繁易自定义工艺库文件使用说明

部门：技术支持部门　　版本：V2.0　　日期：2025-11-30

作者：张金烨

保密等级：☑ ☐

摘　要：

本说明书主要介绍了繁易自定义工艺库中各个 FB 和 FC 的使用，库文件会随着 Package 的安装一并装入 Codesys 的 Library 中。所有功能块都只能下载到我司 PLC 中运行，或者在电脑中处于仿真状态也可以运行，下载到其他厂商的 PLC 中不运行。此外，工程中需要勾选允许使用 Unicode 标识符。

本文档使用硬件设备和软件工具
● 硬件：繁易 FL8、FL6 PLC
● 软件：Codesys V3.5 SP19

文档更新状态：

日期	版本	修订人	修订内容
2024/10/31	V1.0.0	张金烨	初次发版
2025/10/15	V1.1.0	张金烨	① 新增 FB 和 FC ： fbDisableSlave/fbGetFl8Info/ fbFlexemServor/FC_GetDecimaPlaces/ ② 修改功能块名称全部成英文 ③ 所有功能块都增加英文注解
2025/11/28	V2.0	张金烨	整合数学库、基础控制库、通用工艺库

免责声明：

我们已对本文档描述的内容做测试。但是差错在所难免，无法保证绝对正确并完全满足您的使用需求。本文档的内容可能随时更新，如有改动，恕不事先通知，也欢迎您提出改进建议。

上海繁易信息科技股份有限公司

繁易官方：https://flexem.cn/　　技术咨询：4008-033-022 转 1

繁易学院：https://study.flexem.cn/　　邮箱：Training@flexem.com

地址：上海市杨浦区国安路 386 号 INNO 创智 A 栋 9 楼

地址：苏州市虎丘区浒墅关镇青花路 26 号上市科创园二期 4 幢 3 楼

目录

繁易自定义工艺库文件使用说明 1
目录 2
FlexLibrary 7
00 数学类 7
00 浮点数取余/取小数位数/带小数连接 7
Mod_Real 对 REAL 取余取整 7
FC_GetDecimaPlaces 计算小数位数 8
FC_ConnectNumbersWithDecimalPoints 带小数连接数字 9
01 拆分组合 10
011 字与字节拆分组合 10
LHByte_To_Word 10
Word_To_LHByte 11
012 浮点与字拆分组合 12
LHWord_To_Real 12
Real_To_LHWord 14
02 数据转换 15
021 Arr 15
ArrBit_To_ArrWord 15
ArrBit_To_LWord 17
ArrBit_To_Word 19
ArrWord_To_ArrBit 20
Word_To_ArrBit 22
022 Str，ASCII 24
ArrByte_To_ASCIIStr 24
ASCIIStr_To_ArrByte 25
Byte_To_ASCIIStr 26
Word_To_HEXStr 28
023 角度弧度转换 29
DEG_TO_RAD 29
RAD_TO_DEG 30
024 ArrOther 31
ArrSH 31
ArrW_0 33
03_ArrMaxMin 数组求最大最小值 34
ArrMin_ DW/LR/LW/R/W 34
ArrMax_DW/LR/LW/R/W 36
04_数列运算 38
Array_AVG 数列的算术平均值 38
ARRAY_GAV 数列的几何平均值 39
ARRAY_HAV 数列的调和/倒数平均值 40

ARRAY_MAX 数列的最大值 42
ARRAY_MIN 数列的最小值 43
ARRAY_MED 数列的中值 44
ARRAY_VAR 数列的方差 46
ARRAY_SDV 数列的标准差 47
ARRAY_SORTED 数列的升序判断 48
ARRAY_SPR 数列的最大偏差 51
ARRAY_SQL 数列中查询数值 52
ARRAY_SUM 数列的求和 53
ARRAY_ABS 数列的绝对值 54
ARRAY_ADD 数列的偏移 56
ARRAY_MUL 数列的缩放 56
ARRAY_RAN 数列的范围 56
ARRAY_RPT 剔除重复值 56
ARRAY_SQRT 数列的冒泡排序 56
05_直线圆弧相关函数 57
051_圆弧相关 57
FC_Circle_AngleGetEndXY 求圆弧结束点坐标 57
FC_Circle_GetR 求圆半径 58
FC_Circle_XyGetAngle 求圆弧夹角 60
FC_w 角速度计算 61
052_位移速度计算 62
FC_GetPeriodMinL 计算模值下的最短路径 62
FC_VtL 计算一段时间的位移量 63
FC_GetAcc 求加速度 64
FC_ DisplaceTwoSpeedSyn 计算两速产生的位移差 66
FC_GetDisplaceInAcc 计算匀加速阶段的位移和所需时间 67
FC_GetDisplaceInUniform 计算匀速段的位移 69
06_函数与求解类 70
061 求立方根 70
FC_CubicRoot 求立方根 70
062 一元 n 次方程 71
FC_TwoPower 一元二次方程 71
FC_ThreePower 一元三次方程 72
FC_FourPower 一元四次方程 73
FC_FivePower 一元五次方程 74
FC_SevenPower 一元七次方程 76
063 一元 n 次函数导数 77
FC_DerivQuadratic 一元二次导数 77
FC_DerivCubic 一元三次导数 78
FC_DerivQuartic 一元四次导数 79
FC_DerivQuintic 一元五次导数 81
064 解一元 n 次方程 82
FC_2Psolution 解一元二次方程 82

FC_3Psolution 解一元三次方程 84
FC_4Psolution 解一元四次方程 85
FC_5Psolution 解一元五次方程 86
065 正余弦方程 88
sin_calculation 88
cos_calculation 88
solve_cos_function 88
solve_sin_function 88
10 Others 88
FB_LOG_DINT 数据变化记录 88
FunNext 步序加一 89
01 基础控制 90
00 SDO 相关 90
FB_SdoReadLatch SDO 带锁读取 90
FB_SdoWriteLatch SDO 带锁写入 92
01 轴基础控制 94
010_TEC_AxisModel 94
TEC_AxisModel 基础轴控功能块 94
011_单轴控制 101
012_虚轴/实轴基础功能 101
013_单轴单功能控制 102
fbDisableSlave 禁用启用从站 102
FB_Torque_Change 位置模式切换转矩模式功能块 103
MC_JOG_FL 带正反限位的点动 108
MC_MoveTwo 中断定长 110
MC_MoveVelocity_FL 匀速运动变速立刻生效功能 114
02 轴齿轮凸轮控制 117
021 凸轮/齿轮功能合集 117
MC_MoveCam_FL 凸轮功能合集 117
MC_MoveGear_FL 齿轮功能合集 120
022 挺杆 123
FB_AxisSwitch 单个挺杆点 123
FB_AxisSwitch_FL 群挺杆点 124
FB_AxisTappet 挺杆区间 126
023 速度前馈 128
RateFeedForward 从轴速度前馈 128
03 气缸与信号控制相关 131
FB_AirCylinderManualFunction 气缸手动控制 131
FB_AirCylinderMaintainFunction 气缸对应的 IO,延时，报警等维护 134
FC_MappingSignal_1 输入信号常开/常闭/屏蔽处理_1 位 136
FC_MappingSignal_2 输入信号常开/常闭/屏蔽处理_2 位 139
FC_MappingSignal_4 输入信号常开/常闭/屏蔽处理_4 位 141
FC_MappingSignal_6 输入信号常开/常闭/屏蔽处理_6 位 143
FC_MappingSignal_8 输入信号常开/常闭/屏蔽处理_8 位 145

04 通讯相关 148
TCPServer_N 148
TCP_Client_N 152
UDP_N 155
05 滤波类 158
FirstOrderFfiltering 一阶滤波 158
FB_MedianAverageFilter 中位值平均滤波 159
FB_MedianFilter 中位值滤波 159
FB_WeightedRecursiveFilter 加权递推平均滤波法 160
FB_DebounceFilter 消抖滤波法 161
FB_ArithmeticAverageFilter 算术平均滤波 161
FB_RecursiveAverageFilter 递推平均滤波法 162
FB_LimitAverageFilter 限幅平均滤波法 163
FB_LimitDebounceFilter 限幅消抖滤波法 164
FB_LimitFilter 限幅滤波 164
FB_KalmanFilter 卡尔曼滤波 165
06 PID 168
FB_SelfTuningIncrementalPID 自整定 PID 168
10 程序管理类 168
101 产品信息与校验 168
FC_FlexemPLC 判断是否是繁易 PLC 168
fbGetFL8Info 获取 FL8 PLC 信息 168
fbFlexemServor 判断是否是繁易伺服 170
102 时间相关 171
FB_SysTimeGet 时钟获取 171
TimePulse 可变占空比脉冲 172
103 动态分期付款 174
DynamicEncryption 174
02 通用工艺 178
00_插补类 178
FB_ArrLineInterpolation 多轴直线插补 178
FB_CenterAngleCircleInterpolation 两轴圆弧插补（圆心圆弧） 183
FB_EndPosCircleInterpolation 两轴圆弧插补（圆心、结束位置、圈数） 187
FB_HeLixInterpolation 三轴螺旋插补 191
01_凸轮控制切入齿轮 195
FB_FollowGearInPos 特定位置齿轮耦合 195
02_旋切/追剪 198
021 旋切 198
FB_FollowRotaryCutP 飞剪跟随 198
FB_Rotary_Cut_Cam 切原点旋切曲线设置 200
FB_Rotary_Cut_Cam_180 停原点旋切曲线设置 203
022 追剪 204
FB_FollowSyn 追剪跟随 204
FB_Flying_Saw_Cam 追剪曲线设置 207

03_CNC 文件处理 209
04_曲柄滑块结构 209
FB_CenteredCrankSlider 对心曲柄滑块机构的运算 209
fb_UpperCrankSlider_Offset 上偏心曲柄滑块机构的运算 211
FB_LowerCrankSlider_Offset 下偏心曲柄滑块机构的运算 213
05_自定义凸轮 215
051 样条曲线 215
FC_Three/Five/SevenCAM 两点 XYVA 三次/五次/七次曲线运算 215
FC_CosAcc/SinAcc/TvCAM 加速曲线规划（Sin/Cos/T 型） 218
FC_LineCAM 两点 XYVA 直线运算 221
052 跟随实现 221
FB_FollowCam_fl 自定义凸轮主轴求从轴 221
FB_GetFollowCamY 自定义凸轮控制（follow） 223
11_光伏行业 229
FB_CycleCount 循环计数 229
FB_YYX_Plane -XYR 坐标变换 230
FB_SilkScreenPrintingCalculate 刮刀压力计算 231

FlexLibrary

00 数学类

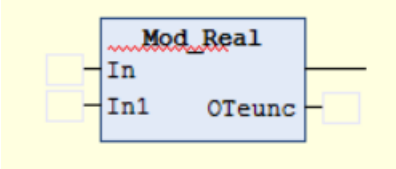
00 浮点数取余/取小数位数/带小数连接

Mod_Real 对 REAL 取余取整

函数功能：对 Real 数据进行取余数和取整运算

1、指令格式

图形表现：



ST 表现：

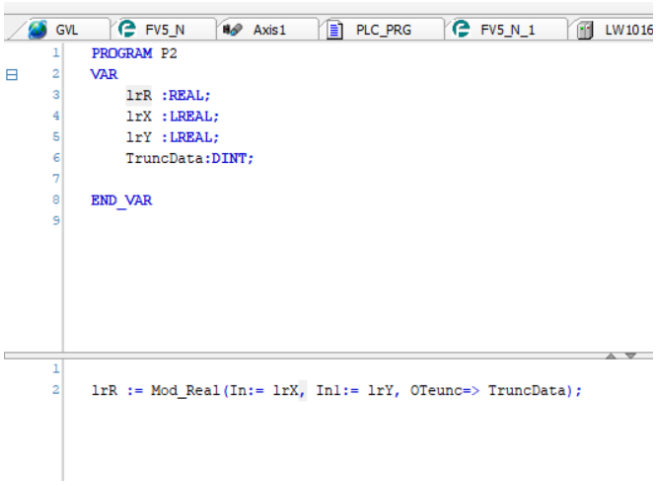
```
Mod_Real(In:= , In1:= , OTeunc=> );
```

2、相关变量

变量名称	名称	数据类型	变量属性	描述
In	被除数	LREAL	输入变量	被除数值
In1	除数	LREAL	输入变量	除数值
OTeunc	商取整	DINT	输出变量	商的整数部分
Mod_Real	余数	REAL	输出变量	余数值

3、具体使用

（1）声明和函数调用



（2）写值与调用

Device.Application.P2

表达式	类型	值	准备值	地址	注释
lrR	REAL	1			
lrX	LREAL	33			
lrY	LREAL	2			
TruncData	DINT	16			

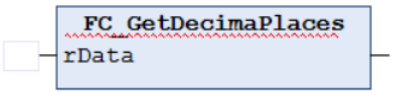
```
1
2   lrR 1 := Mod_Real(In:= lrX 33 , In1:= lrY 2 , OTeunc=> TruncData 16 );RETURN
```

FC_GetDecimaPlaces 计算小数位数

函数名称：FC_GetDecimaPlaces　用于计算小数位数

1、指令格式

图形表现：



ST 表现：

```
FC_GetDecimaPlaces(rData:= );
```

2、相关变量

变量名称	名称	数据类型	变量属性	描述
rData	需要判断的数据	REAL	输入变量	需要判断的数据
FC_GetDecimaPlaces	返回的小数位数	INT	输出变量	小数位数

3、声明和函数调用

```
PROGRAM PLC_PRG
VAR
    r1:REAL;
    iResult :INT;
END_VAR
```

```
iResult:= FC_GetDecimaPlaces(rData:= r1);
```

写值与监控:

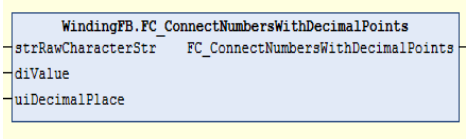
```
iResult 4 := FC_GetDecimaPlaces(rData:= r1 1.23 );
```

FC_ConnectNumbersWithDecimalPoints 带小数连接数字

函数名称: FC_ ConnectNumbersWithDecimalPoints 用于带小数连接整数拼接

1、指令格式

图形表现:



ST 表现:

```
WindingFB.FC_ConnectNumbersWithDecimalPoints(
strRawCharacterStr:= ,
diValue:= ,
uiDecimalPlace:= );
```

2、相关变量

变量名称	名称	数据类型	变量属性	描述
strRawCharacterStr	原字符串	String	输入变量	原字符串
diValue	数值	DINT	输入变量	数值
uiDecimalPlace	小数位	UINT	输入变量	小数位数

3、声明和函数调用

```
PROGRAM FC_ConnectNumbersWithDecimalPoints
VAR
    OUT:STRING(255);
    strRawCharacterStr          : STRING(255) ;//str原字符串 : STRING(255) ;
    diValue                     : DINT ;//di数值 : DINT ;
    uiDecimalPlace              : UINT ;//ui小数位 : UINT ;
END_VAR
```

```
OUT:=WindingFB.FC_ConnectNumbersWithDecimalPoints(
strRawCharacterStr:= strRawCharacterStr,
diValue:= diValue,
uiDecimalPlace:= uiDecimalPlace
);
```

写值与监控:

例: strRawCharacterStr: ='123', diValue:=123 , uiDecimalPlace: =4 时, 拼接返回值是 123.0123

strRawCharacterStr 拼接 diValue , 小数位左移 uiDecimalPlace 位

表达式	类型	值	准备值	地	注释
OUT	STRIN...	'1230.0123'			
strRawCharacterStr	STRIN...	'123'			str原字符串：STRING(255)；
diValue	DINT	123			di数值：DINT；
uiDecimalPlace	UINT	4			ui小数位：UINT；

```
1  OUT '1230.0123' :=WindingFB.FC_ConnectNumbersWithDecimalPoints(
2  strRawCharacterStr:= strRawCharacterStr '123' ,
3  diValue:= diValue 123 ,
4  uiDecimalPlace:= uiDecimalPlace 4
5  );
6
7  RETURN
```

01 拆分组合

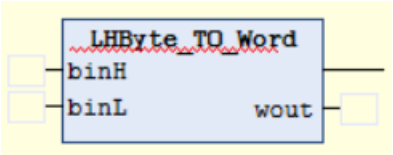
011 字与字节拆分组合

LHByte_To_Word

函数功能：两个 Byte 数据分别存储 Word 数据的高低位，拼成一个 WORD 输出

1、指令格式

图形表现



ST 表现

```
LHByte_TO_Word(binH:= , binL:= , wout=> );
```

2、相关变量

变量名称	名称	数据类型	变量属性	描述
binH	高 8 位	Byte	输入变量	Byte 数据存 WORD 的高 8 位
binL	低 8 位	Byte	输入变量	Byte 数据存 WORD 的低 8 位
wout	输出的 WORD 变量	WORD	输出变量	输出 WORD 变量最终的值

3、具体使用

（1）声明和函数调用

```
    wData_Byte:WORD;
    bByte_Word_H:BYTE;
    bByte_Word_L:BYTE;
END_VAR
```

```
LHByte_TO_Word(binH:=bByte_Word_H , binL:= bByte_Word_L, wout=> wData_Byte);
```

（2）给值测试

wData_Byte	WORD	3073			
bByte_Word_H	BYTE	12			
bByte_Word_L	BYTE	1			

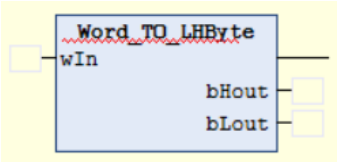
```
1
2
3
4   LHByte_TO_Word(binH:=bByte_Word_H 12 , binL:= bByte_Word_L 1 , wout=> wData_Byte 3073 );RETURN
```

Word_To_LHByte

函数功能：输入一个 WORD 数据，按照高 8 位和低 8 位输出到两个 Byte 变量

1、指令格式

图形表现



ST 表现

```
Word_TO_LHByte(wIn:= , bHout=> , bLout=> );
```

2、相关变量

变量名称	名称	数据类型	变量属性	描述
wIn	输入 Word 的值	WORD	输入变量	输入 Word 变量
bHout	输出高 8	Byte	输出变量	把 word 变量拆分，存高 8 位
bLout	输出低 8	Byte	输出变量	把 word 变量拆分，存低 8 位

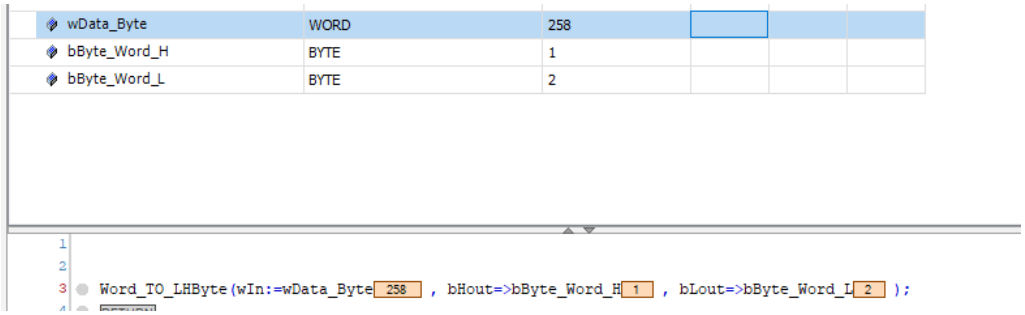
3、具体使用

（1）声明和函数调用

```
    wData_Byte:WORD;
    bByte_Word_H:BYTE;
    bByte_Word_L:BYTE;
END_VAR
```

```
Word_TO_LHByte(wIn:=wData_Byte , bHout=>bByte_Word_H , bLout=>bByte_Word_L )
```

（2）给值和输出。Word 当前值 258，输出高位是 1 低位是 2



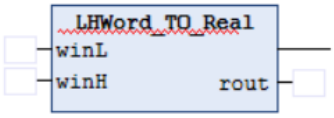
012 浮点与字拆分组合

LHWord_To_Real

函数功能：按照高 16 位和低 16 位输入两个 WORD，计算后输出到 1 个 REAL 变量

1、指令格式

图形表现



ST 表现

```
LHWord_TO_Real(winL:= , winH:= , rout=> );
```

2、相关变量

变量名称	名称	数据类型	变量属性	描述
wInL	输入低 16 位	WORD	输入变量	用 WORD 存 REAL 数据的低 16 位
winH	输入高 16 位	WORD	输入变量	用 WORD 存 REAL 数据的高 16 位

rOut	输出 REAL	REAL	输出变量	计算后，输出 REAL 当前值

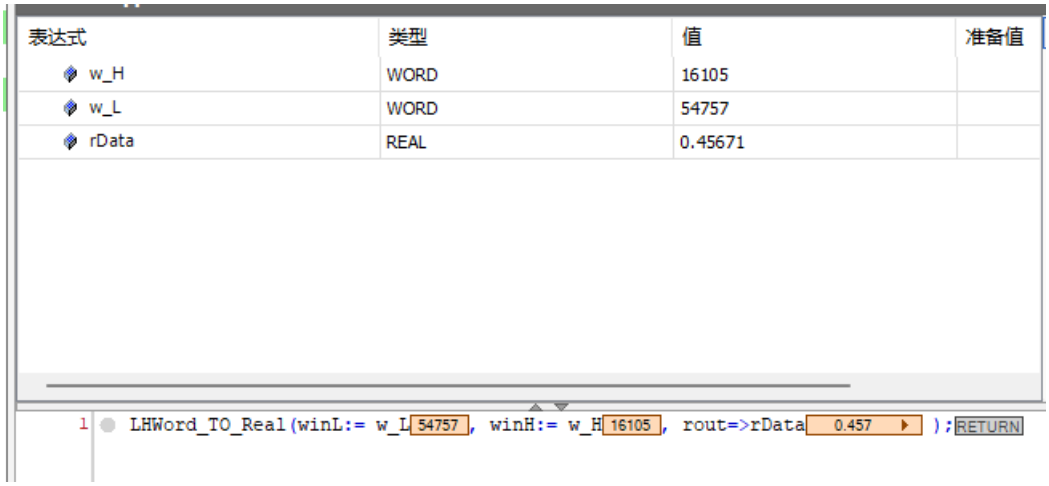
3、具体使用

（1）声明和函数调用

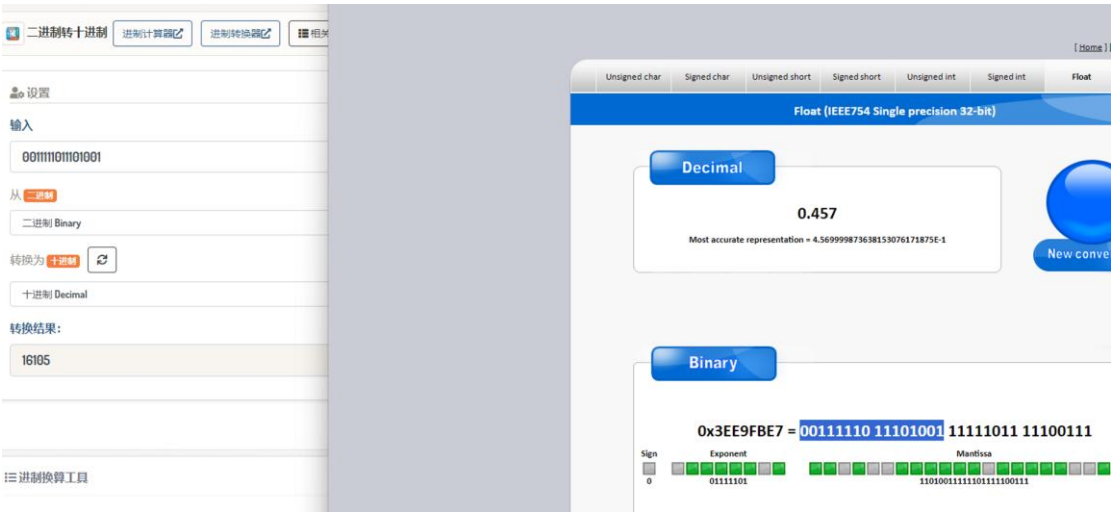
```
PROGRAM P2
VAR

    w_H:WORD;
    w_L:WORD;
    rData:REAL;

END_VAR
```

```
LHWord_TO_Real(winL:= w_L, winH:= w_H, rout=>rData );
```

（2）给值和输出。Word 当前值 258，输出高位是 1 低位是 2



（3）工具验证

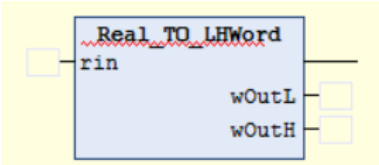


Real_To_LHWord

函数功能：输入一个 REAL 变量，按照高 16 和低 16 位分别存入一个 WORD

1、指令格式

图形表现



ST 表现

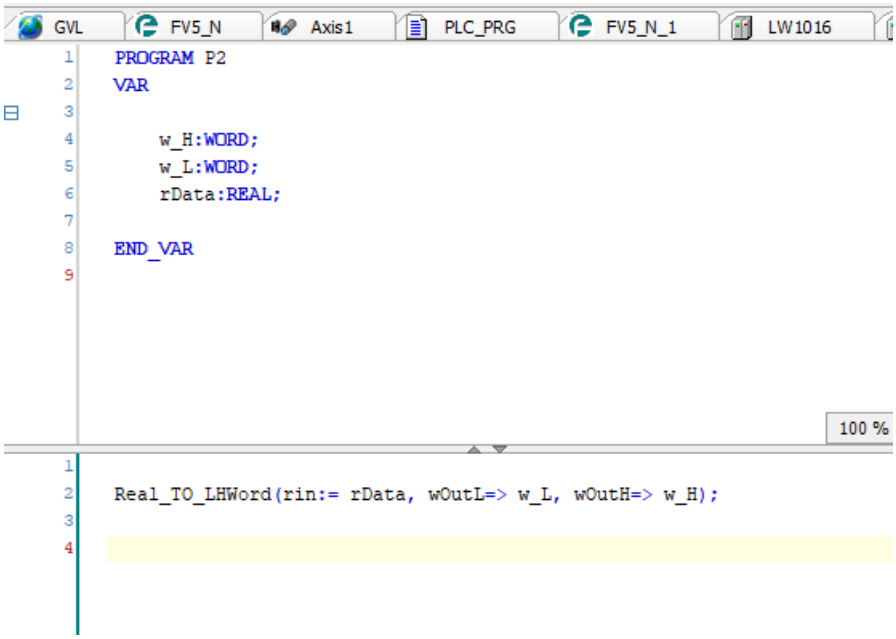
```
Real_TO_LHWord(rin:= , wOutL=> , wOutH=> );
```

2、相关变量

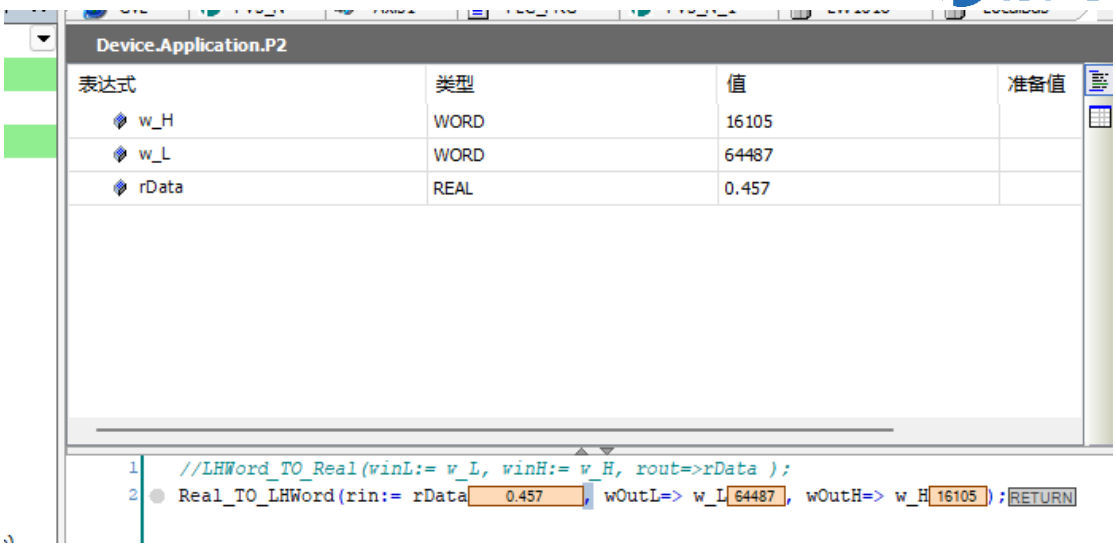
变量名称	名称	数据类型	变量属性	描述
Rin	输入 REAL 值	REAL	输入变量	输入 REAL 值
wOutL	拆分出低 16 位	WORD	输出变量	用 WORD 存 REAL 的低 16 位
wOutH	拆分出高 16 位	WORD	输出变量	用 WORD 存 REAL 的高 16 位

3、具体使用

（1）声明和函数调用



（1） 写值与监控



02 数据转换

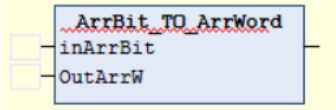
021 Arr

ArrBit_To_ArrWord

函数功能：数组形式输入 bool，16 个 BOOL 组成一个 Word，存入一个 Word 数组

1、指令格式

图形表现：



ST 表现：

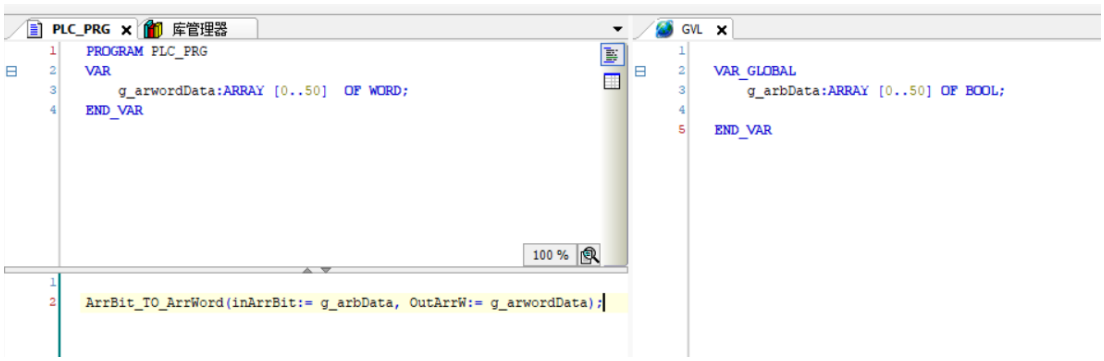
```
ArrBit_TO_ArrWord(inArrBit:= , OutArrW:= );
```

2、相关变量

变量名称	名称	数据类型	变量属性	描述
inArrBit	Bit 数组	ARRAY[*] OF BOOL	输入变量	Bit 数组，一个 Word 对应 16 个 bit
OutArrW	Word 数组	ARRAY[*] OF WORD	输出变量	Word 数据数组

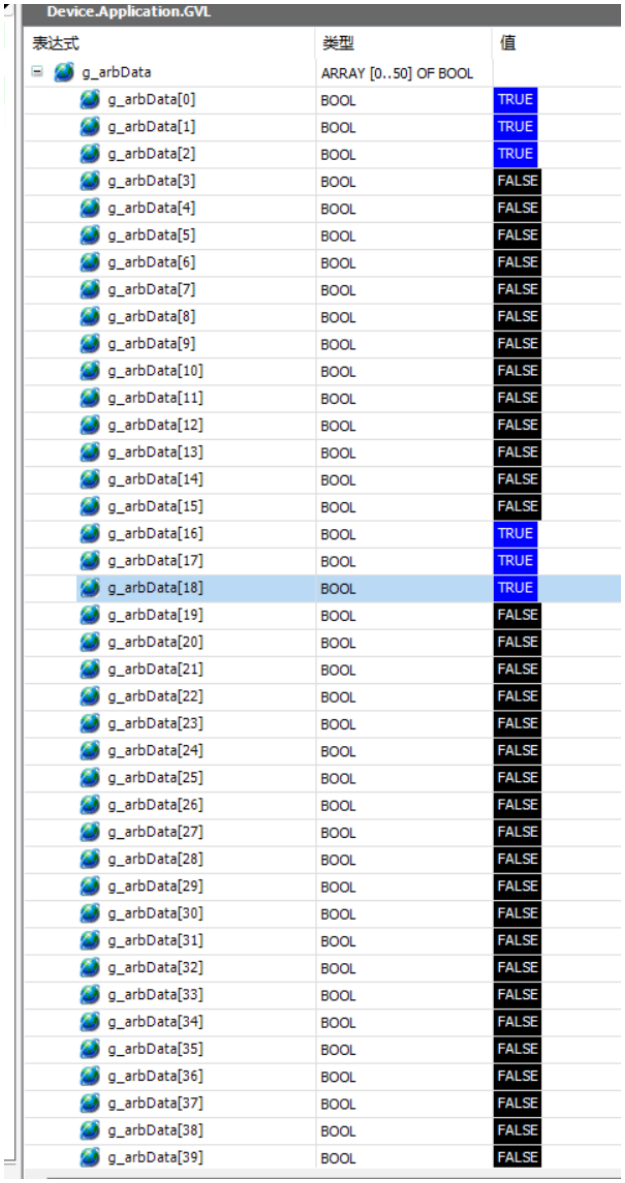
3、具体使用

（1）声明和函数调用



（2）写值与监控（BOOL 数组的 0/1/2 和 16/17/18 均写值为 TRUE，转换成 Word 为 7）

g_arwordData	ARRAY [0..50] OF WORD	
g_arwordData[0]	WORD	7
g_arwordData[1]	WORD	7
g_arwordData[2]	WORD	0



Device.Application.GVL

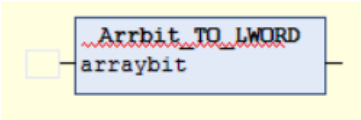
表达式	类型	值
g_arbData	ARRAY [0..50] OF BOOL	
g_arbData[0]	BOOL	TRUE
g_arbData[1]	BOOL	TRUE
g_arbData[2]	BOOL	TRUE
g_arbData[3]	BOOL	FALSE
g_arbData[4]	BOOL	FALSE
g_arbData[5]	BOOL	FALSE
g_arbData[6]	BOOL	FALSE
g_arbData[7]	BOOL	FALSE
g_arbData[8]	BOOL	FALSE
g_arbData[9]	BOOL	FALSE
g_arbData[10]	BOOL	FALSE
g_arbData[11]	BOOL	FALSE
g_arbData[12]	BOOL	FALSE
g_arbData[13]	BOOL	FALSE
g_arbData[14]	BOOL	FALSE
g_arbData[15]	BOOL	FALSE
g_arbData[16]	BOOL	TRUE
g_arbData[17]	BOOL	TRUE
g_arbData[18]	BOOL	TRUE
g_arbData[19]	BOOL	FALSE
g_arbData[20]	BOOL	FALSE
g_arbData[21]	BOOL	FALSE
g_arbData[22]	BOOL	FALSE
g_arbData[23]	BOOL	FALSE
g_arbData[24]	BOOL	FALSE
g_arbData[25]	BOOL	FALSE
g_arbData[26]	BOOL	FALSE
g_arbData[27]	BOOL	FALSE
g_arbData[28]	BOOL	FALSE
g_arbData[29]	BOOL	FALSE
g_arbData[30]	BOOL	FALSE
g_arbData[31]	BOOL	FALSE
g_arbData[32]	BOOL	FALSE
g_arbData[33]	BOOL	FALSE
g_arbData[34]	BOOL	FALSE
g_arbData[35]	BOOL	FALSE
g_arbData[36]	BOOL	FALSE
g_arbData[37]	BOOL	FALSE
g_arbData[38]	BOOL	FALSE
g_arbData[39]	BOOL	FALSE

ArrBit_To_LWord

函数功能：输入不定长的 BOOL 数组，根据每一位的值，存入一个 LWord 类型数据

1、指令格式

图形表现：



ST 表现：

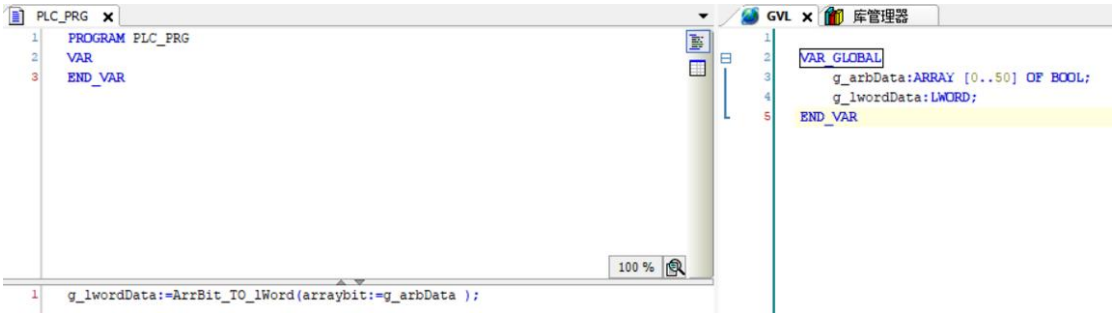
```
Arrbit_TO_LWORD(arraybit:= );
```

2、相关变量

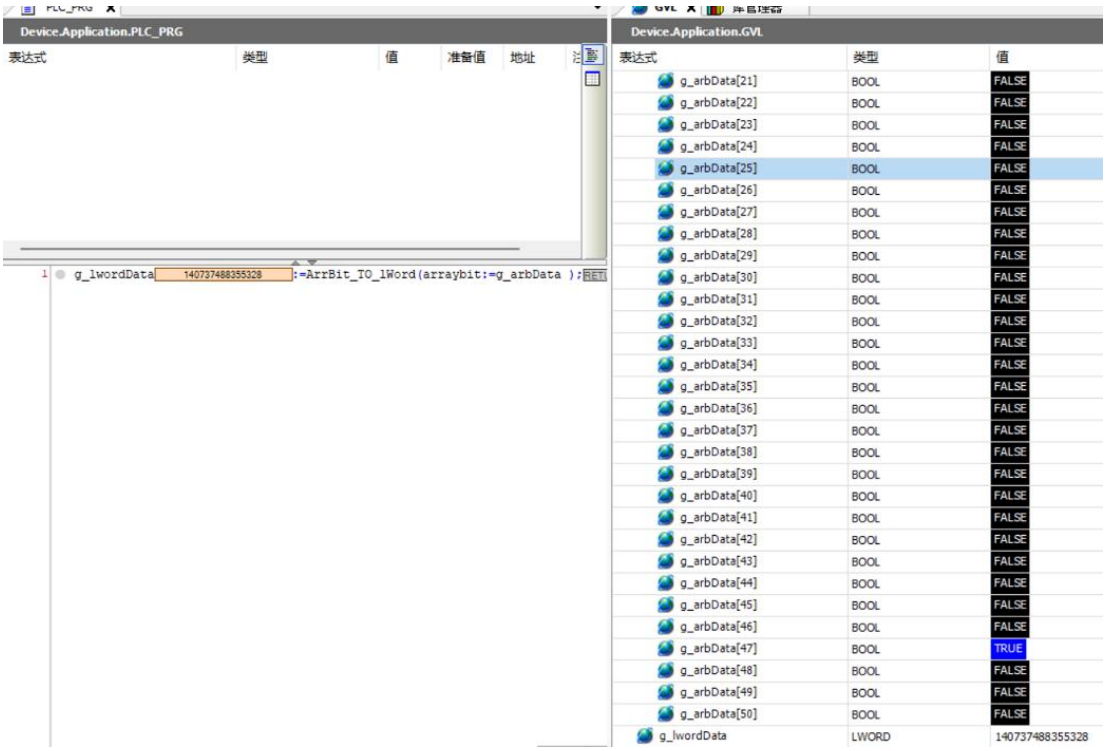
变量名称	名称	数据类型	变量属性	描述
inArrBit	Bit 数组	ARRAY[*] OF BOOL	输入变量	用 Bit 数组来描述一个 LWord 的每一位
Arrbit_To_LWORD	LWORD	LWORD	输出变量	LWord 数据

3、具体使用

（1）声明和函数调用



（2）写值与监控

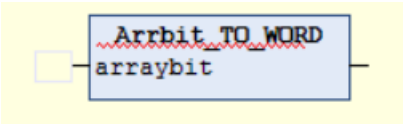


ArrBit_To_Word

函数功能：输入不定长的 BOOL 数组，根据每一位的值，存入一个 Word 类型数据

1、指令格式

图形表现：



ST 表现：

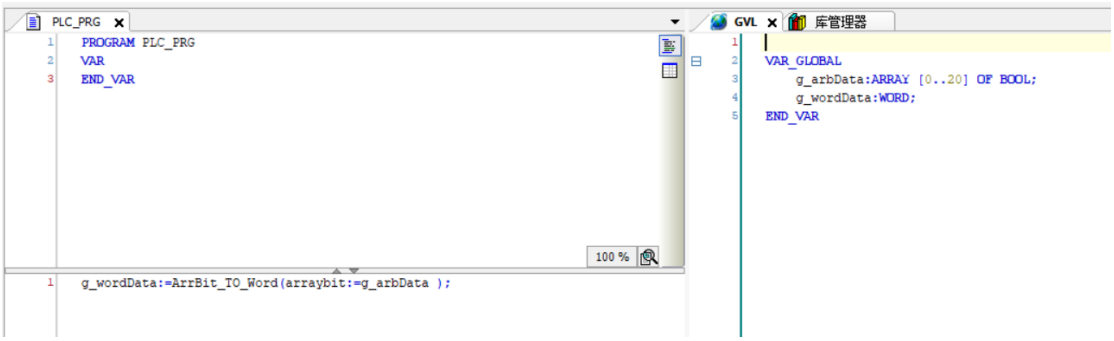
```
Arrbit_TO_WORD(arraybit:= );
```

2、相关变量

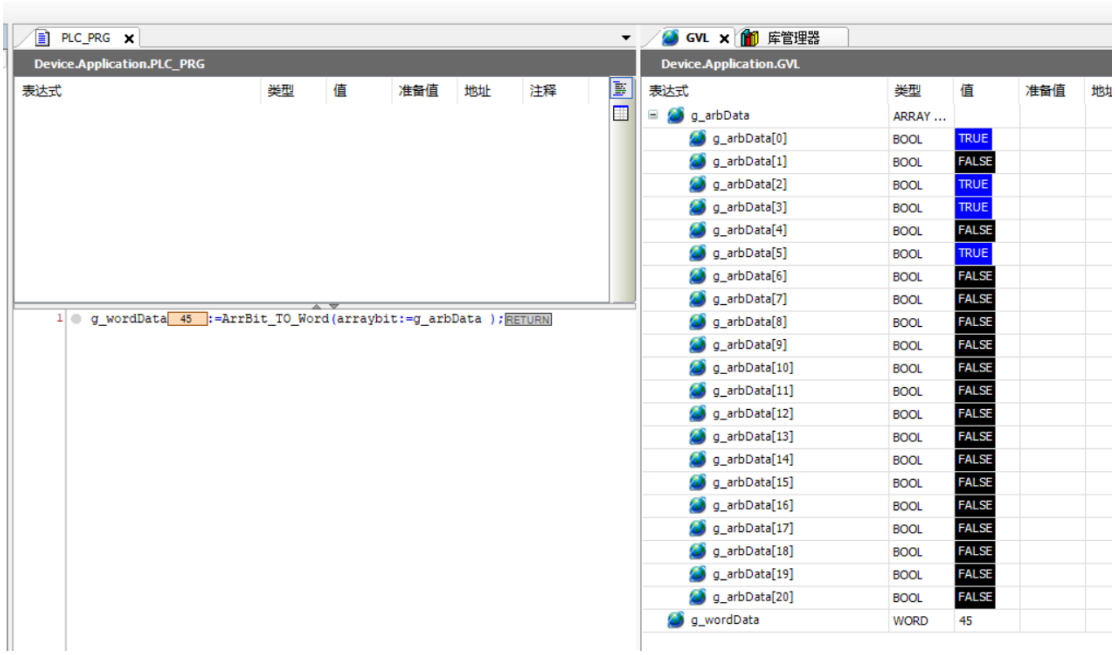
变量名称	名称	数据类型	变量属性	描述
inArrBit	Bit 数组	ARRAY[*] OF BOOL	输入变量	用 Bit 数组来描述一个 LWord 的每一位
Arrbit_To_WORD	WORD 数据	WORD	输出变量	Word 数据

3、具体使用

（1）声明和函数调用



（2）写值与调用（45=2#0010 1101）

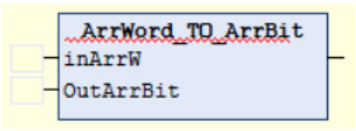


ArrWord_To_ArrBit

函数功能：Word 数组数据，按位处理转换成 BOOL 数组数据

1、指令格式

图形表现：



ST 表现：

```
ArrWord_TO_ArrBit(inArrW:= , OutArrBit:= );
```

2、相关变量

变量名称	名称	数据类型	变量属性	描述
inArrW	Word 数组数据	ARRAY[*] OF Word	输入输出变量	Word 数组数据
OutArrBit	转换后的 BOOL 数组	ARRAY[*] OF BOOL	输入输出变量	BOOL 数组数据

3、具体使用

（1）声明和函数调用

```
PROGRAM P2
VAR

    arW :ARRAY[0..3] OF WORD;
    arB :ARRAY[0..100] OF BOOL;
END_VAR
```

```
ArrWord_TO_ArrBit(inArrW:= arW, OutArrBit:= arB);
```

（2）写值与监控

A_inword4	ARRAY [0..3] OF WORD	
A_inword4[0]	WORD	2
A_inword4[1]	WORD	255
A_inword4[2]	WORD	65535
A_inword4[3]	WORD	0

A_outarrbit64	ARRAY [0..63] OF BOOL	
A_outarrbit64[0]	BOOL	FALSE
A_outarrbit64[1]	BOOL	TRUE
A_outarrbit64[2]	BOOL	FALSE
A_outarrbit64[3]	BOOL	FALSE
A_outarrbit64[4]	BOOL	FALSE
A_outarrbit64[5]	BOOL	FALSE
A_outarrbit64[6]	BOOL	FALSE
A_outarrbit64[7]	BOOL	FALSE
A_outarrbit64[8]	BOOL	FALSE
A_outarrbit64[9]	BOOL	FALSE
A_outarrbit64[10]	BOOL	FALSE
A_outarrbit64[11]	BOOL	FALSE
A_outarrbit64[12]	BOOL	FALSE
A_outarrbit64[13]	BOOL	FALSE
A_outarrbit64[14]	BOOL	FALSE
A_outarrbit64[15]	BOOL	FALSE

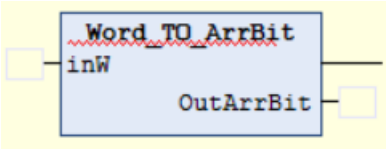
A_outarrbit64[16]	BOOL	TRUE
A_outarrbit64[17]	BOOL	TRUE
A_outarrbit64[18]	BOOL	TRUE
A_outarrbit64[19]	BOOL	TRUE
A_outarrbit64[20]	BOOL	TRUE
A_outarrbit64[21]	BOOL	TRUE
A_outarrbit64[22]	BOOL	TRUE
A_outarrbit64[23]	BOOL	TRUE
A_outarrbit64[24]	BOOL	FALSE
A_outarrbit64[25]	BOOL	FALSE
A_outarrbit64[26]	BOOL	FALSE
A_outarrbit64[27]	BOOL	FALSE
A_outarrbit64[28]	BOOL	FALSE
A_outarrbit64[29]	BOOL	FALSE
A_outarrbit64[30]	BOOL	FALSE
A_outarrbit64[31]	BOOL	FALSE

Word_To_ArrBit

函数功能：Word 变量转换成 16 个 BOOL 存入数组

1、指令格式

图形表现：



ST 表现：

```
Word_TO_ArrBit(inW:= , OutArrBit=> );
```

2、相关变量

变量名称	名称	数据类型	变量属性	描述
inW	Word 数据	Word	输入输出变量	Word 数组数据
OutArrBit	转换后的 BOOL 数组	ARRAY[0..15] OF BOOL	输入输出变量	BOOL 数组数据

3、具体使用

（1）声明和函数调用

PLC_PRG | 库管理器 | GVL | P2

```
PROGRAM P2
VAR
    wData: WORD;
    arBool: ARRAY[0..15] OF BOOL;
END_VAR
```

```
Word_TO_ArrBit(inW:= wData, OutArrBit=> arBool);
```

（2）写值与监控

表达式	类型	值	准备值	地址	注释
wData	WORD	15			
arBool	ARRAY [0..15] OF BOOL				
arBool[0]	BOOL	TRUE			
arBool[1]	BOOL	TRUE			
arBool[2]	BOOL	TRUE			
arBool[3]	BOOL	TRUE			
arBool[4]	BOOL	FALSE			
arBool[5]	BOOL	FALSE			
arBool[6]	BOOL	FALSE			
arBool[7]	BOOL	FALSE			
arBool[8]	BOOL	FALSE			
arBool[9]	BOOL	FALSE			
arBool[10]	BOOL	FALSE			
arBool[11]	BOOL	FALSE			
arBool[12]	BOOL	FALSE			
arBool[13]	BOOL	FALSE			
arBool[14]	BOOL	FALSE			
arBool[15]	BOOL	FALSE			

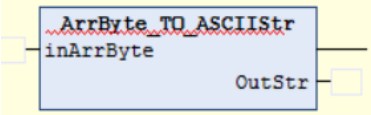
```
Word_TO_ArrBit(inW:= wData 15 , OutArrBit=> arBool);RETURN
```

ArrByte_To_ASCIIStr

函数功能：Byte 数组输入 28 个，转成 ASCII 输出字符串

1、指令格式

图形表现：



ST 表现：

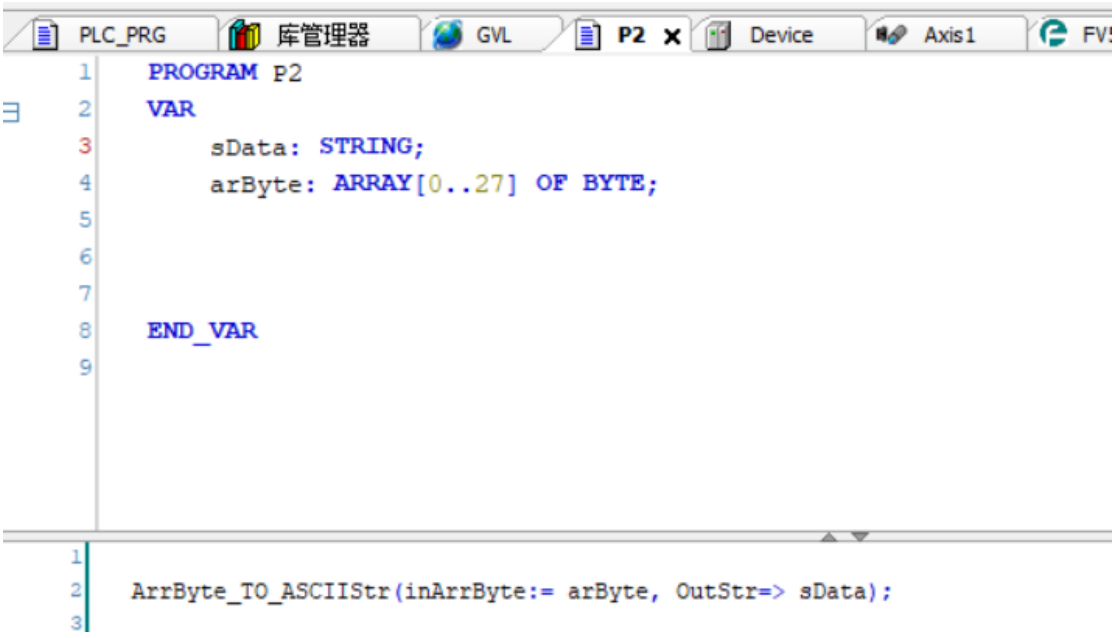
```
ArrByte_TO_ASCIIStr(inArrByte:= , OutStr=> );
```

2、相关变量

变量名称	名称	数据类型	变量属性	描述
inArrByte	输入 Byte 数组	ARRAY [0..27] OF BYTE	输入变量	Byte 数组数据
OutStr	输出 String	STRING(28)	输出变量	String 数据

3、具体使用

（1）声明和函数调用



（2）写值与调用

sData	STRING	'$01$03$04'
arByte	ARRAY [0..27] OF BYTE	
arByte[0]	BYTE	1
arByte[1]	BYTE	3
arByte[2]	BYTE	4
arByte[3]	BYTE	0
arByte[4]	BYTE	0
arByte[5]	BYTE	0
arByte[6]	BYTE	0
arByte[7]	BYTE	0
arByte[8]	BYTE	0
arByte[9]	BYTE	0
arByte[10]	BYTE	0
arByte[11]	BYTE	0
arByte[12]	BYTE	0

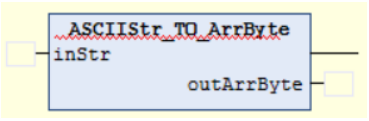
```
1
2   ArrByte_TO_ASCIIStr(inArrByte:= arByte, OutStr=> sData '$01$03$04' );
3
4   RETURN
```

ASCIIStr_To_ArrByte

函数功能：一个 ASCII 码字符串转换成多个 Byte 存入数组

1、指令格式

图形表现：



ST 表现：

```
ASCIIStr_TO_ArrByte(inStr:= , outArrByte=> );
```

2、相关变量

变量名称	名称	数据类型	变量属性	描述
inStr	输入 String	String(28)	输入变量	String 数据
OutArrByte	输出 Byte	ARRAY [0..27] OF BYTE	输出变量	Byte 数组数据

3、具体使用

（1）声明和函数调用

```
PROGRAM P2
VAR
    sData: STRING;
    arByte: ARRAY[0..27] OF BYTE;


END_VAR
```

```


ASCIIStr_TO_ArrByte(inStr:= sData, outArrByte=>arByte )
```

