输入。故障报警锁存时设置该位清除锁存。产生清除故障报警时，指令清零该输入位。<br>缺省状态为清零 |
| OverrideOnInit | BOOL | 自动初始化请求。如果此位置位，则在指令首次扫描期间，指令处于操作员控制状态，Override置位，Hand清零。如果ProgHandReq置位，则Override清零，Hand置位。<br>缺省状态为清零。 |
| OverrideOnFault | BOOL | 自动故障请求。当故障报警时设备进入OverrideState和Override方式时，OverrideOnFault置位。当故障报警清除时，该设备进入操作员控制状态。<br>缺省状态值为清零。 |
| Out0State0 | BOOL | 输出0状态0输入。当设备处于状态0时，该输入值决定Output0的值<br>缺省状态为清零 |
| Out0State1 | BOOL | 输出0状态1输入。当设备处于状态1时，该输入值决定Output0的值。<br>缺省状态为清零 |
| Out0State2 | BOOL | 输出0状态2输入。当设备处于状态2时，该输入值决定Output0的值。<br>缺省状态为清零 |
| Out1State0 | BOOL | 输出1状态0输入。当设备处于状态0时，该输入值决定Output1的值。<br>缺省状态为清零 |
| Out1State1 | BOOL | 输出1状态1输入。当设备处于状态1时，该输入值决定Output1的值。<br>缺省状态为清零 |
| Out1State2 | BOOL | 输出1状态2输入。当设备处于状态2时，该输入值决定Output1的值。<br>缺省状态为清零 |
| Out2State0 | BOOL | 输出1状态0输入。当设备处于状态0时，该输入值决定Output1的值。<br>缺省状态为清零 |
| Out2State1 | BOOL | 输出1状态1输入。当设备处于状态1时，该输入值决定Output1的值。<br>缺省状态为清零 |
| Out2State2 | BOOL | 输出1状态2输入。当设备处于状态2时，该输入值决定Output1的值。<br>缺省状态为清零 |
| OverrideState | DINT | 自动状态输入。在自动方式，通过设置该位来指示设备状态。<br>**数值：** **指示：**<br>2 设备即将进入状态2<br>1 设备即将进入状态1<br>0 设备即将进入状态0<br>输入值无效时，置状态字的相应位，并防止指令进入自动方式。<br>有效值=0到2<br>缺省值=0 |

| 输入参数： | 数据类型： | 说明： |
|---|---|---|
| FB0State0 | BOOL | 状态0反馈0输入。当设备处于状态0时，该输入值决定了FB0的期望值。<br>缺省状态为清零 |
| FB0State1 | BOOL | 状态1反馈0输入。当设备处于状态1时，该输入值决定了FB0的期望值。<br>缺省状态为清零 |
| FB0State2 | BOOL | 状态2反馈0输入。当设备处于状态2时，该输入值决定了FB0的期望值。<br>缺省状态为清零 |
| FB1State0 | BOOL | 状态0反馈1输入。当设备处于状态0时，该输入值决定了FB1的期望值。<br>缺省状态为清零 |
| FB1State1 | BOOL | 状态1反馈1输入。当设备处于状态1时，该输入值决定了FB1的期望值。<br>缺省状态为清零 |
| FB1State2 | BOOL | 状态2反馈1输入。当设备处于状态2时，该输入值决定了FB1的期望值。<br>缺省状态值为清零 |
| FB2State0 | BOOL | 状态0反馈2输入。当设备处于状态0时，该输入值决定了FB2的期望值。<br>缺省状态为清零 |
| FB2State1 | BOOL | 状态1反馈2输入。当设备处于状态1时，该输入值决定了FB2的期望值。<br>缺省状态为清零 |
| FB2State2 | BOOL | 状态2反馈2输入。当设备处于状态2时，该输入值决定了FB2的期望值。<br>缺省状态为清零 |
| FB3State0 | BOOL | 状态0反馈3输入。当设备处于状态0时，该输入值决定了FB3的期望值。<br>缺省状态为清零 |
| FB3State1 | BOOL | 状态1反馈3输入。当设备处于状态1时，该输入值决定了FB3的期望值。<br>缺省状态为清零 |
| FB3State2 | BOOL | 状态2反馈3输入。当设备处于状态2时，该输入值决定了FB3的期望值。<br>缺省状态为清零 |
| ProgProgReq | BOOL | 程序程序请求。用户程序置位该位请求程序控制。 当ProgOperReq置位时该请求被忽略。保持置位对ProgOperReq 清零可使指令进入程序控制状态。<br>缺省状态为清零 |
| ProgOperReq | BOOL | 程序操作员请求。用户程序置位该位请求操作员控制。保持该位置位可使指令进入操作员控制状态。<br>缺省状态为清零 |
| ProgOverrideReq | BOOL | 程序自动请求。用户程序置位该位请求设备进入自动控制。如果ProgHandReq 被置位则忽略该请求。<br>缺省状态为清零 |
| ProgHandReq | BOOL | 程序手动请求。 用户程序置位该位请求设备进入手动方式。<br>缺省状态为清零 |

| 输入参数： | 数据类型： | 说明： |
|---|---|---|
| OperProgReq | BOOL | 操作员界面程序请求。通过操作员界面请求程序控制。指令清零该输入。缺省状态为清零 |
| OperOperReq | BOOL | 操作员操作请求。通过操作员界面请求操作员控制。指令清零该输入。缺省状态为清零 |
| ProgValueReset | BOOL | 复位程序控制值。置位时，每次执行指令行时所有程序请求全部清零。缺省状态为清零 |

| 输出参数： | 数据类型： | 说明： |
|---|---|---|
| EnableOut | BOOL | 输出使能。 |
| Out0 | BOOL | 指令第一输出。 |
| Out1 | BOOL | 指令第二输出。 |
| Out2 | BOOL | 指令第三输出。 |
| Device0State | BOOL | 设备状态0输出。当命令为状态0且反馈指示确实为状态0时，该位置位。 |
| Device1State | BOOL | 设备状态1输出。当命令为状态1且反馈指示确实为状态1时，该位置位。 |
| Device2State | BOOL | 设备状态2输出。当命令为状态2且反馈指示确实为状态2时，该位置位。 |
| Command0Status | BOOL | 设备命令状态0输出。设备命令为状态0时置位，命令为其它状态时清零。 |
| Command1Status | BOOL | 设备命令状态1输出。设备命令为状态1时置位，命令为其它状态时清零。 |
| Command2Status | BOOL | 设备命令状态2输出。设备命令为状态2时置位，命令为其它状态时清零。 |
| FaultAlarm | BOOL | 故障报警输出。当命令为新状态时，已经达到出错次数限制且没有完成新状态转换的反馈指示，则该位置位。已经达到命令状态后，反馈突然指示设备不处于命令状态时，该位也置位。 |
| ModeAlarm | BOOL | 方式报警输出。如果设备处于操作员控制状态，而程序命令改变到与当前操作员命令不同的状态时，则该位置位。该报警用意在于表明设备处于操作员控制状态。 |
| ProgOper | BOOL | 程序/操作员控制指示。程序控制置位，操作员控制清零 |
| Override | BOOL | 自动方式。当设备处于自动方式时该位置位。 |
| Hand | BOOL | 手动方式。当设备处于手动方式状态时该位置位。 |
| Status | DINT | 功能块状态字。 |
| InstructFault(Status.0) | BOOL | 指令检测到下面的执行错误之一。这不是控制器次要或主要错误。检查其他状态位以确定发生的情况。 |
| FaultTimeInv(Status.1) | BOOL | 无效的FaultTime值。指令设置FaultTime=0。 |

| 输出参数： | 数据类型： | 说明： |
|---|---|---|
| OverrideStateInv(Stattus.2) | BOOL | 自动值超出范围。 |
| ProgCommandInv(Status.3) | BOOL | 同时将多个程序状态命令位置1。 |
| OperReqInv(Status.4) | BOOL | 同时将多个操作员请求位置1。 |
| HandCommandInv(Status.5) | BOOL | 同时将多个手动状态请求位置1。 |

**说明：** D3SD指令用于控制的离散型设备具有三种可能状态例如快/慢/停、前进/停止/倒退等。典型的设备包括进料系统、可反转的电动机等。

### 监控D3SD指令

D3SD指令具有操作员面板。需要了解更多信息请参阅附录：功能块面板控制。

**算术状态标志：** 不影响算术状态标志。

**故障条件：** 无

**执行：**

| 条件: | 功能块动作: | 结构文本动作： |
|---|---|---|
| 预扫描 | 无动作。 | 无动作。 |
| 指令首次扫描 | 清零计时器。<br>清零方式报警。<br>清零所有操作员请求输入。<br>如果ProgValueReset置位，清零所有程序请求输入。<br>当OverrideOnInit置位，则ProgOper清零(操作员控制)<br>如果ProgHandReq清零且OverrideOnInit置位，则HA置位ND清零，Override置位(自动方式)。<br>如果ProgHandReq，则HAND置位，Override清零(手动方式) | |
| 指令首次执行 | ProgOper和CommandStatus清零。 | ProgOper和CommandStatus清零。 |
| 输入使能(EnableIn)清零 | 输出使能(EnableOut)清零。 | 不影响。 |
| 输入使能(EnableIn)置位 | 执行指令。<br>输出使能(EnableOut)置位 | 输入使能(EnableIn)一直置位。<br>指令执行。 |
| 后期扫描 | 无动作。 | 无动作。 |

**实例：** D3SD典型用法用于控制具有3种状态设备，例如高/低/关进料系统等。在此例中，D3SD指令控制具有一对电磁阀的进料系统给处理罐输送菜油。其中一个阀门在处理罐大口径进料管道上，另一个阀门在并行的小口径进料管道上。初次进油时，D3SD指令切换到快速进料状态(state2)，此时两个阀门同时开启。当油接近目标数量时，D3SD指令切换到慢速进料状态(state1)，此时大阀门关闭小阀门保持打开状态。当油达到目标量时，D3SD指令切换到关进料状态(state0)，两个阀门都关闭。

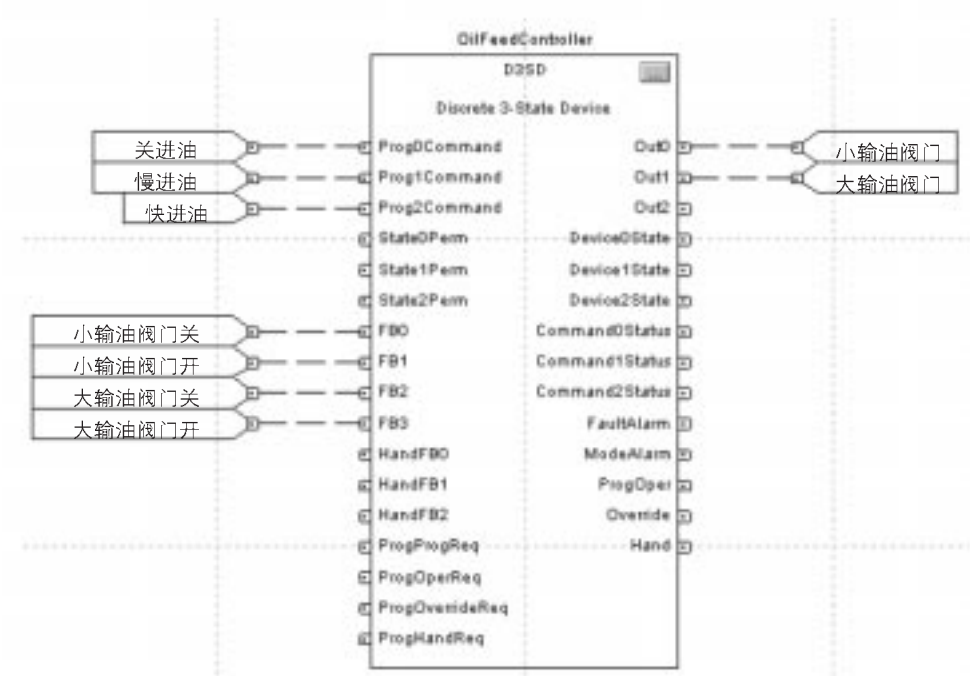
只要D3SD指令处于程序控制状态，阀门的开关取决于CloseOilFeed、SlowOilFeed和FastOilFeed的输入。必要时，操作员可对进料系统的进行操作员控制。此例中的电磁阀具有限位开关，指示阀门全开或全闭。这些开关连结到FB0, FB1, FB2及FB3反馈输入。这就使D3SD指令能在设定的故障计时时间内阀门没有达到命令的状态进行报警。

## 结构文本

```
OilFeedController.Prog0Command := CloseOilFeed;
OilFeedController.Prog1Command := SlowOilFeed;
OilFeedController.Prog2Command := FastOilFeed;
OilFeedController.FB0 := SmallOilValveClosed;
OilFeedController.FB1 := SmallOilValveOpened;
OilFeedController.FB2 := LargeOilValveClosed;
OilFeedController.FB3 := LargeOilValveOpened;
D3SD(OilFeedController);

SmallOilValve := OilFeedController.Out0;
LargeOilValve := OilFeedController.Out1;
```

## 功能块



程序和操作员控制切换

以下图表演示了D3SD指令如何在程序控制与操作员控制之间切换




(1)当ProgOperReq置位时，指令仍处于操作员控制方式

## 程序控制中的命令状态：

下表说明了D3SD指令在程序控制状态下如何工作：

| **Prog0 Command:** | **Prog1 Command:** | **Prog2 Command:** | **State0 Perm:** | **State1 Perm:** | **State2 Perm:** | **说明：** |
|---|---|---|---|---|---|---|
| 清零 | 清零 | 置位 | 任一 | 任一 | 置位 | Command0Status清零<br>Command1Status清零<br>Command2Status置位 |
| 清零 | 置位 | 清零 | 任一 | 置位 | 不影响 | Command0Status清零<br>Command1Status置位<br>Command2Status清零 |
| 置位 | 清零 | 清零 | 置位 | 任一 | 任一 | Command0Status置位<br>Command1Status清零<br>Command2Status清零 |

当有多个程序命令输入置位时：

- 该指令将状态字中的相应位置位
- 如果Override和Hand被清零，指令将保持以前的状态

### 操作员控制中的命令状态：

以下表格说明了D3SD指令在操作员控制状态下如何工作：

| Oper0Req: | Oper1Req: | Oper2Req: | State0 Perm: | State1 Perm: | State2 Perm: | 说明： |
|---|---|---|---|---|---|---|
| 清零 | 清零 | 置位 | 任一 | 任一 | 置位 | Command0Status清零<br>Command1Status清零<br>Command2Status置位 |
| 清零 | 置位 | 清零 | 任一 | 置位 | 任一 | Command0Status清零<br>Command1Status置位<br>Command2Status清零 |
| 置位 | 清零 | 清零 | 置位 | 任一 | 任一 | Command0Status置位<br>Command1Status清零<br>Command2Status清零 |

当有多个操作员命令输入置位时：

- 该指令将状态字中的相应位置位
- 如果Override和Hand被清零，指令将保持以前状态。

在所有指令执行后，该指令：

- 清除所有操作员请求的输入。
- 如果ProgValueReset被置位，则清除所有程序请求输入。

### 手动或自动方式

以下表格说明了D3SD指令如何判断工作在手动方式(Hand)或自动方式(Override)

| ProgHandReq: | ProgOverrideReq: | FaultAlarm 和 OverrideOnFault: | 说明： |
|---|---|---|---|
| 置位 | 不影响 | 不影响 | 手动方式<br>Hand置位<br>Override清零 |
| 清零 | 置位 | 不影响 | 自动方式<br>Hand清零<br>Override置位 |
| 清零 | 不影响 | 置位 | 自动方式<br>Hand清零<br>Override置位 |

当Override置位时，其优先级高于程序控制和操作员控制。下表说明了自动方式下如何影响命令状态。

| **自动方式：** | **OverrideState：** | **说明：** |
|---|---|---|
| 置位 | 2 | Command0Status清零<br>Command1Status清零<br>Command2Status置位 |
| 置位 | 1 | Command0Status清零<br>Command1Status置位<br>Command2Status清零 |
| 置位 | 0 | Command0Status置位<br>Command1Status清零<br>Command2Status清零 |

如果OverrideState无效，该指令将状态字中相应位置位且不进入自动状态。

当Hand 置位时，其优先级高于程序控制和操作员控制。以下表格说明了手动方式如何影响命令状态。

| **Hand：** | **HandFB0：** | **HandFB1：** | **HandFB2：** | **说明：** |
|---|---|---|---|---|
| 置位 | 清零 | 清零 | 置位 | Command0Status清零<br>Command1Status清零<br>Command2Status置位 |
| 置位 | 清零 | 置位 | 清零 | Command0Status清零<br>Command1Status置位<br>Command2Status清零 |
| 置位 | 置位 | 清零 | 清零 | Command0Status置位<br>Command1Status清零<br>Command2Status清零 |

如果多个手动反馈(HandFB)输入被置位，则指令将状态字中相应位置位。如果Hand置位，则指令保持前一状态。

## 输出状态

D3SD指令的输出状态基于命令状态字的状态。

| 命令状态字： | 输出状态： |
|---|---|
| Command0Status置位 | Out0 = Out0State0<br>Out1 = Out1State0<br>Out2 = Out2State0 |
| Command0Status置位**且**<br>FB0 = FB0State0**且**<br>FB1 = FB1State0**且**<br>FB2 = FB2State0**且**<br>FB3 = FB3State0 | 故障计时器停止并清零<br>Device0State置位 |
| Command1Status置位 | Out0 = Out0State1<br>Out1 = Out1State1<br>Out2 = Out2State1 |
| Command1Status置位**且**<br>FB0 = FB0State1**且**<br>FB1 = FB1State1**且**<br>FB2 = FB2State1**且**<br>FB3 = FB3State1 | 故障计时器停止并清零<br>Device1State置位 |
| Command2Status置位 | Out0 = Out0State2<br>Out1 = Out1State2<br>Out2 = Out2State2 |
| Command2Status置位且<br>FB0 = FB0State2**且**<br>FB1 = FB1State2**且**<br>FB2 = FB2State2**且**<br>FB3 = FB3State2 | 故障计时器停止并清零<br>Device2State置位 |

## 故障报警条件

D3SD监测以下故障报警条件。

| 产生故障报警的原因： | 规则： |
|---|---|
| 设备被命令改变状态，但在故障计时时间内反馈指示出设备没有到达预定状态。 | 当$Command0Status_n \neq Command0Status_{n-1}$或$Command1Status_n \neq Command1Status_{n-1}$或$Command2Status_n \neq Command2Status_{n-1}$时启动故障计时器。 |
| 设备在不被命令的情况下，<br>意外地离开某个状态(根据反馈指示)。 | 当故障计时完成且FaultTime > 0.0时，置位FaultAlarm。<br>置位FaultAlarm，当故障计时器不计时且满足以下条件之一时：<br>Command0Status被置位且Device0State被清零；<br>Command1Status被置位且Device1State被清零；<br>Command2Status被置位且Device2State被清零。 |

如果无故障出现，FaultAlarm在遇到以下条件之一时清零：

- Command0Status被置位且Device0State被置位
- Command1Status被置位且Device1State被置位
- Command2Status被置位且Device2State被置位
- FaultTime ≤0

当故障报警锁存(FaultAlarmLatch)置位时，故障报警(FaultAlarm)不能被清除，除非故障报警解锁(FaultAlmUnlatch)被置位且当前无故障状态。

## 方式报警条件

方式报警提醒操作员有设备处于操作员控制方式。仅在操作员控制方式下，程序试图从操作员命令状态下改变设备状态时才产生方式报警。当操作员将设备置为操作员方式并改变方式时，不产生方式报警。D3SD指令采用以下规则检测方式报警条件。

| 方式报警： | 条件： |
|---|---|
| 置位 | Prog2Command ≠Prog2Command n-1**且**<br>Prog2Command n ≠ Command2Status或<br><br>Prog1Command ≠Prog1Command n-1**且**<br>Prog1Command n ≠ Command1Status**或**<br><br>Prog0Command ≠Prog0Command n-1**且**<br>Prog0Command ≠ Command0Status |
| 清零 | Prog2Command = Command2Status**且**<br>Prog2Command = Command2Status**且**<br>Prog2Command = Command2Status**或**<br>设备处于自动、手动或程序控制方式 |

# 死区时间(DEDT)

DEDT指令实现单个输入信号延时功能。用户可选择死区时间的长度。

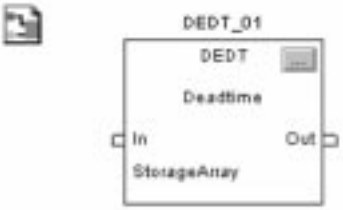
## 操作数：

### 结构文本：

```
DEDT(DEDT_tag,storage);
```

| 操作数 | 类型： | 格式： | 说明： |
|---|---|---|---|
| DEDT标签 | DEADTIME | 结构体 | DEDT结构体 |
| 存储 | REAL | 数组 | 死区时间缓冲区 |

### 功能块：




| 操作数 | 类型： | 格式： | 说明： |
|---|---|---|---|
| DEDT标签 | DEADTIME | 结构体 | DEDT结构体 |
| 存储 | REAL | 数组 | 死区时间缓冲区 |

### DEADTIME结构体

| 输入参数： | 数据类型： | 说明： |
|---|---|---|
| EnableIn | BOOL | 功能块：<br>如果清零，则指令不执行且输出不更新。如果置位，指令执行。<br>缺省状态为置位。<br>结构文本：<br>无影响。指令一直执行。 |
| In | REAL | 模拟量信号输入<br>有效值＝任一浮点数<br>缺省值＝0.0 |
| InFault | BOOL | 输入信号错误指示。如果输入值为模拟量输入，则InFault由模拟量输入的故障状态来控制。<br>如果InFault置位，则指示输入信号出错，指令将置位状态字的相应位，控制算法不执行，输出保持。<br>缺省状态为清零<br>清零＝正常状态 |
| Deadtime | REAL | 指令输入的死区时间。输入以秒为单位的死区时间，如果该值无效，则指令设定该值为零并置位状态字的相应位。<br>有效值＝0到(StorageArray尺寸*DeltaT)<br>缺省值＝0.0 |
| Gain | REAL | 指令输入的增益因子。输入值乘以该数值。这允许进行过程增益仿真。<br>有效值＝浮点数<br>缺省值＝1.0 |
| Bias | REAL | 指令输入的偏移量。输入乘以增益值加上该值。这允许进行环境条件仿真。<br>有效值＝浮点数<br>缺省值＝0.0 |

| 输入参数： | 数据类型： | 说明： |
|---|---|---|
| TimingMode | DINT | 选择定时执行方式。<br>有效值： 说明：<br>0 周期方式<br>1 过采样方式<br>2 实时采样方式<br>有效值=0到2<br>缺省值=0<br>需要了解更多与定时方式有关的信息，请参阅附录：功能块属性。 |
| OversampleDT | REAL | 过采样方式(oversample mode)的执行时间<br>有效值=0到4194.303 秒<br>缺省值=0 |
| RTSTime | DINT | 实时采样方式下模块的更新周期。<br>有效值=1到32,767ms<br>缺省值=1 |
| RTSTimeStamp | DINT | 实时采样方式下模块的时间戳值。<br>有效值=0到32,767ms<br>缺省值=0 |

| 输出参数： | 数据类型： | 说明： |
|---|---|---|
| EnableOut | BOOL | 输出使能。 |
| Out | REAL | 死区算法计算出的输出结果。输出结果将影响算术状态标志。 |
| DeltaT | REAL | 两次更新之间经过的时间。该时间间隔以秒为单位，用于控制算法计算过程量输出值。 |
| Status | DINT | 功能块的状态字。 |
| InstructFault(Status.0) | BOOL | 指令检测以下运行错误。该错误不使控制器产生次要或主要错误。检测其它状态位以确定所发生故障类型。 |
| InFaulted(Status.1) | BOOL | 输入出错。 |
| DeadtimeInv(Status.2) | BOOL | 死区时间值无效。 |
| TimingMode (Status.27) | BOOL | 定时方式值无效。<br>需要了解更多与定时方式有关的信息，请参阅附录：功能块属性。 |
| RTSMissed(Status.28) | BOOL | 只用于实时采样方式。当ABS| DeltaT - RTSTime | >1(0.001秒)时，该位被置位。 |
| RTSTimeInv(Status.29) | BOOL | 实时采样时间值无效。 |
| RTSTimeStampInv(Status.30) | BOOL | 实时采样时间标签值无效。 |
| DeltaTInv(Status.31) | BOOL | DeltaT值无效。 |

**说明：** DEDT指令用一个数据缓冲区来存储延时数据。这样就可得到任意长度的死区时间。DEST指令是为扫描时间是常数的任务而设计的。

使用DEDT指令时，必须创建一个存储数组，用来存储死区时间以保持(In × Gain) +Bias的采样值。存储数组应该足够容纳期望的最大的死区时间的数据，公式如下：

所需StorageArray Size = Maximum Deadtime (secs) / DeltaT (secs)

## 死区时间缓冲区的维护

在运行期间，指令总是检测有效的死区时间。死区时间必须介于0.0与(StorageArray Size ×DeltaT)之间。

如果死区时间无效，则指令将置位状态字中的相应位并且设定Out = (In × Gain) + Bias。

死区时间缓冲区是一个先入先出缓冲区。每次死区时间算法执行时，死区时间缓冲区中最早的数值就被传送到Out。剩余的数据向下移位，当前((In x Gain) + Bias)的值被送到死区时间缓冲区的首端。新数据被放在死区时间缓冲区中，经过Deadtime时间后才出现在Out上。

执行预定的延时所需的数组元素数量是由Deadtime除以DeltaT得到的。如果Deadtime不被DeltaT整除，则数组元素数量和延时时间都将被向上取整到最接近的DeltaT。例如：为了求出给定延时时间Deadtime = 4.25s和DeltaT = 0.50s的数组元素数量：

4.25s / 0.50s = 8.5
向上取整，需要9个数组元素

此例中其确切的延时时间是：

数组元素数量 ×DeltaT＝预定的延时或9 x 0.5s = 4.5s

一旦死区时间(Deadtime)或DeltaT数值在移出缓冲区的时侯改变，运行时间(Runtime)也改变。预定延时所需的元素数量可以被增加或减少。在死区时间缓冲区维护之前，需进行以下更新：

- 如果所需的元素数量增加，新增的缓冲区元素被装入当前死区时间缓冲区中最早的数据。
- 如果所需的元素数量减少，则当前死区时间缓冲区中最早的数据被删除。

### 故障过渡指令动作

当InFault置位时(出错)，指令延缓执行，保持最近的输出，并置位状态字中的相应位。

当InFault从置位状态转换到清零状态，指令设置Out及死区时间缓冲区的所有值都等于In x Gain + Bias。

**算术状态标志：** 该输出影响算术状态标志。

**故障条件：** 无

**执行：**

| 条件: | 功能块动作: | 结构文本动作： |
|---|---|---|
| 预扫描 | 无动作。 | 无动作。 |
| 指令首次扫描 | 如果InFault清零，则Out和所有缓冲区的数据将被设置为(In x Gain + Bias)。 | |
| 指令首次执行 | 如果InFault清零，则Out和所有缓冲区的数据将被设置为(In x Gain + Bias)。 | |
| 输入使能(EnableIn)清零 | 输出使能(EnableOut)清零。 | 不影响 |
| 输入使能(EnableIn)置位 | 执行指令。<br>输出使能(EnableOut)置位 | 输入使能(EnableIn)一直置位。<br>指令执行。 |
| 后期扫描 | 无动作 | 无动作 |

**实例：**在此例中，DEDT指令在过程仿真中模仿一个死区时间延时。PIDE指令的输出经过死区时间延时和一个一阶滞后环节作为仿真的过程量。DEDT_01数组是一个具有100个元素的实型数组，最大支持100个采样点的死区时间。例如，该例程每100ms执行一次，则该数组最大支持10秒的死区时间。

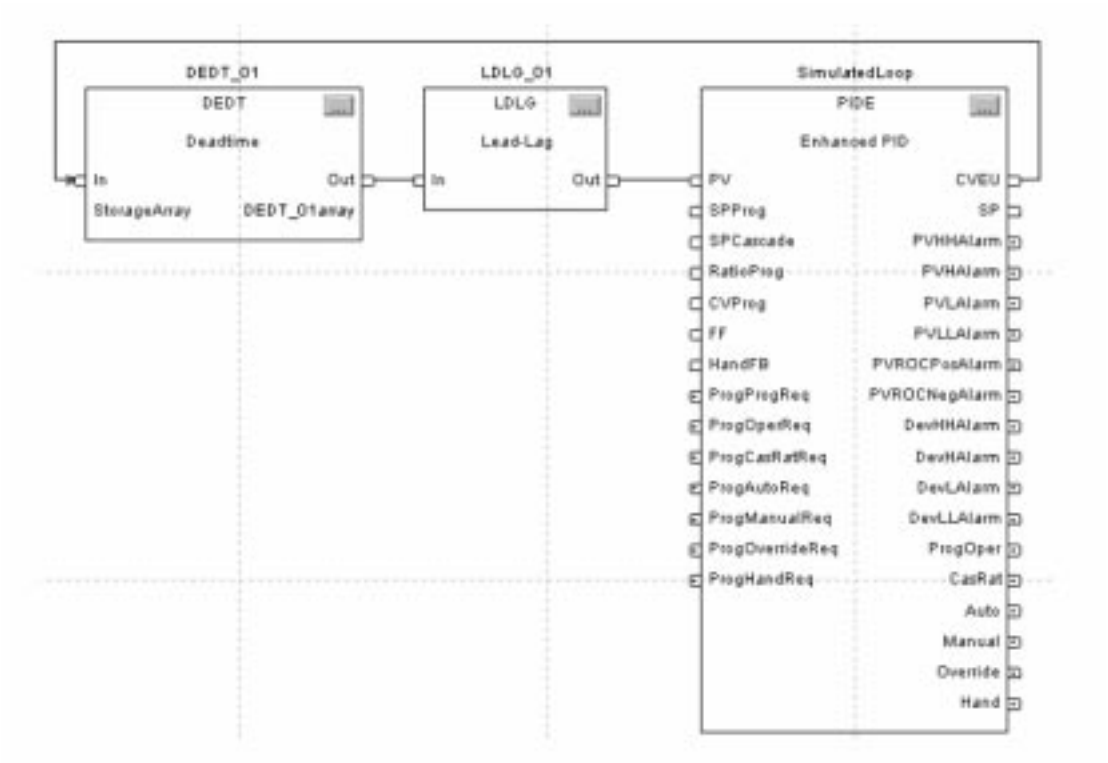
### 结构文本

```
DEDT_01.In := SimulatedLoop.CVEU;
DEDT(DEDT_01,DEDT_01array);

LDLG_01.In := DEDT_01.Out;
LDLG(LDLG_01);

SimulatedLoop.PV := LDLG_01.Out;
PIDE(SimulatedLoop);
```

功能块：



## 函数发生器(FGEN)

FGEN指令基于分段线性函数对输入进行转换。

### 操作数：



```
FGEN(FGEN_tag,X1,Y1,X2,Y2);
```

### 结构文本

| 操作数： | 类型： | 格式： | 说明： |
|---|---|---|---|
| FGEN标签 | FUNCTION_ GENERATOR | 结构体 | FGEN结构体 |
| X1<br>用， | REAL | 数组 | X轴数组，表1。与Y轴数组配合使<br>表1用来定义第一个线性段的各点。<br>有效值＝浮点数 |
| Y1<br>用， | REAL | 数组 | Y轴数组，表1。与X轴数组配合使<br>表1用来定义第一个线性段的点。<br>有效值＝浮点数 |
| X2 | REAL | 数组 | (可选)<br>X轴数组，表2。与Y轴数组配合使用，<br>表2用来定义第二个线性段的点。<br>有效值＝浮点数 |
| Y1 | REAL | 数组 | (可选)<br>Y轴数组，表2。与X轴数组配合使用，<br>表2用来定义第二个线性段的点。<br>有效值＝浮点数 |

### 功能块：



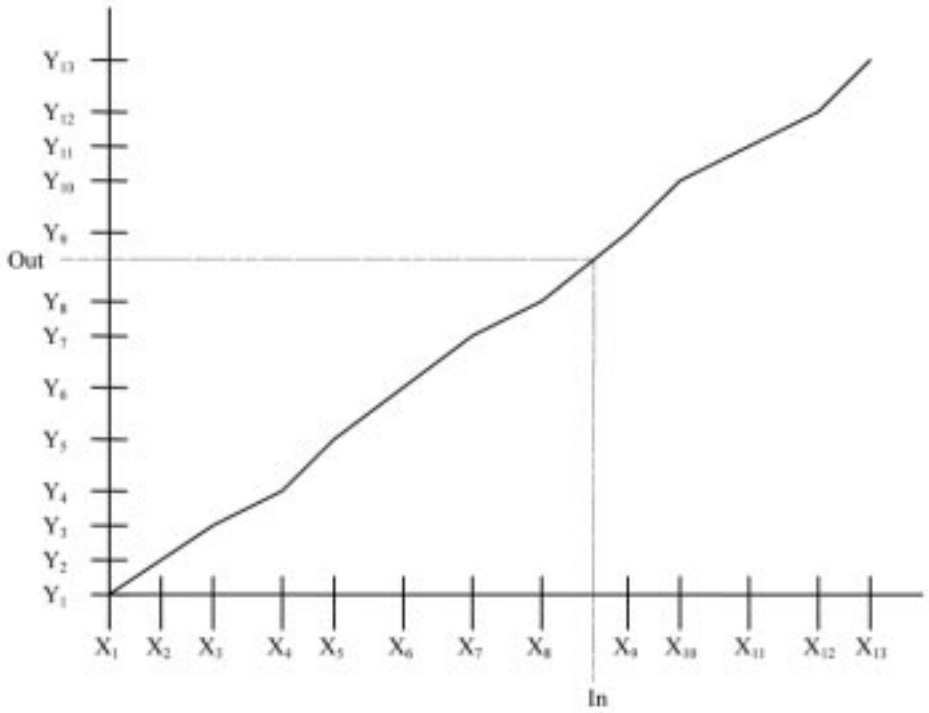



FGEN指令功能块与结构文本操作数相同。

| 输入参数： | 数据类型： | 说明： |
|---|---|---|
| EnableIn | BOOL | 功能块：<br>如果清零，则指令不执行且输出不更新。如果置位，指令执行。<br>缺省状态为置位。<br>结构文本：<br>无影响。指令一直执行。 |
| In | REAL | 指令的模拟量信号输入。<br>有效值＝浮点数<br>缺省值＝0.0 |
| XY1Size | DINT | 每个分段线性曲线使用的从表1中读取的点的数量。如果该值小于1且Select被清零，指令将置位状态字中的相应位并保持输出不变。<br>有效值＝1至(X1和Y1数组尺寸的最小值)<br>缺省值＝1 |
| XY2Size | DINT | 每个分段线性曲线使用的从表2中读取的点的数量。如果该值小于1且Select为清零状态，指令将置位状态字中的相应位并保持输出不变。<br>有效值＝0至(X2和Y2数组尺寸的最小值)<br>缺省值＝0 |
| Select | BOOL | 该输入决定使用哪个表格。清零时，指令使用表1。置位时，使用表2。<br>缺省状态为清零 |

| 输出参数： | 数据类型： | 说明： |
|---|---|---|
| EnableOut | BOOL | 输出使能 |
| Out | REAL | 指令输出。该输出影响算术状态标志。 |
| Status | DINT | 功能块状态字。 |
| InstructFault (Status.0) | BOOL | 指令产生错误。 |
| XY1SizeInv (Status.1) | BOOL | 表1尺寸无效或与数组大小不匹配。 |
| XY2SizeInv (Status.2) | BOOL | 表2尺寸无效或与数组大小不匹配。 |
| XisOutofOrder (Status.3) | BOOL | X参数没有排序。 |

**说明：**以下图表演示了FGEN指令如何转换12段的曲线。




X轴参数必须满足以下关系式：

X[1] < X[2] < X[3] < ... < X[XY<n>Size],

其中，XY<n>Size > 1且是分段线性曲线上的一点，n 等于1或2用于选择表格。用户必须在X数组中创建有序的X轴元素。

Select输入参数决定了指令使用哪个表格。当指令使用其中一个表格时，用户可以对另一个表格的数据进行修改。然后改变Select的状态使用其他表格。

在计算Out值之前，指令先扫描X轴参数。如果不是按升序排列，则置位状态字中的相应位并保持输出不变。同样，如果XY1尺寸或XY2尺寸无效，指令也置位状态字中的相应位并保持输出不变。

该指令用以下算法由输入计算输出：

- 当In ??X[1]，令Out = Y[1]
- 当In > X[XY<n>Size]，令Out = Y[XY<n>Size]
- 当X[n] < In ??X[n+1]，计算Out = ((Y[n+1]-Yn)/(X[n+1]-Xn))*(In-Xn)+Yn

**算术状态标志：**该输出影响算术状态标志。

**故障条件：**故障条件：

**执行：**

| 条件: | 功能块动作 | 结构文本动作： |
|---|---|---|
| 预扫描 | 无动作。 | 无动作。 |
| 指令首次扫描 | 无动作。 | 无动作。 |
| 指令首次执行 | 无动作。 | 无动作。 |
| 输入使能(EnableIn)清零 | 输出使能(EnableOut)清零。指令不工作，输出不更新。 | 不影响。 |
| 输入使能(EnableIn)置位 | 执行指令。<br>输出使能(EnableOut)置位 | 输入使能(EnableIn)一直置位。指令执行。 |
| 后期扫描 | 无动作。 | 无动作。 |

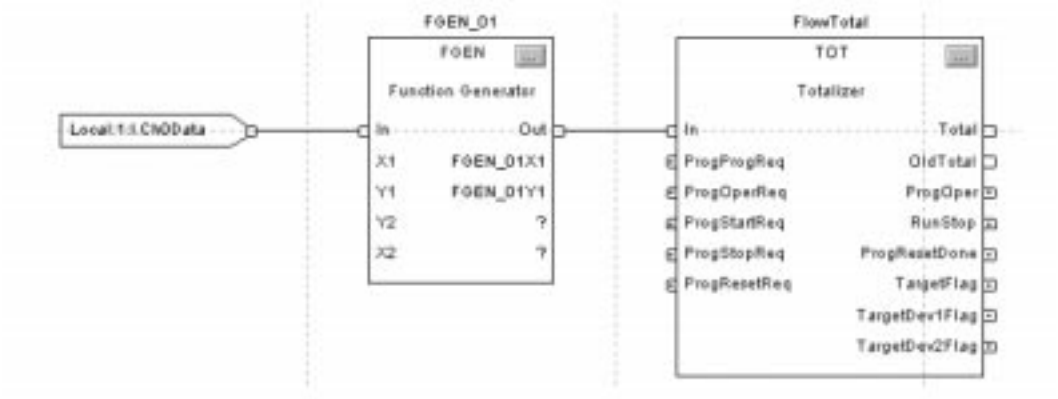
**实例：** 在此例中，FGEN指令转换一个变化的信号，然后用TOT指令求和。FGEN_01X1和FGEN_01Y1数组都是具有10个元素的实型(REAL)数组，最大支持9段曲线。用户可以任意尺寸的数组来得到所希望的分段数。

**结构文本：**

```
FGEN_01.IN := Local:1:I.Ch0Data;
FGEN(FGEN_01,FGEN_01X1,FGEN_01Y1);

FlowTotal.In := FGEN_01.Out;
TOT(FlowTotal);
```

**功能块**



## 超前—滞后补偿(LDLG)

LDLG指令为输入信号提供一个超前—滞后的相位补偿。该指令的典型应用是PID控制提供前馈补偿或进行过程仿真。

### 操作数：

#### 结构文本：

| 操作数： | 类型： | 格式： | 说明： |
|---|---|---|---|
| LDLG标签 | LEAD_LAG | 结构体 | LDLG结构体 |

功能块：

| 操作数： | 类型： | 格式： | 说明： |
|---|---|---|---|
| 块标签 | LEAD_LAG | 结构体 | LDLG结构体 |

#### LDLG结构体

| 输入参数： | 数据类型： | 说明： |
|---|---|---|
| EnableIn | BOOL | 功能块：<br>如果清零，则指令不执行且输出不更新。如果置位，指令执行。<br>缺省状态为置位。<br>结构文本：<br>无影响。指令一直执行。 |
| In | REAL | 指令的模拟量信号输入。<br>有效值＝浮点数<br>缺省值＝0.0 |
| Initialize | BOOL | 过滤器控制算法初始化请求。当Initialize置位时，Out = (In x Gain) + Bias。<br>缺省状态为清零 |
| Lead | REAL | 超前时间(单位：秒)。设置Lead＝0.0可禁止超前控制算法。如果Lead < 0.0，指令将置位状态字中的相应位，且使Lead等于0.0。如果Lead >最大正浮点数，指令将置位状态字中的相应位。<br>有效值＝大于等于0.0的浮点数<br>缺省值＝0.0 |
| Lag | REAL | 滞后时间(单位：秒)。最小的滞后时间为DeltaT/2。如果Lag <DeltaT/2，则指令将置位状态字中的相应位，且限制Lag等于DeltaT/2。如果Lead >最大正浮点数，指令将置位状态字中的相应位。<br>有效值＝大于等于DeltaT/2浮点数<br>缺省值＝0.0 |
| Gain | REAL | 过程量的增益因子。该值允许进行过程增益仿真。输入信号乘以该数值。<br>I = (In x Gain) + Bias<br>有效值＝浮点数<br>缺省值＝1.0 |

| 输入参数： | 数据类型： | 说明： |
|---|---|---|
| Bias | REAL | 过程量的偏移量。该值允许进行环境条件仿真。该值与输入和增益值的乘积相加。<br>I = ( In x Gain ) + Bias<br>有效值＝浮点数<br>缺省值＝1.0 |
| TimingMode | DINT | 选择定时执行方式。<br>**有效值：** **说明：**<br>0 周期方式<br>1 过采样方式<br>2 实时采样方式<br>有效值=0 到2<br>缺省值=0<br>需要了解更多与定时方式有关的信息，请参阅附录：功能块属性。 |
| OversampleDT | REAL | 过采样方式(oversample mode)的执行时间<br>有效值=0 到4194.303 秒<br>缺省值=0 |
| RTSTime | DINT | 实时采样方式下模块的更新周期。<br>有效值=1 到32,767ms<br>缺省值=1 |
| RTSTimeStamp | DINT | 实时采样方式下模块时间戳的值。<br>有效值=0 到32,767ms<br>缺省值=0 |

| 输出参数： | 数据类型： | 说明： |
|---|---|---|
| EnableOut | BOOL | 输出使能。 |
| Out | REAL | 停滞算法计算的输出结果。输出结果将影响算术状态标志。 |
| DeltaT | REAL | 两次更新之间经过的时间。该时间间隔以秒为单位，用于控制算法计算过程量输出值。 |
| Status | DINT | 功能块的状态字。 |
| InstructFault(Status.0) | BOOL | 指令检测以下的运行错误。该错误不使控制器产生次要或主要错误。检测其它状态位以确定所发生故障类型。 |
| LeadInv(Status.1) | BOOL | Lead < 最小值 或Lead >最大值。 |
| LagInv(Status.2) | BOOL | Lag< 最小值 或Lag >最大值。 |
| TimingModeInv(Status.27) | BOOL | 定时方式值无效。<br>需要了解更多与定时方式有关的信息，请参阅附录：功能块属性。 |
| RTSMissed(Status.28) | BOOL | 只用于实时采样方式。当ABS| DeltaT ®C RTSTime | >1(0.001秒)时，该位被置位。 |
| RTSTimeInv(Status.29) | BOOL | 实时采样时间值无效。 |
| RTSTimeStampInv(Status.30) | BOOL | 实时采样时间标签值无效。 |
| DeltaTInv(Status.31) | BOOL | DeltaT值无效。 |

**说明：** LDLG指令支持一个超前环节和一个滞后环节串联。该指令还允许用户配置增益因子和偏移量。LDLG指令是为扫描时间为常数的任务而设计的。

LDLG指令使用该传递函数：

$$H(s) = \left[ \frac{1+Lead \times s}{1+Lag \times s} \right]$$

参数极限如下：

| 参数： | 极限： |
|---|---|
| Lead | LowLimit = 0.0<br>HighLimit = 最大正浮点数 |
| Lag | LowLimit = DeltaT/2 (DeltaT以秒为单位)<br>HighLimit = 最大正浮点数 |

如果计算出来的输出值无效，NAN或±INF，指令设置输出使Out=无效值，并置位算术溢出状态标志。当计算得到的输出值有效时，指令初始化内部参数并设置输出Out = (In x Gain) + Bias。

**算术状态标志：** 该输出影响算术状态标志。

**故障条件：** 无

**执行：**

| 条件: | 结构文本动作: | 功能块动作： |
|---|---|---|
| 预扫描 | 无动作。 | 无动作。 |
| 指令首次扫描 | 无动作。 | 无动作。 |
| 指令首次执行 | 指令设置Out = (In x Gain) + Bias。<br>控制算法不执行。 | |
| 输入使能(EnableIn)清零 | 输出使能(EnableOut)清零。 | 不影响 |
| 输入使能(EnableIn)置位 | 执行指令。<br>输出使能(EnableOut)置位 | 输入使能一直置位。<br>指令执行。 |
| 后期扫描 | 无动作。 | 无动作。 |

**实例：** LDLG指令在此例中将一个一阶滞后环节加到仿真的过程量中。用户可以输入增益因子(Gain)来仿真一个过程增益也可输入偏移量(Bias)来仿真一个环境条件。

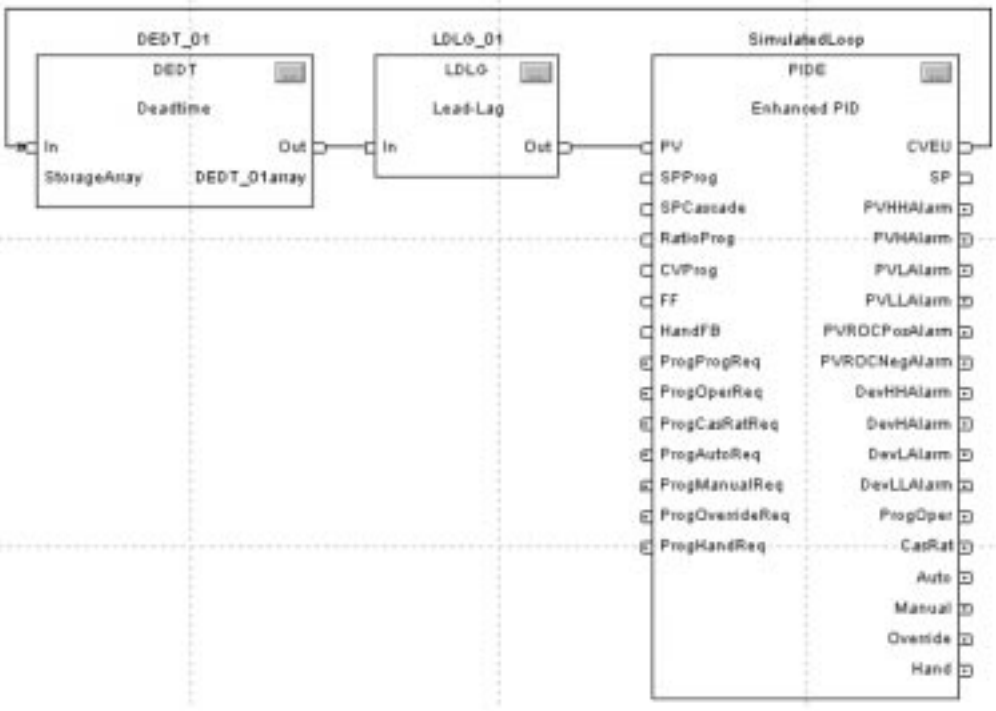
### 结构文本

```
DEDT_01.In := SimulatedLoop.CVEU;
DEDT(DEDT_01,DEDT_01array);

LDLG_01.In := DEDT_01.Out;
LDLG(LDLG_01);

SimulatedLoop.PV := LDLG_01.Out;
PIDE(SimulatedLoop);
```

### 功能块



## 增强型PID(PIDE)

PIDE在标准的PID指令之上提供增强的功能。该指令使用PID算法的速度公式。增益值应用于错误或PV的变化值，而不是错误值或PV值。

### 操作数：

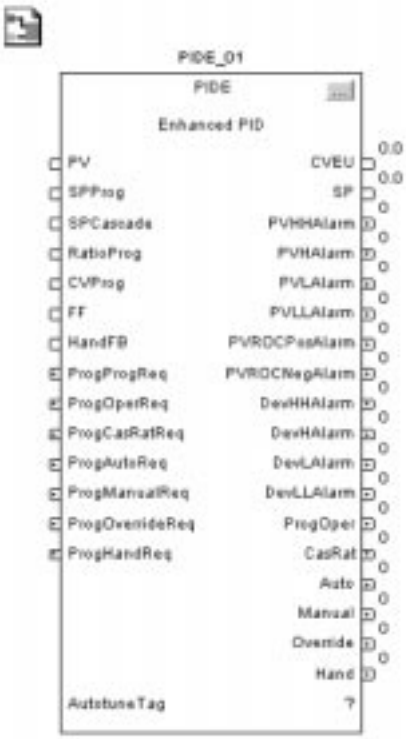



#### 结构文本

| 操作数： | 类型： | 格式： | 说明： |
|---|---|---|---|
| PIDE标签 | PIDE_ENHANCED | 结构体 | PIDE结构体 |

结构文本不支持可用于功能块的自动调节标签。

#### 功能块




| 操作数： | 类型： | 格式： | 说明： |
|---|---|---|---|
| PIDE标签 | PIDE_ENHANCED | 结构体 | PIDE结构体 |
| 自动调节标签 | PIDE_AUTOTUNE | 结构体 | (可选)<br>自动调节结构体，查阅1-53 |

## PIDE结构体

| 输入参数： | 数据类型： | 说明： |
|---|---|---|
| EnableIn | BOOL | 功能块：<br>如果清零，则指令不执行且输出不更新。如果置位，指令执行。<br>缺省状态为置位。<br>结构文本：<br>无影响。指令一直执行。 |
| PV | REAL | 整定的过程变量输入。该数据通常由模拟量模块读入。<br>有效值＝浮点数<br>缺省值＝0.0 |
| PVFault | BOOL | PV故障指示。如果PV值从模拟量输入得到，则PVFault由模拟量输入出错状态控制。<br>当PVFault置位时，表示输入信号出错。<br>缺省值状态为清零＝状态正常 |
| PVEUMax | REAL | PV的最大整定值。该值对应当前过程变量中PV和SP范围的100%。<br>有效值= PVEUMin < PVEUMax？最大正浮点数<br>缺省值＝100.0 |
| PVEUMin | REAL | PV的最小整定值。该值对应当前过程变量中PV和SP范围的0。<br>有效值= 最小负浮点数?PVEUMin < PVEUMax<br>缺省值＝0.0 |
| SPProg | REAL | SP程序值，整定为PV单位。在程序控制和非级联/比率方式下，SP值被设定为该值。<br>当SPProg < SPLLimit 或> SPHLimit时，指令置位状态字中的相应位并限制SP的数值。<br>有效值＝SPLLimit到SPHLimit<br>缺省值＝0.0 |
| SPOper | REAL | SP操作员值，整定为PV单位。在操作员控制和非级联/比例方式下，SP值被设定为该值。当SPOper < SPLLimit 或> SPHLimit时，指令置位状态字中的相应位并限制SP的数值。<br>有效值＝SPLLimit到SPHLimit<br>缺省值＝0.0 |
| SPCascade | REAL | SP级联值，整定为PV单位。当CascadeRatio置位UseRatio清零时，SP=SPCascade。这是典型的主回路的CVEU值。如果CascadeRatio和UseRatio都置位，则SP ＝(SPCascade x Ratio)。当SPCascade < SPLLimit 或> SPHLimit时，指令置位状态字中的相应位并限制SP的数值。<br>有效值＝SPLLimit到SPHLimit<br>缺省值＝0.0 |
| SPHLimit | REAL | SP上限值，整定为PV单位。如果SPHLimit < PVEUMax，指令置位状态字的相应位。<br>有效值= SPLLimit到PVEUMax<br>缺省值＝100.0 |
| SPLLimit | REAL | SP下限值，整定为PV单位。如果SPLLimit < PVEUMin，指令置位状态字的相应位。<br>当 SPHLimit < SPLLimit时，指令置位状态字中的相应位并以SPLLimit限制SP数值。<br>有效值= PVEUMin 到 SPHLimit<br>缺省值＝0.0 |
| UseRatio | BOOL | 允许比率控制。在Cascade/Ratio方式下，置位时使能比例控制。<br>缺省状态为清零 |

| 输入参数： | 数据类型： | 说明： |
|---|---|---|
| RatioProg | REAL | 程序比例因子。在程序控制方式下，Ratio和RatioOper被设定为该数值。如果RatioProg < RatioLLimit 或> RatioHLimit指令将置位状态字中的相应位并限制Ratio的数值。<br>有效值＝RatioLLimit 到 RatioHLimit<br>缺省值＝1.0 |
| RatioOper | REAL | 操作员比例因子。在操作员控制方式下，Ratio等于该数值。<br>如果RatioOper < RatioLLimit 或> RatioHLimit，指令将置位状态字中的相应位并限制Ratio的数值。<br>有效值＝RatioLLimit 到 RatioHLimit<br>缺省值＝1.0 |
| RatioHLimit | REAL | 比例上限值。RatioProg或RatioOper中Ratio的极限数值。如果RatioHLimit < RatioLLimit，指令置位状态字中相应的位，并以RatioLLimit限定Ratio的值。<br>有效值＝RatioLLimit 到最大正浮点数<br>缺省值＝1.0 |
| RatioLLimit | REAL | 比例下限值。RatioProg或RatioOper中Ratio的数值极限。如果RatioLLimit < 0，指令置位状态字中相应的位并限定其值为0。如果RatioHLimit < RatioLLimit，指令置位状态字中相应的位，并以RatioLLimit限定Ratio的值。<br>有效值＝RatioLLimit 到最大正浮点数<br>缺省值＝1.0 |
| CVFault | BOOL | 控制变量故障指示。如果CVEU控制一个模拟量输出，则CVFault由模拟量输出故障状态而来。置位时，CVFault指示输出模块出错并置位状态字中的相应位。<br>缺省状态为清零＝状态正常 |
| CVInitReq | BOOL | CV初始化请求。该信号通常由CVEU控制的模拟量输出模块的In　Hold状态控制，或由第二个PID的InitPrimary 输出控制。<br>缺省状态为清零 |
| CVInitValue | REAL | CVEU初始化值，整定为CVEU单位。当CVInitializing置位，CVEU = CVInitValue且CV等于相应的百分数值。CVInitValue来自由CVEU控制的模拟量输出的反馈或副回路的设定点。CVFaulted 或CVEUSpanInv置位时，指令初始化被禁止。<br>有效值＝浮点数<br>缺省值＝0.0 |
| CVProg | REAL | 程序人工CV值。处于程序人工方式时，CV等于该值。如果CVProg < 0或> 100，或 < CVLLimit 或 > CVHLimit且 CVManLimiting置位时，指令置位状态字中的相应位并限制CV值。<br>有效值＝0.0到100.0<br>缺省值＝0.0 |
| CVOper | REAL | CV操作员人工值。在操作员人工方式下CV等于该值。非操作员人工方式下，指令在每次指令执行的末尾设定CVOper = CV。If CVOper < 0或> 100，或 < CVLLimit 或 > CVHLimit且CVManLimiting置位，则指令置位状态字中的相应位并限制CV值。<br>有效值＝0.0到100.0<br>缺省值＝0.0 |
| CVOverride | REAL | CV自动值。在自动方式下CV等于该数值。该数值与PID环节的安全状态输出相对应。CVOverride < 0 或 >100时指令置位状态字中的相应位并限制CV值。<br>有效值＝0.0到100.0<br>缺省值＝0.0 |

| 输入参数： | 数据类型： | 说明： |
|---|---|---|
| CVPrevious | REAL | CVn-1数值。如果CVSetPrevious置位则CVn-1等于该值。CVn-1为上次指令执行的CV值。在人工、自动、手动方式下或CVInitializing置位时，CVPrevious被忽略。在自动或级联/比例方式下，如果CVPrevious < 0 或 > 100，或< CVLLimit 或 > CVHLimit，指令置位状态中相应的位并限制CVn-1数值。<br>有效值＝0.0到100.0<br>缺省值＝0.0 |
| CVSetPrevious | BOOL | 请求使用CVPrevious。置位时CVn-1 = CVPrevious。<br>缺省状态为清零 |
| CVManLimiting | BOOL | 人工方式CV值限制请求。在人工方式下且CVManLimiting置位，则CV由CVHLimit 和CVLLimit限制。<br>缺省状态为清零 |
| CVEUMax | REAL | CVEU的最大值。对应CV百分数100％的CVEU数值。如果CVEUMax = CVEUMin，指令置位状态字中的相应位。<br>有效值＝浮点数<br>缺省值＝100.0 |
| CVEUMin | REAL | CVEU的最小值。对应CV百分数0的CVEU数值。如果CVEUMax = CVEUMin，指令置位状态字中的相应位。<br>有效值＝浮点数<br>缺省值＝100.0 |
| CVHLimit | REAL | CV上限值。用于设置CVHAlarm报警输出。如果CVManLimiting置位时，在自动或级联/比例方式或人工方式或用于限制CV值。如果CVHLimit >100 或< CVLLimit，指令置位状态字中的相应位。如果CVHLimit < CVLLimit，则指令以CVLLimit来限制CV值。<br>有效值＝CVLLimit < CVHLimit ? 100.0<br>缺省值＝100.0 |
| CVLLimit | REAL | CV下限值。用于设置CVHAlarm报警输出。在自动或级联/比例方式或人工方式或CVManLimiting置位时用于限制CV值。如果CVLLimit < 0或CVHLimit < CVLLimit，指令置位状态字中的相应位。如果CVHLimit < CVLLimit，则指令以CVLLimit来限制CV值。<br>有效值＝0.0 ? CVLLimit< CVHLimit<br>缺省值＝0.0 |
| CVROCLimit | REAL | CV变化率限制，百分数每秒。如果CVManLimiting置位，变化率限制只用于自动或级联/比例方式或人工方式。输入0可禁止CVROCLimit。如果CVROCLimit < 0，则指令置位状态字中的相应位并禁止CVROCLimit。<br>有效值＝0.0到最大正浮点数<br>缺省值＝0.0 |
| FF | REAL | 前馈值。在CV使用死区限制时，CV过零后前馈值与CV值相加。因此FF的变化总是反映在最后的CV输出值上。如果FF < ®C100 或> 100，指令置位状态字中的相应位并限制FF的数值。<br>有效值＝ －100.0到100.0<br>缺省值＝0.0 |
| FFPrevious | REAL | FFn-1数值。如果FFSetPrevious置位，则指令设定FFn-1 = FFPrevious。FFn-1是上次指令执行时的FF值。如果FFPrevious < ®C100 或 > 100，指令置位状态字中的相应位并限制FFn-1的数值<br>有效值＝-100.0到100.0<br>缺省值＝0.0 |

| 输入参数： | 数据类型： | 说明： |
|---|---|---|
| FFSetPrevious | BOOL | 使用FFPrevious请求。置位时FFn-1 = FFPrevious。<br>缺省状态为清零 |
| HandFB | BOOL | CV手动反馈值。在手动方式下且HandFBFault清零(正常状态)时，CV等于该数值。该值通常具有手动/自动状态的区域使用，用于实现从手动状态下无冲击地切换。如果HandFB<0或>100，指令置位状态字中的相应位，并限制CV值。<br>有效值＝0.0到100.0<br>缺省值＝0.0 |
| HandFBFault | BOOL | HandFB数值出错指示。HandFB是由模拟量输入获得，则HandFBFault通常由模拟量输入通道的状态来控制。置位时，HandFBFault指示了模拟量输入出错，指令置位状态字中的相应位。<br>缺省状态为清零＝正常状态 |
| WindupHIn | BOOL | 超调上限请求。置位时，CV不允许该值增加。该信号通常由副回路的WindupHOut 输出得到。<br>缺省状态为清零 |
| WindupLIn | BOOL | 超调下限请求。置位时，CV不允许数值减小。该信号通常由副回路的WindupHOut 输出得到。<br>缺省状态为清零 |
| ControlAction | BOOL | 控制动作请求。置位时计算误差E = PV ®C SP，清零来计算误差E = SP ®CPV<br>缺省状态为清零 |
| DependIndepend | BOOL | 独立/非独立控制请求。置位时，使用独立形式的PID公式。清零时，使用非独立形式的PID公式。<br>缺省状态为清零 |
| PGain | REAL | 比例增益。选择非独立形式的PID算法时，输入无量纲的比例增益到该数值。选择独立形式的PID算法时，输入无量纲的控制增益到该数值。输入0可禁止比例控制。如果PGain < 0，指令置位状态字中的相应位并使PGain = 0。<br>有效值＝0.0至最大正浮点数<br>缺省值＝0.0 |
| IGain | REAL | 积分增益。选择非独立形式的PID算法时，输入以1/minutes为单位的积分增益到该数值。选择独立形式的PID算法时，输入以1/minutes为单位的积分时间常数到该数值。输入0可禁止积分控制。如果IGain < 0，指令置位状态字中的相应位并使IGain = 0。<br>有效值＝0.0到最大正浮点数<br>缺省值＝0.0 |
| DGain | REAL | 微分增益。选择非独立形式的PID算法时，输入以minutes为单位的微分增益到该数值。选择独立形式的PID算法时，输入以分钟为单位的微分时间常数到该数值。输入0可禁止微分控制。如果DGain < 0，指令置位状态字中的相应位并使DGain = 0。<br>有效值＝0.0到最大正浮点数<br>缺省值＝0.0 |
| PVEProportional | BOOL | PV比例控制请求。置位时，使用过程变量的变化量 (DeltaPTerm)来计算比例系数，清零时使用误差变化量(EPercent)计算。<br>缺省状态为清零 |

| 输入参数： | 数据类型： | 说明： |
|---|---|---|
| PVEDerivative | BOOL | PV微分控制请求。置位时，使用过程变量变化量 (DeltaPTerm)来计算微分系数，清零时使用误差变化量(EPercent)计算。<br>缺省状态为置位。 |
| DSmoothing | BOOL | 微分平滑请求。置位时，微分系数的变化是平滑的。微分平滑产生的对PV信号的噪声干扰只引起微小的输出抖动且能有效抑制高微分系数影响。<br>缺省状态为清零。 |
| PVTracking | BOOL | SP跟踪PV请求。在人工方式下置位，使SP跟踪PV。级联/比例或自动时请求被忽略。<br>缺省状态为清零 |
| ZCDeadband | REAL | 过零死区范围，整定为PV单位。定义过零死区的范围。输入0可禁止过零死区检测。如果ZCDeadband < 0，指令置位状态字中的相应位并禁止过零死区检测。<br>有效值＝0.0到最大正浮点数<br>缺省值＝0.0 |
| ZCOff | BOOL | 禁止过零请求。置位时，禁止死区计算过零。<br>缺省状态为清零。 |
| PVHHLimit | REAL | PV高上限报警阈值，整定为PV单位。<br>有效值＝浮点数<br>缺省值＝最大正浮点数 |
| PVHLimit | REAL | PV上限报警阈值，整定为PV单位。<br>有效值＝浮点数<br>缺省值＝最大正浮点数 |
| PVLLimit | REAL | PV下限报警阈值，整定为PV单位。<br>有效值＝浮点数<br>缺省值＝最小负浮点数 |
| PVLLLimit | REAL | PV低下限报警阈值，整定为PV单位。<br>有效值＝浮点数<br>缺省值＝最小负浮点数 |
| PVDeadband | REAL | PV报警极限死区数值，整定为PV单位。Deadband是PV报警极限打开和关闭之间的Delta差值。PVDeadband < 0.0时，指令置位状态字中的相应位并限定PVDeadband等于0。<br>有效值＝0.0到最大正浮点数<br>缺省值＝0.0 |
| PVROCPosLimit | REAL | PV正向变化速率报警极限。PV正向变化(增长)极限值， 以PV单位/秒为单位。输入0.0可禁止正向PVROC报警检测。如果PVROCPosLimit < 0.0，则指令置位状态字中的相应位并禁止PVROC检测。<br>有效值=0.0 到最大正浮点数<br>缺省值= 0.0 PV/second |
| PVROCNegLimit | REAL | PV负向变化速率报警极限。PV负向变化(减小)极限值， 以PV单位/秒为单位。输入0.0可禁止负向PVROC报警检测。如果PVROCNegLimit < 0，则指令置位状态字中的相应位并禁止PVROC检测。<br>有效值=0.0 到最大正浮点数<br>缺省值= 0.0 |

| 输入参数： | 数据类型： | 说明： |
|---|---|---|
| PVROCPeriod | REAL | PV在采样周期的变化率。该时间周期以秒为单位，计算这段时间内PV的变化率。输入0可禁止PVROC报警检测。如果PVROCPeriod< 0.0，指令置位状态字中的相应位并禁止正向和负向的PVROC检测。<br>有效值=大于等于的浮点数<br>缺省值=0.0秒 |
| DevHHLimit | REAL | 高上限报警偏离极限值，整定为PV单位。偏差量是过程变量(PV)和设定点之间的差值。偏差量报警提示操作员过程量和设定值之间存在差异。如果DevHHLimit < 0.0，则指令置位状态字中的相应位并设定DevHHLimit = 0.0。<br>有效值=0.0到最大正浮点数<br>缺省值=最大正浮点数 |
| DevHLimit | REAL | 上限报警偏离极限值，整定为PV单位。偏离量是过程变量(PV)和设定点之间的差值。偏离量报警提示操作员过程量和设定值之间存在差异。如果DevHLimit < 0.0，则指令置位状态字中的相应位并设定DevHLimit = 0.0。<br>有效值=0.0到最大正浮点数<br>缺省值=最大正浮点数 |
| DevLLimit | REAL | 下限报警偏离极限值，整定为PV单位。偏离量是过程变量(PV)和设定点(SP)之间的差值。偏离量报警提示操作员过程量和设定值之间存在差异。如果DevLLimit < 0.0，则指令置位状态字中的相应位并设定DevLLimit = 0.0。<br>有效值=0.0到最大正浮点数<br>缺省值=最大正浮点数 |
| DevLLLimit | REAL | 低下限报警偏离极限值，整定为PV单位。偏离量是过程变量(PV)和设定点之间的差值。偏离量报警提示操作员过程量和设定值之间存在差异。如果DevLLLimit < 0.0，则指令置位状态字中的相应位并设定DevLLLimit = 0.0。<br>有效值=0.0到最大正浮点数<br>缺省值=最大正浮点数 |
| DevDeadband | REAL | 偏离量极限死区数值，整定为PV单位。DevDeadband是报警极限打开和关闭之间的Delta差值。如果DevDeadband < 0.0，则指令置位状态字中的相应位并限定DevDeadband=0。<br>有效值＝0.0到最大正浮点数<br>缺省值＝0.0 |
| AllowCasRat | BOOL | 允许级联/比例方式。置位时，允许用ProgCascadeRatio Req 或 OperCascadeRatioReq选择级联/比例方式。<br>缺省状态为清零 |
| ManualAfterInit | BOOL | 初始化后人工方式请求。置位时，如果CVInitializing也置位则指令切换到人工方式，除非当前方式为自动或手动。当ManualAfterInit清零时，如果无其他方式请求，指令方式不改变。<br>缺省状态为清零 |
| ProgProgReq | BOOL | 程序编程控制请求。由用户程序置位来请求程序控制。如果ProgOperReq置位则忽略该请求。保持该请求且清零ProgOperReq可使指令进入程序控制方式。如果ProgValueReset置位，则指令每次执行完将输入清零。<br>缺省状态为清零 |

| 输入参数： | 数据类型： | 说明： |
|---|---|---|
| ProgOperReq | BOOL | 程序操作员请求。用户程序置位来请求操作员控制。保持置位状态时指令进入操作员控制状态。如果ProgValueReset置位，则指令每次执行完将输入清零。<br>缺省状态为清零 |
| ProgCasRatReq | BOOL | 程序级联/比例方式请求。由用户程序置位请求级联/比例方式控制。保持该位使指令进入操作员控制。如果ProgValue Reset置位，则指令每次执行完将输入清零。<br>缺省状态为清零 |
| ProgAutoReq | BOOL | 程序自动方式请求。由用户程序置位请求自动方式控制。如果ProgValueReset置位，则指令每次执行完将输入清零。<br>缺省状态为清零 |
| ProgManualReq | BOOL | 程序人工方式请求。由用户程序置位请求人工方式控制。如果ProgValueReset置位，则指令每次执行完将输入清零。<br>缺省状态为清零 |
| ProgOverrideReq | BOOL | 程序自动方式请求。由用户程序置位请求自动方式控制。如果ProgValueReset置位，则指令每次执行完将输入清零。<br>缺省状态为清零 |
| ProgHandReq | BOOL | 程序手动方式请求。由用户程序置位请求手动方式控制。如果ProgValueReset置位，则指令每次执行完将输入清零。<br>缺省状态为清零。 |
| OperProgReq | BOOL | 操作员程序请求。由操作员界面置位请求程序控制。指令每次执行完将输入清零。<br>缺省状态为清零 |
| OperOperReq | BOOL | 操作员操作请求。由操作员界面置位请求操作员控制。指令每次执行完将输入清零。<br>缺省状态为清零 |
| OperCasRatReq | BOOL | 操作员级联/比例方式请求。由操作员界面置位请求进入级联/比例方式。指令每次执行完将输入清零。<br>缺省状态为清零。 |
| OperAutoReq | BOOL | 操作员自动方式请求。由操作员界面置位请求进入自动方式。指令每次执行完将输入清零。<br>缺省状态为清零。 |
| OperManualReq | BOOL | 操作员人工方式请求。由操作员界面置位请求人工方式。指令每次执行完将输入清零。<br>缺省状态为清零 |
| ProgValueReset | BOOL | 重置程序控制数值。置位时，指令每次执行要求清零所有程序请求输入。在操作员方式下置位时，指令设定SPProgram = SP 及CVProgram = CV。<br>缺省状态为清零 |
| TimingMode | DINT | 选择定时执行方式：<br>**有效值：** **说明：**<br>0 周期执行方式<br>1 过采样方式<br>2 实时采样方式<br>需要了解更多与定时方式有关的信息，请参阅附录：功能块属性。<br><br>有效值=0到2<br>缺省值=0 |

| 输入参数： | 数据类型： | 说明： |
|---|---|---|
| OversampleDT | REAL | 过采样方式的执行时间。<br>有效值=0到4194.303秒<br>缺省值=0 |
| RTSTime | DINT | 实时采样方式下模块的更新周期。<br>有效值=1到32,767ms<br>缺省值=1 |
| RTSTimeStamp | DINT | 实时采样方式下模块的时间标签。<br>有效值=0到32,767ms<br>缺省值=0 |

| 输出参数： | 数据类型： | 说明： |
|---|---|---|
| EnableOut | BOOL | 输出使能。 |
| CVEU | REAL | 控制变量输出整定。根据CVEUMax和CVEUMin来整定，其中CVEUMax对应100%，CVEUMin对应0%。该输出通常用于控制模拟量输出模块或一个副回路。<br>CVEU = (CV x CVEUSpan / 100) + CVEUMin<br>CVEU范围计算：CVEUSpan = (CVEUMax - CVEUMin) |
| CV | REAL | 控制变量输出。该数值被表示成0－100%。在自动或级联/比例方式下及人工方式下CVManLimiting置位时，CV由CVHLimit 和 CVLLimit限制。其他情况下，该值的极限为0－100%。输出结果将影响算术状态字设置。 |
| CVInitializing | BOOL | 初始化方式指示。在指令首次扫描和将CVHealth过渡(由故障到正常)清零时，如果CVInitReq置位则CVInitializing置位。当指令初始化完成和CVInitReq清零时，CVInitializing被清零。 |
| CVHAlarm | BOOL | CV高报警指示。当CV计算值CV > 100 或CVHLimit时置位。 |
| CVLAlarm | BOOL | CV低报警指示。当CV计算值CV < 0 或 CVLLimit时置位。 |
| CVROCAlarm | BOOL | CV变化速率报警指示。当CV 变化速率超过CVROCLimit时置位。 |
| SP | REAL | 当前设定点数值。SP数值在自动或级联/比例方式下用于控制CV输出。 |
| SPPercent | REAL | 以PV的百分数表示的SP的数值。<br>SPPercent = ((SP ®C PVEUMin) x 100) / PVSpan<br>PV 范围计算： PVSpan = (PVEUMax - PVEUMin) |
| SPHAlarm | BOOL | SP高报警指示。当SP > SPHLimit时置位。 |
| SPLAlarm | BOOL | SP低报警指示。当SP < SPLLimit时置位。 |
| PVPercent | REAL | 以百分数表示的PV值。<br>PVPercent = ((PV- PVEUMin) x 100) / PVSpan<br>PV范围计算 PVSpan = (PVEUMax - PVEUMin) |
| E | REAL | 过程误差。SP与PV之间的差值，整定为PV单位。 |
| EPercent | REAL | 误差以百分数范围表示。 |
| InitPrimary | BOOL | 初始化主环节命令。在非级联/比例方式或CVInitializing置位时该位置位。该信号通常由一个主PID环节的CVInitReq来使用。 |

| 输出参数： | 数据类型： | 说明： |
|---|---|---|
| WindupHOut | BOOL | 超调上限指示。当SP或CV到上限或CV到达下限(由控制动作决定)时，该位置位。该信号通常由主环节的WindupHIn输入控制，以防止主回路的CV输出超调。 |
| WindupLOut | BOOL | 超调下限指示。当SP或CV到上限或CV到达下限(由控制动作决定)时，该位置位。该信号通常由主环节的WindupHIn输入控制，以防止主回路的CV输出超调。 |
| Ratio | REAL | 当前比例系数。 |
| RatioHAlarm | BOOL | 比例系数高报警指示。Ratio > RatioHLimit 时置位。 |
| RatioLAlarm | BOOL | 比例系数低报警指示。Ratio< RatioLLimit 时置位。 |
| ZCDeadbandOn | BOOL | 过零死区指示。置位时，CV保持不变。如果ZCOff置位，则当\| E \|在ZCDeadband范围内时ZCDeadbandOn置位。如果ZCOff清零，则当\| E \|过零且仍在ZCDeadband范围内时，ZCDeadbandOn置位。当\| E \|超出ZCDeadband范围或ZCDeadband = 0时，ZCDeadbandOn清零。 |
| PVHHAlarm | BOOL | PV高上限报警指示。PV ⩾ PVHHLimit时置位，PV < (PVHHLimit - PVDeadband)时清零。 |
| PVHAlarm | BOOL | PV上限报警指示。PV ⩾ PVHLimit时置位，PV < (PVHLimit - PVDeadband)时清零。 |
| PVLAlarm | BOOL | PV下限报警指示。PV⩽ PVLLimit时置位，PV > (PVLLimit + PVDeadband)时清零。 |
| PVLLAlarm | BOOL | PV低下限报警指示。PV ⩽ PVLLLimit时置位，PV > (PVLLLimit + PVDeadband)时清零。 |
| PVROCPosAlarm | BOOL | PV正向变化速率报警指示。当计算出的PV变化速率 ? PVROCPosLimit时置位。 |
| PVROCNegAlarm | BOOL | PV负向变化速率报警指示。当计算出的PV 变化速率? (PVROCNegLimit x -1)时置位。 |
| DevHHAlarm | BOOL | 偏离量高上限报警指示。PV ⩾ (SP + DevHHLimit)时置位，PV < (SP + DevHHLimit - DevDeadband)时清零。 |
| DevHAlarm | BOOL | 偏离量上限报警指示。PV ⩾ (SP + DevHLimit)时置位，PV < (SP + DevHLimit ®C DevDeadband)时清零。 |
| DevLAlarm | BOOL | 偏离量下限报警指示。PV ⩽ (SP - DevLLimit)时置位，PV > (SP - DevLLimit + DevDeadband)时清零。 |
| DevLLAlarm | BOOL | 偏离量低下限报警指示。PV ⩽ (SP ®C DevLLLimit)时置位，PV > (SP ®C DevLLLimit + DevDeadband)时清零。 |
| ProgOper | BOOL | 程序/操作员指示。程序控制方式时置位，操作员控制时清零。 |
| CasRat | BOOL | 级联/比例方式指示。处于级联/比例方式时置位。 |
| Auto | BOOL | 自动方式指示。处于自动方式时置位。 |
| Manual | BOOL | 人工方式指示。处于人工方式时置位。 |
| Override | BOOL | 自动方式指示。处于自动方式时置位。 |
| Hand | BOOL | 手动方式指示。处于手动方式时置位。 |
| DeltaT | REAL | 两次更新之间经过的时间。作为控制算法计算过程量输出值的时间间隔参数，以秒为单位。 |
| Status1 | DINT | 功能块的状态。 |

| 输出参数： | 数据类型： | 说明： |
|---|---|---|
| InstructFault (Status1.0) | BOOL | 指令检测以下的运行错误之一。该错误不使控制器产生次要或主要错误。检测其它状态位以确定所发生故障类型。 |
| PVFaulted (Status1.1) | BOOL | 过程变量(PV)出错。 |
| CVFaulted (Status1.2) | BOOL | 控制变量(CV)出错。 |
| HandFBFaulted (Status1.3) | BOOL | HandFB数值出错。 |
| PVSpanInv (Status1.4) | BOOL | PV范围无效。PVEUMax ? PVEUMin。 |
| SPProgInv (Status1.5) | BOOL | SPProg < SPLLimit 或SPProg > SPHLimit。指令使用极限值设定SP值。 |
| SPOperInv (Status1.6) | BOOL | SPCascade < SPLLimit 或 SPCascade > SPHLimit。指令使用极限值设定SP值。 |
| SPCascadeInv (Status1.7) | BOOL | SPCascade < SPLLimit 或 SPCascade > SPHLimit。指令使用极限值设定SP值 |
| SPLimitsInv (Status1.8) | BOOL | 极限值无效：SPLLimit < PVEUMin， SPHLimit > PVEUMax 或 SPHLimit < SPLLimit。如果SPHLimit < SPLLimit，则指令用SPLLimit作为极限数值。 |
| RatioProgInv (Status1.9) | BOOL | RatioProg < RatioLLimit或RatioProg > RatioHLimit。指令取Ratio做极限数值。 |
| RatioOperInv (Status1.10) | BOOL | RatioOper<RatioLLimit或RatioOper > RatioHLimit。指令取Ratio做极限数值。 |
| RatioLimitsInv (Status1.11) | BOOL | Low limit < 0 或High limit < low limit. |
| CVProgInv (Status1.12) | BOOL | CVProg < 0或CVProg > 100，或 CVProg < CVLLimit 或CVProg > CVHLimit。当CVManLimiting置位时，指令取CV极限数值。 |
| CVOperInv (Status1.13) | BOOL | CVOper < 0或CVOper > 100，或CVOper < CVLLimit 或CVOper > CVHLimit。当CVManLimiting置位时，指令取CV极限数值。 |
| CVOverrideInv (Status1.14) | BOOL | CVOverride < 0 或CVOverride > 100。指令取CV极限数值。 |
| CVPreviousInv (Status1.15) | BOOL | 处于自动或级联/比例方式时CVPrevious<0或CVPrevious >100，或 < CVLLimit 或> CVHLimit。指令取CVn-1极限数值。 |
| CVEUSpanInv (Status1.16) | BOOL | CVEU范围无效。指令取CVEUMax = CVEUMin. |
| CVLimitsInv (Status1.17) | BOOL | CVLLimit < 0，CVHLimit >100，或CVHLimit < CVLLimit。如果CVHLimit < CVLLimit，则指令用CVLLimit来限制CV。 |
| CVROCLimitInv(Status1.18) | BOOL | CVROCLimit < 0。指令禁止ROC极限。 |
| FFInv (Status1.19) | BOOL | FF < ®C100或FF > 100。指令取FF极限数值。 |
| FFPreviousInv(Status1.20) | BOOL | FFPrevious < ®C100 或FFPrevious > 100。指令取FFn-1极限数值。 |
| HandFBInv (Status1.21) | BOOL | HandFB < 0 或HandFB > 100。指令取CV极限数值。 |
| PGainInv (Status1.22) | BOOL | PGain < 0。指令令PGain = 0。 |
| IGainInv (Status1.23) | BOOL | IGain < 0。指令令IGain= 0。 |
| DGainInv (Status1.24) | BOOL | DGain < 0。指令令DGain= 0。 |
| ZCDeadbandInv(Status1.25) | BOOL | ZCDeadband < 0。指令禁止过零死区。 |

| 输出参数： | 数据类型： | 说明： |
|---|---|---|
| PVDeadbandInv(Status1.26) | BOOL | PVDeadband < 0。 |
| PVROCLimitsInv(Status1.27) | BOOL | PVROCPosLimit < 0，PVROCNegLimit < 0，或 PVROCPeriod < 0。 |
| DevHLLimitsInv(Status1.28) | BOOL | 偏差量上下极限值无效。Low-low limit < 0， low limit < 0， high limit < 0， 或 high-high limit < 0。指令将设定无效限值为0。 |
| DevDeadbandInv(Status1.29) | BOOL | Deviation deadband < 0。指令令DevDeadband = 0。 |
| Status2 | DINT | 功能块定时状态字。 |
| TimingModeInv(Status2.27) | BOOL | 定时方式值无效。<br>需要了解更多与定时方式有关的信息，请参阅附录：功能块属性。 |
| RTSMissed(Status2.28) | BOOL | 只用于实时采样方式。当ABS| DeltaT - RTSTime | >1(0.001秒)时，该位被置位。 |
| RTSTimeInv(Status2.29) | BOOL | RTSTime值无效。 |
| RTSTimeStampInv(Status2.30) | BOOL | RTSTime值无效。 |
| DeltaTInv(Status2.31) | BOOL | DeltaT值无效。 |