（2）写值与监控

Device.Application.P2

表达式	类型	值	准备值	地址	注释
sData	STRING	'ABCS'			
arByte	ARRAY [0..27] OF BYTE				
arByte[0]	BYTE	65			
arByte[1]	BYTE	66			
arByte[2]	BYTE	67			
arByte[3]	BYTE	83			
arByte[4]	BYTE	0			
arByte[5]	BYTE	0			
arByte[6]	BYTE	0			
arByte[7]	BYTE	0			
arByte[8]	BYTE	0			
arByte[9]	BYTE	0			

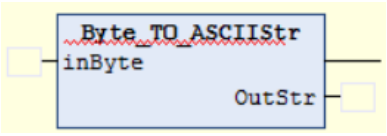
```


ASCIIStr_TO_ArrByte(inStr:= sData 'ABCS' , outArrByte=>arByte );RETURN
```

Byte_To_ASCIIStr

函数功能：单个 byte 转换成 ASCII 码输出字符

1、指令格式

图形表现：　　ST 表现：




```
Byte_TO_ASCIIStr(inByte:= , OutStr=> );
```

2、相关变量

变量名称	名称	数据类型	变量属性	描述
inByte	输入 Byte 数据	Byte	输入变量	Byte 数据
OutStr	输出 String 数据	String(1)	输出变量	String 数据

3、具体使用

（1）声明和函数调用

```
PROGRAM P2
VAR
    sData: STRING;

    bByte:  BYTE;

END_VAR
```

```


Byte_TO_ASCIIStr(inByte:=bByte , OutStr=>sData );
```

（2）写值与监控



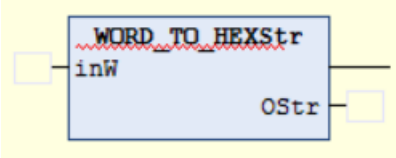
备注：ASC 码表：https://www.hunanit.net/asciicode/

Word_To_HEXStr

函数功能：十进制的数，以 16 进制用字符显示

1、指令格式

图形表现：



ST 表现：

```
WORD_TO_HEXStr(inW:= , OStr=> );
```

2、相关变量

变量名称	名称	数据类型	变量属性	描述
inW	输入 Word 数据	WORD	输入变量	Byte 数据
OStr	输出 String 数据	String	输出变量	String 数据显示 16 进制值

3、具体使用

（1）声明和函数调用

```
PROGRAM P2
VAR
    sData: STRING;

    wData:  WORD;

END_VAR
```

```
WORD_TO_HEXStr(inW:= wData, OStr=>sData );
```

（2）写值与监控



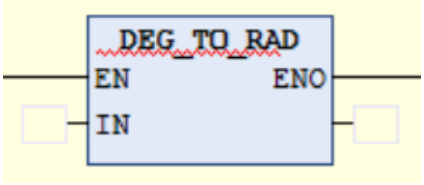
023 角度弧度转换

DEG_TO_RAD

函数功能：角度值转弧度值

1、指令格式

图形表现：　　ST 表现：



RAD_TO_DEG(IN:=);

2、相关变量

变量名称	名称	数据类型	变量属性	描述
IN	数据	LREAL	输入变量	需要被运算的数据

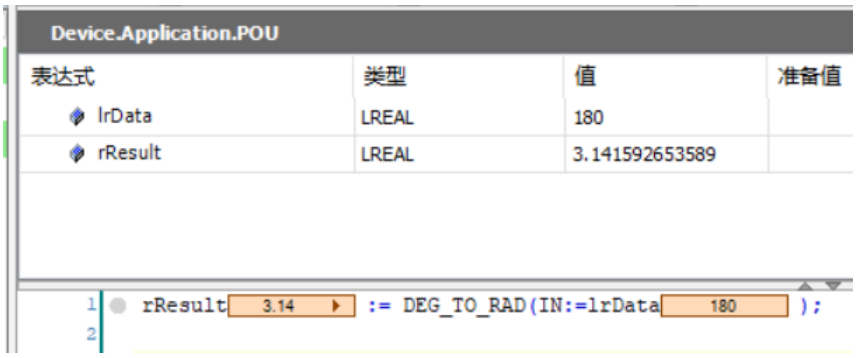
3、具体使用

（1）声明和函数调用

```
PROGRAM POU
VAR
    lrData  :LREAL;
    rResult :LREAL;
END_VAR
```

```
rResult := DEG_TO_RAD(IN:=lrData );
```

（2） 写值与监控



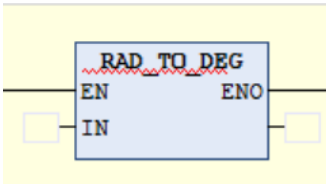
RAD_TO_DEG

函数功能：弧度值转角度值

1、指令格式

图形表现：

ST 表现：



RAD_TO_DEG(IN:=);

2、相关变量

变量名称	名称	数据类型	变量属性	描述
IN	数据	LREAL	输入变量	需要被运算的数据

3、具体使用

（1）声明和函数调用

```
PROGRAM POU
VAR
    lrData  :LREAL;
    rResult :LREAL;
END_VAR
```

```
rResult := RAD_TO_DEG(IN:=lrData );
```

（2）写值与监控

Device.Application.POU

表达式	类型	值	准备值
lrData	LREAL	3.14159	
rResult	LREAL	179.99984796054972	

```
1  rResult 180 := RAD_TO_DEG(IN:=lrData 3.14 );
```

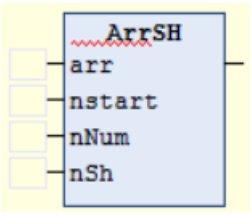
024 ArrOther

ArrSH

函数功能：数组数据的左移或者右移；设定从哪个位置开始移动 一共要移动多少个；左右移动，移动间距都通过 D_nSh 调整

1、指令格式

图形表现：



ST 表现：

```
ArrSH(
    arr:= ,
    nstart:= ,
    nNum:= ,
    nSh:= );
```

2、相关变量

变量名称	名称	数据类型	变量属性	描述
Arr	Word 数组数据	ARRAY[*] OF Word	输入输出变量	Word 数组数据
nStart	移动开始的位置	UINT	输入变量	移动开始的位置
nNum	移动个数	UINT	输入变量	移动的个数
nSh	移动的步距	INT	输入变量	移动步距，大于 0 正向移动，小于 0 反向移动

3、具体使用

（1）声明和函数调用

```
PROGRAM PLC_PRG
VAR
    D_ArrSH:ARRAY[0..99] OF WORD;
    D_nstart:UINT:=0;    //移动开始位置
    D_nNum:UINT:=100;    //移动个数
    D_nSh:INT:=1;        //移动步距 ,>0正向移动 小于0反向移动

END_VAR
```

```

ArrSH(arr:=D_ArrSH , nstart:=D_nstart , nNum:=D_nNum , nSh:=D_nSh );
```

（2）写值与调用

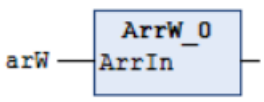


ArrW_0

函数功能：数组清零

1、指令格式

图形表现：



ST 表现：

```
ArrW_0(ArrIn:= arW);
```

2、相关变量

变量名称	名称	数据类型	变量属性	描述

ArrIn	数组	ARRAY[*] OF Word	输入输出变量	需要被清空的 Word 数组数据

3、具体使用

（1）声明和函数调用

```
PROGRAM P2
VAR

    arW :ARRAY[0..3] OF WORD;

END_VAR
```

```
ArrW_0(ArrIn:= arW);
```

（2）写值与监控

表达式	类型	值
arW	ARRAY [0..3] ...	
arW[0]	WORD	0
arW[1]	WORD	0
arW[2]	WORD	0
arW[3]	WORD	0

```
ArrW_0(ArrIn:= arW);
```

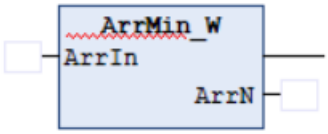
03_ArrMaxMin 数组求最大最小值

ArrMin_ DW/LR/LW/R/W

函数功能：输入任意长度的数组，找出这个数组的最小值，输出最小值所在的数组下标，返回最小值当前值。若最小值相同，输出较小的数组下标。例如 arA[1]=arA[10]=arA[12]=20，20 是最小值；ArrN 输出 1。

1、指令格式

图形表现　　ST 表现




```
ArrMin_W(ArrIn:= , ArrN=> );
```

2、相关变量

变量名称	名称	数据类型	变量属性	描述
ArrIn	输入数组	数组	输入变量	需要查找的数组
ArrN	最大值下标	WORD	输出变量	数组最小值所在下标

3、具体使用

```
PROGRAM P2
VAR
    wData: ARRAY[0..100] OF WORD;
    wMaxDataNum:WORD;
    wMaxData:WORD;

    wMiniDataNum:WORD;
    wMiniData:WORD;
END_VAR
```

```
wMaxData:=ArrMax_W(ArrIn:= wData, ArrN=> wMaxDataNum);
wMiniData:=ArrMin_W(ArrIn:= wData, ArrN=> wMiniDataNum);
```

Device.Application.P2

表达式	类型	值	准备值	地址	注释
wData	ARRAY [0..100] OF WO...				
wData[0]	WORD	100			
wData[1]	WORD	200			
wData[2]	WORD	10			
wData[3]	WORD	0			
wData[4]	WORD	0			
wData[5]	WORD	0			
wData[6]	WORD	0			
wData[7]	WORD	0			
wData[8]	WORD	0			
wData[9]	WORD	0			

```
wMaxData 200 :=ArrMax_W(ArrIn:= wData, ArrN=> wMaxDataNum 1 );
wMiniData 0 :=ArrMin_W(ArrIn:= wData, ArrN=> wMiniDataNum 3 );RETURN
```

4、数据类型

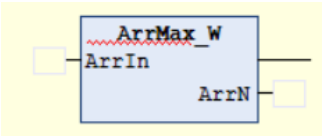


ArrMax_DW/LR/LW/R/W

函数功能：数组找最大值。输出最大值所在的数组下标和当前数组最大值是多少。若最大值相同，输出较小的数组下标。例如 arA[1]=arA[10]=arA[12]=888；ArrN 输出 1

1、指令格式

图形表现 ST 表现




```
ArrMax_W(ArrIn:= , ArrN=> );
```

2、相关变量

变量名称	名称	数据类型	变量属性	描述
ArrIn	输入数组	数组	输入变量	需要查找的数组
ArrN	最大值下标	WORD	输出变量	数组最大值所在下标

3、具体使用

```
PROGRAM P2
VAR
    wData: ARRAY[0..100] OF WORD;
    wMaxDataNum:WORD;
    wMaxData:WORD;

    wMiniDataNum:WORD;
    wMiniData:WORD;
END_VAR
```

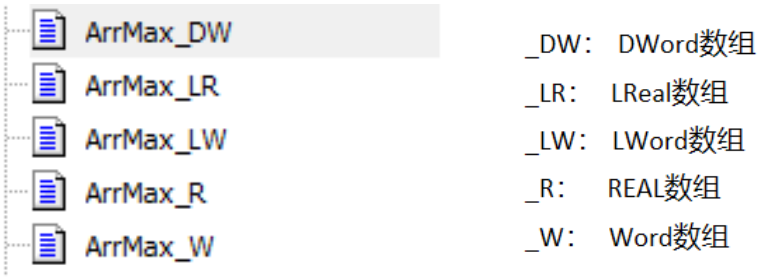
```
wMaxData:=ArrMax_W(ArrIn:= wData, ArrN=> wMaxDataNum);
wMiniData:=ArrMin_W(ArrIn:= wData, ArrN=> wMiniDataNum);
```

Device.Application.P2

表达式	类型	值	准备值	地址	注释
wData	ARRAY [0..100] OF WO...				
wData[0]	WORD	100			
wData[1]	WORD	200			
wData[2]	WORD	10			
wData[3]	WORD	0			
wData[4]	WORD	0			
wData[5]	WORD	0			
wData[6]	WORD	0			
wData[7]	WORD	0			
wData[8]	WORD	0			
wData[9]	WORD	0			

```
wMaxData[200]:=ArrMax_W(ArrIn:= wData, ArrN=> wMaxDataNum[1]);
wMiniData[0]:=ArrMin_W(ArrIn:= wData, ArrN=> wMiniDataNum[3]);RETURN
```

4、不同的数据类型使用不同的函数。如下注解



04_数列运算

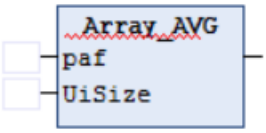
Array_AVG 数列的算术平均值

函数功能：数列的算术平均值。已知一组数列，计算平均值
数组的长度不能小于 uiSize，且不能小于 1；数组最大范围 0~32000

公式：$y=\frac{\sum_{i=0}^{n-1}(x_i)}{n};\quad (n>0)$

1、指令格式

图形表现：



ST 表现：

```
Array_AVG(paf:= , UiSize:= )
```

2、相关变量

变量名称	名称	数据类型	变量属性	描述
paf	输入的数列	POINTER TO ARRAY[*] OF LREAL	输入变量	指向需要计算的数列，即 Xi
UiSize	个数	UINT	输入变量	需要计算数列中多少个数的均值，即 n
ARRAY_AVG	返回值	LREAL	函数返回值	计算的均值

3、具体使用

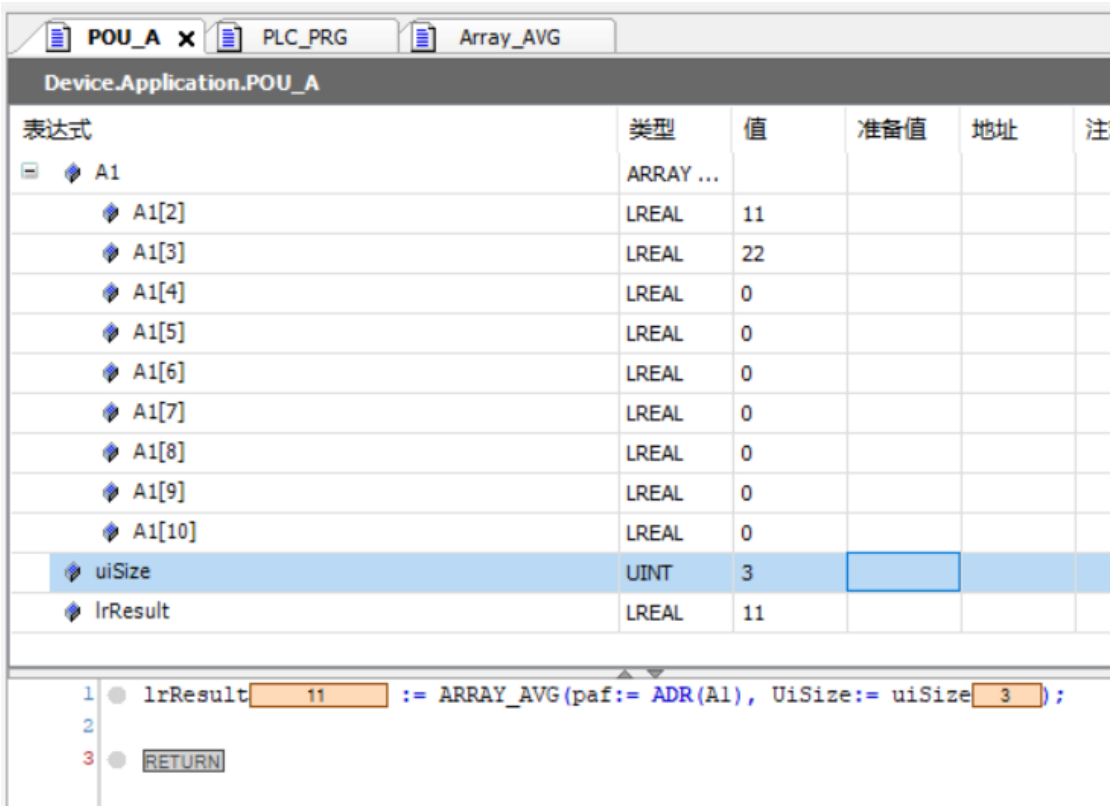
（1）声明和函数调用

POU_A | PLC_PRG | Array_AVG

```
1  PROGRAM POU_A
2  VAR
3      A1          :ARRAY [2..10] OF LREAL;
4      uiSize      :UINT :=2;
5      lrResult    :LREAL;
6  END_VAR
7
```

```
1  lrResult := ARRAY_AVG(paf:= ADR(A1), UiSize:= uiSize);
2
```

（2）写值与监控

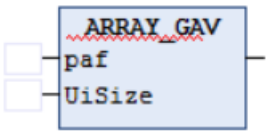


ARRAY_GAV 数列的几何平均值

函数功能：数列的几何平均值。已知一组数列，计算几何平均值
数组的长度不能小于 uiSize，且不能小于 1；数组最大范围 0~32000

公式： $y=\sqrt[n]{x_1*x_2*.....x_{n-1}}$;$(n>0,\ x_i>0)$

1、指令格式
图形表现：　　ST 表现：




```
Array_GAV(paf:= , UiSize:= )
```

2、相关变量

变量名称	名称	数据类型	变量属性	描述
paf	输入的数列	POINTER TO ARRAY[*] OF LREAL	输入变量	指向需要计算的数列，即 Xi
UiSize	个数	UINT	输入变量	需要计算数列中多少个数的均值，即 n
ARRAY_GAV	返回值	LREAL	函数返回值	计算的几何均值

3、具体使用

（1）声明和函数调用

POU_A | PLC_PRG

```
PROGRAM POU_A
VAR
    A1          :ARRAY [2..10] OF LREAL;
    uiSize      :UINT :=2;
    lrResult    :LREAL;
END_VAR
```

```
lrResult := ARRAY_GAV(paf:= ADR(A1), UiSize:= uiSize);
```

（2）写值与监控

Device.Application.POU_A

表达式	类型	值	准备值	地址	注释
A1	ARRAY ...				
A1[2]	LREAL	100			
A1[3]	LREAL	0			
A1[4]	LREAL	0			
A1[5]	LREAL	0			
A1[6]	LREAL	0			
A1[7]	LREAL	0			
A1[8]	LREAL	0			
A1[9]	LREAL	0			
A1[10]	LREAL	0			
uiSize	UINT	2			
lrResult	LREAL	10			

```
lrResult 10 := ARRAY_GAV(paf:= ADR(A1), UiSize:= uiSize 2 );

RETURN
```

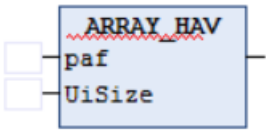
ARRAY_HAV 数列的调和/倒数平均值

函数功能：数列的调和平均值。已知一组数列，计算调和平均值（倒数平均值）
数组的长度不能小于 uiSize，且不能小于 1；数组最大范围 0~32000

公式：$y=\frac{n}{\sum_{i=0}^{n-1}\left(\frac{1}{x_i}\right)}$；$(n>0,\ x_i\neq 0)$

1、指令格式

图形表现：



ST 表现：

```
Array_HAV(paf:= , UiSize:= )
```

2、相关变量

变量名称	名称	数据类型	变量属性	描述
paf	输入的数列	POINTER TO ARRAY[*] OF LREAL	输入变量	指向需要计算的数列，即 Xi
UiSize	个数	UINT	输入变量	需要计算数列中多少个数的均值，即 n
ARRAY_HAV	返回值	LREAL	函数返回值	计算的调和/倒数均值

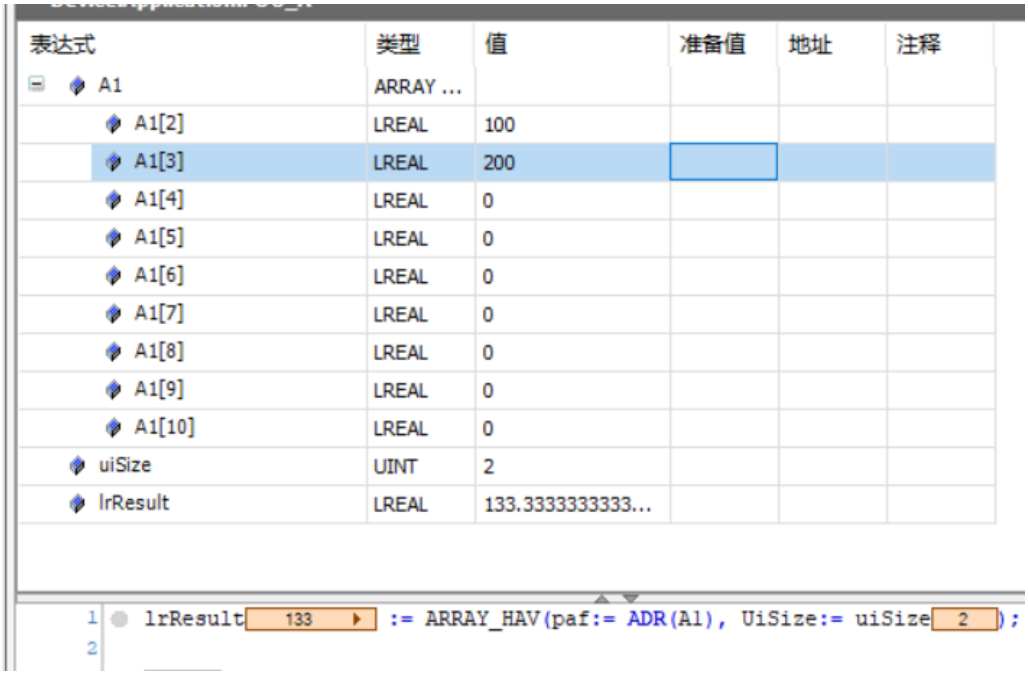
3、具体使用

（1）声明和函数调用

```
PROGRAM POU_A
VAR
    A1          :ARRAY [2..10] OF LREAL;
    uiSize      :UINT :=2;
    lrResult    :LREAL;
END_VAR
```

```
lrResult := ARRAY_HAV(paf:= ADR(A1), UiSize:= uiSize);
```

（2）写值与监控



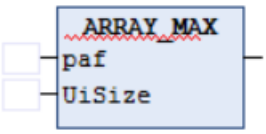
表达式	类型	值	准备值	地址	注释
A1	ARRAY ...				
A1[2]	LREAL	100			
A1[3]	LREAL	200			
A1[4]	LREAL	0			
A1[5]	LREAL	0			
A1[6]	LREAL	0			
A1[7]	LREAL	0			
A1[8]	LREAL	0			
A1[9]	LREAL	0			
A1[10]	LREAL	0			
uiSize	UINT	2			
lrResult	LREAL	133.3333333333...			

```
1  lrResult := ARRAY_HAV(paf:= ADR(A1), UiSize:= uiSize );
2
```

ARRAY_MAX 数列的最大值

函数功能：数列的最大值。已知一组数列的值，找出最大值
数组的长度不能小于 uiSize，且不能小于 1；数组最大范围 0~32000

1、指令格式
图形表现：　　ST 表现：




```
Array_MAX(paf:= , UiSize:= )
```

2、相关变量

变量名称	名称	数据类型	变量属性	描述
paf	输入的数列	POINTER TO ARRAY[*] OF LREAL	输入变量	指向需要计算的数列，即 Xi
UiSize	个数	UINT	输入变量	需要计算数列的个数，即 n
ARRAY_MAX	返回值	LREAL	函数返回值	最大值

3、具体使用
（1）声明和函数调用

```
POU_A | PLC_PRG
1  PROGRAM POU_A
2  VAR
3      A1          :ARRAY [2..10] OF LREAL;
4      uiSize      :UINT :=2;
5      lrResult    :LREAL;
6  END_VAR
7

1  lrResult := ARRAY_MAX(paf:= ADR(A1), UiSize:= uiSize);
```

（2）写值与监控

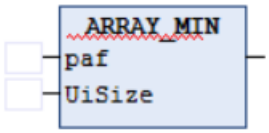
表达式	类型	值	准备值	地址	注释
A1	ARRAY ...				
A1[2]	LREAL	0			
A1[3]	LREAL	100			
A1[4]	LREAL	0			
A1[5]	LREAL	200			
A1[6]	LREAL	0			
A1[7]	LREAL	0			
A1[8]	LREAL	0			
A1[9]	LREAL	0			
A1[10]	LREAL	0			
uiSize	UINT	2			
lrResult	LREAL	100			

```
1  lrResult 100 := ARRAY_MAX(paf:= ADR(A1), UiSize:= uiSize 2 );
2
```

ARRAY_MIN 数列的最小值

函数功能：数列的最小值。已知一组数列的值，找出最小值
数组的长度不能小于 uiSize，且不能小于 1；数组最大范围 0~32000

1、指令格式
图形表现：　　ST 表现：




```
Array_MIN(paf:= , UiSize:= )
```

2、相关变量

变量名称	名称	数据类型	变量属性	描述
paf	输入的数列	POINTER TO ARRAY[*] OF	输入变量	指向需要计算的数列，即 Xi

		LREAL		
UiSize	个数	UINT	输入变量	需要计算数列的个数，即 n
ARRAY_Min	返回值	LREAL	函数返回值	最小值

3、具体使用

（1）声明和函数调用

```
PROGRAM POU_A
VAR
    A1          :ARRAY [2..10] OF LREAL;
    uiSize      :UINT :=2;
    lrResult    :LREAL;
END_VAR
```

```
lrResult := ARRAY_Min(paf:= ADR(A1), UiSize:= uiSize);
```

（2）写值与监控

表达式	类型	值	准备值	地址	注释
A1	ARRAY ...				
A1[2]	LREAL	0			
A1[3]	LREAL	0			
A1[4]	LREAL	0			
A1[5]	LREAL	-10			
A1[6]	LREAL	0			
A1[7]	LREAL	0			
A1[8]	LREAL	0			
A1[9]	LREAL	0			
A1[10]	LREAL	0			
uiSize	UINT	10			
lrResult	LREAL	-10			

```
lrResult -10 := ARRAY_Min(paf:= ADR(A1), UiSize:= uiSize 10 );
```

ARRAY_MED 数列的中值

函数功能：数列的中值。已知一组数列的值，找出中值

数组的长度不能小于 uiSize，且不能小于 1；数组最大范围 0~32000

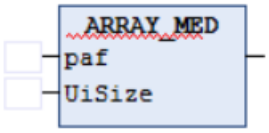
公式：

（1） 若数列长度是偶数，那么 $i=\frac{n}{2}$； $y=\frac{x_i+x_{i-1}}{2}$ $(n>0,)$

（2） 若数列长度是奇数，那么 $i=\frac{n-1}{2}$； $y=x_i$ (n>0,)

1、指令格式

图形表现：



ST 表现：

```
Array_MED(paf:= , UiSize:= )
```

2、相关变量

变量名称	名称	数据类型	变量属性	描述
paf	输入的数列	POINTER TO ARRAY[*] OF LREAL	输入变量	指向需要计算的数列，即 Xi
UiSize	个数	UINT	输入变量	需要计算数列的个数，即 n
ARRAY_MED	返回值	LREAL	函数返回值	中值

3、具体使用

（1）声明和函数调用

```
PROGRAM POU_A
VAR
    A1          :ARRAY [2..10] OF LREAL;
    uiSize      :UINT :=2;
    lrResult    :LREAL;
END_VAR
```

```
lrResult := ARRAY_Min(paf:= ADR(A1), UiSize:= uiSize);
```

（2）写值与监控

A1	ARRAY ...		
A1[2]	LREAL	1	
A1[3]	LREAL	2	
A1[4]	LREAL	3	
A1[5]	LREAL	4	
A1[6]	LREAL	5	
A1[7]	LREAL	0	
A1[8]	LREAL	0	
A1[9]	LREAL	0	
A1[10]	LREAL	0	
uiSize	UINT	5	奇数
lrResult	LREAL	3	

```
lrResult 3 := ARRAY_MED(paf:= ADR(A1), UiSize:= uiSize 5 );
```

A1	ARRAY ...		
A1[2]	LREAL	1	
A1[3]	LREAL	2	
A1[4]	LREAL	4	
A1[5]	LREAL	5	i=3; y=(4+2)/2
A1[6]	LREAL	8	
A1[7]	LREAL	0	
A1[8]	LREAL	0	
A1[9]	LREAL	0	
A1[10]	LREAL	0	
uiSize	UINT	6	偶数
lrResult	LREAL	3	

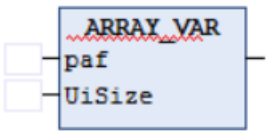
```
1   lrResult 3 := ARRAY_MED(paf:= ADR(A1), UiSize:= uiSize 6 );
2
```

ARRAY_VAR 数列的方差

函数功能：数列的方差。已知一组数列的值，找出方差
数组的长度不能小于 uiSize，且不能小于 1；数组最大范围 0~32000

公式：$$y = \frac{\sum_{i=0}^{n-1}(\bar{x} - x_i)^2}{n} \quad (n>0, \bar{x}\text{是算术平均值})$$

1、指令格式
图形表现：



ST 表现：

```
Array_VAR(paf:= , UiSize:= )
```

2、相关变量

变量名称	名称	数据类型	变量属性	描述
paf	输入的数列	POINTER TO ARRAY[*] OF LREAL	输入变量	指向需要计算的数列，即 Xi
UiSize	个数	UINT	输入变量	需要计算数列的个数，即 n
ARRAY_VAR	返回值	LREAL	函数返回值	方差值

3、具体使用
（1）声明和函数调用

```
PROGRAM POU_A
VAR
    A1          :ARRAY [2..10] OF LREAL;
    uiSize      :UINT :=3;
    lrResult    :LREAL;
END_VAR
```

```
lrResult := ARRAY_VAR(paf:= ADR(A1), UiSize:= uiSize);
```

（2）写值与监控

A1	ARRAY ...				
A1[2]	LREAL	10			
A1[3]	LREAL	15			
A1[4]	LREAL	5			
A1[5]	LREAL	0			
A1[6]	LREAL	0			
A1[7]	LREAL	0			
A1[8]	LREAL	0			
A1[9]	LREAL	0			
A1[10]	LREAL	0			
uiSize	UINT	3			
lrResult	LREAL	16.66666666666...			

```
lrResult 16.7 := ARRAY_VAR(paf:= ADR(A1), UiSize:= uiSize 3 );
```

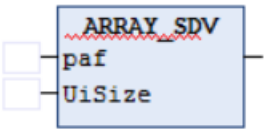
ARRAY_SDV 数列的标准差

函数功能：数列的方差。已知一组数列的值，找出标准差。标准差即方差的算术平方根。
数组的长度不能小于 uiSize，且不能小于 1；数组最大范围 0~32000

公式：$y=\sqrt{ARRAY_VAR}$

1、指令格式

图形表现：



ST 表现：

```
Array_SDV(paf:= , UiSize:= )
```

2、相关变量

变量名称	名称	数据类型	变量属性	描述
paf	输入的数列	POINTER TO ARRAY[*] OF LREAL	输入变量	指向需要计算的数列，即 Xi
UiSize	个数	UINT	输入变量	需要计算数列的个数，即 n
ARRAY_SDV	返回值	LREAL	函数返回值	标准差

3、具体使用

（1）声明和函数调用

```
PROGRAM POU_A
VAR
    A1          :ARRAY [2..10] OF LREAL;
    uiSize      :UINT :=3;
    lrResult    :LREAL;
END_VAR
```

```
lrResult := ARRAY_SDV(paf:= ADR(A1), UiSize:= uiSize);
```

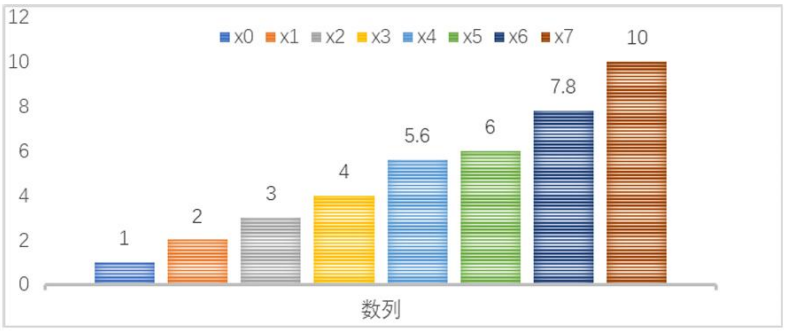
（2）写值与监控

表达式	类型	值	准备值	地址	注释
A1	ARRAY ...				
A1[2]	LREAL	10			
A1[3]	LREAL	15			
A1[4]	LREAL	5			
A1[5]	LREAL	0			
A1[6]	LREAL	0			
A1[7]	LREAL	0			
A1[8]	LREAL	0			
A1[9]	LREAL	0			
A1[10]	LREAL	0			
uiSize	UINT	3			
lrResult	LREAL	4.08248290463863			

```
lrResult 4.08 := ARRAY_SDV(paf:= ADR(A1), UiSize:= uiSize 3 );
```

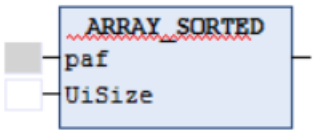
ARRAY_SORTED 数列的升序判断

函数功能：数列的方差。已知一组数列的值，判断数列中元素是按照升序排序的
数组的长度不能小于 uiSize，且不能小于 1；数组最大范围 0~32000
若数列如下按照升序排序，或者值都相同，则判断为 TRUE；反之为 FALSE



1、指令格式

图形表现：



ST 表现：

```
ARRAY_SORTED(paf:= , UiSize:= )
```

2、相关变量

变量名称	名称	数据类型	变量属性	描述
paf	输入的数列	POINTER TO ARRAY[*] OF LREAL	输入变量	指向需要计算的数列，即 Xi
UiSize	个数	UINT	输入变量	需要计算数列的个数，即 n
ARRAY_SORTED	返回值	BOOL	函数返回值	按照升序排序或者值都相同则为 TRUE

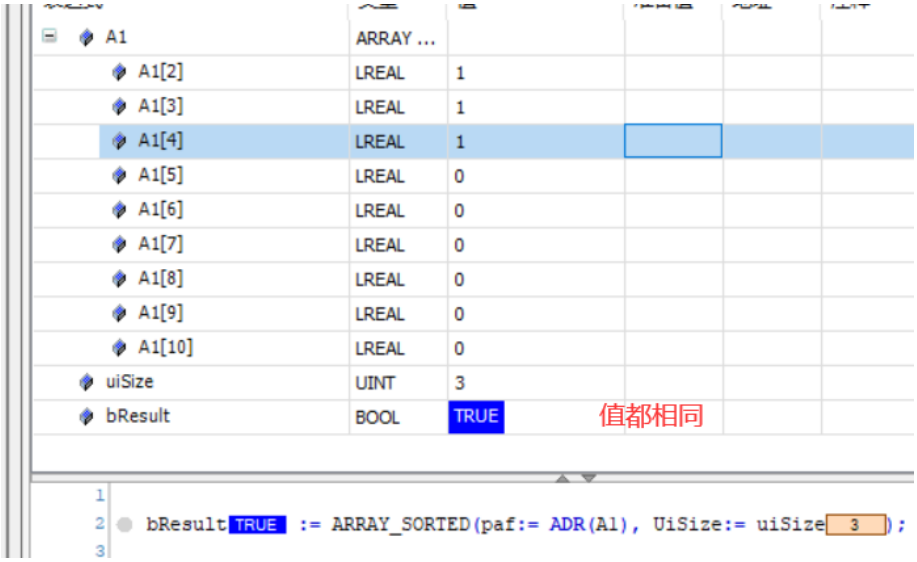
3、具体使用

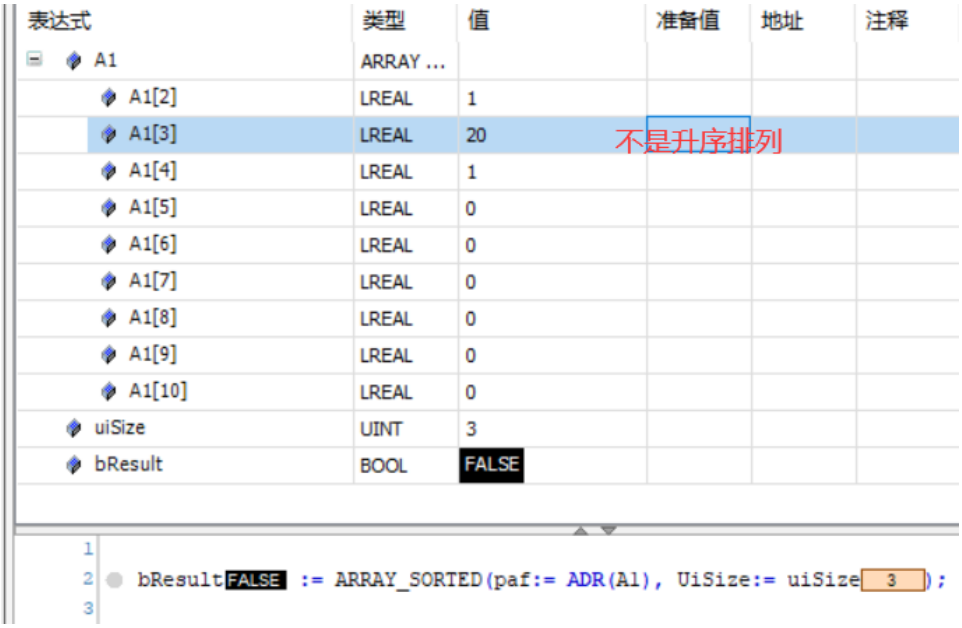
（1）声明和函数调用

```
PROGRAM POU_A
VAR
    A1          :ARRAY [2..10] OF LREAL;
    uiSize      :UINT :=3;
    bResult     :BOOL;
END_VAR
```

```

bResult := ARRAY_SORTED(paf:= ADR(A1), UiSize:= uiSize);
```

（2）写值与监控

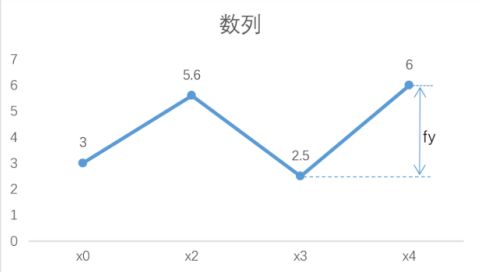

A1 ARRAY ...
A1[2] LREAL 1
A1[3] LREAL 1
A1[4] LREAL 1
A1[5] LREAL 0
A1[6] LREAL 0
A1[7] LREAL 0
A1[8] LREAL 0
A1[9] LREAL 0
A1[10] LREAL 0
uiSize UINT 3
bResult BOOL TRUE
值都相同
bResult TRUE := ARRAY_SORTED(paf:= ADR(A1), UiSize:= uiSize 3);


表达式 类型 值 准备值 地址 注释
A1 ARRAY ...
A1[2] LREAL 1
A1[3] LREAL 20
A1[4] LREAL 1
A1[5] LREAL 0
A1[6] LREAL 0
A1[7] LREAL 0
A1[8] LREAL 0
A1[9] LREAL 0
A1[10] LREAL 0
uiSize UINT 3
bResult BOOL FALSE
不是升序排列
bResult FALSE := ARRAY_SORTED(paf:= ADR(A1), UiSize:= uiSize 3);


表达式 类型 值 准备值 地址 注释
A1 ARRAY ...
A1[2] LREAL 1
A1[3] LREAL 20
A1[4] LREAL 30
A1[5] LREAL 0
A1[6] LREAL 0
A1[7] LREAL 0
A1[8] LREAL 0
A1[9] LREAL 0
A1[10] LREAL 0
uiSize UINT 3
bResult BOOL TRUE
是升序排列
bResult TRUE := ARRAY_SORTED(paf:= ADR(A1), UiSize:= uiSize 3);

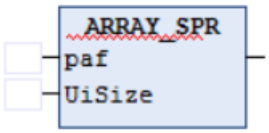
ARRAY_SPR 数列的最大偏差

函数功能：数列的最大偏差值。已知一组数列的值，求数列中的最大值和最小值之间的偏差
数组的长度不能小于 uiSize，且不能小于 1；数组最大范围 0~32000



1、指令格式

图形表现：



ST 表现：

```
ARRAY_SPR(paf:= , UiSize:= )
```

2、相关变量

变量名称	名称	数据类型	变量属性	描述
paf	输入的数列	POINTER TO ARRAY[*] OF LREAL	输入变量	指向需要计算的数列，即 Xi
UiSize	个数	UINT	输入变量	需要计算数列的个数，即 n
ARRAY_SPR	返回值	LREAL	函数返回值	最大偏差

3、具体使用

（1）声明和函数调用

```
PROGRAM POU_A
VAR
    Al          :ARRAY [2..10] OF LREAL;
    uiSize      :UINT :=3;
    lrResult    :LREAL;

END_VAR
```

```
lrResult := ARRAY_SPR(paf:= ADR(Al), UiSize:= uiSize);
```

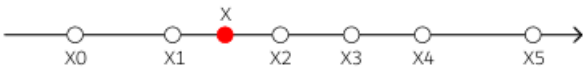
（2）写值与监控

A1	ARRAY ...				
A1[2]	LREAL	11			
A1[3]	LREAL	66.5			
A1[4]	LREAL	33			
A1[5]	LREAL	0			
A1[6]	LREAL	0			
A1[7]	LREAL	0			
A1[8]	LREAL	0			
A1[9]	LREAL	0			
A1[10]	LREAL	0			
uiSize	UINT	3			
lrResult	LREAL	55.5			

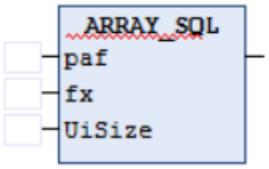
```
1
2
3   lrResult 55.5 := ARRAY_SPR(paf:= ADR(A1), UiSize:= uiSize 3 );
```

ARRAY_SQL 数列中查询数值

函数功能：已知一组升序排列的数列，查询 X 在数列中的位置（数组的编号）
数组的长度不能小于 uiSize，且不能小于 1；数组最大范围 0~32000



1、指令格式
图形表现：　　ST 表现：




```
ARRAY_SQL(paf:= , fx:= , UiSize:= )
```

2、相关变量

变量名称	名称	数据类型	变量属性	描述
paf	输入的数列	POINTER TO ARRAY[*] OF LREAL	输入变量	指向需要计算的数列，即 Xi
UiSize	个数	UINT	输入变量	需要计算数列的个数，即 n
Fx	要查的数值	LREAL	输入变量	查询的数值
ARRAY_SQL	返回值	LREAL	函数返回值	查询结果

3、具体使用

（1）声明和函数调用

```
PROGRAM POU_A
VAR
    A1          :ARRAY [0..5] OF LREAL;
    uiSize      :UINT :=3;
    lrResult    :LREAL;

END_VAR
```

```
lrResult := ARRAY_SQL(paf:= ADR(A1), fx:= 10, UiSize:= uiSize);
```

（2）写值与监控

表达式	类型	值	准备值	地址	注释
A1	ARRAY ...				
A1[0]	LREAL	2			
A1[1]	LREAL	10			
A1[2]	LREAL	11			
A1[3]	LREAL	15			
A1[4]	LREAL	0			
A1[5]	LREAL	0			
uiSize	UINT	3			
lrResult	LREAL	1			

```
1
2
3   lrResult 1 := ARRAY_SQL(paf:= ADR(A1), fx:= 10, UiSize:= uiSize 3 );
4
```

（3） 注意事项

若 fx 小于数组的最小值，那么输出 ARRAY_SQL 为 0。否则，输出 ARRAY_SQL 为左侧数组下标+1。

ARRAY_SUM 数列的求和

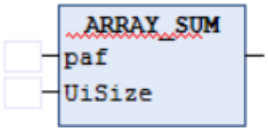
函数功能：已知一组数列的值，求数值的总和

数组的长度不能小于 uiSize，且不能小于 1；数组最大范围 0~32000

公式：$$y = \sum_{i=0}^{n-1}(x_i)\ (n>0)$$

1、指令格式

图形表现： ST 表现：




```
ARRAY_SUM(paf:= , UiSize:= )
```

2、相关变量

变量名称	名称	数据类型	变量属性	描述
paf	输入的数列	POINTER TO ARRAY[*] OF LREAL	输入变量	指向需要计算的数列，即 Xi
UiSize	个数	UINT	输入变量	需要计算数列的个数，即 n
ARRAY_SUM	返回值	LREAL	函数返回值	数值的总和

3、具体使用

（1）声明和函数调用

```
PROGRAM POU_A
VAR
    A1          :ARRAY [0..5] OF LREAL;
    uiSize      :UINT :=3;
    lrResult    :LREAL;

END_VAR
```

```

lrResult := ARRAY_SUM(paf:= ADR(A1),  UiSize:= uiSize);

```

（2）写值与监控

A1	ARRAY ...				
A1[0]	LREAL	4.5			
A1[1]	LREAL	11			
A1[2]	LREAL	77			
A1[3]	LREAL	0			
A1[4]	LREAL	0			
A1[5]	LREAL	0			
uiSize	UINT	3			
lrResult	LREAL	92.5			

```

lrResult 92.5 := ARRAY_SUM(paf:= ADR(A1),  UiSize:= uiSize 3 );

```

ARRAY_ABS 数列的绝对值

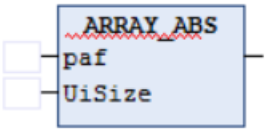
函数功能：已知一组数列的值，求每个要素的绝对值
数组的长度不能小于 uiSize，且不能小于 1；数组最大范围 0~32000

$$X^{!} = \left|X^{!}\right| \quad (\mu>0)$$

公式：

1、指令格式

图形表现：



ST 表现：

```
ARRAY_ABS(paf:= , UiSize:= )
```

2、相关变量

变量名称	名称	数据类型	变量属性	描述
paf	输入的数列	POINTER TO ARRAY[*] OF LREAL	输入变量	指向需要计算的数列，即 Xi
UiSize	个数	UINT	输入变量	需要计算数列的个数，即 n
ARRAY_ABS	返回值	BOOL	函数返回值	True:执行完成

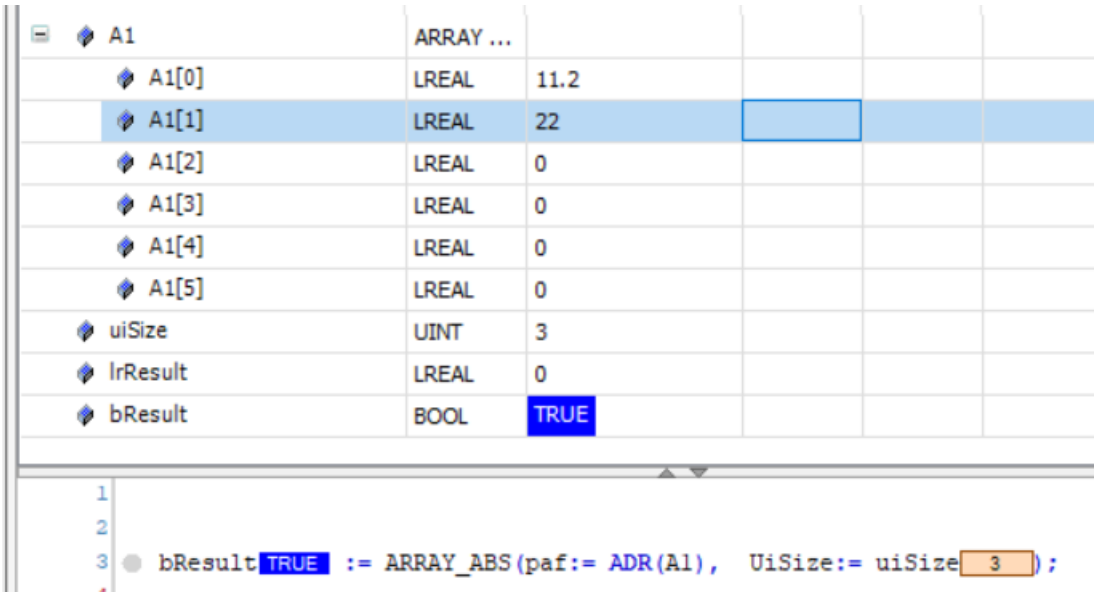
3、具体使用

（1）声明和函数调用

```
PROGRAM POU_A
VAR
    Al          :ARRAY [0..5] OF LREAL;
    uiSize      :UINT :=3;
    lrResult    :LREAL;

    bResult     :BOOL;
END_VAR
```

```


bResult := ARRAY_ABS(paf:= ADR(Al),  UiSize:= uiSize);

```

（2）写值与监控



ARRAY_ADD 数列的偏移

待发

ARRAY_MUL 数列的缩放

待发

ARRAY_RAN 数列的范围

待发

ARRAY_RPT 剔除重复值

待发

ARRAY_SQRT 数列的冒泡排序

待发

05_直线圆弧相关函数

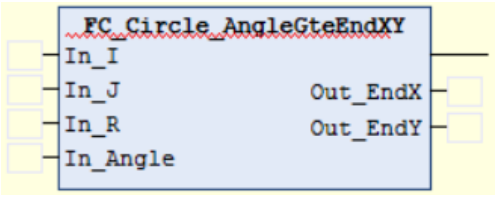
051_圆弧相关

FC_Circle_AngleGetEndXY 求圆弧结束点坐标

函数功能：已知圆心，角度，半径；计算圆弧结束点坐标。 水平第一象限为 0 度

1、指令格式

图形表现：



ST 表现：

```
FC_Circle_AngleGteEndXY(In_I:= ,
                        In_J:= ,
                        In_R:= ,
                        In_Angle:= ,
                        Out_EndX=> ,
                        Out_EndY=> );
```

2、相关变量

变量名称	名称	数据类型	变量属性	描述
In_I	圆心的 X 坐标	LREAL	输入变量	圆心 X 坐标
In_J	圆心的 Y 坐标	LREAL	输入变量	圆心 Y 坐标
In_R	半径	LREAL	输入变量	半径
In_Angle	夹角	LREAL	输入变量	夹角
Out_EndX	输出圆弧终点 X 坐标	LREAL	输出变量	计算后的圆弧终点 X 坐标
Out_EndY	输出圆弧终点 Y 坐标	LREAL	输出变量	计算后的圆弧终点 Y 坐标

3、具体使用

（1）声明和函数调用

```
PROGRAM P2
VAR
    lrR :LREAL;
    lrX :LREAL;
    lrY :LREAL;
    lrAngle :LREAL;

    lr_EndX :LREAL;
    lr_EndY :LREAL;
END_VAR
```

```
FC_Circle_AngleGteEndXY(In_I:=lrX ,
                        In_J:= lrY,
                        In_R:=lrR ,
                        In_Angle:=lrAngle ,
                        Out_EndX=>lr_EndX ,
                        Out_EndY=>lr_EndY );
```

（2）写值与监控

表达式	类型	值	准备值
lrR	LREAL	30	
lrX	LREAL	10	
lrY	LREAL	10	
lrAngle	LREAL	180	
lr_EndX	LREAL	-20	
lr_EndY	LREAL	10.000000000000004	

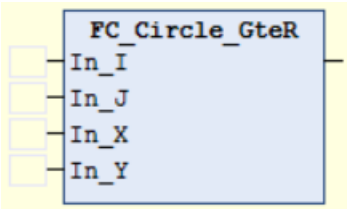
```
FC_Circle_AngleGteEndXY(In_I:=lrX 10 ,
                        In_J:= lrY 10 ,
                        In_R:=lrR 30 ,
                        In_Angle:=lrAngle 180 ,
                        Out_EndX=>lr_EndX -20 ,
                        Out_EndY=>lr_EndY 10 );RETURN
```

FC_Circle_GetR 求圆半径

函数功能：已知圆心，圆弧的坐标；计算圆的半径

1、指令格式

图形表现：



ST 表现：

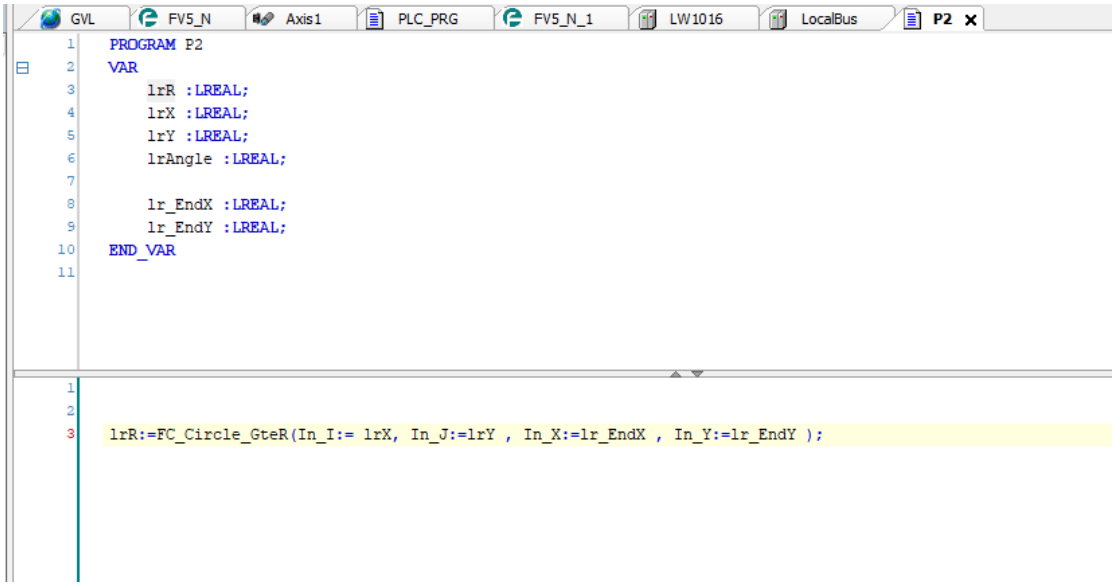
```
FC_Circle_GteR(In_I:= ,
               In_J:= ,
               In_X:= ,
               In_Y:= );
```

2、相关变量

变量名称	名称	数据类型	变量属性	描述
In_I	圆心的 X 坐标	LREAL	输入变量	圆心的 X 坐标
In_J	圆心的 Y 坐标	LREAL	输入变量	圆心的 Y 坐标
In_X	圆弧的 X 坐标	LREAL	输入变量	圆弧的 X 坐标
In_Y	圆弧的 Y 坐标	LREAL	输入变量	圆弧的 Y 坐标
FC_Circle_GteR	输出 R	LREAL	输出变量	计算得到的圆的半径

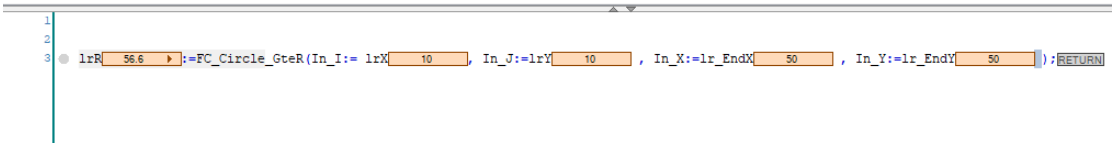
3、具体使用

（1）声明和函数调用



（2）写值与监控

表达式	类型	值	准备值	地址	注释
lrR	LREAL	56.568542494923804			
lrX	LREAL	10			
lrY	LREAL	10			
lrAngle	LREAL	0			
lr_EndX	LREAL	50			
lr_EndY	LREAL	50			

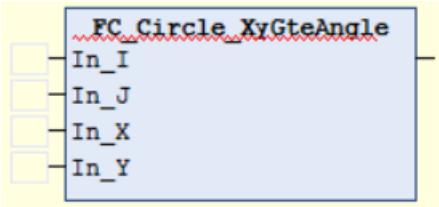


FC_Circle_XyGetAngle 求圆弧夹角

函数功能：已知，起始点、终点、求圆弧夹角。水平第一象限为 0 度

1、指令格式

图形表现：



ST 表现：

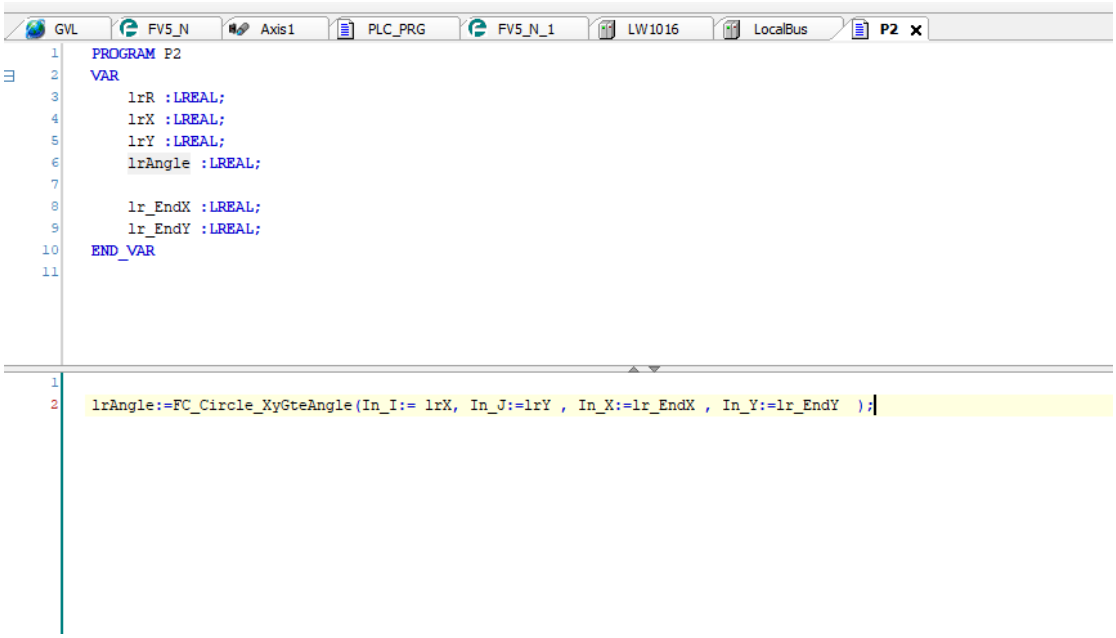
```
FC_Circle_XyGteAngle(In_I:= ,
                     In_J:= ,
                     In_X:= ,
                     In_Y:= );
```

2、相关变量

变量名称	名称	数据类型	变量属性	描述
In_I	圆心的 X 坐标	LREAL	输入变量	圆心的 X 坐标
In_J	圆心的 Y 坐标	LREAL	输入变量	圆心的 Y 坐标
In_X	圆弧的 X 坐标	LREAL	输入变量	圆弧的 X 坐标
In_Y	圆弧的 Y 坐标	LREAL	输入变量	圆弧的 Y 坐标
FC_Circle_XyGteAngle	输出夹角	LREAL	输出变量	计算得到的圆的夹角

3、具体使用

（1）声明和函数调用



（2）写值与监控

Device.Application.P2

表达式	类型	值	准备值	地址	注释
lrR	LREAL	0			
lrX	LREAL	10			
lrY	LREAL	10			
lrAngle	LREAL	45			
lr_EndX	LREAL	50			
lr_EndY	LREAL	50			

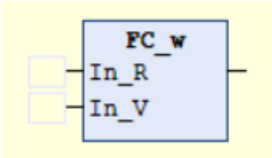
```
1
2   lrAngle 45 :=FC_Circle_XyGteAngle(In_I:= lrX 10 , In_J:=lrY 10 , In_X:=lr_EndX 50 , In_Y:=lr_EndY 50 );RETURN
```

FC_w 角速度计算

函数功能：已知线速度和半径，计算角速度

1、指令格式

图形表现：



ST 表现：

```
FC_Circle_XyGteAngle(In_I:= ,
                     In_J:= ,
                     In_X:= ,
                     In_Y:= );
```

2、相关变量

变量名称	名称	数据类型	变量属性	描述
In_R	半径	LREAL	输入变量	输入半径
In_V	线速度	LREAL	输入变量	输入线速度
FC_w	角速度	LREAL	输出变量	计算后的角速度

3、具体使用

（1）声明和函数调用

```
PROGRAM P2
VAR

    lrR:LREAL;
    lrV:LREAL ;
    lrW:LREAL;
END_VAR
```

```
lrW:= FC_w(In_R:= lrR, In_V:= lrV);
```

（2）写值与调用（半径 30，线速度 2u/s，角速度为 0.066667）

表达式	类型	值
lrR	LREAL	30
lrV	LREAL	2
lrW	LREAL	0.066666666666666666

```
1  lrW 0.0667 := FC_w(In_R:= lrR 30 , In_V:= lrV 2 );
2
```

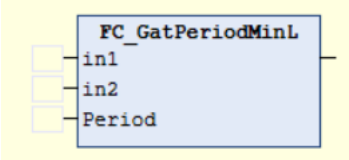
052_位移速度计算

FC_GetPeriodMinL 计算模值下的最短路径

函数功能：根据模值，计算最短路径，有正负区分。例如，插补功能块里面，判断模态轴定位方向使用

1、指令格式

图形表现



ST 表现

```
FC_GatPeriodMinL(in1:= , in2:= , Period:= );
```

2、相关变量

变量名称	名称	数据类型	变量属性	描述
In1	输入位置值 1	LREAL	输入变量	输入位置值 1
In2	输入位置值 2	LREAL	输入变量	输入位置值 2
Period	模值	LREAL	输入变量	模值
FC_GatPeriodMinL	最短距离	LREAL	输出变量	计算后的最短距离

3、具体使用

（1）声明和函数调用

```
PROGRAM P2
VAR
    lrData1:LREAL;
    lrData2:LREAL;
    lrPeriod:LREAL;

    lrPeriodMinL:LREAL;
END_VAR
```

```
lrPeriodMinL:=FC_GatPeriodMinL(in1:=lrData1 , in2:= lrData2, Period:=lrPeriod );
```

（2）给值与计算

表达式	类型	值	准备值	地址	注释
lrData1	LREAL	10			
lrData2	LREAL	340			
lrPeriod	LREAL	360			
lrPeriodMinL	LREAL	-30			

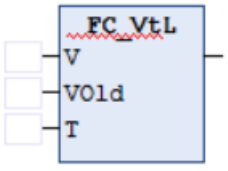
```
lrPeriodMinL -30 :=FC_GatPeriodMinL(in1:=lrData1 10 , in2:= lrData2 340 , Period:=lrPeriod 360 );RETURN
```

FC_VtL 计算一段时间的位移量

函数功能：根据加速度公式，计算一段时间内的位移量

1、指令格式

图形表现　　ST 表现




```
FC_VtL(V:= , VOld:= , T:= );
```

2、相关变量

变量名称	名称	数据类型	变量属性	描述
V	初始速度	LREAL	输入变量	V0 初始速度值

Vold	最终速度	LREAL	输入变量	Vt 最终速度值
T	时间	LREAL	输入变量	时间
FC_VtL	X 位移量	LREAL	输出变量	计算得到的距离

3、具体使用

（1）声明和函数调用

```
PROGRAM P2
VAR
    lrData1:LREAL;
    lrData2:LREAL;
    lrDistance:LREAL;

    lrAcc:LREAL;
END_VAR
```

```
lrDistance:=FC_VtL(V:=lrData1 , VOld:= lrData2, T:= 0.1);
```

（2）写值与监控

Device.Application.P2

表达式	类型	值	准备值	地址	注释
lrData1	LREAL	100			
lrData2	LREAL	0			
lrDistance	LREAL	5			
lrAcc	LREAL	0			

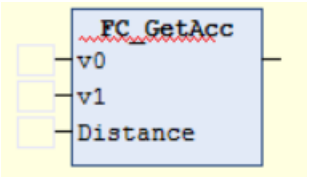
```
lrDistance 5 :=FC_VtL(V:=lrData1 100 , VOld:= lrData2 0 , T:= 0.1);RETURN
```

FC_GetAcc 求加速度

函数功能：根据初始速度、最终需要的运行速度，距离，求得加速度

1、指令格式

图形表现　　ST 表现



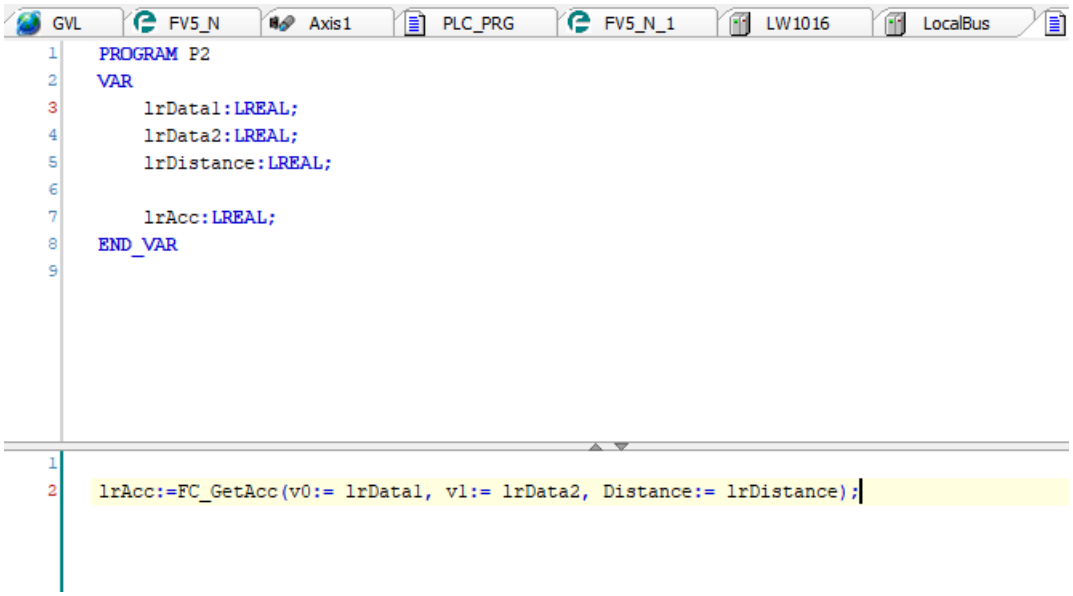

```
FC_GetAcc(v0:= , v1:= , Distance:= );
```

2、相关变量

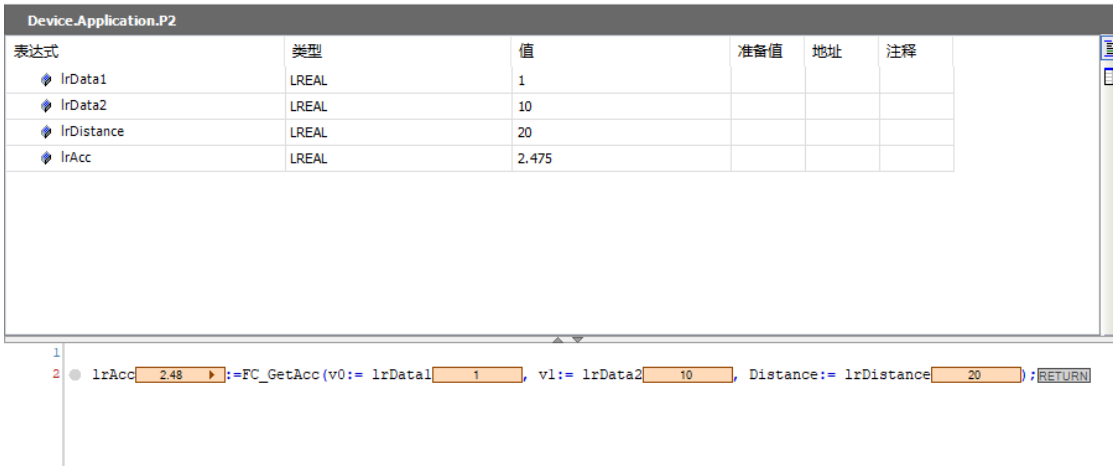
变量名称	名称	数据类型	变量属性	描述
V0	输入速度值 1	LREAL	输入变量	输入速度值 0
V1	输入速度值 2	LREAL	输入变量	输入速度值 1
Distance	距离	LREAL	输入变量	定位距离
FC_GetAcc	计算加速度	LREAL	输出变量	计算后的加速度

3、具体使用

（1）声明和函数调用



（2）写值与监控



备注:

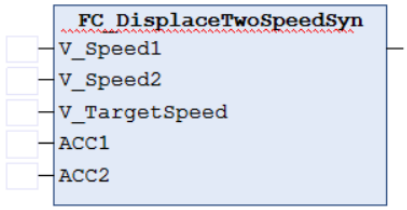
匀加速直线位移公式：X=(V1 * V1 – V0 * V0) / (2 * a)

FC_ DisplaceTwoSpeedSyn 计算两速产生的位移差

函数功能：例如 A 轴用 V 速度 1 和 ACC1（加速度 1）加速到目标速度，B 用 V 速度 2 和 ACC2（加速度 2）运动到目标速度，两者之间产生的位移差。可用于插补位移计算。

1、指令格式

图形表现



ST 表现

```
FC_DisplaceTwoSpeedSyn(
    V_Speed1:= ,
    V_Speed2:= ,
    V_TargetSpeed:= ,
    ACC1:= ,
    ACC2:= )
```

2、相关变量

变量名称	名称	数据类型	变量属性	描述
V_Speed1	速度 1	REAL	输入变量	V1
V_Speed2	速度 2	REAL	输入变量	V2
V_TargetSpeed	时间	REAL	输入变量	目标速度
ACC1	加速度 1	REAL	输入变量	加速度 1（和 V1 一起计算）
ACC2	加速度 2	REAL	输入变量	加速度 2（和 V2 一起计算）
FC_DisplaceTwoSpeedSyn	位移差	REAL	输出变量	计算两组 V 和 a 运动到目标速度时候产生的位移差

3、具体使用

（1）声明和函数调用

```
PROGRAM P2
VAR
    rData1:REAL;
    rData2:REAL;

    rTargetV:REAL;

    rAcc1:REAL;
    rAcc2:REAL;

    rL2_L1:REAL;
END_VAR
```

```
rL2_L1:=
FC_两速度同步V1位移差(
    V_速度1:= rData1,
    V_速度2:= rData2,
    V_目标速度:= rTargetV,
    ACC1:=rAcc1 ,
    ACC2:= rAcc2);
```

（2）写值与监控

Device.Application.P2

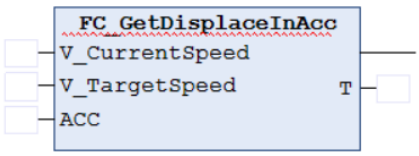
表达式	类型	值	准备值	地址	注释
rData1	REAL	10			
rData2	REAL	100			
rTargetV	REAL	10000			
rAcc1	REAL	10			
rAcc2	REAL	10			
rL2_L1	REAL	-89505			

```
rL2_L1 -8.95E+04 :=
FC_两速度同步V1位移差(
    V_速度1:= rData1 10 ,
    V_速度2:= rData2 100 ,
    V_目标速度:= rTargetV 1E+04 ,
    ACC1:=rAcc1 10 ,
    ACC2:= rAcc2 10 );
```

FC_GetDisplaceInAcc 计算匀加速阶段的位移和所需时间

函数功能：根据当前速度和目标速度，以及加速度。求得匀加速段的位移，以及需要多少时间匀加速结束

1、指令格式
图形表现：



ST 表现：

```
FC_GetDisplaceInAcc(V_CurrentSpeed:= , V_TargetSpeed:= , ACC:= , T=> )
```

2、相关变量

变量名称	名称	数据类型	变量属性	描述
V_CurrentSpeed	当前速度值	REAL	输入变量	当前速度值 u/s
V_TargetSpeed	目标速度值	REAL	输入变量	需要加速到的目标速度值 u/s
ACC	加速度	REAL	输入变量	加速度 u/s^2
T	时间	REAL	输出变量	时间 s
FC_GetDisplaceInAcc	返回匀加速段的位移	REAL	输出变量	匀加速过程中的位移量 u

3、具体使用

（1）声明和函数调用

```
PROGRAM P2
VAR
    rData1  :REAL;
    rTargetV:REAL;
    rAcc1   :REAL;
    rS      :REAL;
    tTime   :REAL;
END_VAR
```

```
rS:=FC_求匀加速位移(V_当前速度:= rData1, V_目标速度:= rTargetV, ACC:= rAcc1, T=> tTime);
```

（2）写值与监控（从 10 用 100000 的加速度加速到 1000，计算得需要 0.0099s，匀加速段位移是 4.9995）

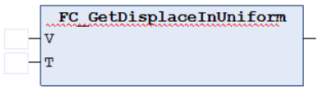


FC_GetDisplaceInUniform 计算匀速段的位移

函数功能：根据匀速段的速度值和时间，计算匀速段的位移量

1、指令格式

图形表现：



ST 表现：

```
FC_GetDisplaceInUniform(V:= , T:= );
```

2、相关变量

变量名称	名称	数据类型	变量属性	描述
V	匀速速度值	REAL	输入变量	匀速速度值
T	匀速运动的时间	REAL	输入变量	匀速运动的时间
FC_GetDisplaceInUniform	匀速运动的位移	REAL	输出变量	计算后的位移量

3、具体使用

（1）声明和函数调用

```
PROGRAM P2
VAR

    rV:REAL;
    rT:REAL;
    rMove:REAL;
END_VAR
```

```


rMove := FC_求匀速位移(V:= rV, T:= rT);

```

（2）写值与监控：匀速段的 S=V * T

rV	REAL	10.1
rT	REAL	0.5
rMove	REAL	5.05

```


rMove 5.05 := FC_求匀速位移(V:= rV 10.1, T:= rT 0.5);

```

06_函数与求解类

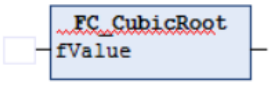
061 求立方根

FC_CubicRoot 求立方根

函数功能：求变量的立方根值

1、指令格式

图形表现：



ST 表现：

```
FC_CubicRoot(fValue:= );
```

2、相关变量

变量名称	名称	数据类型	变量属性	描述
fValue	数据	LREAL	输入变量	需要被开立方根的数据

3、具体使用

（1）声明和函数调用

```
PROGRAM POU
VAR
    lrData  :LREAL;
    rResult :LREAL;
END_VAR
```

```
rResult := FC_CubicRoot(fValue:= lrData);
```

（2）写值与监控

表达式	类型	值	准备值
lrData	LREAL	123.45	
rResult	LREAL	4.9792473177376086	

```
rResult 4.98 := FC_CubicRoot(fValue:= lrData 123 );
RETURN
```

符合预期：



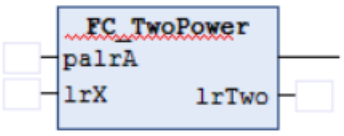
062 一元 n 次方程

用于数学运算场合、运动控制中的跟随曲线拟合、曲线速度求解等

FC_TwoPower 一元二次方程

函数功能：已知 2 次函数的系数和 X，求解 Y
公式：Y=a0+a1*X+a2*X^2

1、指令格式
图形表现：　　ST 表现：




```
FC_TwoPower(palrA:=  , lrX:=  , lrTwo=> );
```

2、相关变量

变量名称	名称	数据类型	变量属性	描述
palrA	公式系数数据	POINTER TO ARRAY[0..2] OF LREAL	输入变量	指向函数系数： pafA[0],即表达式中的 a0 pafA[1],即表达式中的 a1 pafA[2],即表达式中的 a2
lrX	自变量值	LREAL	输入变量	自变量值
lrTwo	方程运算结果	LREAL	输出变量	2 次方程运算结果

3、具体使用
（1）声明和函数调用

```
PROGRAM POU
VAR
    lrData  :ARRAY[0..2] OF LREAL :=[1,2,3];
    lrResult :LREAL;
END_VAR
```

```
FC_TwoPower(palrA:= ADR(lrData) , lrX:= 10 , lrTwo=> lrResult);
```

（2）写值与监控

Device.Application.POU

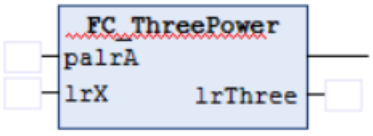
表达式	类型	值	准备值	地址
lrData	ARRAY [0..2] OF LREAL			
lrData[0]	LREAL	1		
lrData[1]	LREAL	2		
lrData[2]	LREAL	3		
lrResult	LREAL	321		

```
1  FC_TwoPower(palrA:= ADR(lrData) , lrX:= 10 , lrTwo=> lrResult 321 );
```

FC_ThreePower 一元三次方程

函数功能：已知 3 次函数的系数和 X，求解 Y
公式：Y=a0+a1*X+a2*X^2+a3*X^3

1、指令格式
图形表现：　　ST 表现：




```
FC_ThreePower(palrA:= , lrX:= , lrThree=> )
```

2、相关变量

变量名称	名称	数据类型	变量属性	描述
palrA	公式系数数据	POINTER TO ARRAY[0..3] OF LREAL	输入变量	指向函数系数： pafA[0],即表达式中的 a0 pafA[1],即表达式中的 a1 pafA[2],即表达式中的 a2 pafA[3],即表达式中的 a3

lrX	自变量值	LREAL	输入变量	自变量值
lrThree	方程运算结果	LREAL	输出变量	3 次方程运算结果

3、具体使用

（1）声明和函数调用

```
PROGRAM POU
VAR
    lrData  :ARRAY[0..3] OF LREAL :=[1,2,3,4];
    lrResult :LREAL;
END_VAR
```

```
FC_ThreePower(palrA:= ADR(lrData), lrX:= 10, lrThree=>lrResult );
```

（2）写值与监控

表达式	类型	值	准备值	地址
lrData	ARRAY [0..3] OF LREAL			
lrData[0]	LREAL	1		
lrData[1]	LREAL	2		
lrData[2]	LREAL	3		
lrData[3]	LREAL	4		
lrResult	LREAL	4321		

```
1
2
3   FC_ThreePower(palrA:= ADR(lrData), lrX:= 10, lrThree=>lrResult 4.32E+03 );
4
5
6
```

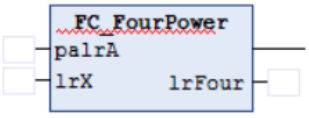
FC_FourPower 一元四次方程

函数功能：已知 4 次函数的系数和 X，求解 Y

公式：Y=a0+a1*X+a2*X^2+a3*X^3+a4*X^4

1、指令格式

图形表现：



ST 表现：

```
FC_FourPower(palrA:= , lrX:= , lrFour=> );
```

2、相关变量

变量名称	名称	数据类型	变量属性	描述
palrA	公式系数数据	POINTER TO ARRAY[0..4] OF LREAL	输入变量	指向函数系数: pafA[0],即表达式中的 a0 pafA[1],即表达式中的 a1 pafA[2],即表达式中的 a2 pafA[3],即表达式中的 a3 pafA[4],即表达式中的 a4
lrX	自变量值	LREAL	输入变量	自变量值
lrFour	方程运算结果	LREAL	输出变量	4 次方程运算结果

3、具体使用

（1）声明和函数调用

```
PROGRAM POU
VAR
    lrData  :ARRAY[0..4] OF LREAL :=[1,2,3,4,5];
    lrResult :LREAL;
END_VAR
```

```
FC_FourPower(palrA:= ADR(lrData), lrX:= 10, lrFour=>lrResult );
```

（2）写值与监控

表达式	类型	值	准备值	地
lrData	ARRAY [0..4] OF LREAL			
lrData[0]	LREAL	1		
lrData[1]	LREAL	2		
lrData[2]	LREAL	3		
lrData[3]	LREAL	4		
lrData[4]	LREAL	5		
lrResult	LREAL	54321		

```
1
2
3   FC_FourPower(palrA:= ADR(lrData), lrX:= 10, lrFour=>lrResult 5.43E+04 );
4
```

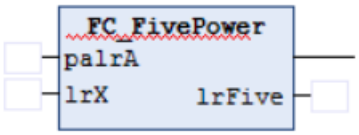
FC_FivePower 一元五次方程

函数功能：已知 5 次函数的系数和 X，求解 Y

公式：Y=a0+a1*X+a2*X^2+a3*X^3+a4*X^4+a5*X^5

1、指令格式

图形表现：



ST 表现：

```
FC_FivePower(palrA:= , lrX:= , lrFive=> );
```

2、相关变量

变量名称	名称	数据类型	变量属性	描述
palrA	公式系数数据	POINTER TO ARRAY[0..5] OF LREAL	输入变量	指向函数系数： pafA[0],即表达式中的 a0 pafA[1],即表达式中的 a1 pafA[2],即表达式中的 a2 pafA[3],即表达式中的 a3 pafA[4],即表达式中的 a4 pafA[5],即表达式中的 a5
lrX	自变量值	LREAL	输入变量	自变量值
lrFive	方程运算结果	LREAL	输出变量	5 次方程运算结果

3、具体使用

（1）声明和函数调用

```
PROGRAM POU
VAR
    lrData  :ARRAY[0..5] OF LREAL :=[1,2,3,4,5,6];
    lrResult :LREAL;
END_VAR
```

```
FC_FivePower(palrA:= ADR(lrData), lrX:= 10, lrFive=>lrResult );
```

（2）写值与监控

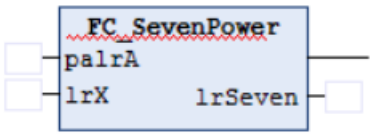
表达式	类型	值	准备值	地址
lrData	ARRAY [0..5] OF LREAL			
lrData[0]	LREAL	1		
lrData[1]	LREAL	2		
lrData[2]	LREAL	3		
lrData[3]	LREAL	4		
lrData[4]	LREAL	5		
lrData[5]	LREAL	6		
lrResult	LREAL	654321		

```
1
2   FC_FivePower(palrA:= ADR(lrData), lrX:= 10, lrFive=>lrResult 6.54E+05 );
3
```

FC_SevenPower 一元七次方程

函数功能：已知 7 次函数的系数和 X，求解 Y
公式：Y=a0+a1*X+a2*X^2+a3*X^3+a4*X^4+a5*X^5+a6*X^6+a7*X^7

1、指令格式
图形表现：　　ST 表现：




```
FC_SevenPower(palrA:= , lrX:= , lrSeven=> );
```

2、相关变量

变量名称	名称	数据类型	变量属性	描述
palrA	公式系数数据	POINTER TO ARRAY[0..7] OF LREAL	输入变量	指向函数系数： pafA[0],即表达式中的 a0 pafA[1],即表达式中的 a1 pafA[2],即表达式中的 a2 pafA[3],即表达式中的 a3 pafA[4],即表达式中的 a4 pafA[5],即表达式中的 a5 pafA[6],即表达式中的 a6 pafA[7],即表达式中的 a7
lrX	自变量值	LREAL	输入变量	自变量值
lrSeven	方程运算结果	LREAL	输出变量	7 次方程运算结果

3、具体使用
（1）声明和函数调用

```
PROGRAM POU
VAR
    lrData  :ARRAY[0..7] OF LREAL :=[1,2,3,4,5,6,7,8];
    lrResult :LREAL;
END_VAR
```

```
FC_SevenPower(palrA:= ADR(lrData), lrX:= 10, lrSeven=>lrResult );
```

（2）写值与监控

表达式	类型	值	准备值	地址
lrData	ARRAY [0..7] OF LREAL			
lrData[0]	LREAL	1		
lrData[1]	LREAL	2		
lrData[2]	LREAL	3		
lrData[3]	LREAL	4		
lrData[4]	LREAL	5		
lrData[5]	LREAL	6		
lrData[6]	LREAL	7		
lrData[7]	LREAL	8		
lrResult	LREAL	87654321		

```
1
2   FC_SevenPower(palrA:= ADR(lrData), lrX:= 10, lrSeven=>lrResult 8.77E+07 );
3
```

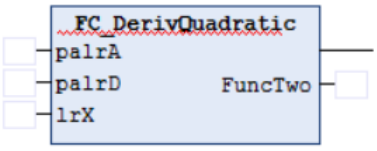
063 一元 n 次函数导数

用于数学运算场合、运动控制中的跟随曲线拟合、曲线速度求解等

FC_DerivQuadratic 一元二次导数

函数功能：已知 2 次函数的系数和 X，求解在 X 点的导函数值，以及函数的导函数系数。
公式： Y'= a1 +2*a2*X

1、指令格式
图形表现：



ST 表现：

```
FC_DerivQuadratic(
    palrA:= ,
    palrD:= ,
    lrX:= ,
    FuncTwo=> );
```

2、相关变量

变量名称	名称	数据类型	变量属性	描述
palrA	函数系数	POINTER TO ARRAY[0..2] OF LREAL	输入变量	指向函数系数: pafA[0],即表达式中的 a0 pafA[1],即表达式中的 a1 pafA[2],即表达式中的 a2
palrD	导函数系数	POINTER TO ARRAY[0..1] OF LREAL	输入变量	指向二次导函数系数: pafD[0],即表达式中的 a1 pafD[1],即表达式中的 2*a2
lrX	自变量值	LREAL	输入变量	自变量值

FuncTwo	运算结果	LREAL	输出变量	二次导函数结果

3、具体使用

（1）声明和函数调用

```
PROGRAM POU
VAR
    lrDataA :ARRAY[0..2] OF LREAL :=[1,2,3];
    lrDataD :ARRAY[0..1] OF LREAL :=[1,2];
    lrResult :LREAL;
END_VAR
```

```
FC_DerivQuadratic(palrA:= ADR(lrDataA), palrD:= ADR(lrDataD), lrX:= 10, FuncTwo=> lrResult);
```

（2）写值与监控

Device.Application.POU

表达式	类型	值	准备值	地址	注释
lrDataA	ARRAY [0..2] OF LREAL				
lrDataA[0]	LREAL	1			
lrDataA[1]	LREAL	2			
lrDataA[2]	LREAL	3			
lrDataD	ARRAY [0..1] OF LREAL				
lrDataD[0]	LREAL	2			
lrDataD[1]	LREAL	6			
lrResult	LREAL	62			

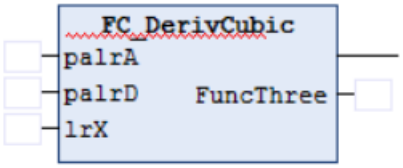
```
1
2   FC_DerivQuadratic(palrA:= ADR(lrDataA), palrD:= ADR(lrDataD), lrX:= 10, FuncTwo=> lrResult 62 );
3
```

FC_DerivCubic 一元三次导数

函数功能：已知 3 次函数的系数和 X，求解在 X 点的导函数值，以及函数的导函数系数。
公式： Y'= a1 +2*a2*X+3*a3*X^2

1、指令格式

图形表现：



ST 表现：

```
FC_DerivCubic
    (palrA:= ,
     palrD:= ,
     lrX:= ,
     FuncThree=> );
```

2、相关变量

变量名称	名称	数据类型	变量属性	描述
palrA	公式系数数据	POINTER TO	输入变量	指向函数系数：

		ARRAY[0..3] OF LREAL		pafA[0],即表达式中的 a0 pafA[1],即表达式中的 a1 pafA[2],即表达式中的 a2 pafA[3],即表达式中的 a3
palrD	导函数系数	POINTER TO ARRAY[0..2] OF LREAL	输入变量	指向二次导函数系数: pafD[0],即表达式中的 a1 pafD[1],即表达式中的 2*a2 pafD[2],即表达式中的 3*a3
lrX	自变量值	LREAL	输入变量	自变量值
FuncThree	运算结果	LREAL	输出变量	3 次导函数结果

3、具体使用

（1）声明和函数调用

```
PROGRAM POU
VAR
    lrDataA :ARRAY[0..3] OF LREAL :=[1,2,3,4];
    lrDataD :ARRAY[0..2] OF LREAL :=[1,2,3];
    lrResult :LREAL;
END_VAR
```

```
FC_DerivCubic  (palrA:= ADR(lrDataA), palrD:= ADR(lrDataD), lrX:= 10, FuncThree=> lrResult);
```

（2）写值与监控

Device.Application.POU

表达式	类型	值	准备值	地址	注释
lrDataA	ARRAY [0..3] OF LREAL				
lrDataA[0]	LREAL	1			
lrDataA[1]	LREAL	2			
lrDataA[2]	LREAL	3			
lrDataA[3]	LREAL	4			
lrDataD	ARRAY [0..2] OF LREAL				
lrDataD[0]	LREAL	2			
lrDataD[1]	LREAL	6			
lrDataD[2]	LREAL	12			
lrResult	LREAL	1262			

```
FC_DerivCubic  (palrA:= ADR(lrDataA), palrD:= ADR(lrDataD), lrX:= 10, FuncThree=> lrResult 1.26E+03 );
```

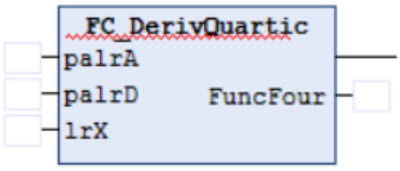
FC_DerivQuartic 一元四次导数

函数功能：已知 4 次函数的系数和 X，求解在 X 点的导函数值，以及函数的导函数系数。

公式： Y'= a1 +2*a2*X+3*a3*X^2+4*a4*X^3

1、指令格式

图形表现：



ST 表现：

```
FC_DerivQuartic(
    palrA:= ,
    palrD:= ,
    lrX:= ,
    FuncFour=> )
```

2、相关变量

变量名称	名称	数据类型	变量属性	描述
palrA	公式系数数据	POINTER TO ARRAY[0..4] OF LREAL	输入变量	指向函数系数: pafA[0],即表达式中的 a0 pafA[1],即表达式中的 a1 pafA[2],即表达式中的 a2 pafA[3],即表达式中的 a3 pafA[4],即表达式中的 a4
palrD	导函数系数	POINTER TO ARRAY[0..3] OF LREAL	输入变量	指向二次导函数系数: pafD[0],即表达式中的 a1 pafD[1],即表达式中的 2*a2 pafD[2],即表达式中的 3*a3 pafD[3],即表达式中的 4*a4
lrX	自变量值	LREAL	输入变量	自变量值
FuncFour	运算结果	LREAL	输出变量	4 次导函数结果

3、具体使用

（1）声明和函数调用

```
PROGRAM POU
VAR
    lrDataA :ARRAY[0..4] OF LREAL :=[1,2,3,4,5];
    lrDataD :ARRAY[0..3] OF LREAL :=[1,2,3,4];
    lrResult :LREAL;
END_VAR
```

```
FC_DerivQuartic (palrA:= ADR(lrDataA), palrD:= ADR(lrDataD), lrX:= 10, FuncFour=> lrResult);
```

（2）写值与监控

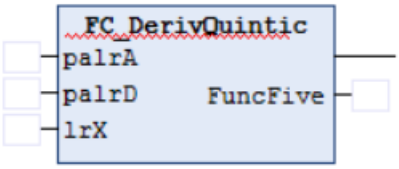
表达式	类型	值	准备值	地址	注释
lrDataA	ARRAY [0..4] OF LREAL				
lrDataA[0]	LREAL	1			
lrDataA[1]	LREAL	2			
lrDataA[2]	LREAL	3			
lrDataA[3]	LREAL	4			
lrDataA[4]	LREAL	5			
lrDataD	ARRAY [0..3] OF LREAL				
lrDataD[0]	LREAL	2			
lrDataD[1]	LREAL	6			
lrDataD[2]	LREAL	12			
lrDataD[3]	LREAL	20			
lrResult	LREAL	21262			

```
1
2   FC_DerivQuartic (palrA:= ADR(lrDataA), palrD:= ADR(lrDataD), lrX:= 10, FuncFour=> lrResult 2.13E+04 );
3
4
```

FC_DerivQuintic 一元五次导数

函数功能：已知 5 次函数的系数和 X，求解在 X 点的导函数值，以及函数的导函数系数。
公式： Y'= a1 +2*a2*X+3*a3*X^2+4*a4*X^3+5*a5*X^4

1、指令格式
图形表现： ST 表现：




```
FC_DerivQuintic(
    palrA:= ,
    palrD:= ,
    lrX:= ,
    FuncFive=> )
```

2、相关变量

变量名称	名称	数据类型	变量属性	描述
palrA	公式系数数据	POINTER TO ARRAY[0..5] OF LREAL	输入变量	指向函数系数: pafA[0],即表达式中的 a0 pafA[1],即表达式中的 a1 pafA[2],即表达式中的 a2 pafA[3],即表达式中的 a3 pafA[4],即表达式中的 a4 pafA[5],即表达式中的 a5
palrD	导函数系数	POINTER TO ARRAY[0..4] OF LREAL	输入变量	指向二次导函数系数: pafD[0],即表达式中的 a1 pafD[1],即表达式中的 2*a2 pafD[2],即表达式中的 3*a3 pafD[3],即表达式中的 4*a4 pafD[4],即表达式中的 5*a5

lrX	自变量值	LREAL	输入变量	自变量值
FuncFive	运算结果	LREAL	输出变量	5 次导函数结果

3、具体使用

（1）声明和函数调用

```
PROGRAM POU
VAR
    lrDataA :ARRAY[0..5] OF LREAL :=[1,2,3,4,5,6];
    lrDataD :ARRAY[0..4] OF LREAL :=[1,2,3,4,5];
    lrResult :LREAL;
END_VAR
```

```
FC_DerivQuintic (palrA:= ADR(lrDataA), palrD:= ADR(lrDataD), lrX:= 10, FuncFive=> lrResult);
```

（2）写值与监控

表达式	类型	值	准备值	地址	注释
lrDataA	ARRAY [0..5] OF LREAL				
lrDataA[0]	LREAL	1			
lrDataA[1]	LREAL	2			
lrDataA[2]	LREAL	3			
lrDataA[3]	LREAL	4			
lrDataA[4]	LREAL	5			
lrDataA[5]	LREAL	6			
lrDataD	ARRAY [0..4] OF LREAL				
lrDataD[0]	LREAL	2			
lrDataD[1]	LREAL	6			
lrDataD[2]	LREAL	12			
lrDataD[3]	LREAL	20			
lrDataD[4]	LREAL	30			
lrResult	LREAL	321262			

```
1
2   FC_DerivQuintic (palrA:= ADR(lrDataA), palrD:= ADR(lrDataD), lrX:= 10, FuncFive=> lrResult 3.21E+05 );
3
```

064 解一元 n 次方程

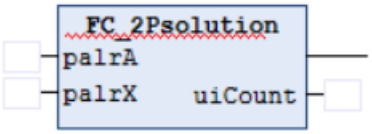
FC_2Psolution 解一元二次方程

函数功能：已知方程表达式的系数，求解方程的 X 值

公式：0=a0+a1*X+a2*X^2

1、指令格式

图形表现：　　ST 表现：




```
FC_2PSolution(palrA:= , palrX:= , uiCount=> )
```

2、相关变量

变量名称	名称	数据类型	变量属性	描述
palrA	公式系数数据	POINTER TO ARRAY[0..2] OF LREAL	输入变量	指向函数系数: pafA[0],即表达式中的 a0 pafA[1],即表达式中的 a1 pafA[2],即表达式中的 a2
palrX	自变量值	POINTER TO ARRAY[0..1] OF LREAL	输入变量	指向解数组: palrX[0]表示第一个解 palrX[1]表示第二个解
uiCount	方程运算结果	UINT	输出变量	求得的实根个数

3、具体使用

（1）声明和函数调用

```
PROGRAM PLC_PRG
VAR
    palrA2f : ARRAY[0..2] OF LREAL:=[2,6,4];
    palrx2 : ARRAY[0..1] OF LREAL;
    uiCountA  :UINT;
END_VAR
```

```
FC_2PSolution(palrA:= ADR(palrA2f), palrX:= ADR(palrx2), uiCount=>uiCountA );
```

（2）写值与监控

Device.Application.PLC_PRG

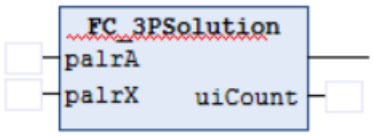
表达式	类型	值	准备值	地址	注释
palrA2f	ARRAY [0..2] OF LREAL				
palrA2f[0]	LREAL	2			
palrA2f[1]	LREAL	6			
palrA2f[2]	LREAL	4			
palrx2	ARRAY [0..1] OF LREAL				
palrx2[0]	LREAL	-0.5			
palrx2[1]	LREAL	-1			
uiCountA	UINT	2			

```
1
2
3   FC_2PSolution(palrA:= ADR(palrA2f), palrX:= ADR(palrx2), uiCount=>uiCountA 2 );
4
```

FC_3Psolution 解一元三次方程

函数功能：已知方程表达式的系数，求解方程的 X 值
公式：0=a0+a1*X+a2*X^2+ a3*X^3

1、指令格式
图形表现：　　ST 表现：




```
FC_3PSolution(palrA:= , palrX:= , uiCount=> )
```

2、相关变量

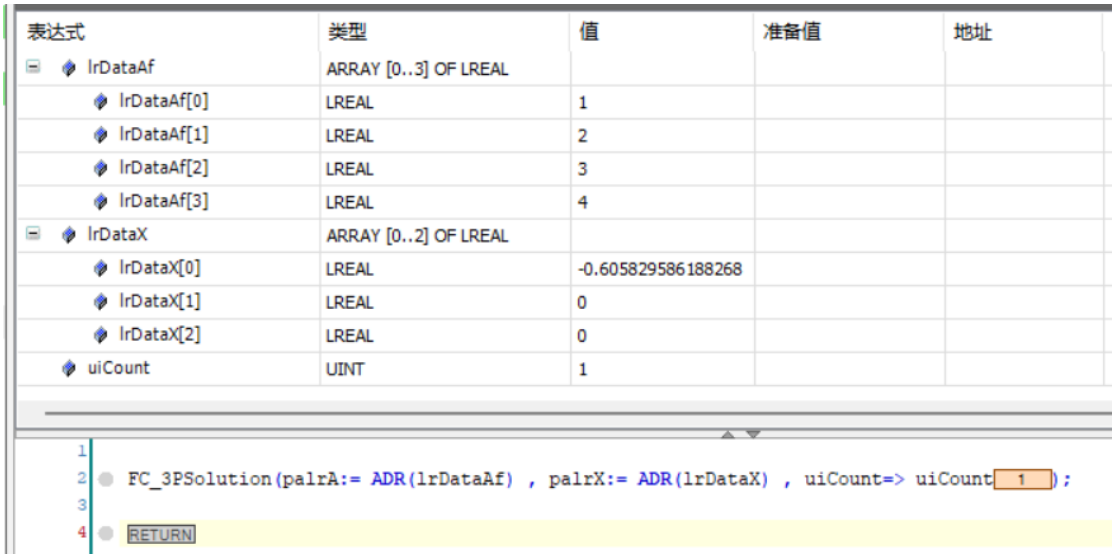
变量名称	名称	数据类型	变量属性	描述
palrA	公式系数数据	POINTER TO ARRAY[0..3] OF LREAL	输入变量	指向函数系数： pafA[0],即表达式中的 a0 pafA[1],即表达式中的 a1 pafA[2],即表达式中的 a2 pafA[3],即表达式中的 a3
palrX	自变量值	POINTER TO ARRAY[0..2] OF LREAL	输入变量	指向解数组： palrX[0]表示第一个解 palrX[1]表示第二个解 palrX[2]表示第三个解
uiCount	方程运算结果	UINT	输出变量	求得的实根个数

3、具体使用
（1）声明和函数调用

```
PROGRAM POU
VAR
    lrDataAf    :ARRAY[0..3] OF LREAL :=[1,2,3,4];
    lrDataX     :ARRAY[0..2] OF LREAL;
    uiCount     :UINT;
END_VAR
```

```

FC_3PSolution(palrA:= ADR(lrDataAf) , palrX:= ADR(lrDataX) , uiCount=> uiCount);


```

（2）写值与监控



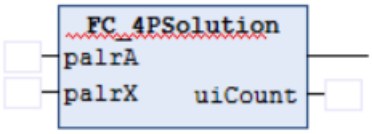
表达式	类型	值	准备值	地址
lrDataAf	ARRAY [0..3] OF LREAL			
lrDataAf[0]	LREAL	1		
lrDataAf[1]	LREAL	2		
lrDataAf[2]	LREAL	3		
lrDataAf[3]	LREAL	4		
lrDataX	ARRAY [0..2] OF LREAL			
lrDataX[0]	LREAL	-0.605829586188268		
lrDataX[1]	LREAL	0		
lrDataX[2]	LREAL	0		
uiCount	UINT	1		

```
1
2   FC_3PSolution(palrA:= ADR(lrDataAf) , palrX:= ADR(lrDataX) , uiCount=> uiCount 1 );
3
4   RETURN
```

FC_4Psolution 解一元四次方程

函数功能：已知方程表达式的系数，求解方程的 X 值
公式：0=a0+a1*X+a2*X^2+ a3*X^3+ a4*X^4

1、指令格式
图形表现：　　ST 表现：




```
FC_4PSolution(palrA:= , palrX:= , uiCount=> )
```

2、相关变量

变量名称	名称	数据类型	变量属性	描述
palrA	公式系数数据	POINTER TO ARRAY[0..4] OF LREAL	输入变量	指向函数系数: pafA[0],即表达式中的 a0 pafA[1],即表达式中的 a1 pafA[2],即表达式中的 a2 pafA[3],即表达式中的 a3 pafA[4],即表达式中的 a4
palrX	自变量值	POINTER TO ARRAY[0..3] OF LREAL	输入变量	指向解数组: palrX[0]表示第一个解 palrX[1]表示第二个解 palrX[2]表示第三个解 palrX[3]表示第四个解
uiCount	方程运算结果	UINT	输出变量	求得的实根个数

3、具体使用

（1）声明和函数调用

```
PROGRAM POU
VAR
    lrDataAf    :ARRAY[0..4] OF LREAL :=[1,2,3,4,5];
    lrDataX     :ARRAY[0..3] OF LREAL;
    uiCount     :UINT;
END_VAR
```

```
FC_4PSolution(palrA:= ADR(lrDataAf) , palrX:= ADR(lrDataX) , uiCount=> uiCount);
```

（2）写值与监控

表达式	类型	值	准备值	地址	注释
palrA4f	ARRAY [0..4] OF LREAL				
palrA4f[0]	LREAL	-1			
palrA4f[1]	LREAL	0.1			
palrA4f[2]	LREAL	5			
palrA4f[3]	LREAL	1			
palrA4f[4]	LREAL	-3			
palrx4	ARRAY [0..3] OF LREAL				
palrx4[0]	LREAL	0.4437...			
palrx4[1]	LREAL	-0.5433...			
palrx4[2]	LREAL	1.4120...			
palrx4[3]	LREAL	-0.9790...			
uiCountA	UINT	4			

```
1
2
3   FC_4PSolution(palrA:= ADR(palrA4f), palrX:= ADR(palrx4), uiCount=>uiCountA 4 );
4
5
```

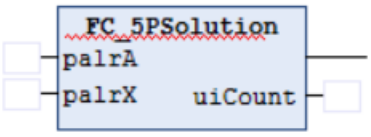
FC_5Psolution 解一元五次方程

函数功能：已知方程表达式的系数，求解方程的 X 值

公式：0=a0+a1*X+a2*X^2+ a3*X^3+ a4*X^4+ a5*X^5

1、指令格式

图形表现：



ST 表现：

```
FC_5PSolution(palrA:= , palrX:= , uiCount=> )
```

2、相关变量

变量名称	名称	数据类型	变量属性	描述
palrA	公式系数数据	POINTER TO ARRAY[0..5] OF LREAL	输入变量	指向函数系数: pafA[0],即表达式中的 a0 pafA[1],即表达式中的 a1 pafA[2],即表达式中的 a2 pafA[3],即表达式中的 a3 pafA[4],即表达式中的 a4 pafA[5],即表达式中的 a5
palrX	自变量值	POINTER TO ARRAY[0..4] OF LREAL	输入变量	指向解数组: palrX[0]表示第一个解 palrX[1]表示第二个解 palrX[2]表示第三个解 palrX[3]表示第四个解 palrX[4]表示第五个解
uiCount	方程运算结果	UINT	输出变量	求得的实根个数

3、具体使用

（1）声明和函数调用

```
PROGRAM PLC_PRG
VAR
    lrDataAf :ARRAY[0..5] OF LREAL :=[-1,5,4,-3,1,2];
    lrDataX  :ARRAY[0..4] OF LREAL;
    uiCountA  :UINT;
END_VAR
```

```

FC_5PSolution(palrA:= ADR(lrDataAf), palrX:= ADR(lrDataX), uiCount=> uiCountA);
```

（2）写值与监控

lrDataAf	ARRAY [0..5] OF LREAL					
lrDataAf[0]	LREAL	-1				
lrDataAf[1]	LREAL	5				
lrDataAf[2]	LREAL	4				
lrDataAf[3]	LREAL	-3				
lrDataAf[4]	LREAL	1				
lrDataAf[5]	LREAL	2				
lrDataX	ARRAY [0..4] OF LREAL					
lrDataX[0]	LREAL	-0.9999...				
lrDataX[1]	LREAL	11.447...				
lrDataX[2]	LREAL	0				
lrDataX[3]	LREAL	0				
lrDataX[4]	LREAL	0				
uiCountA	UINT	2				

```

FC_5PSolution(palrA:= ADR(lrDataAf), palrX:= ADR(lrDataX), uiCount=> uiCountA 2 );
```

065 正余弦方程

sin_calculation

待发

cos_calculation

待发

solve_cos_function

待发

solve_sin_function

待发

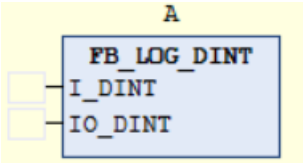
10 Others

FB_LOG_DINT 数据变化记录

功能块说明：数据记录，数据类型 DINT。当产生一个不等于数组最小下标当前值的数值的时候，把当前值存入数组。例如 Datain 当前值不等于 arData[0]的时候，存入 arData[0]；原本的数值都后移。

1、指令格式

图形表现



ST 表现

```
A(I_DINT:= , IO_DINT:= );
```

2、相关变量说明

变量名称	名称	数据类型	变量属性	描述
I_DINT	需要记录的变量	DINT	输入变量	需要记录的变量
IO_DINT	存入的数组	ARRAY[*]OF DINT	输出变量	记录存入的数组

3、具体使用：

Device.Application.P2

表达式	类型	值	准备值	地址	注释
datain	DINT	14			
arData	ARRAY [0..10] OF DINT				
arData[0]	DINT	14			
arData[1]	DINT	13			
arData[2]	DINT	12			
arData[3]	DINT	0			
arData[4]	DINT	0			
arData[5]	DINT	0			
arData[6]	DINT	0			
arData[7]	DINT	0			
arData[8]	DINT	0			
arData[9]	DINT	0			
arData[10]	DINT	0			
logDint	FB_LOG_DINT				

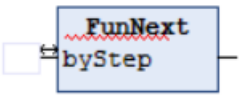
```
1  logDint(I_DINT 14 := datain 14 , IO_DINT 16#000001B0A09E0F18 := arData);RETURN
```

FunNext 步序加一

函数说明：INT 变量的当前值+1 输出

1、指令格式

图形表现：



ST 表现：

```
FunNext(byStep:= iA);
```

2、 相关变量

变量名	名称	数据类型	输入输出属性	描述
byStep	需要+1 的变量	INT	输入输出变量	byStep:=byStep + 1;

3、 使用实例：iA 不断的再加一

表达式	类型	值
iA	INT	15120

```
1  FunNext(byStep:= iA 15120 );
2
```

01 基础控制

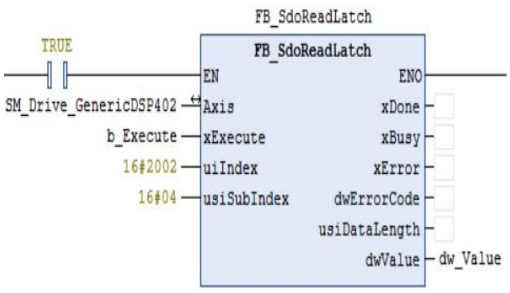
00 SDO 相关

FB_SdoReadLatch SDO 带锁读取

功能块说明：对 SDO 参数值进行读取。防止因大量 SDO 读取操作导致报错，并简化无需写入数据长度。

1、指令格式

图形表现



ST 表现

```
FB_SdoReadLatch(
                Axis:=SM_Drive_GenericDSP402 ,
                xExecute:=b_Execute,
                uiIndex:=16#2002 ,
                usiSubIndex:=16#04 ,
                xDone=> ,
                xBusy=> ,
                xError=> ,
                dwErrorCode=> ,
                usiDataLength=> ,
                dwValue=>dw_Value );
```