**说明：** 在级联/比例或自动方式时，PID算法计算CV输出使PV稳定在SP附近。

当ControlAction置位时，Epercent和PVPIDPercent数值被取反，然后参与控制算法运算。

下表说明如何计算PID系数：

| PID系数： | 计算公式： |
|---|---|
| 比例 | 比例系数计算如下：<br>• PV，当PVEProportional置位时或<br>• Error，当PVEProportional清零时。<br>设PGain = 0可禁止比例控制。 |
| 积分 | 积分系数由误差(Error)计算得来。令IGain = 0可禁止积分控制。在DependIndepend置位时设定PGain = 0也可禁止积分控制。 |
| 微分 | 微分系数计算如下：<br>• PV，当PVEDerivative置位或<br>• Error，当PVEDerivative清零<br>设定DGain = 0 可禁止微分控制。在DependIndepend置位时设定PGain = 0也可禁止积分控制。<br><br>DSmoothing置位时，使能微分平滑，DSmoothing清零时，禁止微分平滑。微分平滑产生的对PV信号的噪声干扰只引起微小的输出抖动且能有效抑制高微分系数影响。 |

## 计算CV值

PID算法通过Delta Pterm、Delta ITerm、Delta Dterm及前一次指令执行的CV值(如CVn-1)的总和来计算CV值。当CVSetPrevious置位时，CVPrevious = CVn-1。这允许用户在计算之前给CVn-1设定特殊的数值。

计算值CV = $CV_{n-1}$ +Δ PTerm + Δ ITerm+ Δ DTerm

## 监控PIDE指令

PIDE指令具有操作员面板。需要更多的信息，请参阅附录：功能块面板控制。

## 自动调节PIDE指令

RSLogix5000软件PIDE指令中内置开环的PIDE自动调节器。用户可以通过PanelView终端或任一其他操作员设备，如RSLogix5000软件实现自动调节。如果用户希望对PIDE功能块进行自动调节，需指定该PIDE功能块的Autotune标签(类型PIDE_AUTOTUNE)。

PIDE自动调节器与RSLogix5000软件同时安装，但用户需要特定的激活匙来使能自动调节器。另外，仅在功能块编程中可使用自动调节功能；在梯形图和结构文本编程中不可利用。

通过Autotune栏为PIDE功能块指定并组态自动调节标签。



关于自动调节器的更多信息，查阅RSLogix5000在线帮助或PIDE自动调节器入门指南，出版号PIDE-GR001。

**算术状态标志：**CV输出影响算术状态标志位。

**故障条件：**无

## 执行：

| 条件: | 功能块动作: | 结构文本动作： |
|---|---|---|
| 预扫描 | InstructionFirstScan置位。 | InstructionFirstScan置位。 |
| 指令首次扫描 | 如果CVFault 和CVEUSpanInv置位，则参考过程故障(Processing Faults )，页1-74。<br>如果 CVFault and CVEUSpanInv清零<br>1. CVInitializing置位<br>2. 如果PVFault 置位，PVSpanInv和SPLimitsInv清零。参阅页1-74过程错误。<br>3. PID控制算法不执行。<br>4. 指令设置CVEU=CVInitValue且CV=相应的百分数。<br><br>CVInitValue不受CVEUMax 或 CVEUMin限制。当指令计算CV相应的百分数时，限制在0 －100 。<br><br>CVEU=CVInitValue<br><br>$CV_{n-1} = CV = \frac{CVEU-CVEUMin}{CVEUMax-CVEUMin} \times 100$<br><br>CVOper=CV<br><br>5. 当CVInitializing和ManualAfterInit置位时，指令禁止自动或级联/比例方式。如果当前方式不是自动或手动方式，指令进入人工方式。如果ManualAfterInit清零，则方式不改变。<br>6. 所有操作员请求输入清零。<br>7. 如果ProgValueReset清零，所有程序输入请求清零。<br>8. 所有的PV上下限 、PV 变化速率和上下偏移量报警都清零。<br>9. 如果CVInitReq 清零，则CVInitializing 清零。 | |
| 指令首次执行 | ProgOper清零。<br>指令切换到人工方式。 | ProgOper清零。<br>指令切换到人工方式。 |
| 输入使能(EnableIn)清零 | 输出使能(EnableOut) 清零。 | 不影响 |
| 输入使能(EnableIn)置位 | 执行指令。<br>输出使能(EnableOut) 置位 | 使能输入(EnableIn)一直置位。<br>指令执行。 |
| 后期扫描 | 无动作。 | 无动作。 |

当CVInitReq置位或在指令首次扫描期间，或CVFault清零(由故障到正常)过渡期间，指令初始化CVEU和CV 输出为CVInitValue数值。如果定时方式为过采样方式且EnableIn由清零到置位，则指令初始化CVEU 和CV数值。在初始化结束或CVInitReq清零时，CVInitialization清零。

CVInitValue数值通常由模拟量输出的返回值得到。CVInitReq通常由受CVEU控制的模拟量输出的 ″In Hold″ 状态位得到。初始化进程是为了防止启动时输出一个信号到静止的设备而产生冲击。

如果CVFault或CVEUSpanInv置位，指令不会进行初始化，且CVEU和CV值也不更新。

使用级联PID环节时，当副回路处于初始化或级联/比例方式时，主PID回路可进行初始化。在这种情况下，将副回路的InitPrimary输出和SP移入到主回路的CVInitReq 和 CVInitValue的输入。

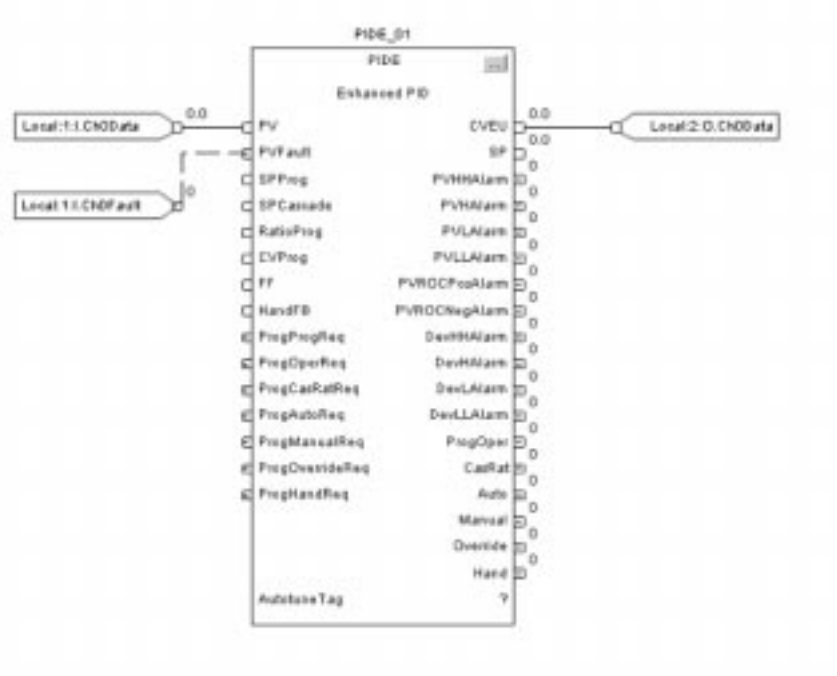
例1：使用PIDE指令最简单的方法是在周期性任务的程序中创建功能块例程。PIDE指令的缺省定时方式是周期型。当PIDE指令在周期性任务中采用周期型定时方式时，指令自动使用周期指令的更新时间作为它的Delta t 更新时间。用户所要做的只是将过程变量模拟量输入连接到PIDE指令的PV参数输入上，将PIDE指令的CVEU输出连接到控制变量模拟量输出。

可选的，用户也可将模拟量输入故障指示(如果某个可用)连接到PIDE指令的PVFault参数上。这将使模拟量输入故障时强制PIDE指令进入人工方式，以防止在PV信号不可用时，PIDE的CVEU输出产生超调。

## 结构文本

```
PIDE_01.PV := Local:1:I.Ch0Data;
PIDE_01.PVFault := Local:1:I.Ch0Fault;
PIDE(PIDE_01);
Local:2:O.Ch0Data := PIDE_01.CVEU;
```

## 功能块



**例2：** 级联控制在外部因素经常干扰控制变量时非常有用，外部因素干扰通常是所要控制的过程变量。例如，通过改变罐周围的加热管的蒸汽量来控制罐中液体的温度。当蒸汽因为逆流现象突然下降时，罐中的温度最终也要下降，这时PIDE指令就会打开蒸汽阀来补偿下降的温度。

在此例中，当蒸汽流下降早于罐中液体温度下降，一个级联环节可为蒸汽阀的开启提供更优化的的控制。使用一个从蒸汽流传感器采集过程变量信号来控制蒸汽阀开启的PIDE指令就能实现级联控制回路。这是级联对的副回路，另一个PIDE指令(称为主回路)用液体的温度作为过程变量，并将CV输出到副回路的设定点。通过这种方式，主温度回路从副回路中读取特定的蒸汽流数量，蒸汽流回路产生相应，提供温度回路所要求的蒸汽量以维持液体温度恒定。

### 结构文本

```
PrimaryLoop.PV := Local:1:I.CH0Data;
PrimaryLoop.CVInitReq := SecondaryLoop.InitPrimary;
PrimaryLoop.CVInitValue := SecondaryLoop.SP;
PrimaryLoop.WindupHIn := SecondaryLoop.WindupHOut;
PrimaryLoop.WindupLIn := SecondaryLoop.WindupLOut;

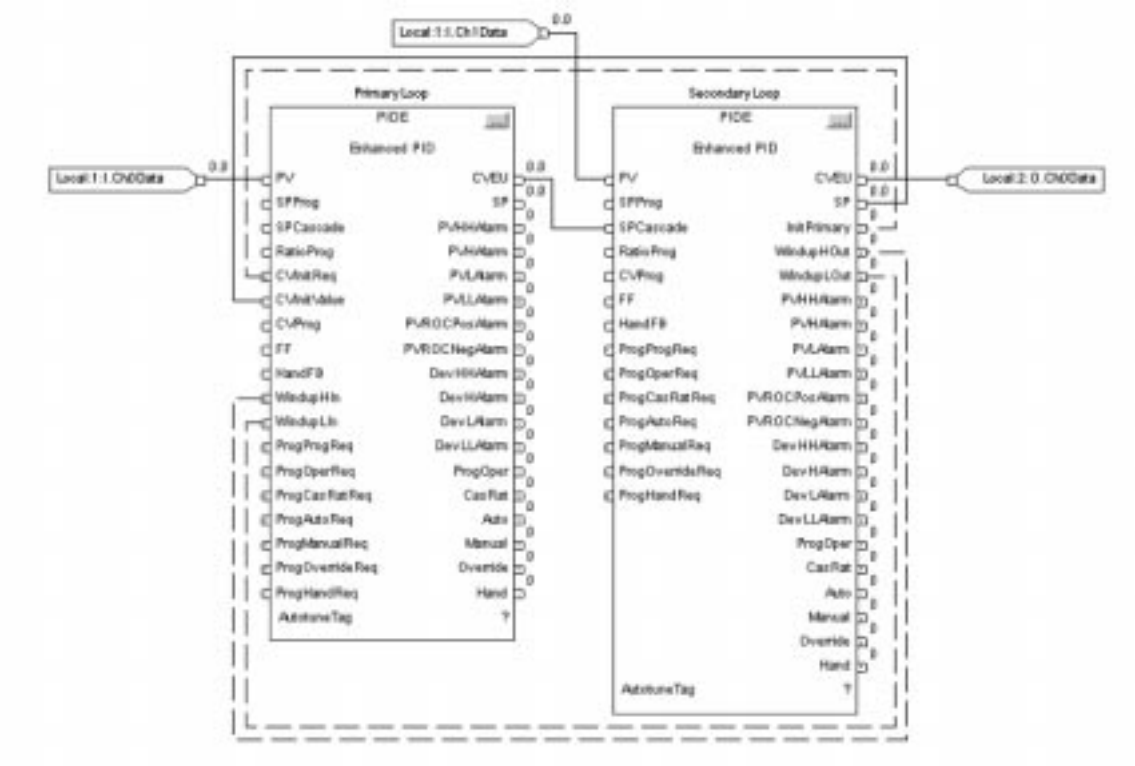
PIDE(PrimaryLoop);

SecondaryLoop.PV := Local:1:I.Ch1Data;
SecondaryLoop.SPCascade := PrimaryLoop.CVEU;

PIDE(SecondaryLoop);

Local:2:O.Ch0Data:= SecondaryLoop.CVEU;
```

## 功能块




对于一个正常工作的级联对回路，副回路必须具有比主回路更快速的响应速度。这是因为副回路必须能够在任何的干扰影响到主回路之前对干扰进行补偿。在此例中，如果蒸汽流下降，副回路控制必须在影响到液体温度之前动作，使蒸汽流增加。

要建立一个级联对PIDE指令，首先将副回路的AllowCasRat输入参数置位，这就允许副回路进入级联/比例方式。然后，将主回路的CVEU接到副回路的SPCascade输入参数上。在副回路处于级联/比例方式时，SPCascade的数值被用作为SP。主回路CVEU的工程量单位应该与副回路的PV的工程量单位一致。这使主回路以0－100%整定其CV能与副回路的设定点所用的工程单位一致。

PIDE指令还支持其它几种特性以支持更有效地级联控制。将副回路的InitPrimary输出连接到主回路的CVInitReq输入，副回路的SP输出连接到主回路的CVInitValue输入上，这使副回路处于级联/比例方式时，主回路的CVEU等于副回路的SP。这使副回路转入级联/比例方式时达到无冲击切换。将副回路的WindupHOut 和WindupLOut输出连接到主回路的WindupHIn和WindupLIn上，这将使主回路的CVEU在副回路达到SP极限或CV极限停止增加或减小，并在这些情况下消除主环节超调状态。

**例3：** 比例控制用于将一个流量按比例地加到另一个流量上。例如：要将两个反应物(称为A和B)按一定的比率加到一个容器中，因为受某种逆流的影响反应物A总在不停的变化，用户可设定利用比例控制器自动调整反应物B的添加比例。这此例中，反应物A称为“非受控”流，因为它不受PIDE指令控制。反应物B称为“受控”流。

置位PIDE输入参数AllowCasRat 和 UseRatio可实现比例控制。将非受控流接到SPCascade输入参数，处于级联/比例方式时，非受控流被乘以RatioOper(操作员控制方式)或RatioProg(程序控制方式)，其结果作为PIDE指令的设定点。

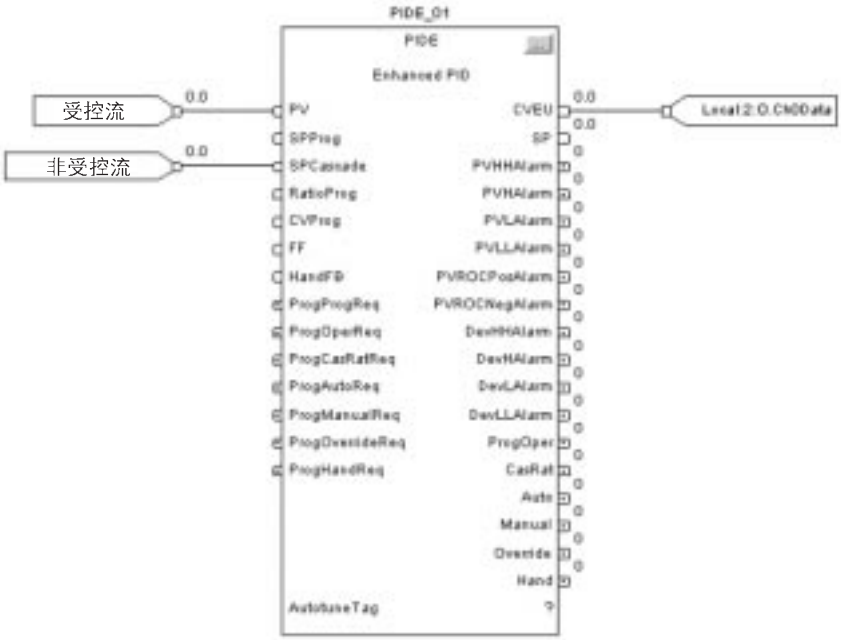
### 结构文本

```
PIDE_01.PV := ControlledFlow;
PIDE_01.SPCascade := UncontrolledFlow;

PIDE(PIDE_01);

Local:2:O.Ch0Data := PIDE_01.CVEU;
```

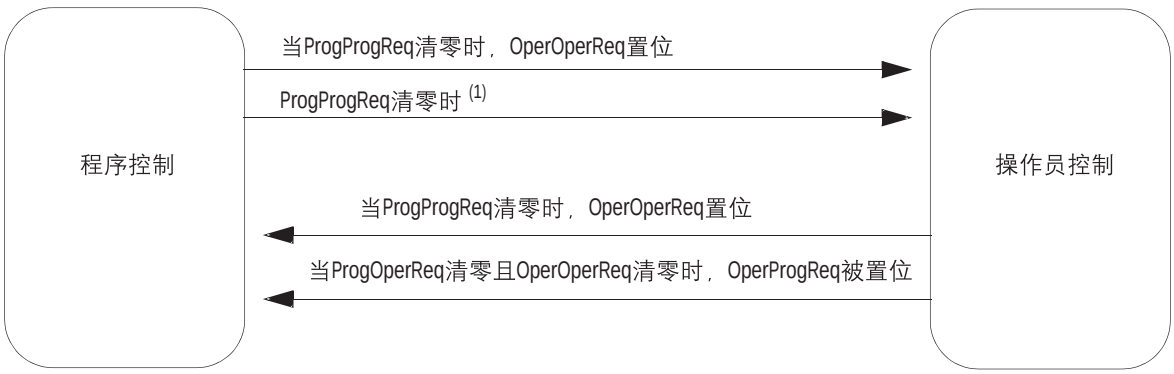
## 功能块




## 程序和操作员控制切换

PIDE指令能够通过用户程序或操作员界面来控制。用户可在任何时刻改变其控制方式。程序控制和操作员控制使用同一个ProgOper输出。当ProgOper置位时，为程序控制，ProgOper清零时，为操作员控制。

以下图表演示了PIDE指令如何在程序控制与操作员控制之间切换




(1)当ProgOperReq置位时，指令仍处于操作员控制方式

## 操作方式

PIDE指令支持以下PID方式：

| PID操作方式： | 说明： |
|---|---|
| Cascade/Ratio | 处于级联/比例方式时，指令计算CV的变化值。指令调节CV使PV靠近SPCascade 数值 或 SPCascade乘以Ratio后的数值。SPCascade来自级联控制主回路的CVEU或者比例控制回路的非受控流。<br><br>使用OperCasRatReq 或ProgCasRatReq来选择级联/比例方式：<br>置位OperCasRatReq来请求级联/比例方式。当ProgOper、ProgOverrideReq、ProgHandReq、 OperAutoReq、或 OperManualReq 置位或 AllowCasRat 清零时，请求被忽略。<br><br>置位ProgCasRatReq请求级联/比例方式。当ProgOper 或 AllowCasRat 被清零或当ProgOverrideReq、 ProgHandReq、 ProgAutoReq或 ProgManualReq 被置位时，忽略请求。 |
| Auto | 处于自动方式时，指令计算CV的变化值。指令控制CV使PV靠近SP数值。如果处于程序控制方式，SP = SPProg；处于操作员方式时，SP = SPOper。<br><br>通过置位OperAutoReq 或 ProgAutoReq来选择自动方式：<br><br>置位OperAutoReq来请求自动方式。当ProgOper、 ProgOverrideReq、ProgHandReq、或 OperManualReq 置位时，忽略该请求。<br><br>置位ProgAutoReq来请求自动方式。当ProgOper 被清零或ProgOverrideReq、ProgHandReq或 ProgManualReq被置位时，忽略该请求。 |
| Manual | 处于人工方式时，指令不计算CV的变化值。CV的值取决于控制。如果处于程序控制方式，则CV = CVProg；如果处于操作员控制方式则CV = CVOper。<br><br>通过OperManualReq 或 ProgManualReq来选择人工方式：<br><br>置位OperManualReq来请求人工方式。当ProgOper、ProgOverrideReq或 ProgHandReq置位时，忽略该请求。<br><br>置位ProgManualReq来请求人工方式。当ProgOper清零或ProgOverrideReq 或 ProgHandReq 置位时，忽略该请求。 |
| Override | 处于自动方式时，指令不计算CV的变化值。CV =CVOverride，与控制方式无关。自动方式通常是用于给PID环节设定安全状态( safe state)的。<br><br>通过ProgOverrideReq来选择自动方式：<br><br>置位ProgOverrideReq来请求自动方式。当ProgHandReq清零时，忽略该请求。 |
| Hand | 处于手动方式时，PID算法不计算CV的变化值。CV = HandFB，与控制方式无关。手动方式通常用于指示控制的最后控制单元被具有手动/自动状态的配置取代了。<br><br>通过ProgHandReq来选择手动方式：<br><br>置位ProgHandReq来请求手动方式。该值通常为数字量输入，从具有手动/自动站点读入。 |

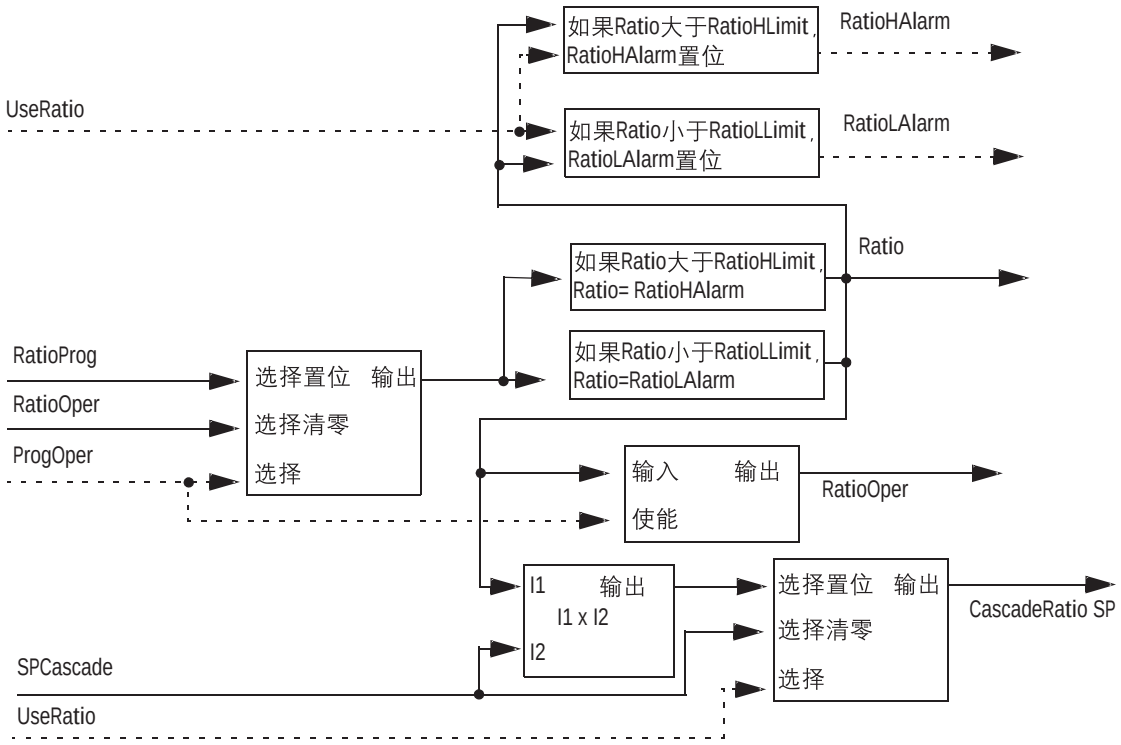
在程序控制方式下，级联/比例、自动、人工方式可由用户程序控制，在操作员控制方式可由操作员界面控制。自动或手动方式只能通过用户程序发出输入请求；这些输入操作可由程序和操作员控制。

## 选择设定点

一旦指令确定了程序或操作员控制及PID方式，指令就得到确定的SP数值。用户可选择级联/比例SP或当前SP。

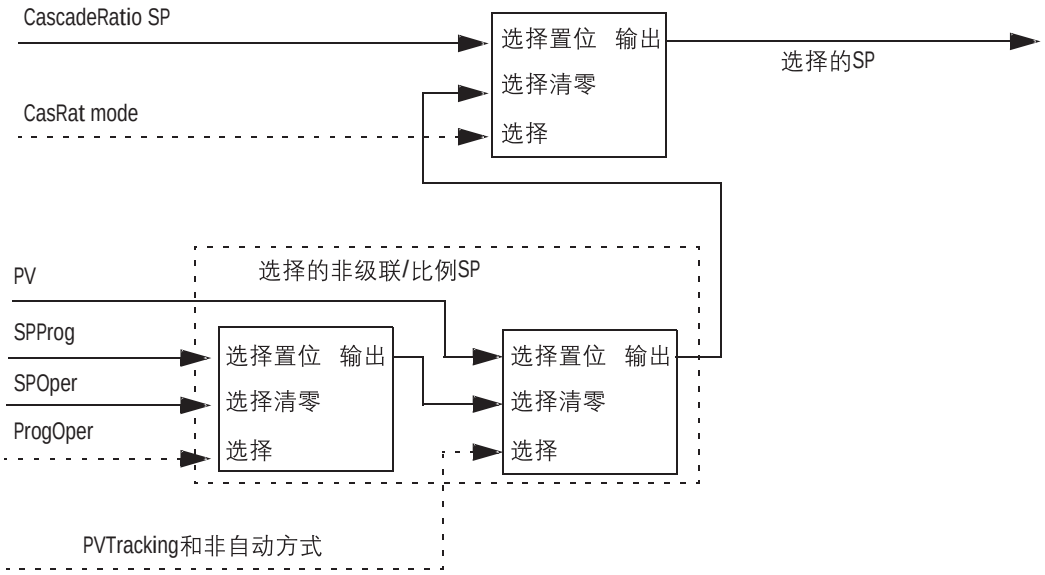
级联/比例SP

级联/比例SP取决于UseRatio和ProgOper的数值。



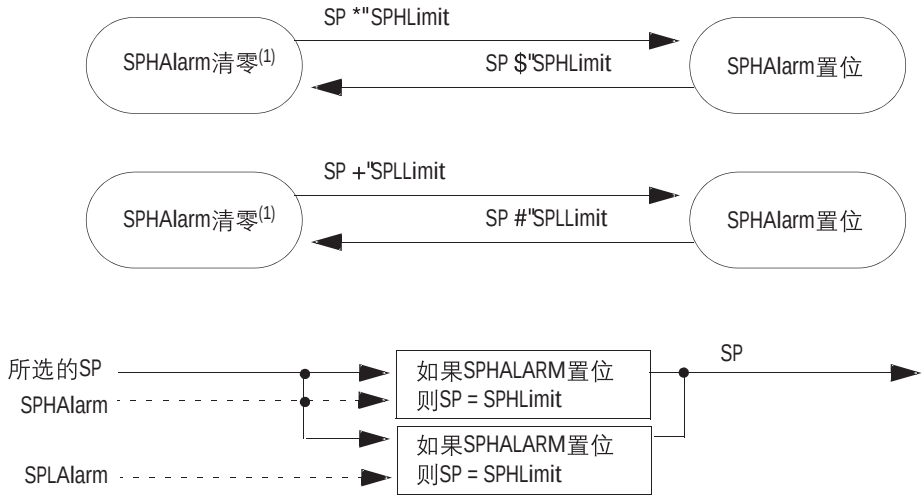
*当前SP*

当前SP取决于级联/比率方式，PVTracking数值，自动方式和ProgOper 数值。




*SP上/下限*

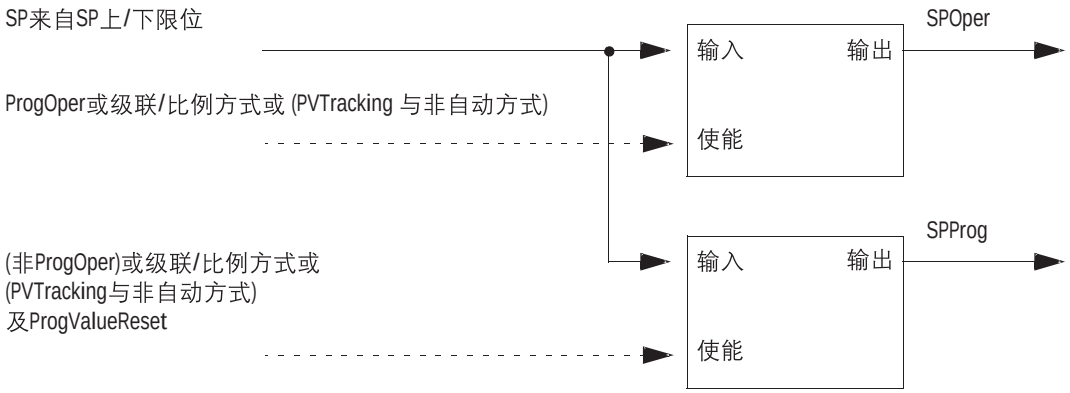
上下限算法将SP与SPHLimit和SPLLimit报警极限进行比较。SPHLimit不能大于PVEUMax且SPLLimit不能小于PVEUMin。




(1) 在指令首次扫描期间，指令清零SP报警输出。同时指令也清零SP报警极限，如果PVSpanInv置位则禁止报警算法。

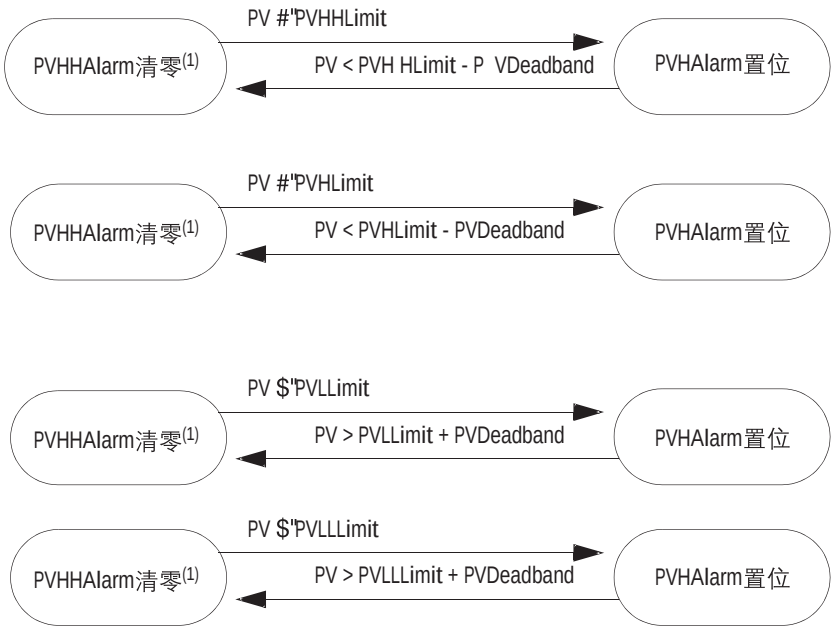
*更新 SPOper 和SPProg数值*

PIDE指令令SPOper = SP或SPProg = SP来实现程序控制和操作员控制之间的切换或从级联/比例方式切换时无冲击。




PV值上/下限报警

高上限到低下限报警算法将PV与PV报警极限及PV值加上或减去报警死区的结果进行比较。




(1) 在指令首次扫描期间，指令清零PV报警输出。同时指令也清零PV报警极限，如果PVFaulted置位则禁止报警算法。

PV变化速率报警

PV变化速率(ROC)报警将PV在PVROCPeriod周期内的变化与PV正向和负向变化速率极限进行比较。PVROCPeriod为变化速率报警提供死区。例如，如果用户使用执行周期为100ms的2?F/秒的ROC报警极限，而模拟量输入模块的分辨率为1?F，则每次输入数值改变都将产生ROC报警，因为指令认为其变化速率为10?F/秒。通过输入输入不小于1秒的PVROCPeriod，这样只有在真正达到2?F/秒时才产生ROC报警。

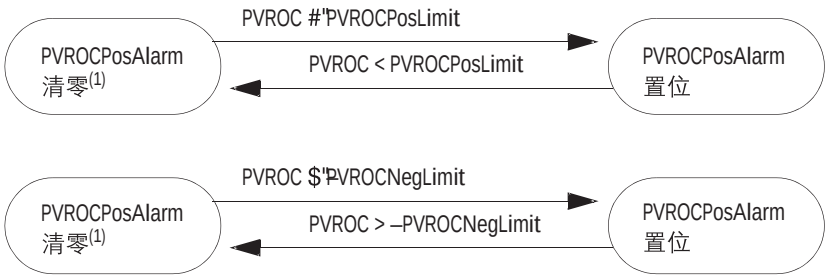
只有达到PVROCPeriod时，ROC计算才进行。变化速率计算公式如下：

ElapsedROCPeriod=ElapsedROCPeriod+ElapsedTimeSinceLastExecution

如果ElapsedROCPeriod ⩾PVROCPeriod则：

| **数值：** | **公式：** |
|---|---|
| PVROC | $\frac{PV_n - PVROC_{n-1}}{PVROCPeriod}$ |
| $PVROC_{n-1}$ | $PVROC_{n-1} = PV_n$ |
| ElapsedROCPeriod | ElapsedROCperiod = 0 |

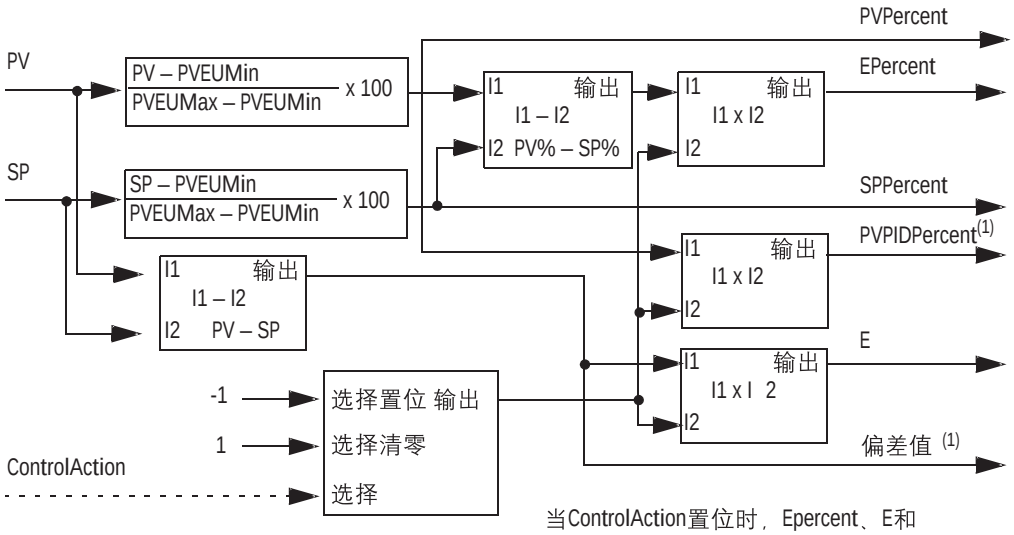
PVROC计算后，PVROC报警由以下关系决定：




(1) 在指令首次扫描期间，指令清零PVROC报警输出。指令同时也清零PVROC报警输出，如果PVFaulted置位则禁止PVROC报警算法。

## 转化PV和SP数值为百分数

指令将PV和SP数值转化为百分数并在执行PID控制算法前计算误差(error)。误差值为PV和SP之间的差值。当ControlAction置位时，则在使用PID算法之前，指令将Epercent、E 和PVPIDPercent取反。




根据ControlAction状态选择比例因子。

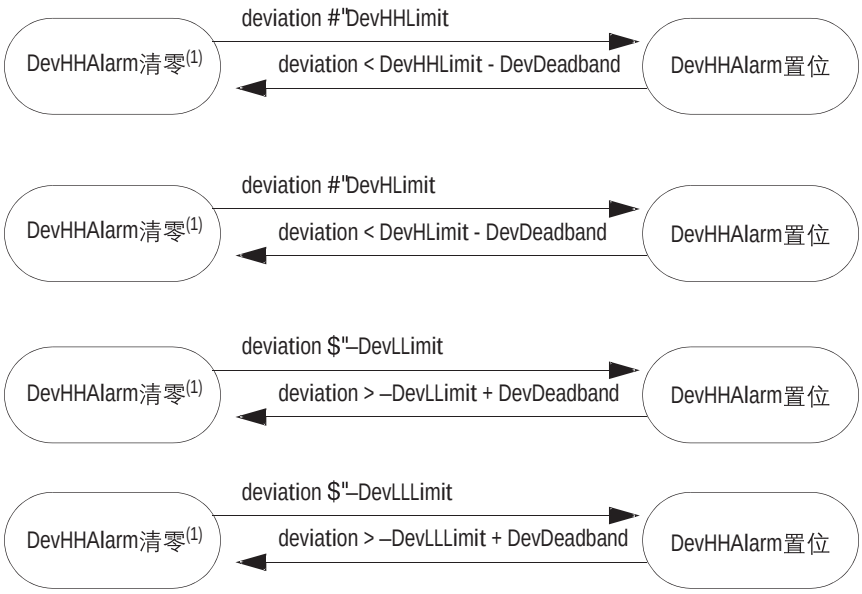
当ControlAction置位时，Epercent、E和PVPIDPercent的数值被取反。

(1) PVPIDPercent 和 Deviation 是PID控制算法所使用的内部参数。

## 偏离量上/下限报警

偏离量是过程变量PV与设定值SP之间的差值。偏离量报警提示操作员过程变量与设定值之间存在差值。

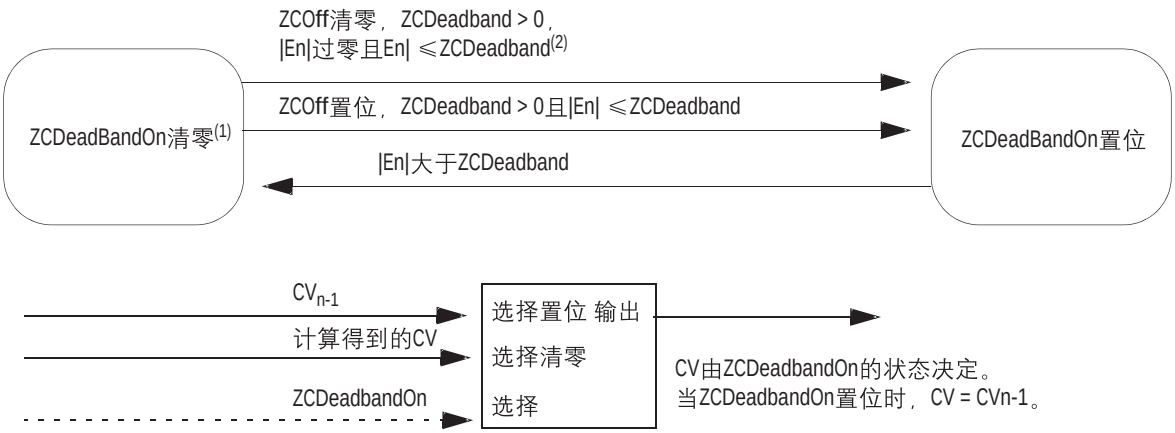
高上－低下限报警算法将偏离量与偏离量报警极限以及报警极限加上或减去死区的数值进行比较。




(1)在指令首次扫描期间，指令清零偏离量报警输出。指令同时也清零偏离量报警输出，如果PVFaulted或PVSpanInv置位则禁止报警算法。

过零死区控制

用户可设定当误差小于给定的ZCDeadband范围内CV保持不变。(| E | ≤ZCDeadband)。




(1) 当ZCOff清零，ZCDeadband > 0，误差第一次过零，(例如：En ≥0 且En-1 < 0 或En ≤0且 En-1 > 0)并且| En | ≤ZCDeadband时，指令置位ZCDeadbandOn。

(2) 当指令切换到自动或级联/比例方式时，指令设定En-1 = En。

在以下条件时，指令禁止过零算法并清零ZCDeadband：

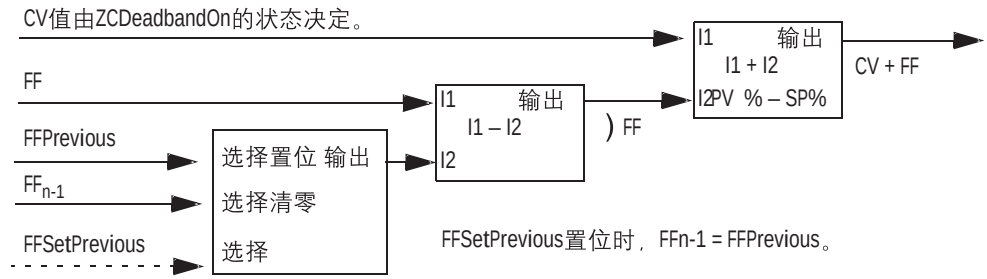
- 指令首次扫描
- ZCDeadband ??0
- 当前方式非自动或级联/比例方式
- PVFaulted置位
- PVSpanInv置位

*前馈控制*

通过对过零算法得到的CV值与?FF求和来计算CV值。其中?FF = FF-FFn-1。当FFSetPrevious置位时，FFn-1=FFPrevious。在指令计算?FF之前，允许用户预设FFn-1值。

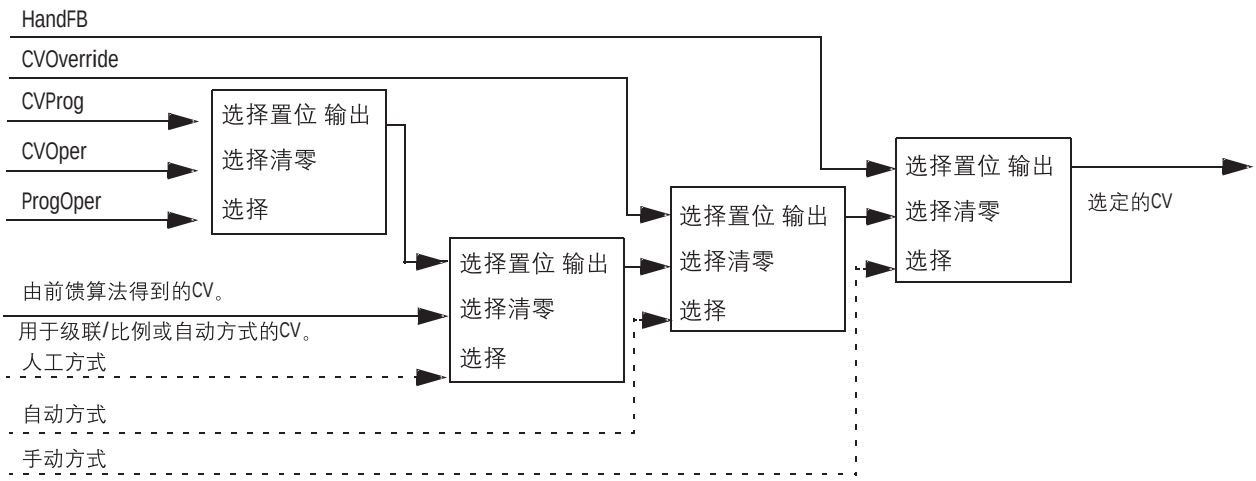
CV value based on the state of ZCDeadbandOn

Set FFn-1 = FFPrevious when FFSetPrevious is set。



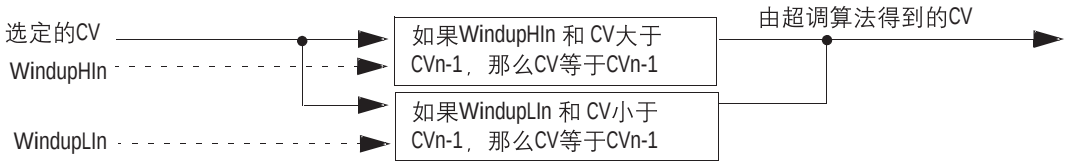

## 选择控制变量

一旦PID算法执行，就必须根据程序控制或操作员控制及当前PID方式选择CV值。



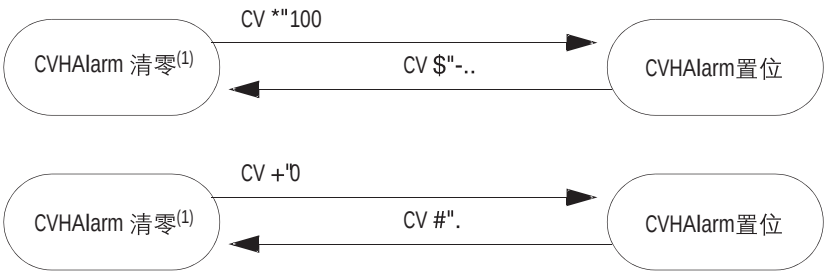
*CV超调极限*

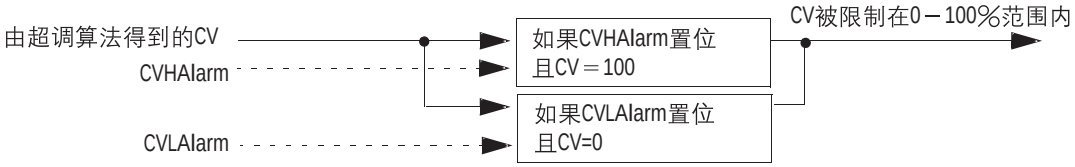
当WindupHIn置位时，限制使CV值不能增长，当WindupLIn置位时，限制使CV值不能减小。
该输入通常来自副回路的WindupHOut 或WindupLOut输出。如果CVInitializing、
CVFault或 CVEUSpanInv 被置位时，WindupHIn 和WindupLIn输入将被忽略。




*CV百分数限制*

以下图表描述了指令如何执行CV百分数限制。

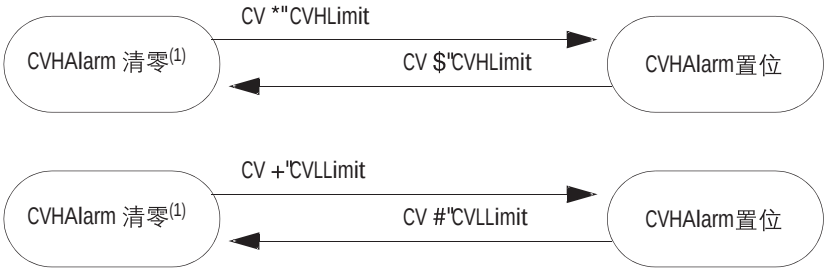


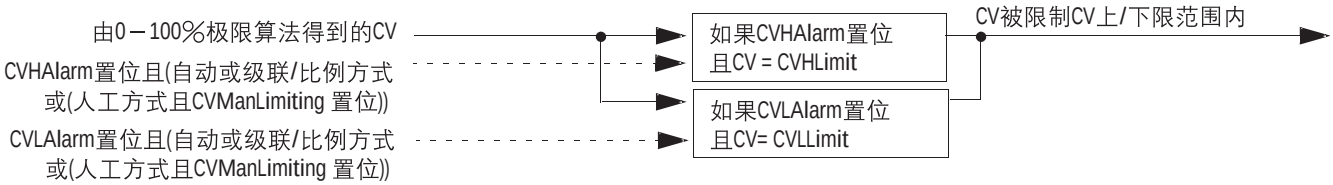




(1)在指令首次扫描期间，指令清零报警输出。

*CV值上/下限*

指令总是由CVHLimit和CVLLimit来决定报警。在自动或级联/比例方式下，CV受CVHLimit 和CVLLimit 限制。处于人工方式时，如果CVManLimiting置位，则CV受CVHLimit 和CVLLimit 限制。否则CV极限为0 到100%。




(1)在指令首次扫描期间，指令清零报警输出。

*CV值变化率限制*

在自动或级联/比例方式或在人工方式时CVManLimiting被置位，则PIDE指令限制CV的变化率。输入数值0可禁止CV变化率限制。

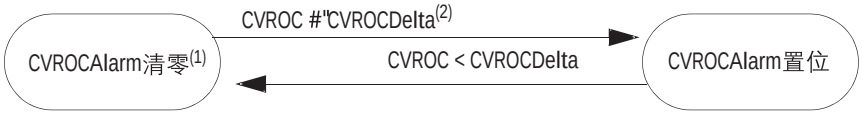
CV变化率计算公式如下：

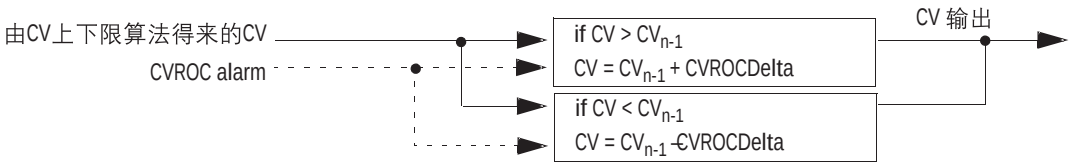
$$CVROC = |CV_n - CV_{n-1}|$$

$$CVROCDelta = CVROCLimit \times DeltaT$$

其中DeltaT以秒为单位

一旦计算出CV变化速率，CV变化速率报警由以下关系决定：



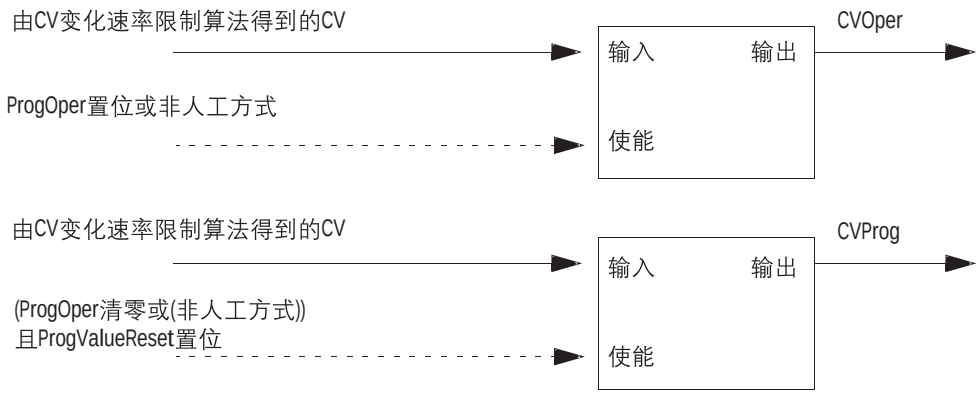




(1) 在指令首次扫描期间，指令清零报警输出。指令同时也清零报警极限，如果CVInitializing置位则禁止CV变化速率报警算法。

(2) 在自动或级联/比例方式或在人工方式时CVManLimiting被置位，则指令限制CV的变化率。

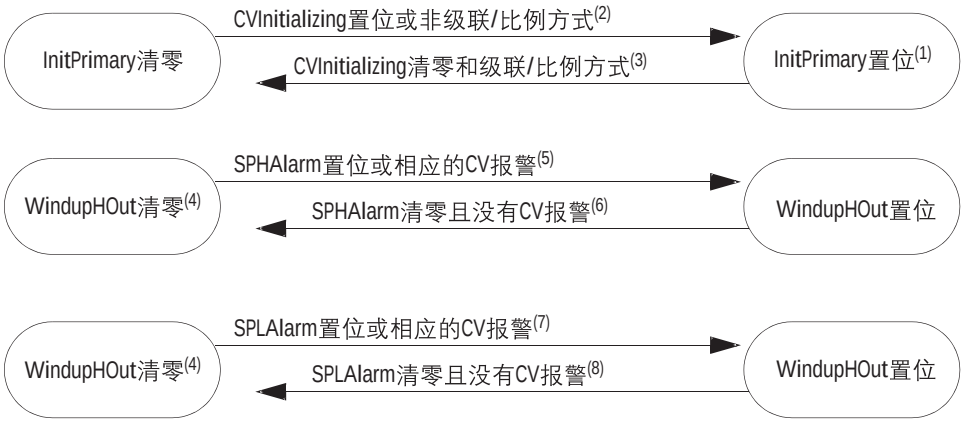
*更新CVOper和CVProg数值*

如果指令不处于操作员人工方式，则PIDE指令设定CVOper = CV。这实现了从任何控制方式到操作员人工方式的无冲击切换。



## 主回路控制

主回路控制通常应用于在使用级联/比例方式时实现无冲击转换和防止超调现象产生。主要环节控制包括初始化主PID回路输出和防止输出超调现象。InitPrimary输出通常由主PID回路的CVInitReq输入来控制。超调输出通常由主回路的超调输入控制来CV输出超调。




(1)在指令首次扫描期间，指令置位InitPrimary。

(2)当CVInitializing置位或处于非级联/比例方式，则指令置位InitPrimary。

(3)当CVInitializing清零且处于级联/比例方式，则指令清零InitPrimary。

(4)在指令首次扫描期间，指令清零输出超调状态。同时指令清零输出超调，如果CVInitializing置位或CVFaulted或CVEUSpanInv置位，则禁止CV超调算法。

(5)当SPHAlarm 置位或当ControlAction 清零且 CVHAlarm置位或ControlAction 置位且CVLAlarm置位时指令置位WindupHOut。

SP和CV的限制操作是独立的。SP上限不能禁止CV继续增长，同样CV上下限也不能禁止SP减小。

(6)当SPHAlarm清零或者非(ControlAction清零且CVHAlarm置位)或者非(ControlAction置位且CVLAlarm置位)时，指令清零WindupHOut。

(7)当SPHAlarm置位或ControlAction清零且CVHAlarm置位或ControlAction置位且CVLAlarm置位时，指令置位WindupHOut。

SP和CV限制操作是独立的。SP下限不能禁止CV继续增长，同样CV上下限也不能禁止SP减小。

(8)当SPLAlarm 清零且非 (ControlAction清零且CVLAlarm置位)且非 (ControlAction 置位且CVHAlarm 置位)时，指令清零WindupLOut。

## 故障处理

以下表格说明了指令如何进行故障处理：

| 故障条件： | 动作： |
|---|---|
| CVFaulted 置位**或**<br>CVEUSpanInv 置位 | • 指令不进行初始化，CVInitializing清零<br>• 计算PV和SP百分数值，计算误差，更新内部参数EPercent 和 PVPIDPercent<br>• PID控制算法不执行<br>• 禁止自动或级联/比例方式。如果当前不处于自动或手动方式则切换到人工方式<br>• 由程序或操作员方或方式(人工、自动或手动)来设置CV值 |
| PVFaulted置位 | • 禁止自动或级联/比例方式。如果当前不处于自动或手动方式则切换到人工方式<br>• 清零PV上/下限，PV变化速率，偏离量上/下限报警输出<br>• PID控制算法不执行<br>• 由程序或操作员方式或方式(人工、自动、手动)来确定CV值 |
| PVSpanInv置位**或**<br>SPLimitsInv 置位 | • 禁止自动或级联/比例方式。如果当前不处于自动或手动方式则切换到人工方式<br>• 不计算PV和SP的百分数值。<br>• PID控制算法不执行<br>• 由程序或操作员方式或方式(人工、自动、手动)来确定CV值 |
| RatioLimitsInv 置位**且**<br>CasRat置位**且**<br>UseRatio置位 | • 如果当前不处于自动或手动方式则切换到人工方式<br>• 禁止级联/比例方式<br>• 由程序或操作员方式或方式(人工、自动、手动)来确定CV值 |
| TimingModeInv置位**或**<br>RTSTimeStampInv 置位**或**DeltaTInv置位 | • 如果当前不处于自动或手动方式则切换到人工方式 |

# 位置比例(POSP)

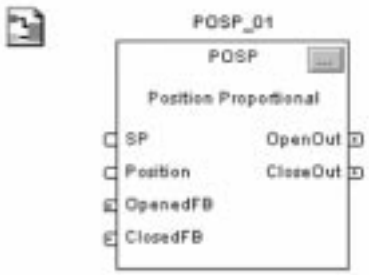
POSP以用户定义的周期时间，使用与期望位置和实际位置差值成比例的脉冲宽度周期地打开或关闭触点来打开或关闭设备。

## 操作数：




### 结构文本

| 操作数： | 类型 | 格式 | 说明 |
|---|---|---|---|
| POSP标签 | POSITION_PROP | 结构体 | POSP结构体 |




### 功能块

| 操作数： | 类型 | 格式 | 说明 |
|---|---|---|---|
| 块标签 | POSITION_PROP | 结构体 | POSP结构体 |

### POSITION_PROP结构体

| 输入参数： | 数据类型 | 说明 |
|---|---|---|
| EnableIn | BOOL | **功能块**：<br>如果该位清零，则指令不执行且输出不更新。如果置位，指令执行。<br>缺省状态为置位<br>**结构文本**：<br>无影响。指令一直执行。 |
| SP | REAL | 设定点。期望位置值。该数值工程单位与Position相同。<br>有效值＝浮点数<br>缺省值＝0.0 |
| Position | REAL | 位置反馈。从设备获得的位置反馈模拟量输入信号。<br>有效值＝浮点数<br>缺省值＝0.0 |
| OpenedFB | BOOL | 打开反馈。当设备完全打开时的输入信号。置位时，Open输出不允许开启。<br>缺省状态为清零 |
| ClosedFB | BOOL | 闭合反馈。当设备完全闭合时的输入信号。置位时，Close输出不允许开启。<br>缺省状态为清零 |
| PositionEUMax | REAL | Position和SP最大的整定值。<br>有效值＝浮点数<br>缺省值＝0.0 |

| 输入参数： | 数据类型 | 说明 |
|---|---|---|
| PositionEUMin | REAL | Position和SP最小的整定值。<br>有效值＝浮点数<br>缺省值＝0.0 |
| CycleTime | REAL | 脉冲输出周期，单位是秒。输入0可清零OpenOut 和 CloseOut。如果该数值无效，指令指定其值为0并置位状态字中的相应位。<br>有效值＝正浮点数<br>缺省值＝0.0 |
| OpenRate | REAL | 设备打开速率，以百分数每秒表示。输入0可清零OpenOut。如果该值无效，则指令指定其值为0并置位状态字中的相应位。<br>有效值＝正浮点数<br>缺省值＝0.0 |
| CloseRate | REAL | 设备闭合速率，以百分数每秒表示。输入0可清零CloseOut。如果该值无效，则指令指定其值为0并置位状态字中的相应位。<br>有效值＝正浮点数<br>缺省值＝0.0 |
| MaxOnTime | REAL | 设备最长的打开或关闭时间。如果计算出OpenTime 或 CloseTime大于该值，则被限定为该值。如果该值无效，则指令指定该值为CycleTime并置位状态字中的相应位。<br>有效值＝0.0到CycleTime<br>缺省值＝0.0 |
| MinOnTime | REAL | 设备最短的打开或关闭时间。如果计算出的OpenTime 或 CloseTime小于该值，则被限定为0。如果该值无效，则指令指定该值为0并置位状态字中的相应位。<br>有效值＝0.0到MaxOnTime<br>缺省值＝0.0 |
| Deadtime | REAL | 克服设备摩擦力的额外脉冲时间，单位为秒。当设备改变方向或处于停止状态时，死区时间就加到OpenTime 或 CloseTime。如果该值无效，则指令置位状态字中的相应位，并使Deadtime = 0.0。<br>有效值＝0.0到MaxOnTime<br>缺省值＝0.0 |

| 输出参数： | 数据类型： | 说明： |
|---|---|---|
| EnableOut | BOOL | 输出使能。 |
| OpenOut | BOOL | 打开设备的脉冲输出。 |
| CloseOut | BOOL | 闭合设备的脉冲输出。 |
| PositionPercent | REAL | 以位置范围百分数表示的位置反馈。该输出影响算术状态标志。 |
| SPPercent | REAL | 以位置范围百分数表示的设定点。 |
| OpenTime | REAL | 当前周期的OpenOutput的脉冲时间，以秒为单位。 |
| CloseTime | REAL | 当前周期的CloseOutput的脉冲时间，以秒为单位。 |
| Status | DINT | 功能块状态字。 |
| InstructFault (Status.0) | BOOL | 指令检测以下的运行错误。该错误不使控制器产生次要或主要错误。检测其它状态位以确定所发生故障类型。 |
| CycleTimeInv (Status.1) | BOOL | 无效CycleTime时间数值。指令使用数值0。 |

| 输出参数： | 数据类型： | 说明： |
|---|---|---|
| OpenRateInv (Status.2) | BOOL | 无效OpenRate数值。指令使用数值0。 |
| CloseRateInv (Status.3) | BOOL | 无效CloseRate数值。指令使用数值0。 |
| MaxOnTimeInv (Status.4) | BOOL | 无效MaxOnTime数值。指令使用CycleTime数值。 |
| MinOnTimeInv (Status.5) | BOOL | 无效MinOnTime数值。指令使用数值0。 |
| DeadtimeInv (Status.6) | BOOL | 无效Deadtime数值。指令使用数值0。 |
| PositionPctInv (Status.7) | BOOL | 计算的PositionPercent结果超出范围。 |
| SPPercentInv (Statius.8) | BOOL | 计算的SPPercent结果超出范围。 |
| PositionSpanInv (Status.9) | BOOL | PositionEUMax = PositionEUMin。 |

**说明：**POSP指令通常用于从PID指令输出接收期望的位置设定值。

## 位置和设定点数值整定

每次指令执行时，更新PositionPercent 和 SPPercent输出。如果其中一个数超出范围(小于0%或者大于100%)，则置位状态字中的相应位，但不限制该值。指令通过以下公式判断该数值是否在范围内：

$$\text{PositionPerent}=\frac{\text{Position-PositionEUMin}}{\text{PositionEUMax-PositionEUMin}}\times 100$$

$$\text{SPPerent}=\frac{\text{SP-PsitionEUMin}}{\text{PositionEUMax-PositionEUMin}}\times 100$$

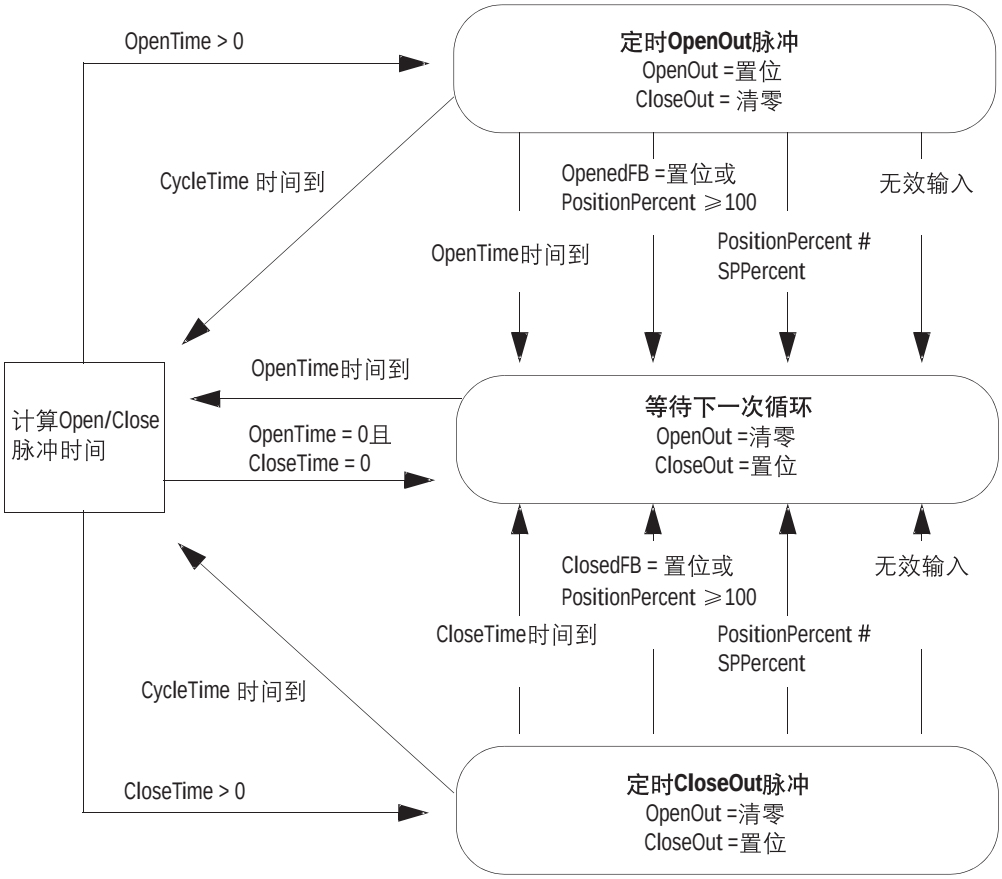
## POSP指令如何使用内部周期定时器

指令使用CycleTime来决定多长时间重新计算Open 和Close 输出脉冲的持续时间。内部定时器由DeltaT来维护和更新。DeltaT是从上次指令执行到当前所消耗的时间。当内部定时器达到或超过给定的CycleTime(周期时间到)则重新计算打开和闭合输出。