2、相关变量

（1）输入输出变量（Var In_Out）

变量名称	数据类型	默认值	有效范围	描述
Axis	AXIS_REF_ETC_SM3	/	/	指定轴，即 AXIS_REF_ETC_SM3 的一个实例

（2）输入变量（VAR_Input）

变量名称	数据类型	默认值	有效范围	描述

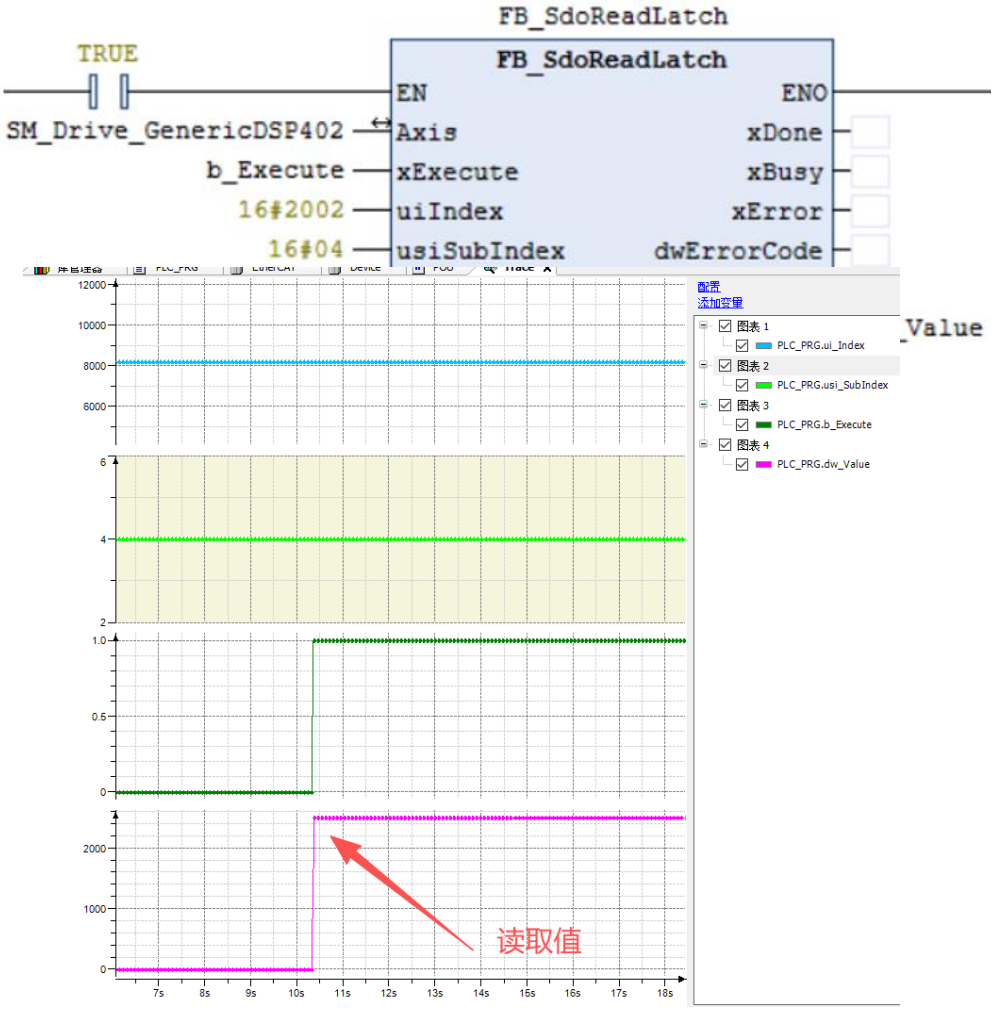
xExecute	BOOL	/	/	功能块触发
uiIndex	UINT	/	/	对象索引
usiSubIndex	USINT	/	/	对象子索引

（3）输出变量（VAR_Output）

输出变量	数据类型	默认值	有效范围	描述
xDone	BOOL	/	/	执行完成
xBusy	BOOL	/	/	执行中
xError	BOOL	/	/	错误
dwErrorCode	DWORD	/	/	错误代码
usiDataLength	USINT	/	/	数据长度
dwValue	DWORD	/	/	读取值

4、使用示例

```
VAR
    FB_SdoReadLatch: FB_SdoReadLatch;        //功能块声明
    b_Execute:BOOL;                          //触发功能块
    dw_Value:DWORD;                          //读取值
END_VAR
```

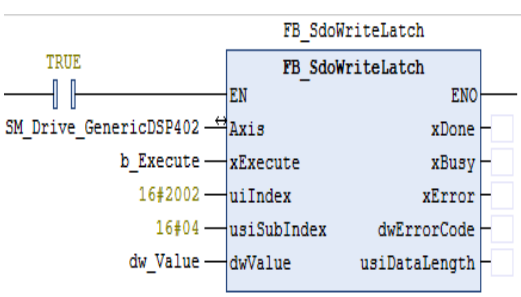


FB_SdoWriteLatch SDO 带锁写入

功能块说明：对 SDO 参数值进行写入。防止因大量 SDO 写入操作导致报错，并简化无需写入数据长度。

1、指令格式

图形表现



ST 表现

```
FB_SdoWriteLatch(
Axis:=SM_Drive_GenericDSP402 ,
xExecute:=b_Execute,
uiIndex:=16#2002 ,
usiSubIndex:=16#04 ,
dwValue:=dw_Value ,
xDone=> ,
xBusy=> ,
xError=> ,
dwErrorCode=> ,
usiDataLength=> );
```

2、相关变量

（1）输入输出变量（Var In_Out）

变量名称	数据类型	默认值	有效范围	描述
Axis	AXIS_REF_ETC_SM3	/	/	指定轴，即 AXIS_REF_ETC_SM3 的一个实例

（2）输入变量（VAR_Input）

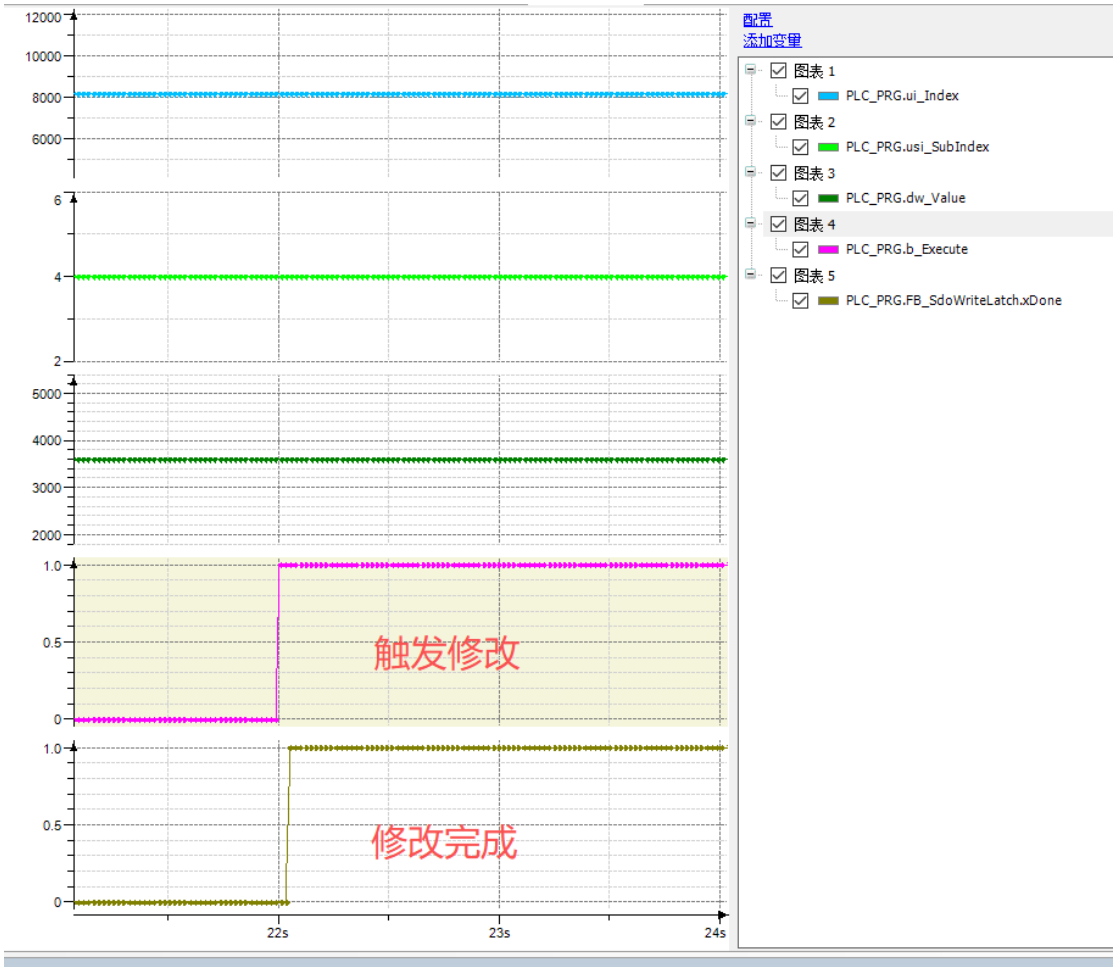
变量名称	数据类型	默认值	有效范围	描述
xExecute	BOOL	/	/	功能块触发
uiIndex	UINT	/	/	对象索引
usiSubIndex	USINT	/	/	对象子索引
dwValue	USINT	/	/	需要写入的值

（3）输出变量（VAR_Output）

输出变量	数据类型	默认值	有效范围	描述
xDone	BOOL	/	/	执行完成
xBusy	BOOL	/	/	执行中
xError	BOOL	/	/	错误
dwErrorCode	DWORD	/	/	错误代码
usiDataLength	USINT	/	/	数据长度

3、使用样例

```
VAR
    FB_SdoWriteLatch: FB_SdoWriteLatch;      //功能块声明
    b_Execute:BOOL;                          //触发功能块
    dw_Value:DWORD;                          //写入值
END_VAR

FB_SdoWriteLatch(
                    Axis:=SM_Drive_GenericDSP402 ,
                    xExecute:=b_Execute,
                    uiIndex:=16#2002 ,
                    usiSubIndex:=16#04 ,
                    dwValue:=dw_Value ,
                    xDone=> ,
                    xBusy=> ,
                    xError=> ,
                    dwErrorCode=> ,
                    usiDataLength=> );
```



01 轴基础控制

010_TEC_AxisModel

TEC_AxisModel 基础轴控功能块

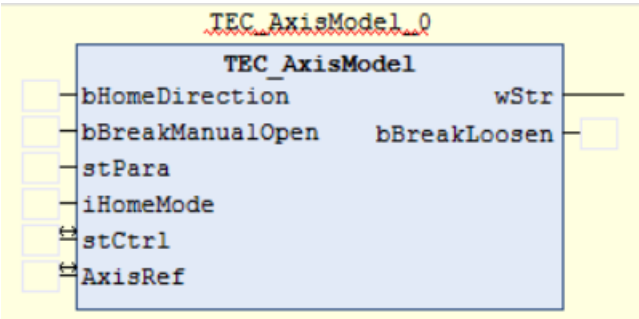
指令名称： TEC_AxisModel 轴基础运动功能块

功能块说明:

本功能块将常用的轴控功能都集合在一个 FB 中，通过结构体变量对轴进行控制。本功能块包含了轴的使能、停止、复位、总线复位、JOG 运行、回原点、绝对定位，相对定位、设定轴当前位置等功能。

1、指令格式

图形表现:



ST 表现:

```
Tec_Axis1(
    bHomeDirection:= ,
    bBreakManualOpen:= ,
    stPara:= ,
    iHomeMode:= ,
    stCtrl:= ,
    AxisRef:= ,
    wStr=> ,
    bBreakLoosen=> );
```

2、 相关变量

变量名	名称	数据类型	输入输出属性	描述
bHomeDirection	回原点方向	BOOL	输入变量	=True 表示先正向运行。仅在 iHomeMode=0/1/3 的时候有效
bBreakManualOpen	抱闸手动解除	BOOL	输入变量	抱闸手动打开
stPara	轴参数类	ST_AxisPara	输入变量	轴参数类变量。详见以下截图
iHomeMode	回原点模式	INT	输入变量	轴回原点模式设置。
stCtrl	轴控制类参数	ST_AxisCtrl	输入输出变量	轴控制类变量。详见以下截图
AxisREF	具体控制的轴	Axis_REF_SM3	输入输出变量	具体的 402 轴名称
wStr	运行信息	STRING	输出变量	运行状态显示
bBreakLoosen	松开抱闸	BOOL	输出变量	强制输出抱闸松开

StPara:

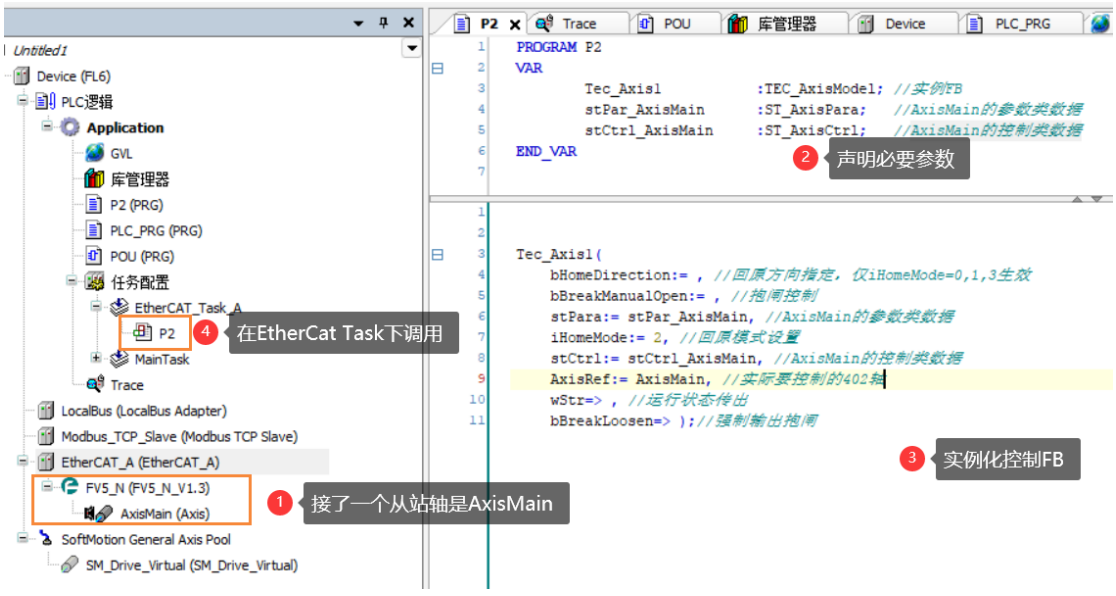
```
TYPE ST_AxisPara :
STRUCT
    rJogVel         :REAL:=100;//JOG速度
    rAcc            :REAL:=10000;//加速度
    rDec            :REAL:=10000;//减速度
    rJerk           :REAL:=10000;//减减速
    rStopDec        :REAL:=10000;//Stop的减速度
    rHomeVel        :REAL:=360;//回原速度
    rMaxLimit       :REAL;//最大软限位
    rMinLimit       :REAL;//最小软限位
    rHomeOffset     :REAL;//回原点偏移
END_STRUCT
END_TYPE
```

stCtrl：

```
TYPE st_AxisCtrl :
STRUCT
    bPower              :BOOL;//使能接口
    iStop               :INT;//停止
    bJogForward         :BOOL;//手动正转
    bJogBackwards       :BOOL;//手动反转
    iReset              :INT;//复位
    iHome               :INT;//回原点
    iSetPosition        :INT;//修改轴当前位置
    iMoveAbs            :INT;//绝对定位(**)
    iMoveRel            :INT;//相对定位
    bHomeSensor         :BOOL;//回原点传感器信号
    rPosition           :REAL;//定位位置值接口
    rVelocity           :REAL;//定位速度值接口
    bError              :BOOL;//报错传出接口
    udErrorId           :UDINT;//报错代码
    rActPos             :REAL;//当前位置
    rActVel             :REAL;//当前速度
    iAxisState          :INT;//轴状态
    iPauseAction        :INT;///暂停轴绝对定位      88暂停完成    -1解除暂停   -88 解除完成
    bHomeFlag       :BOOL;  //回原点标志位
END_STRUCT
END_TYPE
```

3、具体使用

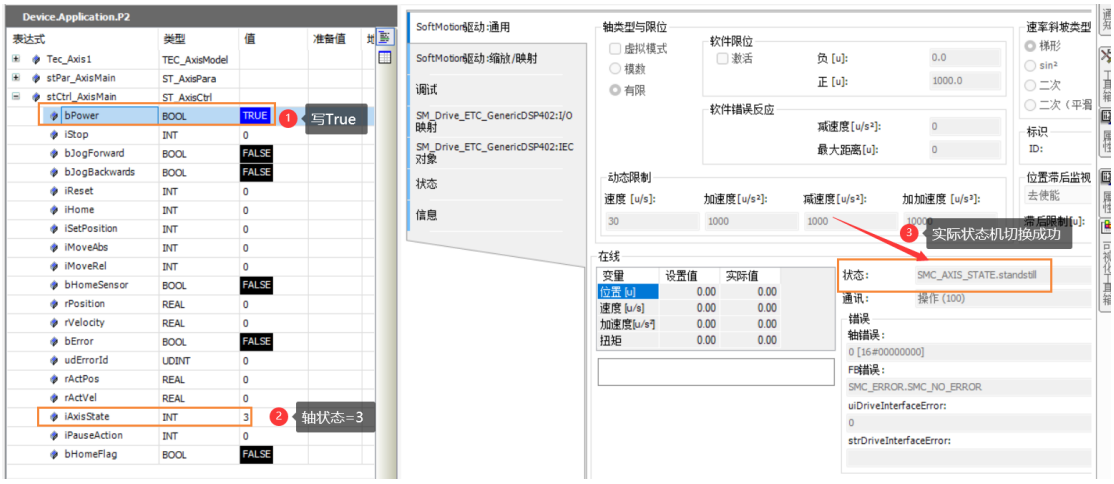
（1） 声明与实例化



（2） 详细功能介绍与使用

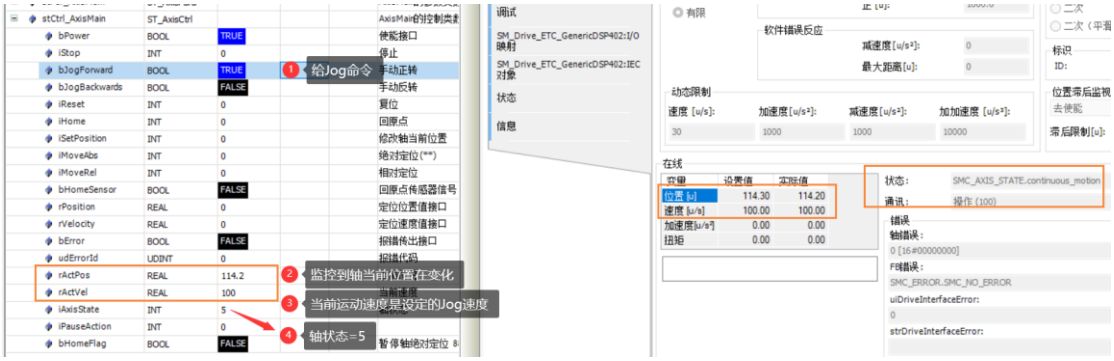
功能 1：使能

stCtrl.bPower:=TRUE 即可，若使能失败 stCtrl.bError=TRUE、并且输出 ErrorID。可以取 stCtrl.iAxisState=3，作为使能成功，并且没有报错的标志位。



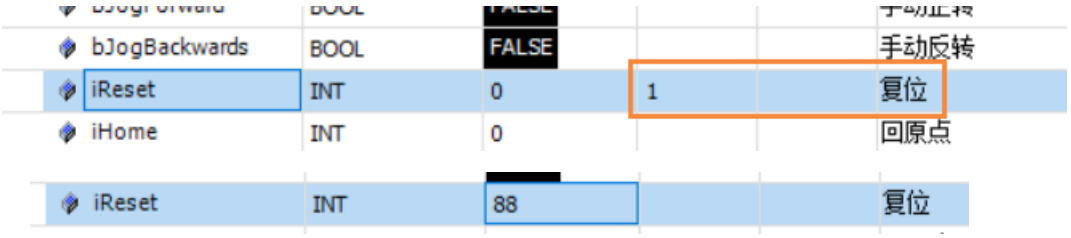
功能 2：JOG

通过 stPara.rJogVel、stPara.rAcc、stPara.rDec、stPara.rJeck 设置 Jog 运动的速度，加速度、减速度、Jeck 值。控制 JOG 运行，使用 stCtrl.bJogForward、stCtrl.bBackwards 等于 TRUE 的时候正向运动或者负向运动，等于 FALSE 的时候停止正向或者负向运动。若产生报错，stCtrl.bError=TRUE、并且输出 ErrorID。



功能 3：复位

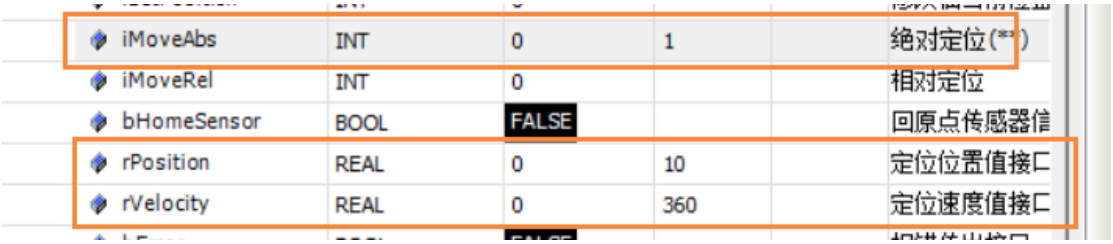
当发生可复位的报错时候，通过 stCtrl.iReset 给复位命令。FB 中做了判断，如果是总线报错，那么会执行总线复位的功能；如果是 MC 指令块报错，执行 Reset 命令；如果没有产生报错，则直接返回完成。复位命令是 stCtrl.iReset:=1；当 stCtrl.iReset=88 则完成复位



功能 4：绝对定位

通过 stPara.rAcc、stPara.rDec、stPara.rJeck 设置绝对定位的加速度、减速度、Jeck 值；绝对定位的位置和速度通过 stCtrl.rPostion 和 stCtrl.rVelocity 设置。开启绝对定位需要对 stCtrl.iMoveAbs 发 1 的命令，定位完成标志位为 stCtrl.iMoveAbs=88.
stCtrl.rPostion、stCtrl.rVelocity、 stCtrl.iMoveAbs 三个可以同时给值进去。在内部已处理了时序，规避由于没有参数导致绝对定位报错的问题。

给值：



运动中：



运动完成：

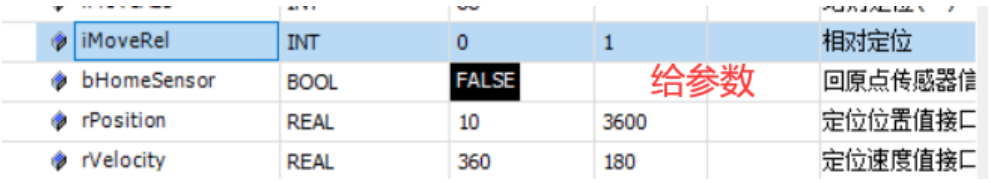


功能 5：相对定位

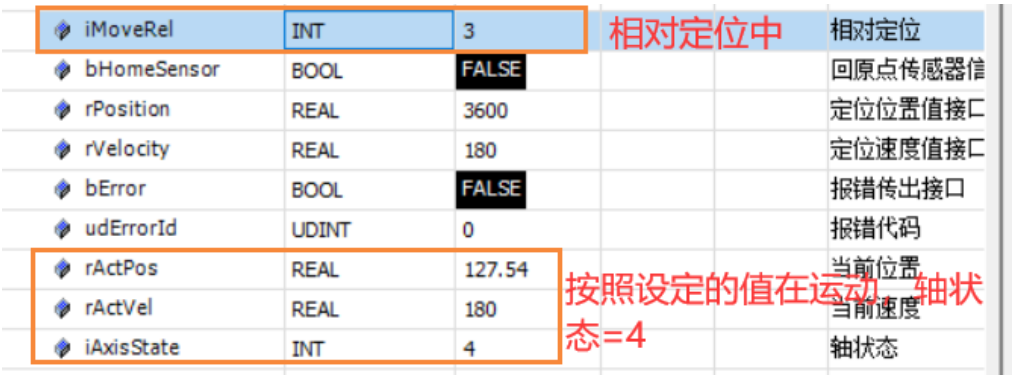
通过 stPara.rAcc、stPara.rDec、stPara.rJeck 设置相对定位的加速度、减速度、Jeck 值；定位的位置和速度通过 stCtrl.rPostion 和 stCtrl.rVelocity 设置。开启相对定位需要对 stCtrl.iMoveRel 发 1 的命令，定位完成标志位为 stCtrl.iMoveRel=88.

stCtrl.rPostion、stCtrl.rVelocity、 stCtrl.iMoveRel 三个可以同时给值进去。在内部已处理了时序，规避由于没有参数导致定位报错的问题。

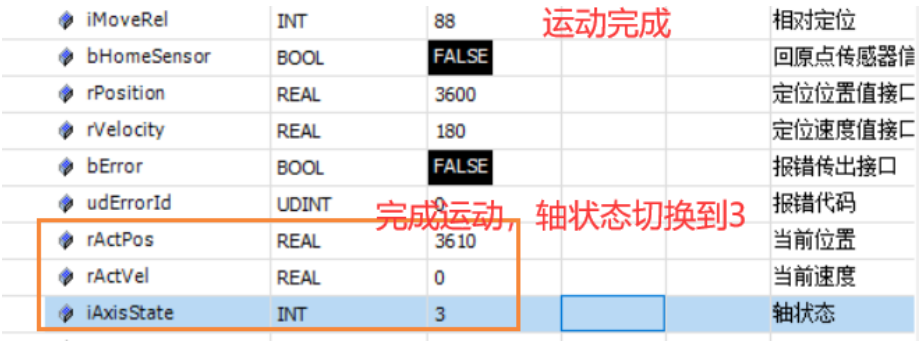
给参数：



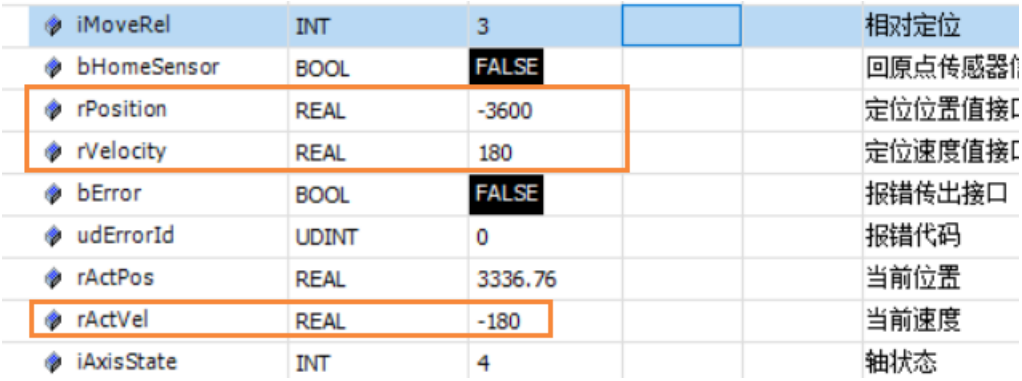
运动中：



运动完成：



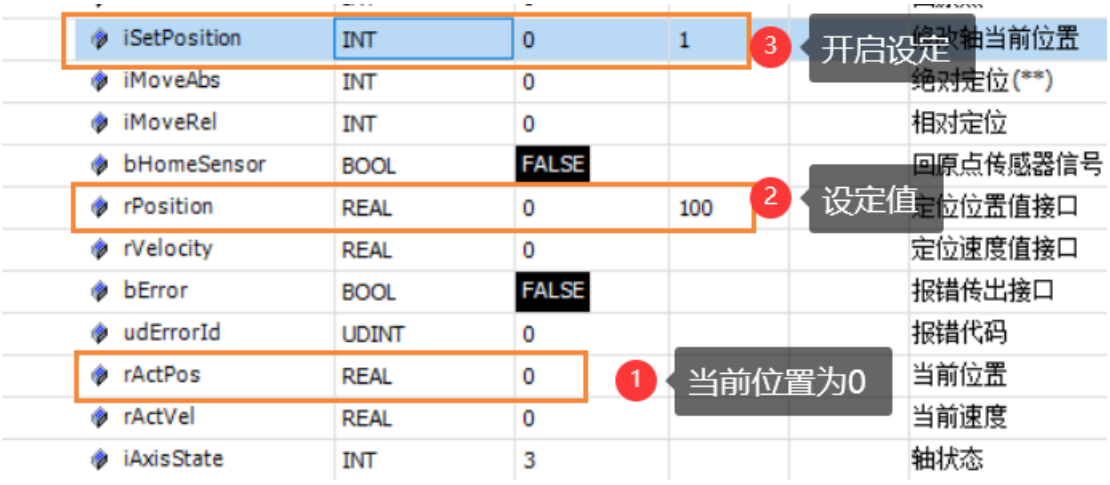
若要产生负向的相对运动；则定位位置给负值即可



功能 6：设定当前位置

通过设置 stCtrl.rPostion 的值，对 stCtrl.iSetPostion 写 1，可以把轴当前位置修改成任意值

给值:



修改完成:



功能 7：回原点

轴回原点根据实际接线（接 PLC 还是接驱动器 DI）和现场要求，可以对 FB 的回原点方式进行设置。

（1） iHomeMode =0 调用 SMC_Homing，寻找传感器原点。原点接在 PLC 的 DI。初始回原方向通过 bHomeDirection 设置，回原速度通过 stPara.rHomeVel 设置，触发 stCtrl.iHome=1，完成返回 88.

（2） iHomeMode =1，先根据设定方向高速回原，然后找到原点下降沿后，返回低速找原点信号。原点接在 PLC 的 DI。初始回原方向通过 bHomeDirection 设置，回原速度通过 stPara.rHomeVel 设置，触发 stCtrl.iHome=1，完成返回 88.

（3） iHomeMode =2，调用 MC_Home，使用 6098 规定回原方式。回原速度通过 6099 设定。要求限位和原点信号都接在驱动器。

（4） iHomeMode =3，先找感应器作为原点，再找一圈内的 Z 相。即在=0 的模式上增加

402 协议内规定的 33 或者 34 回原方式。需要在 SDO 配置 6098/6099,。原点信号接在 PLC。初始回原方向通过 bHomeDirection 设置，回原速度通过 stPara.rHomeVel 设置，触发 stCtrl.iHome=1，完成返回 88.

功能 8：强制开抱闸
抱闸在使能成功后自动打开，使能关断的时候开启抱闸。

功能 9：轴暂停
stCtrl.iPauseAction=1，激活暂停；=88 暂停完成。=-1 关闭暂停轴继续运动。=-88 暂停关闭完成。（仅在绝对定位中生效）

功能 10：轴停止运动
停止运行的减速度通过 stPara.rStopDec 给定。stCtrl.iStop 给命令 1，激活轴停止。返回 88 停止成功。

stPar_AxisMain	ST_AxisPara		AxisMain的参数类数据
rJogVel	REAL	100	JOG速度
rAcc	REAL	10000	加速度
rDec	REAL	10000	减速度
rJerk	REAL	10000	减减速
rStopDec	REAL	10000	Stop的减速度
rHomeVel	REAL	360	回原速度
rMaxLimit	REAL	0	最大软限位
rMinLimit	REAL	0	最小软限位
rHomeOffset	REAL	0	回原点偏移
stCtrl_AxisMain	ST_AxisCtrl		AxisMain的控制类数据
bPower	BOOL	TRUE	使能接口
iStop	INT	88	停止

功能 11：轴信息传出
stCtrl.rActPos：轴当前位置传出
stCtrl.rActVel：轴当前速度传出
stCtrl.bError :轴产生报错，则为 ON
stCtrl.udErrorID :报错时的代码
stCtrl.bHomeFlag :回过原点后的标志位
stCtrl.iAxisState :轴状态机（参考 codesys 轴状态机即可）

若存在 402 轴数比较多，可用数组和 For 循环批量实例：

```
VAR_GLOBAL
    //================================轴控定义================================
    g_RealAxis    : ARRAY[0..cnAxisNum-1] OF POINTER TO AXIS_REF_ETC_DS402_CS ;
    g_stCtrl    :ARRAY[0..cnAxisNum-1]  OF   st_AxisCtrl;
    g_stPar     :ARRAY[0..cnAxisNum-1]  OF   ST_AxisPara;
END_VAR
//================================常量================================
VAR_GLOBAL CONSTANT
    cnAxisNum                       : INT := 32 ;              //轴数量
END_VAR
```

```
PROGRAM prgAxisCtrl
VAR
//轴名称指针数组
    AxisType            :ARRAY [0..cnAxisNum-1]OF POINTER TO AXIS_REF_ETC_DS402_CS;
//Tec 指针数组
    MC_TEC_Lable      : ARRAY[0..cnAxisNum-1] OF TEC_AxisModel ;
//循环遍历变量
    Counter:INT;
END_VAR
```

100 %

```


//轴接口控制
FOR Counter:=0 TO cnAxisNum-1 BY 1 DO
    MC_TEC_Lable[Counter](
        bHomeDirection:= ,
        bBreakManualOpen:= ,
        stPara:= g_stPar[Counter],
        iHomeMode:= 2,
        stCtrl:= g_stCtrl[Counter],
        AxisRef:= g_RealAxis[Counter]^,
        wStr=> ,
        bBreakLoosen=> );
END_FOR
Counter:=0;
```

011_单轴功能控制合集

待发

012_虚轴/实轴基础功能合集

待发

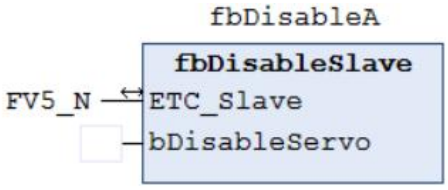
013_单轴单功能控制

fbDisableSlave 禁用启用从站

功能块说明：用于禁用总线下的从站设备

1、指令格式

图形表现：



ST 表现：

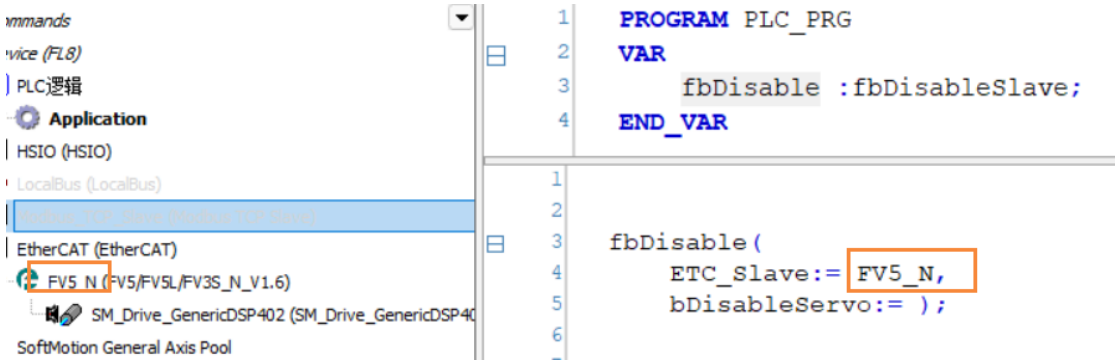
```
fbDisable(
    ETC_Slave:= FV5_N,
    bDisableServo:= );
```

2、相关变量

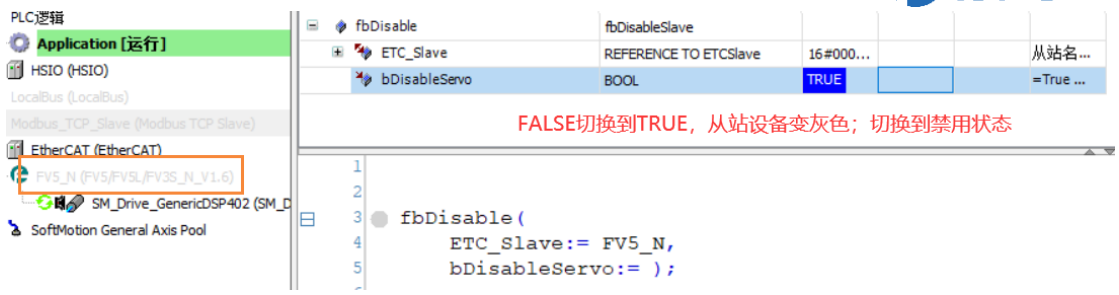
变量名	名称	数据类型	输入输出属性	描述
ETC_Slave	Ethercat 从站名	ETCSlave	输入输出变量	需要被禁用的 EtherCat 从站名称
bDisableServo	启用和禁用	BOOL	输入变量	设置变量从 FALSE 到 TRUE，则禁用从站；设置变量从 TRUE 到 FALSE，则启用从站

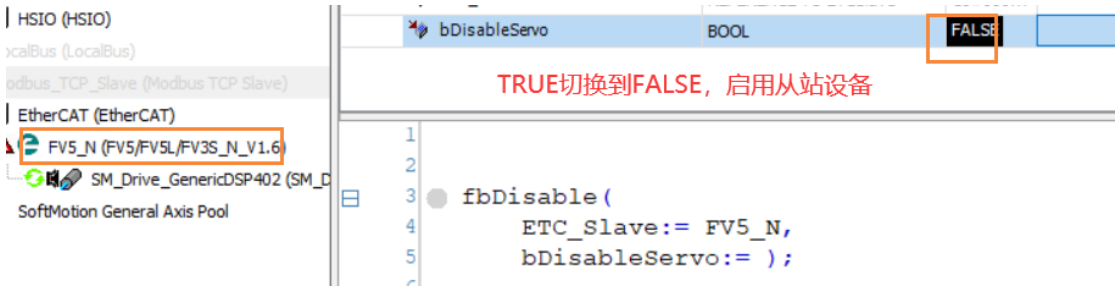
3、具体声明和调用

如下，ETC_Slave 引脚中填入的是需要程序动态禁用启用的从站设备名称；实际数据类型为 ETCSlave



对 bDisableServo 设置 True，则能看到设备树变成灰色，禁用成功。从 TRUE 设置成 FALSE，则设备树恢复，启用成功。



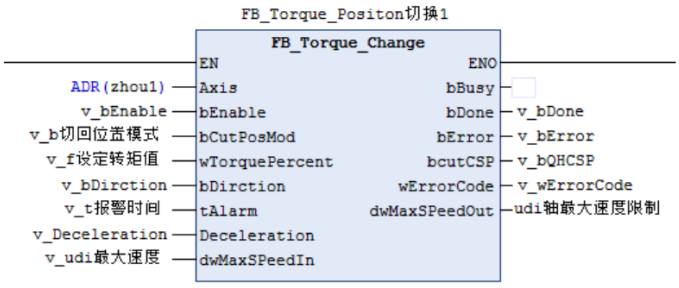


FB_Torque_Change 位置模式切换转矩模式功能块

可实现控制器从位置模式切换成转矩模式，并可以设置转矩大小。转矩达到经过一定时间后再切换回位置模式，可在动作过程中切换。

1、指令格式

图形表现：



ST 表现：

```
FB_Torque_Positon切换1(
    Axis:= ,
    bEnable:= ,
    bCutPosMod:= ,
    wTorquePercent:= ,
    bDirction:= ,
    tAlarm:= ,
    Deceleration:= ,
    dwMaxSPeedIn:= ,
    bBusy=> ,
    bDone=> ,
    bError=> ,
    bcutCSP=> ,
    wErrorCode=> ,
    dwMaxSPeedOut=> );
```

2、相关变量

（1）输入输出变量（Var In_Out）

变量名称	数据类型	默认值	有效范围	描述
Axis	POINTER TO Axis_REF_ETC_DS402_CS	/	/	指定轴的指针变量

（2）输入变量（VAR_Input）

变量名称	数据类型	默认值	有效范围	描述
bEnable	BOOL	FALSE	TRUE/FALSE	TRUE：切换转矩模式 FALSE：切换位置模式
bCutPosMod	BOOL	FALSE	TRUE/FALSE	切回位置模式按钮
wTorquePercent	REAL	0	正数/0	这里的单位需要与伺服的 6071 对应，单位：0.1%
bDirction	BOOL	FALSE	TRUE/FALSE	TRUE：反向扭矩 FALSE：正向扭矩
tAlarm	TIME	T#0S		当切换到扭矩模式后，经过设置的时间 bDone 会置 TRUE，但此时如果 bEnable 还是 TRUE 时，模式并不会切换回位置模式。
Deceleration	LREAL	0	正数/0	发生错误时的减速度
dwMaxSPeedIn	UDINT	0		最大速度限值，单位：转/秒

（3）输出变量（VAR_Output）

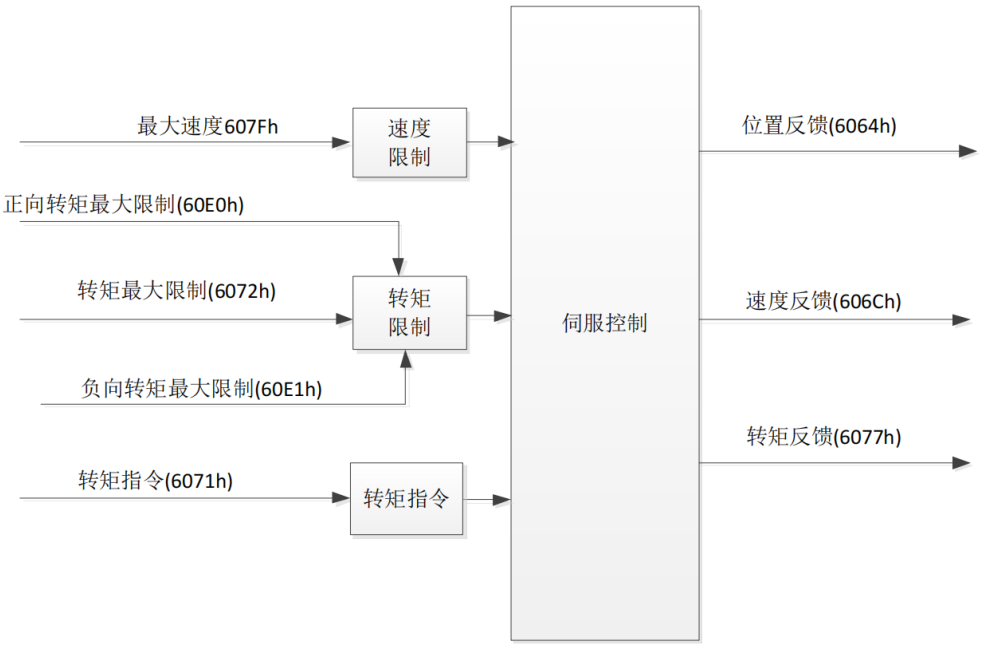
输出变量	数据类型	默认值	有效范围	描述
bBusy	BOOL	/	TRUE/FALSE	TRUE:将轴模式切换为扭矩模式
bDone	BOOL	/	TRUE/FALSE	TRUE:切换扭矩时间到达设置时间
bError	BOOL	/	TRUE/FALSE	TRUE:功能块产生异常
bcutCSP	BOOL	/	TRUE/FALSE	TRUE：轴模式从扭矩模式切换回位置模式 FALSE：bEnable 置 TRUE 时
wErrorCode	WORD	/	正数/0	功能块产生异常时，输出错误代码， 0：无报错 1：轴状态错误 2：自制功能块参数错误 3：模式功能块报错 4：转矩功能块报错 20：切换转矩模式后超过规定时间未达到设定扭矩报警
dwMaxSPeedOut	UDINT	838860800	0~4294967295	指令单位/秒，需要将该输出值绑定对象字典 607F

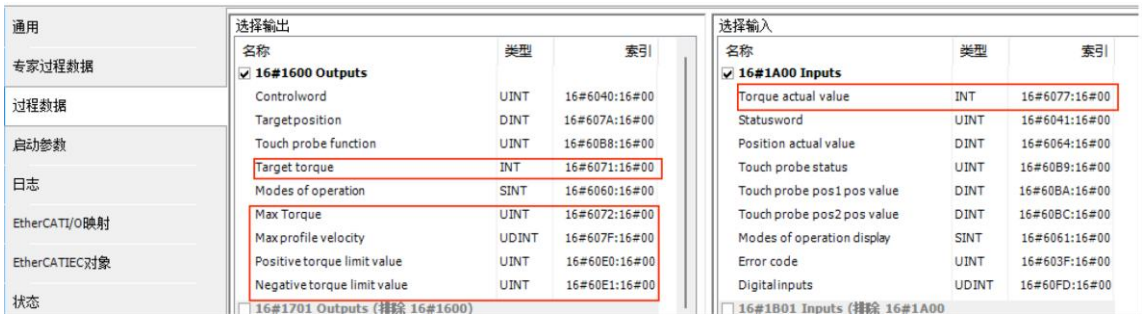
3、具体声明和调用

```
FB_Torque_Positon切换1:FB_Torque_Positon切换;

v_bEnable               :BOOL;
v_b切回位置模式          :BOOL;
v_f设定转矩值            :REAL:=200;
v_bDirction             :BOOL;
v_t报警时间              :TIME:=T#20S;
v_Deceleration          :LREAL:=10000;
v_udi最大速度            :UDINT:=1;

udi轴最大速度限制        :UDINT;
v_bDone                 :BOOL;
v_bQHCSP                :BOOL;
v_bError                :BOOL;
v_wErrorCode            :WORD;

simodes                 :SINT;
```

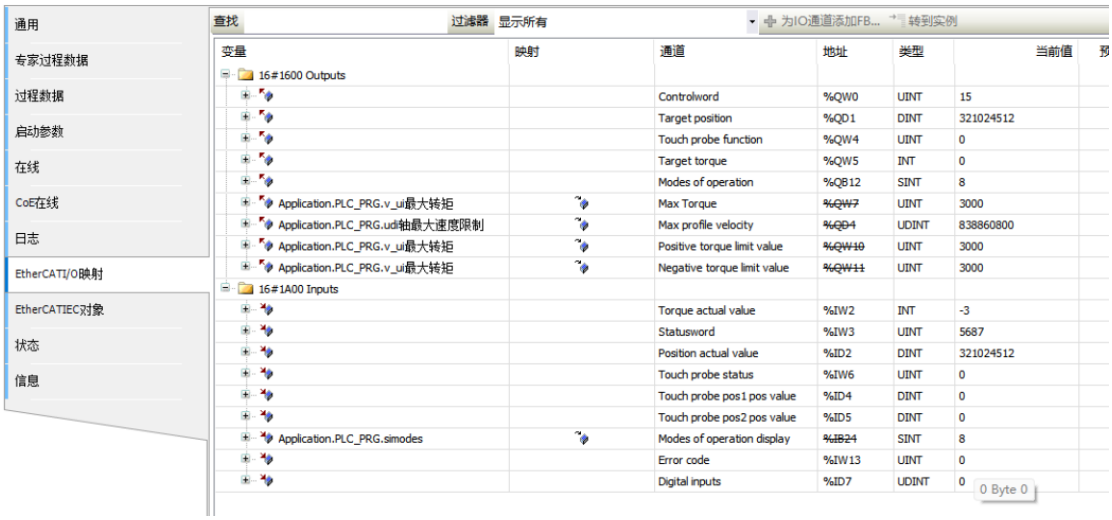
测试前准备工作:

1、添加对象字典 6072（转矩最大限制）、60E0（正向转矩最大限制）、60E1（负向转矩最大限制）、607F（最大速度）、6071（转矩指令）、6077（转矩反馈）。

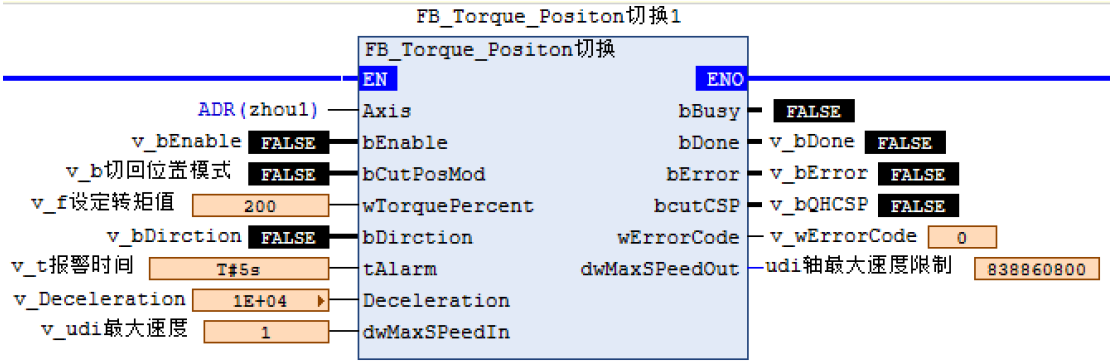




2、并在 6072（转矩最大限制）、60E0（正向转矩最大限制）、60E1 负向转矩最大限制）上映射最大转矩的值，默认 3000,单位：0.1%。然后将功能块输出引脚 dwMaxSPeedOut 映射到 607F（最大速度）上

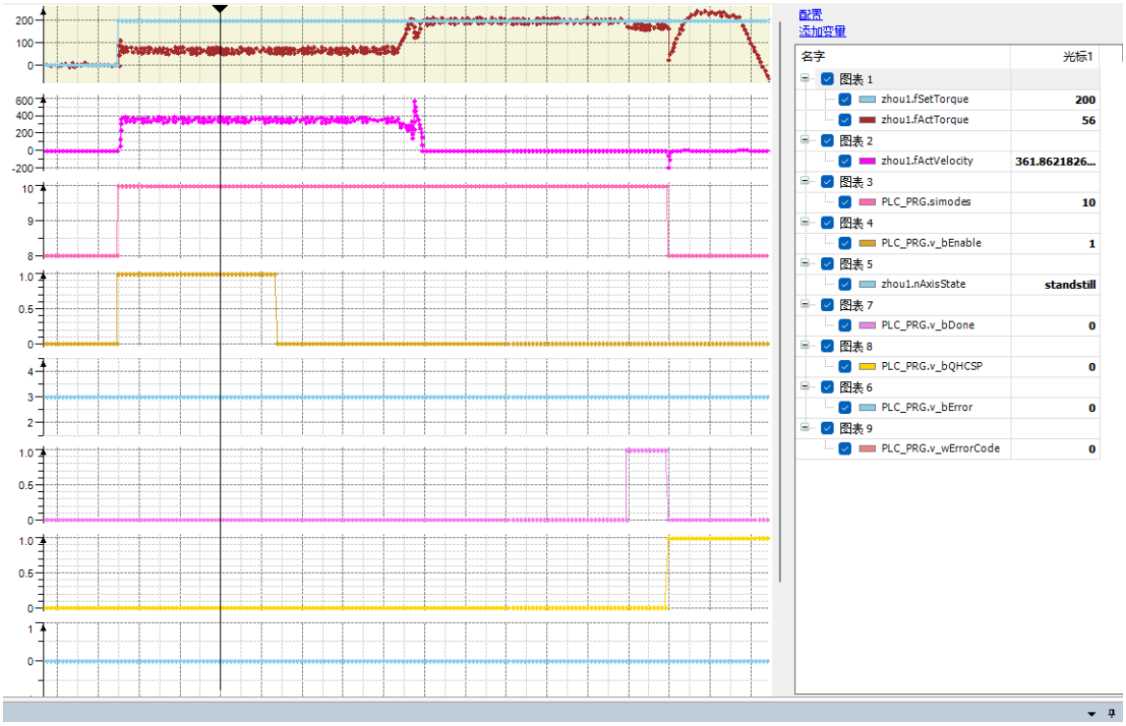


报警时间：5S；设定转矩值：200；最大速度：1 转/秒

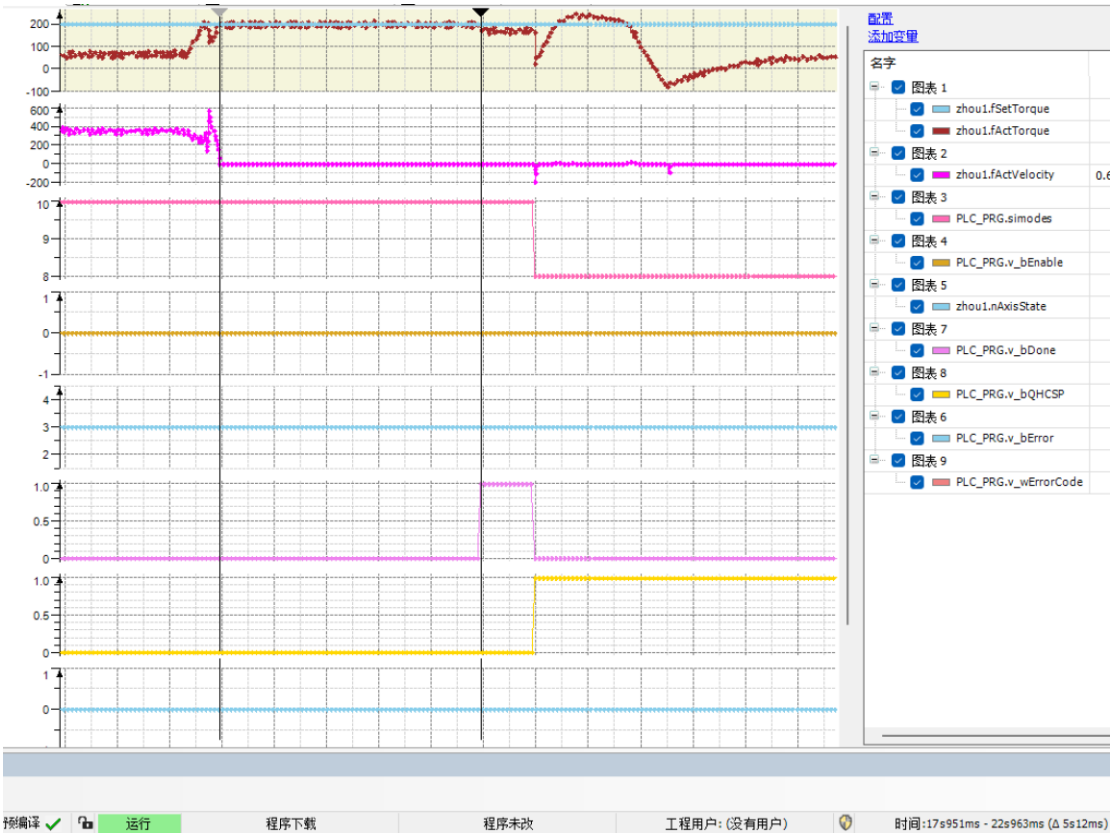


演示案例说明：正常使用演示

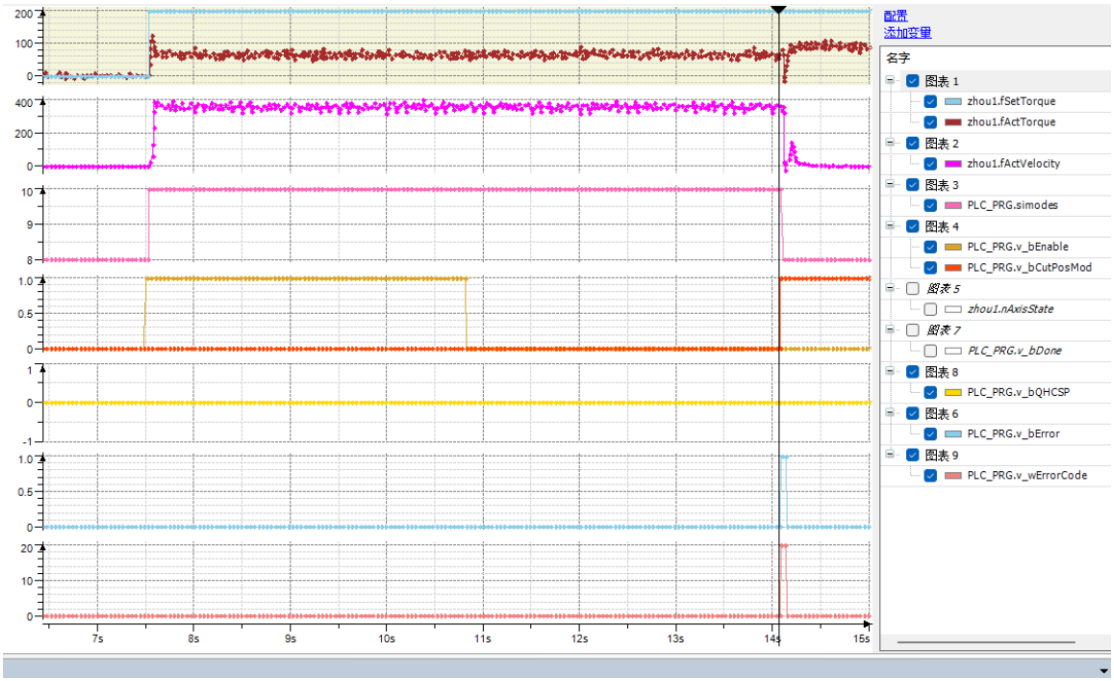
1、使能功能块，轴模式切换到转矩模式，轴速度为 360（旋转轴，行程 360）1 转/秒；



2、关闭使能功能，然后给轴端施加一个力使轴停止，经过 5S 后，Done 信号置 TRUE，之后轴模式从转矩模式切换回位置模式。随后 bcutCSP 信号置 TRUE，Done 信号置 FALSE。



演示案例说明：在切到扭矩模式后也可通过 bCutPosMod 引脚手动切回位置模式。

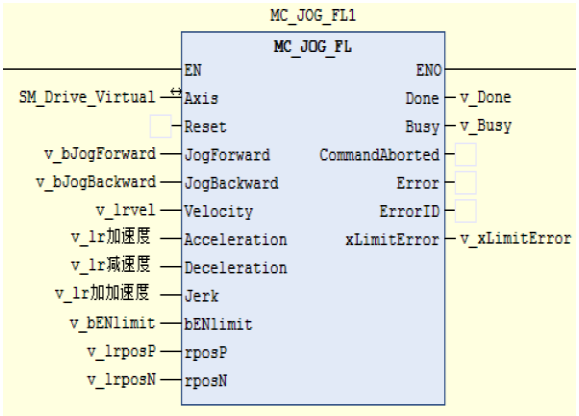


MC_JOG_FL 带正反限位的点动

功能块说明：在原本的 JOG 功能上加上了正反限位保护。现场轴点动需要做正负限位保护时可以使用该功能块

1、指令格式

图形表现



ST 表现

```
MC_JOG_FL1(
    Axis:=SM_Drive_Virtual ,
    Reset:= ,
    JogForward:= ,
    JogBackward:= ,
    Velocity:= ,
    Acceleration:= ,
    Deceleration:= ,
    Jerk:= ,
    bENlimit:= ,
    rposP:= ,
    rposN:= ,
    Done=> ,
    Busy=> ,
    CommandAborted=> ,
    Error=> ,
    ErrorID=> ,
    xLimitError=> );
```

2、相关变量

（1）输入输出变量（Var In_Out）

变量名称	数据类型	默认值	有效范围	描述
Axis	AXIS_REF_ET	/	/	指定轴，即 AXIS_REF_SM3 的一个实例

	C_DS402_CS			

（2）输入变量（VAR_Input）

变量名称	数据类型	默认值	有效范围	描述
Reset	BOOL	FALSE	TRUE/FALSE	复位功能块
JogForward	BOOL	0	TRUE/FALSE	正向点动
JogBackward	BOOL	0	TRUE/FALSE	反向点动
Velocity	LREAL	0	正数/0	指定目标速度，单位为【指令单位/s】
Acceleration	LREAL	0	正数/0	加速度，单位为【指令单位/s^2】
Deceleration	LREAL	0	正数/0	减速度，单位为【指令单位/s^2】
Jerk	LREAL	0	正数/0	加加速度，单位为【指令单位/s^3】
bENlimit	BOOL	FALSE	TRUE/FALSE	启动限位功能
rposP	LREAL	100	/	点动正向限位位置
rposN	LREAL	0	/	点动反向限位位置

（3）输出变量（VAR_Output）

输出变量	数据类型	默认值	有效范围	描述
Done	BOOL	/	TRUE/FALSE	TRUE:点动完成
Busy	BOOL	/	TRUE/FALSE	TRUE:功能块执行中
CommandAborted	BOOL	/	TRUE/FALSE	TRUE:功能块被中止
Error	BOOL	/	TRUE/FALSE	TRUE:功能块产生异常
ErrorID	SMC_ERROR	/		功能块产生异常时，输出错误代码
xLimitError	BOOL	/	TRUE/FALSE	TRUE:限位报警

3、使用样例

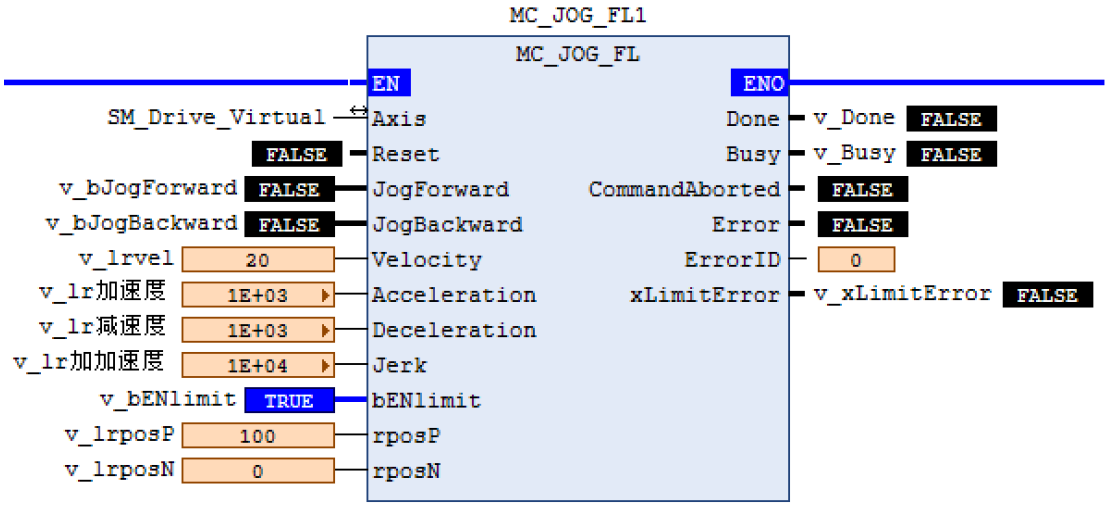
（1）声明与调用

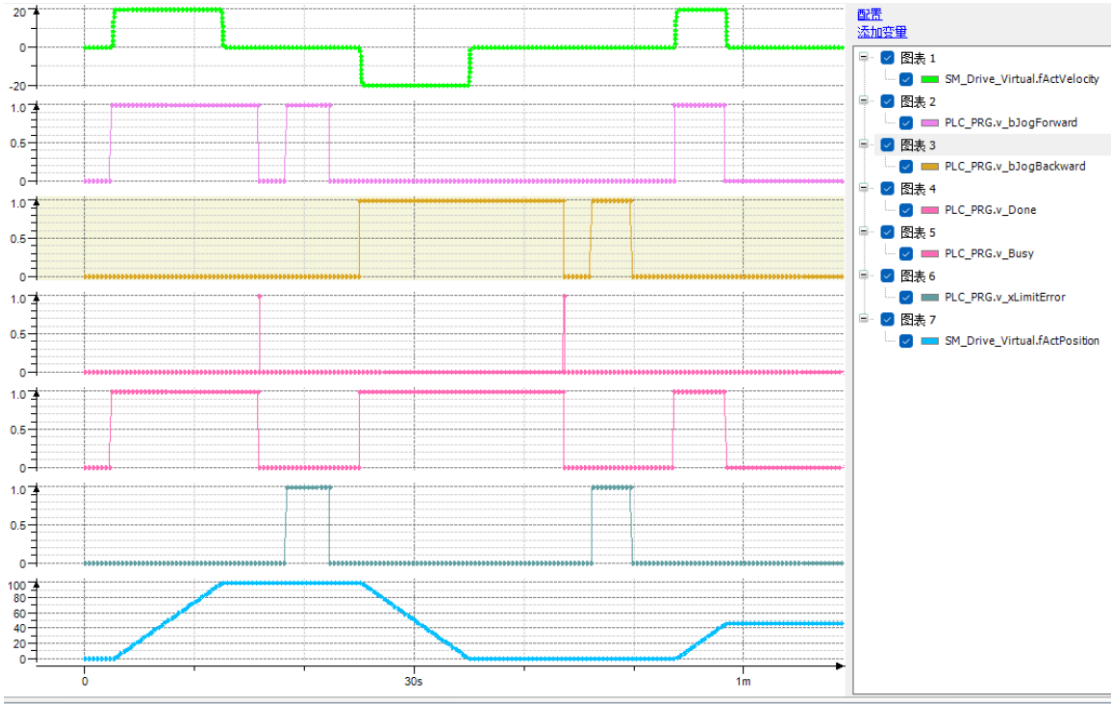
```
MC_JOG_FL1:MC_JOG_FL;

v_bJogForward       :BOOL;
v_bJogBackward      :BOOL;
v_lrvel             :LREAL:=10;
v_lr加速度          :LREAL:=1000;
v_lr减速度          :LREAL:=1000;
v_lr加加速度        :LREAL:=10000;
v_bENlimit          :BOOL;
v_lrposP            :LREAL:=100; //正向限位位置
v_lrposN            :LREAL:=0; //反向限位位置

v_Done              :BOOL;
v_Busy              :BOOL;
v_xLimitError       :BOOL;
```

功能详细演示部分

演示案例说明:

测试全程都启用点动限位功能，正向限位位置=100，反向限位位置=0
第一步：按下正向点动，直到目标位置为 100 后，轴不再动作，之后结束正向点动；
第二步：再次按下正向点动，轴并不会动作并输出限位报警信号；
第三步：按下反向点动，直到目标位置为 0 后，轴不再动作，之后结束反向点动；
第四步：再次按下反向点动，轴并不会动作并输出限位报警信号；



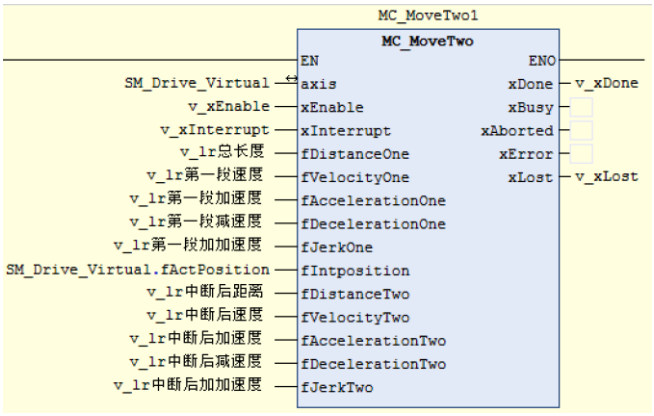


MC_MoveTwo 中断定长

功能块说明：中断定长功能块，在一段固定距离的运动中可能会触发第二段定位动作。常在色标偏移中会使用到，即触发定位后，在运动中遇到色标后再触发第二段定位进行色标偏移。

1、指令格式

图形表现：



ST 表现：

```
MC_MoveTwo1(
    axis:=SM_Drive_Virtual
    xEnable:= ,
    xInterrupt:= ,
    fDistanceOne:= ,
    fVelocityOne:= ,
    fAccelerationOne:= ,
    fDecelerationOne:= ,
    fJerkOne:= ,
    fIntposition:= ,
    fDistanceTwo:= ,
    fVelocityTwo:= ,
    fAccelerationTwo:= ,
    fDecelerationTwo:= ,
    fJerkTwo:= ,
    xDone=> ,
    xBusy=> ,
    xAborted=> ,
    xError=> ,
    xLost=> );
```

2、相关变量

（1）输入输出变量（Var In_Out）

变量名称	数据类型	默认值	有效范围	描述
Axis	AXIS_REF_SM3	/	/	指定轴，即 AXIS_REF_SM3 的一个实例

（2）输入变量（VAR_Input）

变量名称	数据类型	默认值	有效范围	描述
xEnable	BOOL	FALSE	TRUE/FALSE	开始启动总长度段
xInterrupt	BOOL	FALSE	TRUE/FALSE	启动第二段
fDistanceOne	LREAL	100	正数	总距离
fVelocityOne	LREAL	10	正数/0	开始速度
fAccelerationOne	LREAL	1000	正数/0	开始加速度
fDecelerationOne	LREAL	1000	正数/0	开始减速度
fJerkOne	LREAL	10000	正数/0	开始加加速度
fIntposition	LREAL	0	/	启动第二段时轴的位置
fDistanceTwo	LREAL	10	正数	中断后距离
fVelocityTwo	LREAL	5	正数/0	中断速度
fAccelerationTwo	LREAL	1000	正数/0	中断后加速度
fDecelerationTwo	LREAL	1000	正数/0	中断后减速度
fJerkTwo	LREAL	10000	正数/0	中断后加加速度

（3）输出变量（VAR_Output）

输出变量	数据类型	默认值	有效范围	描述
xDone	BOOL	/	TRUE/FALSE	TRUE:中断完成或未触发中断
xBusy	BOOL	/	TRUE/FALSE	TRUE:功能块执行中

xAborted	BOOL	/	TRUE/FALSE	TRUE:功能块被中止
xError	BOOL	/	TRUE/FALSE	TRUE:功能块产生异常
xLost	BOOL	/	TRUE/FALSE	TRUE:全程无中断

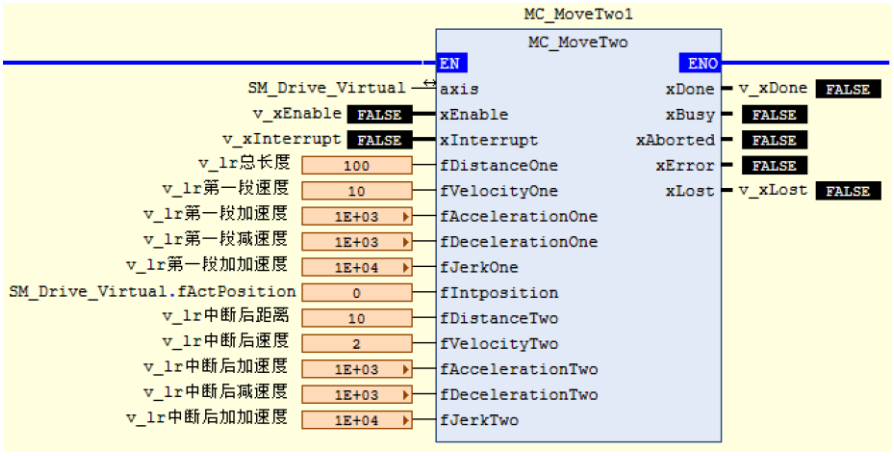
3、使用样例

（1）声明与调用

```
MC_MoveTwo1          :MC_MoveTwo;

v_xEnable            :BOOL;
v_xInterrupt         :BOOL;
v_lr总长度           :LREAL:=100;
v_lr第一段速度       :LREAL:=10;
v_lr第一段加速度     :LREAL:=1000;
v_lr第一段减速度     :LREAL:=1000;
v_lr第一段加加速度   :LREAL:=10000;
v_fIntposition       :LREAL;
v_lr中断后距离       :LREAL:=10;
v_lr中断后速度       :LREAL:=2;
v_lr中断后加速度     :LREAL:=1000;
v_lr中断后减速度     :LREAL:=1000;
v_lr中断后加加速度   :LREAL:=10000;

v_xDone              :BOOL;
v_xLost              :BOOL;
```

功能详细演示部分

总长度：100，第一段速度：10，第二段长度：10

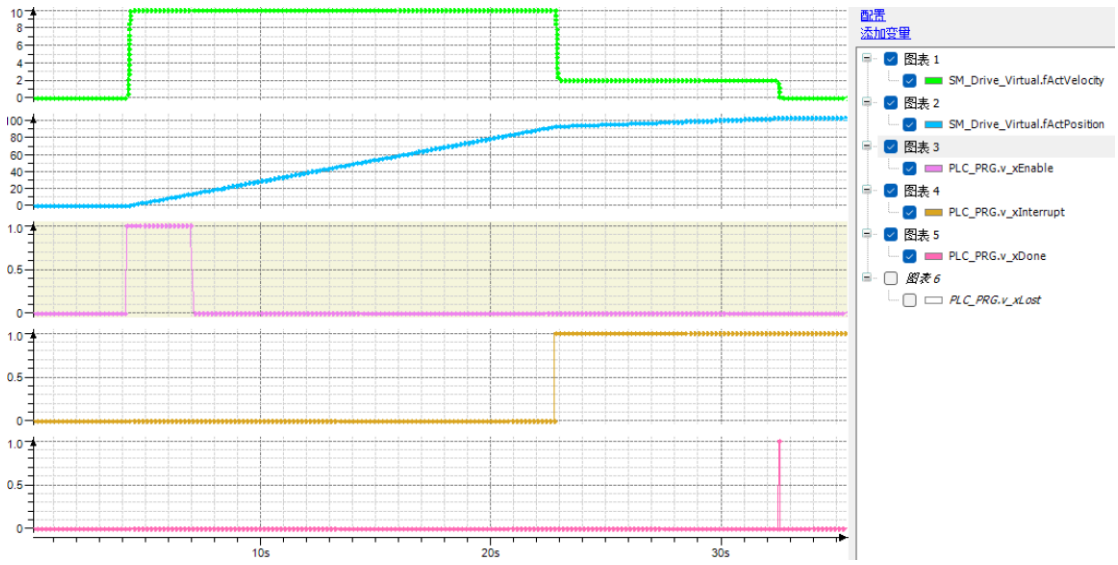


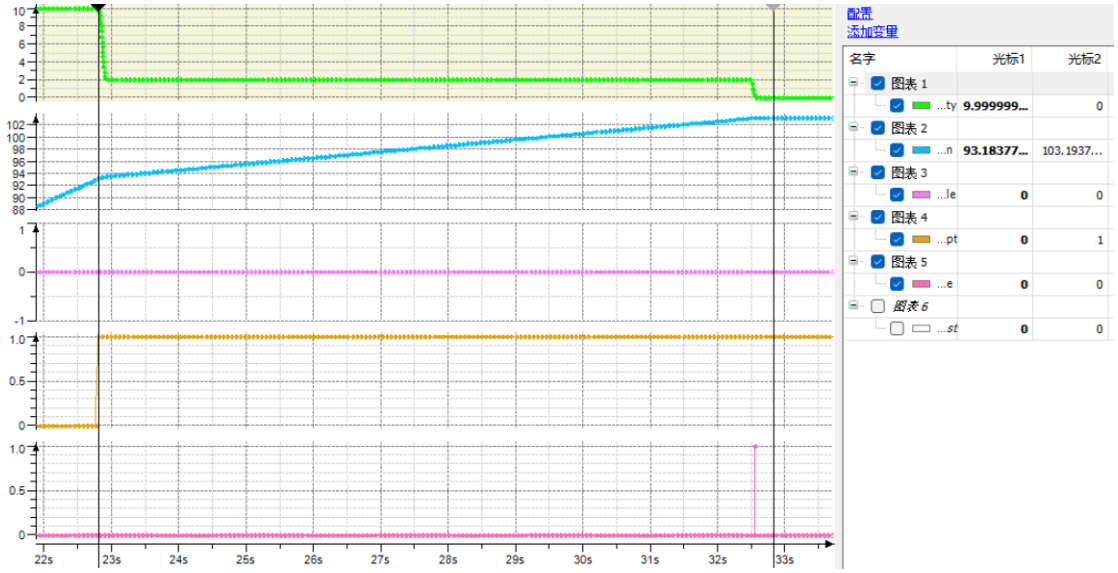
演示案例说明：

第一步：触发中断定长功能块；

第二步：触发第二段位置，触发时轴位置为 93.18；

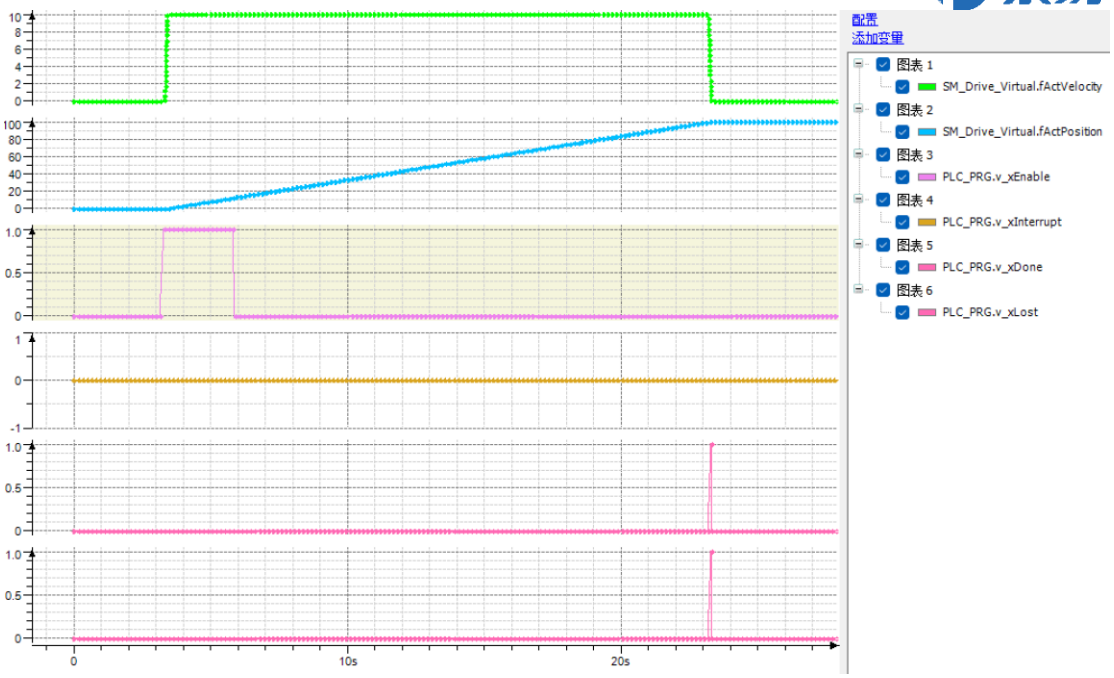
第三步：等待第二段定位完成，完成后轴位置为 103.19；





演示案例说明:

若动作全程未触发第二段定位，则在定位结束时会输出 xLost 信号。

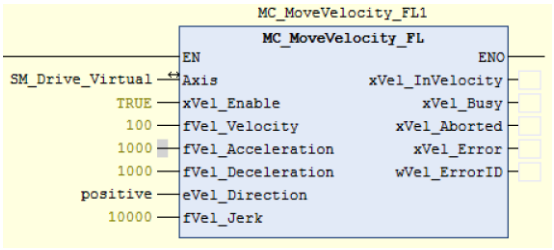


MC_MoveVelocity_FL 匀速运动变速立刻生效功能

功能块说明：可实现匀速功能块速度改变后自动生效。

1、指令格式

图形表现：



ST 表现：

```
MC_MoveVelocity_FL1(
    Axis:=SM_Drive_Virtual ,
    xVel_Enable:=TRUE ,
    fVel_Velocity:=100 ,
    fVel_Acceleration:=1000 ,
    fVel_Deceleration:=1000 ,
    eVel_Direction:=positive ,
    fVel_Jerk:=10000 ,
    xVel_InVelocity=> ,
    xVel_Busy=> ,
    xVel_Aborted=> ,
    xVel_Error=> ,
    wVel_ErrorID=> );
```

2、相关变量

（1）输入输出变量（Var In_Out）

变量名称	数据类型	默认值	有效范围	描述
Axis	AXIS_REF_ET C_DS402_CS	/	/	指定轴，即 AXIS_REF_SM3 的一个实例

（2）输入变量（VAR_Input）

变量名称	数据类型	默认值	有效范围	描述
xVel_Enable	BOOL	FALSE	TRUE/FALSE	使能功能块有效
fVel_Velocity	LREAL	0	正数	指定目标速度
fVel_Acceleration	LREAL	0	正数/0	加速度
fVel_Deceleration	LREAL	0	正数/0	减速度
eVel_Direction	MC_Direction	Positive	fastest, current, Positive, shortest, negative	运动方向选择，详细信息参考 MC_Direction（ENUM）
fVel_Jerk	LREAL	0	正数/0	加加速度

（3）输出变量（VAR_Output）

输出变量	数据类型	默认值	有效范围	描述
xVel_InVelocity	BOOL	/	TRUE/FALSE	TRUE:目标速度达到设定值
xVel_Busy	BOOL	/	TRUE/FALSE	TRUE:功能块执行中
xVel_Aborted	BOOL	/	TRUE/FALSE	TRUE:功能块被中止
xVel_Error	BOOL	/	TRUE/FALSE	TRUE:功能块产生异常
wVel_ErrorID	SMC_ERROR	/		功能块产生异常时，输出错误代码

3、使用样例

（1）声明与调用

```
MC_MoveVelocity_FL1:MC_MoveVelocity_FL;

v_bEnable           :BOOL;
v_lrvel             :LREAL:=100;
v_lr加速度          :LREAL:=1000;
v_lr减速度          :LREAL:=1000;
v_lr加加速度        :LREAL:=10000;
v_e运动方向         :SM3_Basic.MC_DIRECTION :=positive;
v_bInVelocity       :BOOL;
v_bBusy             :BOOL;
```

功能详细演示部分

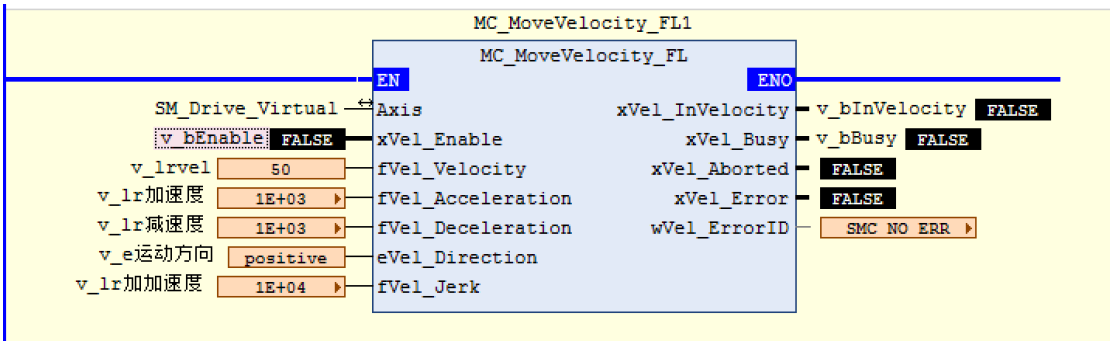
演示案例说明：

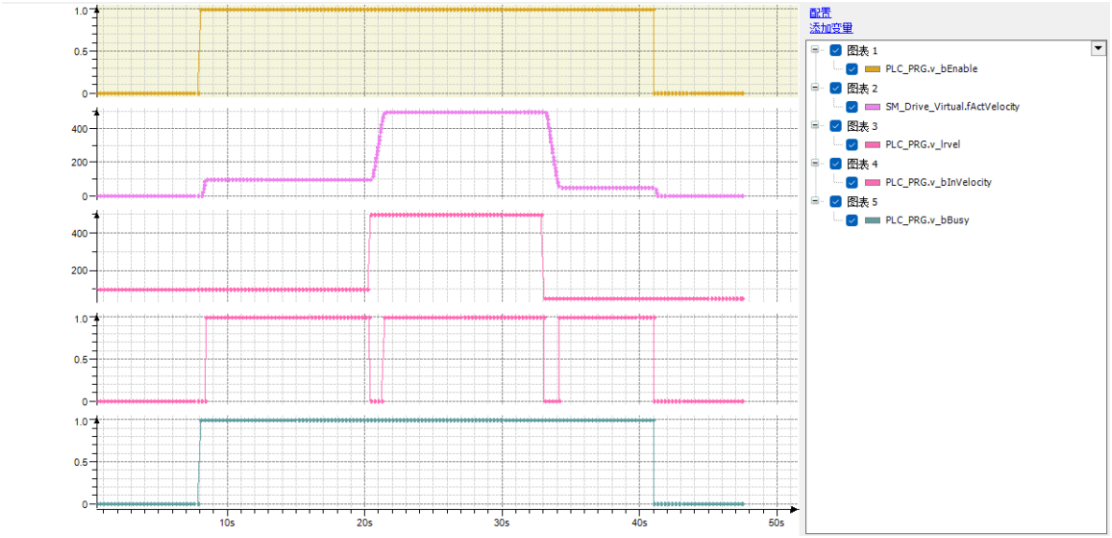
第一步：使能功能块，目标速度为 100；

第二步：更改目标速度为 500；

第三步：更改目标速度为 50；

第四步：关闭使能功能块；


MC_MoveVelocity_FL1
MC_MoveVelocity_FL
EN
ENO
SM_Drive_Virtual
Axis
v_bEnable
FALSE
xVel_Enable
v_lrvel
50
fVel_Velocity
v_lr加速度
1E+03
fVel_Acceleration
v_lr减速度
1E+03
fVel_Deceleration
v_e运动方向
positive
eVel_Direction
v_lr加加速度
1E+04
fVel_Jerk
xVel_InVelocity
v_bInVelocity
FALSE
xVel_Busy
v_bBusy
FALSE
xVel_Aborted
FALSE
xVel_Error
FALSE
wVel_ErrorID
SMC_NO_ERR


配置
添加变量
图表 1
PLC_PRG.v_bEnable
图表 2
SM_Drive_Virtual.fActVelocity
图表 3
PLC_PRG.v_lrvel
图表 4
PLC_PRG.v_bInVelocity
图表 5
PLC_PRG.v_bBusy
10s
20s
30s
40s
50s

02 轴齿轮凸轮控制

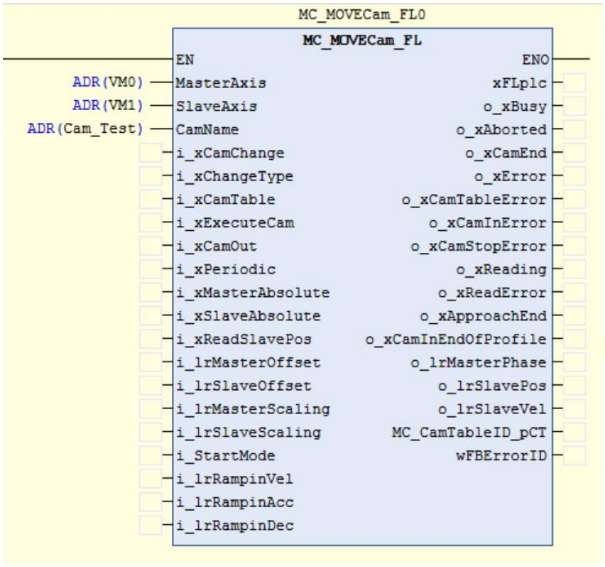
021 凸轮/齿轮功能合集

MC_MoveCam_FL 凸轮功能合集

功能块说明:
通过将 MC_CamTableSelect，MC_CamIn，MC_CamOut，MC_Stop，SMC_GetCamSlaveSetPosition 等功能块组合起来，便于程序中使用，此外添加了自动切换凸轮等便捷功能。可使用在运动控制中任意需要电子凸轮的场合。

1、指令格式

图形表现



ST 表现

```
MC_MOVECam_FL0(
    MasterAxis:= ADR(VM0),
    SlaveAxis:= ADR(VM1),
    CamName:= ADR(CAM_R^),
    i_xCamChange:= TRUE,
    i_xChangeType:= ,
    i_xCamTable:= ,
    i_xExecuteCam:=x耦合凸轮 ,
    i_xCamOut:= x脱离凸轮,
    i_xPeriodic:= b_凸轮周期模式,
    i_xReadSlavePos:= ,
    i_xMasterAbsolute:= ,
    i_xSlaveAbsolute:= ,
    i_lrMasterOffset:= ,
    i_lrSlaveOffset:= ,
    i_lrMasterScaling:= ,
    i_lrSlaveScaling:= ,
    i_StartMode:= ,
    i_lrRampinVel:= ,
    i_lrRampinAcc:= ,
    i_lrRampinDec:= ,
    xFLplc=> ,  o_xBusy=> ,     o_xAborted=> ,
    o_xCamEnd=> ,   o_xError=> ,
    o_xCamTableError=> ,    o_xCamInError=> ,
    o_xCamStopError=> ,     o_xReading=> ,
    o_xReadError=> ,    o_xApproachEnd=> ,
    o_xCamInEndOfProfile=> ,    o_lrMasterPhase=> ,
    o_lrSlavePos=> ,    o_lrSlaveVel=> ,
    MC_CamTableID_pCT=> ,   wFBErrorID=> );
```

2、相关变量

（1）输入变量（VAR_Input）

变量名称	数据类型	默认值	有效范围	描述
MasterAxis	POINTER TO AXIS_REF_SM3	/	/	主轴
SlaveAxis	POINTER TO AXIS_REF_SM3	/	/	从轴
CamName	POINTER TO MC_CAM_REF3	/		凸轮表
i_xCamChange	BOOL	false	/	ture:允许切换凸轮，false：不允许
i_xChangeType	BOOL	false	/	ture:立即切换凸轮，false：周期末点切换凸轮

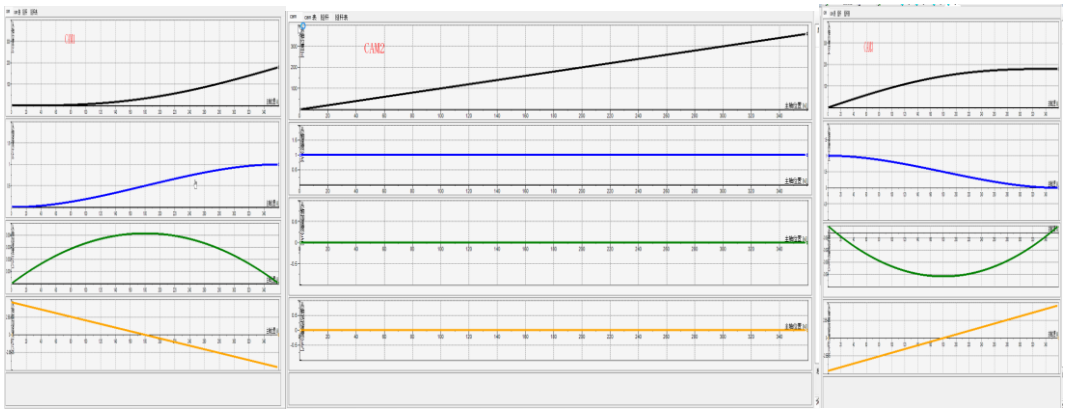
i_xCamTable	BOOL	/	/	更新凸轮表，耦合中如果末点修改时需要触发
i_xExecuteCam	BOOL	/	/	凸轮拟合
i_xCamOut	BOOL	/	/	凸轮脱离
i_xPeriodic	BOOL	false	/	周期模式 < ture：周期>-
i_xMasterAbsolute	BOOL	false	/	主坐标系-< ture ： 绝对坐标>
i_xSlaveAbsolute	BOOL	false	/	从坐标系-< ture ： 绝对坐标>
i_xReadSlavePos	BOOL	/		读取从轴耦合需要的位置
i_lrMasterOffset	LREAL	0	/	主轴偏移
i_lrSlaveOffset	LREAL	0		从轴偏移
i_lrMasterScaling	LREAL	1		主轴缩放
i_lrSlaveScaling	LREAL	1		从轴缩放
i_lrRampinVel	LREAL	100		Rampin 耦合时的追赶速度
i_lrRampinAcc	LREAL	1000		rampin 加速度
i_lrRampinDec	LREAL	1000		rampin 减速度
i_StartMode	MC_StartMode	Absolute		耦合方式: Absolute, Relative, Ramp_in , Ramp_in_pos, Ramp_in_neg

（2）输出变量（VAR_Output）

输出变量	数据类型	默认值	有效范围	描述
o_xBusy	BOOL	/	/	凸轮运行
o_xAborted	BOOL	/	/	凸轮被中断
o_xCamFinish	BOOL	/	/	凸轮结束
o_xError	BOOL	/	/	凸轮故障
o_xCamTableError	BOOL	/	/	凸轮描点故障（MC_CamTableSelect）
o_xCamInError	BOOL	/	/	凸轮拟合故障(MC_CamIn 故障)
o_xCamStopError	BOOL	/	/	凸轮停止故障（MC_Stop）
o_xReading	BOOL	/	/	从轴读取进行
o_xReadError	BOOL	/	/	从轴读取失败
o_xApproachEnd	BOOL	/	/	接近曲线末点，在末点前 1-2 周期
o_xCamInEndOfProfile	BOOL	/	/	凸轮自带的周期终点（CamIn 的引脚）
o_lrMasterPhase	LREAL	/	/	凸轮主轴周期位置
o_lrSlavePos	LREAL	/	/	耦合前位置，主轴相对模式下始终为 0,涉及主轴相对模式，在外部判断这个位置;耦合后表示在凸轮表里的位置。
o_lrSlaveVel	LREAL	/	/	运行中从轴速度
MC_CamTableID_pCT	POINTER TO BYTE	/	/	运行中的凸轮 ID
wFBErrorID	SMC_ERROR	/	/	故障代码
O_xFLplc	BOOL	/	/	繁易 PLC 标志

3、使用样例

（1）声明与调用

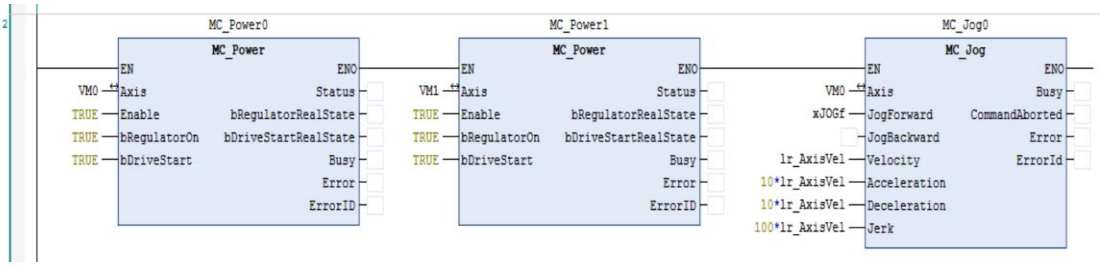


CAM1　　CAM2　　CAM3

声明部分：

```
    MC_MOVECam_FL0: MC_MOVECam_FL;
    xJOGf: BOOL;
    i_凸轮表选择      :INT;
    b_凸轮周期模式    : BOOL;
     xChange      :BOOL;
     xStopping   : BOOL;
     x耦合凸轮   :BOOL;
     x脱离凸轮   :BOOL;
     CAM_R    : POINTER TO MC_CAM_REF ;
END_VAR
```

基础功能块（使能和 JOG）：



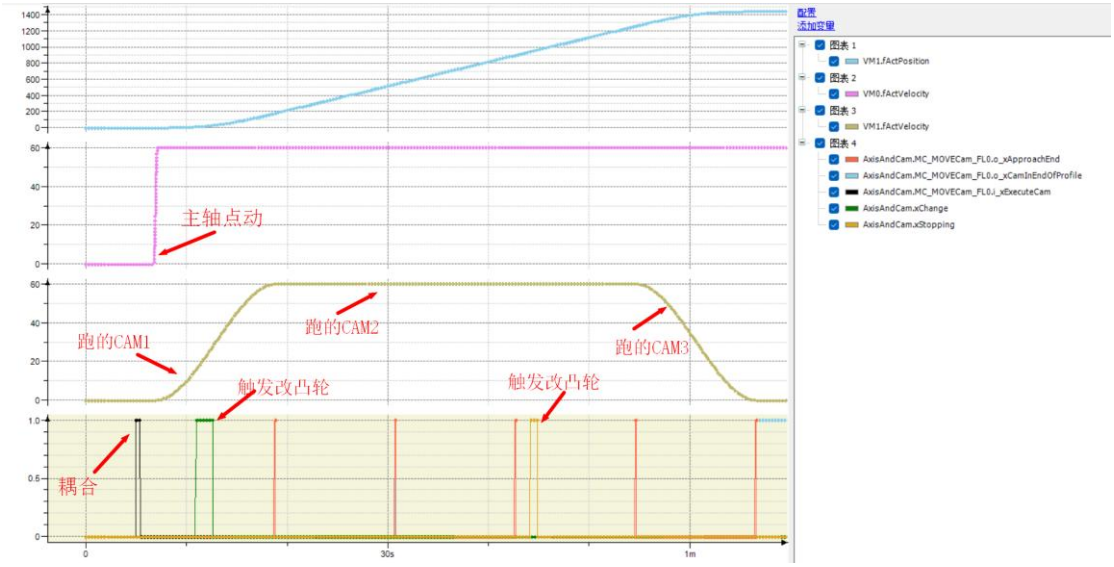
选表和模式设置：

AxisAndCam.ACT_切换凸轮

```
CASE   i_凸轮表选择 OF
    0:
        CAM_R :=ADR( Cam_1);
        b_凸轮周期模式 :=FALSE;
        IF xChange AND MC_MOVECam_FL0.o_xBusy THEN
            i_凸轮表选择 :=10;
        END_IF
    10:
        CAM_R :=ADR( Cam_2);
        b_凸轮周期模式 :=TRUE;
        IF xStopping THEN
            i_凸轮表选择 :=20;
        END_IF

    20:
        CAM_R :=ADR( Cam_3);
        b_凸轮周期模式 :=FALSE;
END_CASE
```

```
PROGRAM AxisAndCam
VAR
    FB_Rotary_Cut_Cam0 :FB_Rotary_Cut_Cam;
    FB_Flying_Saw_Cam0 :FB_Flying_Saw_Cam;

ACT_切换凸轮();

MC_MOVECam_FL0(
    MasterAxis:= ADR(VM0),  SlaveAxis:= ADR(VM1),   CamName:= ADR(CAM_R^),
    i_xCamChange:= TRUE,    i_xChangeType:= ,   i_xCamTable:= ,
    i_xExecuteCam:=x耦合凸轮 ,  i_xCamOut:= X脱离凸轮,  i_xPeriodic:= b_凸轮周期模式,
    i_xMasterAbsolute:= ,   i_xSlaveAbsolute:= ,    i_xReadSlavePos:= ,
    i_lrMasterOffset:= ,    i_lrSlaveOffset:= ,     i_lrMasterScaling:= ,   i_lrSlaveScaling:= ,
    i_StartMode:= ,     i_lrRampinVel:= ,   i_lrRampinAcc:= ,   i_lrRampinDec:= ,
    xFLplc=> ,  o_xBusy=> ,     o_xAborted=> ,  o_xCamEnd=> ,   o_xError=> ,
    o_xCamTableError=> ,    o_xCamInError=> ,   o_xCamStopError=> ,     o_xReading=> ,
    o_xReadError=> ,    o_xApproachEnd=> ,  o_xCamInEndOfProfile=> ,    o_lrMasterPhase=> ,
    o_lrSlavePos=> ,    o_lrSlaveVel=> ,    MC_CamTableID_pCT=> ,   wFBErrorID=> );
```

（2）功能演示部分



如上演示了通过改变凸轮表引脚，自动修改凸轮的操作。

1：首先耦合凸轮，默认先跑 CAM1，耦合后点动主轴。

2：当开始跑 CAM1 时，触发修改凸轮，将 CAM2 加入缓存，当 CAM1 接近终点时自动切换 CAM2 运行。

3：跑几段 CAM2 后，切换凸轮，将 CAM3 加入缓存，在 CAM2 接近终点时，自动切换到 CAM3 运行。

注意：演示的时候，选的是末点切换，如果选择立即切换，那么在凸轮表进来后会立即切换。

如果不需要切换凸轮，就直接把凸轮表挂到 CamName 引脚上，

例如：　CamName:= ADR(CAM_3)。

MC_MoveGear_FL 齿轮功能合集

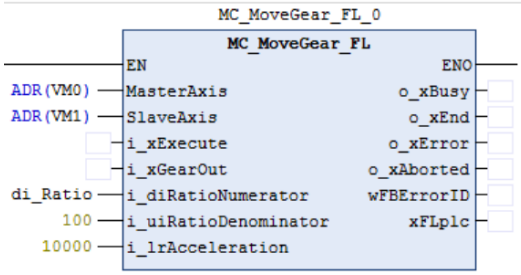
功能块说明：

将 MC_GearIn，MC_GearOut，MC_Stop 几个功能块组合起来，方便使用。可用于任意需要

用到齿轮耦合的场景

1、指令格式

图形表现



ST 表现

```
MC_MoveGear_FL_0(
    MasterAxis:= ADR(VM0),
    SlaveAxis:= ADR(VM1),
    i_xExecute:= ,
    i_xGearOut:= ,
    i_diRatioNumerator:= di_Ratio,
    i_uiRatioDenominator:= 100,
    i_lrAcceleration:= 1000,
    o_xBusy=> ,      o_xEnd=> ,
    o_xError=> ,     o_xAborted=> ,
    wFBErrorID=> ,   xFLplc=> );
```

2、功能块变量说明

（1）输入输出变量（Var In_Out）

变量名称	数据类型	默认值	有效范围	描述
MasterAxis	POINTER TO AXIS_REF_SM3	/	/	主轴
SlaveAxis	POINTER TO AXIS_REF_SM3	/	/	从轴

（2）输入变量（VAR_Input）

变量名称	数据类型	默认值	有效范围	描述
i_xExecute	BOOL	/	/	齿轮拟合
i_xGearOut	BOOL	/	/	齿轮脱离
i_diRatioNumerator	DINT	3600	/	齿轮分子
i_uiRatioDenominator	UDINT	3600	/	齿轮分母
i_lrAcceleration	LREAL	3000	/	追速度时的加速度值

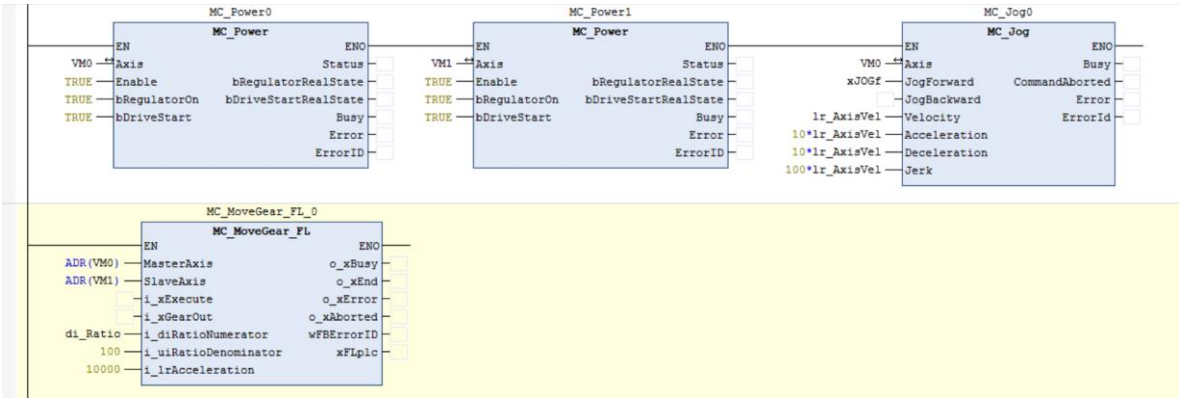
（3）输出变量（VAR_Output）

输出变量	数据类型	默认值	有效范围	描述
o_xBusy	BOOL	/	/	齿轮进行
o_xEnd	BOOL	/	/	齿轮结束
o_xError	BOOL	/	/	齿轮故障
o_xAborted	BOOL	/	/	齿轮中断
wFBErrorID	SMC_ERROR	/	/	故障代码
xFLplc	BOOL	/	/	繁易 PLC 标志

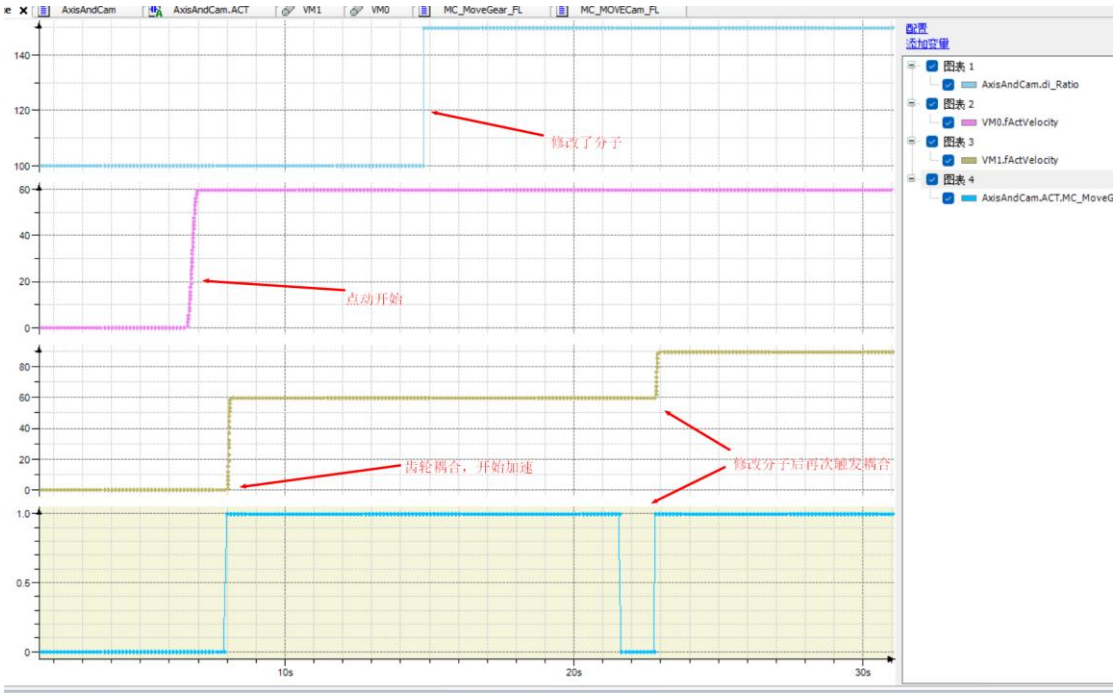
3、使用样例

（1）声明与调用

```
MC_Power0: MC_Power;
MC_Power1: MC_Power;
MC_Jog0: MC_Jog;
lr_AxisVel   :LREAL:=60;
di_Ratio     :DINT:=100;
MC_MoveGear_FL_0: MC_MoveGear_FL;
```



（2）功能详细演示部分



1：首先开启主轴点动，让主轴跑来。

2：随后触发齿轮耦合，让齿轮从轴加速追上主轴。

3：同步过程中，修改齿轮分子，再次触发耦合，齿轮从轴变速。

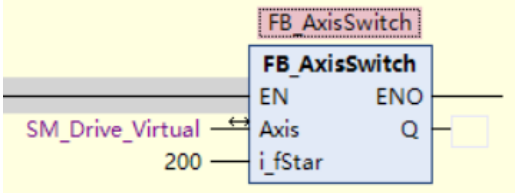
022 挺杆

FB_AxisSwitch 单个挺杆点

功能块说明：轴到达输入挺杆值输出。可用于运动控制中需要单个挺杆输出信号的场合，通过给定挺杆值进行输出。

1、指令格式

图形表现



ST 表现

```
FB_AxisSwitch(  Axis:=SM_Drive_Virtual ,
                i_fStar:=200 ,
                Q=> );
```