用户可随时改变CycleTime数值。
如果CycleTime = 0，则内部定时器清零，OpenOut 和CloseOut也清零。

## 产生输出脉冲

以下图表演示了POSP指令的3个主要状态。



## 脉宽计算

当SP >位置反馈时，OpenOut有脉冲输出。这时指令设置CloseTime = 0，OpenOut打开持续时间计算如下：

$$\text{OpenTime} = \frac{\text{SPPercent-PositionPercent}}{\text{OpenRate}}$$

- 如果OpenTimen-1 < CycleTime，则OpenTime加上Deadtime。
- 如果OpenTime > MaxOnTime，则设定其值为MaxOnTime。
- 如果OpenTime < MinOnTime，则设定其值OpenTime = 0。

如果有以下条件存在，OpenOut没有脉冲输出且OpenTime = 0。

- OpenFB置位或PositionPercent ⩾100
- CycleTime = 0
- OpenRate = 0
- SPPercent无效

当SP <位置反馈时，CloseOut有脉冲输出。这时指令设置OpenTime = 0，CloseOut打开持续时间计算如下：

$$\text{OpenTime} = \frac{\text{PositionPercent-SPPercent}}{\text{CloseRate}}$$

- 如果CloseTimen-1 < CycleTime，则CloseTime加上Deadtime。
- 如果CloseTime > MaxOnTime，则设定极限值为MaxOnTime。
- 如果CloseTime < MinOnTime，则设定数值CloseTime = 0。

如果有以下条件存在，CloseOut没有脉冲输出且CloseTime被清零。

- CloseFB置位或PositionPercent??0
- CycleTime = 0
- CloseRate = 0
- SPPercent无效

当SPPercent 等于PositionPercent时，OpenOut 和CloseOut没有输出。OpenTime 和 CloseTime都被清零。

**算术状态标志：** PositionPercent输出影响算术状态标志。

**故障条件：** 无

**执行：**

| 条件: | 功能块动作: | 结构文本动作： |
|---|---|---|
| 预扫描 | OpenOut 和CloseOut 清零<br>OpenTime = 0<br>CloseTime = 0. | OpenOut 和CloseOut 清零<br>OpenTime = 0<br>CloseTime = 0. |
| 指令首次扫描 | 内部定时器清零。<br>指令计算OpenTime及Close Time。 | 内部定时器清零。<br>指令计算OpenTime及Close Time。 |
| 指令首次执行 | 无动作 | 无动作 |
| 输入使能(EnableIn)清零 | 输出使能(EnableOut)清零。 | 不影响 |
| 输入使能(EnableIn)置位 | 执行指令。<br>输出使能(EnableOut)置位 | 输入使能(EnableIn)一直置位。<br>执行指令 |
| 后期扫描 | 无动作 | 无动作 |

**实例：** 在此例中，POSP指令根据PIDE指令的CVEU输出来打开或关闭一个电动控制阀门。实际阀门位置连接到Position输入上，用来指示阀门完全打开或完全关闭的限位开关是可选的，连接到OpenedFB 和ClosedFB输入上。OpenOut和CloseOut 输出被连接到电动操作阀的打开触点和闭合触点上。

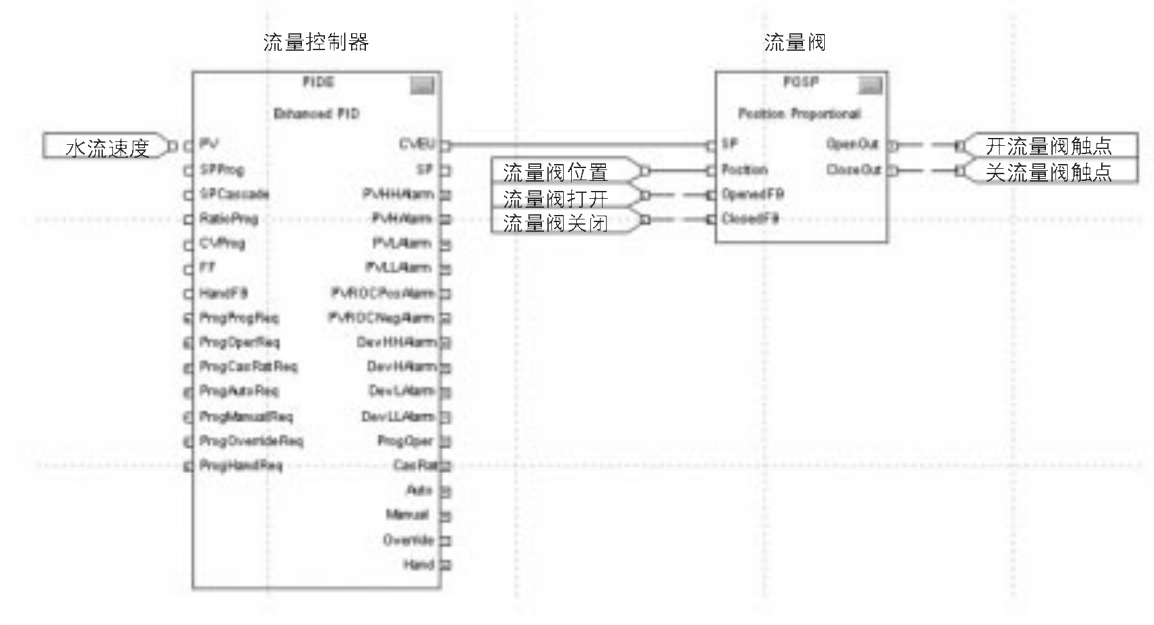
### 结构文本

```
FlowController.PV := WaterFlowRate;
PIDE(FlowController);

FlowValve.SP := FlowController.CVEU;
FlowValve.Position := FlowValvePosition;
FlowValve.OpenedFB := FlowValveOpened;
FlowValve.ClosedFB := FlowValveClosed;
POSP(FlowValve);

OpenFlowValveContact := FlowValve.OpenOut;
CloseFlowValveContact := FlowValve.CloseOut;
```

## 功能块


流量控制器
流量阀
PIDE
Enhanced PID
PV
SPProg
SPCascade
RatioProg
CVProg
FF
HandFB
ProgProgReq
ProgOperReq
ProgCasRatReq
ProgAutoReq
ProgManualReq
ProgOverrideReq
ProgHandReq
CVEU
SP
PVHHAlarm
PVHAlarm
PVLAlarm
PVLLAlarm
PVROCPosAlarm
PVROCNegAlarm
DevHHAlarm
DevHAlarm
DevLAlarm
DevLLAlarm
ProgOper
CasRat
Auto
Manual
Override
Hand
水流速度
POSP
Position Proportional
SP
Position
OpenedFB
ClosedFB
OpenOut
CloseOut
流量阀位置
流量阀打开
流量阀关闭
开流量阀触点
关流量阀触点

## 斜坡/渗透(RMPS)

RMPS指令提供一定数量的交互的斜坡和渗透时间段。

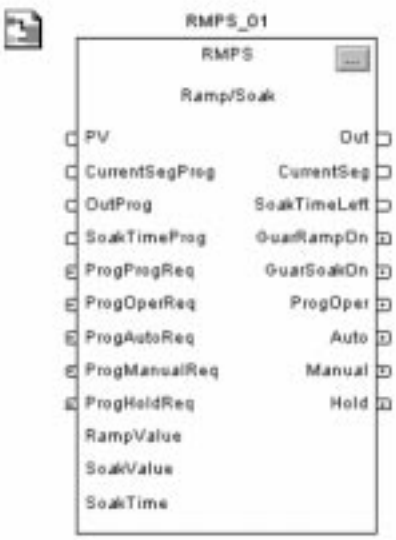
**操作数：**




### 结构文本

| 操作数： | 类型： | 格式： | 说明： |
|---|---|---|---|
| RMPS标签 | RAMP_SOAK | 结构体 | RMPS结构体 |
| RampValue | REAL | 数组 | Ramp数组。输入每一段的斜率值(0到段数－1)。斜率输入值以分钟为单位或是速度/分钟。TimeRate参数反映出用什么方法来指定斜率。如果斜率值无效，指令将置位状态字中的相应位并切换到操作员人工方式或程序保持方式。数组应不小于分段数。<br>有效值＝0.0到最大正浮点数 |
| SoakValue | REAL | 数组 | 渗透值数组。输入每段的渗透值(0到段数－1)。数组最小应不小于分段数。<br>有效值＝浮点数 |
| SoakTime | REAL | 数组 | 渗透时间数组。输入每一段的渗透时间(0到段数－1)。斜率输入值以分钟为单位。如果斜率值无效，指令将置位状态字中的相应位并切换到操作员人工方式或程序保持方式。数组最小应不小于分段数。<br>有效值＝0.0到最大正浮点数 |

### 功能块




该操作数与结构文本的RMPS指令相同。

## RAMP_SOAK结构体：

| 输入参数： | 数据类型： | 说明： |
|---|---|---|
| EnableIn | BOOL | 功能块：<br>如果该位清零，则指令不执行输出不更新。<br>如果置位，指令执行。<br>缺省状态为置位<br>结构文本：<br>无影响。指令一直执行。 |
| PV | REAL | 整定指令的模拟量温度信号输入<br>有效值＝浮点数<br>缺省值＝0.0 |
| PVFault | BOOL | PV故障指示。置位时，如果输入无效，则指令切换到程序保持方式或操作员人工方式并置位状态字中的相应位。<br>缺省状态为清零 |
| NumberOfSegs | DINT | 分段数。指定指令用到的斜坡/渗透段数。RampValue， SoakValue，和 SoakTime 数组的尺寸应不小于分段数。如果该数值无效，则指令切换到操作员人工方式或程序保持方式并置位状态字中的相应位。<br>有效值＝1到(RampValue，SoakValue，或 SoakTime 数组的最小尺寸)<br>缺省值＝1 |
| ManHoldAftInit | BOOL | 初始化后进入人工/保持方式。置位时，初始化完成后斜坡/渗透进入操作员人工方式或程序保持方式。否则，初始化完成后斜坡/渗透保持前一状态。<br>缺省状态为清零 |
| CyclicSingle | BOOL | 周期/单次执行。置位时按周期执行，清零时单次执行。周期执行不断地重复执行斜坡/渗透功能。单独执行则只执行一次就停止。<br>缺省状态为清零 |
| TimeRate | BOOL | 配置时间/速率数值。如果RampValue参数输入的是达到渗透温度的以分钟为单位时间，则置位。如果RampValue参数输入的是以单位/分钟的速率，则清零。<br>缺省状态为清零 |
| GuarRamp | BOOL | 保障斜率。如果该位置位且指令处于自动方式，则PV与输出差值大于RampDeadband也暂时停止调节。<br>缺省状态为清零 |
| RampDeadband | REAL | 保障斜率死区。当GuarRamp置位时，指定允许的PV与输出之间的工程单位差值。如果该值无效，指令设定RampDeadband = 0.0并置位状态字中的相应位。<br>有效值＝大于等于0的浮点数<br>缺省值＝0.0 |
| GuarSoak | REAL | 保障渗透值。如果置位且指令处于自动方式，当PV与输出差值大于SoakDeadband时，渗透计时器清零。<br>缺省状态为清零 |
| SoakDeadband | REAL | 保障渗透死区。当GuarSoak置位时，指定允许的PV与输出之间的工程单位差值。如果该值无效，指令设定SoakDeadband = 0.0并置位状态字中的相应位。<br>有效值＝大于等于0的浮点数<br>缺省值＝0.0 |

| 输入参数： | 数据类型： | 说明： |
|---|---|---|
| CurrentSegProg | DINT | 当前程序段。用户程序写入请求的CurrentSeg段数到该输入中。该数值在斜坡/渗透处于程序人工方式时使用。如果该数值无效，则指令置位状态字中的相应位。<br>有效值＝0到分段数－1<br>缺省值＝0 |
| OutProg | REAL | 程序输出。用户程序向该输入中写入所要求输出的数值。该数值在斜坡/渗透处于程序人工方式时使用。<br>有效值＝浮点数<br>缺省值＝0 |
| SoakTimeProg | REAL | 程序渗透时间。用户程序向该输入中写入所要求的SoakTime的数值。该数值在斜坡/渗透处于程序人工方式时使用。如果该数值无效，则指令置位状态字中的相应位。<br>有效值＝0.0到最大正浮点数<br>缺省值＝0 |
| CurrentSegOper | DINT | 当前操作员段数。操作员界面写入请求的CurrentSeg段数到该数中。该数值在斜坡/渗透处于操作员人工方式时使用。如果该数值无效，则指令置位状态字中的相应位。<br>有效值＝0到分段数－1<br>缺省值＝0 |
| OutOper | REAL | 操作员输出。操作员界面往该输入中写入所要求输出的数值。该数值在斜坡/渗透处于操作员人工方式时使用。<br>有效值＝浮点数<br>缺省值＝0 |
| SoakTimeOper | REAL | 操作员渗透时间。操作员界面往该输入中写入所要求的SoakTimeLeft的数值。<br>该数值在斜坡/渗透处于操作员人工方式时使用。如果该数值无效，则指令置位状态字中的相应位。<br>有效值＝0.0到最大正浮点数<br>缺省值＝0 |
| ProgProgReq | BOOL | 程序编程请求。用户编程置位该位请求程序控制。 当ProgOperReq置位时该请求被忽略。在程序控制方式下，保持置位且通过ProgOperReq清除指令锁定。<br>缺省状态为清零 |
| ProgOperReq | BOOL | 程序操作员请求。操作员界面置位该位请求操作员控制。在操作员控制方式中，保持该位置位可锁定指令。<br>缺省状态为清零 |
| ProgAutoReq | BOOL | 程序自动方式请求。由用户程序置位请求斜坡/渗透进入自动方式。如果该环节处于操作员控制或当ProgManualReq置位或当ProgHoldReq置位，则忽略该请求。<br>缺省状态为清零 |
| ProgManualReq | BOOL | 程序人工方式请求。由用户编程置位请求斜坡/渗透进入人工方式。如果该环节处于操作员控制或当ProgHoldReq置位时，则忽略该请求。<br>缺省状态为清零 |
| ProgHoldReq | BOOL | 程序保持方式请求。由用户程序置位请求停止斜坡/渗透保持Out、CurrentSeg或SoakTimeLeft不变。<br>也有助于PID环节从斜坡/渗透离开级联状态后达到设定点。操作员将斜坡/渗透切换到操作员人工方式也可实现相同的功能。<br>缺省状态为清零 |
| OperProgReq | BOOL | 操作员界面程序请求。通过操作员界面请求程序控制。如果ProgOperReq置位，则忽略该请求。指令清零该输入。<br>缺省状态为清零。 |

| 输入参数： | 数据类型： | 说明： |
|---|---|---|
| OperOperReq | BOOL | 操作员界面操作请求。通过操作员界面请求操作员控制。如果ProgProgReq 置位且ProgOperReq清零，则忽略该请求。指令清零该输入。<br>缺省状态为清零 |
| OperAutoReq | BOOL | 操作员界面自动方式请求。由操作员界面置位请求斜坡/渗透进入自动方式。如果该回路处于程序控制或当OperManualReq置位时，则忽略该请求。指令清零该输入。<br>缺省状态为清零 |
| OperManualReq | BOOL | 操作员界面人工方式请求。由操作员界面置位请求斜坡/渗透进入人工方式。如果该环节处于程序控制时，则忽略该请求。指令清零该输入<br>缺省状态为清零 |
| Initialize | BOOL | 初始化程序和操作员数值。如果该位置位且处于人工方式，则指令设定CurrentSegProg = 0，CurrentSegOper=0，SoakTimeProg=SoakTime[0]和SoakTimeOper = SoakTime[0]。处于自动或保持方式时初始化请求被忽略。指令清零该参数。<br>缺省状态为清零 |
| ProgValueReset | BOOL | 复位程序控制数值。置位时，指令清零ProgProgReq，ProgOperReq，ProgAutoReq，ProgHoldReq和ProgManualReq。<br>缺省状态为清零 |

| 输出参数： | 数据类型： | 说明： |
|---|---|---|
| EnableOut | BOOL | 输出使能。 |
| Out | REAL | 斜坡/渗透指令输出。该输出影响算术状态标志。 |
| CurrentSeg | DINT | 当前分段数。显示斜坡/渗透周期的当前分段数。分段数起始于零。 |
| SoakTimeLeft | REAL | 剩余渗透时间。显示当前渗透所剩余的渗透时间。 |
| GuarRampOn | BOOL | 保障斜率状态。当使用保障斜率参数且斜率因PV与输出差值大于RampDeadband暂缓计算时置位。 |
| GuarSoakOn | BOOL | 保障渗透状态。当使用保障渗透参数且渗透定时器因PV与输出差值大于SoakDeadband而清零时置位。 |
| ProgOper | BOOL | 程序/操作员控制指示。处于程序控制时置位，操作员控制时清零。 |
| Auto | BOOL | 自动方式。当斜坡/渗透处于程序自动方式或操作员自动方式时置位。 |
| Manual | BOOL | 人工方式。当斜坡/渗透处于程序人工方式或操作员人工方式时置位。 |
| Hold | BOOL | 保持方式。当斜坡/渗透处于程序保持方式时置位。 |
| Status | DINT | 功能块状态字。 |
| InstructFault (Status.0) | BOOL | 指令检测以下的运行错误。该错误不使控制器产生次要或主要错误。检测其它状态位以确定所发生故障类型。 |
| PVFaulted (Status.1) | BOOL | PV状态不正常。 |
| NumberOfSegsInv (Status.2) | BOOL | 当前分段数无效或与数组尺寸不匹配。 |
| RampDeadbandInv (Status.3) | BOOL | RampDeadband数值无效。 |

| 输出参数： | 数据类型： | 说明： |
|---|---|---|
| SoakDeadbandInv (Status.4) | BOOL | SoakDeadband数值无效。 |
| CurrSegProgInv (Status.5) | BOOL | CurrSegProg数值无效。 |
| SoakTimeProgInv (Status.6) | BOOL | SoakTimeProg数值无效。 |
| CurrSegOperInv (Status.7) | BOOL | CurrSegOper数值无效。 |
| SoakTimeOperInv (Status.8) | BOOL | SoakTimeOper数值无效。 |
| RampValueInv (Status.9) | BOOL | RampValue数值无效。 |
| SoakTimeInv (Status.10) | BOOL | SoakTime数值无效。 |

**说明：** RMPS指令通常用于为炉的加热过程提供温度调节。该指令的输出通常连接到PID回路的设定点输入。

当输出计算值无效时，NAN或±INF，指令设定Out =该无效值并置位算术溢出状态标志。内部参数不再更新。在后续的扫描过程中，当输出有效时，输出通过前次扫描得到的内部参数计算得出。

## 监控RMPS指令

RMPS具有操作员面板。如需要了解更多的信息请参阅附录：功能块面板控制。

**算术状态标志：**该输出影响算术状态标志。

**故障条件：**无

**执行：**

| 条件： | 功能块动作： | 结构文本动作： |
|---|---|---|
| 预扫描 | 无动作。 | 无动作。 |
| 指令首次扫描 | 清零所有操作员输入请求。<br>如果ProgValueReset置位，则清零所有程序输入请求。如果当前方式是保持方式，则设置为人工方式。<br>见下表。 | |
| 指令首次执行 | CurrentSegment = 0。<br>如果SoakTime[0]有效，则SoakTimeProg 和 SoakTimeOper= SoakTime[0]<br>方式为操作员人工方式。<br>Outn-1 = 0.0 | |
| 输入使能(EnableIn)清零 | 输出使能(EnableOut)清零。 | 不影响 |
| 输入使能(EnableIn)置位 | 执行指令。<br>输出使能(EnableOut)置位 | 使能输入一直置位。<br>指令执行。 |
| 后期扫描 | 无动作 | 无动作 |

## 指令首次扫描应用的初始方式

下表显示了由程序输入请求决定的最后控制方式。

| 首次扫描初始控制方式： | Prog Oper Req： | Prog Prog Req： | Prog Value Reset： | 首次运行 | 首次扫描最终时控制方式： |
|---|---|---|---|---|---|
| 操作员控制 | 清零 | 置位 | 清零 | 不影响 | 程序控制 |
| | 不影响 | 清零 | 不影响 | 不影响 | 操作员控制 |
| 程序控制 | 置位 | 不影响 | 清零 | 清零 | 操作员控制 |
| | 不影响 | 不影响 | 置位 | 置位 | |
| | 清零 | 清零 | 清零 | 置位 | |
| | 清零 | 置位 | 清零 | 不影响 | |
| | 不影响 | 不影响 | 置位 | 清零 | |
| | 清零 | 清零 | 清零 | 清零 | |

下表给出了由手动、自动和保持方式请求决定的最终控制方式。

| 首次扫描初始控制方式： | Oper Auto Req: | Oper Man Req: | Prog Auto Req: | Prog Man Req: | Prog Hold Req: | Manual Hold After Init: | Prog Value Reset: | 首次运行 | 首次扫描最终控制方式： |
|---|---|---|---|---|---|---|---|---|---|
| 操作员控制 | 不影响 | 不影响 | 不影响 | 不影响 | 不影响 | 清零 | 不影响 | 清零 | 操作员当前方式 |
| | 不影响 | 不影响 | 不影响 | 不影响 | 不影响 | 不影响 | 不影响 | 置位 | 操作员手动方式 |
| | 不影响 | 不影响 | 不影响 | 不影响 | 不影响 | 置位 | 不影响 | 不影响 | |
| 程序 | 不影响 | 不影响 | 清零 | 清零 | 清零 | 清零 | 不影响 | 清零 | 程序当前方式 |
| 控制 | 不影响 | 不影响 | 不影响 | 不影响 | 不影响 | 清零 | 置位 | 清零 | |
| | 不影响 | 不影响 | 置位 | 清零 | 清零 | 清零 | 清零 | 不影响 | 程序自动方式 |
| | 不影响 | 不影响 | 不影响 | 置位 | 清零 | 清零 | 清零 | 不影响 | 程序人工方式 |
| | 不影响 | 不影响 | 不影响 | 不影响 | 置位 | 清零 | 清零 | 不影响 | 程序保持方式 |
| | 不影响 | 不影响 | 不影响 | 不影响 | 不影响 | 置位 | 不影响 | 不影响 | |

**实例：** 在此例中，RMPS指令为提供PIDE指令的设定点。当PIDE指令处于级联/比例方式时，RMPS指令的输出被作为设定点。如果用户想使用保障斜率或保障渗透时，PIDE指令的PV可选择连接到RMPS指令的PV上。

在此例中，AutoclaveRSSoak值、AutoclaveRSSoakTime和AutoclaveRSRamp值数组都是10个元素的REAL数组，最多支持10个分段的RMPS功能。

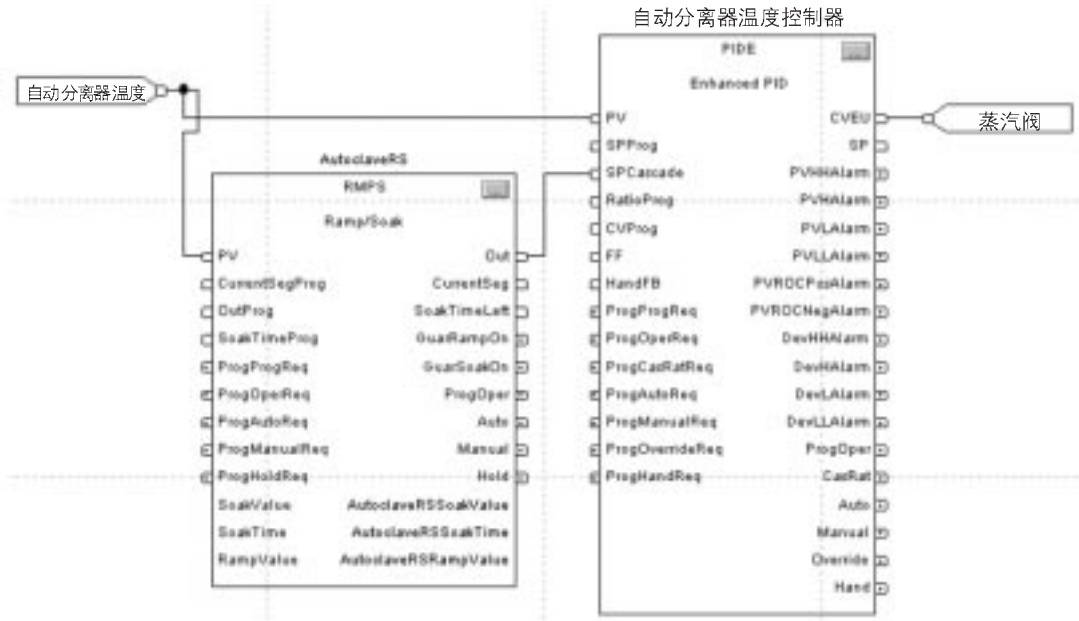
### 结构文本

```
AutoclaveRS.PV := AutoclaveTemp;
RMPS (AutoclaveRS,AutoclaveRSRampValue,
AutoclaveRSSoakValue,AutoclaveRSSoakTime);

AutoclaveTempController.PV := AutoclaveTemp;
AutoclaveTempController.SPCascade :=AutoclaveRS.Out;
PIDE(AutoclaveTempController);

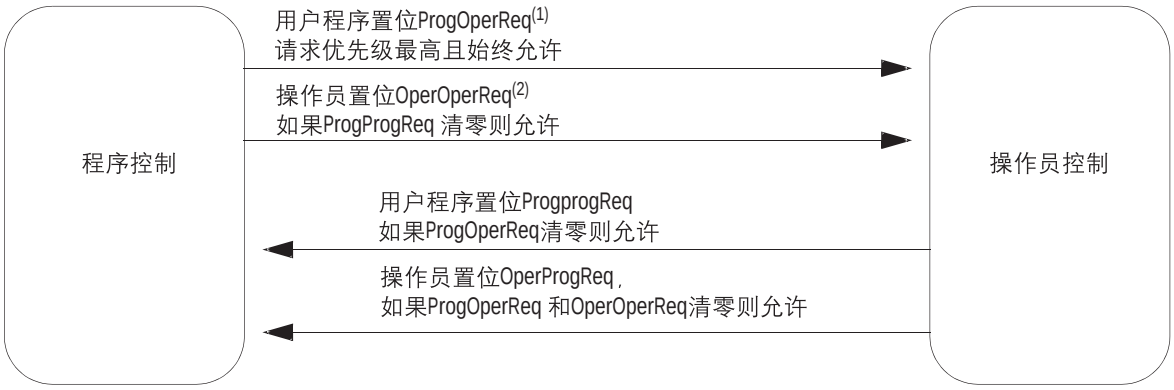
SteamValve := AutoclaveTempController.CVEU;
```

## 功能块




## 程序控制和操作员控制切换

RMPS指令可以通过程序或操作员界面进行控制。可随时改变控制方式。




(1) 保持ProgOperReq置位可将指令锁定为操作员控制方式。

(2) 当ProgOperReq清零时保持PorgProgReq可将指令锁定为程序控制方式

当ProgAutoReq、ProgManualReq和ProgHoldReq输入为清零时，指令从操作员控制切换到程序控制，其方式由以下条件决定：

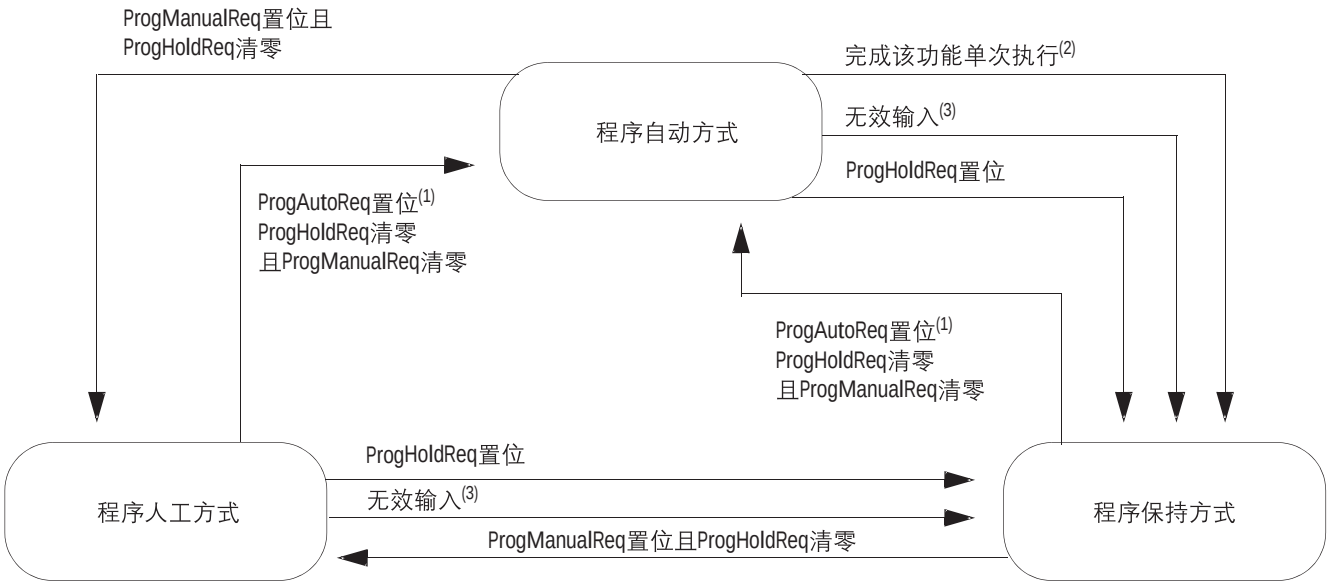
- 如果指令处于操作员自动方式，则切换到程序自动方式。
- 如果指令处于操作员人工方式，则切换到程序人工方式。

当OperAutoReq和OperManualReq输入为清零时，指令从程序控制切换到操作员控制，其方式由以下条件决定：

- 如果指令处于程序自动方式，则切换到操作员自动方式。
- 如果指令处于程序人工或程序保持方式，则切换到操作员人工方式。

## 程序控制

以下图表演示了在程序控制方式下，RMPS指令如何操作。




(1) 在单次(非周期)执行时，如果已经有一个斜坡/渗透执行完成，且用户要执行另一次，则需将ProgAutoReq状态由清零转换到置位。

(2) 如果指令被配置为单次执行且自动方式的斜坡/渗透功能已完成，则指令进入保持方式。

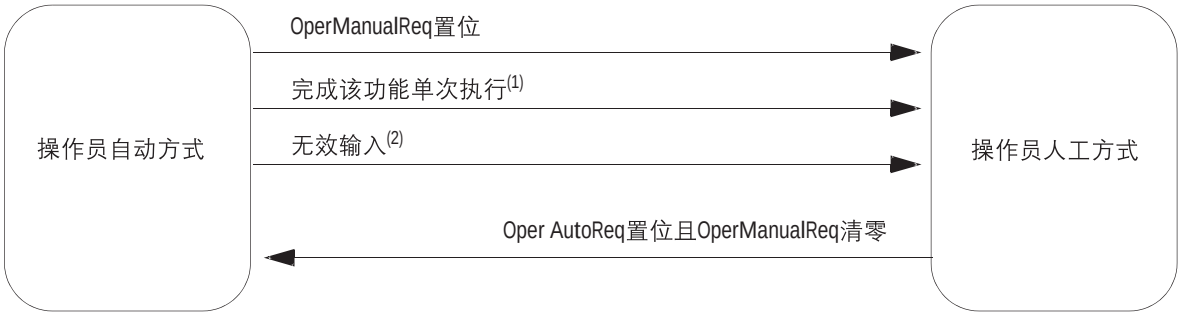
(3) 如果PVFaulted置位或者以下输入无效：NumberOfSegs、CurrentSeg、SoakTimeLeft、CurrentSegProg或SoakTimeProg，则指令进入保持方式。

下表说明了可能出现的程序方式：

| 方式： | 说明： |
|---|---|
| 程序自动方式 | 在自动方式，指令连续地执行斜坡/渗透功能。 |
| 程序人工方式 | 在程序人工方式，用户程序直接控制指令输出。CurrentSegProg、SoakTimeProg和OutProg输入被转化到CurrentSeg、SoakTimeLeft和Out输出。当指令处于自动方式，斜坡/渗透函数从用户程序前次输入值重新开始。如果CurrentSegProg或SoakTimeProg无效则不被切换。<br><br>为了实现无冲击地切换到人工方式，当ProgValueReset置位且指令不处于程序人工方式时，CurrentSegProg、SoakTimeProg和OutProg输入不断地被当前的CurrentSeg、SoakTimeLeft和Out输出所更新。 |
| 程序保持方式 | 处于程序保持方式时，指令的输出保持当前数值不变。在此方式下，如果ProgOperReq被置位来请求操作员控制，则指令切换到操作员人工方式。 |

## 操作员控制

以下图表演示了在操作员控制方式下，RMPS指令如何操作。




(1) 如果指令被配置为单次执行且自动方式的斜坡/渗透功能已完成，则指令进入人工方式。

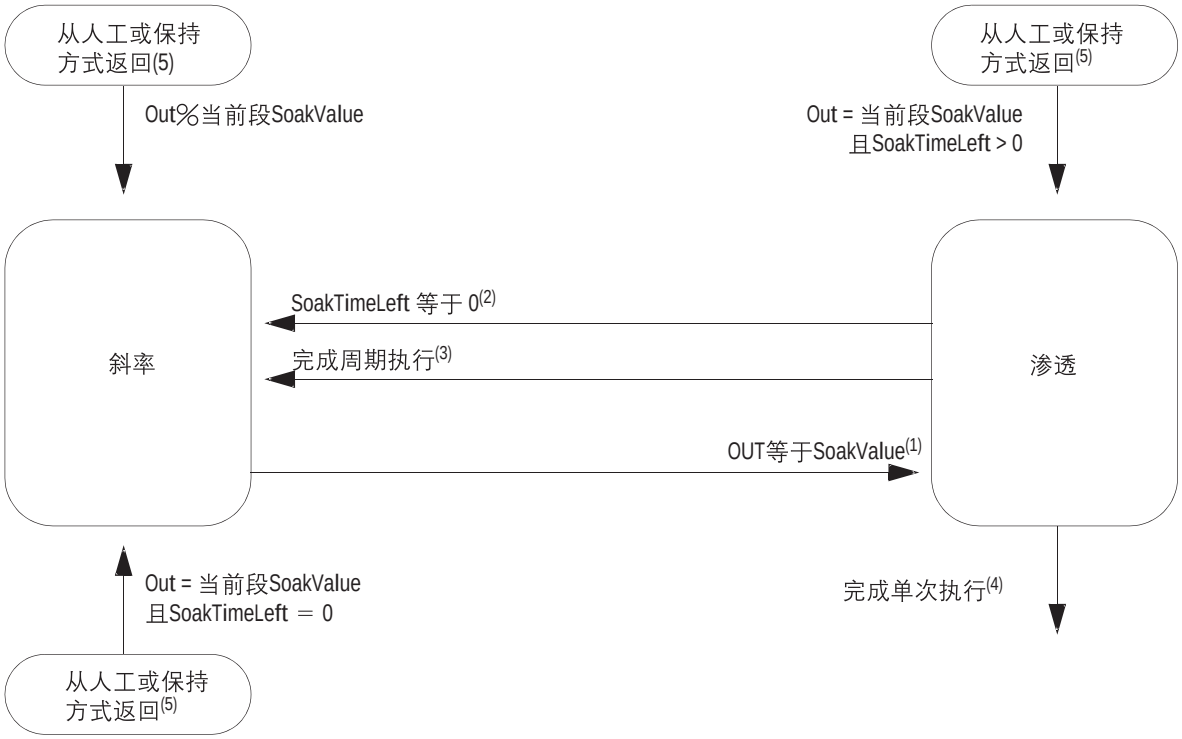
(2) 如果PVFaulted置位或者以下输入无效：NumberOfSegs、CurrentSeg、SoakTimeLeft、CurrentSegOper或SoakTimeOper，则指令进入人工方式

以下表格说明了所有可能的操作员方式

| 方式： | 说明： |
|---|---|
| 操作员自动方式 | 在自动方式，指令连续地执行斜坡/渗透功能。 |
| 操作员人工方式 | 在操作员人工方式操作员界面直接控制指令输出。CurrentSegOper、 SoakTimeOper和OutOper 输入被转化到CurrentSeg、SoakTimeLeft和Out输出。当指令处于自动方式，斜坡/渗透函数从操作员上次输入值重新开始。如果CurrentSegOper 或SoakTime无效则不被切换。<br><br>为了实现无冲击切换到人工方式，指令不处于操作员人工方式时，CurrentSegOper、SoakTimeOper和OutOper输入不断地被当前的CurrentSeg、 SoakTimeLeft和Out输出所更新。 |

## 执行斜坡/渗透功能

以下图表演示了RMPS指令如何执行斜坡/渗透功能。




(1) 当Out = SoakValue时斜率完成。如果在执行期间，Out > SoakValue则被限定为SoakValue。

(2) 当Out保持了当前段的SoakTime时间后，渗透完成。如果所执行的段不是最后一段，则CurrentSeg加1。

(3) 如果渗透已经执行完最后一段且指令被设置为周期执行，则指令设置CurrentSeg = 0.0。

(4) 渗透执行完最后一段如果指令被设置为单次执行则结束。

(5) 当指令返回自动方式，指令判断重新执行斜率或渗透。下个步骤取决于Out、SoakTimeLeft和当前段的SoakValue数值。如果Out = 当前段SoakValue且SoakTimeLeft = 0，则当前段完成，开始执行下一段。

*斜率*

斜率循环地将前一段的SoakValue数值更新为当前段的SoakValue数值。斜率转换的时间是由RampValue参数定义的。

当Out 小于当前段的目标SoakValue值，斜率为正。如果斜率公式计算出的新的Out数值超过当前段的目标SoakValue，则Out被限定为目标SoakValue值。

当Out 大于当前段的目标SoakValue值，斜率为负。如果斜率公式计算出的新的Out数值小于当前段的目标SoakValue，则Out被设置为目标SoakValue值。

每个段都有斜率值。用户可选择以单位时间或速率给定斜率值。但所有段必须以相同的单位给定。以下表格说明了斜率的选项：

| **参数：** | **说明：** |
|---|---|
| 基于时间的斜率 | TimeRate置位为时间表示的斜率(以分钟为单位)。<br>通过加、减指令计算出的当前段的变化速率使Out达到当前段的Soak 值。以下表达式中，DeltaT是自上次指令执行到现在的时间，单位分钟。<br>$Out = Out \pm \frac{(SoakValue_{CurrentSeg} - RampStart)}{RampValue_{CurrentSeg}} \times \Delta t$<br>其中，RampStart是当前段开始的输出值。 |
| 速率表示的斜率 | TimeRate 清零为速率表示的斜率(单位/分钟)<br>通过加、减指令计算出的当前段的变化速率直至Out达到当前段的Soak 值。以下表达式中，DeltaT是自上次指令执行到现在的时间，单位分钟。<br>$Out = Out \pm RampValue_{CurrentSeg} \times \Delta t$ |

*保障斜率*

置位GuarRamp输入来使能保障斜率。使能时，指令监控Out和PV之间的差值。如果差值大于编程的RampDeadband，则输出保持不变直到PV和输Out的差值在死区以内。当Out因保障斜率生效而保持不变时，GuarRampOn输出被置位。

*渗透*

渗透是输出保持不变直到下一个斜坡/渗透段开始的时间值。渗透周期给定时间内保持输出为SoakValue值，直到下一段开始。输出的渗透时间是事先给定到SoakTime参数中的。

每段都有SoakValue和SoakTime。当Out达到当前段的SoakValue值时开始渗透。SoakTimeLeft反映了输出渗透所剩余的时间，单位分钟。在斜率期间，SoakTimeLeft被设定为当前段的SoakTime。一旦斜率结束，SoakTimeLeft根据当前剩余的时间开始减小，单位是分钟。当SoakTime时间到时，SoakTimeLeft = 0。

*保障渗透*

置位GuarSoak来使能保障渗透。使能时，指令监控Out与PV之间的差值，如果差值大于给定的SoakDeadband，则渗透周期被暂停且内部渗透定时器被清零。当PV和OUT的差值在死区以内时，计时重新开始。当保障渗透生效时，GuarRampOn输出被置位。

## 整定(SCL)

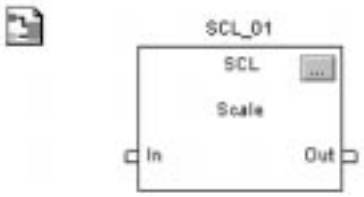
SCL指令将一个非整定的输入值转换成浮点型工程量。

### 操作数：




#### 结构文本

| 操作数： | 类型： | 格式： | 说明： |
|---|---|---|---|
| SCL标签 | SCALE | 结构体 | SCL结构体 |




#### 功能块

| 操作数： | 类型： | 格式： | 说明： |
|---|---|---|---|
| SCL标签 | SCALE | 结构体 | SCL结构体 |

#### SCALE结构体

| 输入参数： | 数据类型： | 说明： |
|---|---|---|
| EnableIn | BOOL | **功能块：**<br>如果该位清零，则指令不执行且输出不更新。如果置位，指令执行。<br>缺省状态为置位<br>**结构文本：**<br>无影响。指令一直执行。 |
| In | REAL | 模拟量信号输入<br>有效值＝实数数值<br>缺省值＝0.0 |
| InRawMax | REAL | 指令输入可达到的最大值。如果InRawMax ? InRawMin，则指令置位状态字中的相应位并停止更新输出。<br>有效值＝InRawMax > InRawMin<br>缺省值＝0.0 |
| InRawMin | REAL | 指令输入可达到的最小值。如果InRawMin ? InRawMax，则指令置位状态字中的相应位并停止更新输出。<br>有效值＝InRawMin < InRawMax<br>缺省值＝0.0 |
| InEUMax | REAL | 与InRawMax对应的输入整定值。<br>有效值＝实数<br>缺省值＝0.0 |
| InEUMin | REAL | 与InRawMin对应的输入整定值。<br>有效值＝实数<br>缺省值＝0.0 |
| Limiting | BOOL | 极限选择。如果置位则输出被限制在InEUMin 和 InEUMax之间。<br>缺省状态为清零 |

| 输出参数： | 数据类型： | 说明： |
|---|---|---|
| EnableOut | BOOL | 输出使能。 |
| Out | REAL | 与模拟量输入对应的整定值输出。输出结果将影响算术状态标志。<br>有效值＝实数<br>缺省值＝InEUMin |
| MaxAlarm | BOOL | 输入报警上限指示。当In> InRawMax时置位。 |
| MinAlarm | BOOL | 输入报警下限指示。当In< InRawMin时置位。 |
| Status | DINT | 功能块的状态字。 |
| InstructFault (Status.0) | BOOL | 指令检测以下的运行错误。该错误不使控制器产生次要或主要错误。检测其它状态位以确定所发生故障类型。 |
| InRawRangeInv (Status.1) | BOOL | InRawMin⩾ InRawMax。 |

**说明：** SCL指令用于不支持高分辨率浮点数整定的模拟量输入模块。

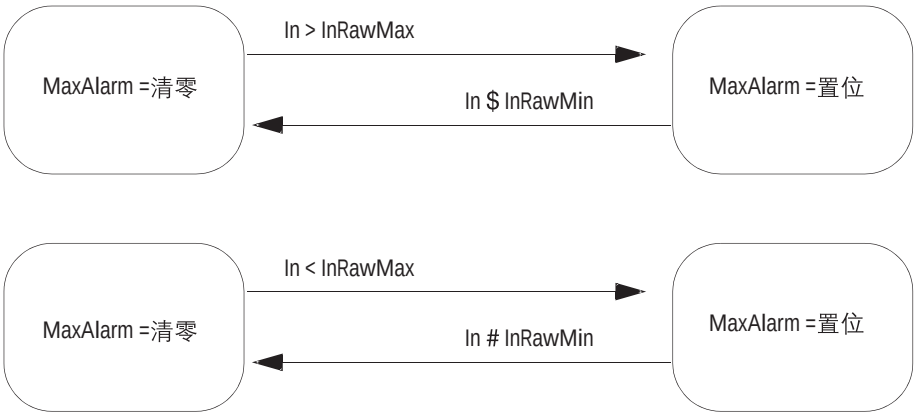
例如：1771-IFE模块是12位的模拟量输入模块，只支持整数值整定。如果用一个1771-IFE模块采集0－100加仑每分钟(gpm)的流量值，通常不需要模块整定为0－100，因为这样会降低模块分辨率。使用SCL指令并配置模块返回一个非整定的(0－4095)数值，SCL指令将其无精度损失地转换为0－100gpm(浮点数)。整定后的数值可当作其他指令的输入。

SCL指令使用以下公式将非整定的输入转化为整定数值：

$$Out = (In\text{-}InRawMin) \times \frac{InEUMax\text{-}InEUMin}{InRax\text{-}InRawMin} + InEUMin$$

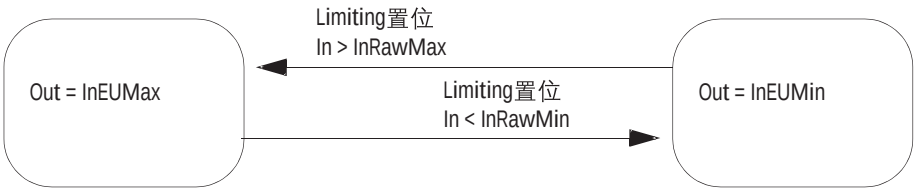
### 报警

一旦指令计算出Out数值，MaxAlarm和MinAlarm就由以下关系决定：




### 极限

当Limiting置位时，则限定Out数值。当In > InRawMax时，指令设定Out = InEUMax。当In < InRawMin，指令设定Out = InEUMin。




**算术状态标志：**该输出影响算术状态标志。

**故障条件：**无

### 执行：

| 条件: | 功能块动作: | 结构文本动作： |
|---|---|---|
| 预扫描 | 无动作。 | 无动作。 |
| 指令首次扫描 | 无动作。 | 无动作。 |
| 指令首次执行 | 无动作。 | 无动作。 |
| 输入使能(EnableIn)清零 | 输出使能(EnableOut)清零。 | 不影响。 |
| 输入使能(EnableIn)置位 | 执行指令。<br>输出使能(EnableOut)置位 | 使能输入(EnableIn)一直置位。指令执行。 |
| 后期扫描 | 无动作。 | 无动作。 |

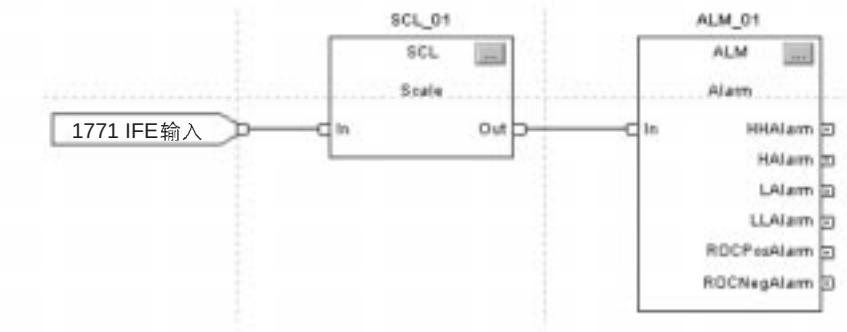
**实例：** SCL指令通常与不支持在线浮点数工程量整定的模拟量输入模块配合使用。在此例中，SCL指令整定由1771－IFE输入的模拟量。指令将结果放在Out中供ALM指令使用。

### 结构文本

```
SCL_01.In := Input0From1771IFE;
SCL(SCL_01);

ALM_01.In := SCL_01.Out;
ALM(ALM_01);
```

### 功能块



## 脉宽调制(SRTP)

SRTP指令将PID回路的0－100%的输出转化成周期性脉冲信号来驱动加热、制冷的离散量输出触点。该指令可用于控制冲压机器的桶的温度。

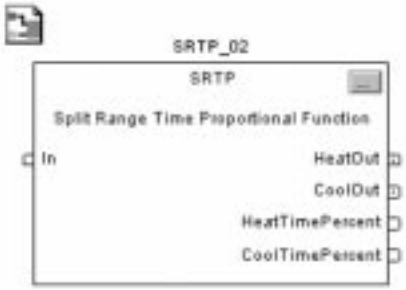
### 操作数：




### 结构文本

| 操作数： | 类型： | 格式： | 说明： |
|---|---|---|---|
| SRTP标签 | SPLIT_RANGE | 结构体 | SRTP结构体 |

### 功能块




| 操作数： | 类型： | 格式： | 说明： |
|---|---|---|---|
| SRTP标签 | SPLIT_RANGE | 结构体 | SRTP结构体 |

### SPLIT_RANGE结构体

| 输入参数： | 数据类型： | 说明： |
|---|---|---|
| EnableIn | BOOL | 功能块：<br>如果该位清零，则指令不执行且输出不更新。如果置位，指令执行。<br>缺省状态为置位<br>结构文本：<br>无影响。指令一直执行。 |
| In | REAL | 模拟量信号输入请求加热或制冷。该输入通常来自PID回路的CVEU。<br>有效值＝浮点数 |
| CycleTime | REAL | 输出脉冲周期，单位为秒。该值为0可关闭加热和制冷输出。如果该值无效，指令设定其值为0并置位状态字中的相应位。<br>有效值＝正浮点数<br>缺省值＝0.0 |
| MaxHeatIn | REAL | 最大加热输入。该数值指定了最大加热输入对应的百分数。对于加热/制冷环节通常为100%。<br>有效值＝浮点数<br>缺省值＝100.0 |
| MinHeatIn | REAL | 最小加热输入。该数值指定了最小加热输入对应的百分数。对于加热/制冷环节通常为50%。<br>有效值＝浮点数<br>缺省值＝50.0 |

| 输入参数： | 数据类型： | 说明： |
|---|---|---|
| MaxCoolIn | REAL | 最大制冷输入。该数值指定了制冷输入的最大百分数。对于加热/制冷环节通常为0%。<br>有效值＝浮点数<br>缺省值＝0.0 |
| MinCoolIn | REAL | 最小制冷输入。该数值指定了制冷输入的最小百分数。对与加热/制冷环节通常为50%。<br>有效值＝浮点数<br>缺省值＝50.0 |
| MaxHeatTime | REAL | 最大加热时间，单位秒。指定以秒为单位的最大加热的脉冲时间。如果指令计算出的HeatTime大于该数值，则HeatTime被限定为MaxHeatTime。如果MaxHeatTime无效，则指令设定其值为CycleTime并置位状态字中的相应位。<br>有效值＝0.0到CycleTime<br>缺省值＝0.0 |
| MinHeatTime | REAL | 最小加热时间，单位秒。指定以秒为单位的最小加热的脉冲时间。如果指令计算出的HeatTime小于该数值，则HeatTime被限定为0。如果MinHeatTime无效，则指令设定其值为0并置位状态字中的相应位。<br>有效值＝0.0到MaxHeatTime<br>缺省值＝0.0 |
| MaxCoolTime | REAL | 最大制冷时间，单位秒。指定以秒为单位的最大制冷的脉冲时间。如果指令计算出的CoolTime大于该数值，则CoolTime被限定为MaxCoolTime。如果MaxCooltTime无效，则指令设定其值为CycleTime并置位状态字中的相应位。<br>有效值＝0.0到CycleTime<br>缺省值＝0.0 |
| MinCoolTime | REAL | 最小制冷时间，单位秒。指定以秒为单位的最小制冷的脉冲时间。如果指令计算出的CoolTime小于该数值，则CoolTime被限定为0。如果MinCoolTime无效，则指令设定其值为0并置位状态字中的相应位。<br>有效值＝0.0到MaxCoolTime<br>缺省值＝0.0 |

| 输出参数： | 数据类型： | 说明： |
|---|---|---|
| EnableOut | BOOL | 输出使能 |
| HeatOut | BOOL | 加热脉冲输出。指令输出脉冲信号给加热触点。 |
| CoolOut | BOOL | 制冷脉冲输出。指令输出脉冲信号给制冷触点。 |
| HeatTimePercent | REAL | 以百分数表示的加热脉冲输出时间。该数值表示HeatingOutput占当前指令周期的百分数。这样如果需要用户就可以使用模拟量输出来加热。该输出影响算术状态标志。 |
| CoolTimePercent | REAL | 以百分数表示的制冷脉冲输出时间。该数值表示CoolingOutput占当前指令周期的百分数。这样如果需要用户就可以使用模拟量输出来制冷。该输出影响算术状态标志。 |
| Status | DINT | 功能块状态字 |
| InstructFault (Status.0) | BOOL | 指令检测以下的运行错误。该错误不使控制器产生次要或主要错误。检测其它状态位以确定所发生故障类型。 |

| 输出参数： | 数据类型： | 说明： |
|---|---|---|
| CycleTimeInv (Status.1) | BOOL | CycleTime数值无效。指令设定为零。 |
| MaxHeatTimeInv (Status.2) | BOOL | MaxHeatTime数值无效。指令使用CycleTime数值。 |
| MinHeatTimeInv (Status.3) | BOOL | MinHeatTime数值无效。指令设定为零。 |
| MaxCoolTimeInv (Status.4) | BOOL | MaxCoolTime数值无效。指令使用CycleTime数值。 |
| MinCoolTimeInv (Status.5) | BOOL | MinCoolTime数值无效。指令设定为零。 |
| HeatSpanInv (Status.6) | BOOL | MaxHeatIn = MinHeatIn。 |
| CoolSpanInv (Status.7) | BOOL | MaxCoolIn = MinCoolIn。 |

**说明：** SRTP脉冲的宽度与PID的输出成比例。该指令参数用于提供给加热和制冷应用。

## 使用内部周期定时器

该指令提供一个始终运行的从0 到编程CycleTime的周期定时器。该内部定时器由Delta T来更新。Delta T是上一次指令执行完到当前所经过的时间。该定时器决定输出是否打开。

用户可随时改变CycleTime。如果CycleTime = 0，则内部定时器清零并且HeatOut和CoolOut也清零。

## 计算加热和制冷时间

每次指令执行都计算加热和制冷时间。

HeatTime是加热输出被打开时间数值，其值小于CycleTime。

$$\text{HeatTime} = \frac{\text{In-MinHeatIn}}{\text{MaxHeatIn-MinHeatIn}} \times \text{CycleTime}$$

- 如果HeatTime < MinHeatTime，则设定HeatTime = 0。
- 如果 HeatTime> MaxHeatTime，则限定HeatTime = MaxHeatTime。

HeatTimePercent是HeatOut时间占CycleTime时间的百分数。

$$\text{HeatTimePercent} = \frac{\text{HeatTime}}{\text{CycleTime}} \times 100$$

CoolTime是制冷输出被打开时间数值，其值小于CycleTime。

$$\text{CoolTime} = \frac{\text{In-MinCoolIn}}{\text{MaxCoolIn-MinCoolIn}} \times \text{CycleTime}$$

- 如果CoolTime < MinCoolTime，则设定CoolTime = 0。
- 如果 CoolTime> MaxCoolTime，则限定CoolTime = MaxCoolTime。

CoolTimePercent是CoolOut占CycleTime时间的百分数。

$$\text{CoolTimePercent} = \frac{\text{CoolTime}}{\text{CycleTime}} \times 100$$

指令使用以下规则控制加热和制冷输出：

- HeatTime≥内部周期时间累加器，则置位HeatOut。
  当内部周期定时器> HeatTime时，清零HeatOut。
- 如果CoolTime≥内部周期时间累加器，则置位CoolOut
  当内部周期定时器> CoolTim时，清零CoolOut。
- 如果CycleTime = 0清零HeatOut和CoolOut

**算术状态标志：** HeatTimePercent和CoolTimePercent 输出影响算术状态标志。

**故障条件：** 无

**执行：**

| 条件: | 功能块动作: | 结构文本动作： |
|---|---|---|
| 预扫描 | HeatOut和CoolOut清零。 | HeatOut和CoolOut清零。 |
| 指令首次扫描 | 内部周期定时器复位。 | 内部周期定时器复位。 |
| 指令首次执行 | 无动作。 | 无动作。 |
| 输入使能(EnableIn)清零 | 输出使能(EnableOut)清零，指令不工作，且输出不更新。 | 不影响 |
| 输入使能(EnableIn)置位 | 执行指令。<br>输出使能(EnableOut)置位 | 使能输入(EnableIn)一直置位。<br>指令执行。 |
| 后期扫描 | 无动作。 | 无动作。 |

**实例：** 在此例中，PIDE指令用于控制温度回路，故在缓慢、优先级较低的任务中执行。因为PIDE指令的输出用于SRTP指令的输入，故其是控制器范围的标签。SRTP指令在高速、优先级较高的任务中执行，以产生更精确的输出脉冲。

**结构文本**

将PIDE指令放在较慢、较低优先级的任务中

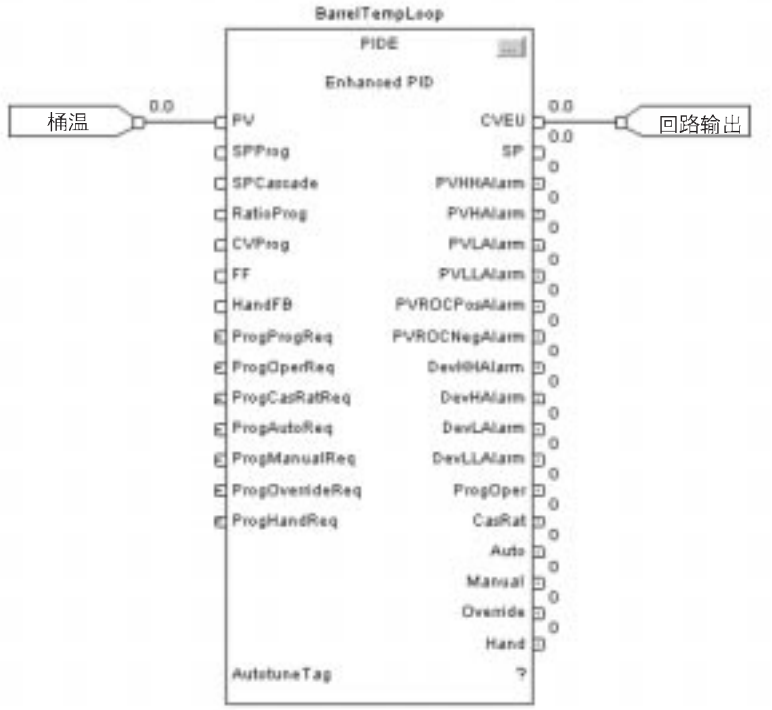
```
BarrelTempLoop.PV := BarrelTemp;
PIDE(BarrelTempLoop);
LoopOutput := BarrelTempLoop.CVEU;
```

将SRTP指令放在较快、较高优先级的任务中

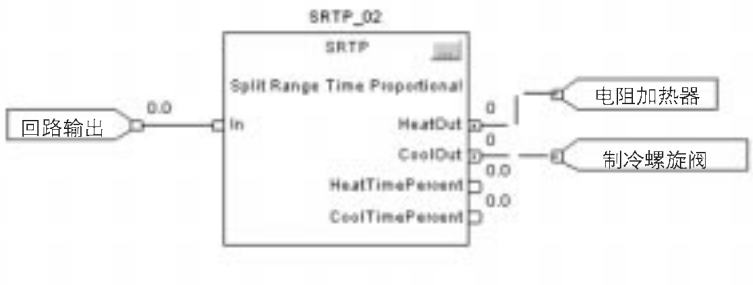
```
SRTP_02.In := LoopOutput;
SRTP(SRTP_02);
ResistiveHeater := SRTP_02.HeatOut;
CoolingSolenoid := SRTP_02.CoolOut;
```

## 功能块

将PIDE指令放在较慢、较低优先级的任务中



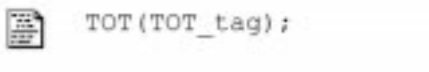

将SRTP指令放在较快、较高优先级的任务中



# 累加器(TOT)

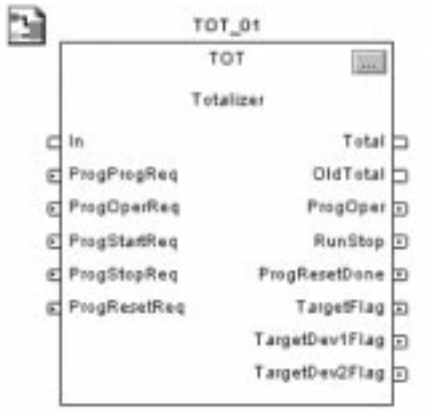
TOT指令累加一定时间范围内的模拟量输入值。

## 操作数：




### 结构文本

| 操作数： | 类型： | 格式： | 说明： |
|---|---|---|---|
| TOT标签 | TOTALIZER | 结构体 | TOT结构体 |




### 功能块

| 操作数： | 类型： | 格式： | 说明： |
|---|---|---|---|
| TOT标签 | TOTALIZER | 结构体 | TOT结构体 |

### TOTALIZER结构体

| 输入参数： | 数据类型： | 说明： |
|---|---|---|
| EnableIn | BOOL | 功能块：<br>如果该位清零，则指令不执行且输出不更新。<br>如果置位，指令执行。<br>缺省状态为置位<br>结构文本：<br>无影响。指令一直执行。 |
| In | REAL | 模拟量信号输入<br>有效值＝浮点数<br>缺省值＝0.0 |
| InFault | BOOL | 输入故障指示。置位时表示输入信号有错误，指令置位状态字中的相应位，控制算法不执行，Total不更新。<br>缺省状态为清零 |

| 输入参数： | 数据类型： | 说明： |
|---|---|---|
| TimeBase | DINT | 时间基输入。累加器计算工程量输入的时间基。<br>**数值：** **说明：**<br>0 秒<br>1 分钟<br>2 小时<br>3 天<br>例如，如果输入单位为gal/min则使用TimeBase =分钟。如果该值无效，则指令置位状态字中的相应位并且不更新Total。如需关于定时方式更多的信息，请参阅附录：功能块属性。<br>有效值＝0到3<br>缺省值＝0 |
| Gain | REAL | 累加值的系数。用户可利用Gain来转换累加值的单位。例如，利用Gain系数将gal/min转换为桶的容量。<br>有效值＝浮点数<br>缺省值＝1.0 |
| ResetValue | REAL | 复位值输入。当OperResetReq 或ProgResetReq从清零转换的置位时给Total复位的值。<br>有效值＝浮点数<br>缺省值＝0.0 |
| Target | REAL | 输入In累加的目标值。<br>有效值＝浮点数<br>缺省值＝0.0 |
| TargetDev1 | REAL | Total预定目标数值与Target之间的较大偏差值。该值反映了其与Target之间的偏离量。<br>有效值＝浮点数<br>缺省值＝0.0 |
| TargetDev2 | REAL | Total预定目标数值与Target之间的小偏差。该值反映了与Target之间的偏离量。<br>有效值＝浮点数<br>缺省值＝0.0 |
| LowInCutoff | REAL | 指令的低输入截断。当输入等于或低于LowInCutoff数值时，累加器停止。<br>有效值＝浮点数<br>缺省值＝0.0 |
| ProgProgReq | BOOL | 程序程序请求。用户程序置位该位请求程序控制。 当ProgOperReq置位时该请求被忽略。保持置位并清零ProgOperReq，可将指令锁定在程序控制方式。<br>缺省状态为清零 |
| ProgOperReq | BOOL | 程序操作员请求。用户程序置位该位请求操作员控制。保持该位可将指令锁定在操作员控制方式。<br>缺省状态为清零 |
| ProgStartReq | BOOL | 程序启动请求输入。置位该位请求开始累加。<br>缺省状态为清零 |
| ProgStopReq | BOOL | 程序停止请求输入。置位该位请求停止累加。<br>缺省状态为清零 |
| ProgResetReq | BOOL | 程序复位请求输入。置位该位请求Total重设为ResetValue。<br>缺省状态为清零 |
| OperProgReq | BOOL | 操作员界面程序请求。通过操作员界面请求程序控制。指令清零该输入。<br>缺省状态为清零 |

| 输入参数： | 数据类型： | 说明： |
|---|---|---|
| OperOperReq | BOOL | 操作员界面操作请求。通过操作员界面请求操作员控制。指令清零该输入。<br>缺省状态为清零 |
| OperStartReq | BOOL | 操作员启动请求输入。通过操作员界面置位该位请求开始累加。<br>缺省状态为清零 |
| OperStopReq | BOOL | 操作员停止请求输入。通过操作员界面置位该位请求停止累加。通过指令清除该输入。<br>缺省状态为清零 |
| Oper ResetReq | BOOL | 操作员复位请求输入。通过操作员界面置位该位请求累加值复位。通过指令清除该输入。<br>缺省状态为清零 |
| ProgValueReset | BOOL | 重设程序控制数值。置位时，每次指令执行后清零所有程序请求输入。<br>缺省状态为清零 |
| TimingMode | DINT | 选择定时执行方式。<br>数值：　　说明：<br>0　　周期方式<br>1　　过采样方式<br>2　　实时采样方式<br>如需更多的信息请参阅附录：功能块属性。<br>有效值＝0到2<br>缺省值＝0 |
| OversampleDT | REAL | 过采样方式的执行时间<br>有效值=0到4194.303 秒<br>缺省值=0 |
| RTSTime | DINT | 实时采样方式下模块的更新周期。<br>有效值=1到32,767ms<br>缺省值=1 |
| RTSTimeStamp | DINT | 实时采样方式下模块的时间戳值。<br>有效值=0到32,767ms<br>缺省值=0 |

| 输出参数： | 数据类型： | 说明： |
|---|---|---|
| EnableOut | BOOL | 输出使能。 |
| Total | REAL | 输入的累加值。该输出影响算术状态标志。 |
| OldTotal | REAL | 复位前的累加值。用户可监控该值，在其复位前读出精确累加值。 |
| ProgOper | BOOL | 程序/操作员控制指示。处于程序控制时置位。处于操作员控制时清零。 |
| RunStop | BOOL | 指示累加器的运行状态。TOT指令运行时，置位。TOT指令停止时清零。 |
| ProgResetDone | BOOL | 该位指示TOT指令已经完成了一次程序复位请求操作。当指令完成<br>ProgResetReq复位时，置位。用户监控该位来判断复位是否成功完成。<br>当ProgResetReq清零时，清零。 |
| TargetFlag | BOOL | Total标志。当Total ? Target时置位。 |

| 输出参数： | 数据类型： | 说明： |
|---|---|---|
| TargetDev1Flag | BOOL | TargetDev1标志。当Total ⩾Target-TargetDev1时置位。 |
| TargetDev2Flag | BOOL | TargetDev2标志。当Total ⩾ Target - TargetDev2时置位。 |
| LowInCutoffFlag | BOOL | 低输入截止标志输出。In ⩾ LowInCutoff时置位。 |
| DeltaT | REAL | 两次更新之间的时间。控制算法利用该时间间隔来计算过程量输出，其单位为秒。 |
| Status | DINT | 功能块状态字 |
| InstructFault(Status.0) | BOOL | 指令检测以下的运行错误。该错误不使控制器产生次要或主要错误。检测其它状态位以确定所发生故障类型。 |
| InFaulted(Status.1) | BOOL | 输入值故障。 |
| TimeBaseInv (Status.2) | BOOL | TimeBase数值无效。 |
| TimingModeInv(Status.27) | BOOL | 定时方式值无效。 |
| RTSMissed(Status.28) | BOOL | 只用于实时采样方式。当ABS| DeltaT - RTSTime | >1(0.001秒)时，该位被置位。 |
| RTSTimeInv(Status.29) | BOOL | RTSTime值无效。 |
| RTSTimeStampInv(Status.30) | BOOL | RTSTimeStamp值无效。 |
| DeltaTInv(Status.31) | BOOL | DeltaT值无效。该情况出现在过采样时OversampleDT无效。 |

**说明：**该指令通常用于累加在一段时间内流量信号的数量。

TOT指令支持：

- 可选择秒、分钟、小时或天作为时间基。
- 可指定一个目标数值，两个预定目标数值。预定目标数是用于切换的慢速进料。用离散量标志来指示达到目标值或预定目标值。
- 允许低流量截止，可用来消除由于小流量计精度误差在流量关闭时引起的累加值负向误差。
- 操作员控制或程序控制启动／停止／复位.
- 用户自定义复位值。
- 梯形数值积分用来提高精度。
- 内部累加以双精度数进行提高了精度

## 监控TOT指令

TOT指令具有操作员面板。如需要了解更多的信息请参阅附录：功能块面板控制。

**算术状态标志：** Total输出影响算术状态标志

**故障条件：** 无

**执行：**

| 条件: | 功能块动作: | 结构文本动作： |
|---|---|---|
| 预扫描 | 无动作。 | 无动作。 |
| 指令首次扫描 | 所有操作员请求输入被清零。<br>如果ProgValueReset置位，则所有程序请求输入被清零。 | |
| 指令首次执行 | 指令初始化内部参数。<br>Total = ResetValue。<br>OldTotal = 0.0。<br>ProgOper清零。 | |
| 输入使能(EnableIn)清零 | 输出使能(EnableOut)清零。 | 不影响 |
| 输入使能(EnableIn)置位 | 执行指令。<br>输出使能(EnableOut)置位 | 使能输入(EnableIn)—直置位。<br>指令执行。 |
| 后期扫描 | 无动作 | 无动作 |

**实例：** 在此例中，TOT指令测量给定数量的水进入罐中，当水加到适当数量时关闭水流。当加水按钮被按下时，TOT重置累加器并重新累加进入罐中的水的流量。一旦目标值达到，TOT指令置位TargetFlag输出，可使电磁阀关闭。此例中，通过置位ProgProgReq 和 ProgStartReq输入可使TOT指令所定在程序运行方式。此例使用了该方式，操作员无需直接控制TOT指令。

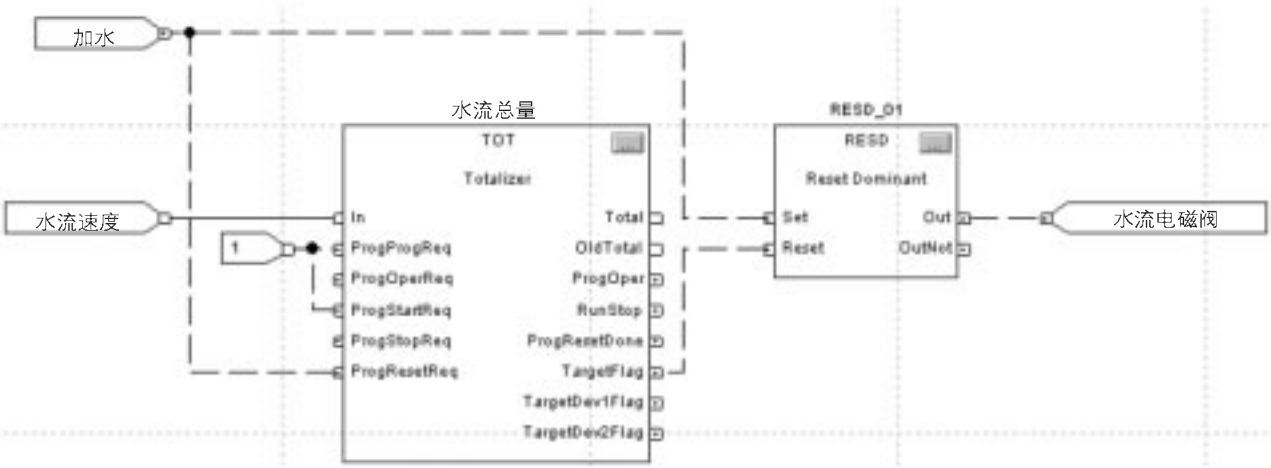
### 结构文本

```
TotalWaterFlow.In := WaterFlowRate;
TotalWaterFlow.ProgProgReq := 1;
TotalWaterFlow.ProgStartReq := 1;
TotalWaterFlow.ProgResetReq := AddWater;
TOT(TotalWaterFlow);

RESD_01.Set := AddWater;
RESD_01.Reset := TotalWaterFlow.TargetFlag;
RESD(RESD_01);

WaterSolenoidValve := RESD_01.Out;
```

### 功能块




### 检测低输入截止

如果(In ⩾LowInCutoff)，指令置位LowInCutoffFlag标志，并使In n-1 = 0.0。否则指令清零LowInCutoffFlag标志。

当LowInCutoffFlag标志置位时，操作方式就确定了，但是累加器停止。当LowInCutoffFlag标志清零时，下次扫描时累加器继续运行。

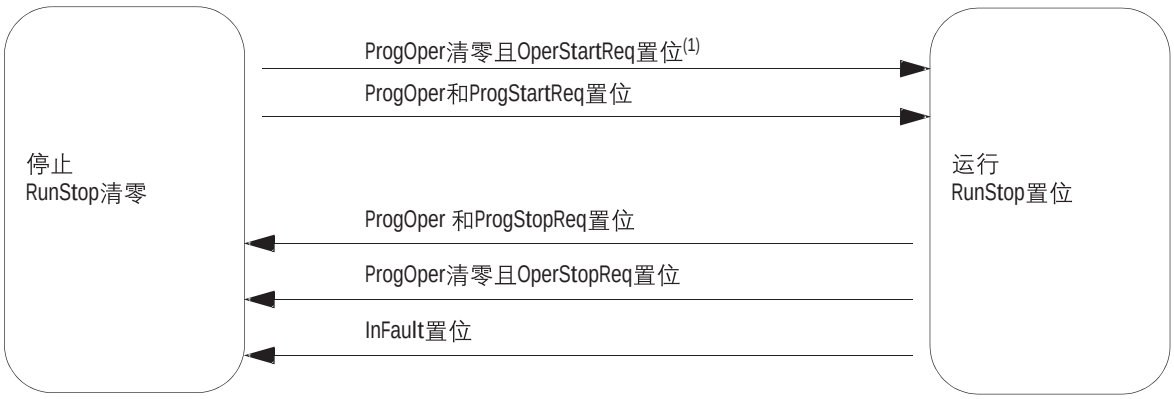
操作方式

以下图表说明了TOT指令如何在程序控制和操作员控制方式之间切换。




(1) 当ProgOperReq置位时，指令仍处于操作员控制方式

以下图表说明了TOT指令如何在运行和停止方式之间切换。




(1) 停止请求优先权高于启动请求。

(2) 停止后的首次扫描时，累加器不计算，但Inn-1更新。在下一次扫描时，累加器重新启动。

每次扫描结束，指令清零所有操作员请求输入。如果ProgValueReset置位，则每次扫描结束，清零所有程序请求输入。

## 复位TOT指令

当ProgOper 置位且ProgResetReq转换到置位状态时，发生下列动作：

- OldTotal = Total
- Total = ResetValue
- ProgResetDone置位

如果ProgResetReq 清零且 ProgResetDone 置位ProgResetDone清零。

当ProgOper 清零且ProgResetReq转换到置位状态时，以下动作发生：

- OldTotal = Total
- Total = ResetValue

## 计算累加值

当RunStop置位且LowInCutoffFlag清零时，采用以下公式来执行累加计算：

$$Total_n = Total_{n-1} + Gain \times \frac{DeltaT}{2 \times TimeBase} \times (In_n + In_{n-1})$$

其中时间基为：

| 数值： | 条件： |
|---|---|
| 1 | TimeBase = 0 (秒) |
| 60 | TimeBase = 1 (分钟) |
| 3600 | TimeBase = 2 (小时) |
| 86400 | TimeBase = 3 (天) |

## 判断是否达到目标数值

一旦计算出累加值，以下条件就用来判断是否达到目标值或预定目标值：

- 当Total≥Target时，TargetFlag 置位。
- 当Total≥(Target - TargetDev1)时，TargetDev1Flag置位。
- 当Total≥(Target - TargetDev2)时，TargetDev2Flag置位。

注释：

# 驱动指令 (INTG,PI,PMUL,SCRV,SOC,UPDN)

## 概述

以下驱动指令可用：

| 如果用户需要： | 使用指令： | 可用语言： | 参考页码： |
|---|---|---|---|
| 执行一个积分操作 | 积分器 (INTG) | 结构文本<br>功能块 | 2月2日 |
| 执行一个比例积分算法 | 比例+积分 (PI) | 结构文本<br>功能块 | 2月8日 |
| 提供一个从位置输入模块(比如解码器或编码器的反馈模块)到数字系统的接口，方法是检测两次扫描间输入信号值的改变量。 | 多路脉冲切换器 (PMUL) | 结构文本<br>功能块 | 2月20日 |
| 实现具有附加加加速度极限的斜率函数 | S-曲线(SCRV) | 结构文本<br>功能块 | 2月28日 |
| 使用一个增益项，一个一阶滞后项，一个二阶超前项进行控制。 | 二阶控制器(SOC) | 结构文本<br>功能块 | 2-38 |
| 加上输入值到累加值和从累加值减去输入值 | 增/减累加器UPDN) | 结构文本<br>功能块 | 2-46 |

## 积分器(INTG)

INTG指令实现积分操作。该指令为扫描时间为常数的任务而设计。

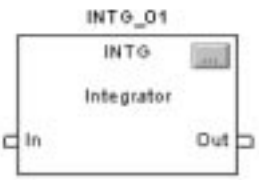
### 操作数：

#### 结构文本

```
INTG(INTG_tag);
```

| 操作数： | 类型： | 格式： | 说明： |
|---|---|---|---|
| INTG标签 | INTEGRATOR | 结构体 | INTG结构体 |

#### 功能块




| 操作数： | 类型： | 格式： | 说明： |
|---|---|---|---|
| INTG标签 | INTEGRATOR | 结构体 | INTG结构体 |

### INTG结构体

| 输入参数： | 数据类型： | 说明： |
|---|---|---|
| EnableIn | BOOL | 功能块：<br>如果该位清零，则指令不执行且输出不更新。如果置位，指令执行。<br>缺省状态为置位<br>结构文本：<br>无影响。指令一直执行。 |
| In | REAL | 指令的模拟量输入信号。<br>有效值＝浮点数<br>缺省值＝0.0 |
| Initialize | BOOL | 初始化控制算法请求。只要Initialize被置位，则Output=InitialValue。<br>有效值=浮点数<br>缺省值=0.0 |
| InitialValue | REAL | 指令的初始化值。只要Initialize被置位，则Output=InitialValue。<br>有效值=浮点数<br>缺省值=0.0 |
| IGain | REAL | 积分增益系数。若IGain<0，则指令设置IGain=0.0并置位状态字中的相应位，并保持输出不变。<br>有效值=0.0到最大正浮点数<br>缺省值=0.0 |
| HighLimit | REAL | 输出值Out的上限值。若HighLimit≤LowLimit，指令置位HighAlarm和LowAlarm并置位状态字中的相应位，且设置输出值Out=LowLimit。<br>有效值=浮点数<br>缺省值=最大正浮点数 |
| LowLimit | REAL | 输出值Out的下限值。若HighLimit≤LowLimit，指令置位HighAlarm和LowAlarm并置位状态字中的相应位，且设置输出值Out=LowLimit。<br>有效值=浮点数<br>缺省值=最小负浮点数 |

| 输入参数： | 数据类型： | 说明： |
|---|---|---|
| HoldHigh | BOOL | 保持输出上限请求。若该位被置位，输出值Out不允许增加。<br>缺省状态为清零。 |
| HoldLow | BOOL | 保持输出下限请求。若该位被置位，输出值Out不允许减少。<br>缺省状态为清零。 |
| TimingMode | DINT | 选择定时执行方式。<br>**有效值：** **说明：**<br>0 周期执行方式<br>1 过采样方式<br>2 实时采样方式<br>需要了解更多与定时方式有关的信息，请参阅附录：功能块属性。<br>有效值=0到2<br>缺省值=0 |
| OversampleDT | REAL | 过采样方式下的执行时间。<br>有效值=0到4194.303秒<br>缺省值=0 |
| RTSTime | DINT | 实时采样方式下模块的更新周期。<br>有效值=1到32,767ms<br>缺省值=1 |
| RTSTimeStamp | DINT | 实时采样方式下模块的时间戳值。<br>有效值=0到32,767ms<br>缺省值=0 |

| 输出参数： | 数据类型： | 说明： |
|---|---|---|
| EnableOut | BOOL | 输出使能。 |
| Out | REAL | 该指令算法计算出的输出结果。输出结果将影响算术状态标志设置。 |
| HighAlarm | BOOL | 上限报警指示。当Out≥HighLimit时，HighAlarm被置位且输出值被限定为HighLimit值。 |
| LowAlarm | BOOL | 下限报警指示。当Out≤LowLimit时，LowAlarm被置位且输出值被限定为LowLimit值。 |
| DeltaT | REAL | 两次更新之间经过的时间，单位：秒。作为控制算法计算过程量输出值的时间间隔参数。 |
| Status | DINT | 功能块的状态字。 |
| InstructFault (Status.0) | BOOL | 指令检测以下的运行错误。该错误不使控制器产生次要或主要错误。检测其它状态位以确定发生错误的类型。 |
| IGainInv (Status.1) | BOOL | IGain>最大值或IGain<最小值。 |
| HighLowLimsInv (Status.2) | BOOL | HighLimit≤LowLimit。 |
| TimingModeInv (Status.27) | BOOL | 定时方式值无效。<br>需要了解更多与定时方式有关的信息，请参阅附录：功能块属性。 |
| RTSMissed (Status.28) | BOOL | 只用于实时采样方式。当ABS \| DeltaT ®C RTSTime \| > 1 (0.001秒)时，该位被置位。 |

| 输出参数： | 数据类型： | 说明： |
|---|---|---|
| RTSTimeInv(Status.29) | BOOL | RTSTime值无效。 |
| RTSTimeStampInv(Status.30) | BOOL | RTSTimeStamp值无效。 |
| DeltaTInv(Status.31) | BOOL | DeltaT值无效。 |

**说明：** INTG指令是为扫描时间为常数的任务而设计的。

INTG指令在Initialize被清零且DeltaT>0时执行以下控制算法：

$$Out = IGain \times \frac{In+In_{n-1}}{2} \times DeltaT+Out_{n-1}$$

无论计算的输出结果值无效、NAN 或 ±INF，该指令设置Out为无效值且置位算术溢出状态标志。内部参数不再更新。在以后的每次扫描中，指令使用的最后一次输出值有效时扫描的内部参数计算输出值。

### 极限

INTG指令根据HoldHigh和HoldLow输入参数的状态来实现积分超调极限，进而限制输出值Out改变。如果HoldHigh被置位且Out>Outn-1，则Out=Outn-1；如果HoldLow被置位且Out<Outn-1，则Out=Outn-1。

INTG指令同时也使用HighLimit和LowLimig对输出值的范围限制。如果Out≥HighLimit，则Out=HighLimit且HighAlarm被置位。如果Out≤LowLimit，则Out=LowLimit且LowAlarm被置位。

**算术状态标志：** 输出值(Out)将影响算术状态椁设置

**故障条件：** 无

**执行：**

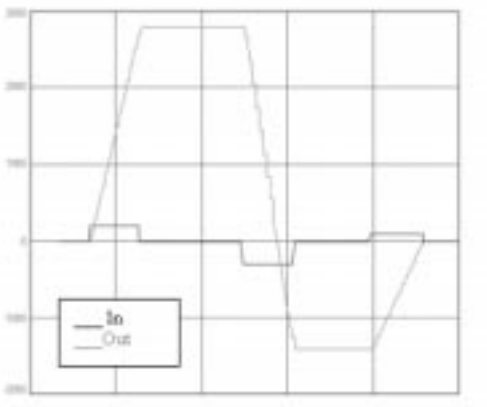
| 条件： | 功能块动作： | 结构文本动作： |
|---|---|---|
| 预扫描 | 无动作。 | 无动作。 |
| 指令首次扫描 | 内部参数和输出值Out被置零。<br>控制算法不执行。 | 内部参数和输出值Out被置零。<br>控制算法不执行。 |
| 指令首次执行 | 内部参数和输出值Out被置零。<br>控制算法不执行。 | 内部参数和输出值Out被置零。<br>控制算法不执行。 |

| 条件： | 功能块动作： | 结构文本动作： |
|---|---|---|
| 输入使能(EnableIn)清零 | 输出使能(EnableOut)清零，指令不执行且输出不更新。 | 不影响。 |
| 输入使能(EnableIn)置位 | 执行指令。<br>输出使能(EnableOut)置位 | 输入使能一直置位。<br>指令执行。 |
| 后期扫描 | 无动作 | 无动作 |

**实 例：**在许多应用中，积分增益环节被包含在闭环控制校正装置的设计中，目的是为了消除或降低被校正系统中的误差。单纯的比例校正装置不能使系统误差趋于0，但一个使用比例增益和积分增益的校正装置，就能使一段时间内的系统误差信号趋于0。INTG指令使用下面的方程式计算它的输出值。

$$Out = IGain \times \frac{In+Inn-1}{2} \times DeltaT + Out_{n-1}$$

在该图表中，功能块的输入从0增加到+200单位。在此期间，功能块的输出值累积到2800单位。当输入In从+200单位变化到0单位时，输出值Out保持在2800单位。当输入In从0变化到-300单位时，输出值Out缓慢积分递减到-1400单位直到输入In从新变化为0。最后，输入值In从0改变到+100，输出值Out积分回到0，此时输入值In回到0同时输出值Out也归为0。




积分器的特性——功能块输入值为0附近时，无论呈现或保持什么样的值，输出值总在一个特定方向上保持连续的推进——是带积分增益的校正装置使系统在一段时间内系统误差趋于0的原因。

下面的例子说明INTG指令在实际应用系统中的使用。在许多实例中，HighLimit和LowLimit输入参数限制控制量的百分比，这样积分增益元素就可作为整个校正装置的输出。HoldHigh和HoldLow输出参数的使用从另一方面防止输出值在正向或负向上超限。在本例中，如果校正装置的输出已经达到100%饱和，

HoldHigh和HoldLow输入参数防止INTG指令在已超过控制量限值的方向产生"超调"现象。

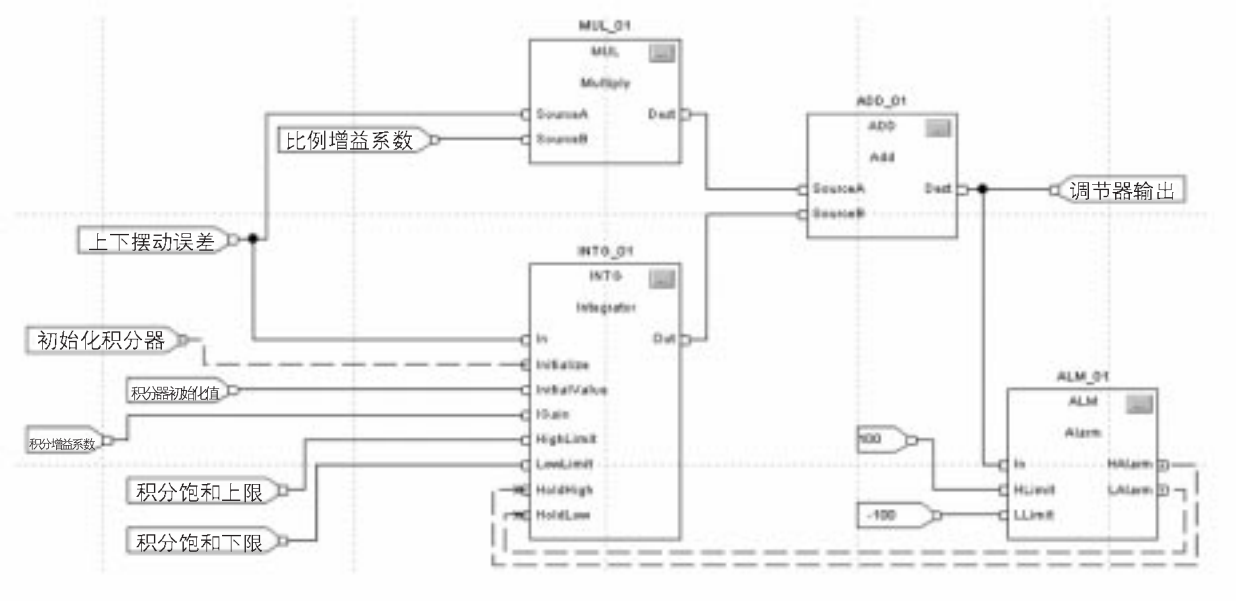
### 结构文本

```
INTG_01.IN := Dancer_Loop_Error;
INTG_01.Initialize := Initialize_Integrator;
INTG_01.InitialValue := Int_Init_Val;
INTG_01.IGain := I_Gain;
INTG_01.HighLimit := Int_saturate_high;
INTG_01.LowLimit := Int_saturate_low;
INTG_01.HoldHigh := ALM_01.HAlarm;
INTG_01.HoldLow := ALM_01.LAlarm;
INTG(INTG_01);

regulator_out := (Dancer_Loop_Error*Proportional_Gain)+ INTG_01.Out;

ALM_01.In := regulator_out;
ALM_01.HLimit := 100;
ALM_01.LLimit := -100;
ALM(ALM_01);
```

## 功能块


MUL_01
MUL
Multiply
SourceA
SourceB
Dest
比例增益系数
ADD_01
ADD
Add
SourceA
SourceB
Dest
调节器输出
上下摆动误差
INTG_01
INTG
Integrator
In
Out
InitialIze
InitialValue
IGain
HighLimit
LowLimit
HoldHigh
HoldLow
初始化积分器
积分器初始化值
积分增益系数
积分饱和上限
积分饱和下限
ALM_01
ALM
Alarm
In
HiLimit
LLimit
HiAlarm
LoAlarm
100
-100

## 比例+积分算法(PI)

PI指令提供两种方式的操作。第一种方式是比例和积分增益系数在整个输入信号(误差信号)范围内保持恒定的通常的PI算法。第二种方式使用非线性算法，比例和积分增益系数在输入信号范围内随着输入信号的变化而变化。输入信号是过程反馈信号和设定点间的偏差。

### 操作数：

#### 结构文本

| 操作数 | 类型 | 格式 | 说明 |
|---|---|---|---|
| PI标签 | PROP_INT | 结构体 | PI结构体 |

#### 功能块

| 操作数 | 类型 | 格式 | 说明 |
|---|---|---|---|
| PI标签 | PROP_INT | 结构体 | PI结构体 |

### PROP_INT结构体

| 输入参数： | 数据类型： | 说明： |
|---|---|---|
| EnableIn | BOOL | 功能块：<br>如果该位清零，则指令不执行且输出不更新。如果置位，指令执行。<br>缺省状态为置位<br>**结构文本：**<br>无影响。指令一直执行。 |
| In | REAL | 过程偏差量输入信号。该偏差是反馈信号同设定点间的差值。<br>有效值＝浮点数<br>缺省值＝0.0 |
| Initialize | BOOL | 指令初始化命令。若被置位，输出值Out和内部积分器被设置为InitialValue的值。<br>默认置位清零 |
| InitialValue | REAL | 输入初始化值。当Initialize被置位时，输出值Out和积分器被设置为InitialValue的值。InitialValue使用HighLimit和LowLimit限制取值范围。<br>有效值=浮点数<br>缺省值=0.0 |
| Kp | REAL | 比例系数。该参数影响比例和积分控制算法计算的值。如果该值无效，指令设置Kp为极限值且置位状态字中的相应位。<br>有效值=大于0.0的浮点数<br>缺省值=最小正浮点数 |

| 输入参数： | 数据类型： | 说明： |
|---|---|---|
| Wld | REAL | 超前频率，单位：弧度/秒。该参数影响积分控制算法计算的值。如果该值无效，指令设置Wld为极限值且置位状态字中的相应位。<br>有效值=参阅下面关于有效值范围的说明<br>缺省值=0.0 |
| HighLimit | REAL | 上限值，即输出的最大值。若HighLimit≤LowLimit，则指令置位HighAlarm和LowAlarm，置位状态字中的相应位，且设置Out=LowLimit。<br>有效值=LowLimit<HighLimit≤最大正浮点数<br>缺省值=最大正浮点数 |
| LowLimit | REAL | 下限值，即输出Out的最小值。若HighLimit≤LowLimit则指令置位HighAlarm和LowAlarm，置位状态字中的相应位，且设置Out=LowLimit。<br>有效值=最小负浮点数≤LowLimit<HighLimit<br>缺省值=最小负浮点数 |
| HoldHigh | BOOL | 上限保持命令。如果该位置位，内部积分器的值不允许增加。<br>缺省状态为清零 |
| HoldLow | BOOL | 下限保持命令。若置位，内部积分器的值不允许减小。<br>缺省状态为清零 |
| ShapeKpPlus | BOOL | 正向Kp修正增益乘数。当输入值In≥0时使用。如果该值无效，指令设置ShapeKpPlus为极限值且置位状态字中的相应位。当NonLinearMode被清零时忽略该参数。<br>有效值=0.1到10.0<br>缺省值=1.0 |
| ShapeKpMinus | REAL | 反向Kp修正增益乘数。当输入值In<0时使用。如果该值无效，指令设置ShapeKpMinus为极限值且置位状态字中的相应位。当NonLinearMode被清零时忽略该参数。<br>有效值=0.1到10.0<br>缺省值=1.0 |
| KpInRange | REAL | 比例增益系数修正范围。定义一个输入(偏差)的范围，在该范围内比例增益系数以|In|/KpInRange比率的函数来增加或减少。当|In|>KpInRange时，指令使用输入的Kp修正增益系数乘以(In-KpInRange)来计算比例误差的变化。如果该值无效，指令设置KpInRange为极限值且置位状态字中的相应位。当NonLinearMode被清零时忽略该参数。<br>有效值=浮点数>0.0<br>缺省值=最大正浮点数 |
| ShapeWldPlus | REAL | 正向Wld修正增益乘数。当输入值In≥0时使用该参数。如果该值无效，指令设置ShapeWldPlus为极限值且置位状态字中的相应位。若NonLinearMode被清零则忽略该参数。<br>有效值=0.0到10.0<br>缺省值=1.0 |
| ShapeWldMinus | REAL | 反向Wld修正增益乘数。当输入值In<0时使用。如果该值无效，指令设置ShapeWldMinus为极限值且置位状态字中的相应位。若NonLinearMode被清零则忽略参数。<br>有效值=0.0到10.0<br>缺省值=1.0 |

| 输入参数： | 数据类型： | 说明： |
|---|---|---|
| WldInRange | REAL | 积分增益系数修正范围。定义一个输入(误差)的范围，在该范围内积分增益系数以|In|/WldInRange比率的函数来增加或减少。若|In|>WldInRange，指令在计算积分误差时设定输入值In为WldInRange。若该值无效，指令设置WldInRange为极限值且置位状态字中的相应位。若NonLinearMode被清零则忽略该参数。<br>有效值=浮点数>0.0<br>缺省值=最大正浮点数 |
| NonLinearMode | BOOL | 非线性增益方式使能。若该位置位，指令使用ParabolicLinear参数所选择的非线性增益方式来计算实际的比例增益系数和积分增益系数。若该位清零，指令禁用非线性增益方式并以Kp和Wld的值作为比例和积分增益系数。<br>缺省状态为清零 |
| ParabolicLinear | BOOL | 选择非线性增益方式。包括直线和抛物线两种。若该位置位，指令使用抛物线增益方式y=a*x^2+b来计算实际的比例和积分增益系数。若该位清零，指令使用直线增益方式y=a*x+b<br>缺省状态为清零。 |
| TimingMode | DINT | 选择定时执行方式：<br>**有效值： 说明：**<br>0 周期执行方式<br>1 过采样方式<br>2 实时采样方式<br>需要了解更多与定时方式有关的信息，请参阅附录：功能块属性。<br>有效值=0到2<br>缺省值=0 |
| OversampleDT | REAL | 自定时隙采样方式下的执行时间。<br>有效值=0到4194.303秒<br>缺省值=0 |
| RTSTime | DINT | 实时采样方式下模块的更新周期。<br>有效值=1到32,767ms<br>缺省值=1 |
| RTSTimeStamp | DINT | 实时采样方式下模块的时间标签。<br>有效值=0到32,767ms<br>缺省值=0 |

| 输出参数： | 数据类型： | 说明： |
|---|---|---|
| EnableOut | BOOL | 输出使能。 |
| Out | REAL | PI算法计算出的输出值。输出结果将影响算术状态标志设置。 |
| HighAlarm | BOOL | 上限报警指示。当输出值Out≥HighLimit时置位，输出和积分器被限定为HighLimit。 |
| LowAlarm | BOOL | 下限报警指示。当输出值Out≤LowLimit时置位，输出和积分器被限定为LowLimit。 |
| DeltaT | REAL | 两次更新之间经过的时间，单位为秒。作为控制算法计算过程量输出值的时间间隔参数。 |
| Status | DINT | 功能块的状态字。 |

| 输出参数： | 数据类型： | 说明： |
|---|---|---|
| InstructFault(Status.0) | BOOL | 指令检测以下的执行错误。该错误不使控制器产生次要或主要错误。检测其它状态位以确定发生错误的类型。 |
| KpInv(Status.1) | BOOL | Kp<最小值或Kp>最大值。 |
| WldInv(Status.2) | BOOL | Wld<最小值或Wld>最大值。 |
| HighLowLimsInv(Status.3) | BOOL | HighLimit≤LowLimit。 |
| ShapeKpPlusInv(Status.4) | BOOL | ShapeKpPlus<最小值或ShapeKpPlus>最大值。 |
| ShapeKpMinusInv(Status.5) | BOOL | ShapeKpMinus<最小值或ShapeKpMinus>最大值。 |
| KpInRangeInv(Status.6) | BOOL | KpInRange<最小值或ShapeKpInRange>最大值。 |
| ShapeWldPlusInv(Status.7) | BOOL | ShapeWldPlus<最小值或ShapeWldPlus>最大值。 |
| ShapeWldMinusInv(Status.8) | BOOL | ShapeWldMinus<最小值或ShapeWldMinus>最大值。 |
| WldInRangeInv(Status.9) | BOOL | WldInRange<最小值或WldInRange>最大值。 |
| TimingModeInv(Status.27) | BOOL | 定时方式无效。<br>需要了解更多与定时方式有关的信息，请参阅附录：功能块属性。 |
| RTSMissed(Status.28) | BOOL | 只用于实时采样方式。当ABS| DeltaT ®C RTSTime | >1(0.001秒)时，该位被置位。 |
| RTSTimeInv(Status.29) | BOOL | RTSTime值无效。 |
| RTSTimeStampInv(Status.30) | BOOL | RTSTimeStamp值无效。 |
| DeltaTInv(Status.31) | BOOL | DeltaT值无效。 |

**说明：** PI指令使用PI算法的引脚形式。这就意味着增益系数项被直接应用到输入信号而非应用到输入信号的改变量。PI指令是为扫描周期是常数的任务而设计的。

在非线性算法方式中，比例和积分增益系数随着输入信号的改变量大小而变化。PI指令支持两种非线性增益方式：直线和抛物线。在直线算法中，增益系数随着输入量的改变量呈直线变化。在抛物线的算法中，增益系数随着输入量的改变量呈抛物线变化。

PI指令使用下面方程式计算输出值：

$$KP \times \frac{s+WLD}{s}$$

无论计算的输出结果值无效、NAN 或 ±INF，该指令设置Out = 无效的值并且置位算术溢出状态标志。内部参数不再更新。在以后的每次扫描中，指令使用的最后一次输出值有效时扫描的内部参数计算输出值。

## 线性方式下的运算

在线性方式下，非线性增益方式不能使用。指令使用Kp和Wld的值作为比例和积分增益系数。指令使用下面方程式计算输出的值：

| 参数项： | 方程式： |
|---|---|
| I项(ITerm) | $Kp \times Wld \times \frac{WldInput+WldInput_{n-1}}{2} \times DeltaT+ITerm_{n-1}$<br>此时DeltaT的单位是秒 |
| P项(PTerm) | $Kp \times In$ |
| 输出值(Out) | ITerm+PTerm |

其中Wld的极限如下：
LoLimit > 0.0
HighLimit=0.7π/DeltaT
WldInput=In

## 非线性方式下的运算

在非线性方式下，指令使用ParabolicLinear所选择的非线性增益方式计算实际的比例和积分增益系数。

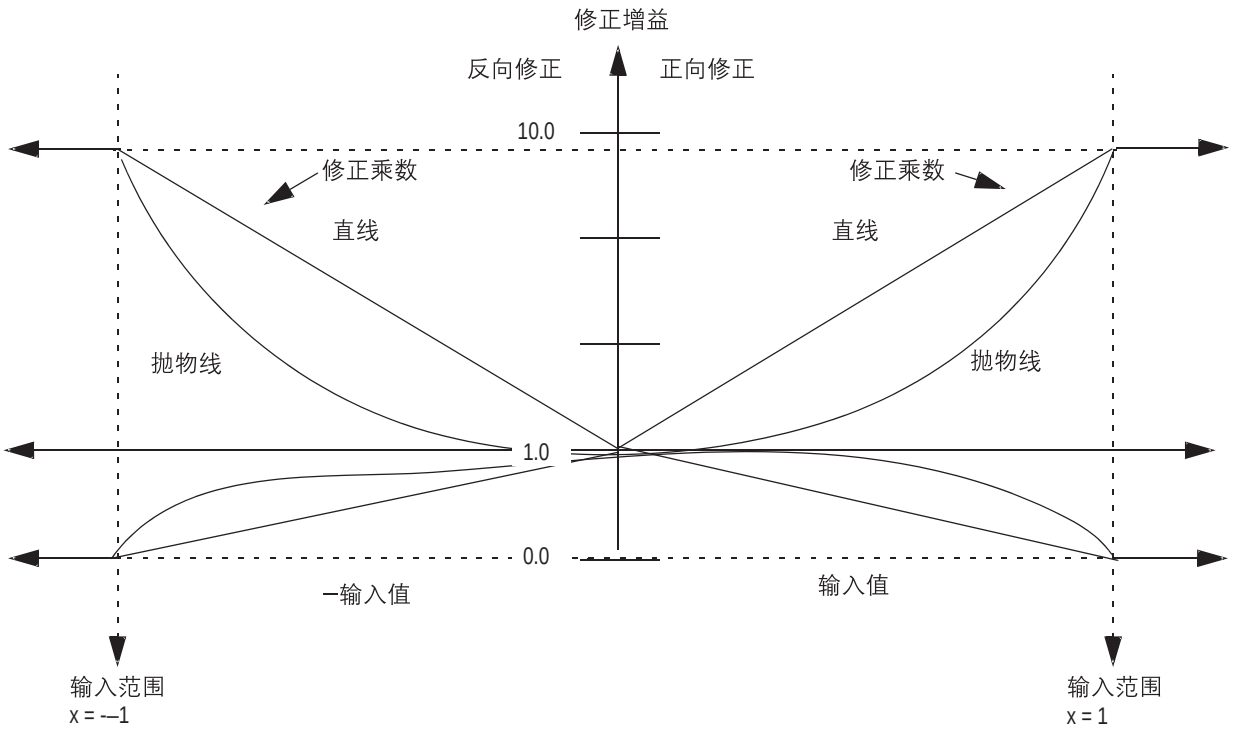
当In=0时，Kp和Wld指定的增益系数乘以1.0。独立的比例和积分算法随着误差量的变化量大小增加或减少比例和积分增益系数。这些算法使用输入范围和修正增益系数参数计算实际的比例和积分增益系数。输入范围决定了输入值In(比如：误差量)的一个范围，在此范围内增益系数被修正。输入范围由KpInRange和WldInRange两个参数设定。修正增益由修正增益参数控制，定义了信号区的增益系数参数。修正增益通过ShapeKpPlus，ShapeKpMinus，ShapeWldPlus，ShapeWldMinus设定。

ParabolicLinear输入参数选择非线性增益的方式。若ParabolicLinear被清零，则选择直线方式；若ParabolicLinear被置位，则选择抛物线方式。

组态一个特定的修正增益曲线，需要输入一个0.0到10.0的修正增益乘数用于修正积分过程，一个0.1到10.0的修正增益乘数用于修正比例过程和一个实施修正的输入范围。Kp和Wld被乘以计算得到的修正乘数求出实际的比例和积分增益系数。输入一个1.0的修正增益乘数可禁止信号区内计算比例和积分增益系数的非线性算法。

当输入值(误差量)的大小超出输入范围时，修正乘数的值为|In|＝输入范围值时计算得到的值。

下图是抛物线和直线方式下增益系数方程式的最大和最小增益系数曲线。



指令使用下面方程式计算输出Out的值：

| 参数项： | 方程式： |
|---|---|
| Kp修正增益系数 | 若In ≥0，则：<br>KpShapeGain=ShapeKpPlus<br>KpRange=KpInRange<br>否则：<br>KpShapeGain=ShapeKpMinus<br>KpRange=-KpInRange |
| Kp输入比率 | 若\|In\|≤KpInRange，则：<br>$KpInputRatio=\|In\| \times \frac{1}{KpInRange)}$<br>否则：<br>KpInputRatio=1 |
| Kp比率 | 若不是抛物线方式，则：<br>KpRatio=KpInputRatio ×0.5<br>若是抛物线方式，则：<br>KpRatio=KpInputRatio^2 ×0.333 |
| Kps修正增益系数 | Kps=Kp ×(((KpShapeGain-1) ×KpRatio)+1) |
| 比例算法输出 | 若\|In\|≤KpInRange，则：<br>PTerm=Kps ×In<br>否则，限制增益：<br>PTerm=Kps ×KpRange+(In-KpRange) ×KpShape |
| Wld修正增益系数 | 若In ≥0，则：<br>WldShapeGain=ShapeWldPlus<br>否则：<br>WldShapeGain=ShapeWldMinus |
| Wld输入值 | 若In>WldRange，则：<br>WldInput=WldInRange<br>否则若In<-WldInRange，则：<br>WldInput=-WldInRange<br>否则：<br>WldInput=In |
| Wld输入比率 | 若\|In\|≤WldInRange，则：<br>$WldInputRange=\|In\| \times \frac{1}{WldInRange}$<br>否则：<br>WldInputRange=1 |

| 参数项： | 方程式： |
|---|---|
| Wld 比率 | 若不是抛物线方式，则：<br>WldRatio=WldInputRatio<br>若是抛物线方式，则：<br>WldRatio=WldInputRatio2 |
| Wld修正增益系数 | Wlds=Wld ×(((WldShapeGain-1) ×WldRatio)+1) |
| Wlds极限 | 下限>0<br>上限= $\frac{0.7\pi}{DeltaT}$ |
| 积分输出 | $ITerm=Kps \times Wlds \times \frac{WldInput+WldInput_{n-1}}{2} \times DeltaT+ITerm_{n-1}$ |
| 输出值 | Out=PTerm+ITerm |

## 极限

指令根据hold输入参数的状态防止I项(ITerm)超调。

| 条件： | 执行动作： |
|---|---|
| 若HoldHigh被置位且ITerm>ITermn-1 | ITerm=ITermn-1 |
| 若HoldLow被置位且ITerm<ITermn-1 | ITerm=ITermn-1 |

指令也根据HighLimit和LowLimit的值防止积分超调。

| 条件： | 执行动作： |
|---|---|
| Integrator>HighLimit | Integrator=HighLimit |
| Integrator<LowLimit | Integrator=LowLimit |

指令根据HighLimit和LowLimit的值限制Out的值。

| 条件： | 执行动作： |
|---|---|
| HighLimit ≤LowLimit | Out=LowLimit<br>ITerm=LowLimit<br>HighLowLimsInv 置位<br>HighAlarm 置位<br>LowAlarm 置位<br>WldInput=0 |
| Out ≥HighLimit | Out=HighLimit<br>ITerm=$ITerm_{n-1}$<br>HighAlarm 置位 |
| ITerm>HighLimit | ITerm=HighLimit |
| Out ≤LowLimit | Out=LowLimit<br>ITerm=$ITerm_{n-1}$<br>LowAlarm 置位 |
| ITerm<LowLimit | ITerm=LowLimit |

**算术状态标志：**输出值将影响算术状态标志设置。

**故障条件：**无

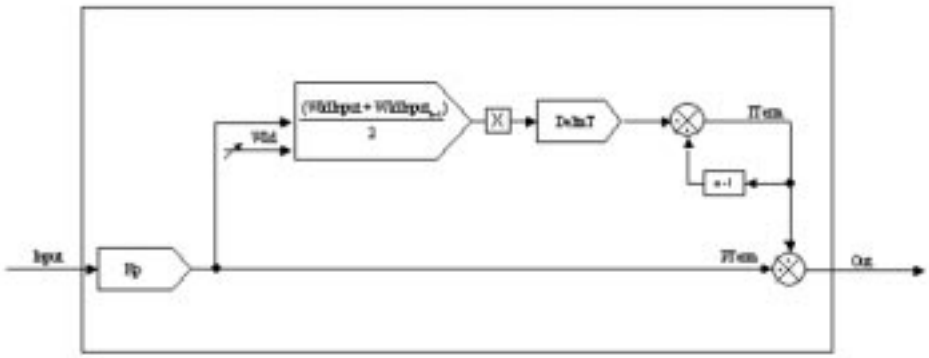
**执行：**

| 条件： | 功能块动作： | 结构文本动作： |
|---|---|---|
| 预扫描 | 无动作。 | 无动作。 |
| 指令首次扫描 | Out=0<br>控制算法不执行 | Out=0<br>控制算法不执行 |
| 指令首次执行 | Out=0<br>控制算法不执行 | Out=0<br>控制算法不执行 |
| 输入使能(EnableIn)清零 | 输出使能(EnableOut)清零。 | 无影响 |
| 输入使能(EnableIn)置位 | 执行指令。<br>输出使能(EnableOut)置位 | 输入使能(EnableIn)一直。<br>置位。<br>指令执行。 |
| 后期扫描 | 无动作。 | 无动作。 |

**实例：**PI指令是具有比例和积分增益的调节指令。用户可设置积分增益系数，单位：弧度/秒；用于设置PI校正装置的基本频率响应。比例增益系数设置整个块的增益，包括块的比例和积分增益。

除了初始化和保持/限定功能，下图显示了不使用初始化和保持/限制功能时，PI块在线性方式下的基本校正回路。

PI指令：线性方式



以下是PI指令作为速率调节器应用实例。在该例子中，速率误差量是由系统的参考速率(通过SCRV指令)减去速率反馈信号(参见PMUL指令实例)得到的。速率误差量直接被传送到PI指令，PI指令将上面图示的功能作用在该信号上。

## 结构文本

```
Reference_Select.In1 := Master_Machine_Ref;
Reference_Select.Select1 := Master_Machine_Select;
Reference_Select.In2 := Section_Jog;
Reference_Select.Select2 := Jog_Select;
SSUM(Reference_Select);

S_Curve.In := Reference_Select.Out;
S_Curve.AccelRate := accel_rate;
S_Curve.DecelRate := accel_rate;
SCRV(S_Curve);

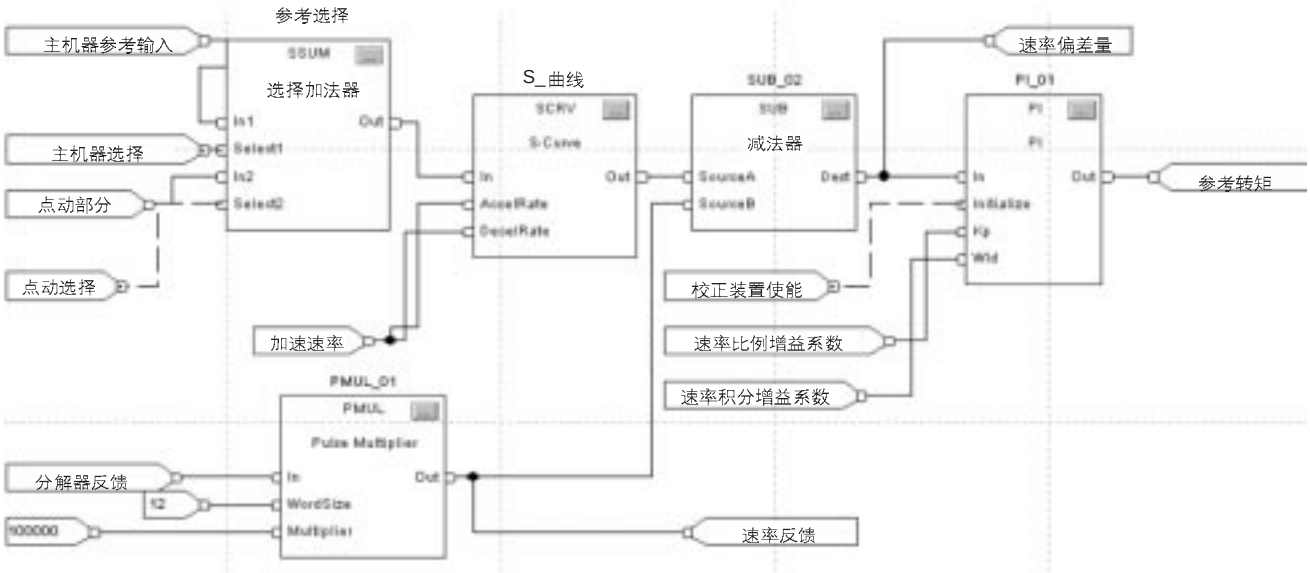
PMUL_01.In := Resolver_Feedback;
PMUL_01.WordSize := 12;
PMUL_01.Multiplier := 100000;
PMUL(PMUL_01);

Speed_Feedback := PMUL_01.Out;
Velocity_Error := S_Curve.Out - Speed_Feedback;
```

```
PI_01.In := Velocity_Error;
PI_01.Initialize := Enable_Regulator;
PI_01.Kp := Velocity_Proportional_Gain;
PI_01.Wld := Velocity_Integral_Gain;
PI(PI_01);

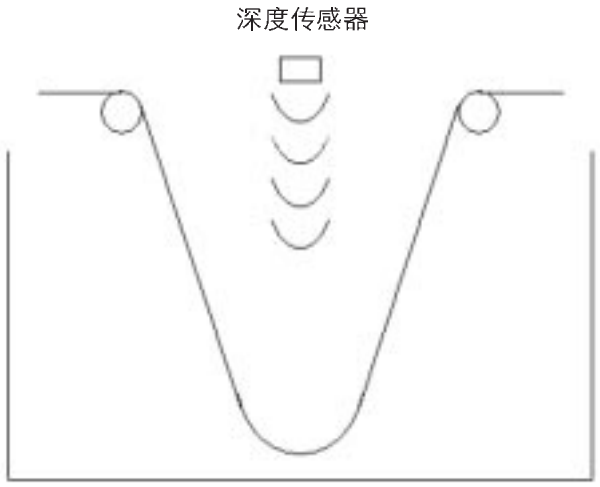
Torque_Reference := PI_01.Out;
```

## 功能块




在非线性方式下，PI指令增益系数可由功能块输入偏差量函数修正。该功能允许配合增益控制，用于建立一个校正装置模型，此时输出同被校正的过程量更加吻合。以下例子是该方式可用于垂线控制中，其中的反馈信号来自于一个处于环形孔的传感器，这样不能反射线性信号，无法代表实际存储的原料的数量。此时，PI调节器的比例增益应该被修正，以便更好的模拟该过程且不使用积分增益部分。使用积分增益部分可能引起频繁的过饱和。

## PI指令：非线性方式



## 脉冲倍频器 (PMUL)

PMUL指令通过计算两次扫描输入信号间的变化量来提供一个从位置输入模块(例如解码器或编码器反馈模块)到数字系统的接口。用户通过选择一个特定的字长，使PMUL指令正确分辨在连续线性输入方式下是否越过饱和边界。

### 操作数：

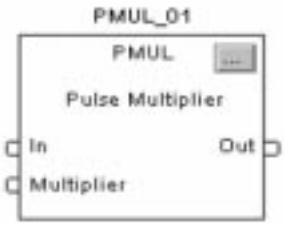


```
PMUL(PMUL_tag);
```

#### 结构文本

| 操作数： | 类型： | 格式： | 说明： |
|---|---|---|---|
| PMUL标签 | PULSE_MULTIPLIER | 结构体 | PMUL结构体 |






#### 功能块

| 操作数： | 类型： | 格式： | 说明： |
|---|---|---|---|
| PMUL标签 | PULSE_MULTIPLIER | 结构体 | PMUL结构体 |

#### PULSE_MULTIPLIER结构体

| 输入参数： | 数据类型： | 说明： |
|---|---|---|
| EnableIn | BOOL | 功能块：<br>如果该位清零，则指令不执行且输出不更新。<br>如果置位，指令执行。<br>缺省状态为置位<br>结构文本：<br>无影响。指令一直执行。 |
| In | DINT | 指令的模拟量输入信号。<br>有效值＝长整数<br>缺省值＝0 |
| Initialize | BOOL | 初始化输入。若该位置位，输出值Out将保持在0.0且所有的内部寄存器被清零。若由置位变为清零，则Inn-1=InitialValue(在绝对方式下无效)。在该位清零时，指令正常执行。如果WordSize参数无效，指令忽略Initialize参数。<br>缺省状态为清零。 |
| InitialValue | DINT | 输入初始化值。若Initialize由置位变为清零，则Inn-1=InitialValue。<br>有效值＝长整数<br>缺省值=0 |

| 输入参数： | 数据类型： | 说明： |
|---|---|---|
| Mode | BOOL | 输入方式选择。置位则使能关联方式，清零则使能绝对方式。<br>缺省状态为置位 |
| WordSize | DINT | 以位为单位的字长。在关联方式下计算(Ｉｎｎ－Ｉｎｎ－１)时指定使用的位数。WordSize在绝对方式下不使用。当输入值In的改变量大于1/2×2(WordSize-1)时，输出值Out改变符号。若WordSize无效，输出值Out被保持且指令置位状态字中的相应位。<br>有效值=2到32<br>缺省值=14 |
| Multiplier | DINT | 倍频系数。除以100,000后用来控制输入值In和输出值Out的比率。如果无效，指令限制该值且置位状态字中的相应位。<br>有效值=-1,000,000到1,000,000<br>缺省值=100,000 |

| 输出参数： | 数据类型： | 说明： |
|---|---|---|
| EnableOut | BOOL | 输出使能。 |
| Out | REAL | 指令的输出结果。若计算输出值时溢出，则输出值Out被强置设为+/-∞且置位状态字中的相应位。输出结果将影响算术状态标志设置。 |
| Status | DINT | 功能块的状态字。 |
| InstructFault(Status.0) | BOOL | 指令检测以下的运行错误。该错误不使控制器产生次要或主要错误。检测其它状态位以确定发生错误的类型。 |
| WordSizeInv(Status.1) | BOOL | 字长(WordSize)值无效 |
| OutOverflow(Status.2) | BOOL | 内部输出计算溢出时置位 |
| LostPrecision(Status.3) | BOOL | Out<-2e24或Out>2e24时置位。若指令在将一个整数输入转换为实数输出时的结果值大于\|2e24\|，则数据将丢失。这是因为REAL数据类型的最大限制是\|2e24\| |
| MultiplierInv(Status.4) | BOOL | 倍频系数值无效 |

**说明：** PMUL在关联或绝对方式下运行。

在关联方式下，指令的输出是两次扫描间输入值的差值乘上(Multiplier/100,000)。在关联方式下，指令在每次扫描过程中保存除法操作后的余数，并且将保存的余数值加回到下一次扫描。在该方式下，数位信息不会在运行过程中丢失。




在绝对方式下，指令可以按比例量化输入值，比如整定输入位置量，且在每次扫描之间不丢失任何信息。




## 计算输出值和余数值

无论在关联或绝对方式下，PMUL指令使用下面方程式计算：

Ans=((DiffInput × Multiplier)+INT_Remainder)

INT_Out=Ans/100,000

INT_Remainder=Ans-(INT_Out × 100,000)

Out=INT_Out

**算术状态标志：** 输出值将影响算术状态标志设置。

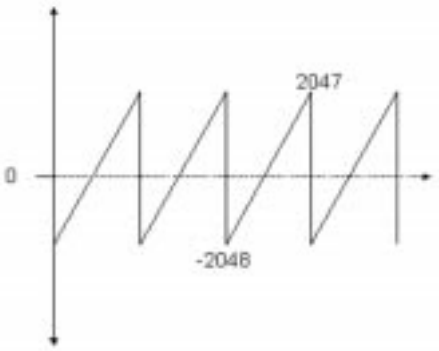
**故障条件：**无

**执行：**

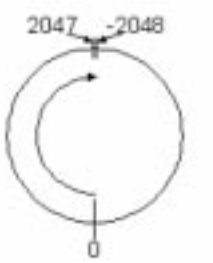
| 条件： | 功能块动作： | 结构文本动作： |
|---|---|---|
| 预扫描 | 无动作 | 无动作 |
| 指令首次扫描 | Inn-1=Inn<br>余数=0 | Inn-1=Inn<br>余数=0 |
| 指令首次执行 | Inn-1=Inn<br>余数=0 | Inn-1=Inn<br>余数=0 |
| 输入使能(EnableIn)清零 | 输出使能(EnableOut)清零，指令不执行，输出不更新 | 不影响 |
| 输入使能(EnableIn)置位 | 执行指令。<br>输出使能(EnableOut)置位。 | 输入使能(EnableIn)一直置位。<br>指令执行。 |
| 后期扫描 | 无动作。 | 无动作。 |

**实例1：**PMUL指令最常用于操作的关联方式中。在该方式下，PMUL指令可用于多种目的。首先，在关联方式下，PMUL指令计算它在每次扫描输入过程中接收到的数据的差值。接收到数据后，指令输出上次扫描与该次扫描输入数据的差值。比如在第n次扫描，输入值In=500，在第n+1次扫描，输入值In=600，则第n+1次扫描的输出数据Out=100。

第二，当处于关联操作方式时，PMUL指令还会对从反馈模块反馈来的二进制数的“回卷”信号进行补偿。比如，一个解码器反馈模块具有12位的分辨率，用带符号的二进制数表示时范围是从-2048到2047。根据从反馈模块得到的原始数据，反馈设备的循环可以用下图表示：



在该例中，若反馈数据的值从2047变化到-2048，位置值的有效变化量等价于4095次计数脉冲，但实际情况是位置值的变化量根据分解器反馈装值的循环只有1次计数。如果指令从反馈模块得到数据的真实字长，指令将按循环的方式监测数据，如下图所示：




已知输入该功能块数据的真实字长，PMUL指令在输入到指令的值从2047回卷至-2048时输出1个计数，而非数学计算得到的4095。

使用该功能块时，如果指令中循环方向与实际不同，应当注意两次扫描间反馈数据的变化量不能超过真实字长的1/2。在上面的例子中，如果设备反馈值沿着顺时针方向变化，在扫描`A`读取值为0且在扫描`B`读取值为-2000，实际的位值变化量等价于在顺时针方向上+2096次计数。然而，因为这两次的值都超过真实字长的1/2(或者说大于物理设备循环的1/2)，PMUL指令会在反馈装置循环的相反方向上计算，返回值是-2000而非+2096。

使用脉冲倍频器功能块的第三个目的是该功能块包含了从两次扫描的余数的小数部分。该小数部分作为缩放比例因子Multiplier/100,000缩放计算的结果。当每次执行结束时，上一次扫描的余数被加回到总的当前值，这样所有的计数或脉冲信号最终都参与计算，系统中没有数据丢失。功能块的输出值Out是浮点数据类型，输出总是产生全部的计数次数值。

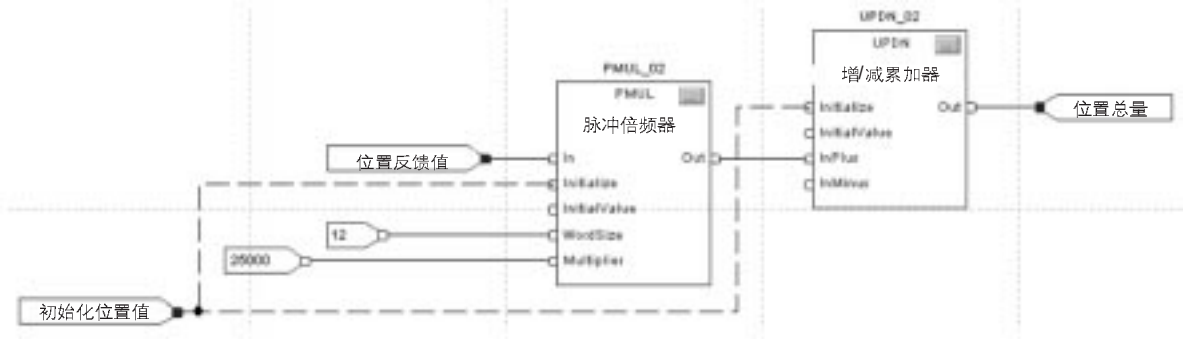
## 结构文本

```
PMUL_02.In := Position_feedback;
PMUL_02.Initalize := Initialize_Position;
PMUL_02.WordSize := 12;
PMUL_02.Multiplier := 25000;
PMUL(PMUL_02);

UPDN_02.Initialize := Initialize_Position;
UPDN_02.InPlus := PMUL_02.Out;
UPDN(UPDN_02);

Total_Position := UPDN_02.Out;
```

## 功能块




设定Initial_Position=0且Multiplier=25000=>(25,000/100,000)

| 扫描(scan)： | 数位反馈值(Position_Feedback)： | PMUL_02.Out： | 位置总量： |
|---|---|---|---|
| n | 0 | 0 | 0 |
| n+1 | 1 | 0 | 0 |
| n+2 | 2 | 0 | 0 |
| n+3 | 3 | 0 | 0 |
| n+4 | 4 | 1 | 1 |
| n+5 | 5 | 0 | 1 |

**实例2：**在控制电子主传动轴的应用系统中，电动机A的转速反馈信号作为电动机B的主参考输入，电动机B按照此值转动。电动机A的反馈信号作为“位置反馈值”，电动机B的反馈信号作为“从动位置反馈值”。由于两个指令的倍频数比率为1/4，电动机A每转4圈，电动机B必须转1圈，目的是为了保持在UPDN累加器中的累加值为0。UPDN指令的输出值中除0外的其他值都被看作位置偏差量，校正该值后回馈到电动机B以保持两个电动机之间的相位同步锁定。

### 结构文本

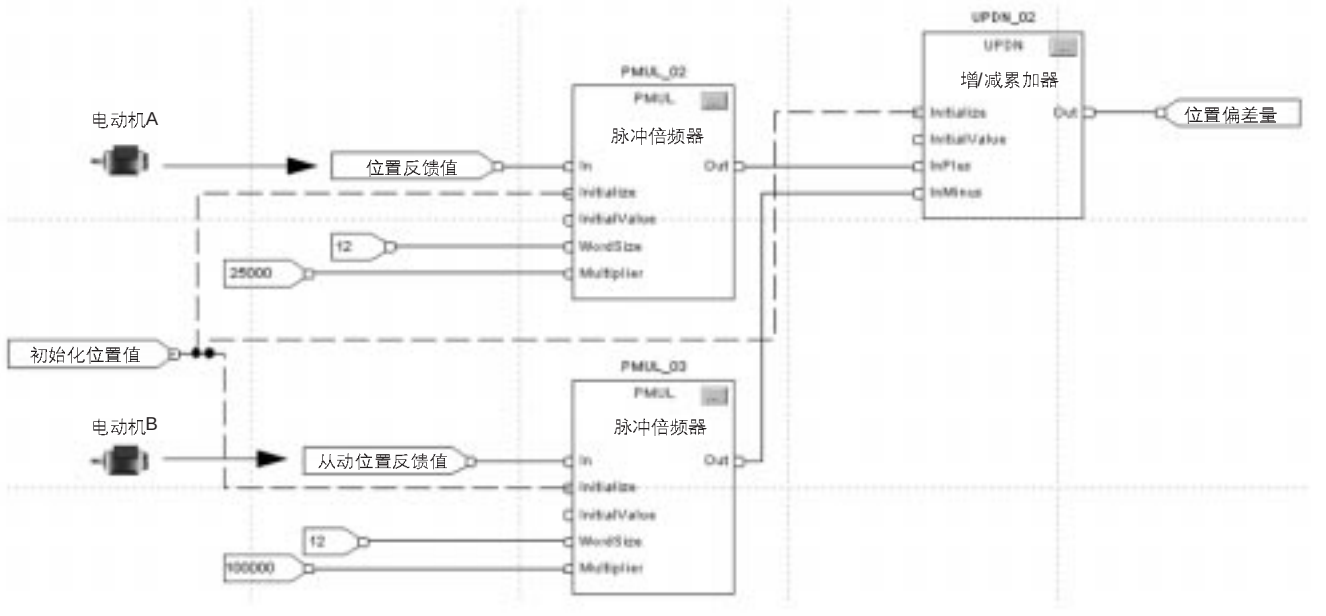
```
PMUL_02.In := Position_feedback;
PMUL_02.Initalize := Initialize_Position;
PMUL_02.WordSize := 12;
PMUL_02.Multiplier := 25000;
PMUL(PMUL_02);

PMUL_03.In := Follower_Position;
PMUL_03.Initalize := Initialize_Position;
PMUL_03.WordSize := 12;
PMUL_03.Multiplier := 100000;
PMUL(PMUL_03);

UPDN_02.Initialize := Initialize_Position;
UPDN_02.InPlus := PMUL_02.Out;
UPDN_02.InMinus := PMUL_03.Out;
UPDN(UPDN_02);

Position_error := UPDN_02.Out;
```

## 功能块


UPDN_02
UPDN
增/减累加器
Initialize
InitialValue
InPlus
InMinus
Out
位置偏差量
PMUL_02
PMUL
脉冲倍频器
In
Out
Initialize
InitialValue
WordSize
Multiplier
电动机A
位置反馈值
12
25000
初始化位置值
PMUL_03
PMUL
脉冲倍频器
电动机B
从动位置反馈值
12
100000

## S-Curve(SCRV)

SCRV指令实现一个具有附加的加加速度斜坡函数。加加速度是加速度改变的最大比率，指令利用该参数由输入值求出输出值。

### 操作数：

#### 结构文本

| 操作数： | 类型： | 格式： | 说明： |
|---|---|---|---|
| SCRV标签 | S_CURVE | 结构体 | SCRV结构体 |

#### 功能块

| 操作数： | 类型： | 格式： | 说明： |
|---|---|---|---|
| SCRV标签 | S_CURVE | 结构体 | SCRV结构体 |

### S_CURVE结构体

| 输入参数： | 数据类型： | 说明： |
|---|---|---|
| EnableIn | BOOL | 功能块：<br>如果该位清零，则指令不执行且输出不更新。<br>如果置位，指令执行。<br>缺省状态为置位<br>结构文本：<br>无影响。指令一直执行。 |
| In | REAL | 指令的模拟量输入信号。<br>有效值＝浮点数<br>缺省值＝0.0 |
| Initialize | BOOL | 指令输入初始化。若被置位，指令保持输出值Out等于InitialValue的值。<br>缺省状态为清零 |
| InitialValue | REAL | S-Curve的初始化值。当Initialize被置位时，输出值Out=InitialValue。<br>有效值= 浮点数<br>缺省值=0.0 |
| AbsAlgRamp | BOOL | 斜坡类型。若置位，指令作为绝对值斜坡函数。若清零，指令作为代数值斜坡函数。<br>缺省状态为置位。 |
| AccelRate<br>AccelRate<0， | REAL | 加速度参数，单位：Units/二次方秒。如果值为0则不允许输出加速。若<br>指令假定AccelRate=0且置位状态字中的相应位。<br>有效值=0.0到最大正浮点数<br>缺省值=0.0 |
| DecelRate | REAL | 减速度参数，单位：Units/二次方秒。如果值为0则不允许输出减速。若<br>DecelRate<0，指令假定DeceRate=0且置位状态字中的相应位。<br>有效值=0.0到最大正浮点数<br>缺省值=0.0 |

| 输入参数： | 数据类型： | 说明： |
|---|---|---|
| JerkRate | REAL | 加加速度参数，单位：Units/三次方秒。当输出与输入呈斜坡关系时，该参数指定了加速度或减速度的最大变化率。当(JerkRate ×DeltaT) ⩾AccelRate和/或DecelRate时，加速度和减速度的速率不受限制。在此情况下，指令的表现为斜坡函数。若JerkRate<0，指令设定JerkRate=0且置位状态字中的相应位。<br>有效值=0.0到最大正浮点数<br>缺省值=0.0 |
| HoldMode | BOOL | S-Curve保持方式参数。该参数与HoldEnable同时使用。当HoldEnable被置位且Rate=0时，如果HoldMode被置位，指令保持Out不变。在此情况下，若HoldEnable置位，指令保持Out值，JerkRate参数被忽略，且输出值Out曲线形状出现一个转折。HoldEnable被置位时，如果HoldMode被清零，指令使用JerkRate将输出值Out保持为常数。当Rate=0时，输出值Out保持。不能在HoldEnable置位时改变HoldMode的状态，因为指令忽略这个变化。<br>缺省状态为清零 |
| HoldEnable | BOOL | S-Curve保持使能参数。置位时，输出值Out被保持。清零时，Out从当前值开始增加直至等于输入值In。<br>缺省状态为清零。 |
| TimingMode | DINT | 选择定时执行方式。<br>**有效值：** **说明：**<br>0 周期执行方式<br>1 过采样方式<br>2 实时采样方式<br>需要了解更多与定时方式有关的信息，请参阅附录：功能块属性。<br>有效值=0到2<br>缺省值=0 |
| OversampleDT | REAL | 过采样方式的执行时间。<br>有效值=0到4194.303秒<br>缺省值=0 |
| RTSTime | DINT | 实时采样方式下模块的更新周期。<br>有效值=1到32,767ms<br>缺省值=1 |
| RTSTimeStamp | DINT | 实时采样方式下模块的时间标签。<br>有效值=0到32,767ms<br>缺省值=0 |

| 输出参数： | 数据类型： | 说明： |
|---|---|---|
| EnableOut | BOOL | 输出使能。 |
| S_Mode | BOOL | S_模式值。当(Jerk ×DeltaT) ⩽Rate且Rate<Accel或Decel时，S_模式置位；否则S_模式清零。 |
| Out | REAL | S-Curve指令的输出结果。输出结果将影响算术状态标志设置。 |
| Rate | REAL | 输出值Out内部变化的速率，单位：Units/秒 |
| DeltaT | REAL | 两次更新之间经过的时间，单位：秒。作为控制算法计算过程量输出值的时间间隔参数。 |
| Status | DINT | 功能块的状态字。 |

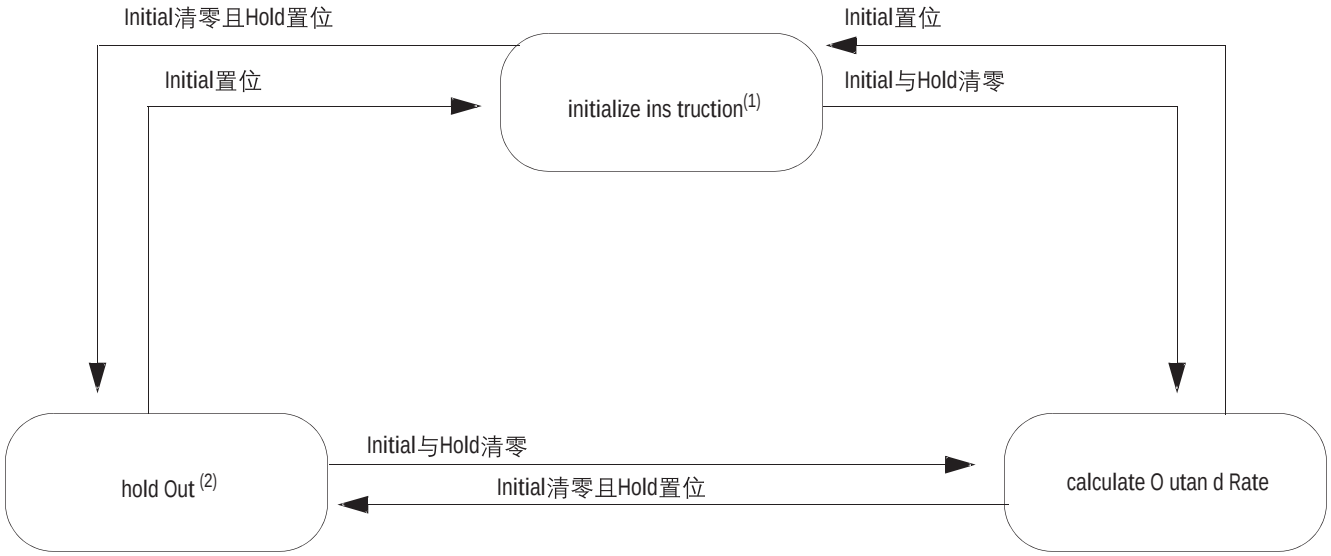
| 输出参数： | 数据类型： | 说明： |
|---|---|---|
| InstructFault(Status.0) | BOOL | 指令检测以下的运行错误。该错误不使控制器产生次要或主要错误。检测其它状态位以确定发生错误的类型。 |
| AccelRateInv(Status.1) | BOOL | AccelRate为负。 |
| DecelRateInv(Status.2) | BOOL | DecelRate为负。 |
| JerkRateInv(Status.3) | BOOL | JerkRate为负。 |
| TimingModeInv(Status.27) | BOOL | 定时方式值无效。<br>需要了解更多与定时方式有关的信息，请参阅附录：功能块属性。 |
| RTSMissed(Status.28) | BOOL | 只用于实时采样方式。当ABS \| DeltaT - RTSTime \| > 1 (0.001秒)时，该位被置位。 |
| RTSTimeInv(Status.29) | BOOL | RTSTime值无效。 |
| RTSTimeStampInv(Status.30) | BOOL | RTSTimeStamp值无效。 |
| DeltaT (Status.31) | BOOL | DeltaT值无效。 |

**说明：**使用SCRV指令最基本的要求是确保速度的变化率不会大于指定的加加速度极限。

用户可通过组态SCRV指令为阶跃输入生成S曲线或斜坡曲线轮廓。

| SCRV轮廓 | 说明： |
|---|---|
| S曲线轮廓 | 为生成S曲线轮廓，设置加加速度为(JerkRate*DeltaT)<AccelRate和/或DecelRate.<br><br>在S曲线轮廓模式中，SCRV指令比率的变化速度不会超过指定的加加速度。用于S曲线轮廓的算法可为阶跃输入生成平滑、均衡的S曲线。输出的梯形积分合并成该S曲线。最终，该轮廓的每部分的变化速率将小于加加速度。<br><br>当输入值发生一个阶跃变化时，输出值改变的速度从加加速度(jerk rate)增加到加速度极限值(AccelRate)或减速度极限值(DecelRate)。然后该速度保持加速度极限值(AccelRate)或减速度极限值(DecelRate)，直到加速度必须减少的点。速度在小于或等于加加速度(jerk rate)后以开始减少，最后加加速度等于0时，输出值等于输入值。<br><br>某些情况下，根据加速度值、减速度值、加加速度，可能在还未达到加速度极限值或减速度极限值时就必须开始按加加速度减少。<br><br>对于很小的阶跃变化，SCRV指令不会尝试生成′S′轮廓。该方式下，整个阶跃将产生输出且速率会影响输出的变化。如果输出Out等于输入In，该动作将发生；如果速率小于或等于预定的加加速度(jerkrate)，输入将发生下步变化。<br><br>SCRV指令支持代数值斜坡和绝对值斜坡。对于代数值斜坡，加速度的条件是输入更大的正数；减速度的条件是输入更小的负数。对于绝对值斜坡，加速度的条件是输入远离零点；减速度的条件是输入向零的方向靠近。 |
| 斜坡轮廓 | 为生成斜坡轮廓，设置加加速度为(JerkRate*DeltaT)大于等于AccelRate和或DecelRate。<br><br>在斜坡轮廓方式中，SCRV指令一直生成变化速率等于预定的AccelRate或DecelRate的曲线，直至输出与输入间差值需要减小，AccelRate或DecelRate达到终点。<br><br>HoldMode=0时与HoldMode＝1操作相同。当HoldEnable置位，输出立刻保持且比率清零。 |

下图说明指令如何修改输出值。




(1) 当指令初始化位置位，指令动作如下：

$Out_n$=InitialValue

$Out_{n-1}$=$Out_n$

$Rate_n$=0

$Rate_{n-1}$=0

(2) 当HoldMode清零，输出值Out向着输入值In靠近，当HoldEnable被置位时，速度开始以加加速度向零减少。由于加加速度(JerkRate)的存在，无论速度值大小，减少到零时输出值Out被保持在当时的值。当输出值Out最终保持恒定时，输出值是一个和HoldEnable刚被置位时不同的值。

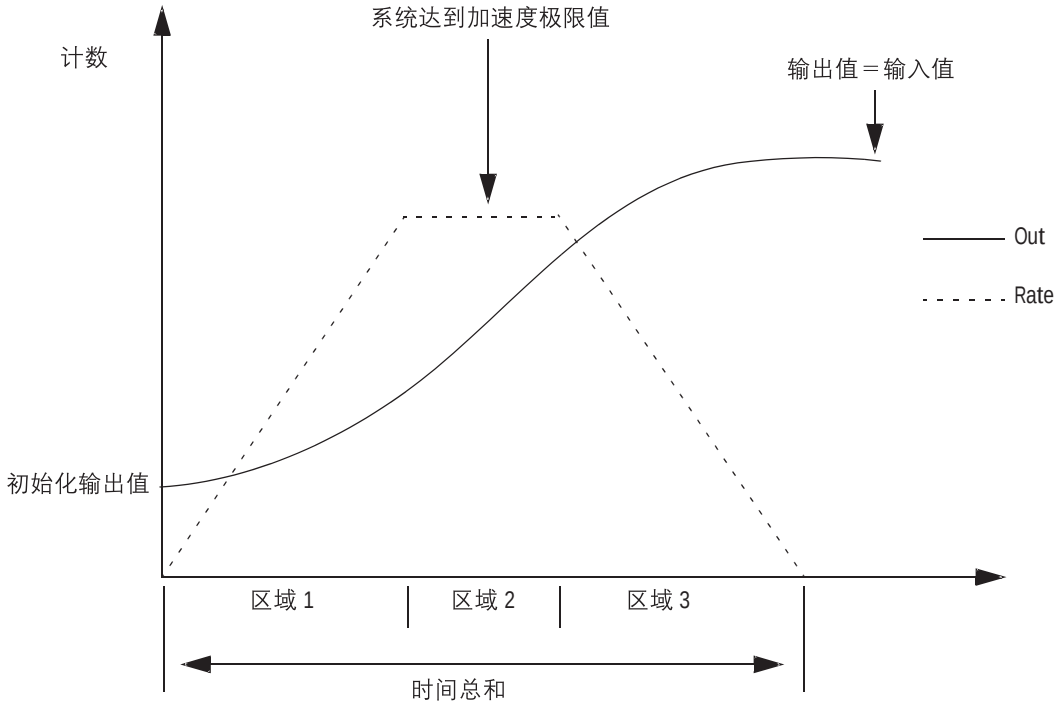
当HoldMode被置位，输出值Out向着输入值In方向靠近，且HoldEnable被置位时，速度立即被置零。输出值Out被保持在HoldEnable被置位时的值。

在转换的过程中减少加加速度(JerkRate)可能会导致输出值Out超过输入值In。如果发生跳变，则强制使用输入的加加速度(JerkRate)。用户可以通过在Out=In时(不是在转换过程中)小幅度调节或改变加加速度的值，使之减少进而避免输出Out超过输入In。

输出值Out需要一段时间后等于输入值In的变化，这是加速度极限值(AccelRate)，加加速度(JerkRate)和输入In与输出Out间偏差值间相互作用产生的结果。

## 计算输出值和速率值

在从初始值到最后结果值转换的过程中，输出值Out经历3个区域。在区域1和区域3，输出值Out的变化速率以加加速度为基础。在区域2，输出值Out的变化速率以加速度极限值(AccelRate)或减速度极限值(DecelRate)为基础。




每个区域输出值Out的计算如下：

$$\text{TotalTime} = \frac{\text{FinalOutput-InitialOutput}}{\text{AccelRate}} + \frac{\text{AccelRate}}{\text{JerkRate}}$$