2、功能块变量说明

（1）输入输出变量（Var In_Out）

变量名称	数据类型	默认值	有效范围	描述
Axis	AXIS_REF_SM3	/	/	指定轴，即 AXIS_REF_SM3 的一个实例

（2）输入变量（VAR_Input）

变量名称	数据类型	默认值	有效范围	描述
i_fStar	LREAL	/	/	挺杆点值

（3）输出变量（VAR_Output）

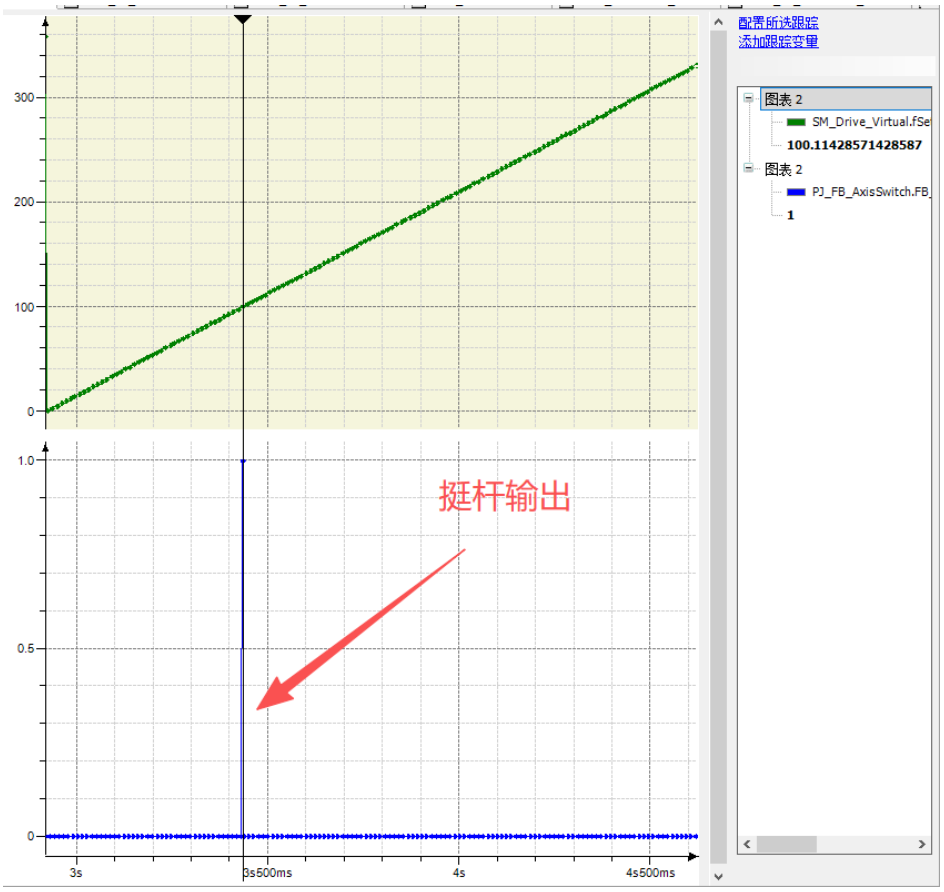
输出变量	数据类型	默认值	有效范围	描述
Q	BOOL	/	/	输出

3、使用样例

（1）声明与调用

```
    FB_AxisSwitch:      FB_AxisSwitch;//功能块声明
    r_TappetPosition:   REAL := 100;//挺杆位置

FB_AxisSwitch(  Axis:=SM_Drive_Virtual ,
                i_fStar:=r_TappetPosition ,
                Q=> );
```

（2）曲线显示：

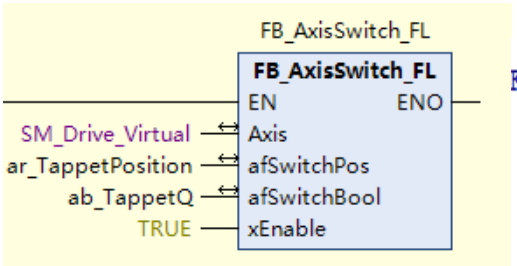


FB_AxisSwitch_FL 群挺杆点

功能块说明：轴到达输入数组挺杆值进行输出。可用于运动控制中需要多个挺杆输出信号的场合，通过给定数组挺杆值进行输出。

1、指令格式

图形表现



ST 表现

```
FB_AxisSwitch_FL(   Axis:=SM_Drive_Virtual ,
                    afSwitchPos:=ar_TappetPosition ,
                    afSwitchBool:=ab_TappetQ ,
                    xEnable:=TRUE );
```

2、功能块变量说明

（1）输入输出变量（Var In_Out）

变量名称	数据类型	默认值	有效范围	描述
Axis	AXIS_REF_SM3	/	/	指定轴，即 AXIS_REF_SM3 的一个实例
afSwitchPos	ARRAY[*]OF LREAL	/	/	输入的挺杆点数组
afSwitchBool	ARRAY[*]OF BOOL	/	/	输出的结果数组

（2）输入变量（VAR_Input）

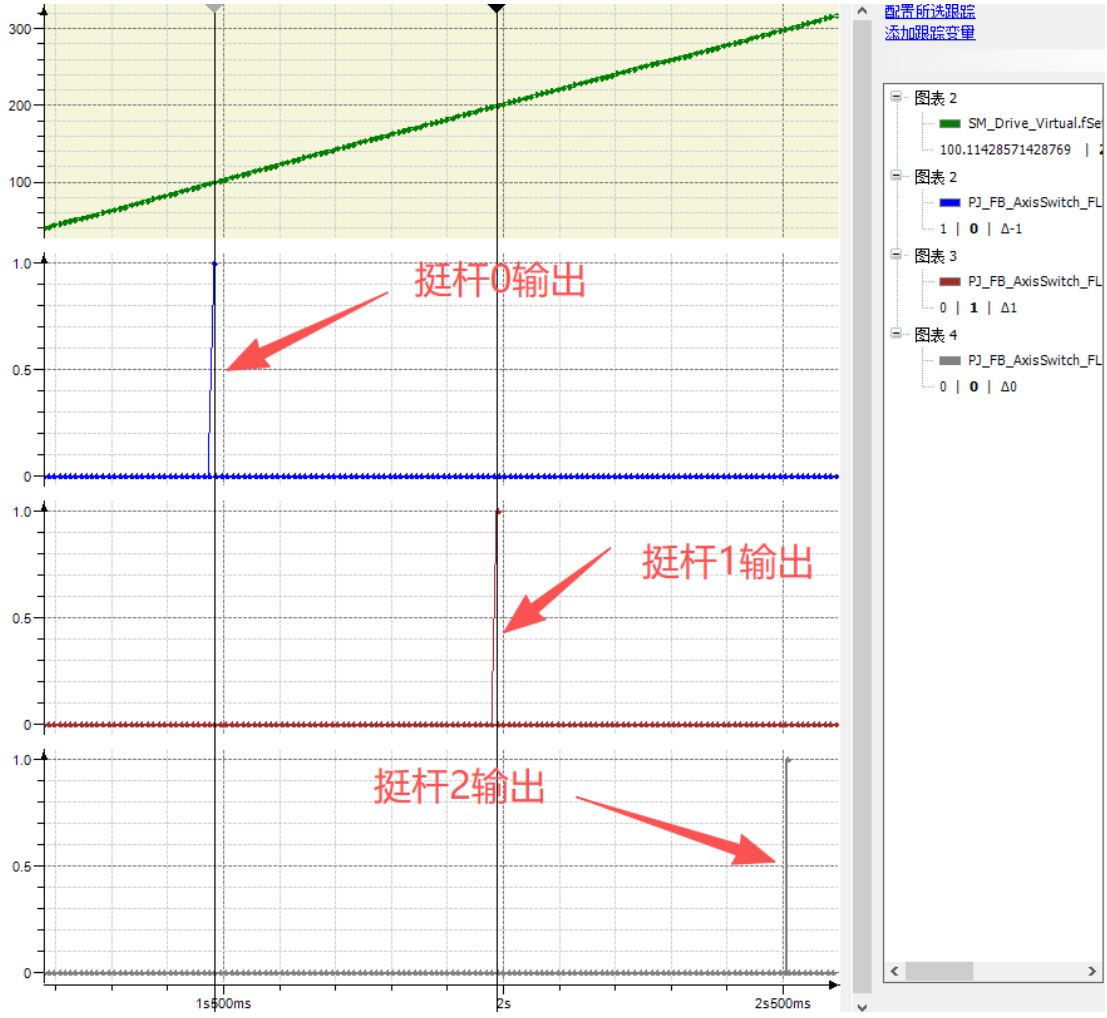
变量名称	数据类型	默认值	有效范围	描述
xEnable	BOOL	/	/	启用功能块

3、使用样例

（1）声明与调用

```
FB_AxisSwitch_FL: FB_AxisSwitch_FL;//功能块声明
ar_TappetPosition : ARRAY[0..2] OF LREAL := [100, 200, 300];//挺杆位置
ab_TappetQ : ARRAY[0..2] OF BOOL;//挺杆输出

FB_AxisSwitch_FL(   Axis:=SM_Drive_Virtual ,
                    afSwitchPos:=ar_TappetPosition ,
                    afSwitchBool:=ab_TappetQ ,
                    xEnable:=TRUE );
```

（2）曲线显示:

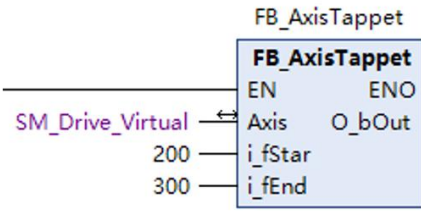


FB_AxisTappet 挺杆区间

功能块说明：轴到达输入挺杆范围输出。可用于运动控制中需要挺杆在特定的范围输出信号的场合

1、指令格式

图形表现



ST 表现

```
FB_AxisTappet(  Axis:=SM_Drive_Virtual ,
                i_fStar:=200 ,
                i_fEnd:=300 ,
                O_bOut=> );
```

2、功能块变量说明

（1）输入输出变量（Var In_Out）

变量名称	数据类型	默认值	有效范围	描述
Axis	AXIS_REF_SM3	/	/	指定轴，即 AXIS_REF_SM3 的一个实例

（2）输入变量（VAR_Input）

变量名称	数据类型	默认值	有效范围	描述
i_fStar	LREAL	/	/	挺杆开始位置
i_fEnd	LREAL	/	/	挺杆结束位置

（3）输出变量（VAR_Output）

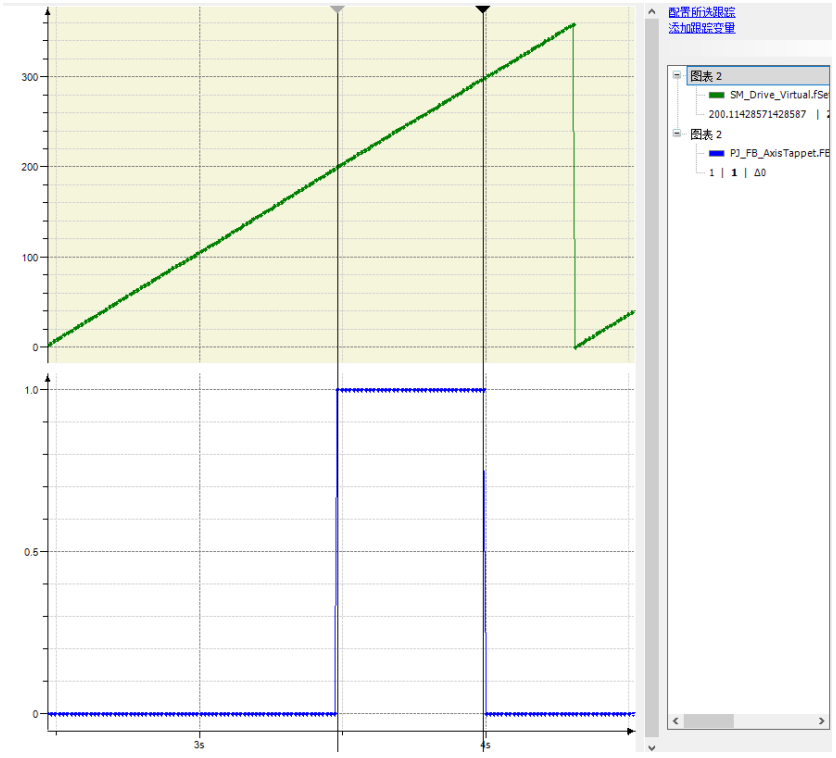
输出变量	数据类型	默认值	有效范围	描述
O_bOut	BOOL	/	/	输出

3、使用样例

（1）声明与调用

```
FB_AxisTappet: FB_AxisTappet;//功能块声明

FB_AxisTappet(  Axis:=SM_Drive_Virtual ,
                i_fStar:=200 ,
                i_fEnd:=300 ,
                O_bOut=> );
```

（2）曲线显示:



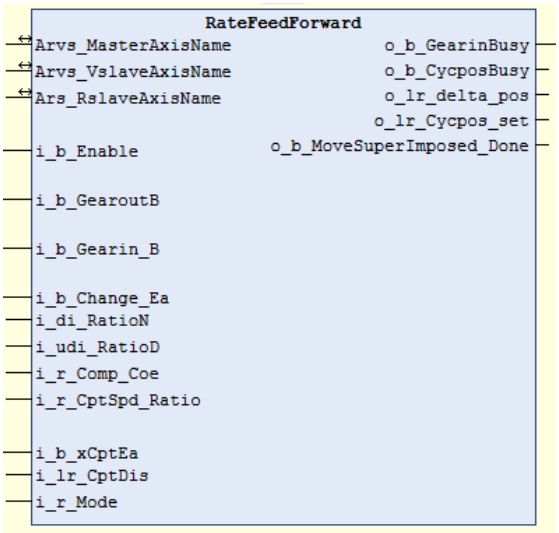
023 速度前馈

RateFeedForward 从轴速度前馈

功能块说明:通过给凸轮从轴一些前馈值,将主从轴在不同速度下维持在一个可控制的范围,缩小主从轴跟随误差，提高精度。

1、指令格式

图形表现



ST 表现

```
RateFeedForward(
Arvs_MasterAxisName:= ,
Arvs_VslaveAxisName:= ,
Ars_RslaveAxisName:= ,
i_b_Enable:= ,
i_b_GearoutB:= ,
i_b_Gearin_B:= ,
i_b_Change_Ea:= ,
i_di_RatioN:= ,
i_udi_RatioD:= ,
i_r_Comp_Coe:= ,
i_r_CptSpd_Ratio:= ,
i_b_xCptEa:= ,
i_lr_CptDis:= ,
i_r_Mode:= ,
o_b_GearinBusy=> ,
o_b_CycposBusy=> ,
o_lr_delta_pos=> ,
o_lr_Cycpos_set=> ,
o_b_MoveSuperImposed_Done=> );
```

2、功能块变量说明

（1）输入输出变量（Var In_Out）

变量名称	数据类型	默认值	有效范围	描述
Arvs_Master AxisName	AXIS_REF_VIRTUAL _SM3	/	/	虚拟主轴
Arvs_VslaveA xisName	AXIS_REF_VIRTUAL _SM3	/	/	虚拟从轴
Ars_RslaveAx isName	AXIS_REF_VIRTUAL _SM3	/	/	实从轴

（2）输入变量（VAR_Input）

变量名称	数据类型	默认值	有效范围	描述
i_b_Enable	BOOL	FALSE	/	功能块使能

i_b _GearoutB	BOOL	FALSE	/	解绑信号（上升沿有效）
i_b _Gearin_B	BOOL	FALSE	/	轴绑定信号（上升沿有效）
i_di_RatioN	DINT	/	/	齿轮绑定分子
i_udi_RatioD	UDINT	/	/	齿轮绑定分母
i_r_Comp_Coe	REAL	/	/	速度前馈系数（万分之一）
i_r_CptSpd_Ratio	REAL	/	/	叠加速度系数（系数*vioArvs_MasterAxisName.fActVelocity*0.01）
i_b_xCptEa	BOOL	FALSE	/	叠加补偿信号输入
i_lr_CptDis	REAL	/	/	叠加补偿值输入
i_r_Mode	REAL	/	/	印刷模式，轮转式/间歇式（变速允许位置（模式 1 时有效，模式 0 直接变速）

（3）输出变量（VAR_Output）

输出变量	数据类型	默认值	有效范围	描述
o_b_GearinBusy	BOOL	FALSE	/	虚拟主轴虚拟从轴绑定 ing
o_b_CycposBusy	BOOL	FALSE	/	实轴绑定 ing
o_lr_delta_pos	LREAL	/	/	误差量（虚拟从轴 setpos-实轴 actpos）
o_lr_Cycpos_set	LREAL	/	/	给定实轴位置
o_b_MoveSuperImposed_Done	BOOL	FALSE	/	叠加完成

3、使用样例

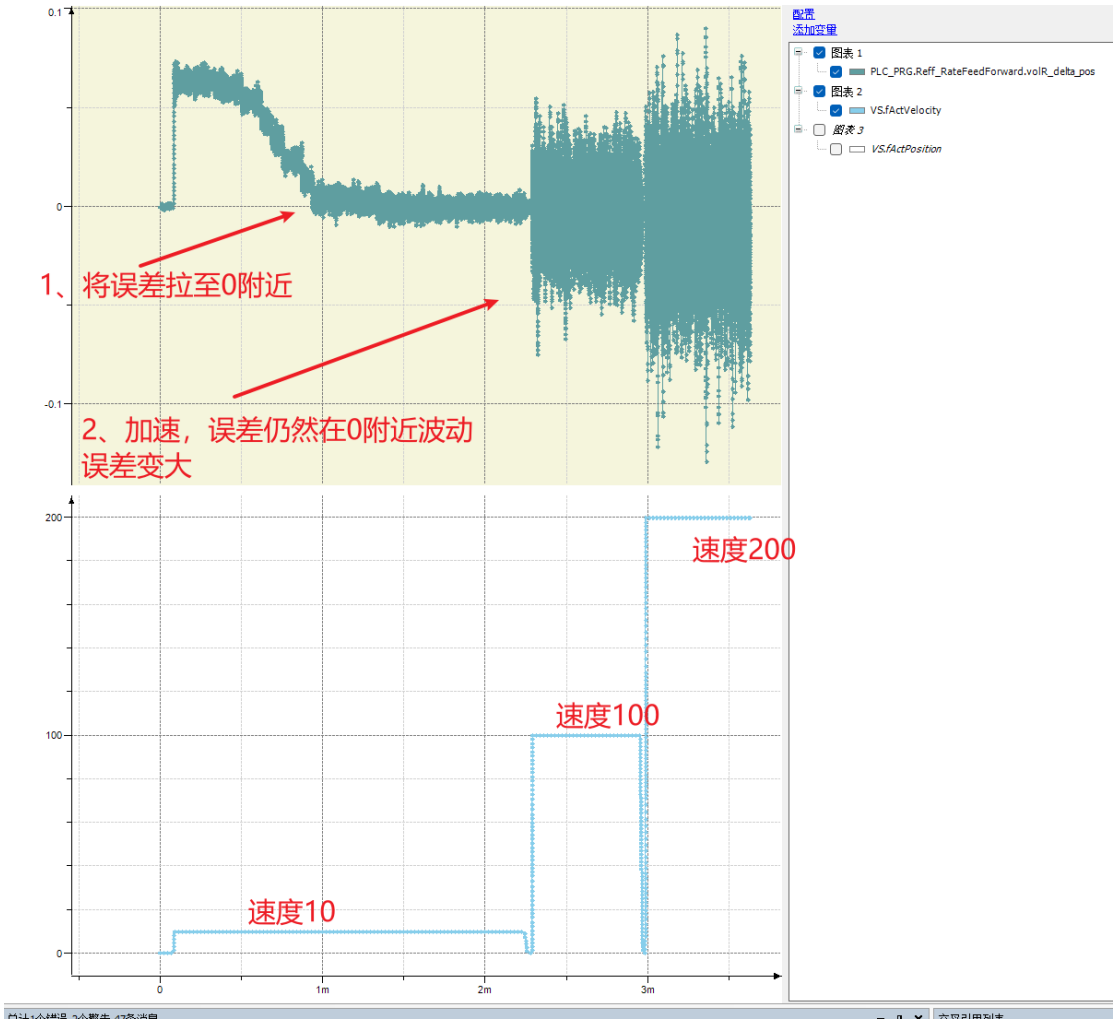
（1） 声明与调用

```
PROGRAM EtherCAT_A
VAR
        //POWER
        Mp_PowerR:MC_Power;
        Mp_PowerM:MC_Power;
        Mp_PowerS:MC_Power;
        //JOG
        Mj_JogM:MC_Jog;
        vB_JogForward   :BOOL;
        vB_JogBackward  :BOOL;
        vR_Velocity     :REAL;      //速度
        vB_JogBusy      :BOOL;      //运行ing

        //速度前馈
        Reff_RateFeedForward:RateFeedForward;


END_VAR
```

```
Reff_RateFeedForward(
Arvs_MasterAxisName:= VM,                                   //虚拟主轴
Arvs_VslaveAxisName:= VS,                                   //虚拟从轴
Ars_RslaveAxisName:= R1,                                    //实从轴
i_b_Enable:= TRUE,                                          //功能块使能
i_b_GearoutB:= GVL_Encryption.giB_m_GearoutB,                   //解绑信号（上升沿有效）
i_b_Gearin_B:= GVL_Encryption.giB_m_Gearin_B,                   //轴绑定信号（上升沿有效）
i_b_Change_Ea:= ,                                               //变速允许位置（模式1时有效，模式0直接变速）
i_di_RatioN:= 1,                                                //齿轮绑定分子
i_udi_RatioD:= 1,                                               //齿轮绑定分母
i_r_Comp_Coe:= GVL_Encryption.giR_m_Comp_Coe,                   //速度前馈系数（万分之一）
i_r_CptSpd_Ratio:= ,                                            //叠加速度系数(系数*vioArvs_MasterAxisName.fActVelocity*0.01)
i_b_xCptEa:= ,                                                      //补偿信号输入
i_lr_CptDis:= ,                                                 //补偿值输入
i_r_Mode:= ,                                                    //印刷模式，轮转式/间歇式（变速允许位置（模式1时有效，模式0直接变速）
o_b_GearinBusy=> ,                                                  //虚拟主轴虚拟从轴绑定ing
o_b_CycposBusy=> ,                                                  //实轴绑定ing
o_lr_delta_pos=> ,                                              //误差量（虚拟从轴setpos-实轴actpos）
o_lr_Cycpos_set=> ,                                             //给定实轴位置
o_b_MoveSuperImposed_Done=> );                                  //叠加完成
```

（2）功能详细演示部分

在正常从轴给定与反馈随着速度增大误差会向一边。
此时增加从轴速度前馈，将误差拉至 0 附近。
主轴加速，误差仍然维持在 0 附近波动。



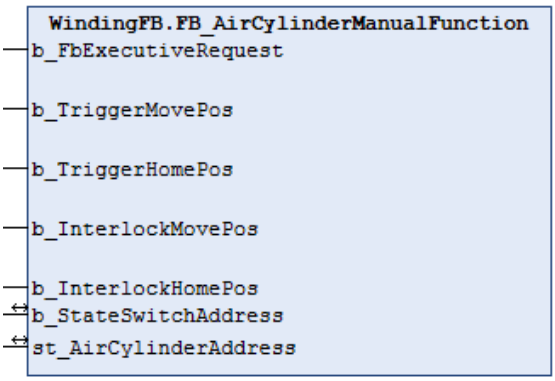
03 气缸与信号控制相关

FB_AirCylinderManualFunction 气缸手动控制

功能描述：（1）根据 ON 和 OFF 按钮切换气缸状态地址 （2）可以根据状态切换地址切换气缸状态，画面只有按钮时可直接控制此地址

1、指令格式

图形表现



ST 表现

```
WindingFB.FB_AirCylinderManualFunction(
    b_FbExecutiveRequest:= ,
    b_TriggerMovePos:= ,
    b_TriggerHomePos:= ,
    b_InterlockMovePos:= ,
    b_InterlockHomePos:= ,
    b_StateSwitchAddress:= ,
    st_AirCylinderAddress:= );
```

2、功能块变量说明

（1）输入输出变量（Var In_Out）

变量名称	数据类型	默认值	有效范围	描述
b_StateSwitchAddress	BOOL	FALSE	/	状态切换地址
st_AirCylinderAddress	ST_AirCylinder	/	/	气缸变量，气缸结构体
i_iMovePosDoneDelay	INT	/	/	st_AirCylinderAddress.成员，I_i 动位到位延时
i_iHomePosDoneDelay	INT	/	/	st_AirCylinderAddress.成员，I_i 原位到位延时
i_iNotTargetPosDelay_Alarm	INT	/	/	st_AirCylinderAddress.成员，I_i 不到位报警时间
i_bMovePosTrigger	BOOL	FALSE	/	st_AirCylinderAddress.成员，I_b 动位触发
i_bHomePosTrigger	BOOL	FALSE	/	st_AirCylinderAddress.成员，i_b 原位触发
i_bMovePosInterlock	BOOL	FALSE	/	st_AirCylinderAddress.成员，I_b 动位互锁条件
i_bHomePosInterlock	BOOL	FALSE	/	st_AirCylinderAddress.成员，I_b 原位互锁条件
o_bMovePosNotTargetPos_Alarm	BOOL	FALSE	/	st_AirCylinderAddress.成员，O_b 动位不到位报警
o_bHomePosNotTargetPos_Alarm	BOOL	FALSE	/	st_AirCylinderAddress.成员，O_b 原位不到位报警
o_bMovePosInterlock_Alarm	BOOL	FALSE	/	st_AirCylinderAddress.成员，O_b 动位互锁报警
o_bHomePosInterlock_Alarm	BOOL	FALSE	/	st_AirCylinderAddress.成员，O_b 原位互锁报警

o_bMovePosOut	BOOL	FALSE	/	st_AirCylinderAddress.成员，O_b 动位输出
o_bHomePosOut	BOOL	FALSE	/	st_AirCylinderAddress.成员，O_b 原位输出
o_bMovePosTargetPos_Sign	BOOL	FALSE	/	st_AirCylinderAddress.成员，O_b 动位到位标志
o_bHomePosTargetPos_Sign	BOOL	FALSE	/	st_AirCylinderAddress.成员，O_b 原位到位标志
iStandby1	BOOL	FALSE	/	st_AirCylinderAddress.成员，备用 1
iStandby2	BOOL	FALSE	/	st_AirCylinderAddress.成员，备用 2
iStandby3	BOOL	FALSE	/	st_AirCylinderAddress.成员，备用 3

（2）输入变量（VAR_Input）

变量名称	数据类型	默认值	有效范围	描述
b_FbExecutiveRequest	BOOL	FALSE	/	b 模块执行请求
b_TriggerMovePos	BOOL	FALSE	/	b 触发动位
b_TriggerHomePos	BOOL	FALSE	/	b 触发原位
b_InterlockMovePos	BOOL	FALSE	/	b 动位互锁条件
b_InterlockHomePos	BOOL	FALSE	/	b 原位互锁条件

3、使用样例

（1）声明与调用

FB_AirCylinderManualFunction [从WindingFB]

```
PROGRAM FB_AirCylinderManualFunction
VAR
    fbAcmf_AirCylinderManualFunction        :WindingFB.FB_AirCylinderManualFunction;
    st_AirCylinderAddress                   :WindingFB.ST_AirCylinder;//st气缸变量 : 气缸结构体;

    b_StateSwitchAddress                    :BOOL;//b状态切换地址:BOOL;
END_VAR
```

```
fbAcmf_AirCylinderManualFunction(
    b_FbExecutiveRequest:= ,        //b模块执行请求 :BOOL;
    b_TriggerMovePos:= ,            //b触发动位:BOOL;
    b_TriggerHomePos:= ,            //b触发原位:BOOL;
    b_InterlockMovePos:= ,          //b动位互锁条件 : BOOL ;
    b_InterlockHomePos:= ,          //b原位互锁条件 : BOOL ;
    b_StateSwitchAddress:=b_StateSwitchAddress ,    //b状态切换地址:BOOL;
    st_AirCylinderAddress:=st_AirCylinderAddress    //st气缸变量 : 气缸结构体;
    );
```

（2）功能详细演示部分

手动输出无互锁条件:

b_TriggerMovePos=TRUE AND b_FbExecutiveRequest=TRUE 时，b_StateSwitchAddress 和 ST_AirCylinder. i_bMovePosTrigger 输出 TRUE



表达式	类型	值	准备值	地	注释
fbAcmf_AirCylinderManualFunction	Windi...				
b_FbExecutiveRequest	BOOL	TRUE			b模块执行请求 :BOOL;
b_TriggerMovePos	BOOL	TRUE			b触发动位:BOOL;
b_TriggerHomePos	BOOL	FALSE			b触发原位:BOOL;
b_InterlockMovePos	BOOL	FALSE			b动位互锁条件 : BOOL ;
b_InterlockHomePos	BOOL	FALSE			b原位互锁条件 : BOOL ;
b_StateSwitchAddress	REFE...	TRUE			b状态切换地址:BOOL;
st_AirCylinderAddress	REFE...				st气缸变量 : 气缸结构体;
i_iMovePosDoneDelay	INT	0			I_i动位到位延时 : INT ;
i_iHomePosDoneDelay	INT	0			I_i原位到位延时 : INT ;
i_iNotTargetPosDelay_Alarm	INT	0			I_i不到位报警时间 : INT ;
i_bMovePosTrigger	BOOL	TRUE			I_b动位触发 : BOOL ;
i_bHomePosTrigger	BOOL	FALSE			i_b原位触发 : BOOL ;
i_bMovePosSignal	BOOL	FALSE			I_b动位信号 : BOOL ;
i_bHomePosSignal	BOOL	FALSE			I_b原位信号 : BOOL ;
i_bMovePosInterlock	BOOL	FALSE			I_b动位互锁条件 : BOOL ;
i_bHomePosInterlock	BOOL	FALSE			I_b原位互锁条件 : BOOL ;
o_bMovePosNotTargetPos_Alarm	BOOL	FALSE			O_b动位不到位报警 : BOOL ;
o_bHomePosNotTargetPos_Alarm	BOOL	FALSE			O_b原位不到位报警 : BOOL ;
o_bMovePosInterlock_Alarm	BOOL	FALSE			O_b动位互锁报警 : BOOL ;
o_bHomePosInterlock_Alarm	BOOL	FALSE			O_b原位互锁报警 : BOOL ;
o_bMovePosOut	BOOL	FALSE			O_b动位输出 : BOOL ;
o_bHomePosOut	BOOL	FALSE			O_b原位输出 : BOOL ;
o_bMovePosTargetPos_Sign	BOOL	FALSE			O_b动位到位标志 : BOOL ;

```
fbAcmf_AirCylinderManualFunction(
    b_FbExecutiveRequest:= ,        //b模块执行请求 :BOOL;
    b_TriggerMovePos:= ,            //b触发动位:BOOL;
    b_TriggerHomePos:= ,            //b触发原位:BOOL;
    b_InterlockMovePos:= ,          //b动位互锁条件 : BOOL ;
    b_InterlockHomePos:= ,          //b原位互锁条件 : BOOL ;
    b_StateSwitchAddress TRUE :=b_StateSwitchAddress TRUE , //b状态切换地址:BOOL;
    st_AirCylinderAddress:=st_AirCylinderAddress    //st气缸变量 : 气缸结构体;
    );RETURN
```

手动输出有互锁条件:

b_TriggerMovePos=TRUE AND b_FbExecutiveRequest=TRUE 并 b_InterlockMovePos=TRUE 时，b_StateSwitchAddress 和 ST_AirCylinder. i o_bMovePosInterlock_Alarm 输出 TREU，ST_AirCylinder. i_bMovePosTrigger 输出 FALSE;



表达式	类型	值	准备值	地	注释
fbAcmf_AirCylinderManualFunction	Windi...				
b_FbExecutiveRequest	BOOL	TRUE			b模块执行请求 :BOOL;
b_TriggerMovePos	BOOL	TRUE			b触发动位:BOOL;
b_TriggerHomePos	BOOL	FALSE			b触发原位:BOOL;
b_InterlockMovePos	BOOL	TRUE			b动位互锁条件 : BOOL ;
b_InterlockHomePos	BOOL	FALSE			b原位互锁条件 : BOOL ;
b_StateSwitchAddress	REFE...	TRUE			b状态切换地址:BOOL;
st_AirCylinderAddress	REFE...				st气缸变量 : 气缸结构体;
i_iMovePosDoneDelay	INT	0			I_i动位到位延时 : INT ;
i_iHomePosDoneDelay	INT	0			I_i原位到位延时 : INT ;
i_iNotTargetPosDelay_Alarm	INT	0			I_i不到位报警时间 : INT ;
i_bMovePosTrigger	BOOL	FALSE			I_b动位触发 : BOOL ;
i_bHomePosTrigger	BOOL	FALSE			i_b原位触发 : BOOL ;
i_bMovePosSignal	BOOL	FALSE			I_b动位信号 : BOOL ;
i_bHomePosSignal	BOOL	FALSE			I_b原位信号 : BOOL ;
i_bMovePosInterlock	BOOL	FALSE			I_b动位互锁条件 : BOOL ;
i_bHomePosInterlock	BOOL	FALSE			I_b原位互锁条件 : BOOL ;
o_bMovePosNotTargetPos_Alarm	BOOL	FALSE			O_b动位不到位报警 : BOOL ;
o_bHomePosNotTargetPos_Alarm	BOOL	FALSE			O_b原位不到位报警 : BOOL ;
o_bMovePosInterlock_Alarm	BOOL	TRUE			O_b动位互锁报警 : BOOL ;
o_bHomePosInterlock_Alarm	BOOL	FALSE			O_b原位互锁报警 : BOOL ;
o_bMovePosOut	BOOL	FALSE			O_b动位输出 : BOOL ;
o_bHomePosOut	BOOL	FALSE			O_b原位输出 : BOOL ;
o_bMovePosTargetPos_Sign	BOOL	FALSE			O_b动位到位标志 : BOOL ;

动位互锁触发

互锁报警

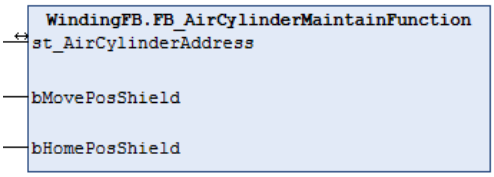
```
fbAcmf_AirCylinderManualFunction(
    b_FbExecutiveRequest:= ,        //b模块执行请求 :BOOL;
    b_TriggerMovePos:= ,            //b触发动位:BOOL;
    b_TriggerHomePos:= ,            //b触发原位:BOOL;
    b_InterlockMovePos:= ,          //b动位互锁条件 : BOOL ;
    b_InterlockHomePos:= ,          //b原位互锁条件 : BOOL ;
    b_StateSwitchAddress TRUE :=b_StateSwitchAddress TRUE , //b状态切换地址:BOOL;
    st_AirCylinderAddress:=st_AirCylinderAddress    //st气缸变量 : 气缸结构体;
    );RETURN
```

FB_AirCylinderMaintainFunction 气缸对应的 IO,延时，报警等维护

功能描述：(1)互锁报警判断 (2)输出点控制（3）判断是否屏蔽动位原位信号（4）到位信号判断（5）不到位报警判断

1、指令格式

图形表现



ST 表现

```
//FB_AirCylinderMaintainFunction   FB_气缸维护
WindingFB.FB_AirCylinderMaintainFunction(
 st_AirCylinderAddress:= ,      //st气缸变量
 bMovePosShield:= ,
 bHomePosShield:=
 );
```

2、功能块变量说明

（1）输入输出变量（Var In_Out）

变量名称	数据类型	默认值	有效范围	描述
st_AirCylinderAddress	ST_AirCylinder	/	/	结构体类型
i_iMovePosDoneDelay	INT	/	/	st_AirCylinderAddress.成员，I_i 动位到位延时
i_iHomePosDoneDelay	INT	/	/	st_AirCylinderAddress.成员，I_i 原位到位延时
i_iNotTargetPosDelay_Alarm	INT	/	/	st_AirCylinderAddress.成员，I_i 不到位报警时间
i_bMovePosTrigger	BOOL	FALSE	/	st_AirCylinderAddress.成员，I_b 动位触发
i_bHomePosTrigger	BOOL	FALSE	/	st_AirCylinderAddress.成员，i_b 原位触发
i_bMovePosInterlock	BOOL	FALSE	/	st_AirCylinderAddress.成员，I_b 动位互锁条件
i_bHomePosInterlock	BOOL	FALSE	/	st_AirCylinderAddress.成员，I_b 原位互锁条件
o_bMovePosNotTargetPos_Alarm	BOOL	FALSE	/	st_AirCylinderAddress.成员，O_b 动位不到位报警
o_bHomePosNotTargetPos_Alarm	BOOL	FALSE	/	st_AirCylinderAddress.成员，O_b 原位不到位报警
o_bMovePosInterlock_Alarm	BOOL	FALSE	/	st_AirCylinderAddress.成员，O_b 动位互锁报警
o_bHomePosInterlock_Alarm	BOOL	FALSE	/	st_AirCylinderAddress.成员，O_b 原位互锁报警
o_bMovePosOut	BOOL	FALSE	/	st_AirCylinderAddress.成员，O_b 动位输出
o_bHomePosOut	BOOL	FALSE	/	st_AirCylinderAddress.成员，O_b 原位输出
o_bMovePosTargetPos_Sign	BOOL	FALSE	/	st_AirCylinderAddress.成员，O_b 动位到位标志
o_bHomePosTargetPos_Sign	BOOL	FALSE	/	st_AirCylinderAddress.成员，O_b 原位到位标志
iStandby1	BOOL	FALSE	/	st_AirCylinderAddress.成员，备用 1
iStandby2	BOOL	FALSE	/	st_AirCylinderAddress.成员，备用 2

iStandby3	BOOL	FALSE	/	st_AirCylinderAddress.成员，备用 3

（2）输入变量（VAR_Input）

变量名称	数据类型	默认值	有效范围	描述
bMovePosShield	BOOL	FALSE	/	功能块使能
bHomePosShield	BOOL	FALSE	/	解绑信号（上升沿有效）

3、使用样例

（1）声明与调用

```
PROGRAM ST_App
VAR

        Facmf_AirCylinderMaintainFunction           :WindingFB.FB_AirCylinderMaintainFunction;
        st_AirCylinderAddress                       :WindingFB.ST_AirCylinder;//st气缸变量 : 气缸结构体;
END_VAR
```

```
//FB_AirCylinderMaintainFunction   FB_气缸维护功能块
Facmf_AirCylinderMaintainFunction(
 st_AirCylinderAddress:=st_AirCylinderAddress ,     //st气缸变量 : 气缸结构体;
 bMovePosShield:= ,                                 //b动位屏蔽: BOOL;
 bHomePosShield:=                                   //b原位屏蔽: BOOL;
 );
```

（2）功能详细演示部分

a、i_iMovePosDoneDelay（动位到位延时）设置 4 秒：i_bMovePosSignal=TRUE 延迟 4 秒输出 o_bMovePosTargetPos_Sign;

b、i_iHomePosDoneDelay（原位到位延时）设置 4 秒：i_bHomePosSignal=TRUE 延迟 4 秒输出 o_bHomePosTargetPos_Sign;

c、i_iNotTargetPosDelay_Alarm（不到位报警时间）设置 8 秒：动位未到位 8 秒 OR 动位信号消失 8/10 秒，报警 o_bMovePosNotTargetPos_Alarm/ o_bHomePosNotTargetPos_Alarm 输出 TRUE;

d、o_bMovePosInterlock_Alarm =TRUE 时，i_bMovePosTrigger 动位触发输出无效。

E、o_bHomePosInterlock_Alarm=TRUE 时，i_bHomePosSignal 原位触发输出无效。



st_AirCylinderAddress	Windi...			st气缸变量：气缸结构体；
i_iMovePosDoneDelay	INT	1		I_i动位到位延时：INT；
i_iHomePosDoneDelay	INT	1		I_i原位到位延时：INT；
i_iNotTargetPosDelay_Alarm	INT	5		I_i不到位报警时间：INT；
i_bMovePosTrigger	BOOL	TRUE		I_b动位触发：BOOL；
i_bHomePosTrigger	BOOL	FALSE		i_b原位触发：BOOL；
i_bMovePosSignal	BOOL	FALSE		I_b动位信号：BOOL；
i_bHomePosSignal	BOOL	FALSE		I_b原位信号：BOOL；
i_bMovePosInterlock	BOOL	FALSE		I_b动位互锁条件：BOOL；
i_bHomePosInterlock	BOOL	FALSE		I_b原位互锁条件：BOOL；
o_bMovePosNotTargetPos_Alarm	BOOL	TRUE		O_b动位不到位报警：BOOL；
o_bHomePosNotTargetPos_Alarm	BOOL	FALSE		O_b原位不到位报警：BOOL；
o_bMovePosInterlock_Alarm	BOOL	FALSE		O_b动位互锁报警：BOOL；
o_bHomePosInterlock_Alarm	BOOL	FALSE		O_b原位互锁报警：BOOL；
o_bMovePosOut	BOOL	TRUE		O_b动位输出：BOOL；
o_bHomePosOut	BOOL	FALSE		O_b原位输出：BOOL；
o_bMovePosTargetPos_Sign	BOOL	FALSE		O_b动位到位标志：BOOL；
o_bHomePosTargetPos_Sign	BOOL	FALSE		O_b原位到位标志：BOOL；
iStandby1	BOOL	FALSE		备用1：INT；
iStandby2	BOOL	FALSE		备用2：INT；

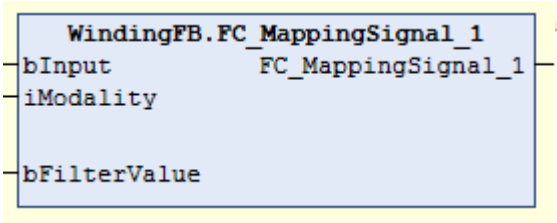
```
1
2
3   //FB_AirCylinderMaintainFunction   FB_气缸维护功能块
4   Facmf_AirCylinderMaintainFunction(
5     st_AirCylinderAddress:=st_AirCylinderAddress ,     //st气缸变量 : 气缸结构体;
6     bMovePosShield:= ,                                 //b动位屏蔽: BOOL;
7     bHomePosShield:= TRUE                                    //b原位屏蔽: BOOL;
8     );RETURN
```

FC_MappingSignal_1 输入信号常开/常闭/屏蔽处理_1 位

功能说明：1）输入映射；2）屏蔽输入；

1、指令格式

图形表现



ST 表现

```
WindingFB.FC_MappingSignal_1(
bInput:= ,
iModality:= ,
bFilterValue:= )
```

2、功能块变量说明

输入变量（VAR_Input）

变量名称	数据类型	默认值	有效范围	描述
bInput	BOOL	FALSE	/	b 输入
iModality	INT	/	/	i 形式
bFilterValue	BOOL	TRUE	/	b 屏蔽值

3、使用样例

（1）声明与调用

FC_MappingSignal_1

```
PROGRAM FC_MappingSignal_1
VAR

    Di_b_X0     :BOOL;
    OUT:BOOL;
END_VAR
```

```
OUT:=WindingFB.FC_MappingSignal_1(
bInput:=Di_b_X0 ,
iModality:=1,
bFilterValue:=TRUE );
```

（2）功能详细演示部分

iModality：=0：OUT:=bInput ;

FC_MappingSignal_1 := bInput_1 ;

IModality == 0 时 对 bInput 取反;

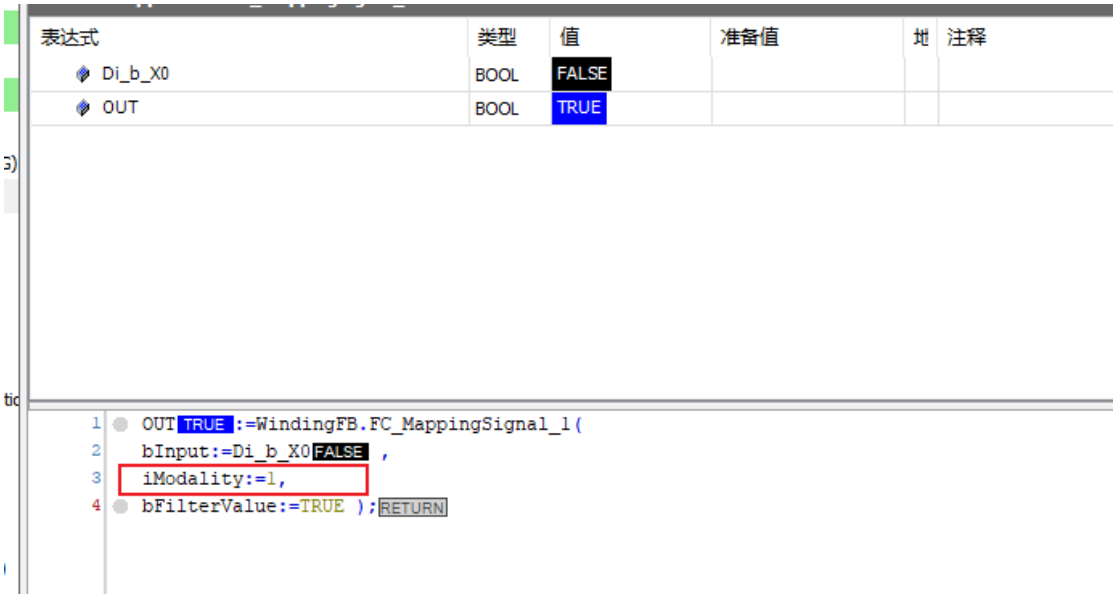
iModality > 1 时 对 bInput :=bFilterValue ;

表达式	类型	值	准备值
Di_b_X0	BOOL	TRUE	
OUT	BOOL	TRUE	

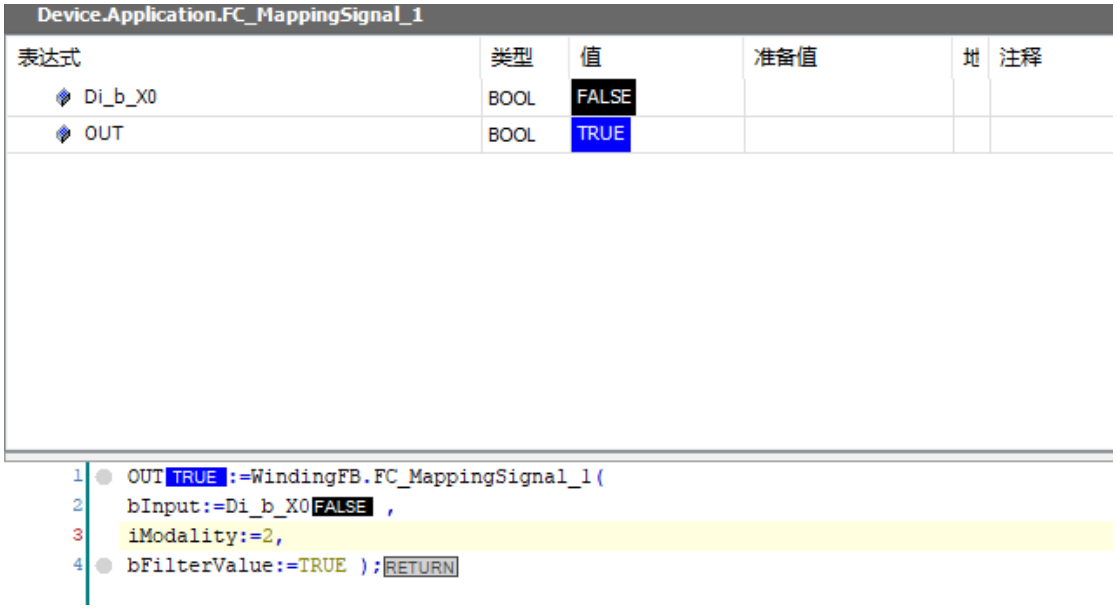
```
OUT TRUE :=WindingFB.FC_MappingSignal_1(
bInput:=Di_b_X0 TRUE ,
iModality:=0,
bFilterValue:= );RETURN
```

形式为0

iModality：=1：OUT:=NOT(bInput)；



iModality：=2：OUT:=bFilterValue；

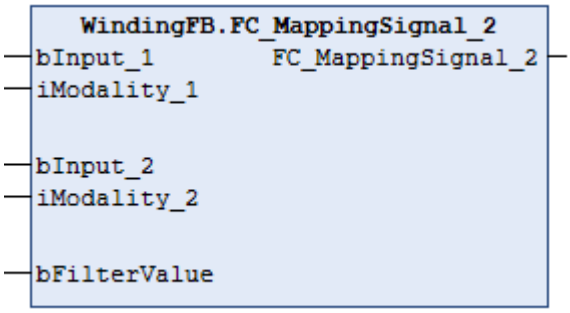


FC_MappingSignal_2 输入信号常开/常闭/屏蔽处理_2 位

功能说明：1）输入映射；2）屏蔽输入；

1、指令格式

图形表现



ST 表现

```
OUT:=   WindingFB.FC_MappingSignal_2(
    bInput_1:= ,
    iModality_1:= ,
    bInput_2:= ,
    iModality_2:= ,
    bFilterValue:=
    );
```

2、功能块变量说明

输入变量（VAR_Input）

变量名称	数据类型	默认值	有效范围	描述
bInput_1	BOOL	FALSE	/	b 输入 1
iModality_1	INT	/	/	i 形式 1
bInput_2	BOOL	FALSE	/	b 输入 2
iModality_2	INT	/	/	i 形式 2
bFilterValue	BOOL	TRUE	/	b 屏蔽值

3、使用样例

（1）声明与调用

FC_MappingSignal_2

```
1  PROGRAM FC_MappingSignal_2
2  VAR
3          OUT         :BOOL;
4          bInput_1    :BOOL;
5          bInput_2    :BOOL;
6  END_VAR
7
```

```
1  OUT:=   WindingFB.FC_MappingSignal_2(
2      bInput_1:= bInput_1,
3      iModality_1:= 0,
4      bInput_2:= bInput_2,
5      iModality_2:= 0,
6      bFilterValue:=
7      );
```

（2）功能详细演示部分

1、iModality_1:=0 ， iModality_2:=0 时：

OUT := bInput_1 AND bInput_2;（如下图）

2、iModality_1:=0 ， iModality_2:=1 时：

OUT := bInput_1 AND NOT(bInput_2);

3、iModality_1:=1 ， iModality_2:=0 时：

OUT := NOT(bInput_1) AND bInput_2;

4、iModality_1:=1 ， iModality_2:=1 时：

OUT := NOT(bInput_1) AND NOT(bInput_2);

5、iModality_1:=2 ， iModality_2:=2 ， bFilterValue： =TRUE 时：

OUT := bFilterValue;

6、iModality_1:=0 ， iModality_2:=2 ， bFilterValue： =TRUE 时：

OUT := NOT(bInput_1) AND bFilterValue;

...