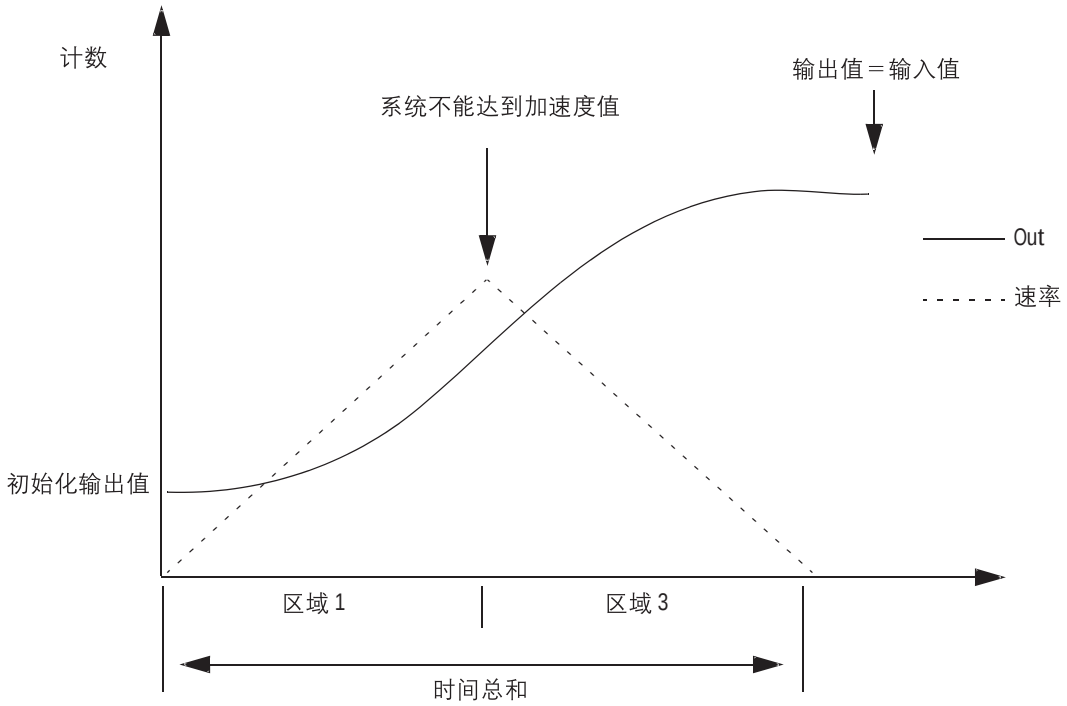
每个区域使用下面公式：

| **区域：** | **公式：** |
|---|---|
| 区域1 | $Time_1 = \frac{AccelRate}{JerkRate}$<br>$Y(Time) = InitialOutput + \frac{1}{2}(JerkRate) \times Time^2$ |
| 区域2 | $Time_2 = \frac{JerkRate \times (FinalOutput - InitialOutput) - AccelRate^2}{JerkRate \times AccelRate}$<br>$Y(Time) = InitialOutput + (AccelRate \times Time) - \frac{AccelRate^2}{2 \times JerkRate}$ |
| 区域3 | $Time_3 = \frac{AccelRate}{JerkRate}$<br>$Y(Time) = FinalOutput - \frac{1}{2}(JerkRate) \times (Time - \frac{FinalOutput - InitiaOutput}{AccelRate} - \frac{AccelRate}{JerkRate})^2$ |

当：

$$|InitialOutput - FinalOutput| < \frac{AccelRate^2}{JerkRate}$$

时，SCRV功能块输出值的变化速率不能达到加速度极限值(AccelRate)或减速度极限值(DecelRate)。输出值Out如下图所示：




其中：

$$TotalTime = \sqrt{\frac{InitialOutput - FinalOutput}{JerkRate}}$$

**算术状态标志：**输出值将影响算术状态标志设置。

**故障条件：**无

### 执行：

| 条件： | 功能块动作： | 结构文本动作： |
|---|---|---|
| 预扫描 | 无动作。 | 无动作。 |
| 指令首次扫描 | 初始化内部变量 | 初始化内部变量 |
| 指令首次执行 | 无动作 | 无动作 |
| 输入使能(EnableIn)清零 | 输出使能(EnableOut)清零。指令不执行，输出不更新。 | 不影响 |
| 输入使能(EnableIn)置位 | 执行指令。<br>输出使能(EnableOut)置位 | 使能输入一直置位。<br>指令执行。 |
| 后期扫描 | 无动作 | 无动作 |

**实例：**在大多数协调驱动应用系统中，主参考输入控制整个驱动组的线速度。由于选择不同的参考输入，驱动过程有不同的惯性负荷、电动机转矩等输入量，而且调节过程不允许独立的驱动部件在协调方式下起作用，因此不能在参考速度使用阶跃变化量。SCRV指令被设计用于斜坡化和修正这些驱动部件的参考输入信号，以便加速度，减速度，加加速度(加速度的导数)等均被控制。该指令提供允许驱动部件的参考输入达到设定点的一种机制，并且可以消去与之相联结机器或设备上多余的影响和强制的特定方式。

### 结构文本

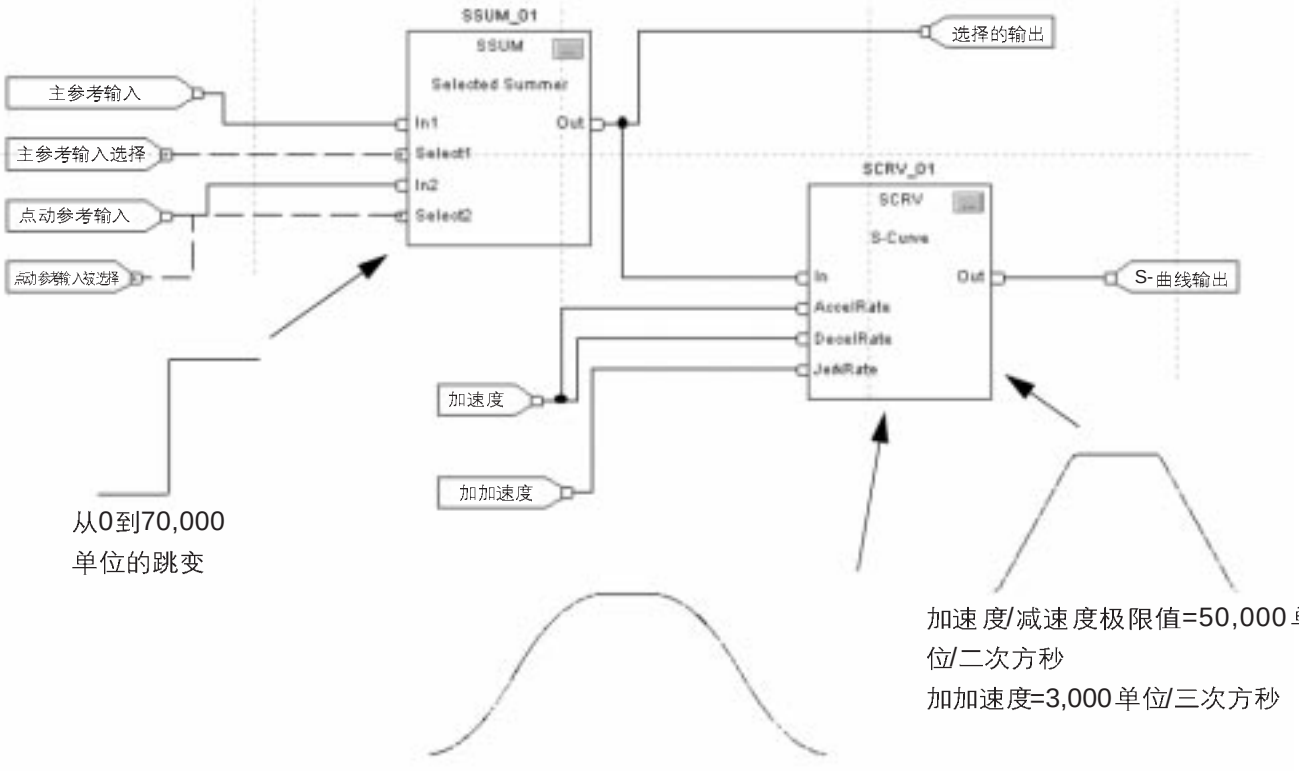
```
SSUM_01.In1 := Master_reference;
SSUM_01.Select1 := master_select;
SSUM_01.In2 := Jog_reference;
SSUM_01.Select2 := jog_select;
SSUM(SSUM_01);

select_out := SSUM_01.Out;

SCRV_01.In := select_out;
SCRV_01.AccelRate := accel;
SCRV_01.DecelRate := accel;
SCRV_01.JerkRate := jerk_rate;
SCRV(SCRV_01);

scurve_out := SCRV_01.Out
```

## 功能块


SSUM_01
SSUM
Selected Summer
主参考输入
主参考输入选择
点动参考输入
点动参输入选择
In1
Select1
In2
Select2
Out
选择的输出
SCRV_01
SCRV
S-Curve
In
Out
S-曲线输出
AccelRate
DecelRate
JerkRate
加速度
加加速度
从0到70,000
单位的跳变
加速度/减速度极限值=50,000单位/二次方秒
加加速度=3,000单位/三次方秒
加速度/减速度极限值=50,000
单位/二次方秒
加加速度=3,000单位/三次方秒

## 二阶控制器(SOC)

SOC指令设计应用于闭环控制系统中。该指令有与PI指令相类似的运行方式。SOC指令提供一个增益项，一个一阶滞后校正项，一个二阶超前校正项。

### 操作数：

#### 结构文本

| 操作数： | 类型： | 格式： | 说明： |
|---|---|---|---|
| SOC标签 | SEC_ORDER_CONTROLLER | 结构体 | SOC结构体 |

#### 功能块

| 操作数： | 类型： | 格式： | 说明： |
|---|---|---|---|
| SOC标签 | SEC_ORDER_CONTROLLER | 结构体 | SOC结构体 |

### SOC_ORDER_CONTROLLER结构体：

| 输入参数： | 数据类型： | 说明： |
|---|---|---|
| EnableIn | BOOL | 功能块：<br>如果该位清零，则指令不执行且输出不更新。<br>如果置位，指令执行。<br>缺省状态为置位<br>结构文本：<br>无影响。指令一直执行。 |
| In | REAL | 指令的模拟量输入信号。<br>有效值＝浮点数<br>缺省值＝0.0 |
| Initialize | BOOL | 指令初始化命令。若置位，输出值Out和内部积分器的值设置为InitialValue的值。<br>缺省值为清零。 |
| InitialValue | REAL | 初始化输入值。当Initialize被置位时，Out和内部积分器的值设为InitialValue的值。<br>InitialValue的值通过HighLimit和LowLimit限制。<br>有效值=浮点数<br>缺省值=0.0 |
| Gain | REAL | 指令的比例增益系数。如果该值超限，指令限制该值为极限值并置位状态字中的相应位。<br>有效值=浮点数>0.0<br>缺省值=最小正浮点数 |
| WLag | REAL | 一阶滞后角频率，单位：弧度/秒。如果该值超限，指令限制该值为极限值且置位状态字中的相应位。<br>有效值=参见下面有关有效值范围说明<br>缺省值=最大正浮点数 |
| WLead | REAL | 二阶超前角频率，单位：弧度/秒。如果该值超限，指令限制该值为极限值且置位状态字中的相应位。<br>有效值=参见下面有关有效值范围说明<br>缺省值=0.0 |

| 输入参数： | 数据类型： | 说明： |
|---|---|---|
| ZetaLead | REAL | 二阶超前阻尼因子。如果该值超限，指令限制该值为极限值且置位状态字中的相应位。<br>有效值=0.0到10.0<br>缺省值=0.0 |
| HighLimit | REAL | 上限值，即输出的最大值。若HighLimit ≤LowLimit则指令置位HighAlarm和LowAlarm，置位状态字中的相应位，并设置Out=LowLimit。<br>有效值=LowLimit<HighLimit ≤最大正浮点数<br>缺省值=最大正浮点数 |
| LowLimit | REAL | 下限值，即输出的最小值。若HighLimit ≤LowLimit则指令置位HighAlarm和LowAlarm，置位状态字中的相应位，且设置Out=LowLimit。<br>有效值=最小负浮点数≤LowLimit<HighLimit<br>缺省值=最小负浮点数 |
| HoldHigh | BOOL | 上限保持命令。若置位，内部积分器的值不允许增大。<br>缺省状态为清零 |
| HoldLow | BOOL | 下限保持命令。若置位，内部积分器的值不允许减小。<br>缺省状态为清零 |
| TimingMode | DINT | 选择定时执行方式。<br>**有效值：　　说明：**<br>0　　周期执行方式<br>1　　过采样方式<br>2　　实时采样方式<br>需要了解更多与定时方式有关的信息，请参阅附录：功能块属性。<br>有效值=0到2<br>缺省值=0 |
| OversampleDT | REAL | 过采样方式的执行时间。<br>有效值=0到4194.303秒<br>缺省值=0 |
| RTSTime | DINT | 实时采样方式下模块的更新周期。<br>有效值=1到32,767ms<br>缺省值=1 |
| RTSTimeStamp | DINT | 实时采样方式下模块的时间标签。<br>有效值=0到32,767ms<br>缺省值=0 |

| 输出参数： | 数据类型： | 说明： |
|---|---|---|
| EnableOut | BOOL | 输出使能。 |
| Out | REAL | 由算法计算出的输出结果。输出结果将影响算术状态标志设置。 |
| HighAlarm | BOOL | 上限报警指示。当输出值Out ≥HighLimit且输出被限制在HighLimit时置位。 |
| LowAlarm | BOOL | 下限报警指示。当输出值Out ≤LowLimit且输出被限制在LowLimit时置位。 |
| DeltaT | REAL | 两次更新之间经过的时间，单位：秒。作为控制算法计算过程量输出值的时间间隔参数。 |
| Status | DINT | 功能块的状态字。 |
| InstructFault(Status.0) | BOOL | 指令检测以下的运行错误。该错误不使控制器产生次要或主要错误。检测其它状态位以确定发生错误的类型。 |
| GainInv(Status.1) | BOOL | Gain>最大值 或Gain<最小值。 |
| WLagInv(Status.2) | BOOL | WLag>最大值或WLag<最小值。 |
| WLeadInv(Status.3) | BOOL | WLead>最大值或WLead<最小值。 |
| ZetaLeadInv(Status.4) | BOOL | ZetaLead>最大值或ZetaLead<最小值。 |
| HighLowLimsInv(Status.5) | BOOL | HighLimit ≤LowLimit。 |
| TimingModeInv(Status.27) | BOOL | 定时方式值无效。<br>需要了解更多与定时方式有关的信息，请参阅附录：功能块属性。 |
| RTSMissed(Status.28) | BOOL | 只用于实时采样方式。当ABS| DeltaT - RTSTime | >1(0.001秒)时，该位被置位。 |
| RTSTimeInv(Status.29) | BOOL | RTSTime值无效。 |
| RTSTimeStampInv(Status.30) | BOOL | RTSTimeStamp值无效。 |
| DeltaT(Status.31) | BOOL | DeltaT值无效。 |

**说明：** SOC指令提供带一个增益项，一个一阶滞后校正项，一个二阶超前校正项的校正算法。滞后频率和超前的频率和阻尼因子均可调整。二阶超前的零点对可以是复数(阻尼比小于1)或实数(阻尼比大于等于1)。SOC指令是为扫描速率为常数的任务而设计的。

SOC指令使用下面的拉普拉斯传递函数。

$$H(s) = \frac{K\left(\frac{s^2}{\omega_{Lead}^{\ 2}} + \frac{2 \times \xi_{Lead} \times s}{\omega_{Lead}} + 1\right)}{s\left(\frac{s}{\omega_{Lag}} + 1\right)}$$

## 参数极限

下列SOC参数为有效值极限：

| 参数： | 限值： |
|---|---|
| WLead | LowLimit=0.00001/DeltaT<br>HighLimit=0.07 –/DeltaT<br>此处DeltaT以秒为单位 |
| Wlag | LowLimit=0.0000001/DeltaT<br>HighLimit=0.07 –/DeltaT<br>此处DeltaT以秒为单位 |
| ZetaLead | LowLimit=0.0<br>HighLimit=10.0 |

无论计算的输出结果值无效或NAN，该指令设置Out为无效的值并置位算术溢出状态标志。内部参数不再更新。在以后的每次扫描中，指令使用的最后一次输出值有效时扫描的内部参数计算输出值。

极限
指令根据Hold输入参数的状态防止超调。

| 如果： | 则： |
|---|---|
| 若HoldHigh被置位且Integrator>Integratorn-1 | Integrator=Integratorn-1 |
| 若HoldLow被置位Integrator<Integratorn-1 | Integrator=Integratorn-1 |

指令也根据HighLimit和LowLimit的值防止积分过饱和。

| 如果： | 则： |
|---|---|
| Integrator>IntegratorHighLimit | Integrator=IntegratorHighLimit |
| Integrator<IntegratorLowLimit | Integrator=IntegratorLowLimit |

其中：

$$IntegratorHighLimit = HighLimit \times \frac{Gain \times WLag}{WLead^2}$$

$$IntegratorLowLimit = LowLimit \times \frac{Gain \times WLag}{WLead^2}$$

指令还能根据HighLimit和LowLimit的值限制Out的值。

| 如果： | 那么： |
|---|---|
| HighLimit ≤LowLimit | Out=LowLimit<br>Integrator=IntegratorLowLimit<br>HighLowLimsInv被置位<br>HighAlarm被置位<br>LowAlarm被置位 |
| Out ≥HighLimit | Out=HighLimit<br>Integrator=Integratorn-1<br>HighAlarm被置位 |
| Out ≤LowLimit | Out=LowLimit<br>Integrator=Integratorn-1<br>LowAlarm被置位 |

**算术状态标志：**输出值将影响算术状态标志设置。

**故障条件：**无

**执行：**

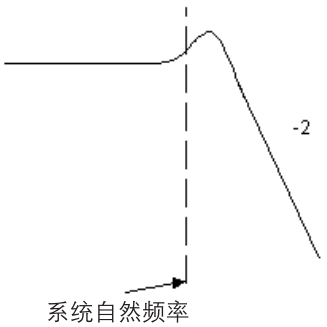
| 条件： | 功能块动作： | 结构文本动作： |
|---|---|---|
| 预扫描 | 无动作。 | 无动作。 |
| 指令首次扫描 | 指令设置内部参数且Out=0。 | |
| (instruction first scan) | 控制算法不执行。 | |
| 指令首次执行 | 指令设置内部参数且Out=0。 | |
| (instruction first run) | 控制算法不执行。 | |
| 输入使能(EnableIn)清零 | 输出使能(EnableOut)清零，指令不执行且输出不更新。 | 不影响 |
| 输入使能(EnableIn)置位 | 执行指令。<br>输出使能(EnableOut)置位 | 使能输入(EnableIn)一直置位。指令执行。 |
| 后期扫描 | 无动作 | 无动作 |

**实例：** SOC指令是一个特定的功能块，应用于能量通过弹性系统在两个不同部件传递的情况。通常情况下，在这一类型应用中，频率响应过程自身可以表现出像以下伯德图A所示的特性：

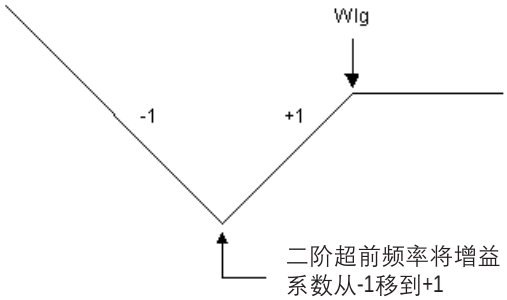
SOC指令包含一个一阶的滞后过滤器，与PID控制器使用积分、一个二阶零点(超前)和一个一阶极点(滞后)的传递函数相同。通过使用该指令PID调节过程被简化，因为校正项已调节好，用户只需将WLead和ZLead作为SOC指令的输入参数即可，远比选择PID指令中Kp，Ki，和Kd的值方便。SOC指令的传递函数如下：

$$H(s) = \frac{K\left(\frac{s^2}{\omega_{Lead}^2} + \frac{2 \times \xi_{Lead} \times s}{\omega_{Lead}} + 1\right)}{s\left(\frac{s}{\omega_{Lag}} + 1\right)}$$

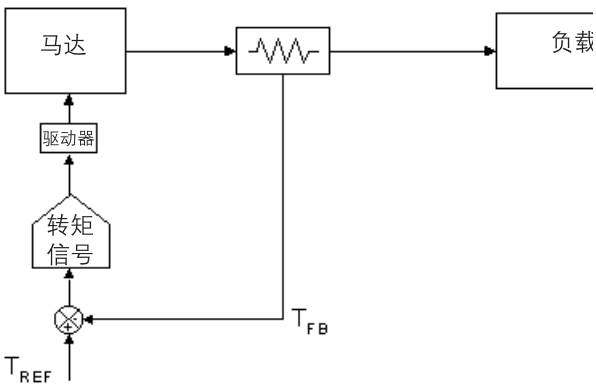
**图A：过程特性**



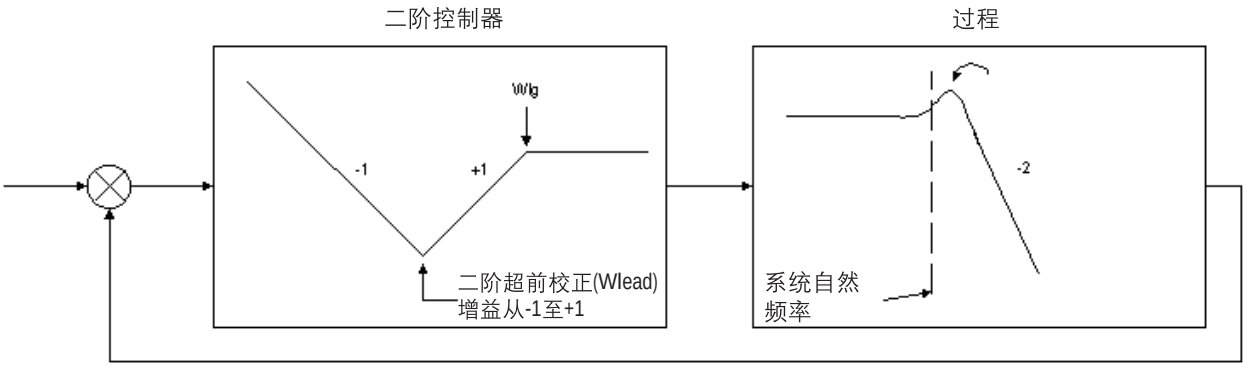

**图B：二阶控制器**



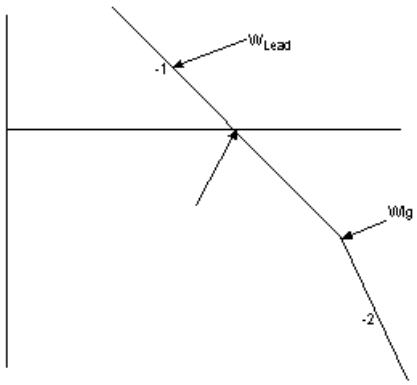

SOC指令可应用于扭矩或牵引力校正系统，此时测压元件或压力传感器用来作为反馈输入，调节方案的输出直接作用到扭矩(当前的)辅助驱动回路上。在很多类似的应用系统中，被控制的系统可能处于机械欠阻尼状态，具有自然频率，因此通过反馈装置时自身会发生反射现象，很难稳定。



使用SOC指令，PID调节过程更为简单，因为调节项被安排好，用户只需将WLead和ZLead作为SOC指令的输入即可，而不必考虑Kp,Ki和Kd的值。在这种方式下，控制器/调节器的角频率的调整变得很容易，设置过程面向实际工艺过程。启动过程中，系统的自然频率和阻尼因子可以根据经验来估量，或者以后现场测量，调节器的参数可以被调整同过程特性匹配，使最终过程具有更大增益和更好稳定性。




在上面的系统中，如果Wlead被设置与系统自然频率相等，且Wlag设置值高于期望的交叉频率(大于交叉频率的5倍)，系统响应结果如下图所示：




在一个实际的应用中，使用和设置该指令的步骤包括：

1. 识别被控制过程的类型。如果系统对阶跃函数的响应结果具有高阶的阻尼振荡或上述过程曲线的特性，该功能块可以提供稳定控制所要求的调节特性。

2. 决定系统/过程的自然频率。此时可以根据经验来估量，或现场测量。调整超前转折频率(WLead)使之等于或略超前过程本身的自然频率。

3. 调节阻尼因子，Zlead，使之可以消去系统的各种超调现象。

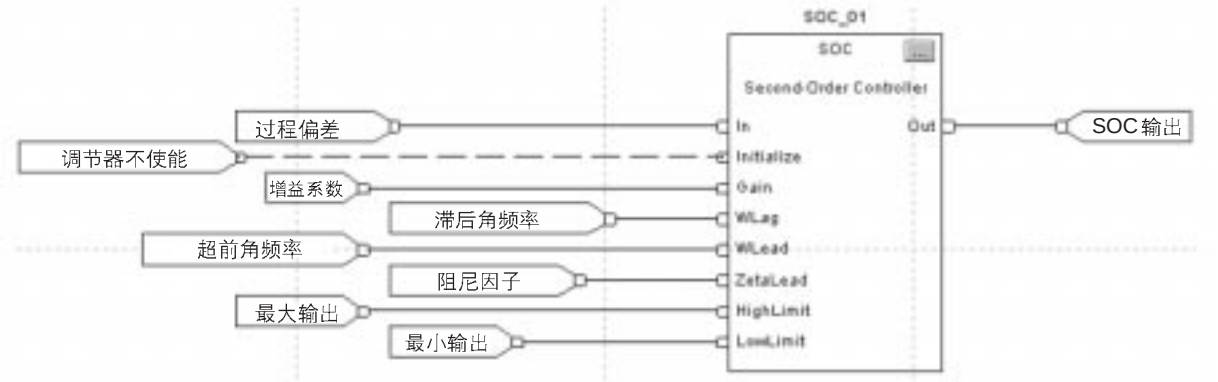
4. 移动WLag，使之远超过系统交叉频率(>5倍)，然后通过增加全面增益得到期望的系统响应。

## 结构文本

```
SOC_01.In := Process_Error;
SOC_01.Initialize := Regulator_Enable_Not;
SOC_01.Gain := Gain;
SOC_01.WLag := Lag_Radians_per_sec;
SOC_01.WLead := Lead_radians_per_sec;
SOC_01.ZetaLead := Damping_Factor;
SOC_01.HighLimit := Max_Out;
SOC_01.LowLimit := Min_Out;
SOC(SOC_01);

SOC_Out := SOC_01.Out;
```

## 功能块



## 增/减累加器(UPDN)

UPDN指令加两个输入值到累加值和从累加值减去两个输入值。

### 操作数:



```
UPDN(UPDN_tag);
```

### 结构文本

| 操作数： | 类型： | 格式： | 说明： |
|---|---|---|---|
| UPDN 标签 | UP_DOWN_ACCUM | 结构体 | UPDN结构体 |






### 功能块

| 操作数： | 类型： | 格式： | 说明： |
|---|---|---|---|
| UPDN 标签 | UP_DOWN_ACCUM | 结构体 | UPDN结构体 |

### UP_DOWN_ACCUM结构体

| 输入参数： | 数据类型： | 说明： |
|---|---|---|
| EnableIn | BOOL | **功能块**：<br>如果该位清零，则指令不执行且输出不更新。<br>如果置位，指令执行。<br>缺省状态为置位<br>**结构文本**：<br>无影响。指令一直执行。 |
| Initialize | BOOL | 指令的初始化请求。当Initialize被置位时，指令设值输出值Out和内部累加器值为InitialValue。<br>缺省状态为清零 |
| InitialValue | REAL | 指令的初始化值。<br>有效值=浮点数<br>缺省值=0.0 |
| InPlus | REAL | 加到累加值的值。<br>有效值=浮点数<br>缺省值=0.0 |
| InMinus | REAL | 从累加值减去的值。<br>有效值=浮点数<br>缺省值=0.0 |
| Hold | BOOL | 指令的保持请求。当Hold被置位且Initialize被清零时，输出值Out被保持。<br>缺省状态为清零 |
| **输出参数：** | **数据类型：** | **说明：** |
| EnableOut | BOOL | 输出使能。 |
| Out | REAL | 该指令的输出。输出结果将影响算术状态标志设置。 |

**说明：** UPDN指令使用下列算法：

| 条件： | 执行动作： |
|---|---|
| Hold清零且Initialize置位 | $AccumValue_n=AccumValue_{n-1}+InPlus-InMinus$<br>$Out=AccumValue_n$ |
| Hold置位且Initialize清零 | $AccumValue_n=AccumValue_{n-1}$<br>$Out=AccumValue_n$ |
| Initialize置位 | $AccumValue_n=InitialValue$<br>$Out=AccumValue_n$ |

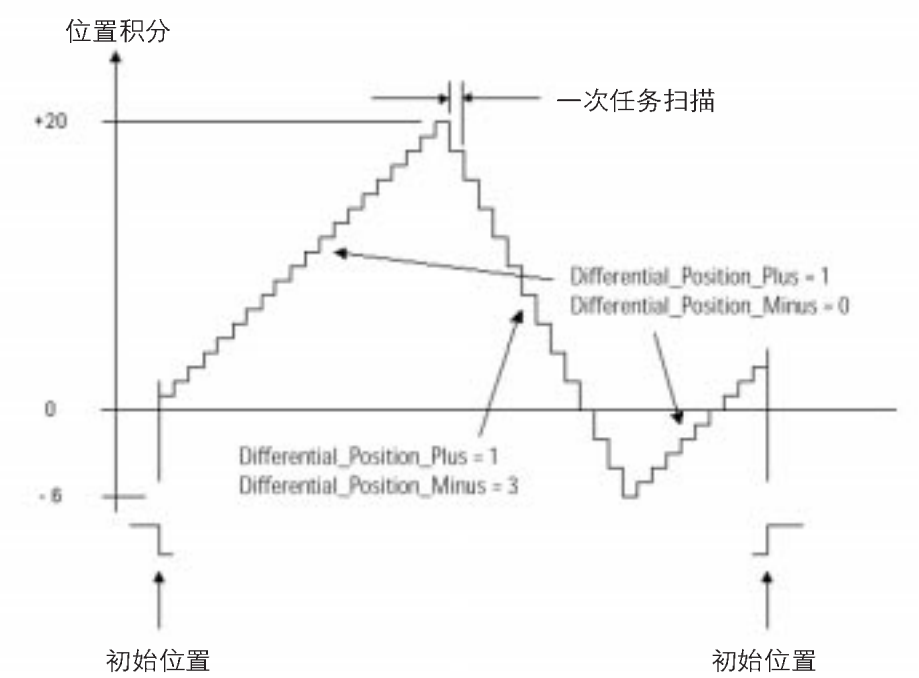
**算术状态标志：** 输出值将影响算术状态标志设置。

**故障条件：** 无

**执行：**

| 条件： | 功能块动作： | 结构文本动作： |
|---|---|---|
| 预扫描 | 无动作 | 无动作 |
| 指令首次扫描 | 无动作 | 无动作 |
| 指令首次执行 | AccumeValuen-1=0.0 | AccumeValuen-1=0.0 |
| 输入使能(EnableIn)清零 | 输出使能(EnableOut)清零。 | 不影响 |
| 输入使能(EnableIn)置位 | 执行指令。<br>输出使能(EnableOut)置位 | 使能输入(EnableIn)一直置位。<br>指令执行。 |
| 后期扫描 | 无动作 | 无动作 |

**实例：** UPDN指令积分连续两次扫描间产生的计数值。该指令可被用于简单位置应用或其它类型的应用，只要该应用需要一个简单的积分功能，根据过程偏差反馈信号计算一个累加值即可。在下面的例子中，Initial_Position设置为0，Differential_Position_Plus和Differential_Position_Minus在一段时间取不同的值。使用该指令时，InPlus和InMinus都能接收负值。

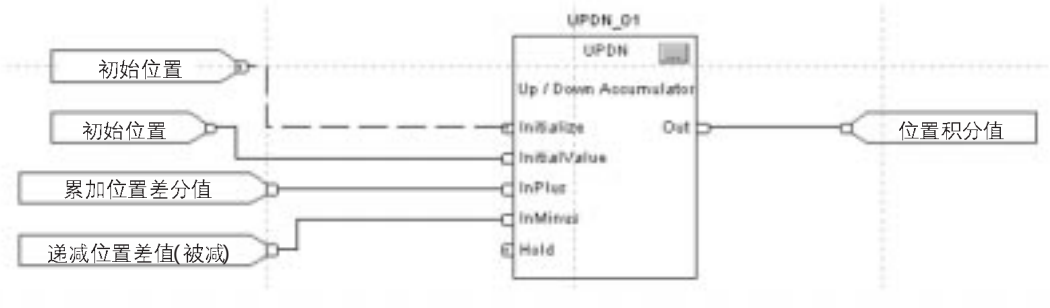


## 结构文本

```
UPDN_01.Initialize := Initialize_Position;
UPDN_01.InitialValue := Initial_Position;
UPDN_01.InPlus := Differential_Position_Plus;
UPDN_01.InMinus := Differential_Position_Minus;
UPDN(UPDN_01);

Position_Integrated := UPDN_01.Out;
```

## 功能块



注释：

# 滤波指令
# (DERV,HPF,LDL2,LPF,NTCH)

## 概述

以下滤波指令可用：

| 如果用户需要： | 使用指令： | 可用语言： | 参考页码： |
|---|---|---|---|
| 计算在单位时间内的信号变化量。 | 微分 (DERV) | 结构文本<br>功能块 | 3月2日 |
| 过滤输入信号中频率低于截止频率的部分。 | 高通滤波 (HPF) | 结构文本<br>功能块 | 3月6日 |
| 以极点对和零点对为参数滤波。 | 二阶超前-滞后 (LDL2) | 结构文本<br>功能块 | 3月12日 |
| 过滤输入信号中频率高于截止频率的部分。 | 低通滤波(LPF) | 结构文本<br>功能块 | 3月18日 |
| 过滤输入信号中频率等于凹波频率的部分。 | 凹波滤波(NTCH) | 结构文本<br>功能块 | 3-24 |

## 微分(DERV)

DERV指令可用于计算输入信号在单位时间内的变化量。

### 操作数：

#### 结构文本

| 操作数 | 类型 | 格式 | 说明 |
|---|---|---|---|
| DERV标签 | DERIVATIVE | 结构体 | DERV结构体 |

#### 功能块

| 操作数 | 类型 | 格式 | 说明 |
|---|---|---|---|
| DERV标签 | DERIVATIVE | 结构体 | DERV结构体 |

DERIVATIVE结构体

| 输入参数： | 数据类型： | 说明： |
|---|---|---|
| EnableIn | BOOL | 功能块：<br>如果该位清零，则指令不执行且输出不更新。<br>如果置位，指令执行。<br>缺省状态为置位<br>结构文本：<br>无影响。指令一直执行。 |
| In | REAL | 指令的模拟量输入信号。<br>有效值＝浮点数<br>缺省值＝0.0 |
| Gain | REAL | 微分系数<br>有效值＝浮点数<br>缺省值＝1.0 |
| ByPass | BOOL | 算法的旁通请求。置位时，指令设置Out = In。<br>缺省状态为清零。 |
| TimingMode | DINT | 选择定时执行方式：<br>**有效值：** **说明：**<br>0 周期执行方式<br>1 过采样方式<br>2 实时采样方式<br>需要了解更多与定时方式有关的信息，请参阅附录：功能块属性。<br>有效值=0到2<br>缺省值=0 |

| 输入参数： | 数据类型： | 说明： |
|---|---|---|
| OversampleDT | REAL | 过采样方式的执行时间。<br>有效值=0到4194.303 秒<br>缺省值=0 |
| RTSTime | DINT | 实时采样方式下模块的更新周期。<br>有效值=1到32,767ms<br>缺省值=1 |
| RTSTimeStamp | DINT | 实时采样方式下模块的时间戳值。<br>有效值=0到32,767ms<br>缺省值=0 |

| 输出参数： | 数据类型： | 说明： |
|---|---|---|
| EnableOut | BOOL | 输出使能。 |
| Out | REAL | 该指令算法计算出的输出结果。该输出影响算术状态字设置。 |
| DeltaT | REAL | 两次更新之间经过的时间，单位：秒。作为控制算法计算过程量输出值的时间间隔参数。 |
| Status | DINT | 功能块的状态字。 |
| InstructFault(Status.0) | BOOL | 指令检测以下的运行错误。该错误不使控制器产生次要或主要错误。检测其它状态位以确定发生错误的类型。 |
| TimingModeInv(Status.27) | BOOL | 定时方式值无效。<br>需要了解更多与定时方式有关的信息，请参阅附录：功能块属性。 |
| RTSMissed(Status.28) | BOOL | 只用于实时采样方式。当ABS\| DeltaT - RTSTime \| >1(0.001秒)时，该位被置位。 |
| RTSTimeInv(Status.29) | BOOL | RTSTime数值无效。 |
| RTSTimeStampInv(Status.30) | BOOL | RTSTimeStamp数值无效。 |
| DeltaTInv(Status.31) | BOOL | DeltaT数值无效。 |

**说明：** DERV指令支持旁通输入，允许停止输入信号的变化率而直接将输入信号作为输出信号。

| 当Bypass为： | 指令使用下列公式： |
|---|---|
| 置位 | $Out = In_n$<br>$In_{n-1}=In_n$ |
| 清零且DeltaT>0 | $Out = Gain\frac{In_n - In_{n-1}}{DeltaT}$<br>$In_{n-1} = In_n$<br>其中，DeltaT以秒为单位。 |

**算术状态字：**输出值将影响算术状态字设置。

**故障条件：**无。

**执行：**

| **条件：** | **功能块动作：** | **结构文本动作：** |
|---|---|---|
| 预扫描 | 无动作 | 无动作 |
| 指令首次扫描 | Inn-1=$In_n$ | Inn-1=$In_n$ |
| 指令首次执行 | Inn-1=$In_n$ | Inn-1=$In_n$ |
| 输入使能(EnableIn)清零 | 输出使能(EnableOut)清零。 | 不影响 |
| 输入使能(EnableIn)置位 | 执行指令。<br>输出使能(EnableOut)置位。 | 使能输入一直置位。<br>指令执行 |
| 后期扫描 | 无动作 | 无动作 |

**实例：** 微分指令计算输入信号在单位时间(秒)内的变化量。该指令通常用于闭环控制过程，为调节器中大惯性过程量提供前馈输入补偿。

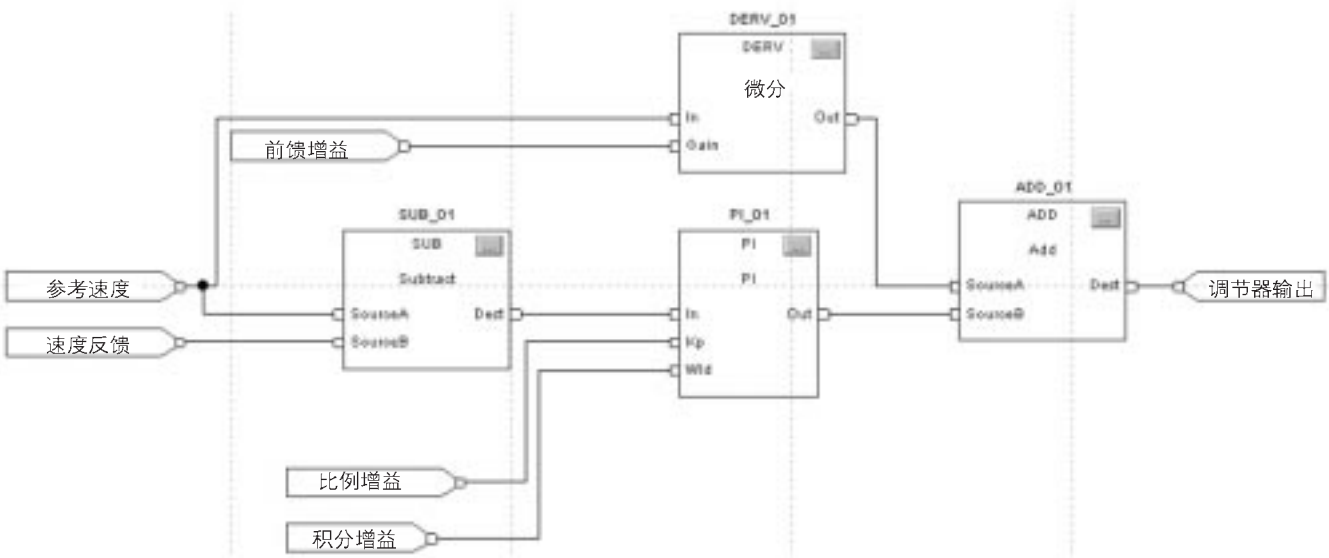
### 结构文本

```
DERV_01.In := Speed_Reference;
DERV_01.Gain := Feedforward_Gain;
DERV(DERV_01);

PI_01.In := Speed_Reference - Speed_feedback;
PI_01.Kp := Proportional_Gain;
PI_01.Wld := Integral_Gain;
PI(PI_01);

regulator_out := DERV_01.Out + PI_01.Out;
```

### 功能块



## 高通滤波器 (HPF)

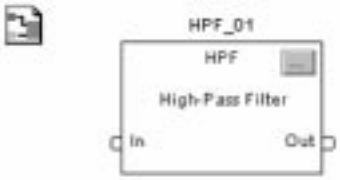
HPF指令提供削弱输入信号中频率低于截止频率部分的滤波。

### 操作数：




#### 结构文本

| 操作数 | 类型 | 格式 | 说明 |
|---|---|---|---|
| HPF标签 | FILTER_HIGH_PASS | 结构体 | HPF结构体 |




#### 功能块

| 操作数 | 类型 | 格式 | 说明 |
|---|---|---|---|
| HPF标签 | FILTER_HIGH_PASS | 结构体 | HPF结构体 |

### FILER_HIGH_PASS结构体

| 输入参数： | 数据类型： | 说明： |
|---|---|---|
| EnableIn | BOOL | 功能块：<br>如果该位清零，则指令不执行且输出不更新。如果置位，指令执行。<br>缺省状态为置位<br>结构文本：<br>无影响。指令一直执行。 |
| In | REAL | 指令的模拟量输入信号。<br>有效值＝浮点数<br>缺省值＝0.0 |
| Initialize | BOOL | 初始化滤波控制算法请求。当置位时，指令设置Out = In。<br>缺省状态为清零。 |
| WLead | REAL | 超前频率，单位：弧度/秒。如果WLead<最小值或WLead>最大值，指令置位状态字中的相应位，并且设置WLead为极限值。<br>有效值=参见下面有关有效范围的说明。<br>缺省值=0.0。 |
| Order | REAL | 滤波的阶数。阶数控制截断的锐度。如果阶数无效，则指令置位状态字中的相应位并设置阶数=1。<br>有效值=1到3<br>缺省值=1 |
| TimingMode | DINT | 选择定时执行方式。<br>**有效值：** **说明：**<br>0 周期执行方式<br>1 过采样方式<br>2 实时采样方式<br>需要了解更多与定时方式有关的信息，请参阅附录：功能块属性。<br>有效值=0到2<br>缺省值=0 |

| 输入参数： | 数据类型： | 说明： |
|---|---|---|
| OversampleDT | REAL | 过采样方式的执行时间。<br>有效值=0到4194.303秒<br>缺省值=0 |
| RTSTime | DINT | 实时采样方式下模块更新周期。<br>有效值=1到32,767ms<br>缺省值=1 |
| RTSTimeStamp | DINT | 实时采样方式下模块的时间戳的值。<br>有效值=0到32,767ms<br>缺省值=0 |

| 输出参数： | 数据类型： | 说明： |
|---|---|---|
| EnableOut | BOOL | 输出使能。 |
| Out | REAL | 该指令算法计算出的输出。输出结果将影响算术状态字设置。 |
| DeltaT | REAL | 两次更新之间经过的时间，单位：秒。作为控制算法计算过程量输出值的时间间隔参数。 |
| Status | DINT | 功能块的状态字。 |
| InstructFault(Status.0) | BOOL | 指令检测以下的运行错误。该错误不使控制器产生次要或主要错误。检测其它状态位以确定发生错误的类型。 |
| WLeadInv(Status.1) | BOOL | WLead<最小值 或 WLead>最大值。 |
| OrderInv(Status.2) | BOOL | 阶数值无效。 |
| TimingModeInv(Status.27) | BOOL | 定时方式值无效。<br>需要了解更多与定时方式有关的信息，请参阅附录：功能块属性。 |
| RTSMissed(Status.28) | BOOL | 只用于实时采样方式。当ABS(DeltaT-RTSTime)>1(0.001秒)时，该位被置位。 |
| RTSTimeInv(Status.29) | BOOL | RTSTime值无效。 |
| RTSTimeStampInv(Status.30) | BOOL | RTSTimeStamp值无效。 |
| DeltaTInv(Status.31) | BOOL | DeltaT值无效。 |

**说明：** HPF指令使用阶数(Order)参数来控制频率截止时的锐度。HPF指令是为扫描时间为常数的任务而设计。

HPF指令使用下列公式：

| 如果： | LPF指令使用下列传递函数： |
|---|---|
| Order=1 | $\frac{s}{s+\omega}$ |
| Order=2 | $\frac{s^2}{s^2+\sqrt{2}\times s\times\omega+\omega^2}$ |
| Order=3 | $\frac{s^3}{s^3+(2\times s^2\times\omega)+2\times s\times\omega^2+\omega^3}$ |

使用下列极限参数(其中DeltaT以秒为单位)

| 参数： | 极限值： |
|---|---|
| WLead 一阶下限 | 0.0000001/DeltaT |
| WLead 二阶下限 | 0.00005/DletaT |
| WLead 三阶下限 | 0.001/DeltaT |
| 上限 | 0.7 –/DeltaT |

无论计算的输出结果值无效、 NAN 或 ±INF，HPF指令设置Out等于无效的值且置位算术溢出位。当计算的输出结果变为有效时，指令初始化内部参数并且设置Out等于In。

**算术状态字：**输出值将影响算术状态字设置。

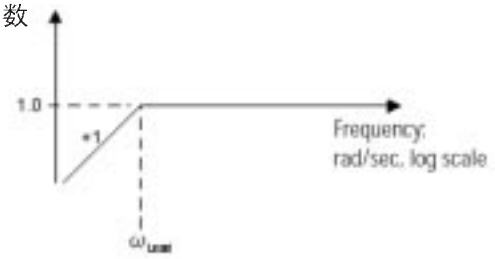
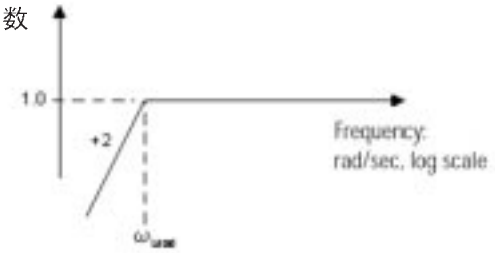
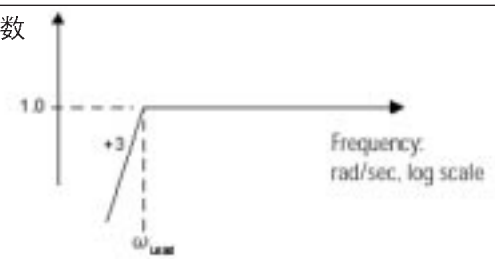
**故障条件：**无

**执行：**

| 条件： | 功能块动作： | 结构文本动作： |
|---|---|---|
| 预扫描 | 无动作。 | 无动作。 |
| 指令首次扫描 | 无动作。 | 无动作。 |
| 指令首次执行 | 指令设置Out = In。<br>控制算法不执行。 | 指令设置Out = In。<br>控制算法不执行。 |
| 输入使能(EnableIn)清零 | 输出使能(EnableOut)清零，指令不执行且输出不更新。 | 不影响 |
| 输入使能(EnableIn)置位 | 执行指令。<br>输出使能(EnableOut)置位 | 输入使能(EnableIn)一直置位。指令执行。 |
| 后期扫描 | 无动作 | 无动作 |

**实例：**HPF指令削弱信号中频率低于预设的截止频率部分。该指令的通常用于过滤电气或机械源产生的低频噪音或扰动。用户可以选择特定的阶数来完成不同程度的衰减。注意，阶数越高，滤波指令的执行时间越长。

下图说明了阶数不同对给定截止频率滤波后的效果。在图中，理想渐进线由增益系数和以对数为刻度的频率为纵横坐标。滤波后实际响应曲线逼近图示曲线，但与此曲线并不精确匹配。

| **滤波：** | **图形：** |
|---|---|
| 一阶滤波： |  |
| 二阶滤波： |  |
| 三阶滤波： |  |

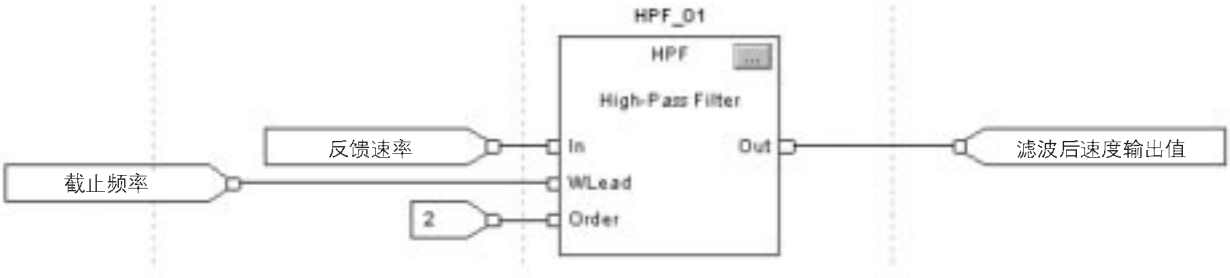
### 结构文本

```
HPF_01.In := Velocity_Feedback;
HPF_01.WLead := Cutoff_frequency;
HPF_01.Order := 2;

HPF(HPF_01);

filtered_velocity_output := HPF_01.Out
```

### 功能块



## 二阶超前/滞后滤波器 (LDL2)

LDL2指令提供以极点对和零点对作为过滤控制参数的滤波。极点对和零点对的频率和阻尼是可调的。极点对与零点对可以是复数(阻尼比小于1)或实数(阻尼比大于等于1)。

### 操作数：




#### 结构文本

| 操作数： | 类型： | 格式： | 说明： |
|---|---|---|---|
| LDL2标签 | LEAD_LAG_SEC_ORDER | 结构体 | LDL2结构体 |

#### 功能块

| 操作数： | 类型： | 格式： | 说明： |
|---|---|---|---|
| LDL2标签 | LEAD_LAG_SEC_ORDER | 结构体 | LDL2结构体 |

### LEAD_LAG_SEC_ORDER结构体

| 输入参数： | 数据类型： | 说明： |
|---|---|---|
| EnableIn | BOOL | 功能块：<br>如果该位清零，则指令不执行且输出不更新。<br>如果置位，指令执行。<br>缺省状态为置位<br>结构文本：<br>无影响。指令一直执行。 |
| In | REAL | 指令的模拟量输入信号。<br>有效值＝浮点数<br>缺省值＝0.0 |
| Initialize | BOOL | 初始化滤波控制算法请求。当置位时，指令设置Out = In。<br>缺省状态为清零 |
| WLead | REAL | 超前角频率，单位：弧度/秒。如果WLead<最小值或WLead>最大值，指令置位状态字中的相应位，并且设置WLead为极限值。如果Wlag:Wlead大于最大比率，指令置位状态字中的相应位，并且设置WLag为极限值。<br>有效值=参见下面关于有效范围的说明<br>缺省值=0.0 |
| WLag | REAL | 滞后角频率，单位：弧度/秒。如果WLag<最小值或WLag>最大值，指令置位状态字中的相应位，并且设置WLag为极限值。如果Wlag：Wlead大于最大比率，指令置位状态字中的相应位，并且设置WLag为极限值。<br>有效值=参见下面关于有效范围的说明<br>缺省值=0.0 |
| ZetaLead | REAL | 二阶超前阻尼因子。仅当Order=2时使用。如果ZetaLead<最小值或ZetaLead>最大值，指令置位状态字中的相应位，并且设置ZetaLead为极限值。<br>有效值=0.0 到4.0<br>缺省值=0.0 |

| 输入参数： | 数据类型： | 说明： |
|---|---|---|
| ZetaLag | REAL | 二阶滞后阻尼因子。仅当Order=2时使用。如果ZetaLag<最小值或ZetaLag>最大值，指令置位状态字中的相应位，并且设置ZetaLag为极限值。<br>有效值=0.05 到 4.0<br>缺省值=0.0 |
| Order | REAL | 滤波器的阶数。选择一阶或二阶滤波算法。如果阶数无效，则指令置位状态字中的相应位并使阶数为2。<br>有效值=1到2<br>缺省值=2 |
| TimingMode | DINT | 选择定时执行方式。<br>**有效值:说明：**<br>0 周期执行方式<br>1 过采样方式<br>2 实时采样方式<br>需要了解更多与定时方式有关的信息，请参阅附录：功能块属性。<br>有效值=0到2<br>缺省值=0 |
| OversampleDT | REAL | 过采样方式的执行时间。<br>有效值=0到4194.303秒<br>缺省值=0 |
| RTSTime | DINT | 实时采样方式下模块的更新周期。<br>有效值=1到32,767ms<br>缺省值=1 |
| RTSTimeStamp | DINT | 实时采样方式下模块的时间戳值。<br>有效值=0到32,767ms<br>缺省值=0 |

| 输出参数： | 数据类型： | 说明： |
|---|---|---|
| EnableOut | BOOL | 输出使能。 |
| Out | REAL | 该指令算法计算出的输出结果。输出结果将影响算术状态字设置。 |
| DeltaT | REAL | 两次更新之间经过的时间，单位：秒。作为控制算法计算过程量输出值的时间间隔参数。 |
| Status | DINT | 功能块的状态字。 |
| InstructFault(Status.0) | BOOL | 指令检测以下的运行错误。该错误不使控制器产生次要或主要错误。检测其它状态位以确定发生错误的类型。 |
| WLeadInv(Status.1) | BOOL | WLead<最小值 或 WLead>最大值 时置位。 |
| WLagInv(Status.2) | BOOL | WLag<最小值 或 WLag>最大值 时置位。 |
| ZetaLeadInv(Status.3) | BOOL | 当超前阻尼因子<最小值 或 超前阻尼因子>最大值时置位。 |
| ZetaLagInv(Status.4) | BOOL | 当滞后阻尼因子<最小值 或 滞后阻尼因子>最大值时置位。 |
| OrderInv(Status.5) | BOOL | 阶数值无效。 |
| WLagRatioInv | BOOL | Wlag:WLead>最大值. |
| TimingModeInv(Status.27) | BOOL | 定时方式值无效。<br>需要了解更多与定时方式有关的信息，请参阅附录：功能块属性。 |
| RTSMissed(Status.28) | BOOL | 只用于实时采样方式。当ABS(DeltaT-RTSTime)>1(0.001秒)时，该位被置位。 |

| 输出参数： | 数据类型： | 说明： |
|---|---|---|
| RTSTimeInv(Status.29) | BOOL | RTSTime值无效。 |
| RTSTimeStampInv(Status.30) | BOOL | RTSTimeStamp值无效。 |
| DeltaTInv(Status.31) | BOOL | DeltaT值无效。 |

**说明：** LDL2滤波指令用于强制参考输入控制和强制反馈控制方式。LDL2指令是为扫描时间为常数的任务而设计。

LDL2指令使用下列公式：

如果：

一阶

LPF指令使用下列拉普拉斯变换传递函数：

$$H(s) = \frac{\frac{s}{\omega_{Lead}} + 1}{\frac{s}{\omega_{Lag}} + 1}$$

二阶

$$H(s) = \frac{\frac{s^2}{\omega_{Lead}^2} + \frac{2 \times \xi_{Lead} \times s}{\omega_{Lead}} + 1}{\frac{s^2}{\omega_{Lag}^2} + \frac{2 \times \xi_{Lag} \times s}{\omega_{Lag}} + 1}$$

标准化滤波使得$\omega_{Lead}$=1

$$H(s) = \frac{s^2 + 2 \times \xi_{Lead} \times s + 1}{\frac{s^2}{\omega_{Lag}^2} + \frac{2 \times \xi_{Lag} \times s}{\omega_{Lag}} + 1}$$

使用以下极限参数(其中DeltaT以秒为单位)：

| 参数： | 限制值： |
|---|---|
| WLead 一阶下限 | 0.0000001/DeltaT |
| WLead 二阶下限 | 0.00005/DletaT |
| 上限 | 0.7π/DeltaT |

| 参数： | 极限值： |
|---|---|
| Wlead:Wlag比率 | 如果WLead>WLag 则没有限制 |
| | 如果WLag>WLead ：<br>• 对WLag:WLead没有最小限制<br>• 在一阶情况下，WLag:WLead最大比=40:1 超过则指令限制WLag的值使得比率为40:1。<br>• 在二阶情况下，WLag:WLead最大比=10:1 超过则指令限制WLag的值使得比率为10:1。 |
| ZetaLead (只在二阶情况下有效) | LowLimit=0.0<br>HighLimit=4.0 |
| ZetaLag (只在二阶情况下有效) | LowLimit=0.05<br>HighLimit=4.0 |

无论计算的输出结果值无效， NAN 或 ±INF，LDL2指令设置Out 等于无效值并且置位算术溢出位。当计算的输出结果变为有效时，指令初始化内部参数并且设置Out 等于 In。

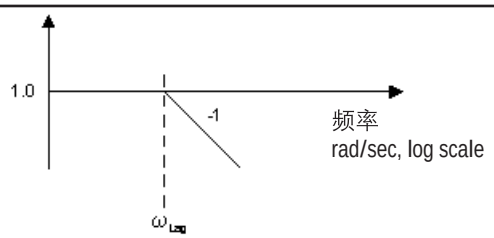
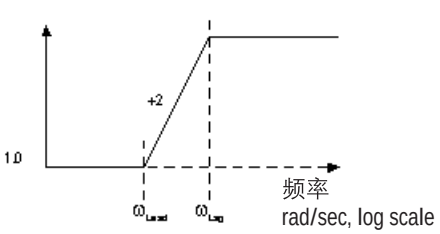
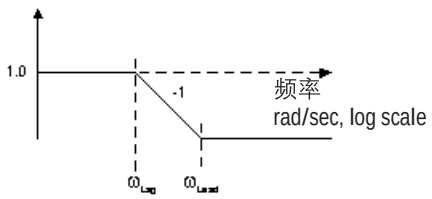
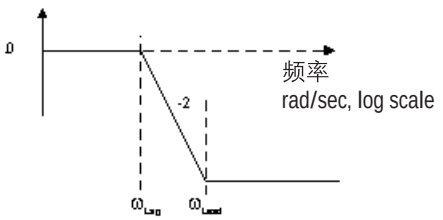
**算术状态字：**输出值将影响算术状态字设置。

**故障条件：**无

**执行：**

| 条件： | 功能块动作： | |
|---|---|---|
| 预扫描 | 无动作。 | 无动作。 |
| 指令首次扫描 | 无动作。 | 无动作。 |
| 指令首次执行 | 指令设置Out = In。<br>控制算法不执行。 | 指令设置Out = In。<br>控制算法不执行。 |
| 输入使能(EnableIn)清零 | 输出使能(EnableOut)清零，指令不执行且输出不更新。 | 不影响 |
| 输入使能(EnableIn)置位 | 执行指令。<br>输出使能(EnableOut)置位 | 输入使能(EnableIn)一直置位。<br>指令执行。 |
| 后期扫描 | 无动作 | 无动作 |

**实例：** LDL2指令能根据对指令的组态不同来削弱或增强信号中频率处于两个预设频率之间部分。由于超前或滞后频率值的大小可以任意设定，根据设定的频率次序可以分为超前-滞后块或滞后-超前块。注意随着阶数越高，滤波指令的执行时间越长。

| **滤波：** | **图形：** |
|---|---|
| 一阶超前-滞后<br>($\omega_{Lead}<\omega_{Lag}$)： | 增益系数<br> |
| 二阶超前-滞后<br>($\omega_{Lead}<\omega_{Lag}$)： | 增益系数<br> |
| 一阶超前-滞后<br>($\omega_{Lag}<\omega_{Lag}$)： | 增益系数<br> |
| 二阶超前-滞后<br>($\omega_{Lag}<\omega_{Lead}$)： | 增益系数<br> |

## 结构文本

```
LDL2_01.In := frequency_input;
LDL2_01.WLead := Lead_frequency;
LDL2_01.WLag := Lag_frequency;

LDL2(LDL2_01);

Lead_lag_output := LDL2_01.Out;
```

## 功能块



## 低通滤波器 (LPF)

LPF指令提供用于削弱输入信号中频率高于截止频率部分的滤波。

### 操作数：

#### 结构文本

| 操作数： | 类型： | 格式： | 说明： |
|---|---|---|---|
| LPF 标签 | FILTER_LOW_PASS | 结构体 | LPF 结构体 |

#### 功能块

| 操作数： | 类型： | 格式： | 说明： |
|---|---|---|---|
| LPF 标签 | FILTER_LOW_PASS | 结构体 | LPF 结构体 |

### FILTER_LOW_PASS 结构体

| 输入参数： | 数据类型： | 说明： |
|---|---|---|
| EnableIn | BOOL | 功能块：<br>如果该位清零，则指令不执行且输出不更新。<br>如果置位，指令执行。<br>缺省状态为置位<br>结构文本：<br>无影响。指令一直执行。 |
| In | REAL | 指令的模拟量输入信号。<br>有效值＝浮点数<br>缺省值＝0.0 |
| Initialize | BOOL | 初始化滤波控制算法的请求。当置位时，指令设置Out = In。<br>缺省状态为清零。 |
| WLag | REAL | 超前频率，单位：弧度/秒。如果WLag<最小值或WLag>最大值，指令置位状态字中相应的位并设置WLag为极限值。<br>有效值=参见下面有关有效范围的说明<br>缺省值=最大正浮点数 |
| Order | REAL | 滤波器的阶数。阶数控制截断频率的锐度。如果阶数无效，则指令置位状态字中相应的位并设置阶数=1。<br>有效值=1到3<br>缺省值=1 |
| TimingMode | DINT | 选择定时执行方式。<br>**有效值：** **说明：**<br>0 周期执行方式<br>1 过采样方式<br>2 实时采样方式<br>需要了解更多与定时方式有关的信息，请参阅附录：功能块属性。<br>有效值=0到2<br>缺省值=0 |

| 输入参数： | 数据类型： | 说明： |
|---|---|---|
| OversampleDT | REAL | 过采样方式执行时间。<br>有效值=0到4194.303秒<br>缺省值=0 |
| RTSTime | DINT | 实时采样方式下模块的更新周期。<br>有效值=1到32,767ms<br>缺省值=1 |
| RTSTimeStamp | DINT | 实时采样方式下模块的时间戳值。<br>有效值=0到32,767ms<br>缺省值=0 |

| 输出参数： | 数据类型： | 说明： |
|---|---|---|
| EnableOut | BOOL | 输出使能。 |
| Out | REAL | 该指令算法计算出的输出结果。输出结果将影响算术状态字设置。 |
| DeltaT | REAL | 两次更新之间经过的时间，单位：秒。作为控制算法计算过程量输出值的时间间隔参数。 |
| Status | DINT | 功能块的状态字。 |
| InstructFault(Status.0) | BOOL | 指令检测以下的运行错误。该错误不使控制器产生次要或主要错误。检测其它状态位以确定发生错误的类型。 |
| WLagInv(Status.1) | BOOL | WLag<最小值 或 WLag>最大值。 |
| OrderInv(Status.2) | BOOL | 阶数值无效。 |
| TimingModeInv(Status.27) | BOOL | 定时方式值无效。<br>需要了解更多与定时方式有关的信息，请参阅附录：功能块属性。 |
| RTSMissed(Status.28) | BOOL | 只用于实时采样方式。当ABS(DeltaT-RTSTime)>1(0.001秒)时，该位被置位。 |
| RTSTimeInv(Status.29) | BOOL | RTSTime值无效。 |
| RTSTimeStampInv(Status.30) | BOOL | RTSTimeStamp值无效。 |
| DeltaTInv(Status.31) | BOOL | DeltaT值无效。 |

**说明：** LPF指令使用阶数来控制截止频率的锐度。LPF指令设计在扫描速率保持恒定的任务中运行。

LPF指令使用下列公式：

| 如果： | LPF指令使用下列传递函数： |
|---|---|
| 一阶 | $\frac{\omega}{s+\omega}$ |
| 二阶 | $\frac{\omega^2}{s^2+\sqrt{2}\times s\times\omega+\omega^2}$ |
| 三阶 | $\frac{\omega^3}{s^3+(2\times s^2\times\omega)+(2\times s\times\omega^2)\times\omega^3}$ |

使用下列极限参数(其中DeltaT以秒为单位)：

| 参数： | 极限值： |
|---|---|
| WLag一阶下限 | 0.0000001/DeltaT |
| WLag二阶下限 | 0.00005/DletaT |
| WlLag 三阶下限 | 0.001/DeltaT |
| 上限 | 0.7π/DeltaT |

无论计算的输出结果值无效、NAN 或 ±INF，LPF指令设置Out 等于无效的值且置位算术溢出位。当计算的输出结果变为有效时，指令初始化内部参数并且设置Out = In。

**算术状态字：**输出值将影响算术状态字设置。

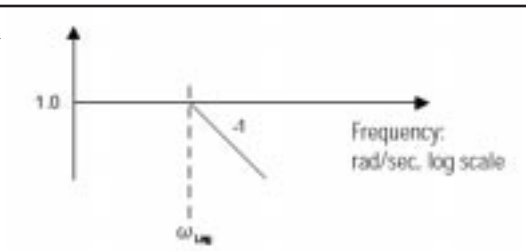
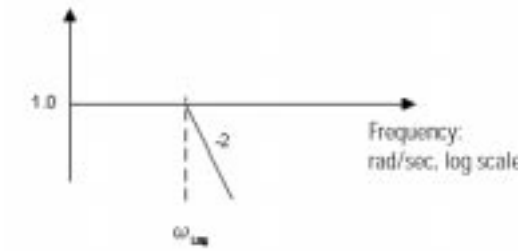
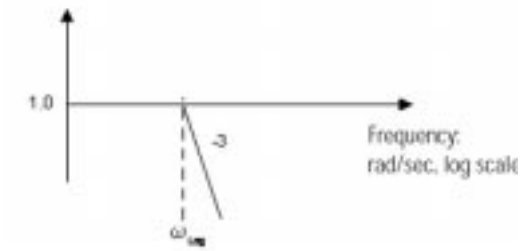
**故障条件：**无

**执行：**

| 条件： | 功能块动作： | 结构文本动作： |
|---|---|---|
| 预扫描 | 无动作。 | 无动作。 |
| 指令首次扫描 | 无动作。 | 无动作。 |
| 指令首次执行 | 指令设置Out = In。<br>控制算法不执行。 | 指令设置Out = In。<br>控制算法不执行。 |
| 输入使能(EnableIn)清零 | 输出使能(EnableOut)清零。 | 不影响 |
| 输入使能(EnableIn)置位 | 执行指令。<br>输出使能(EnableOut)置位 | 输入使能(EnableIn)一直置位。<br>指令执行。 |
| 后期扫描 | 无动作 | 无动作 |

**实例：**LPF指令削弱信号中频率高于预设截止频率的部分。该指令的通常用于滤波电气或机械源产生的高频噪声或扰动。用户可以选择不同的阶数来完成不同程度的滤波。注意随着阶数越高，执行时间越长。

下图说明了阶数不同对给定截止频率滤波后的效果。在图中，理想渐进线由增益系数和以对数为刻度的频率为纵横坐标。滤波后实际响应曲线逼近图示曲线，但并不精确匹配。

| **滤波：** | **图形：** |
| --- | --- |
| 一阶滤波： | 增益系数<br> |
| 二阶滤波： | 增益系数<br> |
| 三阶滤波： | 增益系数<br> |

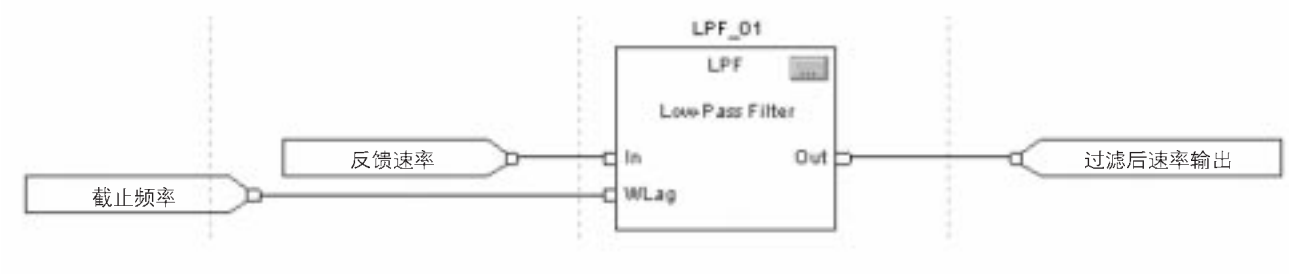
## 结构文本

```
LPF_01.In := Velocity_Feedback;
LPF_01.WLag := Cutoff_frequency;

LPF(LPF_01);

filtered_velocity_output := LPF_01.Out
```

## 功能块



## 凹波滤波器 (NTCH)

NTCH指令提供滤波功能用于削弱输入信号频率在凹波频率附近的部分。

### 操作数：




#### 结构文本

| 操作数 | 类型 | 格式 | 说明 |
|---|---|---|---|
| NTCH标签 | FILTER_NOTCH | 结构体 | NTCH结构体 |




#### 功能块

| 操作数 | 类型 | 格式 | 说明 |
|---|---|---|---|
| NTCH标签 | FILTER_NOTCH | 结构体 | NTCH结构体 |

### FILTER_NTCH结构体

| 输入参数： | 数据类型： | 说明： |
|---|---|---|
| EnableIn | BOOL | 功能块：<br>如果该位清零，则指令不执行且输出不更新。<br>如果置位，指令执行。<br>缺省状态为置位<br>结构文本：<br>无影响。指令一直执行。 |
| In | REAL | 指令的模拟量输入信号。<br>有效值＝浮点数<br>缺省值＝0.0 |
| Initialize | BOOL | 初始化滤波控制算法的请求。当置位时，指令设置Out = In。<br>缺省状态为清零. |
| WNotch | REAL | 滤波中心频率，单位弧度/秒。如果WNotch<最小值或WNotch>最大值，指令置位状态字中相应的位，并且设置WNotch为极限值。<br>有效值=参见下面有关有效范围的说明<br>缺省值=最大正浮点数 |
| QFactor | REAL | 控制凹波宽度和深度的比率。设置QFactor=1/(2 × 期望的阻尼因子)。如果QFactor<最小值或QFactor>最大值，指令置位状态字中相应的位，并且设置QFactor为极限值。<br>有效值＝0.5到100.0<br>缺省值＝0.5 |
| Order | REAL | 滤波的阶数。阶数控制截止频率的锐度。如果阶数无效，则指令置位状态字中相应的位并设置阶数等于二阶。<br>有效值=2或4<br>缺省值=2 |

| 输入参数： | 数据类型： | 说明： |
|---|---|---|
| TimingMode | DINT | 选择定时执行方式。<br>**有效值：　说明：**<br>0　周期执行方式<br>1　过采样方式<br>2　实时采样方式<br>需要了解更多与定时方式有关的信息，请参阅附录：功能块属性。<br>有效值=0到2<br>缺省值=0 |
| OversampleDT | REAL | 过采样方式执行时间。<br>有效值=0到4194.303秒<br>缺省值=0 |
| RTSTime | DINT | 在实时采样方式下模块的更新周期。<br>有效值=1到32,767ms<br>缺省值=1 |
| RTSTimeStamp | DINT | 在实时采样方式下模块的时标。<br>有效值=0到32,767ms<br>缺省值=0 |

| 输出参数： | 数据类型： | 说明： |
|---|---|---|
| EnableOut | BOOL | 输出使能。 |
| Out | REAL | 该指令算法计算出的输出结果。输出结果将影响算术状态字设置。 |
| DeltaT | REAL | 两次更新之间经过的时间，单位：秒。作为控制算法计算过程量输出值的时间间隔参数。 |
| Status | DINT | 功能块的状态字。 |
| InstructFault(Status.0) | BOOL | 指令检测以下的运行错误。该错误不使控制器产生次要或主要错误。检测其它状态位以确定发生错误的类型。。 |
| WNotchInv(Status.1) | BOOL | WNotch<最小值 或 WNotch>最大值。 |
| QFactorInv(Status.2_ | BOOL | QFactor<最小值 或 QFactor>最大值。 |
| OrderInv(Status.2) | BOOL | 阶数值无效。 |
| TimingModeInv(Status.27) | BOOL | 定时方式值无效。<br>需要了解更多与定时方式有关的信息，请参阅附录：功能块属性。 |
| RTSMissed(Status.28) | BOOL | 只用于实时采样方式。当ABS(DeltaT-RTSTime) >1(0.001秒)时，该位被置位。 |
| RTSTimeInv(Status.29) | BOOL | RTSTime值无效。 |
| RTSTimeStampInv(Status.30) | BOOL | RTSTimeStamp值无效。 |
| DeltaTInv(Status.31) | BOOL | DeltaT值无效。 |

**说明：** NTCH指令通过阶数参数来控制截止频率的锐度。QFactor参数控制凹波宽度和深度的比率。NTCH指令是为扫描时间为常数的任务而设计的。

NTCH指令使用下面公式：

$$\frac{(s^2+\omega^2)^i}{\left(s^2+s\times\frac{\omega}{Q}+\omega^2\right)^i}$$

其中i为阶数，由以下参数限制(其中DeltaT以秒为单位)：

| 参数： | 限值： |
|---|---|
| WNotch 二阶下限 | 0.0000001/DeltaT |
| WNotch 四阶下限 | 0.001/DletaT |
| 上限 | 0.7 –/DeltaT |
| QFactor | LowLimit=0.5<br>HighLimit=100.0 |

无论计算的输出结果值无效， NAN 或 ±INF，LPF指令设置Out等于无效值。并且置位算术溢出位。当计算的输出结果变为有效时，指令初始化内部参数并且设置输出Out等于输入 In。

**算术状态字：** 输出值将影响算术状态字设置。

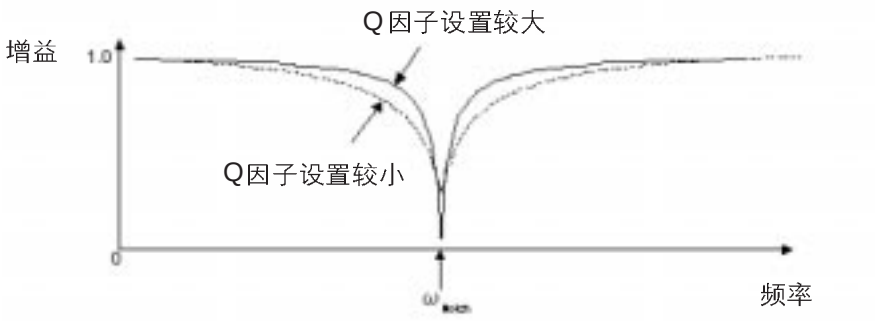
**故障条件：** 无

**执行：**

| 条件： | 功能块动作： | 结构文本动作： |
|---|---|---|
| 预扫描 | 无动作。 | 无动作。 |
| 指令首次扫描 | 无动作。 | 无动作。 |
| 指令首次执行 | 指令设置Out = In。<br>控制算法不执行。 | 指令设置Out = In。<br>控制算法不执行。 |
| 输入使能(EnableIn)清零 | 输出使能(EnableOut)清零。 | 不影响 |
| 输入使能(EnableIn)置位 | 指令执行。<br>输出使能(EnableOut)置位 | 输入使能(EnaleIn)一直置位。<br>指令执行。 |
| 后期扫描 | 无动作。 | 无动作。 |

**实例：**NTCH指令用于削弱信号中特定的谐振频率部分。通常，这些谐振频率处于闭环控制系统所调节的响应范围内。谐振频率信号大都是由松散的机械联动装置在系统中形成的冲击和震荡产生的。尽管最优的解决方案是校正机器的机械顺从性，凹波滤波指令也可用在闭环调节方案中减小这些信号产生的影响。

下图绘出了特定的凹波中心频率和Q因子时在某一频率范围内的理想增益曲线。随着Q因子增大，凹波更宽且更浅；随着Q因子减小，凹波更窄且更宽。该指令可以被设置成为二阶或四阶。阶数越大，执行时间越长。

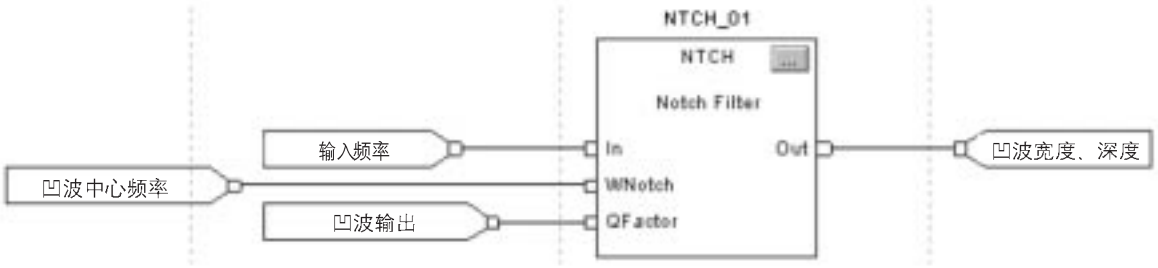


## 结构文本

```
NTCH_01.In := frequency_input;
NTCH_01.WNotch := center_frequency;
NTCH_01.QFactor := Notch_width_depth;
NTCH(NTCH_01);

Notch_output := NTCH_01.Out;
```

## 功能块



# 选择/限制指令
# (ESEL, HLL, MUX, RLIM, SEL, SNEG, SSUM)

## 概述

选择/限制指令可用于：

| 如果用户需要： | 使用指令： | 可用语言： | 参考页码： |
|---|---|---|---|
| 从最多6路输入中选择1路。 | 增强型选择(ESEL) | 结构文本<br>功能块 | 4月2日 |
| 限制模拟量输入值在两个预设值之间。 | 上/下限(HLL) | 结构文本<br>功能块 | 4月9日 |
| 从8路输入中选择1路。 | 多路切换器(MUX) | 功能块 | 4月12日 |
| 限制信号在一段时间内变化量。 | 速率极限(RLIM) | 结构文本<br>功能块 | 4月15日 |
| 从2路输入中选择1路。 | 选择(SEL) | 功能块 | 4月19日 |
| 选择输入值或输入值相反值。 | 选择取反 (SNEG) | 结构文本<br>功能块 | 4月21日 |
| 选择实际输入并累加。 | 选择求和 (SSUM) | 结构文本<br>功能块 | 4月23日 |

## 增强型选择(ESEL)

ESEL指令允许用户从最多6路输入中选择1路。
选择方式包括：

- 手动选择(由操作员或程序)
- 选择最大值
- 选择最小值
- 选择中间值
- 选择平均值

### 操作数：



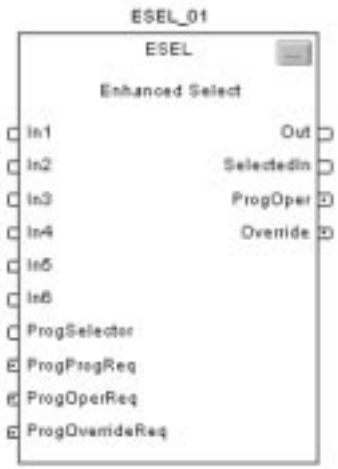
```
ESEL(ESEL_tag);
```

### 结构文本

| 操作数： | 类型： | 格式： | 说明： |
|---|---|---|---|
| ESEL标签 | SELECT_ENHANCED | 结构体 | ESEL结构体 |



### 功能块




| 操作数： | 类型： | 格式： | 说明： |
|---|---|---|---|
| ESEL标签 | SELECT_ENHANCED | 结构体 | ESEL结构体 |

### SELECT_ENHANCED结构体

| 输入参数： | 数据类型： | 说明： |
|---|---|---|
| EnableIn | BOOL | 功能块：<br>如果该位清零，则指令不执行输出不更新。<br>如果置位，指令执行。<br>缺省状态为置位<br>结构文本：<br>无影响。指令一直执行。 |
| In1 | REAL | 指令的模拟量输入信号1。<br>有效值＝浮点数<br>缺省值＝0.0 |
| In2 | REAL | 指令的模拟量输入信号2。<br>有效值＝浮点数<br>缺省值＝0.0 |
| In3 | REAL | 指令的模拟量输入信号3。<br>有效值＝浮点数<br>缺省值＝0.0 |

| 输入参数： | 数据类型： | 说明： |
|---|---|---|
| In4 | REAL | 指令的模拟量输入信号4。<br>有效值＝浮点数<br>缺省值＝0.0 |
| In5 | REAL | 指令的模拟量输入信号5。<br>有效值＝浮点数<br>缺省值＝0.0 |
| In6 | REAL | 指令的模拟量输入信号6。<br>有效值＝浮点数<br>缺省值＝0.0 |
| In1Fault | BOOL | In1输入状态错误指示。通常如果In1读自模拟量输入，则In1Fault由该模拟量输入的错误状态位控制。如果所有的InnFault参数被置位，指令置位状态字中的相应位，控制算法不执行，且输出结果不更新。<br>缺省状态为清零。 |
| In2Fault | BOOL | In2输入状态错误指示。通常如果In2读自模拟量输入，则In2Fault由该模拟量输入的错误状态位控制。如果所有的InnFault参数被置位，指令置位状态字中的相应位，控制算法不执行，且输出结果不更新。<br>缺省状态为清零。 |
| In3Fault | BOOL | In3输入状态错误指示。通常如果In3是读自模拟量输入，则In3Fault由该模拟量输入的错误状态位控制。如果所有的InnFault参数被置位，指令置位状态字中的相应位，控制算法不执行，且输出结果不更新。<br>缺省状态为清零。 |
| In4Fault | BOOL | In4输入状态错误指示。通常如果In4读自模拟量输入，则In4Fault由该模拟量输入的错误状态位控制。如果所有的InnFault参数被置位，指令置位状态字中的相应位，控制算法不执行，且输出结果不更新。<br>缺省状态为清零。 |
| In5Fault | BOOL | In5输入状态错误指示。通常如果In5读自模拟量输入，则In5Fault由该模拟量输入的错误状态位控制。如果所有的InnFault参数被置位，指令置位状态字中的相应位，控制算法不执行，且输出结果不更新。<br>缺省状态为清零。 |
| In6Fault | BOOL | In6输入状态错误指示。通常如果In6是读自模拟量输入，则In6Fault由该模拟量输入的错误状态位控制。如果所有的InnFault参数被置位，指令置位状态字中的相应位，控制算法不执行，且输出结果不更新。<br>缺省状态为清零。 |
| InsUsed | DINT | 可使用的输入通道数。该参数确定指令可使用的输入通道数。指令在选择最大值、选择最小值、选择中间值和选择平均值方式下只使用In1到InInsUsed的输入通道。如果该值无效，指令置位状态字中的相应位。如果该值无效，指令在非手动选择输入方式下，且Override位被请零，不更新输出结果。<br>有效值=1到6<br>缺省值=1 |

| 输入参数： | 数据类型： | 说明： |
|---|---|---|
| SelectorMode | DINT | 选择器方式输入。该值决定指令的执行方式。<br>**值：** **说明：**<br>0 手动选择<br>1 选择最大值<br>2 选择最小值<br>3 选择中间值<br>4 选择平均值<br>如果该值无效，指令置位状态字中的相应位且不更新输出结果。<br>有效值=0到4<br>缺省值=0 |
| ProgSelector | DINT | 程序选择器输入。当选择器选择方式为手动选择且指令处于程序控制方式时，ProgSelector决定被送往输出Out的输入通道(In1-In6)。如果ProgSelector=0，指令不更新输出结果。如果ProgSelector无效，指令置位状态字中的相应位。如果该值无效，指令处于程序控制方式，且选择器方式是手动选择或Override位被置位，指令不更新输出结果。<br>有效值=0到6<br>缺省值=0 |
| OperSelector | DINT | 操作员选择器输入。当选择器方式为手动选择且指令为操作员控制时，OperSelector决定被送往输出Out的输入通道(In1-In6)。如果OperSelector=0，指令不更新输出。如果OperSelector无效，指令置位状态字中的相应位。如果该值无效，指令处于操作员控制方式且选择手动选择或Override位被置位，指令不更新输出结果。<br>有效值=0到6<br>缺省值=0 |
| ProgProgReq | BOOL | 程序请求进入程序控制方式。用户程序通过置位该位请求程序控制。如果ProgOperReq被置位，该位被忽略。保持该位置位且ProgOperReq清零，指令被锁定到程序控制方式。<br>缺省状态为清零 |
| ProgOperReq | BOOL | 程序请求进入操作员控制方式。用户程序通过设置该位请求操作员控制方式。保持该位置位，指令被锁定到操作员控制方式。<br>缺省状态为清零 |
| ProgOverrideReq | BOOL | 程序请求进入自动方式。用户程序通过设置该位使设备进入自动方式。如果ProgOper清零，该请求被忽略。<br>缺省状态为清零 |
| OperProgReq | BOOL | 操作员请求进入程序控制方式。通过操作员界面置位该位请求进入程序控制方式。指令自动清零该输入。<br>缺省状态为清零。 |
| OperOperReq | BOOL | 操作员进入操作员控制方式请求。通过操作员界面置位该位请求进入操作员控制方式。指令自动清零该输入。<br>缺省状态为清零。 |
| ProgValueReset | BOOL | 复位程序控制值。如果置位，每次指令执行都将复位全部程序请求的输入参数。<br>缺省状态为清零 |

| 输入参数： | 数据类型： | 说明： |
|---|---|---|
| EnableOut | BOOL | 输出使能。 |
| Out | REAL | 该指令算法计算出的输出结果。输出结果将影响算术状态字设置。 |
| SelectedIn | DINT | 选择的输入通道号。指令通过该值指出当前被送往输出的输入通道号。如果选择方式为选择平均值，指令设置SelectedIn=0。 |
| ProgOper | BOOL | 程序/操作员控制方式指示。当处于程序控制方式时置位；当处于操作员控制方式时清零。 |
| Override | BOOL | 自动方式。当指令处于自动方式时该位置位。 |
| Status | DINT | 功能块的状态字。 |
| InstructFault(Status.0) | BOOL | 指令检测以下运行错误。该错误不使控制器产生次要或主要错误。检测其它状态位以确定产生错误的类型。 |
| InsFaulted(Status.1) | BOOL | 所有被使用输入通道n(Inn)的InnFault参数均置位时该位置位。 |
| InsUsedInv(Status.2) | BOOL | InsUsed值无效。 |
| SelectorModeInv(Status.3) | BOOL | SelectorMode值无效。 |
| ProgSelectorInv(Status.4) | BOOL | ProgSelector值无效。 |
| OperSelectorInv(Status.5) | BOOL | OperSelector值无效。 |

**说明：**ESEL指令操作方式如下：

| 条件： | 执行动作： |
|---|---|
| SelectorMode=0(手动选择)或Override置位，<br>ProgOper被清零且OperSelector≠0 | Out= In[OperSelector]<br>SelectedIn=OperSelector |
| SelectorMode=0(手动选择)或Override置位，<br>ProgOper被置位且ProgSelector ≠0 | Out=In[ProgSelector]<br>SelectedIn=ProgSelector |
| SelectorMode=1(选择最大值)且Override被清零 | Out=In[InsUsed]中最大值<br>SelectedIn为最大输入值所在的通道号 |
| SelectorMode=2(选择最小值)且Override被清零 | Out= In[InsUsed]中最小值<br>SelectedIn为最小输入值所在的通道号 |
| SelectorMode=3(选择中间值)且Override被清零 | Out=In[InsUsed]中间值<br>SelectedIn为中间输入值所在的通道号 |
| SelectorMode=4(选择平均值)且Override被清零 | Out= In[InsUsed]平均值。<br>SelectedIn=0 |

对于选择方式(SelectorMode)1至4，在选择过程中输入错误引起的出错指示将被忽略。例如：当SelectorMode=1(选择最大值)且通道6的输入值最大但状态出错，然后下一具有正确输入状态的最大输入值将被送往输出。

对于选择方式为选择最大值或选择最小值，当两个输入值相等且为最大或最小值时，指令选择第一个发现的输入通道作为输出。若选择方式为选择中间值，被选择的中间值总是可用输入通道输入值中的一个。当超过一个以上的值为中间值，指令选择第一个发现的输入通道作为输出。

### 监控ESEL指令

ESEL指令具有一个操作员面板。需要更多有关信息，请参阅附录：功能块控制面板。

**算术状态字：** 输出值将影响算术状态字设置。

**故障条件：** 无

**执行：**

| 条件： | 功能块动作： | 结构文本动作： |
|---|---|---|
| 预扫描 | 无动作。 | 无动作。 |
| 指令首次扫描 | 所有操作员请求输入被清零。<br>当ProgValueReset置位时，所有程序请求输入被清零。 | 所有操作员请求输入被清零。<br>当ProgValueReset置位时，所有程序请求输入被清零。 |
| 指令首次执行 | 指令被设置成操作员控制方式。 | 指令被设置成操作员控制方式。 |
| 输入使能(EnableIn)清零 | 输出使能(EnableOut)清零，指令不执行且输出不更新。 | 不影响 |
| 输入使能(EnableIn)置位 | 执行指令。<br>输出使能(EnableOut)置位 | 输入使能(EnableIn)一直置位。执行指令。 |
| 后期扫描 | 无动作 | 无动作 |

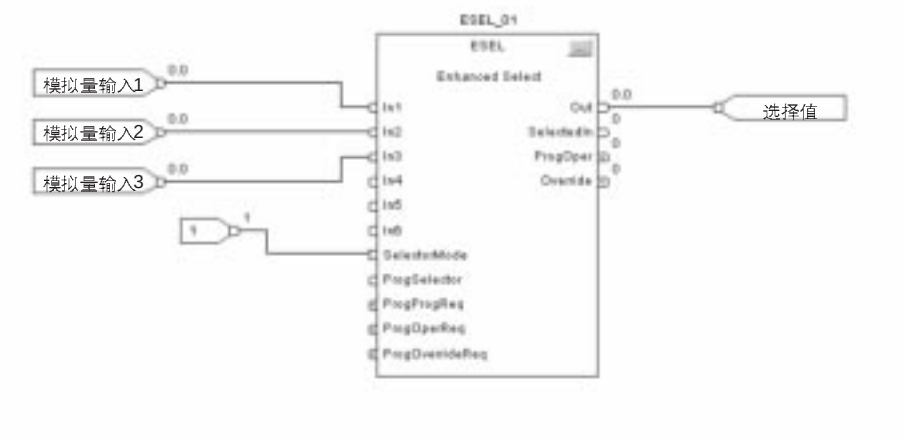
**实例：** ESEL指令根据选择方式选择通道1、2或3(In1,In2 或 In3)。在本例中，SelectorMode=1，表示选择最大值。指令判断最大的输入值并且设置输出值=最大输入值。

### 结构文本

```
ESEL_01.In1 := analog_input1;
ESEL_01.In2 := analog_input2;
ESEL_01.In3 := analog_input3;
ESEL_01.SelectorMode := 1;
ESEL(ESEL_01);

selected_value := ESEL_01.Out;
```

### 功能块



## 程序控制和操作员控制间切换

下图显示了ESEL指令如何在程序控制和操作员控制方式间切换。




(1) 用户可以保持ProgOperReq置位使指令锁定在操作员控制方式。

(2) 用户可以在ProgOperReq清零的情况下通过保持ProgProgReq置位使指令锁定在程序控制方式下。

## 上/下限限幅(HLL)

HLL指令可将模拟量输入值限制在两个预设值之间。用户可以选择上/下限、上限或下限。

### 操作数：

#### 结构文本

| 操作数 | 类型 | 格式 | 说明 |
|---|---|---|---|
| HLL标签 | HL_LIMIT | 结构体 | HLL结构体 |

#### 功能块

| 操作数 | 类型 | 格式 | 说明 |
|---|---|---|---|
| HLL标签 | HL_LIMIT | 结构体 | HLL结构体 |

#### HL_LIMIT结构体

| 输入参数： | 数据类型： | 说明： |
|---|---|---|
| EnableIn | BOOL | 功能块：<br>如果该位清零，则指令不执行输出不更新。<br>如果置位，指令执行。<br>缺省状态为置位<br>结构文本：<br>无影响。指令一直执行。 |
| In | REAL | 指令的模拟量输入信号。<br>有效值＝浮点数<br>缺省值＝0.0 |
| HighLimit | REAL | 输入上限。当HighLimit≤LowLimit时，指令置位状态字中的相应位且设置Out=LowLimt。<br>有效值=HighLimit>LowLimit<br>缺省值=最大正浮点数 |
| LowLimit | REAL | 输入下限。当HighLimit≤LowLimit时，指令置位状态字中的相应位且设置Out=LowLimt。<br>有效值=LowLimit<HighLimit<br>缺省值=最小负浮点数 |
| SelectLimit | DINT | 选择输入限幅方式。该输入参数有3种设置值：<br>**值：** **说明：**<br>0 使用上/下限<br>1 使用上限<br>2 使用下限<br><br>如果选择的限制方式无效，指令设定SelectLimit=0且置位状态字中相应位。<br>有效值=0到2<br>缺省值=0 |

| 输出参数： | 数据类型： | 说明： |
|---|---|---|
| EnableOut | BOOL | 输出使能。 |
| Out | REAL | 该指令算法计算出的输出结果。输出结果将影响算术状态字设置。 |
| HighAlarm | BOOL | 上限报警指示。当In ⩾HighLimit时置位。 |
| LowAlarm | BOOL | 下限报警指示。当In ⩽LowLimit时置位。 |
| Status | DINT | 功能块的状态字。 |
| InstructFault(Status.0) | BOOL | 指令检测到以下的运行错误。该错误不使控制器产生次要或主要错误。检测其它状态位以确定发生错误的类型。。 |
| LimitsInv(Status.1) | BOOL | HighLimit ⩽LowLimit时置位。 |
| SelectLimitInv(Status.2) | BOOL | 输入的限制方式选择(SelectLimit)值不是0,1,或2 |

**说明：**HLL指令使用下列规则确定Out输出值：

| 选择： | 条件： | 执行动作： |
|---|---|---|
| SelectLimit=0 (使用上/下限) | In<HighLimit且 In>LowLimit | Out=In |
| | In ⩾HighLimit | Out=HighLimit<br>HighAlarm被置位 |
| | In ⩽LowLimit | Out=LowLimit<br>LowAlarm被置位 |
| | HighLimit ⩽LowLimit | Out=LowLimit<br>HighAlarm被置位<br>LowAlarm被置位<br>LimitsInv被置位 |
| SelectLimit=1 (仅用上限) | In<HighLimit | Out=In |
| | In ⩾HighLimit | Out=HighLimit<br>HighAlarm被置位 |
| SelectLimit=2 (仅用下限) | In>LowLimit | Out=In |
| | In ⩽LowLimit | Out=LowLimit<br>LowAlarm被置位 |

**算术状态字：**输出值将影响算术状态字设置。

**故障条件：**无

| 条件： | 功能块动作： | 结构文本动作： |
|---|---|---|
| 预扫描 | 无动作。 | 无动作。 |
| 指令首次扫描 | 无动作。 | 无动作。 |
| 指令首次执行 | 无动作。 | 无动作。 |
| 输入使能(EnableIn)清零 | 输出使能(EnableOut)清零，指令不执行且输出不更新。 | 不影响 |
| 输入使能(EnableIn)置位 | 执行指令。<br>输出使能(EnableOut)置位 | 输入使能(EnableIn)一直置位。执行指令。 |
| 后期扫描 | 无动作 | 无动作 |

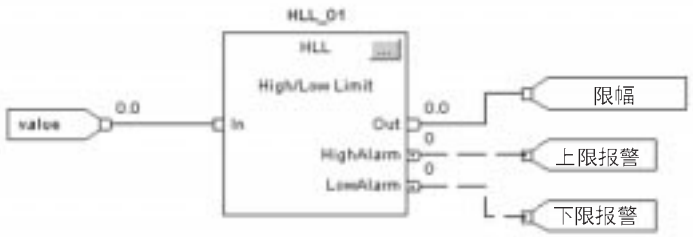
**实例：** HLL指令将输入限制在两个预设值之间，且在输入值超出限制范围时输出上限和下限报警信号。指令设置Out等于被限制的输入值。

### 结构文本

```
HLL_01.In := value;
HLL(HLL_01);

limited_value := HLL_01.Out;
high_alarm := HLL_01.HighAlarm;
low_alarm := HLL_01.LowAlarm;
```

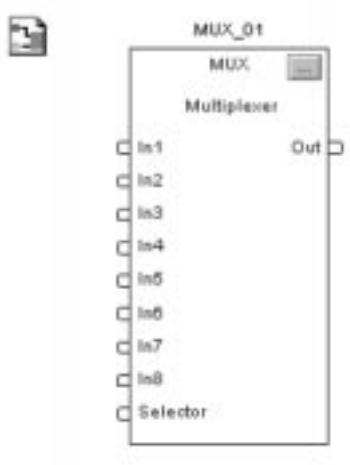
### 功能块



## 多路切换器(MUX)

MUX指令根据输入选择器方式从8路输入中选择1路。

### 操作数：




### 功能块

| 操作数： | 类型： | 格式： | 说明： |
|---|---|---|---|
| 块标签 | MULTIPLEXER | 结构体 | MUX结构体 |

### MULTIPLEXER结构体：

| 输入参数： | 数据类型： | 说明： |
|---|---|---|
| EnableIn | BOOL | 输入使能。如果该位清零，则指令不执行输出不更新。<br>缺省状态为置位 |
| In1 | REAL | 指令的模拟量输入信号1。<br>有效值＝浮点数<br>缺省值＝0.0 |
| In2 | REAL | 指令的模拟量输入信号2。<br>有效值＝浮点数<br>缺省值＝0.0 |
| In3 | REAL | 指令的模拟量输入信号3。<br>有效值＝浮点数<br>缺省值＝0.0 |
| In4 | REAL | 指令的模拟量输入信号4。<br>有效值＝浮点数<br>缺省值＝0.0 |
| In5 | REAL | 指令的模拟量输入信号5。<br>有效值＝浮点数<br>缺省值＝0.0 |
| In6 | REAL | 指令的模拟量输入信号6。<br>有效值＝浮点数<br>缺省值＝0.0 |

| 输入参数： | 数据类型： | 说明： |
|---|---|---|
| In7 | REAL | 指令的模拟量输入信号7。<br>有效值＝浮点数<br>缺省值＝0.0 |
| In8 | REAL | 指令的模拟量输入信号8。<br>有效值＝浮点数<br>缺省值＝0.0 |
| Selector | DINT | 指令的输入通道选择器。该输入参数决定送往输出值Out的输入通道号(1－8)。如果该值无效(包括0)，指令置位状态字中的相应位且保持Out(输出值)为当前值。<br>有效值=1到8<br>缺省值=0 |

| 输出参数： | 数据类型： | 说明： |
|---|---|---|
| EnableOut | BOOL | 输出使能。 |
| Out | REAL | 选择算法的输出结果。输出结果将影响算术状态字设置。 |
| Status | DINT | 功能块的状态字。 |
| InstructFault(Status.0) | BOOL | 指令检测以下的运行错误。该错误不使控制器产生次要或主要错误。检测其它状态位以确定发生错误的类型。 |
| SelectorInv(Status.1) | BOOL | Selector值无效。 |

**说明：**根据选择器数值，MUX指令设置输出Out等于8个输入值中的一个。

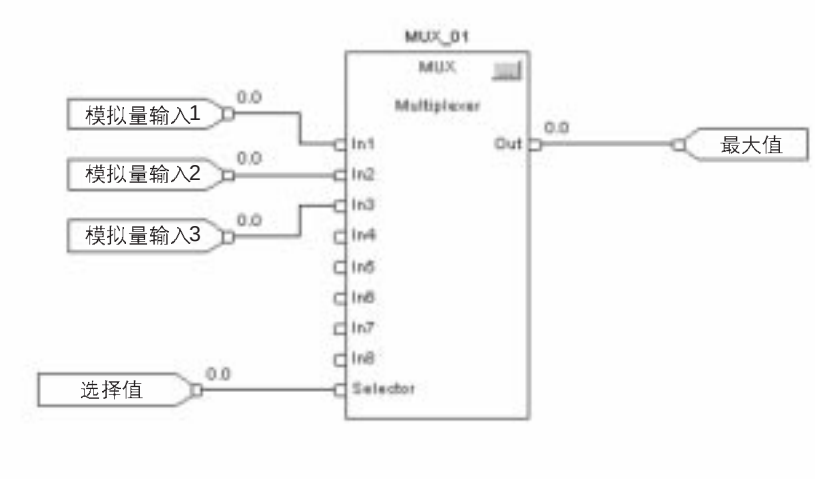
**算术状态字：**输出值将影响算术状态字设置。

**故障条件：**无

**执行：**

| 条件： | 功能块动作： |
|---|---|
| 预扫描 | 无动作。 |
| 指令首次扫描 | 内部参数清零 |
| 指令首次执行 | 无动作。 |
| 输入使能(EnableIn)清零 | 输出使能(EnableOut)清零，指令不执行，且输出不更新。 |
| 输入使能(EnableIn)置位 | 执行指令。<br>输出使能(EnableOut)置位 |
| 后期扫描 | 无动作 |

**实例：**MUX指令根据输入选择器值选择输入通道1、2或3。指令设置输出值Out=Inn。例如：当如果选择器值等于2，指令设置输出值Out等于模拟量输入2。



## 速率限幅器(RLIM)

RLIM指令用于限制输入信号在一段时间内的变化量。

### 操作数：



```
RLIM(RLIM_tag);
```

### 结构文本

| 操作数： | 类型： | 格式： | 说明： |
|---|---|---|---|
| RLIM标签 | RATE_LIMITER | 结构体 | RLIM结构体 |






### 功能块

| 操作数： | 类型： | 格式： | 说明： |
|---|---|---|---|
| RLIM标签 | RATE_LIMITER | 结构体 | RLIM结构体 |

### RATE_LIMITER结构体

| 输入参数： | 数据类型： | 说明： |
|---|---|---|
| EnableIn | BOOL | 功能块：<br>如果该位清零，则指令不执行且输出不更新。<br>如果置位，指令执行。<br>缺省状态为置位<br>结构文本：<br>无影响。指令一直执行。 |
| In | REAL | 指令的模拟量输入信号。<br>有效值＝浮点数<br>缺省值＝0.0 |
| IncRate | REAL | 输出值每秒增加的最大值。如果该值无效，指令设置IncRate=0.0且置位状态字中的相应位。<br>有效值=大于等于0.0的浮点数<br>缺省值=0.0 |
| DecRate | REAL | 输出值每秒减小的最大值。如果该值无效，指令设置DecRate=0.0且置位状态字中的相应位。<br>有效值=大于等于0.0的浮点数<br>缺省值=0.0 |
| ByPass | BOOL | 旁通算法请求。当置位时，指令设置Out = In。<br>缺省状态为清零 |
| TimingMode | DINT | 选择定时执行方式：<br>**有效值：** **说明：**<br>0 周期执行方式<br>1 过采样方式<br>2 实时采样方式<br>需要了解更多与定时方式有关的信息，请参阅附录：功能块属性。<br>有效值=0到2<br>缺省值=0 |

| 输入参数： | 数据类型： | 说明： |
|---|---|---|
| OversampleDT | REAL | 过采样方式的执行时间。<br>有效值=0到4194.303秒<br>缺省值=0 |
| RTSTime | DINT | 实时采样方式下模块的更新周期。<br>有效值=1到32,767ms<br>缺省值=1 |
| RTSTimeStamp | DINT | 实时采样方式下模块的时间戳值。<br>有效值=0到32,767ms<br>缺省值=0 |

| 输出参数： | 数据类型： | 说明： |
|---|---|---|
| EnableOut | BOOL | 输出使能。 |
| Out | REAL | 该指令算法计算出的输出结果。输出结果将影响算术状态字设置。 |
| DeltaT | REAL | 两次更新之间经过的时间，单位：秒。作为控制算法计算过程量输出值的时间间隔参数。 |
| Status | DINT | 功能块的状态字。 |
| InstructFault(Status.0) | BOOL | 指令检测以下的运行错误。该错误不使控制器产生次要或主要错误。检测其它状态位以确定发生错误的类型。 |
| IncRateInv(Status.1) | BOOL | IncRate<0时置位。指令设置IncRate=0 。 |
| DecRate(Status.2) | BOOL | DecRate<0时置位。指令设置DecRate=0 。 |
| TimingModeInv(Status.27) | BOOL | 定时方式值(TimingMode)无效。<br>需要了解更多与定时方式有关的信息，请参阅附录：功能块属性。 |
| RTSMissed(Status.28) | BOOL | 只用于实时采样方式。当ABS| DeltaT ®C RTSTime | >1(0.001秒)时，该位被置位。 |
| RTSTimeInv(Status.29) | BOOL | RTSTime值无效。 |
| RTSTimeStampInv(Status.30) | BOOL | RTSTimeStamp值无效。 |
| DeltaTInv(Status.31) | BOOL | DeltaT值无效。 |

说明：RLIM指令用于提供每秒内独立的增量或减量速率限幅。旁通输入参数允许用户暂停速率限幅，而直接把输入信号送往输出。

| 条件： | 执行动作： |
|---|---|
| 旁通置位 | $Out_n=In_n$<br>$Out_{n-1}=In_n$ |
| 旁通清零且DeltaT>0 | $Slope = \frac{In_n - Out_{n-1}}{DeltaT}$<br>如果Slope≤-DecRate则YSlope=-DecRate<br>如果-DecRate≤Slope≤IncRate则YSlope=Slope<br>如果IncRate≤Slope则YSlope=IncRate<br>Outn=Outn-1+DeltaT×YSlope<br>Outn-1=Outn<br>其中DeltaT以秒为单位。 |

**算术状态字：**输出值将影响算术状态字设置。

**故障条件：**无

**执行：**

| 条件： | 功能块动作： | 结构文本动作： |
|---|---|---|
| 预扫描 | 无动作 | 无动作 |
| 指令首次扫描 | $Out_{n-1}=In_n$ | $Out_{n-1}=In_n$ |
| 指令首次执行 | $Out_{n-1}=In_n$ | $Out_{n-1}=In_n$ |
| 输入使能(EnableIn)清零 | 输出使能(EnableOut)清零，指令不执行且输出不更新。 | 不影响 |
| 输入使能(EnableIn)置位 | 执行指令。<br>输出使能(EnableOut)置位 | 输入使能(EnableIn)一直置位。<br>执行指令。 |
| 后期扫描 | 无动作 | 无动作 |

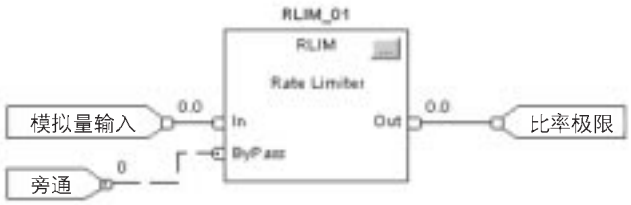
**实例：** RLIM指令使用IncRate值限制输入值。如果模拟量输入信号1的增量速率超过IncRate值，指令限制输入值增量。指令设置Out等于限制速率后的输入值。

### 结构文本

```
RLIM_01.In := analog_input1;
RLIM_01.Bypass := bypass;
RLIM(RLIM_01);

rate_limited := RLIM_01.Out;
```

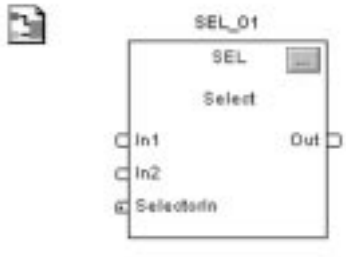
### 功能块



## 选择(SEL)

SEL指令根据一个数字量输入从两路输入信号中选择一路。

### 操作数：




### 功能块

| 操作数： | 类型： | 格式： | 说明： |
|---|---|---|---|
| SEL标签 | SELECT | 结构体 | SEL结构体 |

SELECT结构体

| 输入参数： | 数据类型： | 说明： |
|---|---|---|
| EnableIn | BOOL | 输入使能。如果该位清零，则指令不执行输出不更新。<br>缺省状态为置位 |
| In1 | REAL | 指令的模拟量输入信号1。<br>有效值＝浮点数<br>缺省值＝0.0 |
| In2 | REAL | 指令的模拟量输入信号2。<br>有效值＝浮点数<br>缺省值＝0.0 |
| SelectorIn | BOOL | 输入选择器，从In1和In2中选择一路输入。<br>缺省状态为清零。 |
| **输出参数：** | **数据类型：** | **说明：** |
| EnableOut | BOOL | 输出使能。 |
| Out | REAL | 该指令算法计算出的输出结果。输出结果将影响算术状态字设置。 |

说明：SEL指令执行操作如下：

| 条件： | 执行动作： |
|---|---|
| SelectorIn被置位 | Out=In2 |
| SelectorIn被清零 | Out=In1 |

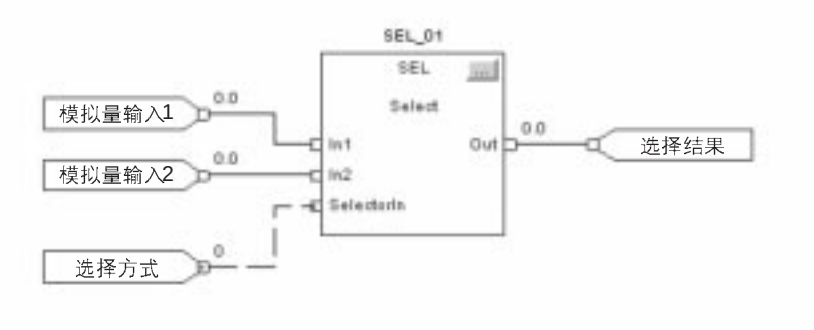
**算术状态字：**输出值将影响算术状态字设置。

**故障条件：**无

**执行：**

| 条件： | 功能块动作： |
|---|---|
| 预扫描 | 无动作 |
| 指令首次扫描 | 无动作 |
| 指令首次执行 | 无动作 |
| 输入使能(EnableIn)清零 | 输出使能(EnableOut)清零。 |
| 输入使能(EnableIn)置位 | 执行指令。<br>输出使能(EnableOut)置位 |
| 后期扫描 | 无动作 |

**实例：**SEL指令根据SelectorIn选择In1或In2。如果SelectorIn置位，指令设置Out=In2。如果SelectorIn清零，指令设置Out=In1。



## 选择取反(SNEG)

SNEG指令根据一个数字量输入选择输入值或输入值的相反数。

### 操作数：



```
SNEG(SNEG_tag);
```

**结构文本**

| 操作数： | 类型： | 格式： | 说明： |
|---|---|---|---|
| SNEG标签 | SELECTABLE_NEGATE | 结构体 | SNEG结构体 |






**功能块**

| 操作数： | 类型： | 格式： | 说明： |
|---|---|---|---|
| SNEG标签 | SELECTABLE_NEGATE | 结构体 | SNEG结构体 |

**SELECTABLE_NEGATE结构体**

| 输入参数： | 数据类型： | 说明： |
|---|---|---|
| EnableIn | BOOL | **功能块：**<br>如果该位清零，则指令不执行输出不更新。<br>如果置位，指令执行。<br>缺省状态为置位<br>**结构文本：**<br>无影响。指令一直执行。 |
| In | REAL | 指令的模拟量输入信号。 |
| NegateEnable | BOOL | 取反使能。如果NegateEnable设置，指令选择输入值的相反数作为输出结果。<br>缺省状态为置位。 |
| **输出参数：** | **数据类型：** | **说明：** |
| EnableOut | BOOL | 输出使能。 |
| Out | REAL | 该指令算法计算出的输出结果。输出结果将影响算术状态字设置。 |

说明：SNEG指令执行如下操作：

| 条件： | 执行动作： |
|---|---|
| NegateEnable被置位 | Out=-In |
| NegateEnable被清零 | Out=In |

**算术状态字：**算术状态字：输出值将影响算术状态字设置。

**故障条件：**故障条件：无

**执行：**

| 条件： | 功能块动作： | 结构文本动作： |
|---|---|---|
| 预扫描 | 无动作 | 无动作 |
| 指令首次扫描 | 无动作 | 无动作 |
| 指令首次执行 | 无动作 | 无动作 |
| 输入使能(EnableIn)清零 | 输出使能(EnableOut)清零，指令不执行且输出不更新。 | 不影响。 |
| 输入使能(EnableIn)置位 | 执行指令。<br>输出使能(EnableOut)置位 | 输入使能(EnableIn)一直置位。<br>指令执行。 |
| 后期扫描 | 无动作。 | 无动作。 |

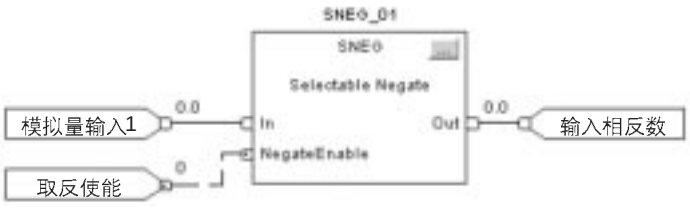
**实例：** 标签negate_enable决定是否对该指令的输入取反。如果NegateEnable清零，指令设置Out等于In；如果NegateEnable被置位，指令设置Out等于-In。

### 结构文本

```
SNEG_01.In := analog_input1
SNEG_01.NegateEnable := negate_enable;
SNEG(SNEG_01);

negate_value := SNEG_01.Out;
```

### 功能块



## 选择求和 (SSUM)

SSUM指令根据布尔输入值选择实际输入并求代数和。

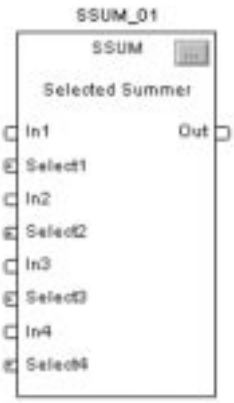
### 操作数：

```
SSUM(SSUM_tag);
```

### 结构文本

| 操作数： | 类型： | 格式： | 说明： |
|---|---|---|---|
| SSUM标签 | SELECTABLE_SUMMER | 结构体 | SSUM结构体 |

### 功能块




| 操作数： | 类型： | 格式： | 说明： |
|---|---|---|---|
| SSUM标签 | SELECTABLE_SUMMER | 结构体 | SSUM结构体 |

### SELECTABLE_SUMMER结构体

| 输入参数： | 数据类型： | 说明： |
|---|---|---|
| EnableIn | BOOL | **功能块：**<br>如果该位清零，则指令不执行且输出不更新。<br>如果置位，指令执行。<br>缺省状态为置位<br>**结构文本：**<br>无影响。指令一直执行。 |
| In1 | REAL | 求和输入信号1。<br>有效值＝浮点数<br>缺省值＝0.0 |
| Gain1 | REAL | 输入信号1的增益因数。<br>有效值＝浮点数<br>缺省值＝1.0 |
| Select1 | BOOL | 输入信号1的选择信号。<br>缺省状态为清零 |
| In2 | REAL | 求和输入信号2。<br>有效值＝浮点数<br>缺省值＝0.0 |
| Gain2 | REAL | 输入信号2的增益因子。<br>有效值＝浮点数<br>缺省值＝1.0 |
| Select2 | BOOL | 输入信号2的选择信号。<br>缺省状态为清零 |

| 输入参数： | 数据类型： | 说明： |
|---|---|---|
| In3 | REAL | 求和输入信号3。<br>有效值＝浮点数<br>缺省值＝0.0 |
| Gain3 | REAL | 输入信号3的增益因子。<br>有效值＝浮点数<br>缺省值＝1.0 |
| Select3 | BOOL | 输入信号3的选择信号。<br>缺省状态为清零 |
| In4 | REAL | 求和输入信号4。<br>有效值＝浮点数<br>缺省值＝0.0 |
| Gain4 | REAL | 输入信号4的增益因子。<br>有效值＝浮点数<br>缺省值＝1.0 |
| Select4 | BOOL | 输入信号4的选择信号。<br>缺省状态为清零 |
| In5 | REAL | 求和输入信号5。<br>有效值＝浮点数<br>缺省值＝0.0 |
| Gain5 | REAL | 输入信号5的增益因子。<br>有效值＝浮点数<br>缺省值＝1.0 |
| Select5 | BOOL | 输入信号5的选择信号。<br>缺省状态为清零 |
| In6 | REAL | 求和输入信号6。<br>有效值＝浮点数<br>缺省值＝0.0 |
| Gain6 | REAL | 输入信号6的增益因子。<br>有效值＝浮点数<br>缺省值＝1.0 |
| Select6 | BOOL | 输入信号6的选择信号。<br>缺省状态为清零 |
| In7 | REAL | 求和输入信号7。<br>有效值＝浮点数<br>缺省值＝0.0 |
| Gain7 | REAL | 输入信号7的增益因子。<br>有效值＝浮点数<br>缺省值＝1.0 |
| Select7 | BOOL | 输入信号7的选择信号。<br>缺省状态为清零 |
| In8 | REAL | 求和输入信号8。<br>有效值＝浮点数<br>缺省值＝0.0 |

| 输入参数： | 数据类型： | 说明： |
|---|---|---|
| Gain8 | REAL | 输入信号8的增益因子。<br>有效值＝浮点数<br>缺省值＝1.0 |
| Select8 | BOOL | 输入信号8的选择信号。<br>缺省状态为清零 |
| Bias | REAL | 偏移量信号输入。指令将Bias与输入累加值相加。<br>有效值=浮点数<br>缺省值=0.0 |

| 输出参数： | 数据类型： | 说明： |
|---|---|---|
| EnableOut | BOOL | 输出使能。 |
| Out | REAL | 该指令算法计算出的输出结果。输出结果将影响算术状态字设置。 |

**说明：** SSUM执行如下操作：

| 条件： | 动作： |
|---|---|
| 无输入In被选择 | Out=Bias |
| 输入In被选择 | $Out = \sum_{n=1}^{8}(In_n \times Gain_n) + Bias$ |

**算术状态字：** 输出值将影响算术状态字设置。

**故障条件：** 无

**执行：**

| 条件： | 功能块动作： | 结构文本动作： |
|---|---|---|
| 预扫描 | 无动作 | 无动作 |
| 指令首次扫描 | 无动作 | 无动作 |
| 指令首次执行 | 无动作 | 无动作 |
| 输入使能(EnableIn)清零 | 输出使能(EnableOut)清零，指令不执行且输出不更新。 | 不影响 |
| 输入使能(EnableIn)置位 | 执行指令。<br>输出使能(EnableOut)置位 | 输入使能(EnableIn)一直置位。<br>指令执行。 |
| 后期扫描 | 无动作 | 无动作 |

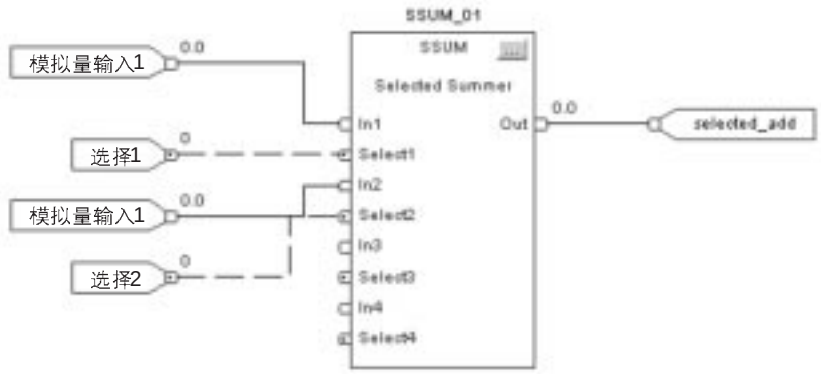
**实例：** select1和select2的值分别决定是否选择模拟量输入1和模拟量输入2。指令累加被选择的输入值，并把结果送往输出Out。

### 结构文本

```
SSUM_01.In1 := analog_input1;
SSUM_01.Select1 := select1;
SSUM_01.In2 := analog_input2;
SSUM_01.Select2 := select2;
SSUM(SSUM_01);

selected_add := SSUM_01.Out;
```

### 功能块



# 统计指令
# (MAVE, MAXC, MINC, MSTD)

## 概述

以下统计指令可用：

| 如果用户需要： | 所用指令： | 可用语言： | 参考页码： |
|---|---|---|---|
| 计算时间平均值 | 移动均值运算(MAVE) | 结构文本<br>功能块 | 5月2日 |
| 获取规定时间内信号最大值 | 获取最大值 (MAXC) | 结构文本<br>功能块 | 5月6日 |
| 获取规定时间内信号最小值 | 获取最小值 (MINC) | 结构文本<br>功能块 | 5月8日 |
| 计算移动标准偏差值 | 移动标准偏差(MSTD) | 结构文本<br>功能块 | 5月10日 |

## 移动均值运算(MAVE)

MAVE指令可计算In信号的时间平均值。该条指令可选择支持用户特定的权重。

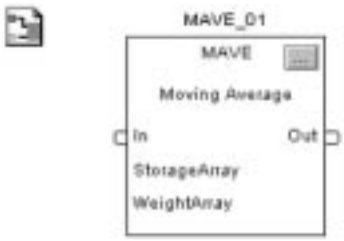
### 操作数：



```
MAVE(MAVE_tag,storage,weight);
```

### 结构文本

| 操作数： | 类型： | 格式： | 说明： |
|---|---|---|---|
| MAVE标签 | MOVING-AVERAGE | 结构体 | MAVE结构体 |
| 存储 | REAL | 数组 | 保持移动时间平均采样值。数组尺寸不少于NumberOfSamples中的元素数。 |
| 权重 | REAL | 数组 | (可选)<br>用于加权平均。<br>数组元素数不少于NumberOfSamples中的元素数。<br>[0]元素代表最新采样值；[n]元素代表最早采样值。 |




### 功能块

操作数与结构文本的FGEN指令相同。

### MOVING_AVERAGE 结构体

| 输入参数： | 数据类型： | 说明： |
|---|---|---|
| EnableIn | BOOL | 功能块：<br>如果该位清零，则指令不执行输出不更新。<br>如果置位，指令执行。<br>缺省状态为置位。<br>结构文本：<br>无影响。指令一直执行。 |
| In | REAL | 指令的模拟量信号输入。<br>有效值=浮点数<br>缺省值=0.0 |
| InFault | BOOL | 输入错误指示。In值从模拟量输入读取时，InFault值通常由模拟量输入的错误状态字控制。InFault置位时，表明输入信号有偏差，指令置位状态字中的相应位。指令保持Out的当前值。InFault由置位转为清零状态，则指令初始化平均运算算法并继续执行。<br>缺省状态为清零。 |

| 输入参数： | 数据类型： | 说明： |
|---|---|---|
| Initialize | BOOL | 初始化指令输入。当Initialize置位时，指令设置Out＝In；除非InFault置位，否则指令保持Out等于当前值。如果Initialize由置位变为清零状态，指令初始化均值算法并连续执行。<br>缺省状态为清零。 |
| SampleEnable | BOOL | In输入采样使能。SampleEnable置位时，指令将输入值In至存储数组并计算Out的值。SampleEnable和Initialize被清零，指令保持Out值为当前值。<br>缺省时为置位状态。 |
| NumberOfSamples | DINT | 计算使用采样点数。若该值无效，则指令置位状态字中的相应位并保持Out为当前值。当NumberOfSamples再次变为有效时，指令将初始化均值算法并连续执行。<br>有效值=1到(StorageArray或WeightArray(如果使用)的最小值)<br>缺省值=1 |
| UseWeights | BOOL | 指令的均值输入。置位时，采用加权的方法计算Out值；清零时，采用统一的方法计算Out值。<br>缺省为清零状态。 |

| 输出参数： | 数据类型： | 说明： |
|---|---|---|
| EnableOut | BOOL | 输出使能。 |
| Out | REAL | 该指令算法计算出的输出结果。输出结果将影响算术状态字设置。 |
| Status | DINT | 功能块状态字。 |
| InstructFault(Status.0) | BOOL | 指令检测以下的运行错误。该错误不使控制器产生次要或主要错误。检测其它状态位以确定发生错误的类型。 |
| InFaulted(Status.1) | BOOL | In状态错误(InFault置位)。 |
| NumberOfSamplnv (Status.2) | BOOL | NumberOfSamples无效或与数组尺寸不匹配。 |

**说明：** MAVE指令计算输入信号的加权或非加权移动平均时间。NumberOfSamples指定平均移动时间的范围。当SampleEnable置位时，则每次指令扫描都将In的值存入存储数组并覆盖原来的值。当UseWeights置位时，用户可配置每个Inn值的Weightn数值。

MAVE指令使用以下公式：

| 条件： | 公式： |
| --- | --- |
| 加权平均算法<br>UseWeights置位 | $Out = \sum_{n=1}^{NumberOfSamples} Weight_n \times In_n$ |
| 统一平均算法<br>UseWeights 清零 | $Out = \frac{\sum_{n=1} In_n}{NumberOfSamples}$ |

指令不能将无效的In值(NAN或±INF)输入到存储数组。In 值无效时，指令设置Out等于In并置位算术溢出状态标志。In值变为有效时，指令初始化均值算法并连续执行。

用户可以在运行时改变NumberOfSamples参数。如果用户增加采样点数，指令将增加从当前的采样数到新的采样数的平均值。减小采样数，指令将重新计算采样数组中起始值到新的NumberOfSamples值之间数据的平均值。

## 初始化均值算法

一定条件下，例如首次扫描指令和首次执行指令时，指令需要初始化移动均值算法。初始化后，指令将采样点数组清空并逐次平均1到NumberOfSamples之间的采样值。

例如：

NumberOfSamples=3且UseWeights 置位

扫描1：Out=Inn*Weight1

扫描2：Out=(Inn*Weight1)+( Inn-1*Weight2)

扫描3：Out=(Inn*Weight1)+ ( Inn-1*Weight2)+ ( Inn-2*Weight3)

NumberOfSamples=3且UseWeights 清零

扫描1：Out=Inn/1

扫描2：Out=(Inn+ Inn-1)/2

扫描3：Out=(Inn+ Inn-1+ Inn-2)/NumberOfSamples

**算术状态标志：**输出将影响算术状态标志。

**故障条件：**无

### 执行：

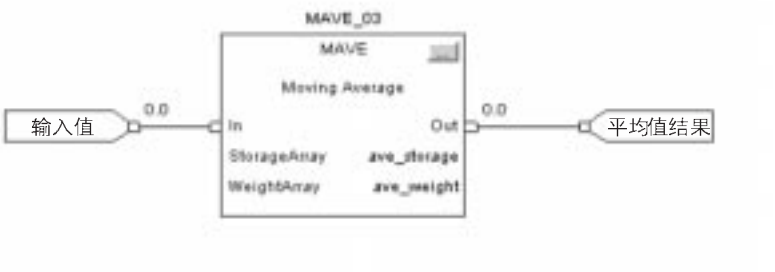
| 条件： | 功能块动作： | 结构文本动作： |
|---|---|---|
| 预扫描 | 无动作 | 无动作 |
| 首次指令扫描 | 如InFault清零，指令初始化运算算法并连续执行。 | 如InFault清零，指令初始化运算算法并连续执行。 |
| 首次指令执行 | 如InFault清零，指令初始化运算算法并连续执行。 | 如InFault清零，指令初始化运算算法并连续执行。 |
| 输入使能(EnableIn) 清零 | EnableOut 清零，指令不执行且输出不更新。 | 不影响 |
| 输入使能(EnableIn) 置位 | 执行指令。<br>输出使能(EnableOut)置位 | 输入使能(EnableIn)一直置位。<br>指令执行。 |
| 后期扫描 | 无动作 | 无动作 |

**实例：**每次扫描，指令将input_value放入存储数组。指令可选择使用加权数组中的权值计算存储数组的平均值，并将结果存入Out中。

### 结构文本

```
MAVE_03.In := input_value;
MAVE(MAVE_03,ave_storage,ave_weight);
ave_result := MAVE_03.Out;
```

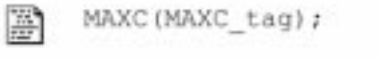
### 功能块



## 获取最大值(MAXC)

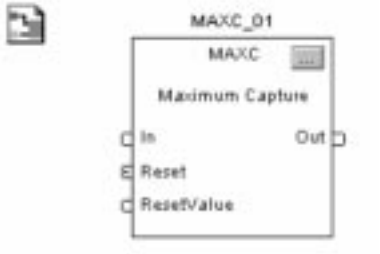
MAXC指令可获取先前的输入信号中最大值。

### 操作数：




#### 结构文本

| 操作数： | 类型： | 格式： | 说明： |
|---|---|---|---|
| MAXC标签 | MAXIMUM_CAPTURE | 结构体 | MAXC结构体 |




#### 功能块

| 操作数： | 类型： | 格式： | 说明： |
|---|---|---|---|
| MAXC标签 | MAXIMUM_CAPTURE | 结构体 | MAXC结构体 |

MAXIMUM_CAPTURE结构体

| 输入参数： | 数据类型： | 说明： |
|---|---|---|
| EnableIn | BOOL | **功能块：**<br>如果该位清零，则指令不执行且输出不更新。<br>如果置位，指令执行。<br>缺省状态为置位<br>**结构文本：**<br>无影响。指令一直执行。 |
| In | REAL | 指令的模拟量信号输入。<br>有效值=浮点数<br>缺省值=0.0 |
| Reset | BOOL | 复位控制算法请求。Reset置位时，Out等于ResetValues。<br>缺省状态为清零。 |
| ResetValue | REAL | 指令输出复位值。Reset置位时，Out等于ResetValues。<br>缺省值=0.0 |
| **输出参数：** | **数据类型：** | **说明：** |
| EnableOut | BOOL | 输出使能。 |
| Out | REAL | 该指令算法计算出的输出结果。输出结果将影响算术状态字设置。 |

**说明：**MAXC指令执行的运算算法：

| 条件： | 动作： |
|---|---|
| Reset置位 | $Out_{n-1}$=ResetValue<br>Out=ResetValue |
| Reset清零 | Out=In 当 In>$Out_{n-1}$<br>Out=$Out_{n-1}$ 当 In≤$Out_{n-1}$<br>$Out_{n-1}$=Out |

**算术状态标志：**该输出影响算术状态标志。

**故障条件：**无

**执行：**

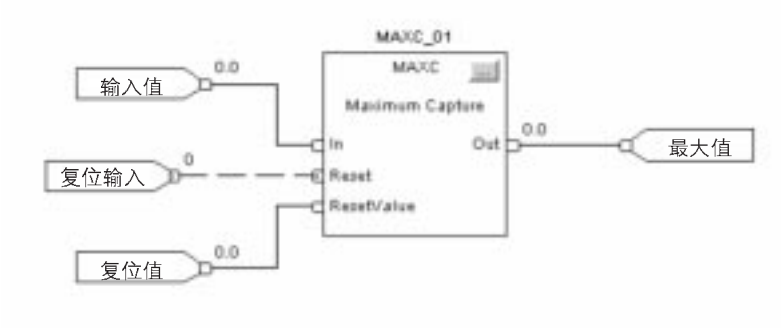
| 条件： | 功能块动作： | 结构文本动作： |
|---|---|---|
| 预先扫描 | 无动作 | 无动作 |
| 首次指令扫描 | $Out_{n-1}$=In | $Out_{n-1}$=In |
| 首次指令执行 | $Out_{n-1}$=In | $Out_{n-1}$=In |
| 输入使能(EnableIn) 清零 | 输出使能清零。指令不工作且输出不更新。 | 不影响 |
| 输入使能(EnableIn) 置位 | 执行指令。<br>输出使能置位。 | 输入使能(EnableIn)一直置位。<br>指令执行。 |
| 后期扫描 | 无动作 | 无动作 |

**实例：**如果Reset置位，则设置Out等于ResetValue。如果Reset清零且In>Outn-1则指令使Out等于In，否则Out等于 Outn-1。

### 结构文本

```
MAXC_01.In := input_value;
MAXC_01.Reset := reset_input;
MAXC_01.ResetValue := reset_value;
MAXC(MAXC_01);
maximum := MAXC_01.Out;
```

### 功能块



## 获取最小值(MINC)

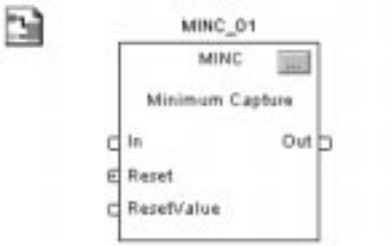
MINC指令可获取先前的输入信号中最小值。

**操作数：**




### 结构文本

| 操作数： | 类型： | 格式： | 说明： |
|---|---|---|---|
| MINC标签 | MINIMUM_CAPTURE | 结构体 | MINC结构体 |




### 功能块

| 操作数： | 类型： | 格式： | 说明： |
|---|---|---|---|
| MINC标签 | MINIMUM_CAPTURE | 结构体 | MINC结构体 |

### MINIMUM_CAPTURE结构体

| 输入参数： | 数据类型： | 说明： |
|---|---|---|
| EnableIn | BOOL | **功能块：**<br>如果该位清零，则指令不执行且输出不更新。<br>如果置位，指令执行。<br>缺省状态为置位<br>**结构文本：**<br>无影响。指令一直执行。 |
| In | REAL | 指令的模拟量信号输入。<br>有效值=浮点数<br>缺省值=0.0 |
| Reset | BOOL | 控制算法复位请求。Reset置位，Out=ResetValues<br>缺省状态为清零。 |
| ResetValue | REAL | 指令的复位值。Reset置位时，指令设置Out等于ResetValues。<br>有效值＝浮点数<br>缺省值=0.0 |
| **输出参数：** | **数据类型：** | **说明：** |
| EnableOut | BOOL | 输出使能。 |
| Out | REAL | 该指令算法计算出的输出结果。输出结果将影响算术状态字设置。 |

**说明：** MINC指令执行的运算算法：

| 条件： | 动作： |
|---|---|
| Reset置位 | $Out_{n-1}$=ResetValue<br>Out=ResetValue |
| Reset清零 | 当 In<$Out_{n-1}$，Out=In<br>当 In≥$Out_{n-1}$，Out=$Out_{n-1}$<br>$Out_{n-1}$=Out |

**算术状态标志：**该输出影响算术状态标志。

**故障条件：**无

**执行：**

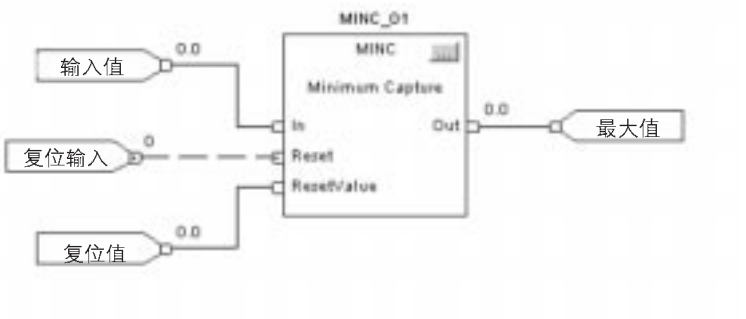
| 条件： | 功能块动作： | 结构文本动作： |
|---|---|---|
| 预扫描 | 无动作 | 无动作 |
| 首次指令扫描 | $Out_{n-1}$=In | $Out_{n-1}$=In |
| 首次指令执行 | $Out_{n-1}$=In | $Out_{n-1}$=In |
| 输入使能清零 | 输出使能(EnableOut)清零，指令不执行，输入不更新。 | 不影响 |
| 输入使能置位 | 执行指令。<br>输出使能(EnableOut)置位。 | 输入使能(EnableIn)一直置位。<br>指令执行。 |
| 后期扫描 | 无动作 | 无动作 |

**举例：**Reset置位时，指令设 置Out等于ResetValue。如果Reset清零且In<Outn-1，则指令设置Out等于In。否则，指令设置Out等于 Outn-1。

### 结构文本

```
MINC_01.In := input_value;
MINC_01.Reset := reset_input;
MINC_01.ResetValue := reset_value;
MINC(MINC_01);
minimum := MINC_01.Out;
```

### 功能块



## 移动标准偏差 (MSTD)

MSTD指令计算In信号的移动标准偏差和平均值。

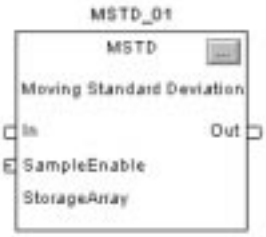
### 操作数：

### 结构文本

```
MSTD(MSTD_tag,storage);
```

| 操作数： | 类型： | 格式： | 说明： |
|---|---|---|---|
| MSTD标签 | MOVING_STD_DEV | 结构体 | MSTD结构体 |
| 存储 | REAL | 数组 | 保持In 采样值。数组尺寸至少为NumberOfSamples数值。 |

### 功能块




| 操作数： | 类型： | 格式： | 说明： |
|---|---|---|---|
| MSTD标签 | MOVING_STD_DEV | 结构体 | MSTD结构体 |
| 存储 | REAL | 数组 | 保持In 采样值。数组尺寸至少为NumberOfSamples数值。 |

### MOVING_STD_DEV结构

| 输入参数： | 数据类型： | 说明： |
|---|---|---|
| EnableIn | BOOL | 功能块：<br>如果该位清零，则指令不执行且输出不更新。<br>如果置位，指令执行。<br>缺省状态为置位<br>结构文本：<br>无影响。指令一直执行。 |
| In | REAL | 指令的模拟量信号输入。<br>有效值=浮点数<br>缺省值=0.0 |
| InFault | BOOL | 输入错误指示。In值从模拟量输入读取时，InFault通常由模拟量输入的错误状态字控制。InFault置位时，表明输入信号有错误，指令置位状态字中的相应位并保持Out的当前值和Average当前值。当InFault由置位变为清零状态时，指令初始化均值算法并连续执行。<br>缺省状态为清零 |

| 输入参数： | 数据类型： | 说明： |
|---|---|---|
| Initialize | BOOL | 初始化指令的输入。Initialize置位时，指令设定Out=0.0且Average=In；如果InFault不置位，指令保持Out和Average的当前值。当Initialize由置位变为清零状态，指令初始化标准偏差运算算法并连续执行。<br>缺省状态为清零 |
| SampleEnable | BOOL | 输入In采样使能。置位时，输入In数值送至存储数组并计算出Out和Average的值。当SampleEnable清零且Initialize 清零时，指令保持Out和Average的当前值。<br>缺省状态为清零 |
| NumberOfSamples | DINT | 计算用到的采样点个数。若该值无效，则指令置位状态字中相应位，并保持Out和Average为当前值。当NumberOfSamples再次有效时，指令将初始化标准偏差算法并连续执行。<br>有效值=1到存储数组的尺寸<br>缺省值=1 |

| 输出参数： | 数据类型： | 说明： |
|---|---|---|
| EnableOut | BOOL | 输出使能 |
| Out | REAL | 该指令算法计算出的输出结果。SampleEnable清零时，指令保持Out当前值。输出结果将影响算术状态字设置。 |
| Average | REAL | 运算算法计算的平均值。 |
| Status | DINT | 功能块状态字 |
| InstructFault(Status.0) | BOOL | 指令检测以下的运行错误。该错误不使控制器产生次要或主要错误。检测其它状态位以确定发生错误的类型。 |
| InFaulted(Status.1) | BOOL | In 状态错误。InFault置位。 |
| NumberOfSamplnv (Status.2) | BOOL | NumberOfSamples无效或与数组尺寸不匹配。 |

**说明：** MSTD指令支持任意长度的输入队列。SampleEnable置位时，每次扫描，输入一个In值到存储数组。存储数组输入满时，新In值的输入将旧值覆盖。

MSTD指令采用下面的方程控制输出。

| 条件： | 等式： |
|---|---|
| Average | $Average = \frac{\sum_{n=1}^{NumberOfSamples} In_n}{NumberOfSamples}$ |
| Out | $Out = \sqrt{\frac{\sum_{n=1}^{NumberOfSamples} (In_n - Average)^2}{NumberOfSamples}}$ |

指令不能将无效的In值(NAN或±INF)输入到存储数组。In 值无效时，指令设定Out等于In，Average等于In并置位算术溢出状态标志位。In值变为有效时，指令初始化标准偏差算法并连续执行。

用户可以在运行时修改NumberOfSamples参数。如果用户增大参数值，指令从当前的采样量逐渐增加到新的采样量。如果用户减小参数值，指令将重新计算采样数组中从起始值到NumberOfSamples值的标准偏差。

## 初始化标准偏差算法

一定条件下，例如首次扫描指令和首次执行指令，需要指令初始化标准偏差算法。初始化后，指令将采样值数组清空并重新计算从1到NumberOfSamples采样值数组的标准偏差。例如：

NumberOfSamples=3

扫描1:Average=$In_n$/1

Out=Square root(($(In_n-Average)^2$)/1)

扫描2:Average=($In_n+In_{n-1}$)/2

Out=Square root(($(In_n-Average)^2+(In_n-1-Average)^2$)/2)

扫描3:Average=($In_n+In_{n-1}+In_{n-2}$)/NumberOfSamples

Out=Square root(($(In_n-Average)^2+(In_{n-1}-Average)^2+(In_{n-2}-Average)^2$) /NumberOfSamples)

**算术状态标志：**该输出影响算术状态标志。

**故障条件：**无

**执行：**

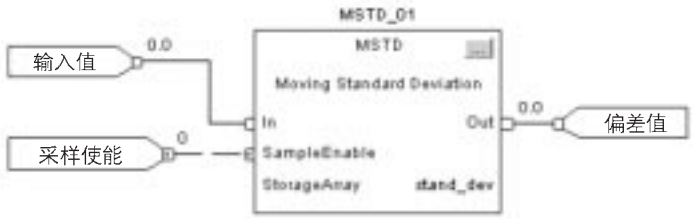
| 条件： | 功能块动作： | 结构文本动作： |
|---|---|---|
| 预扫描 | 无动作 | 无动作 |
| 首次指令扫描 | 若InFault清零，指令初始化算法并连续执行。 | 若InFault清零，指令初始化算法并连续执行。 |
| 首次指令执行 | 若InFault清零，指令初始化算法并连续执行。 | 若InFault清零，指令初始化算法并连续执行。 |
| 输入使能(EnableIn)清零 | EnableOut 清零，指令不执行且输出不更新。 | 不影响 |
| 输入使能(EnableIn)置位 | 执行指令。<br>输出使能(EnableOut)置位 | 输入使能(EnableIn)一直置位。<br>指令执行。 |
| 后期扫描 | 无动作 | 无动作 |

**实例：** SampleEnable置位时，则每次指令扫描将In值放入存储数组中，并计算存储数组的标准偏差，将结果存入Out。

### 结构文本

```
MSTD_01.In := input_value;
MSTD_01.Sample_enable := enable_sample;
MSTD(MSTD_01,stand_dev);
deviation := MSTD_01.Out;
```

### 功能块



注释：

# 移动/逻辑指令
# (DFF, JKFF, RESD, SETD)

## 概述

以下移动/逻辑指令可用：

| 如果用户需要： | 使用指令： | 可用语言： | 参考页码： |
|---|---|---|---|
| Clock输入跳变沿时将Q输出设置为D输入状态。 | D触发器(DFF) | 结构文本<br>功能块 | 6月2日 |
| Clock输入跳变沿时将Q和QNot输出逻辑值取反。 | JK触发器(JKFF) | 结构文本<br>功能块 | 6月4日 |
| 使用Set和Reset输入来控制输出锁存，<br>此时Reset输入的优先级高于Set输入。 | 复位优先(RESD) | 结构文本<br>功能块 | 6月6日 |
| 使用Set和Reset输入来控制输出锁存，<br>此时Set输入的优先级高于Reset输入。 | 置位优先(SETD) | 结构文本<br>功能块 | 6月8日 |

## D 触发器(DFF)

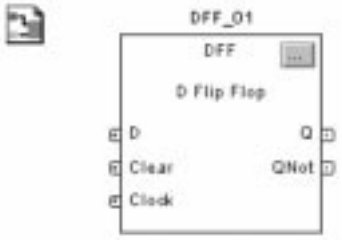
Clock输入由清零变为置位状态时，DFF指令将Q输出值设置为D输入值。设置QNot输出与Q输出值相反。

**操作数：**

```
DFF(DFF_tag);
```

**结构文本**

| 操作数： | 类型： | 格式： | 说明： |
|---|---|---|---|
| DFF标签 | FLIP_FLOP_D | 结构体 | DFF结构体 |




**功能块**

| 操作数： | 类型： | 格式： | 说明： |
|---|---|---|---|
| DFF标签 | FLIP_FLOP_D | 结构体 | DFF结构体 |

**FLIP_FLOP_D结构体**

| 输入参数： | 数据类型： | 说明： |
|---|---|---|
| EnableIn | BOOL | **功能块：**<br>如果该位清零，则指令不执行输出不更新。<br>如果置位，指令执行。<br>缺省状态为置位<br>**结构文本：**<br>无影响。指令一直执行。 |
| D | BOOL | 指令输入。<br>缺省状态为清零。 |
| Clear | BOOL | 指令的Clear输入。如果该位置位，指令清零Q并置位QNot。<br>缺省为清零状态。 |
| Clock | BOOL | 指令的时钟输入。<br>缺省状态为清零。 |
| **输出参数：** | **数据类型：** | **说明：** |
| EnableOut | BOOL | 输出使能。 |
| Q | BOOL | 指令输出 |
| QNot | BOOL | Q输出值取反 |