不一一举例，原则：

FC_MappingSignal_2 := bInput_1 AND bInput_2;

IModality == 0 时 对 bInput 取反;

iModality > 1 时 对 bInput :=bFilterValue ;

Device.Application.FC_MappingSignal_2

表达式	类型	值	准备值
OUT	BOOL	TRUE	
bInput_1	BOOL	TRUE	
bInput_2	BOOL	TRUE	

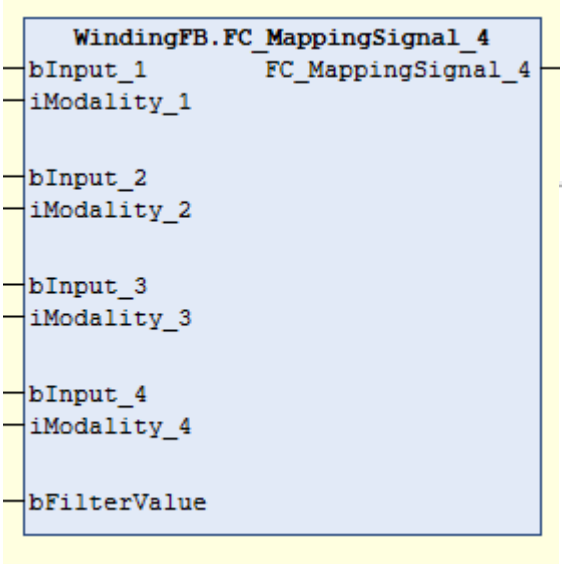
```
1  OUT TRUE := WindingFB.FC_MappingSignal_2(
2      bInput_1:= bInput_1 TRUE ,
3      iModality_1:= 0,
4      bInput_2:= bInput_2 TRUE ,
5      iModality_2:= 0,
6      bFilterValue:=
7      );RETURN
```

FC_MappingSignal_4 输入信号常开/常闭/屏蔽处理_4 位

功能说明：1）输入映射；2）屏蔽输入；

1、指令格式

图形表现



ST 表现

```
WindingFB.FC_MappingSignal_4(
    bInput_1:= ,
    iModality_1:= ,
    bInput_2:= ,
    iModality_2:= ,
    bInput_3:= ,
    iModality_3:= ,
    bInput_4:= ,
    iModality_4:= ,
    bFilterValue:= )
```

2、功能块变量说明

输入变量（VAR_Input）

变量名称	数据类型	默认值	有效范围	描述
bInput_1	BOOL	FALSE	/	b 输入 1
iModality_1	INT	/	/	i 形式 1
bInput_2	BOOL	FALSE	/	b 输入 2
iModality_2	INT	/	/	i 形式 2
bInput_3	BOOL	FALSE	/	b 输入 3
iModality_3	INT	/	/	i 形式 3
bInput_4	BOOL	FALSE	/	b 输入 4
iModality_4	INT	/	/	i 形式 4
bFilterValue	BOOL	TRUE	/	b 屏蔽值

3、使用样例

（1）声明与调用

FC_MappingSignal_4

```
PROGRAM FC_MappingSignal_4
VAR
        OUT         :BOOL;
        bInput_1    :BOOL;
        bInput_2    :BOOL;
        bInput_3    :BOOL;
        bInput_4    :BOOL;

END_VAR
```

```
OUT:=WindingFB.FC_MappingSignal_4(
    bInput_1:=bInput_1 ,
    iModality_1:=0 ,
    bInput_2:=bInput_2 ,
    iModality_2:=0 ,
    bInput_3:=bInput_3 ,
    iModality_3:= 0,
    bInput_4:=bInput_4 ,
    iModality_4:= 0,
    bFilterValue:= );
```

（2）功能详细演示部分

1、iModality_1:=0 ， iModality_2:=0，iModality_3:=0，iModality_4:=0 时：

OUT := bInput_1 AND bInput_2 AND bInput_3 AND bInput_4;（如下图）

…

不一一举例，原则：

FC_MappingSignal_4 :=

bInput_1 AND bInput_2 AND bInput_3 AND bInput_4;

IModality == 0 时 对 bInput 取反;

iModality > 1 时 对 bInput :=bFilterValue ;

Device.Application.FC_MappingSignal_4

表达式	类型	值	准备值
OUT	BOOL	TRUE	
bInput_1	BOOL	TRUE	
bInput_2	BOOL	TRUE	
bInput_3	BOOL	TRUE	
bInput_4	BOOL	TRUE	

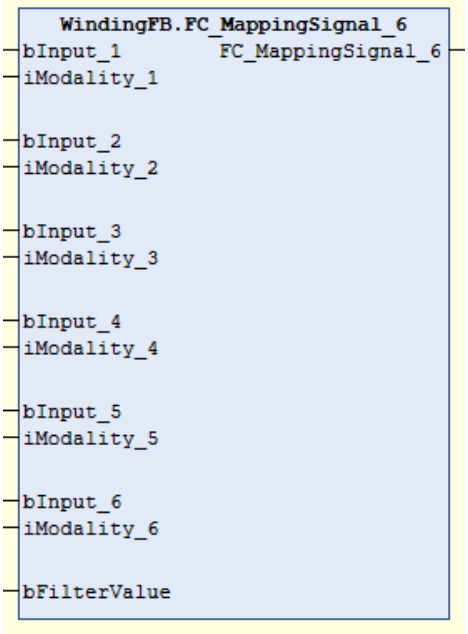
```
1  OUT TRUE :=WindingFB.FC_MappingSignal_4(
2      bInput_1:=bInput_1 TRUE ,
3      iModality_1:=0 ,
4      bInput_2:=bInput_2 TRUE ,
5      iModality_2:=0 ,
6      bInput_3:=bInput_3 TRUE ,
7      iModality_3:= 0,
8      bInput_4:=bInput_4 TRUE ,
9      iModality_4:= 0,
10     bFilterValue:= );RETURN
```

FC_MappingSignal_6 输入信号常开/常闭/屏蔽处理_6 位

功能说明：1）输入映射；2）屏蔽输入；

1、指令格式

图形表现



ST 表现

```
WindingFB.FC_MappingSignal_6(
    bInput_1:= ,
    iModality_1:= ,
    bInput_2:= ,
    iModality_2:= ,
    bInput_3:= ,
    iModality_3:= ,
    bInput_4:= ,
    iModality_4:= ,
    bInput_5:= ,
    iModality_5:= ,
    bInput_6:= ,
    iModality_6:= ,
    bFilterValue:= )
```

输入变量（VAR_Input）

变量名称	数据类型	默认值	有效范围	描述
bInput_1	BOOL	FALSE	/	b 输入 1
iModality_1	INT	/	/	i 形式 1
bInput_2	BOOL	FALSE	/	b 输入 2
iModality_2	INT	/	/	i 形式 2
bInput_3	BOOL	FALSE	/	b 输入 3
iModality_3	INT	/	/	i 形式 3
bInput_4	BOOL	FALSE	/	b 输入 4
iModality_4	INT	/	/	i 形式 4
bInput_5	BOOL	FALSE	/	b 输入 5
iModality_5	INT	/	/	i 形式 5
bInput_6	BOOL	FALSE	/	b 输入 6
iModality_6	INT	/	/	i 形式 6
bFilterValue	BOOL	TRUE	/	b 屏蔽值

3、使用样例

（1）声明与调用

```
PROGRAM FC_MappingSignal_6
VAR
        OUT         :BOOL;
        bInput_1    :BOOL;
        bInput_2    :BOOL;
        bInput_3    :BOOL;
        bInput_4    :BOOL;
        bInput_5    :BOOL;
        bInput_6    :BOOL;
END_VAR
```

```
OUT:=WindingFB.FC_MappingSignal_6(
    bInput_1:= bInput_1,
    iModality_1:= 0,
    bInput_2:= bInput_2,
    iModality_2:= 0,
    bInput_3:= bInput_3,
    iModality_3:= 0,
    bInput_4:= bInput_4,
    iModality_4:= 0,
    bInput_5:= bInput_5,
    iModality_5:= 0,
    bInput_6:= bInput_6,
    iModality_6:= 0,
    bFilterValue:= );
```

（2）功能详细演示部分

1、iModality_1:=0 ，iModality_2:=0，iModality_3:=0，iModality_4:=0， iModality_5:=0，iModality_6:=0 时：

OUT := bInput_1 AND bInput_2 AND bInput_3 AND bInput_4 AND bInput_5 AND bInput_6;

（如下图）

...

不一一举例，原则：

FC_MappingSignal_6 :=

bInput_1 AND bInput_2 AND bInput_3 AND bInput_4 AND bInput_5 AND bInput_6;

IModality == 0 时 对 bInput 取反;

iModality > 1 时 对 bInput :=bFilterValue ;



表达式	类型	值	准备值
OUT	BOOL	TRUE	
bInput_1	BOOL	TRUE	
bInput_2	BOOL	TRUE	
bInput_3	BOOL	TRUE	
bInput_4	BOOL	TRUE	
bInput_5	BOOL	TRUE	
bInput_6	BOOL	TRUE	

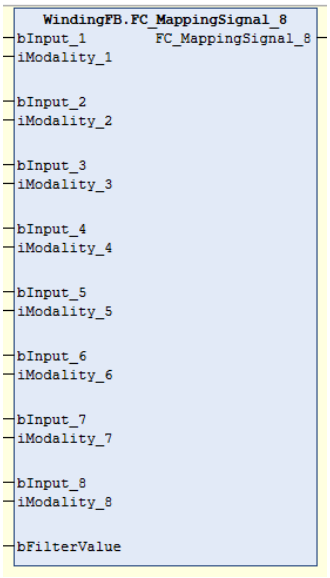
```
OUT TRUE :=WindingFB.FC_MappingSignal_6(
    bInput_1:= bInput_1 TRUE ,
    iModality_1:= 0,
    bInput_2:= bInput_2 TRUE ,
    iModality_2:= 0,
    bInput_3:= bInput_3 TRUE ,
    iModality_3:= 0,
    bInput_4:= bInput_4 TRUE ,
    iModality_4:= 0,
    bInput_5:= bInput_5 TRUE ,
    iModality_5:= 0,
    bInput_6:= bInput_6 TRUE ,
    iModality_6:= 0,
    bFilterValue:= );RETURN
```

FC_MappingSignal_8 输入信号常开/常闭/屏蔽处理_8 位

功能说明：1）输入映射；2）屏蔽输入；

1、指令格式

图形表现



ST 表现

```
WindingFB.FC_MappingSignal_8(
    bInput_1:= ,
    iModality_1:= ,
    bInput_2:= ,
    iModality_2:= ,
    bInput_3:= ,
    iModality_3:= ,
    bInput_4:= ,
    iModality_4:= ,
    bInput_5:= ,
    iModality_5:= ,
    bInput_6:= ,
    iModality_6:= ,
    bInput_7:= ,
    iModality_7:= ,
    bInput_8:= ,
    iModality_8:= ,
    bFilterValue:= );
```

输入变量（VAR_Input）

变量名称	数据类型	默认值	有效范围	描述
bInput_1	BOOL	FALSE	/	b 输入 1
iModality_1	INT	/	/	i 形式 1
bInput_2	BOOL	FALSE	/	b 输入 2
iModality_2	INT	/	/	i 形式 2
bInput_3	BOOL	FALSE	/	b 输入 3
iModality_3	INT	/	/	i 形式 3
bInput_4	BOOL	FALSE	/	b 输入 4
iModality_4	INT	/	/	i 形式 4
bInput_5	BOOL	FALSE	/	b 输入 5
iModality_5	INT	/	/	i 形式 5
bInput_6	BOOL	FALSE	/	b 输入 6
iModality_6	INT	/	/	i 形式 6
bInput_7	BOOL	FALSE	/	b 输入 7
iModality_7	INT	/	/	i 形式 7
bInput_8	BOOL	FALSE	/	b 输入 8
iModality_8	INT	/	/	i 形式 8
bFilterValue	BOOL	TRUE	/	b 屏蔽值

3、使用样例

（1）声明与调用

FC_MappingSignal_8

```
PROGRAM FC_MappingSignal_8
VAR
        OUT         :BOOL;
        bInput_1    :BOOL;
        bInput_2    :BOOL;
        bInput_3    :BOOL;
        bInput_4    :BOOL;
        bInput_5    :BOOL;
        bInput_6    :BOOL;
        bInput_7    :BOOL;
        bInput_8    :BOOL;
END_VAR
```

```
OUT:=WindingFB.FC_MappingSignal_8(
    bInput_1:= bInput_1,
    iModality_1:= 0,
    bInput_2:= bInput_2,
    iModality_2:= 0,
    bInput_3:= bInput_3,
    iModality_3:= 0,
    bInput_4:= bInput_4,
    iModality_4:= 0,
    bInput_5:= bInput_5,
    iModality_5:= 0,
    bInput_6:= bInput_6,
    iModality_6:= 0,
    bInput_7:= bInput_7,
    iModality_7:= 0,
    bInput_8:= bInput_8,
    iModality_8:= 0,
    bFilterValue:= );
```

（2）功能详细演示部分

1、iModality_1:=0 ，iModality_2:=0，iModality_3:=0，iModality_4:=0， iModality_5:=0，iModality_6:=0 时：

OUT := bInput_1 AND bInput_2 AND bInput_3 AND bInput_4 AND bInput_5 AND bInput_6 AND bInput_7 AND bInput_8;（如下图）

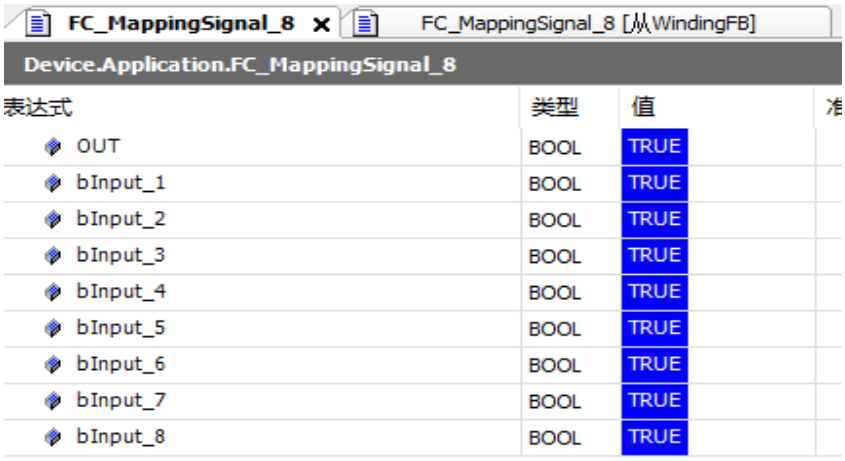
...

不一一举例，原则：

FC_MappingSignal_6 := bInput_1 AND bInput_2 AND bInput_3 AND bInput_4 AND bInput_5 AND bInput_6 AND bInput_7 AND bInput_8;

IModality == 0 时 对 bInput 取反;

iModality > 1 时 对 bInput :=bFilterValue ;




```
OUT TRUE :=WindingFB.FC_MappingSignal_8(
    bInput_1:= bInput_1 TRUE,
    iModality_1:= 0,
    bInput_2:= bInput_2 TRUE,
    iModality_2:= 0,
    bInput_3:= bInput_3 TRUE,
    iModality_3:= 0,
    bInput_4:= bInput_4 TRUE,
    iModality_4:= 0,
    bInput_5:= bInput_5 TRUE,
    iModality_5:= 0,
    bInput_6:= bInput_6 TRUE,
    iModality_6:= 0,
    bInput_7:= bInput_7 TRUE,
    iModality_7:= 0,
    bInput_8:= bInput_8 TRUE,
    iModality_8:= 0,
    bFilterValue:= );RETURN
```

04 通讯相关

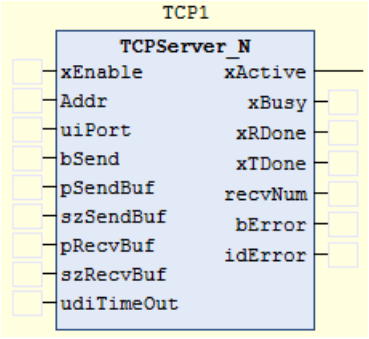
TCPServer_N

指令名称：TCPServer_N　　TCP 通讯（PLC 做 Server）基于 NBS.CAA 库

功能块说明：本功能块用于 PLC 和其他设备之间做 TCP Socket 通讯（PLC 做 Server），并且进行数据的收发。

1、指令格式

图形表现：



ST 表现：

```
TCP1(
    xEnable:= ,
    Addr:= ,
    uiPort:= ,
    bSend:= ,
    pSendBuf:= ,
    szSendBuf:= ,
    pRecvBuf:= ,
    szRecvBuf:= ,
    udiTimeOut:= ,
    xActive=> ,
    xBusy=> ,
    xRDone=> ,
    xTDone=> ,
    recvNum=> ,
    bError=> ,
    idError=> );
```

2、相关变量说明

变量	名称	数据类型	输入输出属性	描述
xEnable	创建服务器	BOOL	输入变量	创建服务器
Addr	IP 地址	String	输入变量	IP
uiPort	端口号	UINT	输入变量	端口号
bSend	发送数据	BOOL	输入变量	发送
pSendBuf	发送数据的地址	POINTER TO BYTE	输入变量	发送数据的地址指针
szSendBuf	发送数据的长度	UDINT	输入变量	发送长度
pRecvBuf	接收数据的指针	POINTER TO BYTE	输入变量	接收数据的地址指针
szRecvBuf	接收数据的长度	UDINT	输入变量	接收长度
udiTimeOut	超时时间	UDINT	输入变量	超时时间 ms
xAxtive	侦听到客户端	BOOL	输出变量	侦听到客户端
xBusy	服务器正在运行	BOOL	输出变量	服务器正在运行中
xRDone	完成信号	BOOL	输出变量	侦听到数据的完成信号（只存在一个周期）
xTDone	发送完成	BOOL	输出变量	发送数据帧完成
RecvNum	收到的数据个数	NBS.CAA.SIZE	输出变量	接收到的数据个数

bError	报错标志	BOOL	输出变量	错误标志位
idError	Error ID	NBS.ERROR	输出变量	报错 ID；取自 NBS.Error

3.具体使用

（1）声明与实例化

```
PROGRAM P2
VAR
        TCP_Server_0: TCPServer_N;//功能块声明
        RevData:DWORD;//接收的数据
        SendData:DWORD;//发送的数据
END_VAR
```

```
TCP_Server_0(
    xEnable:= ,
    Addr:= '192.168.1.100',  //PLC  IP
    uiPort:=  8181,             //端口号
    bSend:= ,
    pSendBuf:= ADR(SendData), //发送数据地址指针,
    szSendBuf:=SIZEOF(SendData), //发送到的数据大小
    pRecvBuf:= ADR(RevData) , //接收到数据的地址指针,
    szRecvBuf:= SIZEOF(RevData) , //接收到数据的大小
    udiTimeOut:= 100,
    xActive=> ,
    xBusy=> ,
    xRDone=> ,
    xTDone=> ,
    recvNum=> ,
    bError=> ,
    idError=> );
```

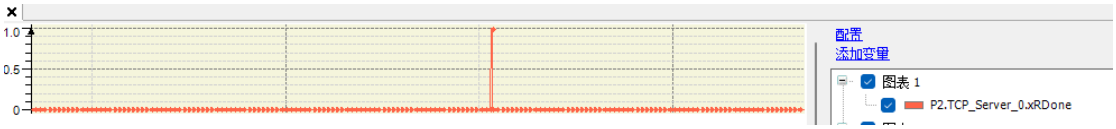
（2）连接。PLC IP：192.168.1.100；电脑 IP ：192.168.1.19；PLC 做 Server，那么电脑软件就做 Client，电脑端使用 SSCOM 工具



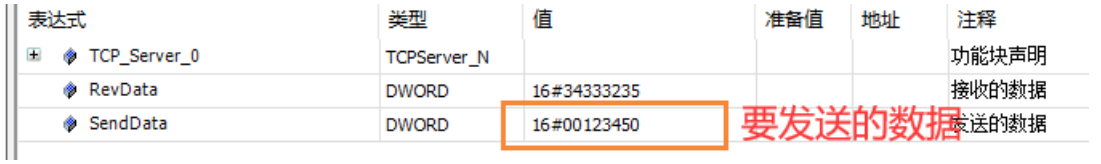
（3）PLC 接收数据。用调试工具发送数据，PLC 可以收到数据。数据在调试助手中显示是低位在左侧，Codesys 数据中低位在右。 .xRDone 只存在一个周期，可以用 Trace 抓取

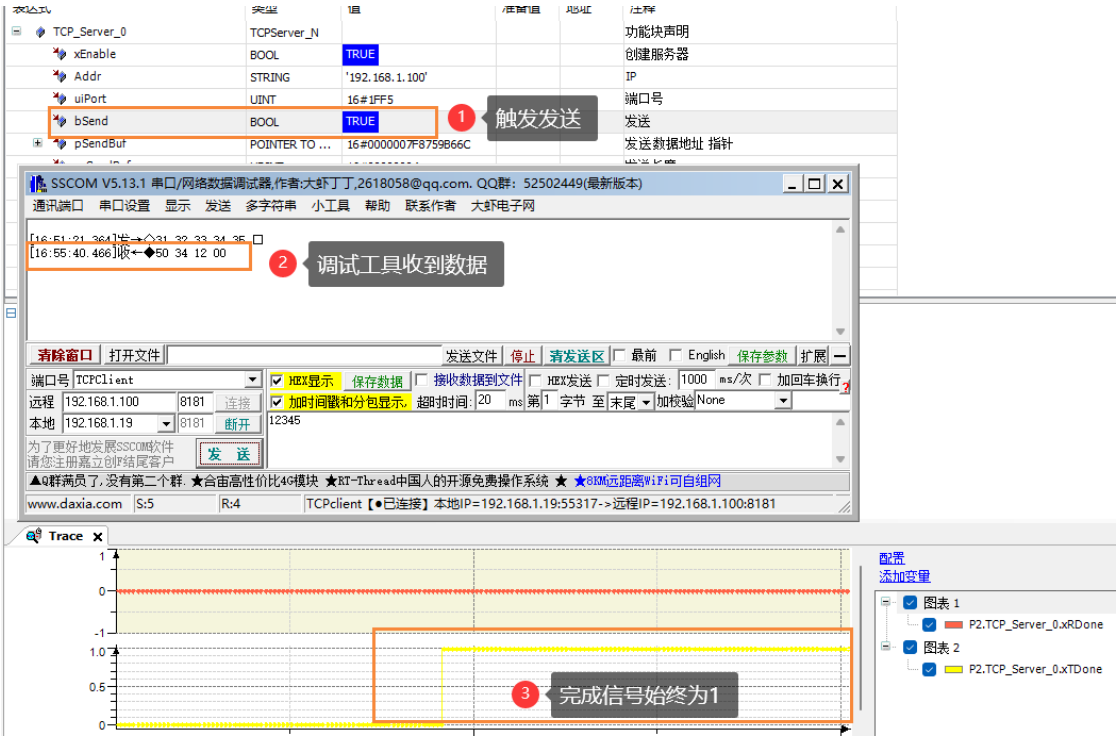


完成信号只存在一个周期，可以用 Trace 抓取



（4）PLC 发送数据。用调试工具接收来自 PLC 发送的数据。数据在调试助手中显示是低位在左侧，Codesys 数据中低位在右。 .xTDone 发送成功后始终为 1.


表达式
类型
值
准备值
地址
注释
TCP_Server_0
TCPServer_N
功能块声明
RevData
DWORD
16#34333235
接收的数据
SendData
DWORD
16#00123450
发送的数据
要发送的数据


TCP_Server_0
TCPServer_N
功能块声明
xEnable
BOOL
TRUE
创建服务器
Addr
STRING
'192.168.1.100'
IP
uiPort
UINT
16#1FF5
端口号
bSend
BOOL
TRUE
发送
pSendBuf
POINTER TO ...
16#0000007F8759B66C
发送数据地址 指针
1
触发发送
SSCOM V5.13.1 串口/网络数据调试器,作者:大虾丁丁,2618058@qq.com. QQ群：52502449(最新版本)
通讯端口 串口设置 显示 发送 多字符串 小工具 帮助 联系作者 大虾电子网
[16:55:40.466]收←◆50 34 12 00
2
调试工具收到数据
清除窗口
打开文件
发送文件
停止
清发送区
最前
English
保存参数
扩展
端口号
TCPClient
HEX显示
保存数据
接收数据到文件
HEX发送
定时发送:
1000
ms/次
加回车换行
远程
192.168.1.100
8181
连接
加时间戳和分包显示,
超时时间:
20
ms
第
1
字节 至
末尾
加校验
None
本地
192.168.1.19
8181
断开
12345
为了更好地发展SSCOM软件
请您注册嘉立创P结尾客户
发 送
▲Q群满员了，没有第二个群. ★合宙高性价比4G模块 ★RT-Thread中国人的开源免费操作系统 ★ ★8KM远距离WiFi可自组网
www.daxia.com
S:5
R:4
TCPclient【●已连接】本地IP=192.168.1.19:55317->远程IP=192.168.1.100:8181
Trace
1
0
-1
1.0
0.5
0
3
完成信号始终为1
配置
添加变量
图表 1
P2.TCP_Server_0.xRDone
图表 2
P2.TCP_Server_0.xTDone

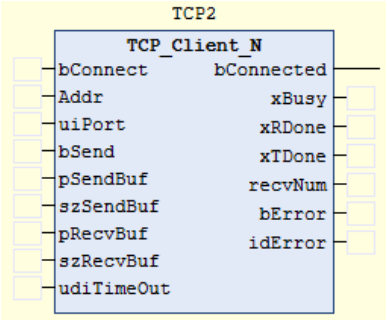
TCP_Client_N

指令名称：TCP_Client_N　　TCP 通讯（PLC 做 Client）基于 NBS.CAA 库

功能块说明：本功能块用于 PLC 与其他设备之间做 TCP Socket 通讯（PLC 做 Client），并且进行数据的收发。

1、指令格式

图形表现：



ST 表现：

```
TCP2(
    bConnect:= ,
    Addr:= ,
    uiPort:= ,
    bSend:= ,
    pSendBuf:= ,
    szSendBuf:= ,
    pRecvBuf:= ,
    szRecvBuf:= ,
    udiTimeOut:= ,
    bConnected=> ,
    xBusy=> ,
    xRDone=> ,
    xTDone=> ,
    recvNum=> ,
    bError=> ,
    idError=> );
```

2、相关变量说明

变量	名称	数据类型	输入输出属性	描述
bConnect	创建服务器	BOOL	输入变量	连接服务器
Addr	IP 地址	String	输入变量	IP
uiPort	端口号	UINT	输入变量	端口号
bSend	发送数据	BOOL	输入变量	发送
pSendBuf	发送数据的地址	POINTER TO BYTE	输入变量	发送数据的地址指针
szSendBuf	发送数据的长度	UDINT	输入变量	发送长度
pRecvBuf	接收数据的指针	POINTER TO BYTE	输入变量	接收数据的地址指针
szRecvBuf	接收数据的长度	UDINT	输入变量	接收长度
udiTimeOut	超时时间	UDINT	输入变量	超时时间 ms
bConnected	连上服务器标志	BOOL	输出变量	=TRUE 连接到服务器
xBusy	服务器正在运行	BOOL	输出变量	服务器正在运行中
xRDone	完成信号	BOOL	输出变量	侦听到数据的完成信号（只存在一个周期）
xTDone	发送完成	BOOL	输出变量	发送数据帧完成
RecvNum	收到的数据个数	NBS.CAA.SIZE	输出变量	接收到的数据个数
bError	报错标志	BOOL	输出变量	错误标志位
idError	Error ID	NBS.ERROR	输出变量	报错 ID；取自 NBS.Error

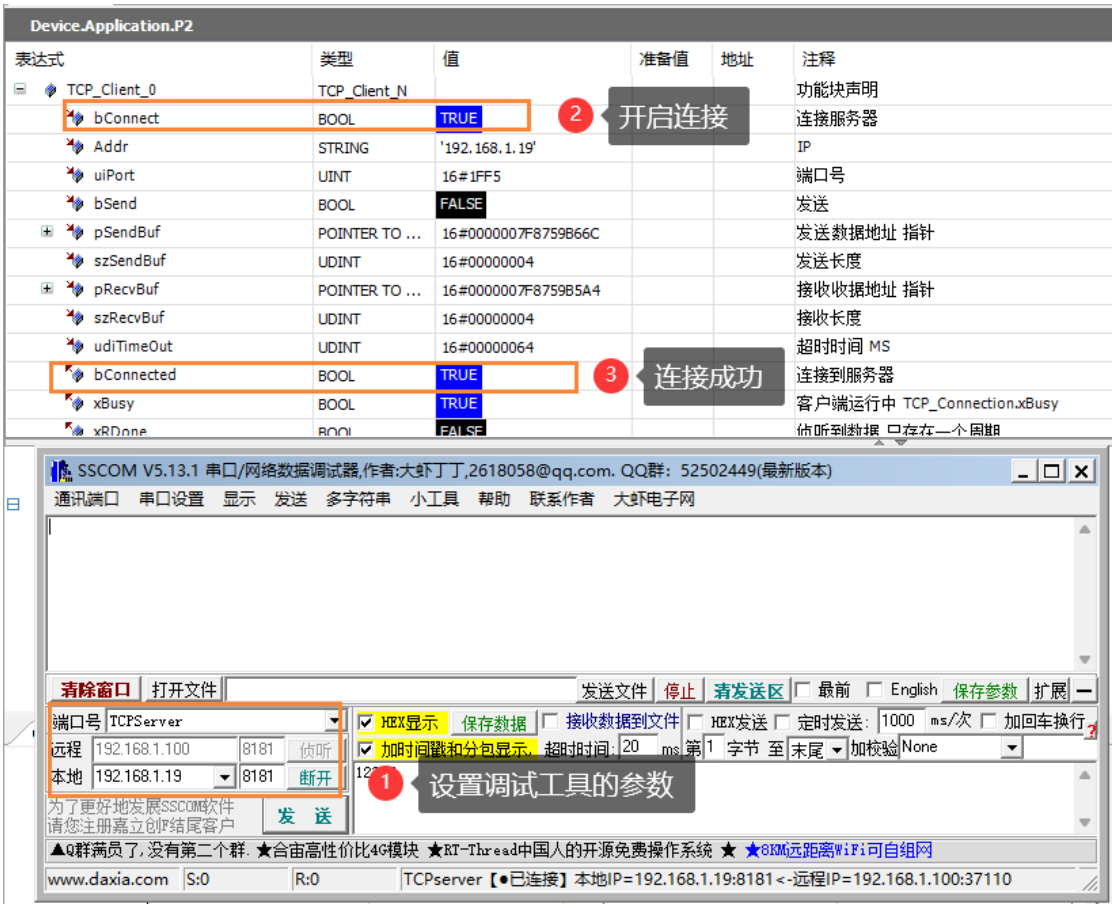
3.具体使用

（1）声明与实例化

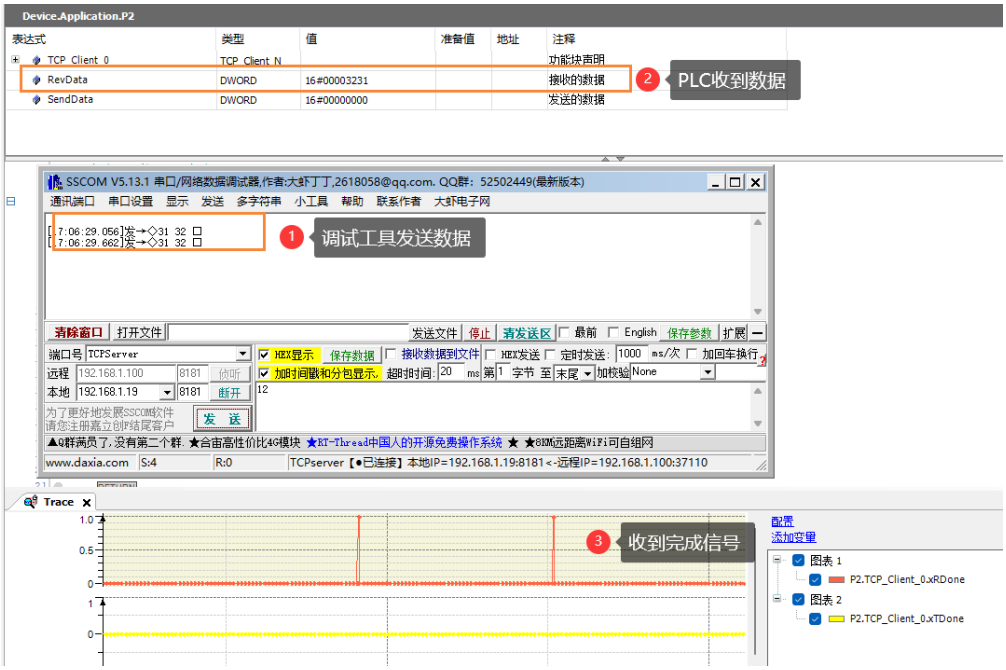
```
PROGRAM P2
VAR
        TCP_Client_0: TCP_Client_N;//功能块声明
        RevData:DWORD;//接收的数据
        SendData:DWORD;//发送的数据
END_VAR
```

```
//电脑网口接PLC。电脑IP：192.168.1.19；PLC IP：192.168.1.100
//电脑使用SSCOM软件仿真
TCP_Client_0(
    bConnect:= ,
    Addr:= '192.168.1.19',  //PLC  IP
    uiPort:=  8181,             //端口号
    bSend:= ,
    pSendBuf:= ADR(SendData), //发送数据地址指针,
    szSendBuf:=SIZEOF(SendData), //发送到的数据大小
    pRecvBuf:= ADR(RevData) , //接收到数据的地址指针,
    szRecvBuf:= SIZEOF(RevData) , //接收到数据的大小
    udiTimeOut:= 100,
    bConnected=> ,
    xBusy=> ,
    xRDone=> ,
    xTDone=> ,
    recvNum=> ,
    bError=> ,
    idError=> );
```

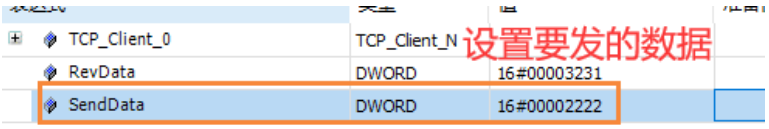
（2）连接。本样例中，电脑网口接 PLC。电脑 IP：192.168.1.19；PLC IP：192.168.1.100。电脑使用 SSCOM 软件连接。

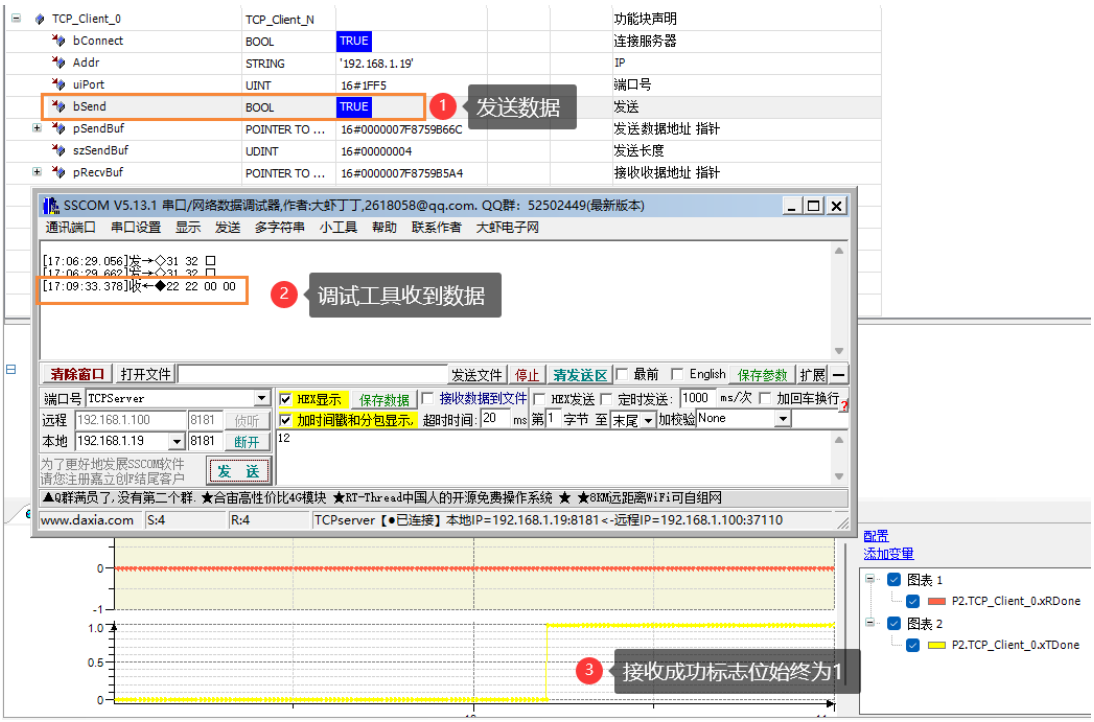


（3）PLC 接收数据。用调试工具发送数据，PLC 接收。数据在调试助手中显示是低位在左侧，Codesys 数据中低位在右。 .xRDone 只存在一个周期，可以用 Trace 抓取。



（4）PLC 发送数据。PLC 端发送数据，调试工具接收数据。数据在调试助手中显示是低位在左侧，Codesys 数据中低位在右。 .xTDone 发送成功后始终为 1.





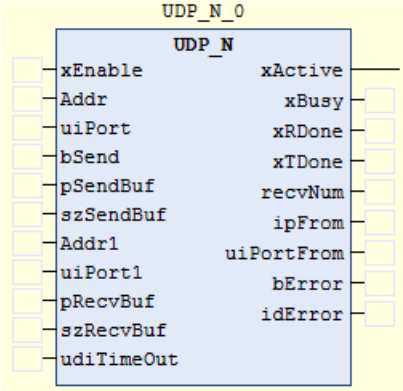
UDP_N

指令名称：UDP_N　　UDP 通讯，基于 NBS.CAA 库。

功能块说明：本功能块用于 PLC 与其他设备之间做 UDP 通讯，并且进行数据的收发。

1、指令格式

图形表现：



ST 表现：

```
UDP_N_0(
    xEnable:= ,
    Addr:= ,
    uiPort:= ,
    bSend:= ,
    pSendBuf:= ,
    szSendBuf:= ,
    Addr1:= ,
    uiPort1:= ,
    pRecvBuf:= ,
    szRecvBuf:= ,
    udiTimeOut:= ,
    xActive=> ,
    xBusy=> ,
    xRDone=> ,
    xTDone=> ,
    recvNum=> ,
    ipFrom=> ,
    uiPortFrom=> ,
    bError=> ,
    idError=> );
```

2、相关变量说明

变量	名称	数据类型	输入输出属性	描述
xEnable	运行	BOOL	输入变量	运行
Addr	本地 IP	String	输入变量	本地 IP
uiPort	本地端口号	UINT	输入变量	本地端口号
bSend	发送数据	BOOL	输入变量	发送
pSendBuf	发送数据的地址	POINTER TO BYTE	输入变量	发送数据的地址指针
szSendBuf	发送数据的长度	UDINT	输入变量	发送长度
Addr1	目标 IP	String	输入变量	目标 IP
uiPort1	目标端口号	UINT	输入变量	目标端口号
pRecvBuf	接收数据的指针	POINTER TO BYTE	输入变量	接收数据的地址指针
szRecvBuf	接收数据的长度	UDINT	输入变量	接收长度
udiTimeOut	超时时间	UDINT	输入变量	超时时间 ms
xAxtive	侦听到客户端	BOOL	输出变量	侦听到客户端
xBusy	服务器正在运行	BOOL	输出变量	服务器正在运行中
xRDone	完成信号	BOOL	输出变量	侦听到数据的完成信号（只存在一个周期）
xTDone	发送完成	BOOL	输出变量	发送数据帧完成

RecvNum	收到的数据个数	NBS.CAA.SIZE	输出变量	接收到的数据个数
ipFrom	来源 IP	String	输出变量	读取到数据包的源 IP
uiPortFrom	来源端口号	UINT	输出变量	读取到数据包的源端口号
bError	报错标志	BOOL	输出变量	错误标志位
idError	Error ID	NBS.ERROR	输出变量	报错 ID；取自 NBS.Error

3、具体使用介绍：

（1）声明与实例化

本样例中，电脑网口接 PLC。电脑 IP：192.168.1.19；PLC IP：192.168.1.100。电脑使用 SSCOM 软件连接。声明如下所示：

```
PROGRAM P2
VAR
        UDP_N_0: UDP_N;//UDP功能块声明
        RevData:DWORD;//接收的数据
        SendData:DWORD;//发送的数据
END_VAR
```

```
//电脑网口接PLC。电脑IP：192.168.1.19；PLC IP：192.168.
//电脑使用SSCOM软件仿真
UDP_N_0(
    xEnable:= ,
    Addr:= '192.168.1.100',  //PLC  IP
    uiPort:= 8181,           //端口号
    bSend:= ,
    pSendBuf:= ADR(SendData), //发送数据地址指针
    szSendBuf:= SIZEOF(SendData), //发送到的数据大小
    Addr1:= '192.168.1.19',  //设备IP 本例程为电脑IP
    uiPort1:= 8181,          //端口号
    pRecvBuf:=ADR(RevData) , //接收到数据的地址指针
    szRecvBuf:=SIZEOF(RevData) , //接收到数据的大小
    udiTimeOut:= 100,
    xActive=> ,
    xBusy=> ,
    xRDone=> ,
    xTDone=> ,
    recvNum=> ,
    ipFrom=> ,
    uiPortFrom=> ,
    bError=> ,
    idError=> );
```

（2）连接 UDP

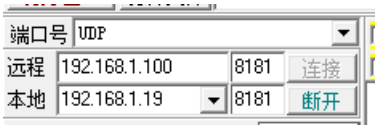
Device.Application.P2

表达式	类型	值	准备值	地址	注释
UDP_N_0	UDP_N				UDP功能块声明
xEnable	BOOL	TRUE			运行
Addr	STRING	'192.168.1.100'			本地IP
uiPort	UINT	16#1FF5			本地端口号
bSend	BOOL	TRUE			发送
pSendBuf	POINTER TO ...	16#0000007F8759B66C			发送数据地址 指针
szSendBuf	UDINT	16#00000004			发送长度
Addr1	STRING	'192.168.1.19'			发送目标IP
uiPort1	UINT	16#1FF5			发送目标端口号
pRecvBuf	POINTER TO ...	16#0000007F8759B5A4			接收收据地址 指针
szRecvBuf	UDINT	16#00000004			接收长度
udiTimeOut	UDINT	16#00000064			超时时间MS
xActive	BOOL	TRUE			UDP通讯创建成功
xBusy	BOOL	TRUE			Peerbusy

开启 连接

连接成功

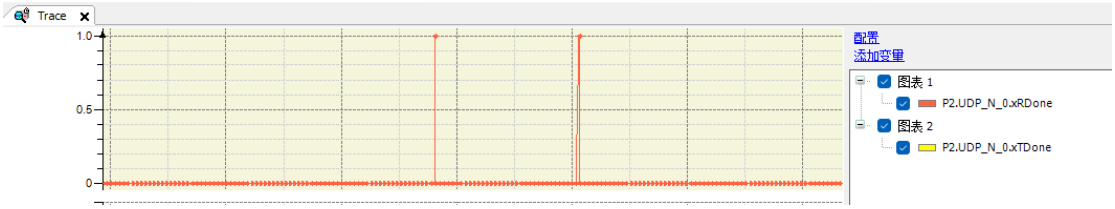
调试工具的设置：



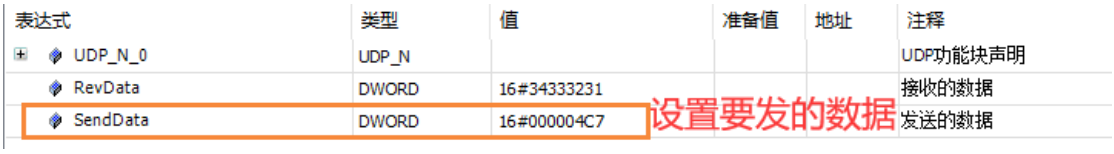
（3）PLC 接收数据：用调试工具发送数据，能看到 RevData 变量收到数据。为了方便查看，在显示模式中设置为数据按照 16 进制显示即可。数据在调试助手中显示是低位在左侧，Codesys 数据中低位在右。 .xRDone 只存在一个周期，可以用 Trace 抓取



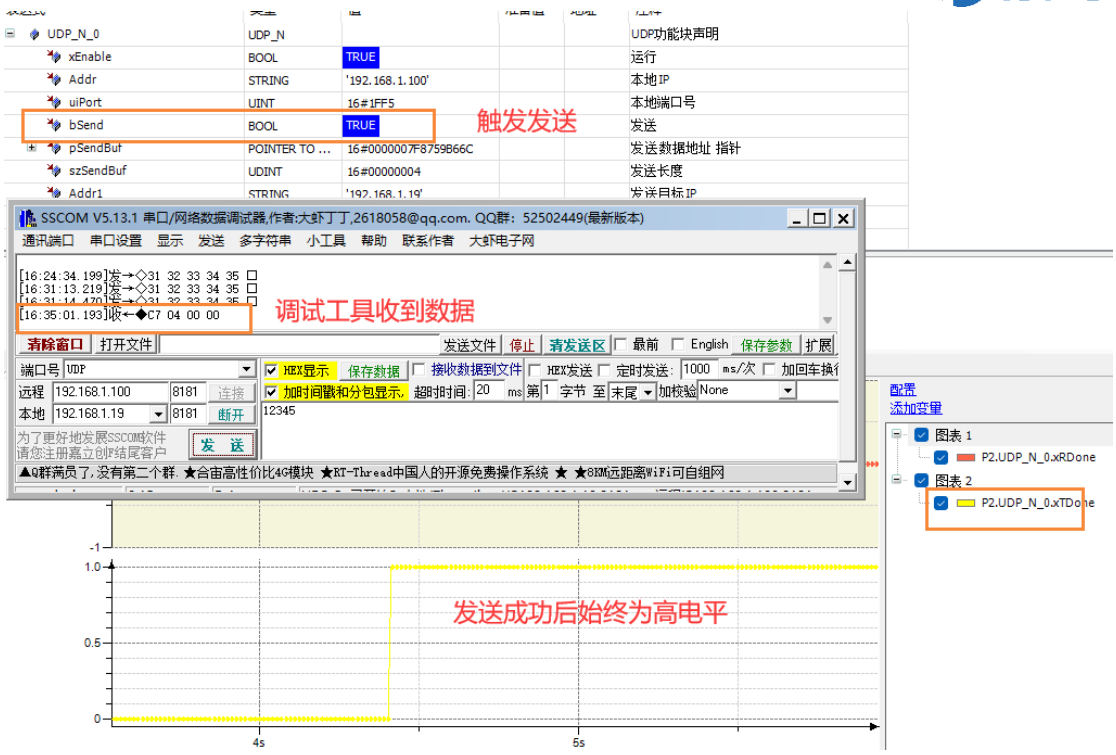
.xRDone 只存在一个周期，可以用 Trace 抓取



(4)PLC 发送数据：用 PLC 发送数据，能在调试工具中看到收到数据。为了方便查看，在显示模式中设置为数据按照 16 进制显示即可。数据在调试助手中显示是低位在左侧，Codesys 数据中低位在右。 .xTDone 发送成功后一直为 1.



通过 bSend 发送

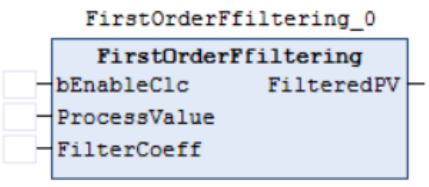


05 滤波类

FirstOrderFfiltering 一阶滤波

功能：一阶滤波
new_value = (1 - a) * last_value + a * IN

1、指令格式
图形表现：



ST 表现：

```
FirstOrderFfiltering(
    bEnableClc:= ,
    ProcessValue:= ,
    FilterCoeff:= ,
    FilteredPV=> );
```

2、相关变量

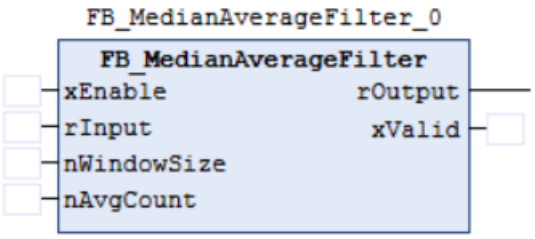
变量名称	功能	数据类型	变量属性	描述
bEnableClc	开启功能	BOOL	输入变量	设置为 TRUE 开启滤波
ProcessValue	输入值	REAL	输入变量	输入值
FilterCoeff	滤波系数	REAL	输入变量	滤波系数越小越强
FilteredPV	过程值滤波	REAL	输出变量	过程值滤波

FB_MedianAverageFilter 中位值平均滤波

用循环缓冲区存储最新的 nWindowSize 个数据。
对缓冲区数据排序，去除两端极端值。
对中间 nAvgCount 个数据求平均，作为输出。

1、指令格式
图形表现：　　ST 表现：

```
FB_MedianAverageFilter(
    xEnable:= ,
    rInput:= ,
    nWindowSize:= ,
    nAvgCount:= ,
    rOutput=> ,
    xValid=> );
```



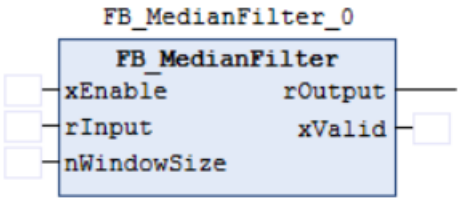
2、相关变量

变量名称	初始值	数据类型	变量属性	描述
xEnable	/	BOOL	输入变量	使能滤波功能
rInput	/	REAL	输入变量	输入原始数据
nWindowSize	5	UINT	输入变量	滤波数组长度（建议设为奇数，如 5、7，需≥3，最大支持 100）。
nAvgCount	3	UINT	输入变量	参与平均的数据个数（需≤窗口大小，如窗口为 5 时可设为 3，即去除最大和最小值后取中间 3 个平均）
rOutput	/	REAL	输出变量	滤波后输出
xValid	/	BOOL	输出变量	输出有效标志（窗口填满后有效）

FB_MedianFilter 中位值滤波

采用循环缓冲区存储最新的 nWindowSize 个数据，新数据会覆盖最旧的数据。
当缓冲区填满后，对数据进行升序排序，取中间位置的值作为输出（奇数窗口直接取正中间，偶数窗口取中间偏左，可根据需求调整）。

1、指令格式
图形表现：　　ST 表现：




```
MedianFilter(
    xEnable:= ,
    rInput:= ,
    nWindowSize:= ,
    rOutput=> ,
    xValid=> );
```

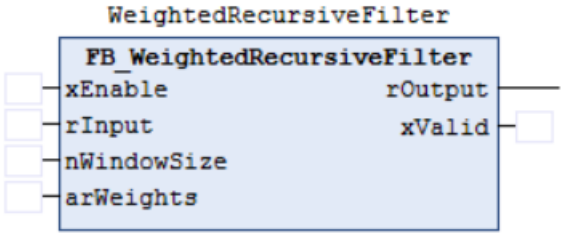
2、相关变量

变量名称	初始值	数据类型	变量属性	描述
xEnable	/	BOOL	输入变量	使能滤波功能
rInput	/	REAL	输入变量	输入原始数据
nWindowSize	5	UINT	输入变量	滤波数组长度（建议设为奇数，如 3、5、7，需≥3，默认 5；最大支持 100，可通过修改数组长度扩展）。
rOutput	/	REAL	输出变量	滤波后的结果（窗口填满后为中位值，否则为原始输入）
xValid	/	BOOL	输出变量	输出有效标志（窗口填满后有效）

FB_WeightedRecursiveFilter 加权递推平均滤波法

采用循环缓冲区存储最新的 nWindowSize 个数据，新数据覆盖最旧数据。
权重数组 arWeights 需用户提前配置（例如窗口大小为 5 时，可设为[0.1, 0.1, 0.2, 0.3, 0.3]，最新数据权重最高）。
当缓冲区填满后，按权重对数据加权求和，结果作为输出。

1、指令格式
图形表现：



ST 表现：

```
WeightedRecursiveFilter(
    xEnable:= ,
    rInput:= ,
    nWindowSize:= ,
    arWeights:= ,
    rOutput=> ,
    xValid=> );
```

2、相关变量

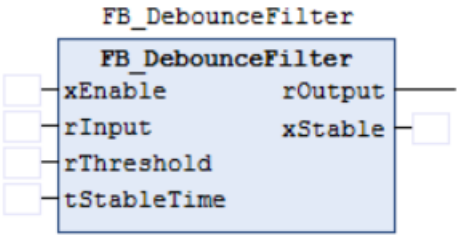
变量名称	初始值	数据类型	变量属性	描述
xEnable	/	BOOL	输入变量	使能滤波功能
rInput	/	REAL	输入变量	输入原始数据
nWindowSize	5	UINT	输入变量	滤波数组长度（≥1，默认 5，最大支持 100，可通过修改数组长度扩展）。
arWeights	/	ARRAY[1..1	输入变量	权重数组（长度需与窗口大小一致，建议权重

		00] OF REAL		总和为 1.0，最新数据权重通常最大）。
rOutput	/	REAL	输出变量	滤波后的加权平均值（窗口填满后有效）
xValid	/	BOOL	输出变量	输出有效标志（窗口填满后有效）

FB_DebounceFilter 消抖滤波法

通过比较当前输入与上一周期输入的差值（绝对值）判断是否抖动。
若差值在阈值内，启动计时；持续稳定时间达到 tStableTime 后，输出当前值并标记稳定。
若差值超过阈值，视为抖动，重置计时，保持上一次稳定输出。

1、指令格式
图形表现：



ST 表现：

```
FB_DebounceFilter(
    xEnable:= ,
    rInput:= ,
    rThreshold:= ,
    tStableTime:= ,
    rOutput=> ,
    xStable=> );
```

2、相关变量

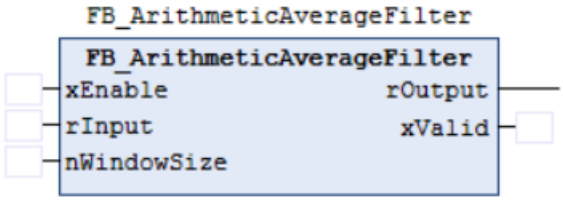
变量名称	初始值	数据类型	变量属性	描述
xEnable	/	BOOL	输入变量	使能滤波功能
rInput	/	REAL	输入变量	原始输入数据（如模拟量传感器信号、经 AD 转换的开关量信号）
rThreshold	0.5	REAL	输入变量	抖动阈值（信号变化量的允许范围，小于此值视为“未抖动”，默认 0.5）
tStableTime	T#100MS	TIME	输入变量	稳定时间（信号需持续稳定的时长，达到此时长才更新输出，默认 100ms）
rOutput	/	REAL	输出变量	滤波后输出值（仅在信号稳定后更新）
xValid	/	BOOL	输出变量	输出有效标志（窗口填满后有效）

FB_ArithmeticAverageFilter 算术平均滤波

采用循环缓冲区存储最新数据，新数据会覆盖最旧的数据（先进先出）。
通过维护一个“数据总和”变量 rSum，避免每次计算平均值时遍历整个缓冲区，提高效率

（窗口满时，rSum = 旧总和 - 被覆盖的旧数据 + 新数据）。
窗口填满后，平均值 = 总和 / 窗口大小，输出有效；未填满时，输出原始数据（可根据需求改为输出当前已有数据的平均值）。

1、指令格式
图形表现：



ST 表现：

```
FB_ArithmeticAverageFilter(
    xEnable:= ,
    rInput:= ,
    nWindowSize:= ,
    rOutput=> ,
    xValid=> );
```

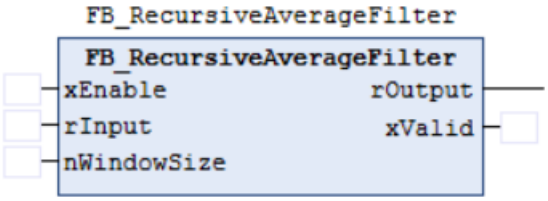
2、相关变量

变量名称	初始值	数据类型	变量属性	描述
xEnable	/	BOOL	输入变量	使能滤波功能
rInput	/	REAL	输入变量	输入原始数据
nWindowSize	10	UINT	输入变量	滤波数组长度（≥1，最大支持 100）
rOutput	/	REAL	输出变量	滤波后的平均值（窗口填满后有效，否则为原始输入）
xValid	/	BOOL	输出变量	输出有效标志（缓冲区数据个数达到窗口大小时为 TRUE）

FB_RecursiveAverageFilter 递推平均滤波法

采用循环缓冲区实现滑动窗口，新数据按顺序存储，超过窗口大小时自动覆盖最旧数据。
通过变量 rSum 维护窗口内数据的总和：缓冲区未满时直接累加新数据；缓冲区已满时，用“新数据 - 被覆盖的旧数据”更新总和，避免重复计算。
窗口填满后，平均值 = 总和 / 窗口大小，输出有效；未填满时，输出原始数据（可根据需求改为输出当前已有数据的平均值）。

1、指令格式
图形表现：



ST 表现：

```
FB_RecursiveAverageFilter(
    xEnable:= ,
    rInput:= ,
    nWindowSize:= ,
    rOutput=> ,
    xValid=> );
```