**说明：** Clear置位时，指令将Q清零并将QNot置位。反之，若Clock被置位且Clockn-1为清零状态，那么指令设置Q=D且QNot=NOT(D)。
每次扫描时指令设定Clockn-1 =Clock。

**算术状态标志：** 无影响

**故障条件：**无

**执行：**

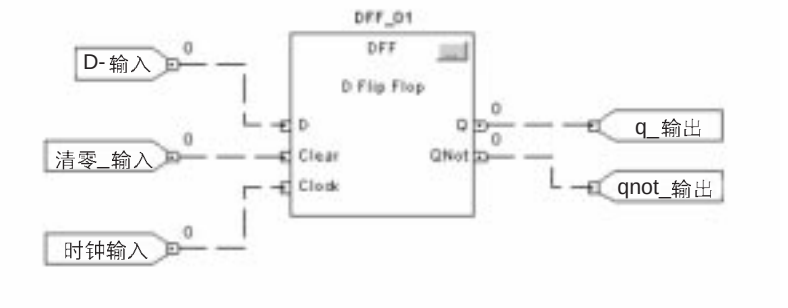
| 条件： | 功能块动作： | 结构文本动作： |
|---|---|---|
| 预扫描 | 无动作 | 无动作 |
| 指令首次扫描 | $Clock_{n-1}$置位 | $Clock_{n-1}$置位 |
| | Q清零 | Q清零 |
| | QNot置位 | QNot置位 |
| 指令首次执行 | $Clock_{n-1}$置位 | $Clock_{n-1}$置位 |
| | Q清零 | Q清零 |
| | QNot置位 | QNot置位 |
| 输入使能(EnableIn)清零 | 输出使能(EnableOut) 清零，指令不执行且输出不更新。 | 不影响 |
| 输入使能(EnableIn)置位 | 指令执行。 | 输入使能(EnableIn)一直置位。 |
| | 输出使能(EnableOut)置位。 | 指令执行。 |
| 后期扫描 | 无动作 | 无动作 |

**实例：**Clock上升沿时，DFF指令设定Q等于D 。若Clear置位，则指令将Q清零。DFF指令设定QNot为Q的相反状态。

### 结构文本

```
DFF_01.D := d_input;
DFF_01.Clear := clear_input;
DFF_01.Clock := clock_input;
DFF(DFF_01);
q_output := DFF_01.Q;
qnot_output := DFF_01.QNot;
```

### 功能块



## JK触发器(JKFF)

Clock输入由清零跳变为置位状态，JKFF指令将Q和QNot输出逻辑值取反。

**操作数：**

```
JKFF(JKFF_tag);
```

### 结构文本

| 操作数： | 类型： | 格式： | 说明： |
|---|---|---|---|
| JKFF标签 | FLIP_FLOP_JK | 结构体 | JKFF结构体 |

### 功能块




| 操作数： | 类型： | 格式： | 说明： |
|---|---|---|---|
| JKFF标签 | FLIP_FLOP_JK | 结构体 | JKFF结构体 |

### FLIP_FLOP_JK结构体

| 输入参数： | 数据类型： | 说明： |
|---|---|---|
| EnableIn | BOOL | **功能块：**<br>如果该位清零，则指令不执行输出不更新。<br>如果置位，指令执行。<br>缺省状态为置位<br>**结构文本：**<br>无影响。指令一直执行。 |
| Clear | BOOL | Clear指令输入。置位时，指令将Q清零并将QNot置位。<br>缺省状态为清零。 |
| Clock | BOOL | 指令的Clock输入。<br>缺省状态为清零。 |
| **输出参数：** | **数据类型：** | **说明：** |
| EnableOut | BOOL | 输出使能。 |
| Q | BOOL | 指令输出。 |
| QNot | BOOL | Q输出值取反。 |

**说明：** Clear置位时，则指令将Q清零并将QNot置位。反之，若Clock被置位且Clockn-1为清零状态，那么指令触发Q和QNot。

每次扫描指令设定Clockn-1等于Clock。

**算术状态标志：** 算术状态标志：无影响

**故障条件：** 故障条件：无

**执行：**

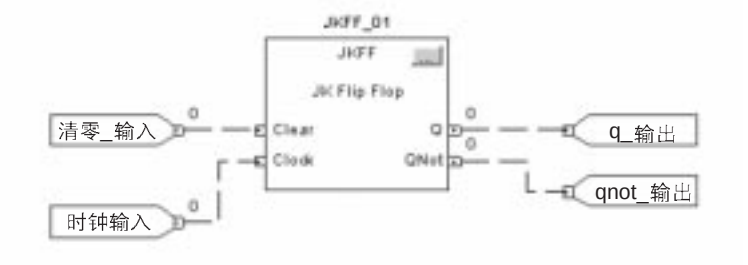
| 条件： | 功能块动作： | 结构文本动作： |
|---|---|---|
| 预扫描 | 无动作 | 无动作 |
| 指令首次扫描 | Clockn-1置位<br>Q清零<br>QNot置位 | Clockn-1置位<br>Q清零<br>QNot置位 |
| 指令首次执行 | Clockn-1置位<br>Q清零<br>QNot置位 | Clockn-1置位<br>Q清零<br>QNot置位 |
| 输入使能(EnableIn)清零 | 输出使能(EnableOut) 清零，指令不执行且输出不更新。 | 不影响 |
| 输入使能(EnableIn)置位 | 执行指令<br>输出使能(EnableOut)置位 | 输入使能(EnableIn)一直置位。<br>指令执行。 |
| 后期扫描 | 无动作 | 无动作 |

**实例：** Clock上升沿时，JKFF指令触发Q值 。若Clear置位，则Q将保持清零。JKFF指令设定QNot为Q的相反状态。

### 结构文本

```
JKFF_01.Clear := clear_input;
JKFF_01.Clock := clock_input;
JKFF(JKFF_01);
q_output := JKFF_01.Q;
qnot_output := JKFF_01.QNot;
```

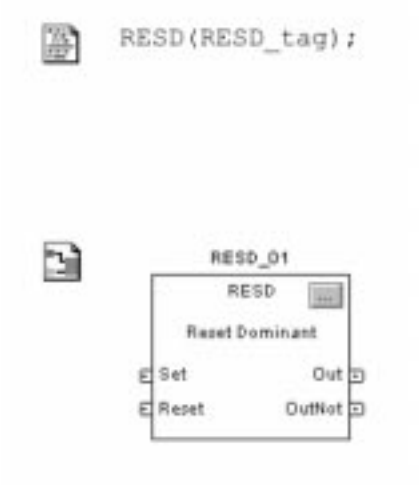
### 功能块



## 复位优先(RESD)

RESD指令使用Set和Reset输入来控制锁存输出。Reset输入优先级高于Set输入。

**操作数：**




### 结构文本

| 操作数： | 类型： | 格式： | 说明： |
|---|---|---|---|
| RESD标签 | DOMINANT_RESET | 结构体 | RESD结构体 |

### 功能块

| 操作数： | 类型： | 格式： | 说明： |
|---|---|---|---|
| RESD标签 | DOMINANT_RESET | 结构体 | RESD结构体 |

### DOMINANT_RESET结构体

| 输入参数： | 数据类型： | 说明： |
|---|---|---|
| EnableIn | BOOL | 功能块：<br>如果该位清零，则指令不执行输出不更新。<br>如果置位，指令执行。<br>缺省状态为置位<br>结构文本：<br>无影响。指令一直执行。 |
| Set | BOOL | 指令的Set输入。<br>缺省状态为清零 |
| Reset | BOOL | 指令的Reset输入。<br>缺省状态为清零 |
| **输出参数：** | **数据类型：** | **说明：** |
| EnableOut | BOOL | 输出使能。 |
| Out | BOOL | 指令输出 |
| OutNot | BOOL | 指令输出值取反 |

**说明：**下面的图表说明了RESD指令是如何操作的。




**算术状态标志：**无影响

**故障条件：**无

### 执行：

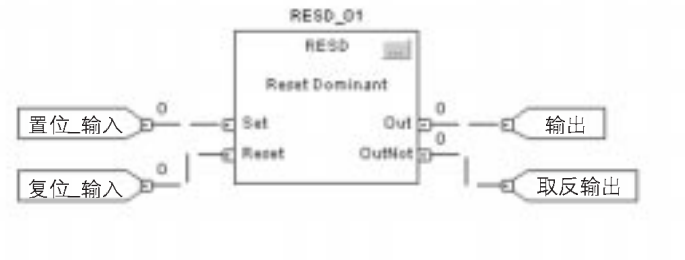
| 条件： | 功能块动作： | 结构文本动作： |
|---|---|---|
| 预扫描 | 无动作。 | 无动作。 |
| 指令首次扫描 | 无动作。 | 无动作。 |
| 指令首次执行 | 输出Out清零。<br>OutNot置位。 | 输出Out清零。<br>OutNot置位。 |
| 输入使能(EnableIn)清零 | 输出使能(EnableOut) 清零。 | 不影响 |
| 输入使能(EnableIn)置位 | 执行指令。<br>输出使能(EnableOut)置位。 | 输入使能(EnableIn)一直置位。<br>指令执行。 |
| 后期扫描 | 无动作 | 无动作 |

**实例：**当Set置位，Out置位；当Reset置位，Out清零。Reset的优先级高于Set。

### 结构文本

```
RESD_01.Set := set_input;
RESD_01.Reset := reset_input;
RESD(RESD_01);
output := RESD_01.Out;
not_output := RESD_01.OutNot;
```

### 功能块



## 置位优先(SETD)

SETD指令使用Set输入和Reset输入来控制输出锁存。Set输入优先级高于Reset输入。

**操作数：**



```
SETD(SETD_tag);
```

### 结构文本

| 操作数： | 类型： | 格式： | 说明： |
|---|---|---|---|
| SETD标签 | DOMINANT_SET | 结构体 | SETD结构体 |






### 功能块

| 操作数： | 类型： | 格式： | 说明： |
|---|---|---|---|
| SETD标签 | DOMINANT_SET | 结构体 | SETD结构体 |

### DOMINANT_SET结构体

| 输入参数： | 数据类型： | 说明： |
|---|---|---|
| EnableIn | BOOL | **功能块：**<br>如果该位清零，则指令不执行且输出不更新。<br>如果置位，指令执行。<br>缺省状态为置位<br>**结构文本：**<br>无影响。指令一直执行 |
| Set | BOOL | 指令Set输入。<br>缺省状态为清零 |
| Reset | BOOL | 指令Reset输入。<br>缺省状态为清零 |
| **输出参数：** | **数据类型：** | **说明：** |
| EnableOut | BOOL | 输出使能。 |
| Out | BOOL | 指令输出 |
| OutNot | BOOL | 指令输出值取反 |

**说明：**下面的图表说明了SETD指令是如何操作的。




**算术状态标志：**无影响

**故障条件：**无

### 执行：

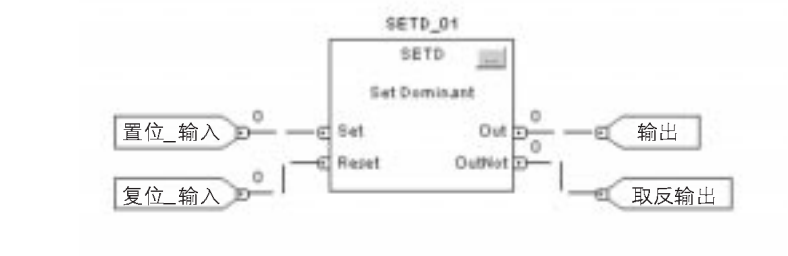
| 条件： | 功能块动作： | 结构文本动作： |
|---|---|---|
| 预扫描 | 无动作 | 无动作 |
| 指令首次扫描 | 无动作 | 无动作 |
| 指令首次执行 | Out 置位。<br>OutNot清零。 | Out 置位。<br>OutNot清零。 |
| 输入使能(EnableIn)清零 | 输出使能(EnableOut) 清零，指令不执行且输出不更新。 | 无影响 |
| 输入使能(EnableIn)置位 | 执行指令。<br>输出使能(EnableOut) 置位 | 输入使能(EnableIn)一直置位。<br>指令执行。 |
| 后期扫描 | 无动作 | 无动作 |

**实例：** Set置位时，Out为置位状态。Reset置位，Out为清零状态。Set输入的优先级高于Reset输入。

### 结构文本

```
SETD_01.Set := set_input;
SETD_01.Reset := reset_input;
SETD(SETD_01);
output := SETD_01.Out;
not_output := SETD_01.OutNot;
```

### 功能块



注释：

# 功能块属性

## 概 述

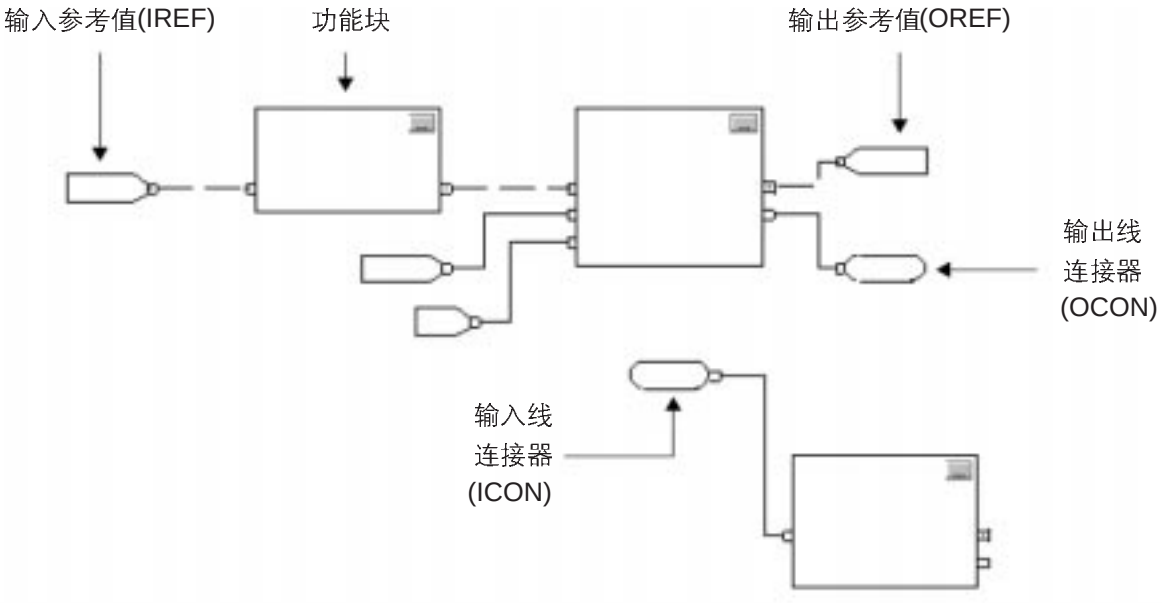
本附录描述了功能块指令的特点。阅读这部分内容有助于用户理解功能块例程如何执行。

**重　　要**

在功能块编程时，限定工程单位在+/-10 +/-15范围内，这是因为内部浮点运算采用的是单精度浮点数。若工程单位超出该范围，则当计算结果接近单精度浮点数的极限值(+/-10 +/-38)，可能导致精度降低。

## 选择功能块单元

使用以下单元实现对设备的控制：



使用下表选择用户的功能块单元：

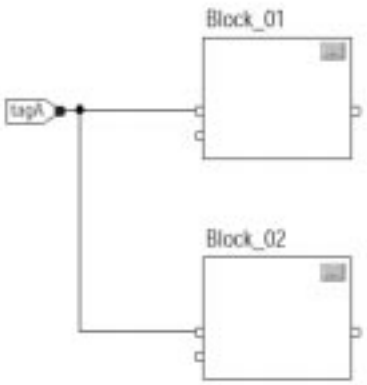
| 如果用户考虑： | 则选用： |
|---|---|
| 由输入设备或标签提供数据 | 输入参考(IREF) |
| 向输出设备或标签发送数据 | 输出参考(OREF) |
| 对一个或多个输入执行操作并产生一个或多个输出数据 | 功能块 |
| 在两个功能块间传送数据，当它们：<br>• 位于同一表内但距离较远<br>• 位于同一例程不同表内 | 输出线连接器(OCON)和输入线连接器(ICON) |
| 分散数据到例程中多个点 | 单个输出线连接器(OCON)和多个输入线连接器(ICON) |

## 数据锁存

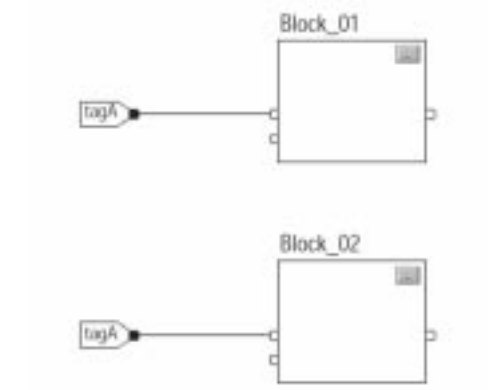
如果用户使用IREF指定功能块指令的输入数据，则在功能块例程扫描期间，IREF内的数据被锁存。IREF锁存来自程序-范围和控制器-范围的标签数据。控制器在每次扫描开始时更新所有的IREF数据。



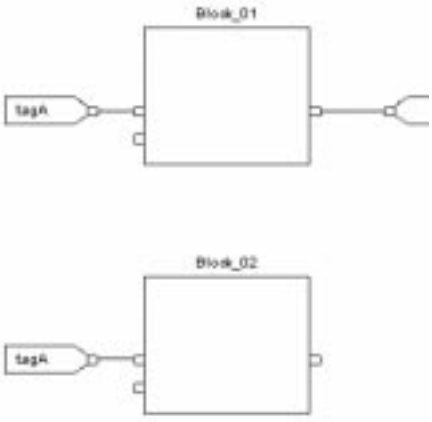

本例中，tagA的数值在开始执行例程时被存储。当Block_01执行时使用该存储值。当Block_02执行时使用同一存储值。如果在程序执行期间tagA存储值改变，在IREF内的tagA的存储值直到下一次执行程序时才改变。



本例与上例相同。tagA的值只有在例程开始执行时才被存储。在整个例程内都使用该存储值。




RSLogix5000版本11开始，用户可以在一个例程中多个IREF和一个OREF中使用同一标签。IREF中标签的数值在程序扫描过程中被锁定，因此在例程执行过程中，即使OREF中有多个标签值，所有的IREF仍为相同值。该例中，如果例程开始执行扫描时，tagA值为25.4，Block_01将tagA 的数值变为50.9，但Block_02执行此次扫描时，连接到Block_02的第二个IREF仍然使用25.4。直至下次扫描开始，tagA的新值50.9才能被多个IREF使用。



## 执行顺序

当用户进行如下操作时，RSLogix 5000编程软件自动确定例程内功能块的执行顺序：

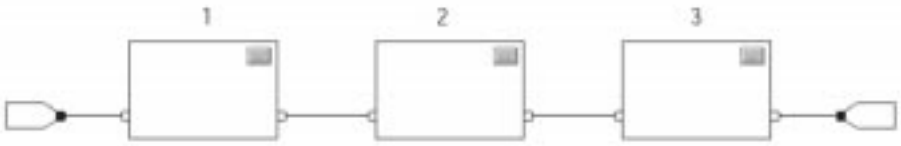
- 校验功能块例程
- 校验含有功能块例程的项目
- 下载含有功能块例程的项目

如有必要，用户可通过连接功能块并指明反馈线数据流来指定执行顺序。

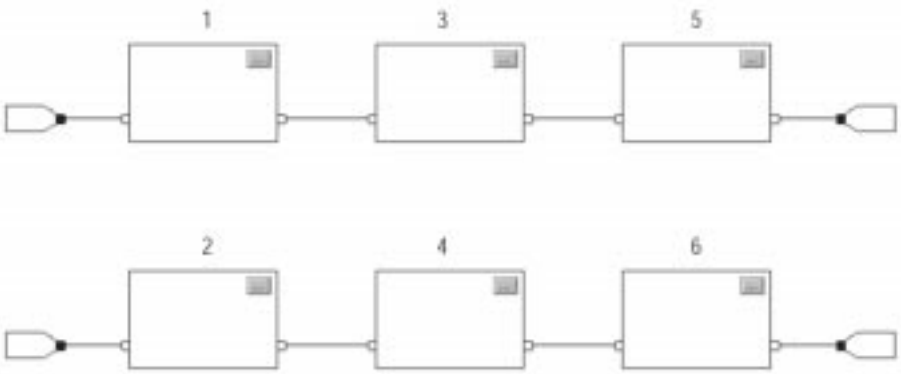
如果功能块没有连接在一起，哪个功能块首先执行没有影响。此时，功能块间无数据流。



如果用户顺序连接功能块，则执行顺序为从输入到输出。功能块的输入必须在控制器执行该功能块前可用。例如，功能块2必须在功能块3前执行，因为功能块2的输出是功能块3的输入。

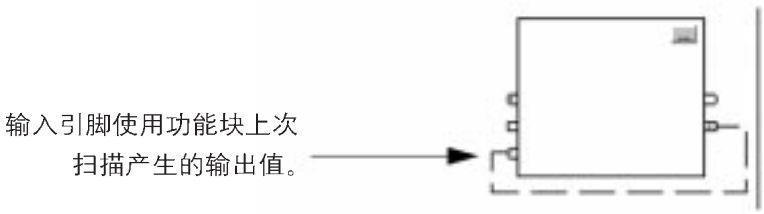



执行顺序只与连接的功能块相关。下例中的两个功能块组没有连接在一起。在特定组内的功能块执行顺序只与该组内的功能块有关。

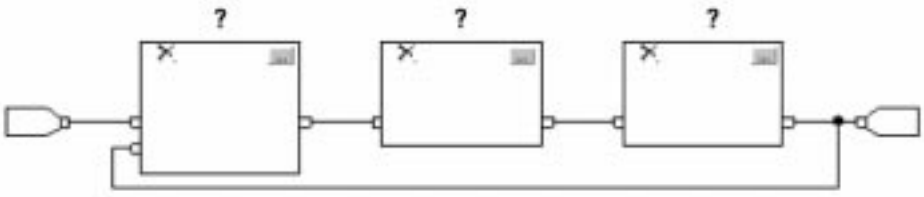


## 解析回路

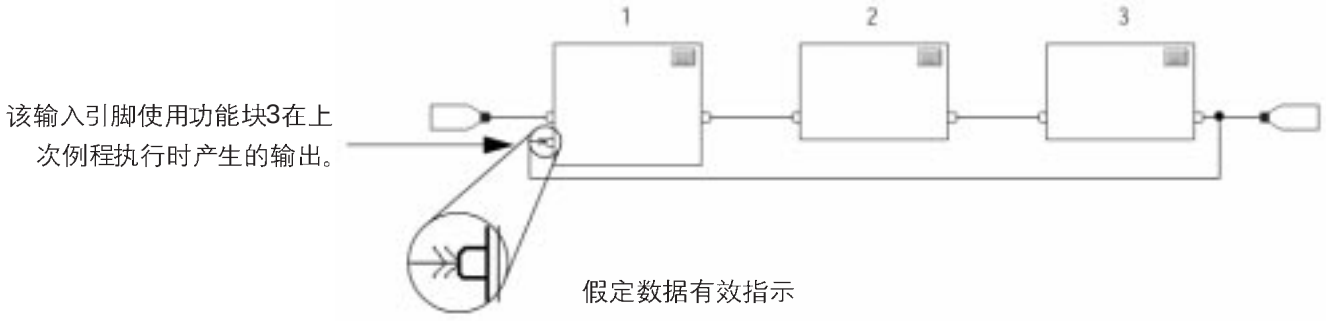
创建功能块反馈回路需将块输出引脚连接到同一功能块的输入引脚上。下图中已经实现。该回路仅包含单个功能块，因此不受执行顺序的影响。




如果一组功能块在同一闭环内，则控制器无法确定哪个功能块最先执行。也就是说，它无法解析回路。

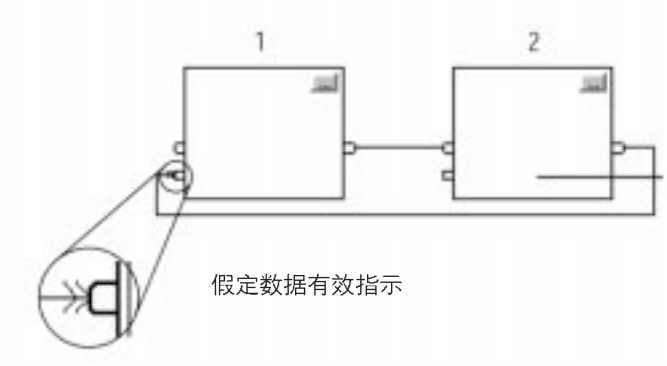
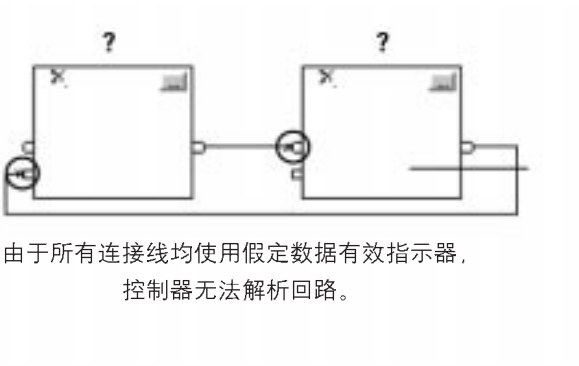



为确定功能块的执行顺序，可使用假定数据有效指示来标记用于创建回路(反馈线)的输入线。本例中，功能块1使用功能块3在上次例程执行时产生的输出。



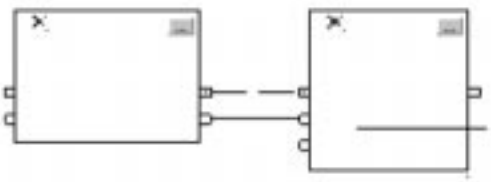
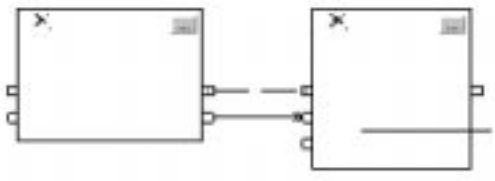
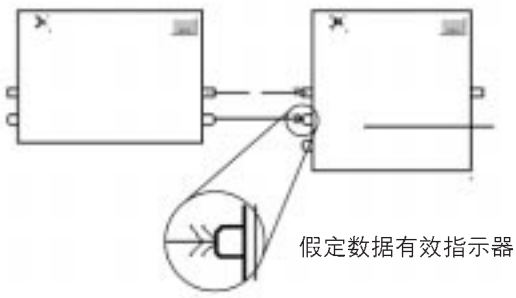

假定数据有效指示确定了回路内数据流向。箭头指示了数据输入到回路的首个功能块。

切忌使用假定数据有效指示来标记回路中所有连接线。

| 正确接法 | 错误接法 |
| --- | --- |
|  |  |
| 假定数据有效指示确定了回路内的数据流。 | 由于所有连接线均使用假定数据有效指示器，控制器无法解析回路。 |

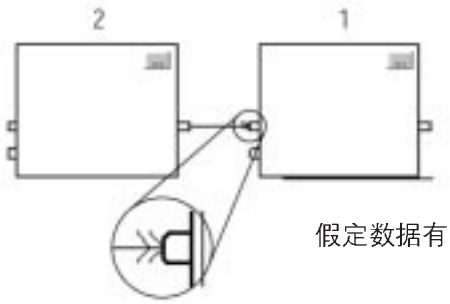
## 解析两功能块间数据流

如果用户使用两条或更多连接线连接两个功能块，则所有连接线需使用相同的数据流指示。

| 正确接法 | 错误接法 |
| --- | --- |
|  |  |
| 所有连接线均不使用假定数据有效指示。 | 一条连接线使用假定数据有效指示，而另一条不使用。 |
|  | |
| 两条连接线均使用假定数据有效指示器。 | |

### 创建一个扫描周期的延迟

使用假定数据有效指示在两个功能块间产生一个扫描周期延迟。在下例中，功能块1首先执行。它使用例程上次扫描时功能块2的输出数据。




假定数据有效指示

### 总结

总之，功能块程序按以下顺序执行：

1．控制器锁存多个IREF内的所有数据。

2．控制器按顺序(由功能块接线方式决定)执行其他功能块。

3．控制器将输出写入所有OREF。

## 功能块对溢出条件的响应

通常，发生溢出时功能块指令会保持历史记录而不使用±NAN或 ±INF值来更新记录。每条指令都有相对于溢出条件的响应：

| 响应1： | | 响应2： | 响应3： | |
|---|---|---|---|---|
| 功能块执行其算法并检查结果是否为±NAN或±INF。如果是±NAN或±INF，则功能块输出±NAN或±INF。 | | 带输出极限的功能块执行算法并检查结果是否为±NAN或±INF。输出极限通过HighLimit(上限)和LowLimit(下限)输入参数来定义。如果是±INF，则功能块输出一极限结果。如果是±NAN，则不使用输出极限，功能块输出±NAN。 | 不接受溢出条件。这些指令通常具有布尔型输出。 | |
| ALM | NTCH | HLL | BAND | OSRI |
| DEDT | PMUL | INTG | BNOT | RESD |
| DERV | POSP | PI | BOR | RTOR |
| ESEL | RLIM | PIDE | BXOR | SETD |
| FGEN | RMPS | SCL | CUTD | TOFR |
| HPF | SCRV | SOC | D2SD | TONR |
| LDL2 | SEL | | D3SD | |
| LDLG | SNEG | | DFF | |
| LPF | SRTP | | JKFF | |
| MAVE | SSUM | | OSFI | |
| MAXC | TOT | | | |
| MINC | UPDN | | | |
| MSTD | | | | |
| MUX | | | | |

## 定时方式

以下过程控制和驱动指令支持不同的定时方式。

| | | |
|---|---|---|
| DEDT | LDLG | RLIM |
| DERV | LPF | SCRV |
| HPF | NTCH | SOC |
| INTG | PI | TOT |
| LDL2 | PIDE | |

有三种不同的定时方式：

<table>
<tr><th>定时方式：</th><th>说明：</th></tr>
<tr><td>周期</td><td>周期方式是缺省方式，适用于大多数的控制应用系统。如果用户例程在周期性任务中执行，我们推荐其中指令采用此方式。指令使用的时间增量(DeltaT)由下列条件确定：
<table>
<tr><th>如果指令执行在：</th><th>那么DeltaT等于：</th></tr>
<tr><td>周期任务</td><td>任务的执行周期<br><br>设置DeltaT为任务执行周期的整数部分，单位为毫秒。控制器截掉任务执行周期的小数部分。例如，任务执行周期为10.5ms，控制器设置DeltaT等于10ms。<br><br>如果用户需要使用任务执行周期的小数，选用过采样方式。用户可以在过采样方式下设置OversampleDT参数等于任务执行周期，包含任一小数。</td></tr>
<tr><td>事件或连续任务</td><td>自上次任务执行后的运行时间<br><br>控制器截取运行时间的整数部分(毫秒)。例如，如果运行时间等于10.5ms，控制器设置DeltaT等于10ms。</td></tr>
<tr><td colspan="2">过程输入的更新需要与任务执行同步，或采样速度比任务执行快5-10倍，以达到输入与指令之间的采样误差最小。</td></tr>
</table></td></tr>
<tr><td>过采样</td><td>在过采样方式下，指令使用的时间增量(DeltaT)为写入该指令OversampleDT参数内的值。在事件或连续任务内执行该指令，且过程量输入不具有与其更新相关联的时间戳时使用这种方式。如果过程量输入具有时间戳值，则使用实时采样方式。<br><br>在用户程序中增加逻辑来控制何时执行该指令。例如，用户使用定时器设置OversampleDeltaT值，通过EnableIn输入来控制指令执行。<br><br>为了使输入与指令之间的采样误差最小，过程输入需要的采样时间比指令的执行时间快5-10倍。</td></tr>
</table>

| 定时方式： | 说明： |
| --- | --- |
| 实时采样 | 在实时采样方式下，指令使用的时间增量(DeltaT)是两个对应的过程输入的更新时间戳的差值。在事件或连续任务内执行该指令，且工程输入具有与其更新相关联的时间戳值时使用这种方式。<br><br>时间戳值读自指令的RTSTimeStamp参数中输入的标签名。通常该标签名是与过程输入相关联的输入模块上的一个参数。<br><br>指令将组态的RTSTime值(预定更新周期)与计算的DeltaT进行比较，从而确定指令是否读取过程输入的每次更新值。如果DeltaT与组态时间相差超过1毫秒，则指令置位RTSMissed状态位来表示读取模块输入更新存在问题。 |

基于时间的指令要求DeltaT为常数，这是为了使控制算法能够正确计算过程输出。如果DeltaT变化，则过程输出的出现不连续。不连续的严重程度取决于指令和DeltaT的变化范围。在下述情况下发生输出不连续：

- 该指令在扫描期间不执行。
- 该指令在任务期间多次执行。
- 任务运行时任务扫描速率或者过程输入的采样时间改变。
- 运行任务时用户改变时间基方式。
- 任务运行时过滤器功能块上的阶数参数改变。阶数参数的改变将在指令内选择不同的控制算法。

## 用于定时方式的通用指令参数

支持时间基方式的指令具有下述输入与输出参数：

*输入参数*

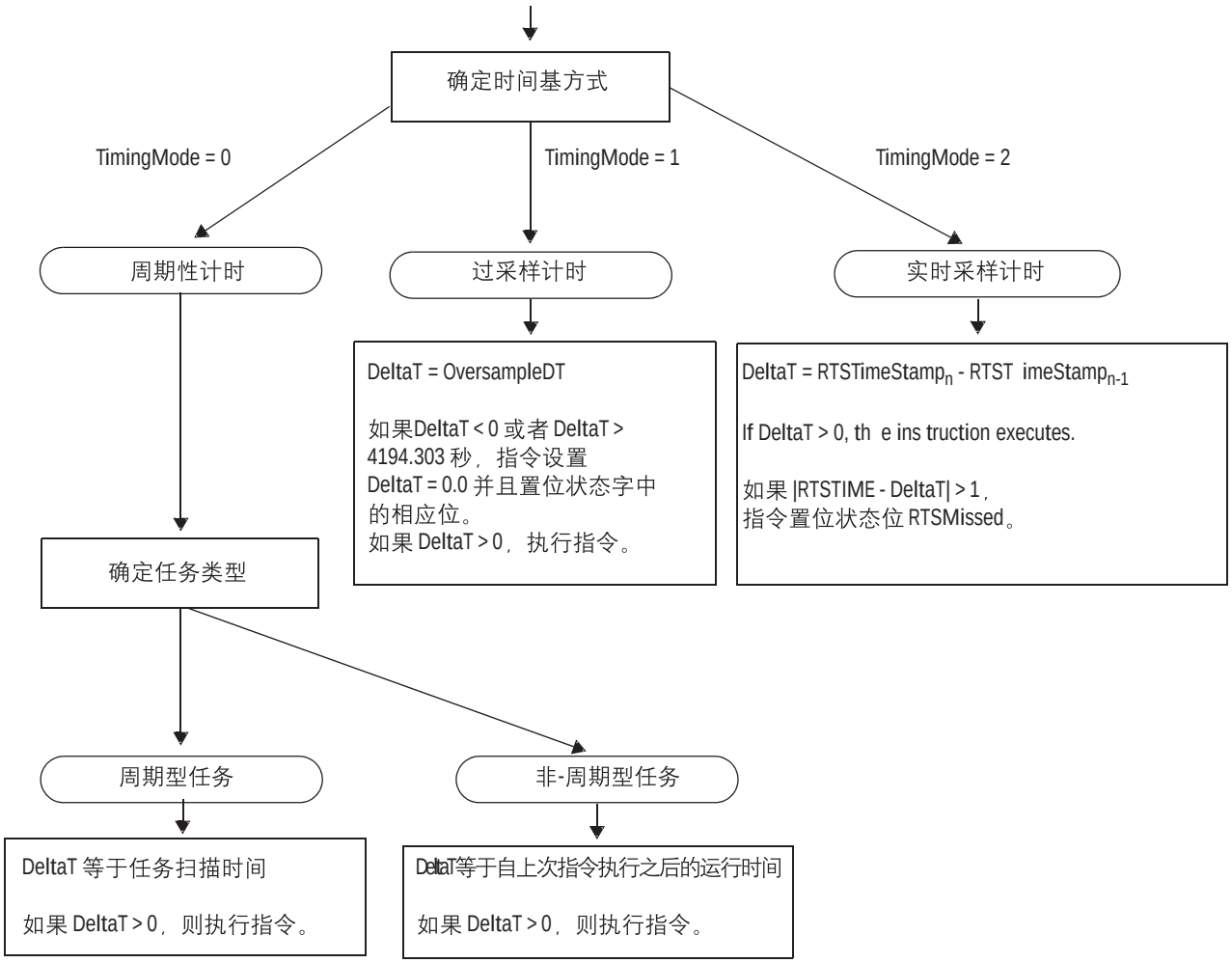
| 输入参数： | 数据类型： | 说明： |
|---|---|---|
| TimingMode | DINT | 选择定时执行方式。<br>**数值：** **说明：**<br>0 周期方式<br>1 过采样方式<br>2 实时采样方式<br><br>有效值 = 0到2<br>缺省值 = 0<br><br>当TimingMode = 0且为周期性任务时，周期定时使能，DeltaT设置为任务扫描速率。<br>当TimingMode = 0并且为事件或连续性任务时，周期定时使能，DeltaT等于自上次指令执行之后的运行时间。<br><br>当TimingMode = 1时，过采样定时使能，DeltaT设置为OversampleDT参数值。<br><br>当TimingMode = 2时，实时采样定时使能，DeltaT为当前与上一次时间戳值(读自与输入相关联的模块)之差。<br><br>如果TimingMode无效，指令设置状态字中相应位。 |
| OversampleDT | REAL | 用于过采样方式的执行时间。DeltaT以秒为单位。如果TimingMode = 1，则OversampleDT = 0.0将禁止控制算法的执行。如果无效，指令设置DeltaT = 0.0并且置位状态字中的相应位。<br>有效值 = 0 到4194.303秒<br>缺省值 = 0.0 |
| RTSTime | DINT | 实时采样定时方式的模块更新周期。期望的DeltaT更新周期以毫秒为单位。更新周期通常为用于配置模块更新时间的值。如果无效，指令置位状态字中的相应位并禁止RTSMissed的检查。<br>有效值 = 1 到32,767 ms<br>缺省值 = 1 |
| RTSTimeStamp | DINT | 实时采样定时的模块时间戳。时间戳对应于输入信号的上次更新。该值用于计算DeltaT。如果无效，指令置位状态字中的相应位并禁止控制算法的执行以及RTSMissed的检测。<br>有效值 = 1 到32,767 ms(覆盖从32767到0)<br>1个计数= 1毫秒<br>缺省值 = 0 |

*输出参数*

| 输出参数： | 数据类型： | 说明： |
|---|---|---|
| DeltaT | REAL | 两次更新之间的运行时间，以秒为单位。控制算法用该时间值来计算过程输出。<br><br>周期方式：DeltaT为任务扫描速率(如果是周期任务)，DeltaT为自上次指令执行之后的运行时间(如果是事件或连续任务)<br><br>过采样方式：DeltaT = OversampleDT<br><br>实时采样方式：DeltaT = (RTSTimeStamp n - RTSTimeStamp n-1 ) |
| Status | DINT | 功能块的状态字。 |
| TimingModeInv (Status.27) | BOOL | TimingMode值无效。 |
| RTSMissed (Status.28) | BOOL | 只用于实时采样方式。当ABS \| DeltaT - RTSTime \| > 1 (.001 second)时置位。 |
| RTSTimeInv (Status.29) | BOOL | RTSTime值无效。 |
| RTSTimeStampInv (Status.30) | BOOL | RTSTimeStamp值无效。 |
| DeltaTInv (Status.31) | BOOL | DeltaT值无效。 |

## 定时方式概述

下面的图表给出一条指令确定合适的定时方式的方法。



## 程序/操作员控制

一些指令支持程序/操作员(Program/Operator)控制。这些指令有：

- 增强选择 (ESEL)
- 累加器 (TOT)
- 增强型PID (PIDE)
- 斜坡/渗透 (RMPS)
- 离散型2-态设备 (D2SD)
- 离散型3-态设备(D3SD)

程序/操作员控制允许用户由程序和操作员界面设备同时控制这些指令。在程序控制方式下，指令由程序输入控制；在操作员控制方式下，指令由操作员输入控制。

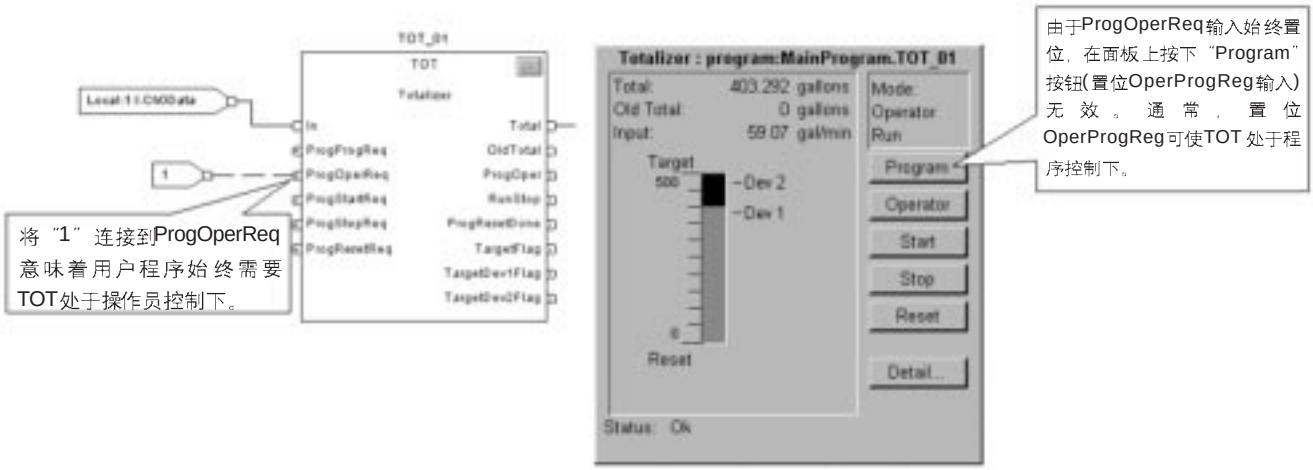
通过以下输入状态确定由程序还是操作员来控制：

| 输入： | 说明： |
|---|---|
| .ProgProgReq | 程序请求进入程序控制方式。 |
| .ProgOperReq | 程序请求进入操作员控制方式。 |
| .OperProgReq | 操作员请求进入程序控制方式。 |
| .OperOperReq | 操作员请求进入操作员控制方式。 |

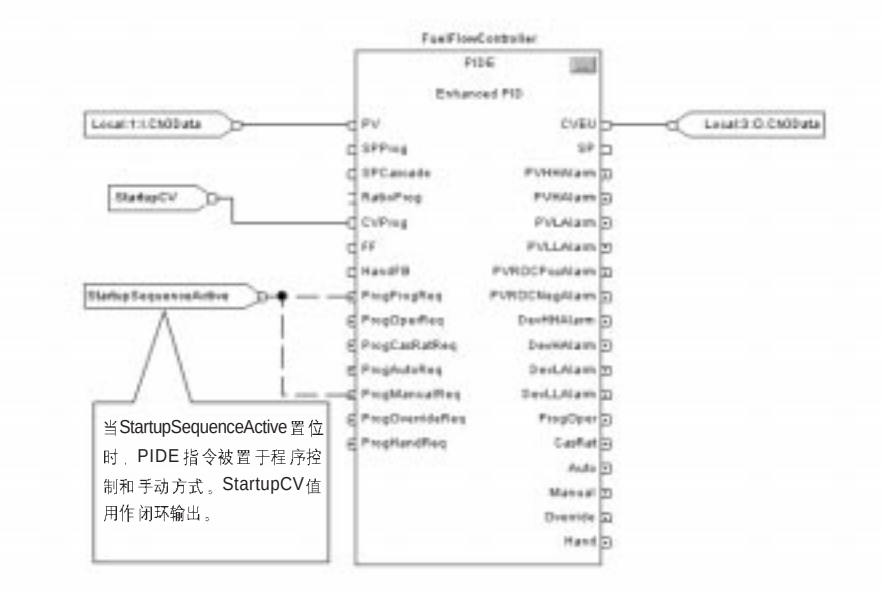
为了确定指令处于程序或操作员控制方式，可检查ProgOper输出。如果ProgOper置位，则该指令处于程序控制方式；如果ProgOper清零，则该指令处于操作员控制方式。

如果两个输入请求位都置位，则操作员控制优先于程序控制。例如，如果ProgProgReq 和ProgOperReq都置位，则指令进入操作员控制。

程序请求输入优先于操作员请求输入。因此可使用ProgProgReq和ProgOperReq输入“锁定”指令在期望的控制方式下。例如，假定一条Totalizer指令始终处于操作员控制下，并且用户程序始终不能控制该累加器的运行或停止；这种情况下，用户可以将数值1连接到ProgOperReq。这就防止了通过操作员界面设备置位OperProgReq将累加器置于程序控制方式。



同样，始终置位ProgProgReq能够“锁定”指令处于程序控制。这在自动设置起动顺序时很有用，由程序来控制指令的操作不用担心操作员疏忽对指令进行控制。例如，在起动期间由程序置位ProgProgReq输入，然后在起动完成时清零ProgProgReq输入。一旦ProgProgReq输入清零，指令保持为程序控制直到接到改变的请求。例如，操作员可以从面板置位OperOperReq输入，从而接管该指令的控制。下例给出如何锁定指令处于程序控制。




当指令执行时，该指令的操作员请求输入始终被清零。这样就通过设置期望方式对应的请求位来实现操作员界面对指令的操作。用户不必编程操作员界面复位请求位。例如，若操作员界面置位PIDE指令的OperAutoReq输入，当PIDE指令执行时，它决定适当的响应并清零OperAutoReq。

程序请求输入通常不由指令清零，因为它们通常连接到指令的输入。如果指令清零这些输入，那么输入刚好再次被接入的输入置位。用户可能会遇到这种情形：在希望程序请求被本指令清零同时，还希望使用其它逻辑置位程序请求。在这种情况下，用户可以置位ProgValueReset输入，且在指令执行时，指令始终清零程序方式请求输入。

本例中，另一梯形图例程中的一条梯级用于在按下按钮时单步锁存PIDE指令的ProgAutoReq。由于PIDE指令自动清零程序方式请求，用户不必编程梯形图逻辑在程序执行完后清零ProgAutoReq，并且每次按下按钮时PIDE指令只收到一个转到Auto的请求。

当按下TIC101AutoReq按钮时，单步锁存PIDE指令TIC101的ProgAutoReq。TIC101已被组态为ProgValueReset输入置位，所以PIDE指令执行时，指令自动清零ProgAutoReq。



注释：

# 结构文本编程

## 概述

本附录描述了结构文本编程的特点。阅读这部分内容有助于用户理解结构文本程序如何执行。

| 关于： | 查阅： |
|---|---|
| 结构文本语法 | B-1 |
| 赋值 | B-3 |
| 表达式 | B-5 |
| 指令 | B-12 |
| 结构 | B-13 |
| 注释 | B-28 |

## 结构文本语法

结构文本是一种使用语句来定义执行动作的文本化编程语言。结构文本不区分字母大小写。它包含以下要素：

| 术语： | 定义： | | 实例： |
|---|---|---|---|
| 赋值<br>(查阅页B-3) | 使用赋值语句指定标签的数值。<br>赋值运算符是:= 。<br>赋值语句以半角;结束。 | | tag:=expression; |
| expression<br>(查阅页B-5) | 表达式是完整的赋值或结构语句的一部分。<br>表达式计算出数值(数值表达式)或真假状态(布尔表达式)。<br>一个表达式包括： | | value1<br>4<br>tag1+tag2 |
| | 标签 | 用于存储数据的命名内存区域<br>(BOOL,SINT,INT,DINT,REAL,string) | tag1>=value1<br>function(tag1) |
| | 立即数 | 常数 | |
| | 运算符 | 在表达式中特指一定操作的符号或助记符 | |
| | 功能 | 一个功能执行后产生一个数值，括号内是其操作数。<br><br>功能与指令的语法相似，区别在于功能仅能用在表达式中，而指令不能用在表达式中。 | |

| 术语： | 定义： | 实例： |
| --- | --- | --- |
| 指令<br>(查阅页B-12) | 每条指令是独立的语句。<br>每条指令使用括号包括其操作数。<br>根据指令不同，数可以为0、1或多个操作数。<br>当指令执行时，产生数据结构的部分值。<br>赋值语句以半角;结束。<br><br>指令与功能的语法相似，区别在于指令不能用在表达式中。<br>而功能仅用在表达式中。 | instruction();<br>instruction(operand);<br>instruction(operand1,<br>operand2,operand3); |
| 结构<br>(查阅页B-13) | 用于触发结构文本代码的条件语句(例如，other语句)。<br>赋值语句以半角;结束。 | IF…THEN<br>CASE<br>FOR…DO<br>WHILE…DO<br>REPEAT…UNTIL<br>EXIT |
| 注释<br>(查阅页B28) | 用于解释或阐明一段结构文本代码功能的文本。<br>• 使用注释提高结构文本程序的可读性<br>• 注释不影响结构文本执行<br>• 注释可以写在结构文本程序的任意位置 | //comment<br>(*start of comment… end of comment*)<br>/*start of comment…end of comment*/ |

在结构文本编程时可选键入空格。空格对结构文本程序执行没有影响。例如，以下两个语句相同：

```
Tag_B:=Tag_A

Tag_B := Tag_A
```

## 赋值语句

使用赋值语句改变标签内存储的数值。赋值语句的语法是：

tag:=expression;

| 元素： | 描述： | |
|---|---|---|
| tag | 取得新值的标签名称<br>其数据类型必须为BOOL,SINT,INT,DINT或REAL | |
| := | 赋值符号 | |
| expression | 赋给标签的新值 | |
| | **如果tag的数据类型为：** | **使用以下类型表达式：** |
| | BOOL | 布尔表达式 |
| | SINT<br>INT<br>DINT<br>REAL | 数值表达式 |
| ; | 语句结束符 | |

该标签将数值保持至其他赋值语句改变该值。

表达式可以很简单，如立即数或其他标签名；也可以很复杂，包括多个运算符和/或功能。详细信息请查阅页B-5 ″表达式″一节。

## 指定非保持赋值语句

非保持型赋值语句与上面所描述的赋值语句不同，每次控制器进入下的状态，非保持型赋值语句中的标签复位为0：

- 进入RUN模式
- 如果用户组态SFC为自动复位，离开SFC的步时(仅适用于用户在步动作间嵌入赋值语句或使用JSR指令调用结构文本例程的情况)

一个非保持型赋值语句的语法：

tag[:=]expression;

其中：

| 元素： | 描述： | |
|---|---|---|
| tag | 取得新值的标签名称<br>其数据类型必须为BOOL,SINT,INT,DINT或REAL | |
| [:=] | 非保持型赋值符号 | |
| expression | 赋给标签的新值 | |
| | **如果tag的数据类型为：** | **使用以下类型表达式：** |
| | BOOL | 布尔表达式 |
| | SINT<br>INT<br>DINT<br>REAL | 数值表达式 |
| ; | 语句结束符 | |

## 为字符串赋值ASCII码

通过赋值运算符可将ASCII码赋给字符串标签中DATA成员的元素。通过指定字符值或指定标签名称、DATA成员和字符元素来为字符串赋值。例如：

| 正确语法： | 错误语法： |
|---|---|
| string1.DATA[0]:= 65; | string1.DATA[0] := A; |
| string1.DATA[0]:=string2.DATA[0]; | string1 := string2; |

ASCII码字符串指令用于增加或插入ASCII码字符串。

| 目的： | 所用指令： |
|---|---|
| 在字符串末尾添加字符 | CONCAT |
| 在字符串中插入字符 | INSERT |

表达式

一个表达式包括标签名、相等或比较关系。使用以下元素写出表达式：

- 存储数值的标签名(变量)
- 用户直接输入到表达式的数值(立即数)
- 功能，如：ABS,TRUNC
- 运算符，如：+,-,<,>,And,Or

用户写表达式时必须遵守的规则：

- 表达式中可混合使用大小写字母。例如，三种变形的“AND”：AND，And和and均可使用。
- 对于较复杂的要求，可使用括号将多个表达式集合到一个表达式内。这样可以提高整个表达式的可读性，并使其按照期望的顺序执行。查阅页B11“确定执行顺序”一节。

在结构文本中，用户可使用以下两种类型的表达式：

**布尔表达式**：产生1(真)或0(假)布尔值的表达式。

- 布尔表达式中使用布尔标签，关系运算符和逻辑运算符来比较或检查条件是否为真或假。例如，tag1>65。
- 简单的布尔表达式可以是单独BOOL标签。
- 用户使用布尔表达式作为其他逻辑执行的条件。

数值表达式：计算出整数或浮点数值的表达式。

- 数值表达式使用算术运算符、算术功能和位运算符。例如，tag1+5。
- 通常，用户会将数值表达式嵌套到布尔表达式中。例如，(tag1+5)>65。

使用下表为用户表达式选择运算符：

| 如果用户希望： | 那么： |
|---|---|
| 计算算术值 | 使用算术运算符和功能，页B-7。 |
| 比较两数值或字符串 | 使用关系运算符，页B-8。 |
| 检查条件为真或假 | 使用逻辑运算符，页B-10。 |
| 比较数值内各位 | 使用位运算符，页B-11。 |

## 使用算术运算符和功能

用户可以在同一算术表达式内使用多个运算符和功能。

使用下列算术运算符计算数值：

| 目的： | 运算符： | 最优数据类型： |
|---|---|---|
| 加 | + | DINT，REAL |
| 减 / 非 | - | DINT，REAL |
| 乘 | * | DINT，REAL |
| 指数(x是y的幂) | ** | DINT，REAL |
| 除 | / | DINT，REAL |
| 模数-除 | MOD | DINT，REAL |

算术功能执行数学操作。在功能中需指定常数、非布尔型标签或表达式。

| 目的： | 运算符： | 最优数据类型： |
|---|---|---|
| 绝对值 | ABS(numeric_expression) | DINT，REAL |
| 反余弦 | ACOS(numeric_expression) | REAL |
| 反正弦 | ASIN(numeric_expression) | REAL |
| 反正切 | ATAN(numeric_expression) | REAL |
| 余弦 | COS(numeric_expression) | REAL |
| 弧度转换为角度 | DEG(numeric_expression) | DINT，REAL |
| 自然对数 | LN(numeric_expression) | REAL |
| 以10为底的对数 | LOG(numeric_expression) | REAL |
| 角度转换成弧度 | RAD(numeric_expression) | DINT，REAL |
| 正弦 | SIN(numeric_expression) | REAL |
| 平方根 | SQR(numeric_expression) | DINT，REAL |
| 正切 | TAN(numeric_expression) | REAL |
| 截断 | TRN(numeric_expression) | DINT，REAL |

例如：

| 使用该格式： | 实例：<br>此情形下： | <br>用户需编程： |
|---|---|---|
| value1 operator value2 | 如果gain_4和gain_4_adj均为DINT型标签且要求：gain_4加上15并将结果存到gain_4_adj。 | gain_4_adj := gain_4+15; |
| operator value1 | 如果alarm和high_alarm均为DINT型标签且要求：将high_alarm取反并将结果存到alarm。 | alarm:= -high_alarm; |
| function(numeric_expression) | 如果overtravel和overtravel _POS均为DINT标签且要求计算overtravel的绝对值并将结果保存到overtravel_POS。 | overtravel_POS := ABS(overtravel); |
| value1 operator (function((value2+value3)/2) | 如果adjustment和position均为DINT型标签且sensor1和sensor2均为REAL型标签且要求：计算sensor1和sensor2平均值的绝对值再加上adjustment，并将结果保存到position。 | position := adjustment + ABS((sensor1 + sensor2)/2); |

## 使用关系运算符

关系运算符将两数值或字符串进行比较，产生一个真或假的结果。关系运算的结果为BOOL值：

| 比较关系为： | 结果： |
|---|---|
| 真 | 1 |
| 假 | 0 |

使用下例关系运算符：

| 比较关系： | 使用运算符： | 最优数据类型： |
|---|---|---|
| 等于 | = | DINT，REAL，string |
| 小于 | < | DINT，REAL，string |
| 小于或等于 | <= | DINT，REAL，string |
| 大于 | > | DINT，REAL，string |
| 大于或等于 | >= | DINT，REAL，string |
| 不等于 | <> | DINT，REAL，string |

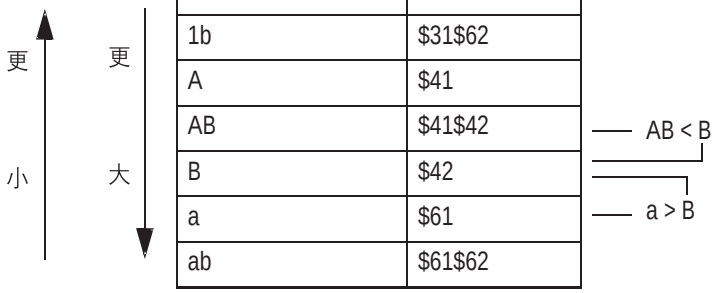
例如：

| 使用该格式： | 实例： | |
|---|---|---|
| | 此情形下： | 用户需编程： |
| value1 operator value2 | 如果temp是DINT标签且要求：如果temp小于100?，则…. | IF temp<100 THEN... |
| stringtag1 operator stringtag2 | 如果bar_code和dest均为字符串标签且要求：如果bar_code等于dest，那么… | IF bar_code=dest THEN... |
| char1 operator char2<br>将ASCII码对应的十进制数值直接输入表达式。 | 如果bar_code是字符串标签且要求：如果bar_code.DATA[0]等于′A′则 | IF bar_code.DATA[0]=65 THEN... |
| bool_tag := bool_expressions | 如果count和length均为DINT型标签，done是BOOL型标签，且要求：如果count大于或等于length，计算完成。 | done := (count >= length); |

*如何计算字符串*

ASCII字符对应的十六进制值决定了一个字符串小于或者大于另一个字符串。

- 两个字符串按照电话号码簿方式分类时，它们的大小由字符串的顺序决定。




| ASCII字符 | 十六进制代码 |
|---|---|
| 1ab | $31$61$62 |
| 1b | $31$62 |
| A | $41 |
| AB | $41$42 |
| B | $42 |
| a | $61 |
| ab | $61$62 |

- 如果字符匹配则字符串相等。
- ASCII字符区分大小写。大写 ″A″ ($41)不等于小写 ″a″ ($61)。

字符的十进制数值和十六进制代码，参见本手册的封底。

## 使用逻辑运算符

用户可使用逻辑运算符检查多个条件为真或假。逻辑操作的结果是BOOL值：

| 如果比较关系为： | 结果是： |
|---|---|
| 真 | 1 |
| 假 | 0 |

使用以下逻辑运算符：

| 目的： | 运算符： | 数据类型： |
|---|---|---|
| 按位与 | &,AND | BOOL |
| 按位非 | NOT | BOOL |
| 按位或 | OR | BOOL |
| 按位异或 | XOR | BOOL |

例如：

| 使用该格式： | 实例：<br>此情形下： | <br>用户需编程： |
|---|---|---|
| BOOLtag | 如果photoeye是BOOL型标签且要求：<br>如果photoeye_1为真，那么… | IF photoeye THEN... |
| NOT BOOLtag | 如果photoeye是BOOL型标签且要求：<br>如果photoeye_1为假，那么… | IF NOT photoeye THEN... |
| expression1 & expression2 | 如果photoeye是BOOL型标签,temp是DINT标签且要求：<br>如果photoeye是真且temp小于100?，那么… | IF photoeye & (temp<100) THEN... |
| expression1 OR expression2 | 如果photoeye是BOOL型标签, temp是DINT标签且要求：<br>如果photoeye是真或temp小于100?那么… | IF photoeye OR (temp<100) THEN... |
| expression1 XOR expression2 | 如果photoeye1和photoeye2是BOOL型标签且要求：<br>如果：<br>• photoeye1为真同时photoeye2为假<br>或<br>• photoeye1为假同时photoeye2为真那么… | IF photoeye1 XOR photoeye2 THEN... |
| BOOLtag := expression1 & expression2 | 如果photoeye1和photoeye2是BOOL型标签，open是BOOL型标签且要求：如果photoeye1和photoeye2同为真，置位open为真。 | open := photoeye1 & photoeye2; |

## 使用位运算符

位运算符基于两个数值处理一个数值中各位。

| 目的： | 所用运算符： | 最优数据类型： |
|---|---|---|
| 位与 | &,AND | DINT |
| 位或 | OR | DINT |
| 位异或 | XOR | DINT |
| 位补码 | NOT | DINT |

例如：

| 所用格式： | 实例： | |
|---|---|---|
| | 此情形下： | 用户需编程： |
| value1 operator value2 | 如果input1,input2和result1均为DINT型标签且要求：计算input1和input2的位结果，并将其保存到result1。 | result1 := input1 AND input2; |

## 确定运算顺序

指令按预先规定的顺序，而非一定是从左到右来执行写入表达式的操作。

- 同级运算的顺序是从左向右执行。
- 如果表达式中包含多个运算符或功能，可通过圆括号 "()" 将条件分组。这样可以确保执行顺序正确并使得表达式更具可读性。

| 顺序： | 运算： |
|---|---|
| 1 | ( ) |
| 2 | function(…) |
| 3 | ** |
| 4 | - (取反) |
| 5 | NOT |
| 6 | *，/，MOD |
| 7 | +，- (减) |
| 8 | <，<=，>，>=， |
| 9 | =,<> |
| 10 | &,AND |
| 11 | XOR |
| 12 | OR |

## 指令

结构文本编程语句也可以是指令。请查阅本手册开始的指令列表获取结构文本语言可用指令。每扫描一次结构文本程序，指令执行一遍。如果结构得条件为真，则结构中结构文本指令执行。如果结构条件为假，则结构中的语句不被扫描。此时没有梯级条件或状态转换触发指令执行。

与功能块的区别在于，功能块指令通过输入使能(EnableIn)触发其执行，而结构文本指令执行时相当于输入使能(EnableIn)一直置位。

与梯形图的区别在于，梯形图指令通过输入梯级条件触发执行，一些梯形图指令仅当输入梯级条件触发上升沿时执行。这些是边沿触发型梯形图指令。在结构文本程序中，除非用户预先规定结构文本指令的执行顺序，否则指令将会在其被扫描时执行。

例如，ABL指令是边沿触发梯形图指令。该例中，当tag_xic由清零到置位跃变一次，ABL指令执行一次。当tag_xic一直置位或清零时，ABL指令不执行。




在结构文本语言中，如果用户将该例写为：

```
IF tag_xic THEN ABL(0,serial_control);

END_IF;
```

那么，当tag_xic置位而不仅是从清零到置位跃变时， ABL指令每次扫描时均执行。

如果用户希望仅当tag_xic为上升沿时ABL指令执行，必须编写结构文本条件指令。使用上升沿指令触发执行。

```
osri_1.InputBit := tag_xic;
OSRI(osri_1);

IF (osri_1.OutputBit) THENABL(0,serial_control);
END_IF;
```

## 结构

结构可以单独或与其他结构嵌套编程。

| 如果用户希望： | 使用此结构： | 可用编程语言： | 查阅页码： |
|---|---|---|---|
| 当特定条件发生时，执行某项操作 | IF…THEN | 结构文本 | B-14 |
| 基于数值选择执行的操作 | CASE…OF | 结构文本 | B-17 |
| 根据指定的次数重复执行某项操作 | FOR…DO | 结构文本 | B-19 |
| 一旦特定条件为真，重复执行某项操作 | WHILE…DO | 结构文本 | B-22 |
| 直到条件为真，否则重复执行某项操作 | REPEAT…UNTIL | 结构文本 | B-25 |

## IF...THEN

当特定条件发生时，使用IF...THEN语句执行某项操作。

**操作数：**



```
IF bool_expression THEN
    <statement>;
END_IF;
```

**结构文本**

| 操作数： | 类型： | 格式： | 输入： |
|---|---|---|---|
| bool_expression | BOOL | 标签表达式 | 计算出布尔值(布尔表达式)的布尔型标签或表达式 |

**说明：**语法是：

```
IF bool_expression1 THEN
      <statement >;
                .
                .
                .
ELSIF bool_expression2 THEN
      <statement>;
                .
                .
                .
ELSE
      <statement>;
                .
                .
                .
END_IF;
```

当bool_expression1为真时，执行语句。

可选 — 当bool_expression2为真时，执行语句。

可选 — 当两个表达式均为假时，执行语句。

根据以下准则使用ELSIF或ELSE：

1.如果从多个语句组中选出执行条件，需添加一个或多个ELSIF语句。

- 每个ELSIF表示一个选择路径。
- 根据用户需要指定的ELSIF路径数。
- 控制器执行首个条件为真的IF或ELSIF语句，并跳过其余的ELSIFs和ELSE语句。

2、 如果要求当所有IF或ELSIF条件为假时执行语句，需添加一条ELSE语句。

下表总结了IF,WHEN,ELSIF和ELSE语句的多种组合。

| 如果用户要求： | 并且： | 那么使用此结构： |
|---|---|---|
| 当条件为真时，执行语句 | 如果条件为假，不执行语句 | IF…THEN |
| | 如果条件为假，执行其他操作 | IF…THEN…ESLE |
| 根据输入条件，从多个语句(语句组)中选择 | 如果条件为假，不执行语句 | IF…THEN…ELSIF |
| | 如果所有条件为假，分配缺省语句 | IF…THEN…ELSIF…ELSE |

### 实例1：IF...THEN

| 如果用户希望： | 输入结构文本程序： |
|---|---|
| 如果 rejects > 3 那么 | IF rejects > 3 THEN |
| conveyor = off (0) | conveyor := 0; |
| alarm = on (1) | alarm := 1; |
| | END_IF; |

### 实例2：IF...THEN...ELSE

| 如果用户希望： | 输入结构文本程序： |
|---|---|
| 如果传送带方向触点为前进 (1) 那么 | IF conveyor_direction THEN |
| 指示灯灭 | light := 0; |
| 否则 指示灯亮 | ELSE |
| | light [:=] 1; |
| | END_IF; |

当控制器进入以下状态，[:=]会将light清零：

- 进入RUN模式
- 如果用户组态SFC为自动复位，离开SFC的步时(仅适用于用户在步动作中嵌入赋值语句或使用JSR指令调用结构文本的例程的情况)

### 实例3：IF...THEN...ELSIF

| 如果用户希望： | 输入结构文本： |
| --- | --- |
| 如果糖料上下限位开关均导通，那么<br>进给阀打开(导通)<br>直至糖料上限位开关关闭 | IF Sugar.Low & Sugar.High THEN<br>Sugar.Inlet [:=] 1;<br>ELSIF NOT(Sugar.High) THEN<br>Sugar.Inlet := 0;<br>END_IF; |

当控制器进入以下状态，[:=]将Sugar.Inlet清零：

- 进入RUN模式
- 如果用户组态SFC为自动复位，离开SFC的步时(仅适用于用户在步动作中嵌入赋值语句或使用JSR指令调用结构文本的例程的情况)

### 实例4：IF...THEN...ELSIF...ELSE

| 如果用户希望： | 输入结构文本程序： |
| --- | --- |
| 如果罐温度>100<br>那么设置泵缓慢运行<br>如果罐温度>200<br>那么设置泵快速运行<br>其他情况关闭泵 | IF tank.temp > 200 THEN<br>pump.fast :=1; pump.slow :=0; pump.off :=0;<br>ELSIF tank.temp > 100 THEN<br>pump.fast :=0; pump.slow :=1; pump.off :=0;<br>ELSE<br>pump.fast :=0; pump.slow :=0; pump.off :=1;<br>END_IF; |

## CASE...OF

使用CASE语句根据数值选择执行的操作。

### 操作数：

### 结构文本

```
CASE numeric_expression OF
    selector1: statement;
    selectorN: statement;
ELSE
    statement;
END_CASE;
```

| 操作数： | 类型： | 格式： | 输入： |
|---|---|---|---|
| 数值表达式 | SINT<br>INT<br>DINT<br>REAL | 标签表达式 | 计算出数值(数值表达式)的标签或表达式 |
| 选择器 | SINT<br>INT<br>DINT<br>REAL | 立即数 | 与numeric_expression类型相同 |

**重　要**

REAL型数值极有可能在一定的数据范围而非精确匹配某个特定数值，因此如果用户使用REAL数值，必须为选择器设定一个数值范围。

### 说明：

语法：

```
CASE numeric_expression OF
    selector1: <statement>;
               .
               .
               .
    selector2: <statement>;
               .
               .
               .
    selector3: <statement>;
               .
               .
               .
ELSE
           <statement>;
               .
               .
               .
END_CASE;
```

- selector1: <statement>; ← 当numeric_expression = 选择器1，执行的语句
- selector2: <statement>; ← 当numeric_expression = 选择器2，执行的语句
- selector3: <statement>; ← 当numeric_expression = 选择器3，执行的语句
- 根据用户需要指定的选择器的值(路径数)。
- ELSE <statement>; ← 当numeric_expression ≠ 任一选择器，执行的语句
- 可选

获取有效的选择器数值，请参阅下页表格。

输入的选择器数值的语法为：

| 当选择器为： | 输入： |
|---|---|
| 值 | 数值: 语句 |
| 多个，分散的数值 | 数值1,数值2,数值N:<语句><br>使用逗号(,)将每个值分开。 |
| 数值范围 | 数值1..数值N:<语句><br>使用两个句点(..)标记范围。 |
| 分散得数值加上一个数值范围 | 数值a,数值b,数值1..数值N:<语句> |

## 实例：

| 如果用户希望： | 输入结构文本程序： |
|---|---|
| 如果配方号＝1，那么 | CASE recipe_number OF |
| 成分A出口1＝打开(1) | 1: Ingredient_A.Outlet_1 :=1; |
| 成分B出口4＝打开(1) | Ingredient_B.Outlet_4 :=1; |
| 如果配方号＝2或3，那么 | 2,3: Ingredient_A.Outlet_4 :=1; |
| 成分A出口4＝打开(1) | Ingredient_B.Outlet_2 :=1; |
| 成分B出口2＝打开(1) | |
| 如果配方号＝4,5,6或7，那么 | 4..7: Ingredient_A.Outlet_4 :=1; |
| 成分A出口4＝打开(1) | Ingredient_B.Outlet_2 :=1; |
| 成分B出口2＝打开(1) | 8,11..13 Ingredient_A.Outlet_1 :=1; |
| 如果配方号＝8,11,12或13，那么 | Ingredient_B.Outlet_4 :=1; |
| 成分A出口1＝打开(1) | |
| 成分B出口4＝打开(1) | ELSE |
| 其他所有的出口＝关闭(0) | Ingredient_A.Outlet_1 [:=]0; |
| | Ingredient_A.Outlet_4 [:=]0; |
| | Ingredient_B.Outlet_2 [:=]0; |
| | Ingredient_B.Outlet_4 [:=]0; |
| | END_CASE; |

当控制器进入以下状态，[:=]会将出口的标签清零：

- 进入RUN模式
- 如果用户组态SFC为自动复位，离开SFC的步时(仅适用于用户在步动作中嵌入赋值语句或使用JSR指令调用结构文本的例程的情况)

# FOR...DO

**操作数：**

## 结构文本



For 计数值:=初始值TO结束值BY增量值DO<语句块>;
END_FOR;

| 操作数： | 数据类型： | 格式： | 说明： |
|---|---|---|---|
| 计数值 | SINT<br>INT<br>DINT | 标签 | FOR…DO结构执行时存储计数值的位置 |
| 初始值 | SINT<br>INT<br>DINT | 标签<br>表达式<br>立即数 | 必须指定计数初始值 |
| 结束值 | SINT<br>INT<br>DINT | 标签<br>表达式<br>立即数 | 指定计数结束值，确定何时退出循环 |
| 增量 | SINT<br>INT<br>DINT | 标签<br>表达式<br>立即数 | (可选)循环一次时计数值的增量<br>如果用户不指定增量，计数值增量默认为1。 |

**重　要**

确定在一个扫描周期内重复循环执行的次数不能过多。

- 控制器在完成循环前无法执行例程中其他语句。
- 如果循环完成所用的时间超出了任务看门狗定时器的值，产生一个主要故障。
- 考虑使用不同结构，如IF...THEN。

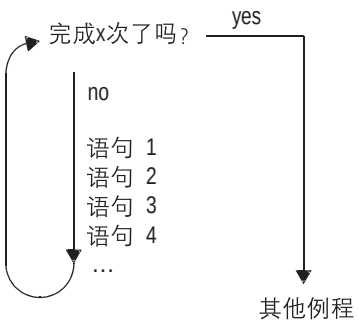
**说明：**语法是：

```
FOR count := initial_value
   TO final_value
可选 ( BY increment
   DO
   <statement>;
可选 ( IF bool_expression THEN
          EXIT;
       END_IF;
END_FOR;
```

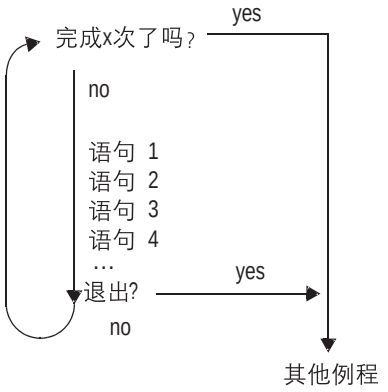
如果用户没有指定增量值，循环增量值为1。

如果需要提前退出循环，可使用其他语句，例如采用IF...THEN结构作为EXIT语句。

以下图示了FOR...DO循环如何执行以及如何使用EXIT语句提前退出循环。




**FOR...DO循环结构执行特定次数。**




**使用EXIT语句在计数值达到最终值前终止循环。**

## 实例1：

| 如果用户希望： | 输入结构文本： |
|---|---|
| 清除布尔型数组的0-31位：<br>1. 初始化subscript标签为0。<br>2. 清除array[subscript]。例如，当subscript =5，清除array[5]。<br>3. subscript加1。<br>4. 如果subscript<=31，重复步骤2和3。否则退出循环。 | For subscript:=0 to 31 by 1 do<br>array[subscript] := 0;<br>End_for; |

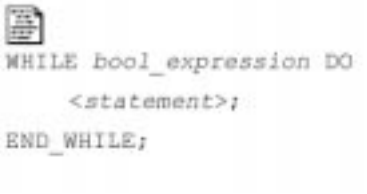
## 实例2：

| 如果用户希望： | 输入结构文本： |
| --- | --- |
| 通过用户自定义数据类型(结构)存储用户的货物库存信息：<br>• 货物的条形码ID号(字符串数据类型)<br>• 货物的采购数量(DINT数据类型)<br>上面的自定义结构数组包括了库存货物的各个元素。用户希望使用条形码ID在数组中搜索特定产品，并确定采购数量。<br>1. 设置Inventory数组的大小(货物数)并将结果存储到Inventory_Items(DINT标签)。<br>2. 初始化position标签值为0。<br>3. 如果Barcode与数组中某条目的ID一致，那么：<br>a. 设置Quantity标签=Inventory[position].Qty，生成该货物库存量。<br>b. 结束。<br>Barcode是存储货物条形码ID号的字符串标签，用户搜索该项。例如，当position=5，比较Barcode和Inventory[5].ID。<br>4. 将position值加1。<br>5. 如果position<=(Inventory_Items-1)，重复步骤3和4。元素编号起始于0，且最后一个元素编号比数组的元素号少1。否则，停止。 | SIZE(Inventory,0,Inventory_Items);<br>For position:=0 to Inventory_Items - 1 do<br>If Barcode = Inventory[position].ID then<br>Quantity := Inventory[position].Qty;<br>Exit;<br>End_if;<br>End_for; |

## WHILE...DO

只要条件为真，则使用WHILE...DO循环连续工作。

**操作数：**



```
WHILE bool_expression DO
    <statement>;
END_WHILE;
```

**结构文本**

| 操作数： | 类型： | 格式： | 输入： |
|---|---|---|---|
| 布尔表达式 | BOOL | 标签表达式 | 计算出布尔值的BOOL标签或表达式 |

**重　要**

确定在一个扫描周期内重复循环执行的次数不能太多。

- 控制器在完成循环前无法执行例程中其他语句。
- 如果循环完成所用的时间超出了任务看门狗定时器的值，产生一个主要故障。
- 考虑使用不同的结构，如IF...THEN。

**说明：**语法为：

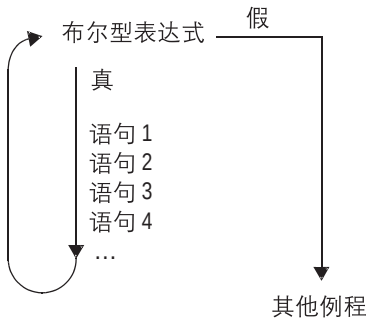
```
WHILE bool_expression1 DO
    <statement>;
    IF bool_expression2 THEN
      EXIT;
    END_IF;
END_WHILE;
```

当bool_expression1为真时，执行语句。

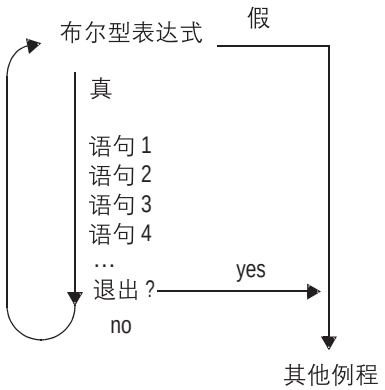
可选

如果需要提前退出循环，可使用其他语句，例如采用IF...THEN结构作为EXIT语句退出条件。

以下图示了FOR...DO循环如何执行以及如何使用EXIT语句提前退出循环。




**当bool_expression表达式值为真，控制器仅执行WHILE...DO循环内的语句。**




**在条件为真前，使用EXIT语句退出循环。**

## 实例1：

| 如果用户希望： | 输入结构文本： |
|---|---|
| WHILE…DO循环首先判断其条件。如果条件为真，然后控制器执行循环内的语句。<br><br>与REPEAT…UNTIL 循环的区别在于REPEAT…UNTIL循环执行结构内语句，然后再次执行语句前确定条件是否为真。这样，REPEAT…UNTIL 循环至少执行一次。而WHILE…DO循环中的语句可能从不执行。 | pos := 0;<br>While ((pos <= 100) & structarray[pos].value<br><> targetvalue)) do<br>pos := pos + 2;<br>String_tag.DATA[pos] :=SINT_array[pos];<br>end_while; |

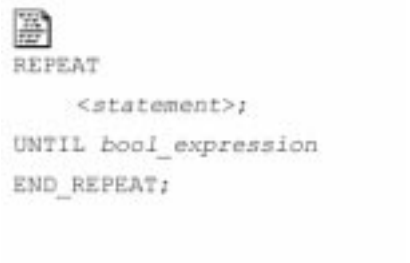
## 实例2：

| 如果用户希望： | 输入结构文本： |
| --- | --- |
| 将SINT数组中ASCII字符移入字符串标签。(在SINT数组中，每个元素保存一个字符。)当到达结束符时停止。<br>1. 初始化Element_number值为0。<br>2. 计算SINT_array中的元素数(数组中包含ASCII字符)并将结果存入SINT_array_size(DINT标签)。<br>3. 如果SINT_array[element_number]的字符为13(结束符对应的十进制数值)，则停止循环。<br>4. 设置String_tag[element_number]= SINT_array[element_number]。<br>5. element_number加1。控制器检查SINT_array中下一个字符。<br>6. 设置String_tag的长度值为element_number。<br>(该值记录当前String_tag中的字符号。)<br>7. 如果element_number＝SINT_array_size，则停止循环。<br>(已执行到数组末尾但无结束符。)<br>8. 跳到步骤3。 | element_number := 0;<br>SIZE(SINT_array, 0, SINT_array_size);<br>While SINT_array[element_number] <> 13 do<br>String_tag.DATA[element_number] :=<br>SINT_array[element_number];<br>element_number := element_number + 1;<br>String_tag.LEN := element_number;<br>If element_number = SINT_array_size then<br>exit;<br>end_if;<br>end_while; |

## REPEAT...UNTIL

REPEAT<statement>;UNTIL bool_expressionEND_REPEAT;
REPEAT...UNTIL循环连续执行语句，直至条件为真。

**操作数：**




**结构文本**

| 操作数： | 类型： | 格式： | 输入： |
|---|---|---|---|
| 布尔表达式 | BOOL | 标签表达式 | 计算出布尔值(布尔型表达式)的BOOL标签或表达式 |

**重　要**

确定在一个扫描周期内重复循环执行的次数不能过多。

- 控制器在完成循环前无法执行例程中其他语句。
- 如果循环完成所用的时间超出了任务看门狗定时器的值，产生一个主要故障。
- 考虑使用不同的结构，如IF...THEN。

**说明：**语法是：

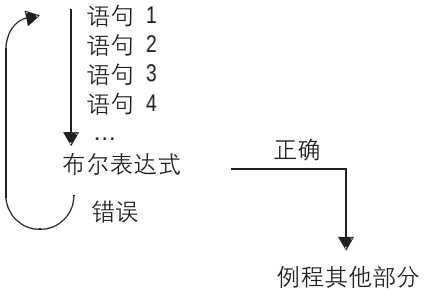
```
REPEAT
   <statement>;
   IF bool_expression2 THEN
     EXIT;
   END_IF;
UNTIL bool_expression1
END_REPEAT;
```

当bool_expression1为假时，执行语句

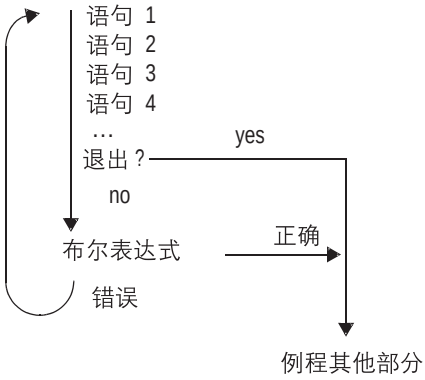
可选

如果需要提前退出循环，可使用其他语句，例如采用IF...THEN结构作为EXIT语句。

以下图示了FOR...DO循环如何执行以及如何使用EXIT语句提前退出循环。




**当bool_expression条件为假，控制器仅执行REPEAT...UNTIL循环内的语句。**




**在条件为假前退出循环，使用EXIT语句。**

## 实例1

| 如果用户希望： | 输入结构文本： |
|---|---|
| REPEAT…UNTIL首先执行循环内的语句，在再次执行循环内语句前，确定条件是否为真。 | pos := -1;<br>REPEAT |
| REPEAT…UNTIL与WHILE…DO循环的区别在于WHILE…DO循环首先判断条件。如果条件为真，处理器执行循环内语句。REPEAT…UNTIL循环内语句至少执行一次。而WHILE…DO循环内的语句可能从不执行。 | pos := pos + 2;<br>UNTIL ((pos = 101) OR<br>(structarray[pos].value =targetvalue))<br>end_repeat; |

## 实例2：

| 如果用户希望： | 输入此结构文本： |
| --- | --- |
| 将SINT数组中ASCII字符移入字符串标签。(在SINT数组中，每个元素保存一个字符。)到达结束符时停止循环。<br>1.初始化Element_number为0。<br>2.计算SINT_array中的元素数(数组中包含ASCII字符)并将结果存入SINT_array_size(DINT标签)。<br>3.设置String_tag[element_number]= SINT_array[element_number]中字符。SINT_array[element_number];<br>4.element_number加1。使得控制器检查SINT_array中下一个字符。<br>5.设置String_tag的长度值为element_number。<br>(该值记录当前String_tag中的字符号。)<br>6.如果element_number＝SINT_array_size，则停止循环。<br>(已执行到数组末尾但无结束符。)<br>跳到步骤3。<br>7. 如果SINT_array[element_number]的字符为13(结束符的十进制数值)，则停止循环。否则，跳转到步骤3。 | element_number := 0;<br>SIZE(SINT_array, 0,SINT_array_size);<br>Repeat<br>String_tag.DATA[element_number] :=<br>element_number := element_number + 1;<br>String_tag.LEN := element_number;<br>If element_number = SINT_array_size then<br>exit;<br>end_if;<br>Until SINT_array[element_number] = 13<br>end_repeat; |

## 注释

为使用户编写的结构文本程序更具可读性，需添加注释。

- 用户通过注释可以使用清晰的语言来描述结构文本程序的工作流程
- 注释不影响结构文本程序执行

在用户结构文本程序中添加注释：

| 添加注释： | 使用以下格式： |
|---|---|
| 单独一行 | //comment |
| 结构文本程序的最后一行 | (*comment*)<br>/*comment*/ |
| 一行结构文本程序内 | (*comment*)<br>/*comment*/ |
| 范围超过一行 | (*注释起始….注释结束*)<br>/*注释起始….注释结束*/ |

实例：

| 格式： | 实例： |
|---|---|
| //comment | 行起始<br>//Check conveyor belt direction IF conveyor_direction THEN...<br><br>行尾<br>ELSE //If conveyor isn.t moving, set alarm light<br>light := 1;<br>END_IF; |
| (*comment*) | Sugar.Inlet[:=]1;(*open the inlet*)<br><br>IF Sugar.Low (*low level LS*)& Sugar.High (*high level LS*)THEN...<br><br>(*Controls the speed of the recirculation pump. The speed depends on the temperature in the tank.*)IF tank.temp > 200 THEN... |
| /*comment*/ | Sugar.Inlet:=0;/*close the inlet*/<br><br>IF bar_code=65 /*A*/ THEN...<br><br>/*Gets the number of elements in the Inventory array and stores the value in the Inventory_Items tag*/SIZE(Inventory,0,Inventory_Items); |

# 通用属性

## 概述

本附录描述了Logix指令的通用属性。

| 相关信息： | 参见页码： |
|---|---|
| 立即数 | C - 1 |
| 数据转换 | B - 1 |

## 立即数

无论用户何时输入十进制格式的立即数(常数)(如 -2，3)，控制器都使用32位存储该数值。如果用户输入的值为基数形式而不是十进制，例如二进制或十六进制，并且没有指定所有的32位，则控制器将零放入没有指定的位(零表示填充)。

**举　例**　立即数的零-填充。

| 举例：　立即数的零-填充。 | |
|---|---|
| 若输入： | 控制器存储： |
| -1 | 16#ffff ffff (-1) |
| 16#ffff (-1) | 16#0000 ffff (65535) |
| 8#1234 (668) | 16#0000 029c (668) |
| 2#1010 (10) | 16#0000 000a (10) |

## 数据转换

当用户在程序内混合使用数据类型时，将进行数据转换。

| 在…内编程时： | 下述情况发生数据转换： |
|---|---|
| 梯形图 | 在一条指令内的参数为混合数据类型 |
| 功能块 | 连接两个具有不同数据类型的参数 |

如果指令的所有操作数使用下述类型，则指令的执行时间更短、需求的内存更小：

- 同一数据类型
- 最优数据类型：
  - 本手册每条指令的“操作数”部分，粗体数据类型代表最优数据类型。
  - DINT和REAL数据类型是典型的最优数据类型。
  - 大部分功能块指令的操作数只支持一种数据类型(最优数据类型)。

如果用户混合使用数据类型并且使用非最优数据类型的标签，控制器按照下述原则转换数据：

- 其中一个操作数是REAL值吗？

| 如果： | 那么输入操作数(例如，源，表达式中标签，极限值)转换为： |
|---|---|
| Yes | REALs |
| No | DINTs |

- 指令执行后，如有必要，结果(DINT或REAL值)将转换为目标数据类型。

用户不能在一条使用整数或REAL数据类型的指令内指定BOOL型标签。

由于数据转换占用额外的时间和内存，用户可通过下述方法提高程序的效率：

- 整条指令使用同样的数据类型
- 尽量不使用SINT或者INT数据类型

也就是说，在指令内，包括立即数，使用全部DINT标签或者REAL标签。

下面解释了用户使用SINT或者INT标签时或者混合数据类型时，数据如何转换。

## SINT或INT转换为DINT

对于那些将SINT或INT值转换到DINT值的指令，本手册的“操作数”部分标出了转换方法。

| 转换方法： | 转换数据过程： |
|---|---|
| 符号-扩充 | 将最左边位(数值符号)的值放入现有位以左的每一位内，直到满32位。 |
| 零-填充 | 将零放入现有位以左的每一位内，直到满32位。 |

下例给出了使用符号-扩充和零-填充方法转换数值的结果。

| | | |
|---|---|---|
| 数值 | 2#1111_1111_1111_1111 | (-1) |
| 通过符号-扩充转换数值 | 2#1111_1111_1111_1111_1111_1111_1111_1111 | (-1) |
| 通过零-填充转换数值 | 2#0000_0000_0000_0000_1111_1111_1111_1111 | (65535) |

由于立即数一直按零-填充存储，所以SINT或INT值的转换可能会产生非预期结果。在下例中，比较结果为假，这是因为源A是一个INT值，通过符号-扩充转换；而源B是一个立即数，按零-填充存储。



如果用户在一条通过符号-扩充转换数据的指令内使用SINT或INT标签和立即数，那么使用下述方法对处理立即数：

- 以十进制基数指定每一个立即数。
- 如果用户输入基数而非十进制的值，则指定立即数的全部32位。也就是将最左边位的值输入它以左的每位内，直到具有32位。
- 为每个操作数创建标签并且整条指令使用相同的数据类型。为了赋值一个常数，可采取下列方法之一：
  - 将其输入到多个标签之一
  - 增加一条MOV指令，将该值移到标签里。
- 使用MEQ指令仅检查所要求的位

下例给出将立即数与INT标签混合使用的两种方法。两个例子均检查1771 I/O模块的各位，从而确定所有位是否接通。由于1771 I/O模块的输入数据字是INT标签，易于使用16-位的常数。

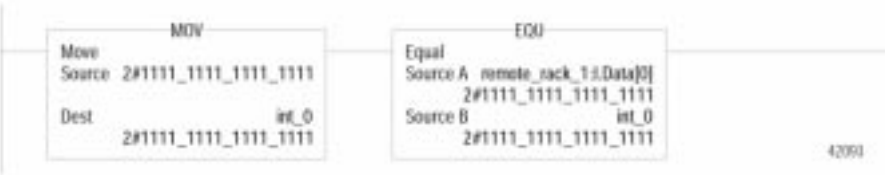
**举 例** 混合使用INT标签与立即数

由于remote_rack_1:I.Data[0]是一个INT标签，故检测它的值也以一个INT标签输入。




**举 例** 混合使用INT标签与立即数

由于remote_rack_1:I.Data[0]是一个INT标签，故检测它的值首先移到INT标签的int_0内，然后EQU指令比较这两个标签。



## Integer转换为REAL

控制器以IEEE单精度、浮点数格式存储REAL数值。其中一位用作数值符号，23位用于基值，8位用于指数(总共32位)。如果在同一条指令内将混合的整数标签(SINT， INT或 DINT)和REAL标签作为输入，那么在指令执行前控制器将整数转换为REAL值。

- SINT或者INT值总是转换为相同的REAL值。
- DINT值可能不会转换为相同的REAL值：
  - 一个REAL值的基值占用24位(23个存储位加一个“隐藏”位)。
  - 一个DINT值占用32位(1位用于符号，31位用于数值)。
  - 如果DINT值要求24位以上的有效位，那么它不可能转换为相同的REAL值。控制器将其四舍五入为最接近的使用24个有效位的REAL值。

## DINT转换为SINT或INT

为了将DINT值转换到SINT或INT值，如有必要，控制器将截去DINT的上面部分并置位溢出状态标志位。下例给出了DINT转换到SINT或INT的结果。

**举 例** DINT转换为SINT以及INT

| DINT值： | 转换为更小的值： | |
|---|---|---|
| 16#0001_0081 (65,665) | INT: | 16#0081 (129) |
| | SINT: | 16#81 (-127) |

## REAL转换为整数

为了转换REAL值为整数值，控制器将小数部分四舍五入并对非-小数部分向上取整。如果数据丢失，控制器置位溢出状态标志位。

按下列原则取整数据：

- 除了x.5以外，其他数都四舍五入为最接近的整数。
- x.5圆整为最接近的偶数。

下例给出了转换REAL值为DINT值的结果。

**举　例**　转换REAL值为DINT值

| REAL值： | 转换为DINT值： |
|---|---|
| -2.5 | -2 |
| -1.6 | -2 |
| -1.5 | -2 |
| -1.4 | -1 |
| 1.4 | 1 |
| 1.5 | 2 |
| 1.6 | 2 |
| 2.5 | 2 |

**重　要**　算术状态标志位由所存储的值决定。一般不影响算术状态关键字的指令可能也会对其产生影响，这是因为指令参数的混合数据类型如果发生类型转换，则类型转换过程将设置算术状态关键字。

# 功能块面板控制

## 概述

RSLogix 5000编程软件中包括用于部分功能块指令的面板控制。这些面板是用户能够在RSView32软件内或者其他支持Active-X控件的应用软件内使用的Active-X控件。面板通过RSLinx OPC Server与控制器通讯。

**重　要**

RSLogix 5000编程软件不是一个有效的Active-X载体。要使用面板，用户必须具有一个Active-X载体，如RSView32软件。

以下指令具有面板控制：

| 指令： | 参见页码： |
|---|---|
| 报警 (ALM) | D - 6 |
| 增强型选择(ESEL) | D - 8 |
| 累加器 (TOT) | D - 9 |
| 斜坡/渗透(RMPS) | D - 11 |
| 离散型2－态设备(D2SD) | D - 14 |
| 离散型3－态设备(D3SD) | D - 16 |
| 增强型PID (PIDE) | D - 18 |

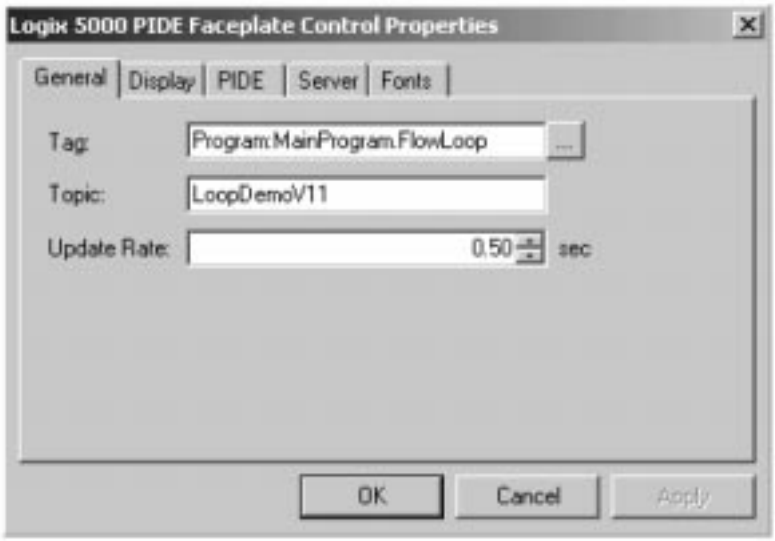
打开Active-X载体应用软件，如RSView32软件，通过其属性页组态面板。

所有的面板有四个公共的属性页。

- 基本属性
- 显示属性
- 服务器属性
- 字体属性

## 组态基本属性

基本属性页确定了控制如何操作。




| 属性页上要素： | 说明： |
|---|---|
| Tag | 该入口连接指定的功能块指令与控制。 |
| Topic | 该选项组态访问路径。该值需要连接到RSLinx OPC Server。 |
| Update Rate | 该选项组态控制的更新速率，以秒为单位。使用滚动条控制以0.25秒的增量修改该速率。<br>缺省值 = 1.00秒 |

**重　要**

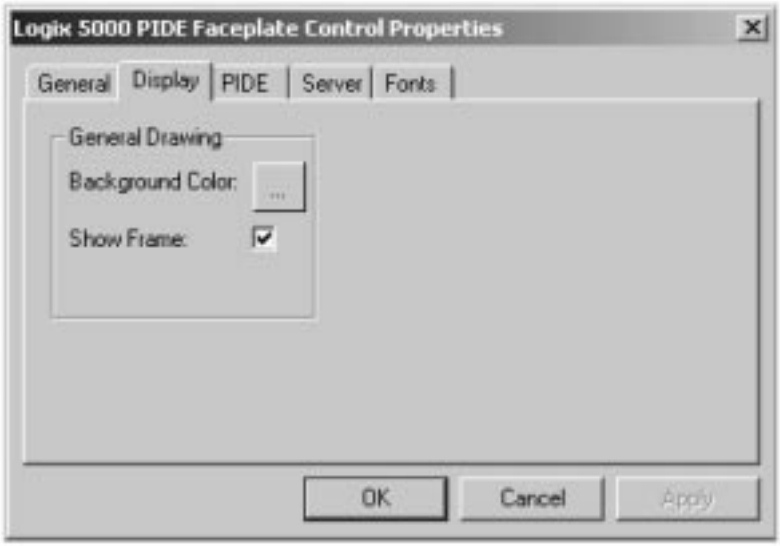
上面屏幕内的实例Block Tag显示了控制器-范围的标签名称。缺省值时，当用户插入功能块时，它会自动分配一个程序-范围的功能块标签。要指定一个程序-范围的功能块标签名PID1，输入：

程序：*program_name*.PID1

其中*program_name*是程序的名称。

## 组态显示属性

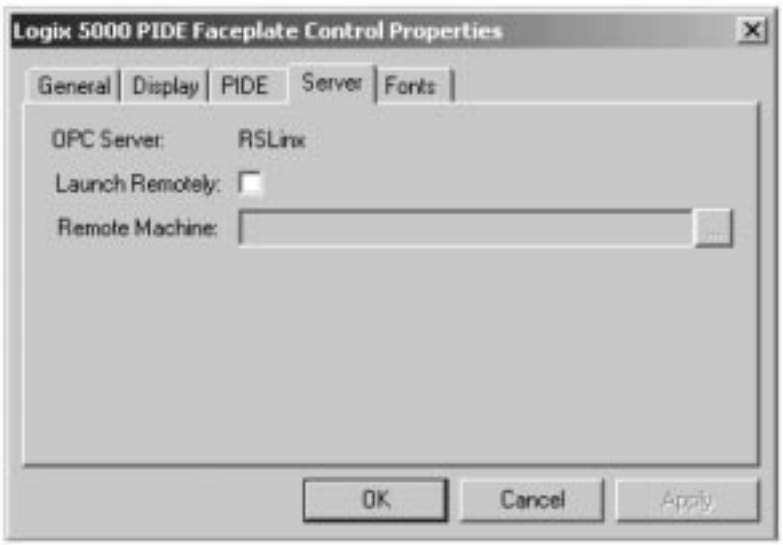
显示属性页确定了屏幕的基本属性。




| 属性页上要素： | 说明： |
|---|---|
| Background Color | 该按钮设置面板的背景颜色。<br>缺省值 = 浅灰 |
| Show Frame | 该选项起动和关闭用于控制的3-维边框。它可用来区分控制和其他显示的项目。<br>缺省值 = 选中 |

## 组态服务器属性

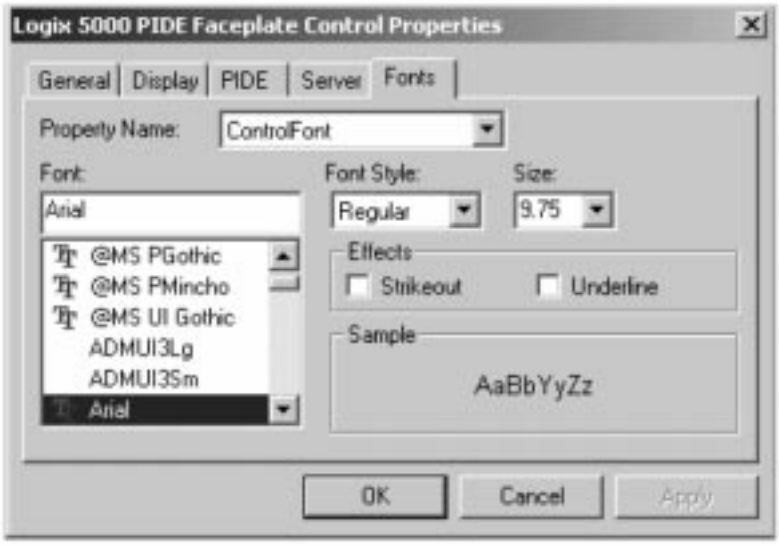
用户通过服务器属性页组态RSLinx OPC server，进而能够在远程工作站上使用面板。




| 属性页上要素： | 说明： |
|---|---|
| Launch Remotely | 选择是否与远程设备连接。<br>缺省值＝未选中(不与远程工作站连接) |
| Remote Machine | 选中Launch Remotely框，使能该区域。指定面板连接的远程工作站的名称。为正确操作面板，远程工作站必须安装RSLinx Gateway软件，并为该客户端工作站和用户设定合适的安全设置。点击省略号扫描本地局域网内的设备。 |

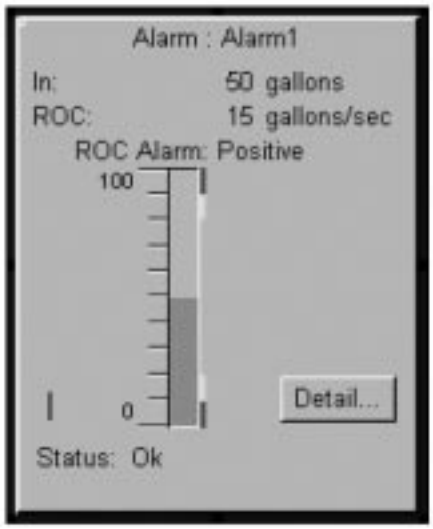
## 组态字体属性

字体属性页确定了显示在面板上的字体。ControlFont用于面板的主要部分，而MinorFont用于数值以及面板的其他次要部分。

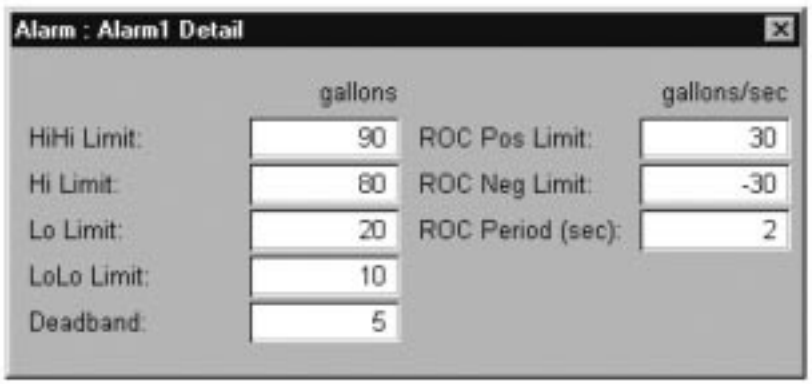



| 属性页上要素： | 说明： |
|---|---|
| Property Name | 使用下拉键选择字体。选择ControlFont或MinorFont。<br>缺省值 = ControlFont |
| Font | 选择控制的字体。列表包含系统内所有可用字体。<br>缺省值 = Arial |
| Size | 配置字体的大小。<br>缺省值ControlFont = 10.5磅<br>缺省值MinorFont = 8.25磅 |
| Effects | 选择字体是否加下划线以及/或者加删除线。<br>缺省值 = 两个都不选中 |

# ALM控制




| 控制面板的要素： | 显示： |
|---|---|
| In | 当前输入值。 |
| Rate Of Change (ROC) | ROC(变化率)的数值。<br>如果ROCPosAlarm或ROCNegAlarm置位，则文字颜色变为红色。当光标指向该控制时，出现Rate Of Change工具提示。 |
| Alarm Bar Meter | 与功能块的Alarm Limits关联时，该值为功能块的输入值。<br>如果HAlarm或LAlarm置位，仪表棒状图为黄色。同样，如果HHAlarm或LLAlarm置位，仪表棒状图为红色。如果没有报警，则仪表棒状图为绿色。 |
| Alarm Marking Bars | HHLim，HLim，LLim和LLLim的数值。<br>HHLim 和 LLLim棒状图为红色。HLim 和 LLim棒状图为黄色。 |
| Alarm Meter Scale | 报警条的刻度。<br>刻度的上限 = HHLim + Deadband。刻度的下限 = LLLim ®CDeadband。 |
| Detail Button | 弹出详细内容对话框。 |
| Status | 功能块内所有置位的状态位。如果没有状态位置位，该状态显示OK。 |

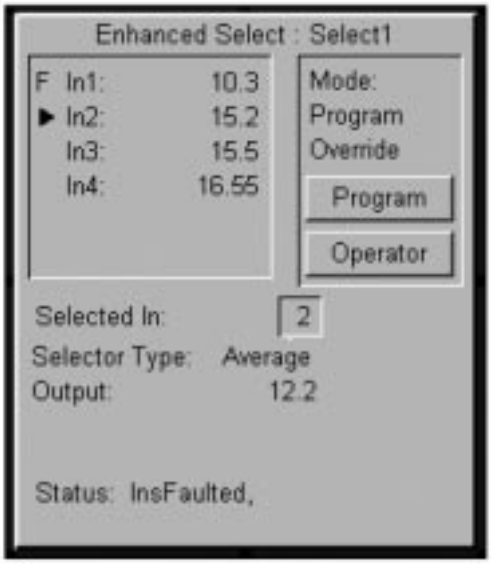


ALM控制面板具有以下附加属性页。



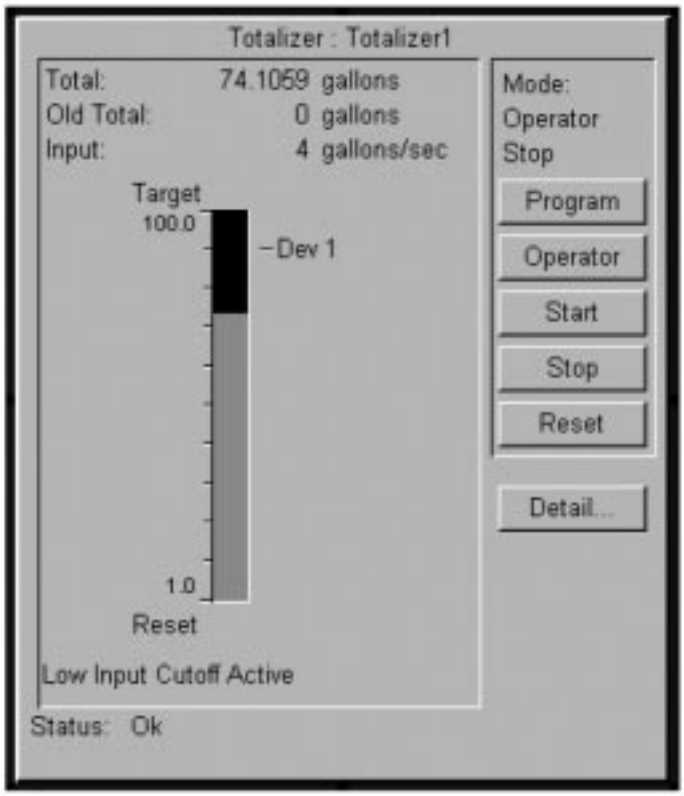

| 属性页的要素： | 说明： |
|---|---|
| In Units | 该控制面板上输入区域的单位。 |
| Meter Color | 当前无报警时的仪表栏颜色。 |
| H-L Color | 指令处于Low或者High报警状态时仪表栏的颜色。 |
| HH-LL Color | 指令处于Low-Low或者High-High报警状态时仪表栏的颜色。 |

# ESEL控制




| 控制面板的要素： | 显示： |
|---|---|
| Mode | 功能块的模式。 |
| Input | 功能块的输入。<br>显示数值(1-6)取决于InsUsed数量。 |
| Fault Indicator | 如果特定输入出现故障，则输入左侧显示字母F。 |
| Selected Indicator | 输入左侧显示的三角号表示了所选择的输入。 |
| Program Button | 用户点击该按钮时，OperProgReq置位。 |
| Operator Button | 用户点击该按钮时，OperOperReq置位。 |
| Selected In | SelectedIn的值。 |
| Selector Type | 选择模式。 |
| Output | 输出Out值。 |
| Status | 功能块内所有置位的状态位。如果没有状态位置位，该状态显示OK。 |

# TOT控制




| 控制面板的要素： | 显示： |
|---|---|
| Mode | 功能块的模式。 |
| Total | 累加值。 |
| Old Total | 上次累加值。 |
| Input | In值。 |
| Total Meter | 累加值范围。 |
| Target Dev1 and Dev2 Tick Marks | TargetDev1 和TagetDev2的值。 |
| Total Scale | 累加表的刻度。<br>刻度的上限 = Target。刻度的下限 = Reset。 |
| Program Button | 当用户点击该按钮时，OperProgReq置位。 |
| Operator Button | 当用户点击该按钮时，OperOperReq置位。 |
| Start Button | 当用户点击该按钮时，OperStartReq置位。 |
| Stop Button | 当用户点击该按钮时，OperStopReq置位。 |
| Reset Button | 当用户点击该按钮时，OperResetReq置位。 |

| 控制面板的要素： | 显示： |
|---|---|
| Detail Button | 弹出详细内容对话框。 |



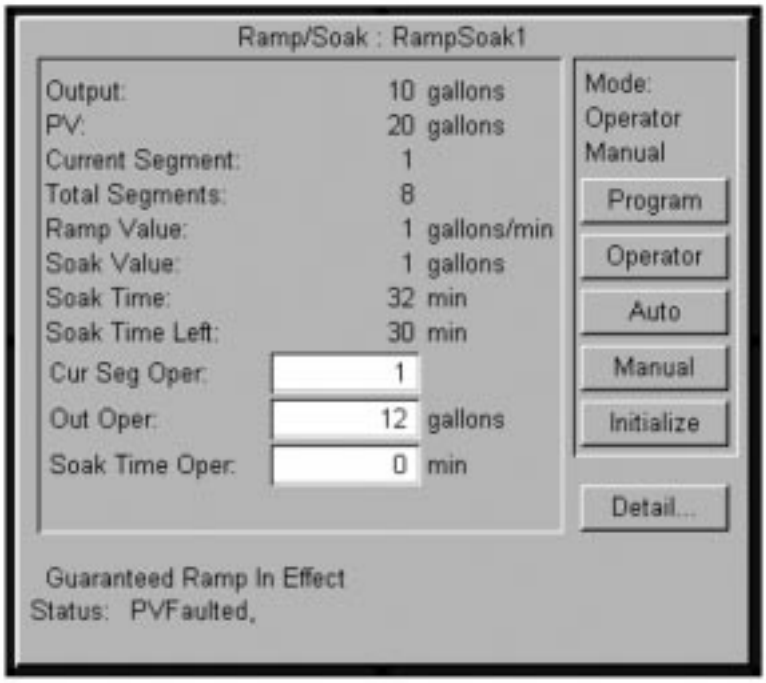

| | |
|---|---|
| Low Input Cutoff Active | 只当LowInCutoffFlag置位时，说明Low Input Cutoff Active。 |
| Status | 功能块内所有置位的状态位。如果没有状态位置位，该状态显示OK。 |

TOT控制有以下附加属性页。



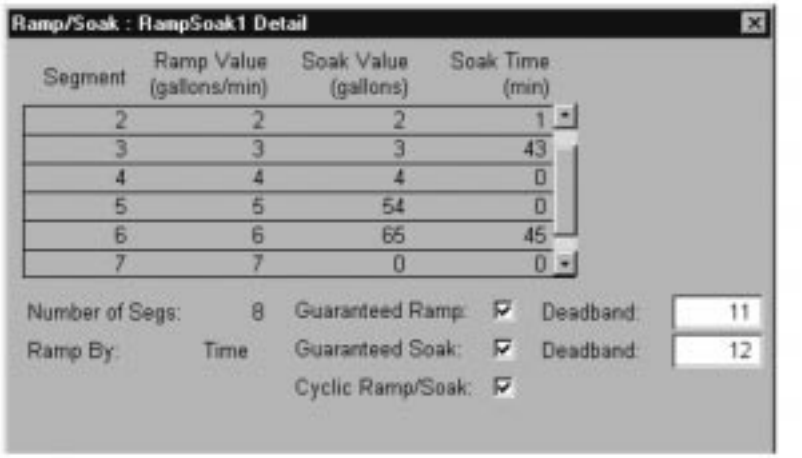

| 属性页的要素： | 说明： |
|---|---|
| In Units | 该控制面板上输入区域的单位。 |
| Total Units | 控制面板上Total 和Old Total区域的单位。 |
| Meter Color | 仪表栏颜色。 |

# RMPS控制



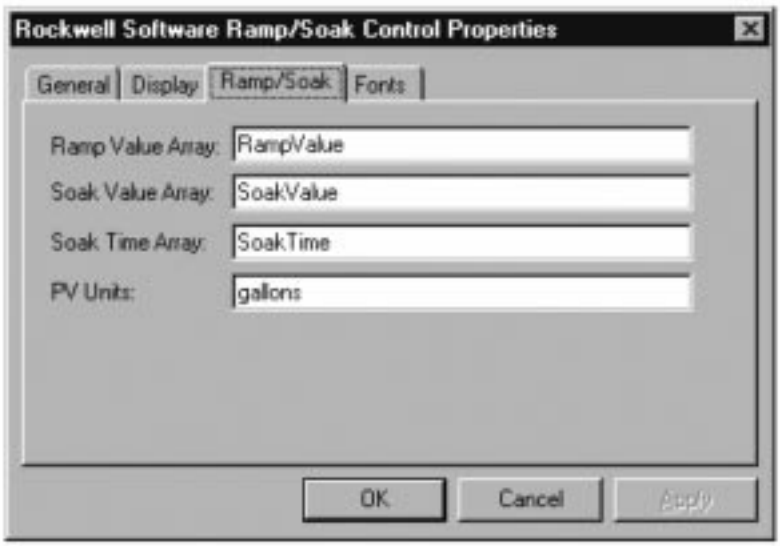

| 控制面板的要素： | 显示： |
|---|---|
| Mode | 功能块模式。 |
| Output | Out的值。 |
| PV | PV值。 |
| Current Segment | 当前段(Current Segment)的值。 |
| Ramp Value | 当前段的RampValue[ ] 值。 |
| Soak Value | 当前段的SoakValue[ ] 值。 |
| Soak Time | 当前段的SoakTime[ ] 值。 |
| Soak Time Left | SoakTimeLeft的值。 |
| Program Button | 当用户点击该按钮时，OperProgReq置位。 |
| Operator Button | 当用户点击该按钮时，OperOperReq置位。 |
| Auto Button | 当用户点击该按钮时，OperAutoReq置位。 |
| Manual Button | 当用户点击该按钮时，OperManualReq置位。 |
| Initialize Button | 当用户点击该按钮时，Initialize置位。<br>该按钮只有功能块处于操作员人工模式下被使能。 |

| 控制面板的要素： | 显示： |
|---|---|
| Detail Button | 弹出详细内容对话框。 |



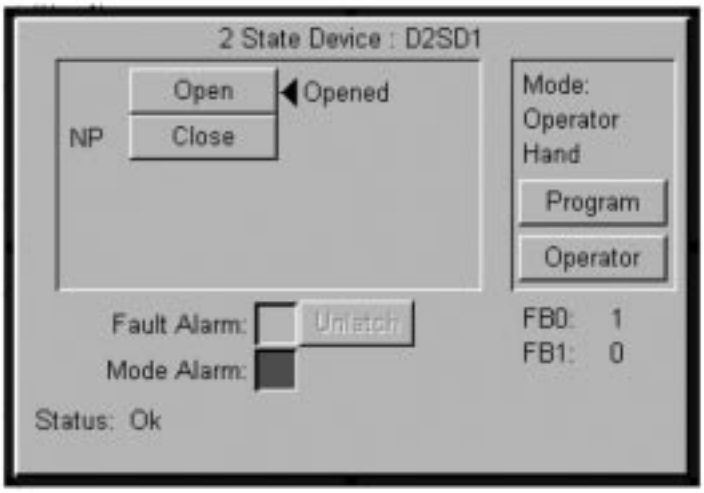

| | |
|---|---|
| Cur Seg Oper | CurrentSegOper的值。 |
| Out Oper | OutOper 的值。 |
| Soak Time Oper | SoakTimeOper的值。 |
| Guaranteed Ramp or Soak in Effect | 当对应的GuarRamp 或 GuarSoak位置位时，说明Guaranteed Ramp有效 或者 Guaranteed Soak 有效 。 |
| Status | 功能块内所有置位的状态位。如果没有状态位置位，该状态显示OK。 |

RMPS控制面板具有以下附加属性页。




| 属性页的要素： | 说明： |
|---|---|
| Ramp Value Array | 带有斜率值的控制器内数组。 |
| Soak Value Array | 带渗透值的控制器内数组。 |
| Soak Time Array | 带有渗透时间的控制器数组。 |
| PV Units | 显示在控制面板上的单位。 |

# D2SD控制



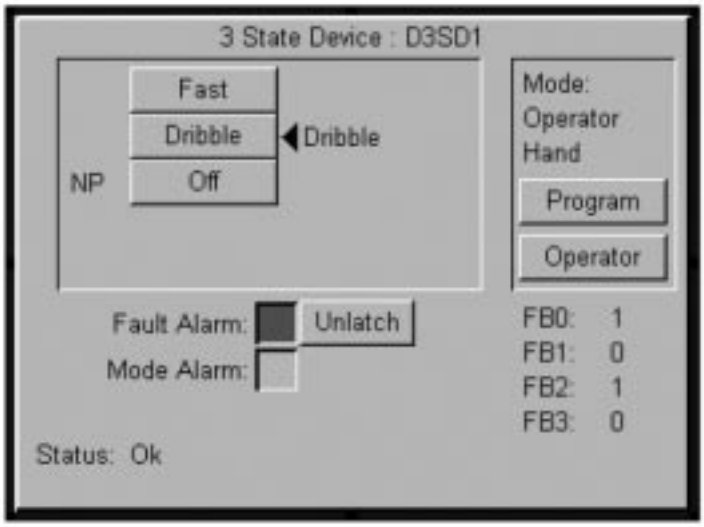

| 控制面板的要素： | 显示： |
|---|---|
| Mode | 功能块的模式。 |
| State Buttons | 在控制的属性页内定义的命令状态标签的打开或关闭状态。<br>顶部按钮设定Oper1Req。底部按钮设定Oper0Req。点击时，该按钮设定OperReq区域为特定的状态。 |
| Ordered State Indicator | 指向所需的按钮得到其状态的命令状态(Command Status)的值。 |
| Actual State Indicators | 实际状态值。<br>如果DeviceStatus置位，实际状态组态为预定的状态。 |
| Non-Permissive Indicator | 如果StatePerm没有置位该状态，则在按钮的左边显示NP字母。 |
| Fault Alarm Indicator | 指示FaultAlarm是否置位。 |
| Mode Alarm Indicator | 指示ModeAlarm是否置位。 |
| Unlatch Button | FaultAlmUnlatch的状态。<br>点击该按钮时，FaultAlmUnlatch置位。仅当FaultAlarm 和 FaultAlmLatch置位时，该按钮使能。 |
| Program Button | 当用户点击该按钮时，OperProgReq置位。 |
| Operator Button | 当用户点击该按钮时，OperOperReq置位。 |
| FB1 | FB1的值。 |
| FB0 | FB0的值。 |
| Status | 功能块内所有置位的状态位。如果没有状态位置位，该状态显示OK。 |

D2SD控制面板具有以下附加属性页。




| 属性页的要素： | 说明： |
|---|---|
| Commanded State 0 Label | Commanded State 0的标签。 |
| Commanded State 1 Label | Commanded State 1的标签。 |
| Actual State 0 Label | Actual State 0的标签。 |
| Actual State 1 Label | Actual State 1的标签。 |

# D3SD控制



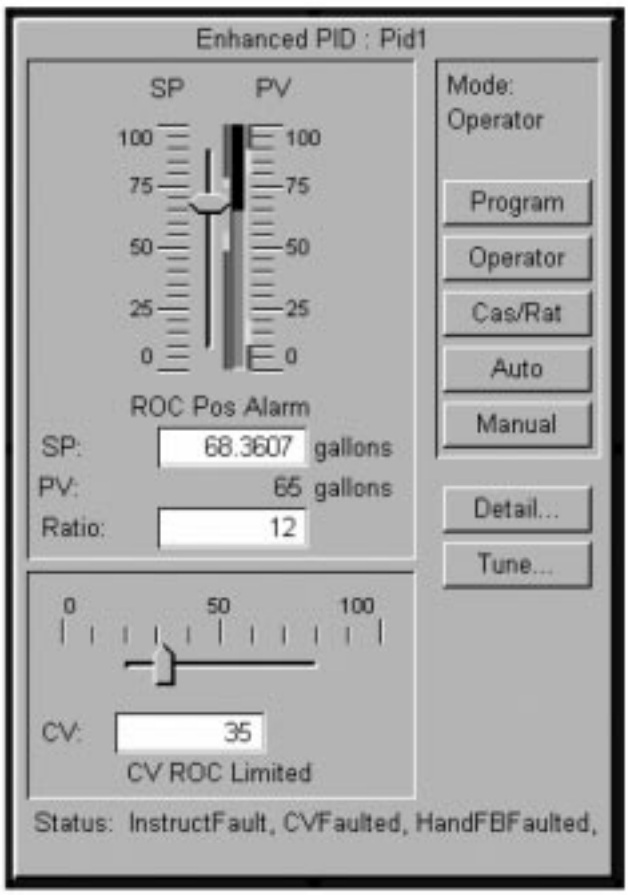

| 控制面板的要素： | 显示： |
|---|---|
| Mode | 功能块模式。 |
| State Buttons | 在控制的属性页内定义的命令状态标签的打开或关闭状态。<br>顶部按钮设定Oper2Req。中间按钮设定Oper1Req。底部按钮设定Oper0Req。点击后，该按钮设定OperReq区域为特定的状态。 |
| Ordered State Indicator | 指向所需按钮得到其状态的Command Status的值。 |
| Actual State Indicators | 实际状态值。<br>如果DeviceStatus置位，实际状态组态为给定状态值。 |
| Non-Permissive Indicator | 如果该状态的StatePerm没有置位，则按钮左侧显示NP字母。 |
| Fault Alarm Indicator | 指示FaultAlarm是否置位。 |
| Mode Alarm Indicator | 指示ModeAlarm是否置位。 |
| Program Button | 当用户点击该按钮时，OperProgReq置位。 |
| Operator Button | 当用户点击该按钮时，OperOperReq置位。 |
| FB3 | FB3的值。 |
| FB2 | FB2的值。 |
| FB1 | FB1的值。 |
| FB0 | FB0的值。 |
| Unlatch Button | FaultAlmUnlatch的状态。<br>点击该按钮时，FaultAlmUnlatch置位。仅当FaultAlarm 和 FaultAlmLatch置位，该按钮使能。 |
| Status | 功能块内所有置位的状态位。如果没有状态位置位，该状态显示OK。 |

D3SD控制面板具有以下附加属性页。



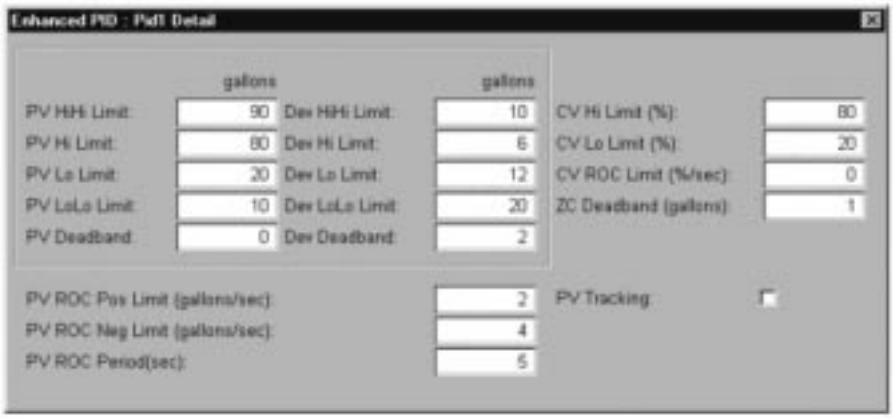

| 属性页的要素： | 说明： |
|---|---|
| Commanded State 0 Label | Commanded State 0的标签。 |
| Commanded State 1 Label | Commanded State 1的标签。 |
| Commanded State 2 Label | Commanded State 2的标签。 |
| Actual State 0 Label | Actual State 0的标签。 |
| Actual State 1 Label | Actual State 1的标签。 |
| Actual State 2 Label | Actual State 2的标签。 |

# PIDE控制

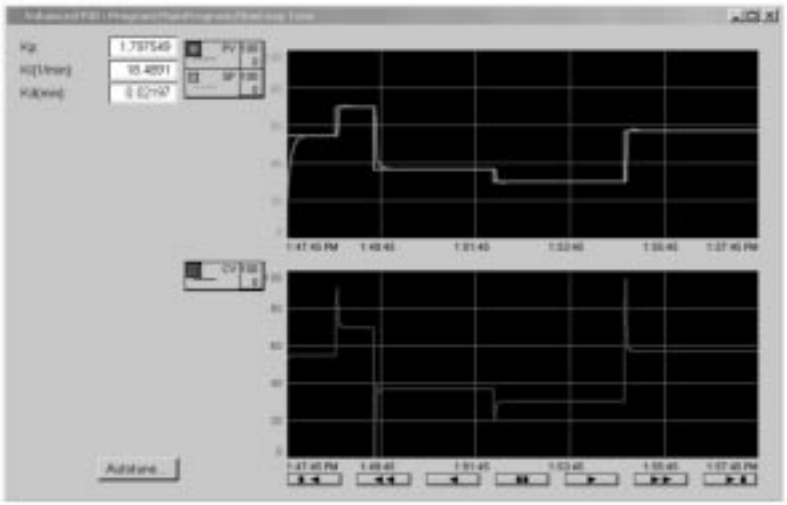



| 控制面板的要素： | 显示： |
|---|---|
| Mode | 功能块模式。 |
| PV Barmeter | PV值。仪表栏限幅为PVEUMax 和 PVEUMin。 |
| Alarm Bars | Deviation 和PV Limit Alarms的限值。<br>Deviation Alarm Bars在左侧，随SP移动。PV Limit Alarms在右侧，保持静止。 |
| SP Slider | SP的值。<br>游标界限为PVEUMax 和PVEUMin。游标通过通道限制在SPHLimit 与SPLLimit间(可能无法完全覆盖PV值范围)。 |
| ROC Alarm Indicator | PVROCPosAlarm 和 PVROCNegAlarm的状态。 |
| Ratio | Ratio值。<br>只有在AllowCasRat 和UseRatio位均置位时显示该值。 |
| SP | SP值。用户可以在这种编辑状态下输入新的SP值。 |
| PV | PV值。 |
| CV Slider | CV值。<br>滑块的限幅为0% 到100%。 |
| CV | CV值。 |

| 控制面板的要素： | 显示： |
|---|---|
| Program Button | 当用户点击该按钮时，OperProgReq置位。 |
| Operator Button | 当用户点击该按钮时，OperOperReq置位。 |
| Cas/Rat Button | 当用户点击该按钮时，OperCasRatReq置位。 |
| Auto Button | 当用户点击该按钮时，OperAutoReq置位。 |
| Manual Button | 当用户点击该按钮时，OperManualReq置位。 |
| Detail Button | 弹出详细内容对话框。 |



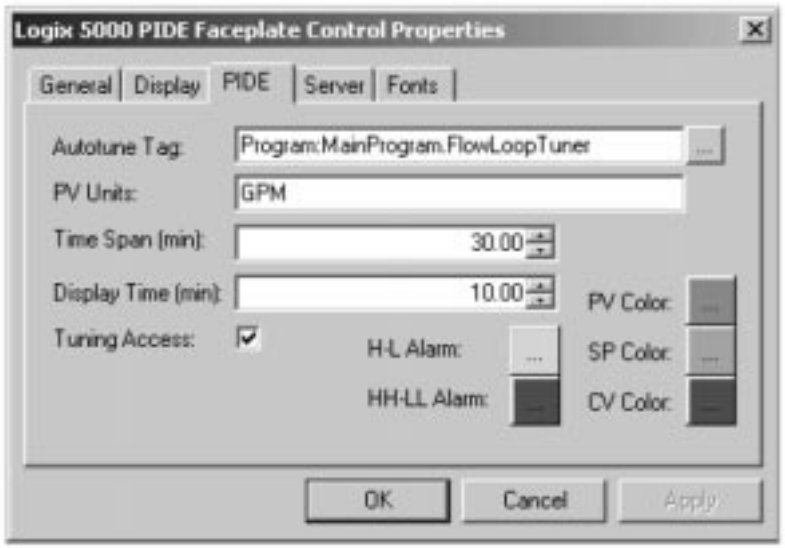

| | |
|---|---|
| Tune Button | 弹出调试对话框。 |



| 控制面板的要素： | 显示： |
|---|---|
| Autotune Button | 弹出自动调节对话框(点击上图所示调试对话框中的自动调节按钮) |
| Status | 功能块内所有置位的状态位。如果没有状态位置位，该状态显示OK。 |




PIDE控制面板具有以下附加属性页。




| 组态该属性： | 指定： |
|---|---|
| PV Units | 控制面板上PV和SP单位的字符串。 |
| Time Span | 保存趋势图数据的时间长度。 |
| Display Time | 显示趋势图数据的时间长度。 |
| PV, SP, and CV Colors | 每个参数的趋势曲线颜色。 |
| Alarm Colors | 色彩栏内提供了代表报警的颜色。 |
| | Alarm 1 Color代表上限或下限报警。Alarm 2 Color代表高上限或低下限报警。 |

**A**

报警 1-2
**ALARM**结构 1-2
**ALM** 1-2
算术状态标志
溢出 A-7
**ASCII**指令
STOD B-14
设定数据可用 A-5,A-6,A-7
属性
数据类型转换C-1
立即数 C-1
自动调节 1-53

**C**

**CASE** B-17
通用属性 C-1
数据类型转换 C-1
立即数 C-1
数据类型转换 C-1

**D**

**D**触发器 6-2
**D2SD** 1-6
**D3SD** 1-15
死区时间 1-28
**DEDT** 1-28
微分 3-2
**DERV** 3-2
**DFF** 6-2
离散型_2态设备 1-6
离散型_3态设备 1-15
**DISCRETE_2STATE**结构 1-6
**DOMINANT_RESET**结构 6-6
**DOMINANT_SET**结构 6-8
驱动指令
INTG 2-2
PI 2-8
PMUL 2-20
SCRV 2-28
SOC 2-38
UPDN 2-46

**E**

增强型**PID** 1-41
增强型选择 4-2
**ESEL** 4-2
执行顺序 A-4

**F**

面板
ALM 1-4, D-6
D2SD 1-9, D-14
D3SD 1-20, D-16
显示属性 D-3
ESEL 4-6, D-8
字体属性 D-5
基本属性 D-2
PIDE 1-53, D-18
RMPS 1-86, D-11
TOT 1-110, D-9
反馈回路
功能块图 A-5
**FGEN** 1-33
滤波指令
DERV 3-2
HPF 3-6
LDL2 3-12
LPF 3-18
NTCH 3-24
**FILTER_HIGH_PASS**结构 3-6
**FILTER_LOW_PASS**结构 3-18
**FILTER_NOTCH_PASS**结构 3-24
**FLIP_FLOP_D**结构 6-2
**FLIP_FLOP_JK**结构 6-4
功能块图
创建一个扫描周期的延迟 A-7
解析回路 A-5
解析功能块间数据流 A-6
函数发生器 1-33
**FUNCTION_GENERATOR** 结构 1-34

**H**

高通滤波器 3-6
上/下极限 4-9
**HL_LIMIT**结构 4-9
**HLL** 4-9
**HPF** 3-6

I

立即数 C-1
积分器 2-2
**INTEGRATOR**结构 2-2
**INTG** 2-2

J

**JK**触发器 6-4
**JKFF** 6-4

L

数据锁定 A-2
**LDL2** 3-12
**LDLG** 1-37
**LEAD_LAG**结构 1-37
**LEAD_LAG_SEC_ORDER** 结构 3-12
超前_滞后补偿 1-37
低通滤波器 3-18
**LPF** 3-18

M

**MAVE** 5-2
**MAXC** 5-6
捕捉最大值 5-6
**MAXIMUM_CAPTURE**结构 5-6
**MINC** 5-8
捕捉最小值 5-8
混合数据类型 C-1
移动/逻辑指令
- DFF 6-2
- JKFF 6-4
- RESD 6-6
- SETD 6-8

移动平均值 5-2
移动标准偏差 5-10
**MOVING_AVERAGE** 结构 5-2
**MOVING_STD_DEV**结构 5-10
**MSTD** 5-10
多路切换器 4-12
**MULTIPLEXER** 结构 4-12
**MUX** 4-12

N

凹波器 3-24
**NTCH** 3-24

O

执行顺序 A-4
溢出条件 A-7

P

**PI** 2-8
**PIDE** 1-41
**PIDE**自动调节1-53
**PIDE_AUTOTUNE**结构 1-53
**PMUL** 2-20
位置比例 1-75
**POSITION_PROP**结构 1-75
**POSP** 1-75
过程控制指令
- ALM 1-2
- D2SD 1-6
- D3SD 1-15
- DEDT 1-28
- FGEN 1-33
- LDLG 1-37
- PIDE 1-41
- POSP 1-75
- RMPS 1-82
- SCL 1-96
- SRTP 1-100
- TOT 1-106

程序/操作员控制
- D2SD 1-11
- D3SD 1-23
- ESEL 4-8
- 概述 A-14
- PIDE 1-60
- RMPS 1-89
- TOT 1-112

编程实例(programming examples)
- ESEL 4-7
- POSP 1-81
- RMPS 1-89
- SCL 1-99
- SRTP 1-105

**PROP_INT** 结构 2-8
比例＋积分 2-8
多路脉冲切换器 2-20

**R**

斜坡/渗透 1-82
**RAMP_SOAK** 结构 1-83
速率极限 4-15
**RATE_LIMITER** 结构 4-15
**RESD** 6-6
复位优先 6-6
**RLIM** 4-15
**RMPS** 1-82

**S**

**S_CURVE**结构 2-28
**Scale** 1-96
**SCALE** 结构 1-96
扫描延时
功能图 A-7
**SCL** 1-96
**SCRV** 2-28
**S-Curve** 2-28
**SEC_ORDER_CONTROLLER**结构 2-38
二阶控制器**(SOC)** 2-38
二阶超前_滞后补偿 3-12
**SEL** 4-19
**Select** 4-19
**SELECT**结构 4-19
选择限制指令
ESEL 4-2
HLL 4-9
MUX 4-12
RLIM 4-15
SEL 4-19
SNEG 4-21
SSUM 4-23
**SELECT_ENHANCED** 结构 4-2
**SELECTABLE_NEGATE** 结构 4-21
**SELECTABLE_SUMMER** 结构 4-23
取反选择 4-21
求和选择 4-23
置位优先 6-8
**SETD** 6-8
**SNEG** 4-21
**SOC** 2-38
脉宽调制 1-100
**SRTP** 1-100
**SSUM** 4-23
统计指令
MAVE 5-2
MAXC 5-6
MINC 5-8
MSTD 5-10
**STOD**指令 B-14
字符串转换指令
STOD B-14
字符串转为DINT B-14
结构文本
CASE B-17
结构
ALARM 1-2
DISCRETE_2STATE 1-6
DOMINANT_RESET 6-6
DOMINANT_SET 6-8
FILTER_HIGH_PASS 3-6
FILTER_LOW_PASS 3-18
FILTER_NOTCH 3-24
FLIP_FLOP_D 6-2
FLIP_FLOP_JK 6-4
FUNCTION_GENERATOR 1-34
HL_LIMIT 4-9
INTEGRATOR 2-2
LEAD_LAG 1-37
LEAD_LAG_SEC_ORDER 3-12
MAXIMUM_CAPTURE 5-6
MINIMUM_CAPTURE 5-8
MOVING_AVERAGE 5-2
MOVING_STD_DEV 5-10
MULTIPLEXER 4-12
PIDE_AUTOTUNE 1-53
POSITION_PROP 1-75
PROP_INT 2-8
PULSE_MULTIPLIER 2-20
RAMP_SOAK 1-83
RATE_LIMITER 4-15
S_CURVE 2-28
SCALE 1-96
SEC_ORDER_CONTROLLER 2-38
SELECT 4-19
SELECT_ENHANCED 4-2
SELECTABLE_NEGATE 4-21
SELECTABLE_SUMMER 4-23
UP_DOWN_ACCUM 2-46

**T**

定时方式 A-9
**TOT** 1-106
累加器 1-106

**U**

未解析回路
功能块图 A-5
增/减累加器 2-46
**UPDN** 2-46

# ASCII字符代码

| 字符 | 十进制 | 十六进制 |
|---|---|---|
| [ctrl-@] NUL | 0 | $0 |
| [ctrl-A] SOH | 1 | $1 |
| [ctrl-B] STX | 2 | $2 |
| [ctrl-C] ETX | 3 | $3 |
| [ctrl-D] EOT | 4 | $4 |
| [ctrl-E] ENQ | 5 | $5 |
| [ctrl-F] ACK | 6 | $6 |
| [ctrl-G]BEL | 7 | $7 |
| [ctrl-H] BS | 8 | $8 |
| [ctrl-I] HT | 9 | $9 |
| [ctrl-J] LF | 10 | $l ($0A) |
| [ctrl-K] VT | 11 | $0B |
| [ctrl-L] FF | 12 | $0C |
| [ctrl-M] CR | 13 | $r ($0D) |
| [ctrl-N] SO | 14 | $0E |
| [ctrl-O] SI | 15 | $0F |
| [ctrl-P] DLE | 16 | $10 |
| [ctrl-Q] DC1 | 17 | $11 |
| [ctrl-R] DC2 | 18 | $12 |
| [ctrl-S] DC3 | 19 | $13 |
| [ctrl-T] DC4 | 20 | $14 |
| [ctrl-U] NAK | 21 | $15 |
| [ctrl-V] SYN | 22 | $16 |
| [ctrl-W] ETB | 23 | $17 |
| [ctrl-X] CAN | 24 | $18 |
| [ctrl-Y] EM | 25 | $19 |
| [ctrl-Z] SUB | 26 | $1A |
| ctrl-[ ESC | 27 | $1B |
| [ctrl-\] FS | 28 | $1C |
| ctrl-] GS | 29 | $1D |
| [ctrl-^] RS | 30 | $1E |
| [ctrl-_] US | 31 | $1F |

| 字符 | 十进制 | 十六进制 |
|---|---|---|
| SPACE | 32 | $20 |
| ! | 33 | $21 |
| " | 34 | $22 |
| # | 35 | $23 |
| $ | 36 | $24 |
| % | 37 | $25 |
| & | 38 | $26 |
| ' | 39 | $27 |
| ( | 40 | $28 |
| ) | 41 | $29 |
| * | 42 | $2A |
| + | 43 | $2B |
| "," | 44 | $2C |
| - | 45 | $2D |
| . | 46 | $2E |
| / | 47 | $2F |
| 0 | 48 | $30 |
| 1 | 49 | $31 |
| 2 | 50 | $32 |
| 3 | 51 | $33 |
| 4 | 52 | $34 |
| 5 | 53 | $35 |
| 6 | 54 | $36 |
| 7 | 55 | $37 |
| 8 | 56 | $38 |
| 9 | 57 | $39 |
| : | 58 | $3A |
| ; | 59 | $3B |
| < | 60 | $3C |
| = | 61 | $3D |
| > | 62 | $3E |
| ? | 63 | $3F |

| 字符 | 十进制 | 十六进制 |
|---|---|---|
| @ | 64 | $40 |
| A | 65 | $41 |
| B | 66 | $42 |
| C | 67 | $43 |
| D | 68 | $44 |
| E | 69 | $45 |
| F | 70 | $46 |
| G | 71 | $47 |
| H | 72 | $48 |
| I | 73 | $49 |
| J | 74 | $4A |
| K | 75 | $4B |
| L | 76 | $4C |
| M | 77 | $4D |
| N | 78 | $4E |
| O | 79 | $4F |
| P | 80 | $50 |
| Q | 81 | $51 |
| R | 82 | $52 |
| S | 83 | $53 |
| T | 84 | $54 |
| U | 85 | $55 |
| V | 86 | $56 |
| W | 87 | $57 |
| X | 88 | $58 |
| Y | 89 | $59 |
| Z | 90 | $5A |
| [ | 91 | $5B |
| \ | 92 | $5C |
| ] | 93 | $5D |
| ^ | 94 | $5E |
| _ | 95 | $5F |

| 字符 | 十进制 | 十六进制 |
|---|---|---|
| ` | 96 | $60 |
| a | 97 | $61 |
| b | 98 | $62 |
| c | 99 | $63 |
| d | 100 | $64 |
| e | 101 | $65 |
| f | 102 | $66 |
| g | 103 | $67 |
| h | 104 | $68 |
| i | 105 | $69 |
| j | 106 | $6A |
| k | 107 | $6B |
| l | 108 | $6C |
| m | 109 | $6D |
| n | 110 | $6E |
| o | 111 | $6F |
| p | 112 | $70 |
| q | 113 | $71 |
| r | 114 | $72 |
| s | 115 | $73 |
| t | 116 | $74 |
| u | 117 | $75 |
| v | 118 | $76 |
| w | 119 | $77 |
| x | 120 | $78 |
| y | 121 | $79 |
| z | 122 | $7A |
| { | 123 | $7B |
| \| | 124 | $7C |
| } | 125 | $7D |
| ~ | 126 | $7E |
| DEL | 127 | $7F |

## 罗克韦尔自动化的技术支持

罗克韦尔自动化在网站上提供技术信息帮助用户使用自己的产品。在http://support.rockwellautomation.com网站上，用户可以找到技术手册、FAQs知识库、技术和应用记录、实例代码和与软件补丁包的链接而且用户可以自定义MySupport功能以更好的使用这些工具。

对于附加级别用于安装、组态和诊断的电话技术支持，我们提供TechConnect支持程序。如果需要更多信息，请于本地分销商或罗克韦尔自动化代表联系，或访问网站http://support.rockwellautomation.com网站。

### 安装助理

如果用户在安装硬件模块的最开始的24小时里遇到了问题，请重新阅读该手册中的相关信息。用户还可以联系专用的客户支持电话号码，该号码为用户模块启动和运行提供初始帮助：

| | |
|---|---|
| 美国 | 1.440.646.3223<br>星期一到星期五，早8点到晚5点（美国东部时间） |
| 美国境外 | 任何技术支持问题，请联系当地罗克韦尔自动化代表 |

### 新产品满意返厂

罗克韦尔对全部产品进行测试，以确保它们出厂后可以稳定运行。然而，如果用户的产品不能正常运行并且需要返厂：

| | |
|---|---|
| 美国 | 联系分销商。用户必须向分销商提供一个用户支持代码(使用上面的电话号码来获得一个)以便完成退货过程。 |
| 美国境外 | 请与用户本地罗克韦尔自动化公司代表联系办理返厂手续 |

欢迎访问我们的网址：

**www.rockwellautomation.com.cn**

**www.rockwellautomation.com**

**www.theautomationbookstore.com**




Rockwell Automation Headquarters 1201 South Second Street, Milwaukee, WI 53204, USA, Tel: (1)414 382-2000, Fax: (1)414 382-4444

**香港**－香港数码港道100号数码港3座F区14楼 电话：(852)28874788 传真：(852)25109436

**北京**－北京市建国门内大街18号恒基中心办公楼1座4层 邮编：100005 电话：(8610)65182535 传真：(8610)65182536

**上海**－上海市仙霞路319号远东国际广场A幢7楼 邮编：200051 电话：(8621)62351098 传真：(8621)62351099

**厦门**－厦门市湖里工业区悦华路38号 邮编：361006 电话：(86592)6022084 传真：(86592)6021832

**沈阳**－沈阳市沈河区青年大街219号华新国际大厦15-F单元 邮编：110015 电话：(8624)23961518 传真：(8624)23963539

**武汉**－武汉市青山区和平大道939号13层 邮编：430081 电话：(8627)86543885 传真：(8627)86545529

**广州**－广州市环市东路362号好世界广场2703-04室 邮编：510060 电话：(8620)83849977 传真：(8620)83849989

**重庆**－重庆市渝中区邹容路68号大都会商厦3112-13室 邮编：400010 电话：(8623)63702668 传真：(8623)63702558

**大连**－大连市西岗区中山路147号森茂大厦2305层 邮编：116011 电话：(86411)83687799 传真：(86411)83679970

**西安**－西安市南大街30号中大国际大厦712室 邮编：710002 电话：(8629)7203577 传真：(8629)7203123

**深圳**－深圳市深南东路5047号深圳发展银行大厦15L 邮编：518001 电话：(86755)25847099 传真：(86755)25870900

**南京**－南京市中山南路49号商茂世纪广场44楼A3-A4座 邮编：210005 电话：(8625)86890445 传真：(8625)86890142

**青岛**－青岛市香港中路36号新世界数码港招银大厦1006室 邮编：266071 电话：(86532)6678338 传真：(86532)6678339


Rockwell Automation


**Publication 1756-RM006C-ZH April 2005**



**Copyright 2005 Rockwell Automation Inc. Printed in CHINA.**