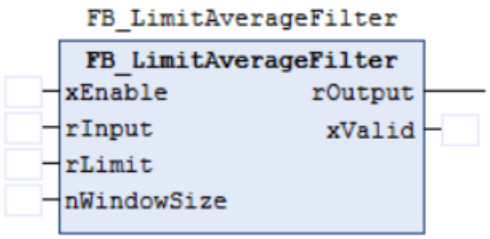
2、相关变量

变量名称	初始值	数据类型	变量属性	描述
xEnable	/	BOOL	输入变量	使能滤波功能
rInput	/	REAL	输入变量	输入原始数据
nWindowSize	5	UINT	输入变量	滑动窗口大小（≥1，默认 5，最大支持 100，可通过修改数组长度扩展）
rOutput	/	REAL	输出变量	滤波后的滑动平均值（窗口填满后有效，否则为原始输入）
xValid	/	BOOL	输出变量	输出有效标志（窗口填满后有效）

FB_LimitAverageFilter 限幅平均滤波法

首次数据直接视为有效，用于初始化基准值。
非首次数据通过与上一次有效数据的差值判断是否超限：超限则用旧值替代，未超限则更新基准值。
有效数据存入循环缓冲区，通过维护总和 rSum 快速计算平均值（窗口满时，总和 = 旧总和 - 被覆盖的旧数据 + 新数据）。

1、指令格式
图形表现：



ST 表现：

```
FB_LimitAverageFilter(
    xEnable:= ,
    rInput:= ,
    rLimit:= ,
    nWindowSize:= ,
    rOutput=> ,
    xValid=> );
```

2、相关变量

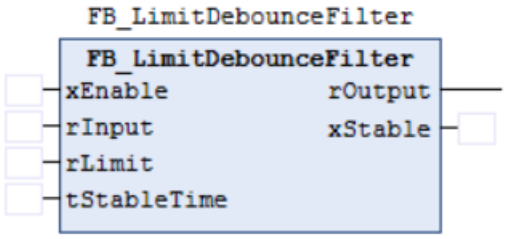
变量名称	初始值	数据类型	变量属性	描述
xEnable	/	BOOL	输入变量	使能滤波功能
rInput	/	REAL	输入变量	输入原始数据
rLimit	1.0	REAL	输入变量	限幅阈值（相邻两次数据的最大允许变化量，超过此值视为突变，默认 1.0）。
nWindowSize	T#100MS	TIME	输入变量	平均窗口大小（参与平均的有效数据个数，≥1，默认 5，最大 100）
rOutput	/	REAL	输出变量	滤波后结果（窗口填满后为有效数据的平均值，否则为当前有效数据）。
xValid	/	BOOL	输出变量	输出有效标志（窗口填满有效数据后为 TRUE）

FB_LimitDebounceFilter 限幅消抖滤波法

首次数据直接作为基准值，避免初始化阶段的误判。
限幅处理确保只有在合理范围内的变化才被视为有效，突变信号被强制替换为上一次有效数据。
消抖处理通过计时判断：当有效数据发生变化时，启动计时；若能在设定时间内保持稳定，则更新输出并标记有效；若期间再次变化，则重置计时。

1、指令格式

图形表现：



ST 表现：

```
FB_LimitDebounceFilter(
    xEnable:= ,
    rInput:= ,
    rLimit:= ,
    tStableTime:= ,
    rOutput=> ,
    xStable=> );
```

2、相关变量

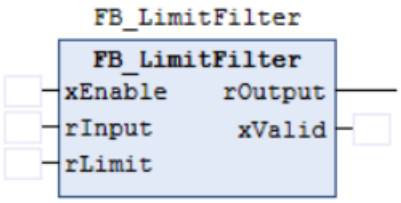
变量名称	初始值	数据类型	变量属性	描述
xEnable	/	BOOL	输入变量	使能滤波功能
rInput	/	REAL	输入变量	输入原始数据
rLimit	1.0	REAL	输入变量	限幅阈值（相邻两次数据的最大允许变化量，超过此值视为突变，默认 1.0）。
tStableTime	T#100MS	TIME	输入变量	稳定时间（限幅后的信号需持续稳定的时长，达标后才更新输出，默认 100ms）。
rOutput	/	REAL	输出变量	滤波后结果（窗口填满后为有效数据的平均值，否则为当前有效数据）。
xValid	/	BOOL	输出变量	输出有效标志（窗口填满有效数据后为 TRUE）

FB_LimitFilter 限幅滤波

首次数据直接作为基准值，无需限幅判断，避免初始化阶段的偏差。
非首次数据通过绝对值差值判断是否超限：未超限则更新基准值和输出，超限则保持基准值不变，输出不更新。
使能关闭时，所有状态重置，再次使能需重新初始化。

1、指令格式

图形表现：



ST 表现：

```
FB_LimitFilter(
    xEnable:= ,
    rInput:= ,
    rLimit:= ,
    rOutput=> ,
    xValid=> );
```

2、相关变量

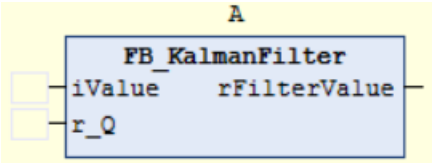
变量名称	初始值	数据类型	变量属性	描述
xEnable	/	BOOL	输入变量	使能滤波功能
rInput	/	REAL	输入变量	输入原始数据
rLimit	1.0	REAL	输入变量	限幅阈值（相邻两次有效数据的最大允许变化量，超过此值视为突变，默认 1.0）。
rOutput	/	REAL	输出变量	滤波后结果（窗口填满后为有效数据的平均值，否则为当前有效数据）。
xValid	/	BOOL	输出变量	输出有效标志（窗口填满有效数据后为 TRUE）

FB_KalmanFilter 卡尔曼滤波

功能块说明：本功能块基于卡尔曼滤波原理进行实现，可设定调控比例，用于过滤一些杂波。例如转矩控制张力时候，过滤一些突变的力矩值，防止张力突变。

1、指令格式

图形表现：



ST 表现：

```
A(iValue:= ,
  r_Q:= ,
  rFilterValue=> );
```

2、相关变量说明

变量	名称	数据类型	输入输出属性	描述
iValue	需要过滤的数据	INT	输入变量	需要滤波的数据
r_Q	调节比例	REAL	输入变量	过滤的比例系数，约接近 0 过滤效果越好
rFilterValue	REAL	REAL	输出变量	滤波处理后的数值

3、具体使用介绍：

如下，产生 0-10 的随机数

```
PROGRAM PLC_PRG
VAR
    klm     :FB_KalmanFilter;

    qs          :REAL;
    FilterValue:REAL;

(*  Rndi,i:UINT;
    next:DWORD:=10;
    iInput:UINT;*)

    bintlization:BOOL;
    aa:SM0.RndI_Range;
    dd:BOOL;
    Result:DINT;
END_VAR
```

```
klm(iValue:= DINT_TO_INT(Result), r_Q:= qs, rFilterValue=> FilterValue);
(*
i:=i+1;
IF  (i  MOD 10)=0   THEN
    Rndi:=REAL_TO_BYTE((10*(DWORD_TO_REAL(Rand(Next))/65535)));
END_IF
*)
IF  NOT bintlization    THEN
    dd:=aa.init(udiSeed:=1,nMax:=10,nMin:=0);
    bintlization:=TRUE;
END_IF
aa.Advance();
Result:=aa.GetCur();
```

产生随机数

采集波形，蓝色是原始的随机数值，绿色的是滤波后的数值。当前的设定比例是 0.1



修改比例到 0.000001（越接近 0），波形产生变化，杂波过滤后越接近 5（随机数 0-10）



06 PID

FB_SelfTuningIncrementalPID 自整定 PID

待发

10 程序管理类

101 产品信息与校验

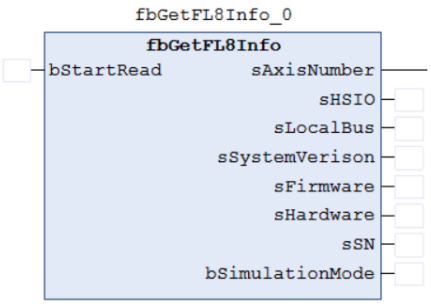
FC_FlexemPLC 判断是否是繁易 PLC

读取产品 ID 判断当前控制器是不是繁易 PLC;
TRUE: 仿真 或者是 FlexemID; FALSE: 非 FlexemID
可加在程序的最上方，判断若不是繁易的总线 PLC，则直接 Return 不执行其他程序。

```
IF NOT FC_FlexmPLC() THEN
    RETURN;
END_IF
```

fbGetFL8Info 获取 FL8 PLC 信息

功能块说明：用本功能块获取 PLC 的一些系统信息。
1、指令格式
图形表现：



ST 表现：

```
fbGetPlcInfo(
    bStartRead:= ,
    sAxisNumber=> ,
    sHSIO=> ,
    sLocalBus=> ,
    sSystemVerison=> ,
    sFirmware=> ,
    sHardware=> ,
    sSN=> ,
    bSimulationMode=> );
```

2、相关变量

变量名称	名称	数据类型	变量属性	描述
bStartRead	读取	BOOL	输入变量	=True 读取，读取完成后自动恢复成 False

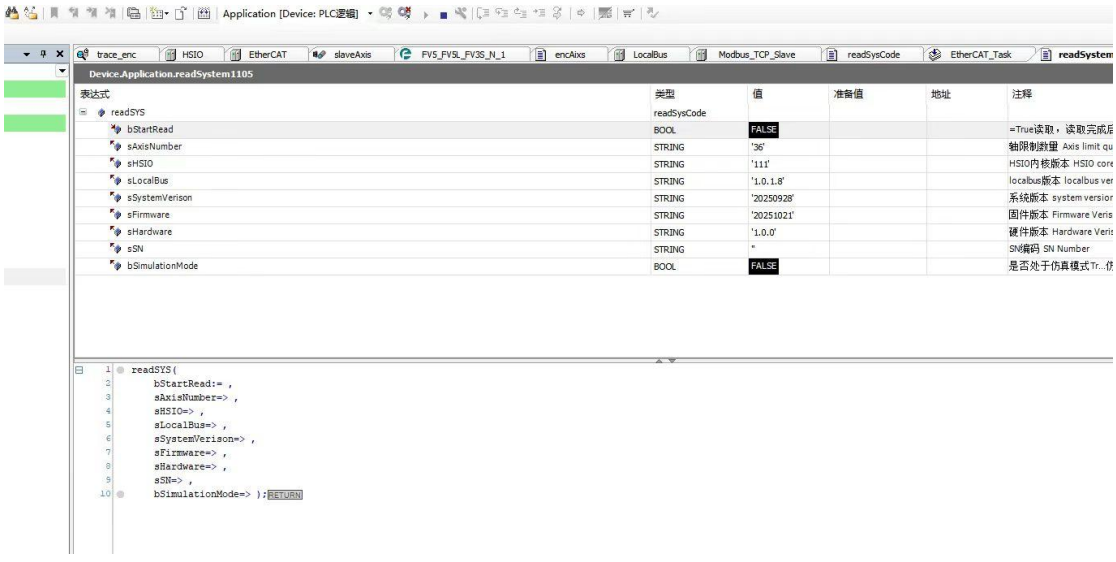
sAxisNumber	轴限制数量	STRING	输出变量	轴限制数量
sHSIO	HSIO 内核版本	STRING	输出变量	HSIO 内核版本
sLocalBus	localbus 版本	STRING	输出变量	localbus 版本
sSystemVerison	系统版本	STRING	输出变量	系统版本
sFirmware	固件版本	STRING	输出变量	固件版本
sHardware	硬件版本	STRING	输出变量	硬件版本
sSN	SN 编码	STRING	输出变量	SN 编码
bSimulationMode	PLC 是否处于仿真模式运行	BOOL	输出变量	是否处于仿真模式 True 为仿真模式, False 为在实际 PLC 中运行

3、具体使用

声明和调用如下所示，在运行的时候对 bStartRead 触发 TRUE；则能正常读取出 PLC 的信息。并且 bStartRead 引脚自动恢复成 FALSE。若运行环境处于仿真模式，则 bSimulationMode=TRUE。

```
PROGRAM PLC_PRG
VAR
    fbGetPlcInfo:fbGetFL8Info;
END_VAR
```

```
fbGetPlcInfo(
    bStartRead:= ,
    sAxisNumber=> ,
    sHSIO=> ,
    sLocalBus=> ,
    sSystemVerison=> ,
    sFirmware=> ,
    sHardware=> ,
    sSN=> ,
    bSimulationMode=> );
```

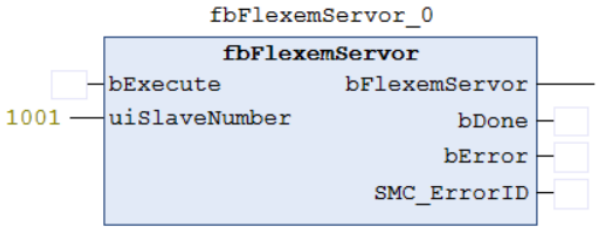


fbFlexemServor 判断是否是繁易伺服

功能块名称：fbFlexemServor　用于判断接的伺服是否是繁易伺服

1、指令格式

图形表现：



ST 表现：

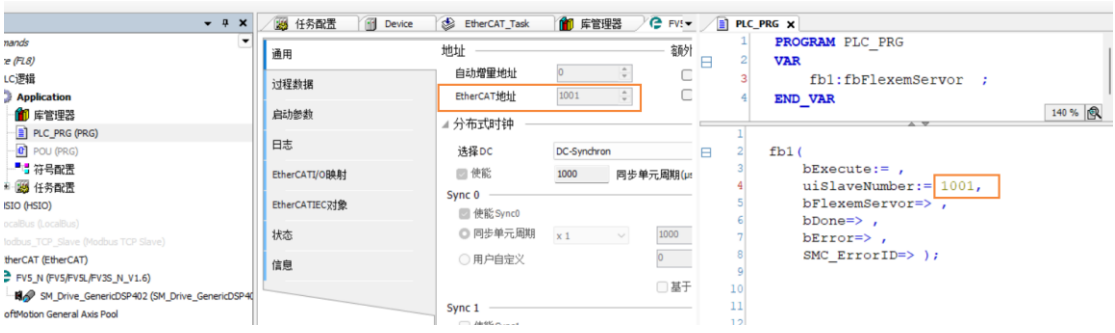
```
fb1(
    bExecute:= ,
    uiSlaveNumber:= ,
    bFlexemServor=> ,
    bDone=> ,
    bError=> ,
    SMC_ErrorID=> );
```

2、相关变量

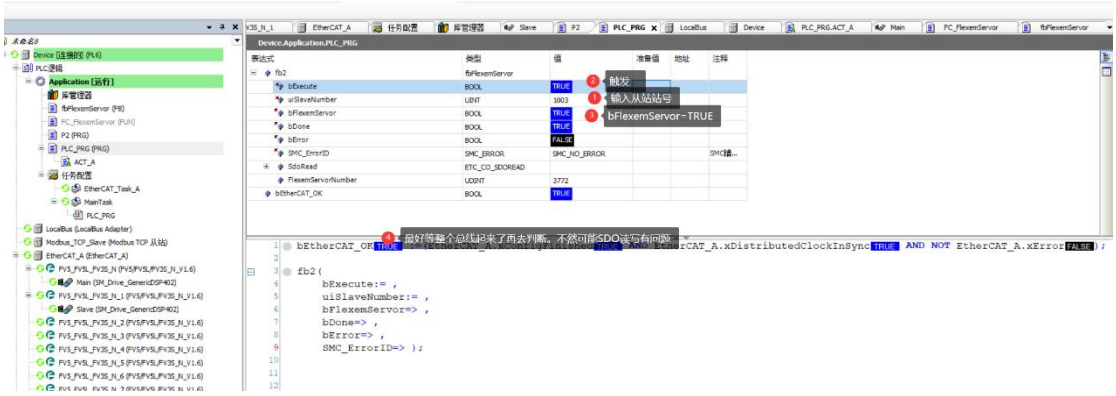
变量名称	名称	数据类型	变量属性	描述
bExecute	触发功能块	BOOL	输入变量	触发判断，上升沿有效
uiSlaveNumber	从站站号	UINT	输入变量	从站站号
bFlexemServor	是否是繁易伺服	BOOL	输出变量	=TRUE，则表示是繁易伺服或者 PLC 处于仿真模式。=FALSE 则不是繁易的伺服
bDone	读取完成	BOOL	输出变量	读取完成
bError	报错	BOOL	输出变量	读取失败报错
SMC_ErrorID	报错 ID	SM3_Basic.SMC_ERROR	输出变量	报错 ID

3、具体使用

声明和调用如下所示。其中，从站站号来自于从站的 EtherCat 地址设置。



另外，由于本功能块使用的是 SDO 的读写，整个判断最好是等总线都运行起来，并且没有什么问题的时候再去做判断。当判断为繁易伺服时候，bFlexemServor=True；否则为 FALSE。此外，当运行在仿真下的时候，bFlexemServor=TRUE;



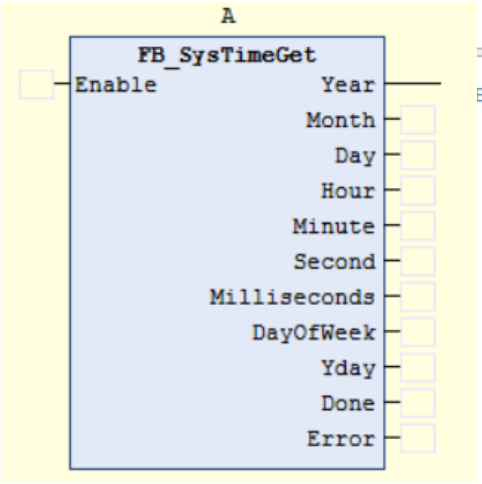
102 时间相关

FB_SysTimeGet 时钟获取

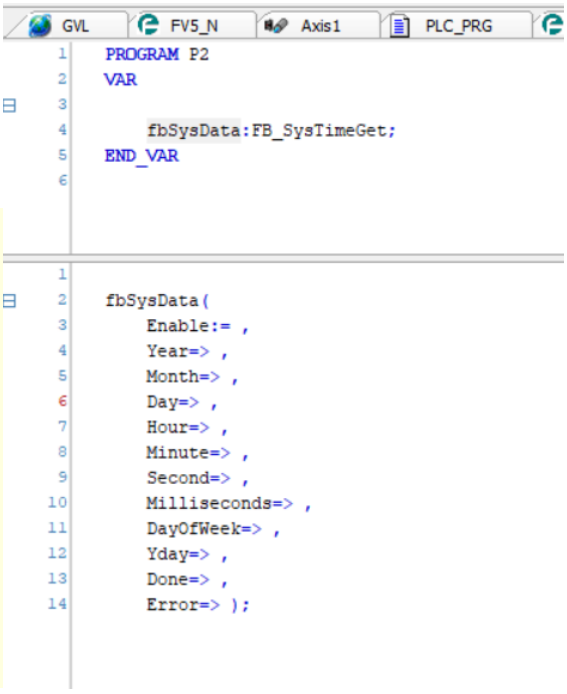
功能块说明：获取时钟。拆分成年月日时分秒毫秒，星期几，一年中的第几天。

1、指令格式

图形表现



ST 表现



2、相关变量说明

变量名称	名称	数据类型	变量属性	描述
Enable	使能功能块	BOOL	输入变量	需要记录的变量
Year	年	UINT	输出变量	输出年份；例如：2020 年

Month	月	UINT	输出变量	输出月份；1-12 月
Day	日	UINT	输出变量	输出日期；例如：第 31 天
Hour	小时	UINT	输出变量	输出小时；0-23
Minute	分钟	UINT	输出变量	分钟
Second	秒	UINT	输出变量	秒
Milliseconds	毫秒	UINT	输出变量	毫秒
DayOfWeek	星期几	UINT	输出变量	星期几
YDay	一年中第几天	UINT	输出变量	一年中的第几天
Done	完成	BOOL	输出变量	功能块完成信号
Error	报错	BOOL	输出变量	功能块报错信号

3、调用举例：

Device.Application.P2

表达式	类型	值	准备值	地址
fbSysData	FB_SysTimeGet			
Enable	BOOL	TRUE		
Year	UINT	2024		
Month	UINT	10		
Day	UINT	24		
Hour	UINT	16		
Minute	UINT	2		
Second	UINT	23		
Milliseconds	UINT	0		
DayOfWeek	UINT	4		
Yday	UINT	298		
Done	BOOL	TRUE		
Error	BOOL	FALSE		

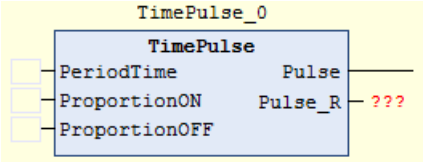
```
1
2  fbSysData(
3      Enable:= ,
4      Year=> ,
5      Month=> ,
6      Day=> ,
7      Hour=> ,
8      Minute=> ,
9      Second=> ,
10     Milliseconds=> ,
11     DayOfWeek=> ,
12     Yday=> ,
13     Done=> ,
14     Error=> );RETURN
```

TimePulse 可变占空比脉冲

功能块说明：发出定时脉冲，并且可以设置占空比。

1、指令格式

图形表现



ST 表现

```
fbTimePulse(
    PeriodTime:= ,
    ProportionON:= ,
    ProportionOFF:= ,
    Pulse=> ,
    Pulse_R=> );
```

2、相关变量说明

变量名称	名称	数据类型	变量属性	描述
PeriodTime	周期时间	Time	输入变量	一个周期的时间设定
ProportionON	脉冲为 ON 的占比	UINT	输入变量	脉冲为 ON 的占比
ProportionOFF	脉冲为 OFF 的占比	UINT	输入变量	脉冲为 OFF 的占比
Pulse	输出脉冲	BOOL	输出变量	脉冲输出
Pulse_R	脉冲输出的上升沿	BOOL	输出变量	脉冲输出的上升沿

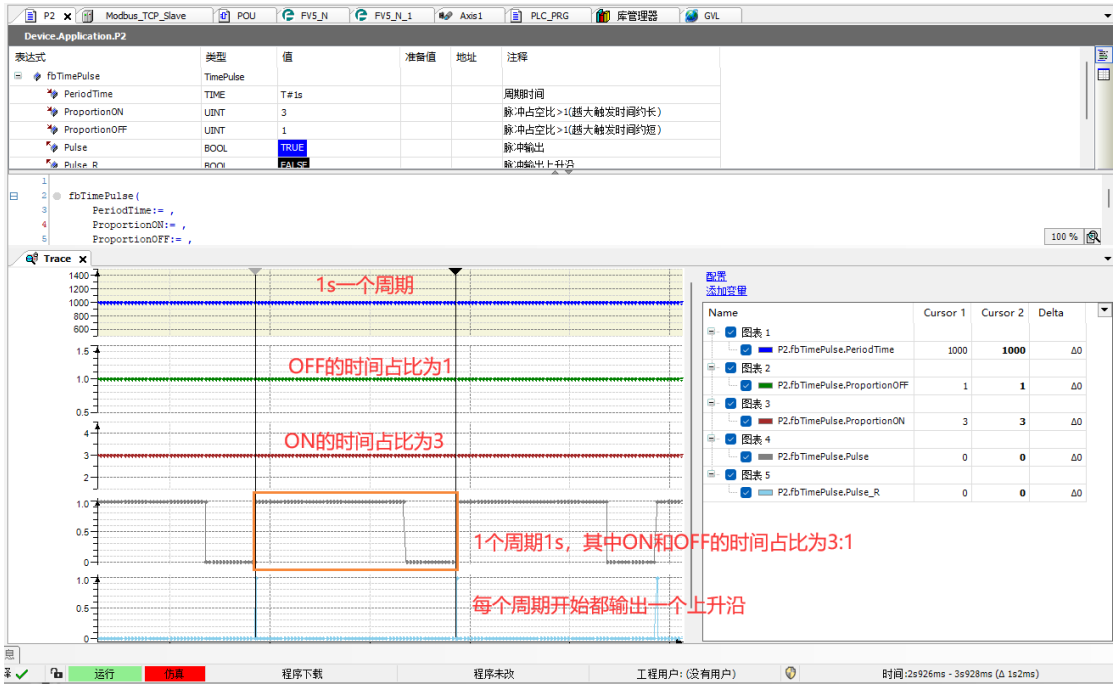
3、调用举例:

（1）声明与实例:

```
PROGRAM P2
VAR
    fbTimePulse:TimePulse;
END_VAR
```

```
fbTimePulse(
    PeriodTime:= ,
    ProportionON:= ,
    ProportionOFF:= ,
    Pulse=> ,
    Pulse_R=> );
```

（2）写值与监控



103 动态分期付款

DynamicEncryption

功能块解释说明：用于客户催款程序，动态分期付款。

1、关于库：需要 SysTimeRtc 库 、需要 Random 库。

2、关于解锁：1）采用产商码+随机 ID+目标解锁天数结合；2）需要使用配套的 Excel 表格解锁；

3、关于随机 ID：1）随机 ID 有防止重复功能；2）生成的随机 ID 为 32 位 16 进制表示；

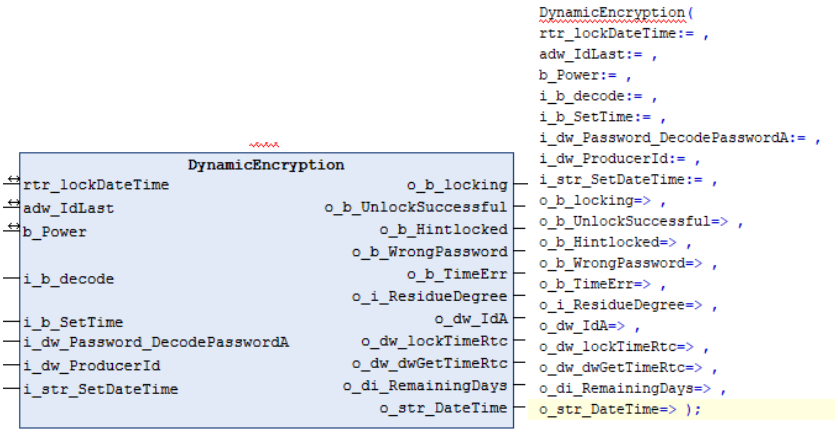
4、关于厂商码：1）厂商码为 32 位 16 进制表示，但只采用低 12 位。

5、关于时间：1）防止时间倒退，时间倒退输出标志锁机，输入任意期密码解锁。

6、关于解锁天数：1）目标天数为 0 时候，永久解锁；2）时间锁机日期为 PLC 日期+设定天数；3）PLC 时间可以修改；4）最大解锁天数为 999；

1、指令格式

图形表现 ST 表现



2、功能块变量说明

（1）输入输出变量（Var In_Out）

变量名称	数据类型	默认值	有效范围	描述
rtr_lockDateTime	SysTimeRtc.RTS_SYSTIMEDATE	/	/	RTS_SYSTIMEDATE 一个时间结构体，锁定时间日期（需要掉电保持）
adw_IdLast	DWORD[20]	/	/	DWORD[20]防止重复 id（需要掉电保持）
b_Power	BOOL	FALSE	/	解锁按钮(不自复位)

（2）输入变量（VAR_Input）

变量名称	数据类型	默认值	有效范围	描述
i_b_decode	BOOL	FALSE	/	触发执行。True，执行修改时间日期
i_b_SetTime	BOOL	FALSE	/	设定时间
i_dw_Password_DecodePasswordA	DWORD	/	/	输入解锁密码
i_dw_ProducerId	DWORD	/	/	产商码（只采用前 12bit）(用来区分产商，防止同样的程序可以互相解码）（需要掉电保持）
i_str_SetDateTime	ysTimeRtc.RTS_SYSTIMEDATE	/	/	RTS_SYSTIMEDATE 一个时间结构体，设定的时间日期

（3）输出变量（VAR_Output）

输出变量	数据类型	默认值	有效范围	描述
o_b_locking	BOOL	FALSE	/	输出是否锁机中
o_b_UnlockSuccessful	BOOL	FALSE	/	输出解码成功标志
o_b_Hintlocked	BOOL	FALSE	/	输出提示剩余时间小于 4 天
o_b_WrongPassword	BOOL	FALSE	/	输出密码错误
o_b_TimeErr	BOOL	FALSE	/	输出时间倒退，将会锁机

o_i_ResidueDegree	INT	5	/	输出剩余解码次数，0 次将刷新随机 id
o_dw_IdA	DWORD	/	/	输出随机 ID，解码超次数、每次开机刷新
o_dw_lockTimeRtc	DWORD	/	/	输出锁机时间戳
o_dw_dwGetTimeRtc	DWORD	/	/	输出 PLC 当前时间戳
o_di_RemainingDays	DINT	/	/	输出剩余锁机时间
o_str_DateTime	ysTimeRtc.RTS_SYSTIMEDATE	/	/	RTS_SYSTIMEDATE 一个时间结构体，输出当前日期

3、使用样例

（1）声明与调用

Device.Application.PLC_PRG

表达式	类型	值
DynamicEncryptions	DynamicEncryption	
rtr_lockDateTime	REFERENCE TO SysTimeRtc.RTS_SYSTIMEDATE	
adw_IdLast	REFERENCE TO ARRAY [0..19] OF DWORD	
b_Power	REFERENCE TO BOOL	TRUE
i_b_decode	BOOL	FALSE
i_b_SetTime	BOOL	FALSE
i_dw_Password_DecodePasswordA	DWORD	3700350481
i_dw_ProducerId	DWORD	4095
i_str_SetDateTime	SysTimeRtc.RTS_SYSTIMEDATE	
o_b_locking	BOOL	FALSE
o_b_UnlockSuccessful	BOOL	TRUE
o_b_Hintlocked	BOOL	FALSE
o_b_WrongPassword	BOOL	FALSE
o_b_TimeErr	BOOL	FALSE
o_i_ResidueDegree	INT	4
o_dw_IdA	DWORD	1591881527
o_dw_lockTimeRtc	DWORD	1794932307
o_dw_dwGetTimeRtc	DWORD	1763397003
o_di_RemainingDays	DINT	364
o_str_DateTime	SysTimeRtc.RTS_SYSTIMEDATE	

```
//  vdo_m_lockTimeRtc=> ,                                   //锁机时间戳
//  vdo_m_dwGetTimeRtc=> ,                                  //当前时间戳
//  vdo_m_RemainingDays=>           GVL_Encryption.gdo_m_RemainingDays,     //剩余可用天数
//  vStro_m_DateTime=>              GVL_Encryption.gmStr_m_ActDateTime      //当前年月日
// );


DynamicEncryptions(
    rtr_lockDateTime:= GVL_Encryption.gmStr_hm_lockDateTime,
    adw_IdLast:=GVL_Encryption.gd_hm_IdLast ,
    b_Power:= GVL_Encryption.gbi_m_Power,
    i_b_decode:= ,
    i_b_SetTime:= ,
    i_dw_Password_DecodePasswordA:= ,
    i_dw_ProducerId:= ,
    i_str_SetDateTime:= ,
    o_b_locking=> ,
    o_b_UnlockSuccessful=> ,
    o_b_Hintlocked=> ,
    o_b_WrongPassword=> ,
    o_b_TimeErr=> ,
    o_i_ResidueDegree=> ,
    o_dw_IdA=> ,
    o_dw_lockTimeRtc=> ,
    o_dw_dwGetTimeRtc=> ,
    o_di_RemainingDays=> ,
```

（2）功能详细演示部分

在表格上，当前日期为 2025/11/13。

输入 ID 42F1A3AA，输入厂商码 13F1，目标日期为 2026/12/21。

输出解锁天数：403 天，密码为 4EDBE7BA。

下图为表格内容

当前日期:	2025/11/13
输入ID:	5018CCAB
输入厂商码:	13F1
目标日期:	2025/11/22
目标解锁天数:	9
生成时间:	2025/11/12
密码:	9C24C921
天数:	9

下图为 ST 内容(锁机状态)

```
ep_Encryption(
    vioStr_hm_lockDateTime:=GVL_Encryption.gmStr_hm_lockDateTime,
    vioDw_hm_IdLast:= GVL_Encryption.gd_hm_IdLast,
    vioB_m_Power TRUE :=GVL_Encryption.gbi_m_Power TRUE ,
    viB_m_decode FALSE :=GVL_Encryption.gbi_m_decode FALSE ,
    viB_m_SetTime:= ,
    viDw_m_Password_DecodePasswordA 16#9C24C921 := GVL_Encryption.gdi_m_Password_DecodePasswordA 16#9C24C921 ,
    viDw_hm_ProducerId 16#000013F1 := GVL_Encryption.gdi_hm_ProducerId 16#13F1 ,
    viStr_m_SetDateTime:= ,
    voB_m_locking TRUE => GVL_Encryption.gdo_m_locking TRUE ,
    voB_m_UnlockSuccessful=> ,
    voB_m_Hintlocked FALSE => GVL_Encryption.gbo_m_Hintlocked FALSE ,
    voB_m_WrongPassword FALSE =>GVL_Encryption.gbo_m_WrongPassword FALSE ,
    voB_m_TimeErr FALSE =>GVL_Encryption.gbo_m_TimeErr FALSE ,
    voI_m_ResidueDegree 16#0005 => GVL_Encryption.gdo_m_ResidueDegree 16#0005 ,
    voDw_m_IdA 16#5018CCAB =>  GVL_Encryption.gdo_m_IdA 16#5018CCAB ,
    voDw_m_lockTimeRtc=> ,
    voDw_m_dwGetTimeRtc=> ,
    voDi_m_RemainingDays 16#00000000 =>GVL_Encryption.gdo_m_RemainingDays 16#00000000 ,
    voStr_m_DateTime=>  GVL_Encryption.gmStr_m_ActDateTime);
```

下图为 ST 内容(解锁状态)

```
ep_Encryption(
    vioStr_hm_lockDateTime:=GVL_Encryption.gmStr_hm_lockDateTime,
    vioDw_hm_IdLast:= GVL_Encryption.gd_hm_IdLast,
    vioB_m_Power TRUE :=GVL_Encryption.gbi_m_Power TRUE ,
    viB_m_decode FALSE :=GVL_Encryption.gbi_m_decode FALSE ,
    viB_m_SetTime:= ,
    viDw_m_Password_DecodePasswordA 2619656481 := GVL_Encryption.gdi_m_Password_DecodePasswordA 2619656481 ,
    viDw_hm_ProducerId 5105 := GVL_Encryption.gdi_hm_ProducerId 5105 ,
    viStr_m_SetDateTime:= ,
    voB_m_locking FALSE => GVL_Encryption.gdo_m_locking FALSE ,
    voB_m_UnlockSuccessful=> ,
    voB_m_Hintlocked FALSE => GVL_Encryption.gbo_m_Hintlocked FALSE ,
    voB_m_WrongPassword FALSE =>GVL_Encryption.gbo_m_WrongPassword FALSE ,
    voB_m_TimeErr FALSE =>GVL_Encryption.gbo_m_TimeErr FALSE ,
    voI_m_ResidueDegree 5 => GVL_Encryption.gdo_m_ResidueDegree 5 ,
    voDw_m_IdA 1624792242 =>  GVL_Encryption.gdo_m_IdA 1624792242 ,
    voDw_m_lockTimeRtc=> ,
    voDw_m_dwGetTimeRtc=> ,
    voDi_m_RemainingDays 8 =>GVL_Encryption.gdo_m_RemainingDays 8 ,
    voStr_m_DateTime=>  GVL_Encryption.gmStr_m_ActDateTime);
```

表格下载：

动态分期付款(无宏，兼容wps)1.0.0.x

02 通用工艺

00_插补类

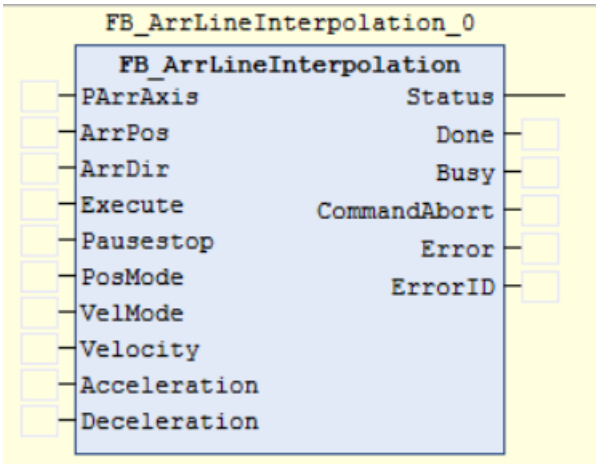
FB_ArrLineInterpolation 多轴直线插补

直线插补。最大支持 32 轴,通过指针数组给轴名字，支持线性以及模态轴。

注意：目前还不支持速度衔接，触发运动该指令时 需要确保轴速度为 0，如果不为 0 速度会从当前速度突变到 0 然后开始正常规划运动

1、指令格式

图形表现：



ST 表现：

```
FB_ArrLineInterpolation_0(
    PArrAxis:= ,
    ArrPos:= ,
    ArrDir:= ,
    Execute:= ,
    Pausestop:= ,
    PosMode:= ,
    VelMode:= ,
    Velocity:= ,
    Acceleration:= ,
    Deceleration:= ,
    Status=> ,
    Done=> ,
    Busy=> ,
    CommandAbort=> ,
    Error=> ,
    ErrorID=> );
```

2、相关变量

变量名	名称	数据类型	输入输出属性	描述
PArrAxis	轴指针数组	ARRAY[*] OF POINTER TO AXIS_REF_ETC_DS402_CS	输入输出变量	轴指针数组
ArrPos	目标位置	ARRAY[*] OF LREAL	输入输出变量	目标位置
ArrDir	方向设定	ARRAY[*] OF INT	输入输出变量	方向，-1：反向，0：最短路径，1：正向，。如果非 -1、0 、1 则认为是最短路径
Execute	运行	BOOL	输入变量	上升沿有效
Pausestop	暂停	BOOL	输入变量	=TRUE 暂停
PosMode	坐标模式设置	BOOL	输入变量	=0 绝对坐标，=1 相对坐标
VelMode	速度模式设置	INT	输入变量	=0 梯形加速，=1 S 型加速
Velocity	速度值	LREAL	输入变量	速度
Acceleration	加速度值	LREAL	输入变量	加速度
Deceleration	加速度值	LREAL	输入变量	减速度
Status	状态输出	INT	输出变量	0:无状态，1:加速,2:匀速，3 减速，4 结束，5 暂停减速中，6 暂停完成
Done	完成	BOOL	输出变量	=TRUE 完成
Busy	运行中	BOOL	输出变量	=TRUE 执行中
CommandAbort	中断执行	BOOL	输出变量	=TRUE 指令中断
Error	报错	BOOL	输出变量	=True 错误
ErrorID	错误代码	INT	输出变量	0 无错误，1 轴名字、位置、方向数组长度不一致或长度超 32 ，2 速度类参数错误，3 轴未准备好，

3、具体使用

（1） 声明与调用

```
PROGRAM PLC_PRG
VAR
    fbLineInter         :FB_ArrLineInterpolation;
    VPArrAxis           :ARRAY[1..3] OF POINTER TO AXIS_REF_ETC_DS402_CS:=[ADR(AxisX),ADR(AxisY),ADR(AxisZ)];//轴指针数组
    VArrPos             :ARRAY[1..3] OF LREAL:=[100,200,300];//目标位置
    VArrDir             :ARRAY[1..3] OF INT;//0 最短路径，1正向，-1反向
    V: LREAL := 100;
    ACC: LREAL := 100;
    DEC: LREAL := 100;
END_VAR
```

```
fbLineInter(
    PArrAxis:= VPArrAxis,
    ArrPos:= VArrPos,
    ArrDir:= VArrDir,
    Execute:= ,
    Pausestop:= ,
    PosMode:= ,
    VelMode:= ,
    Velocity:= V,
    Acceleration:= ACC,
    Deceleration:= DEC,
    Status=> ,
    Done=> ,
    Busy=> ,
    CommandAbort=> ,
    Error=> ,
    ErrorID=> );
```

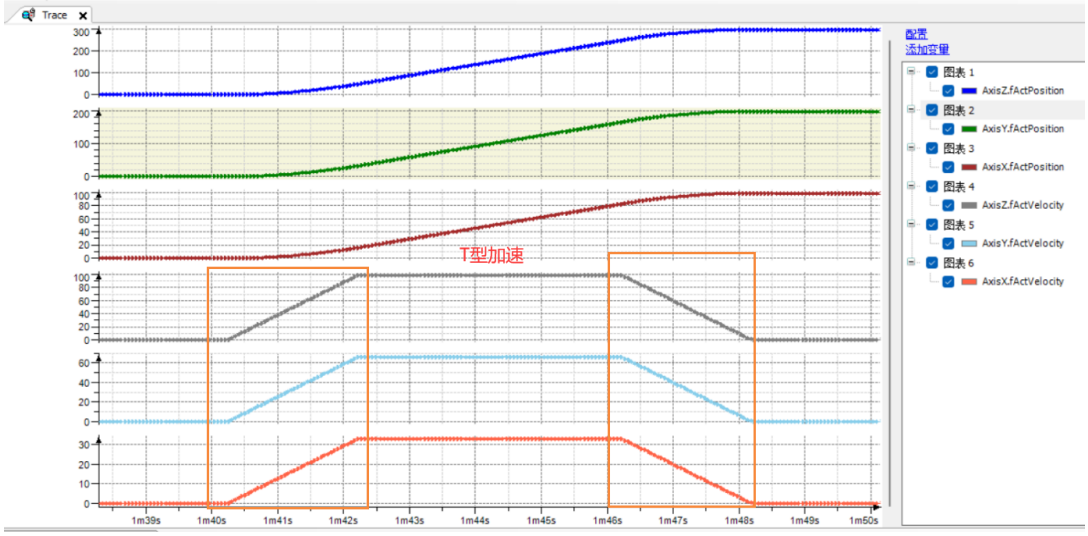
声明数组轴。需要多少个轴同时插补就声明数组长度对应多少

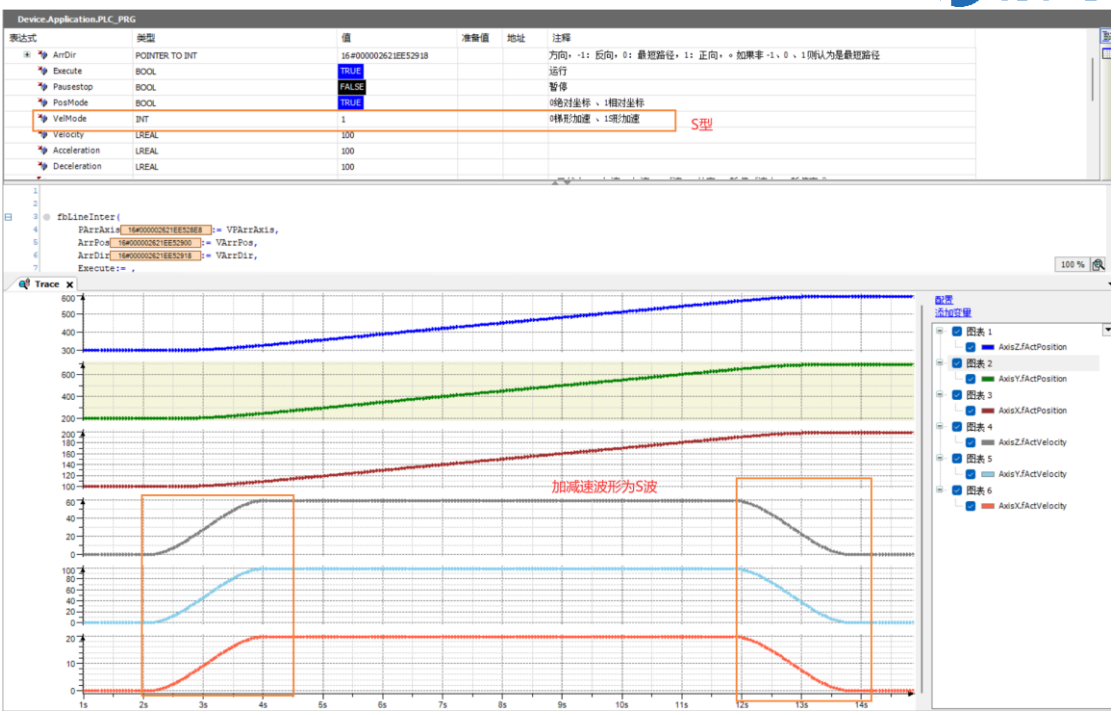
每个轴对应的位置和方向设置

（2） T 型和 S 型加减速模式通过 VelMode 设定，=0 T 型加速、=1S 型加速。波形对比：




```
fbLineInter(
    PArrAxis 16#000002621EE528E8 := VPArrAxis,
    ArrPos 16#000002621EE52900 := VArrPos,
    ArrDir 16#000002621EE52918 := VArrDir,
    Execute:= ,
```



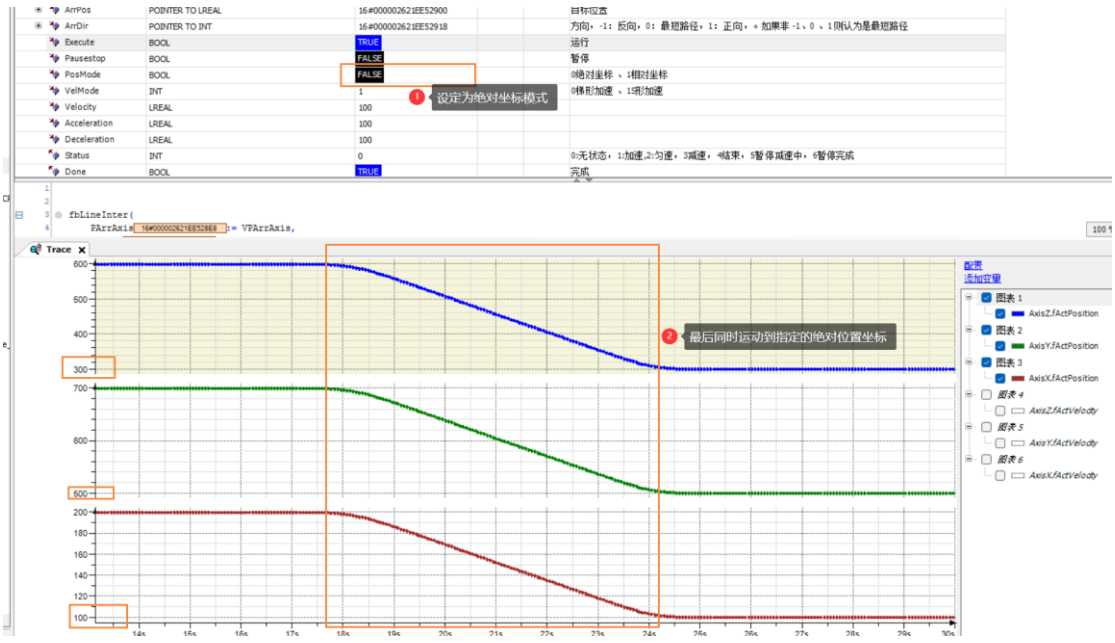


（3） 位置值为相对坐标或绝对坐标。PosMode=0，设定的位置值为绝对坐标；=1，设定的位置值为相对坐标。

相同的参数值，在绝对和相对坐标下的不同表现：

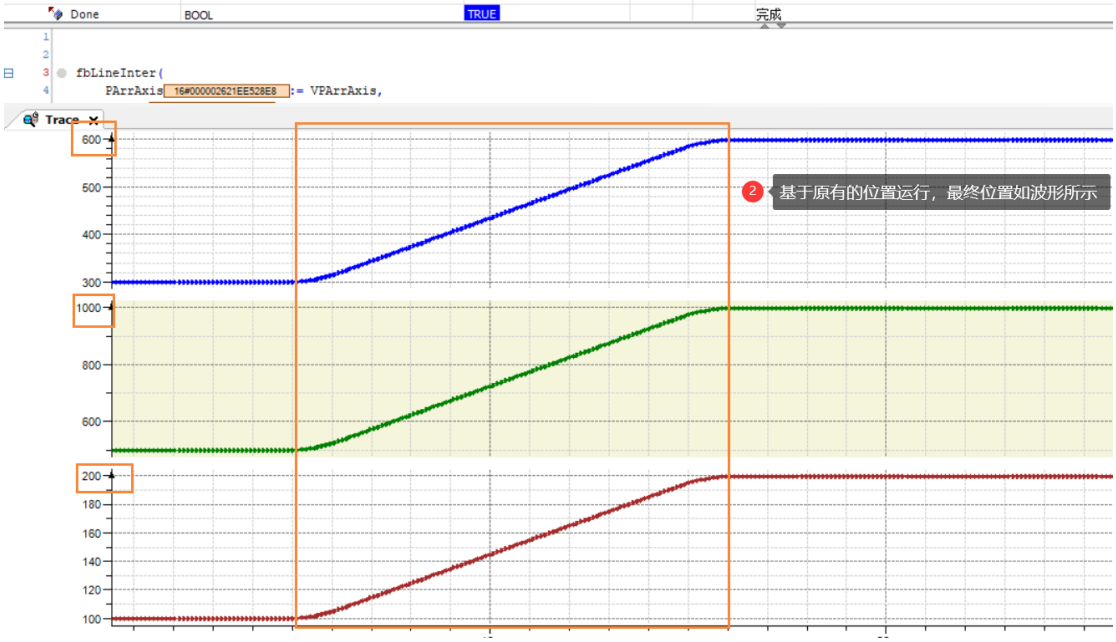
VArrPos	ARRAY [1..3] OF LREAL	
VArrPos[1]	LREAL	100
VArrPos[2]	LREAL	500
VArrPos[3]	LREAL	300

绝对模式：

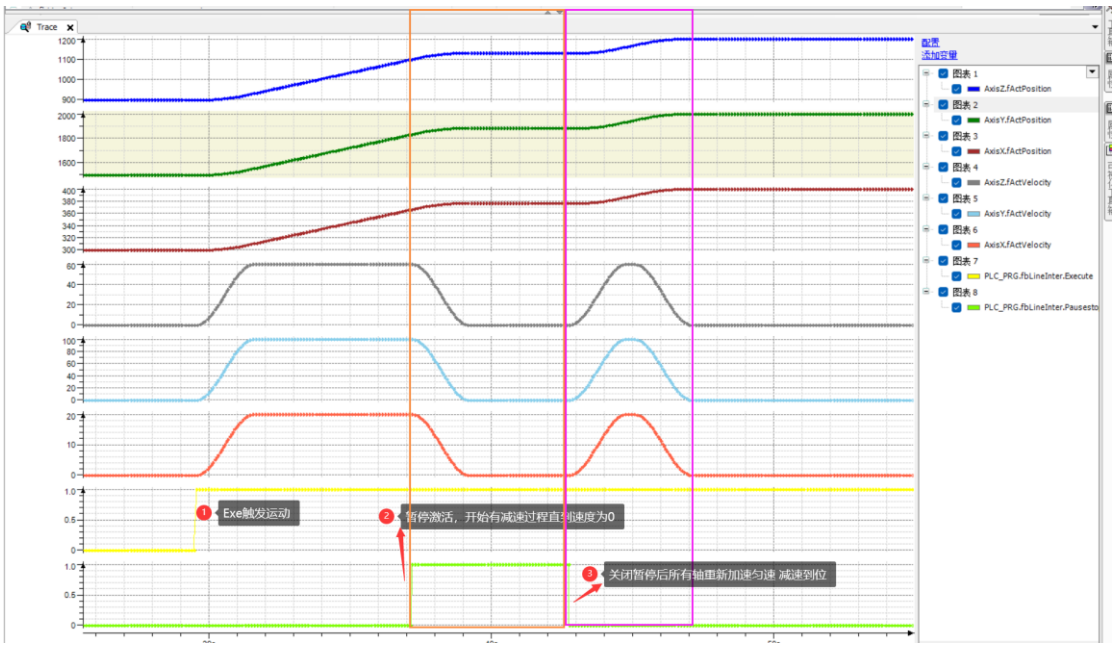


相对模式：

ArrPos	POINTER TO LREAL	16#000002621EE52900	目标位置
ArrDir	POINTER TO INT	16#000002621EE52918	方向，-1：反向，0：最短路径，1：正向，。如果非 -1、0 、1
Execute	BOOL	TRUE	运行
Pausestop	BOOL	FALSE	暂停
PosMode	BOOL	TRUE	0绝对坐标 、1相对坐标
VelMode	INT	1	0梯形加速 、1形加速
Velocity	LREAL	100	
Acceleration	LREAL	100	
Deceleration	LREAL	100	
Status	INT	0	0:无状态，1:加速,2:匀速，3减速，4结束，5暂停减速中，6暂停
Done	BOOL	TRUE	完成



（4） 暂停和暂停后再启动



（5）每个轴运动的路径可以选，是用最短路径还是正反向

```
    VArrPos             :ARRAY[1..3] OF LREAL:=[100,200,300];//目标位置
    VArrDir             :ARRAY[1..3] OF INT;//0 最短路径，1正向，-1反向
    V: LREAL := 100;
    ACC: LREAL := 100;
    DEC: LREAL := 100;
END_VAR
```

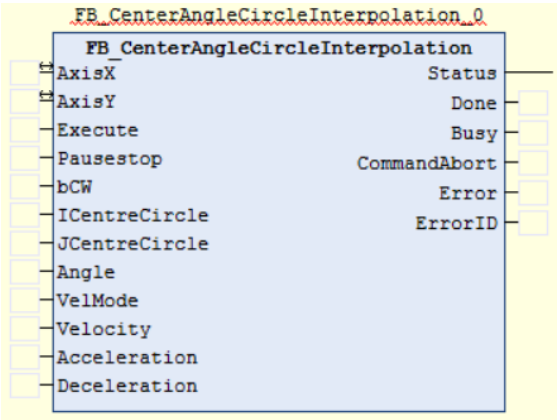
```
fbLineInter(
    PArrAxis:= VPArrAxis,
    ArrPos:= VArrPos,
    ArrDir:= VArrDir,
    Execute:= ,
```

FB_CenterAngleCircleInterpolation 两轴圆弧插补（圆心圆弧）

功能块说明：圆弧插补 ,当前位置为起点、根据圆心、和圆弧角度，规划路径
注意：目前还不支持速度衔接，触发运动该指令时 需要确保轴速度为 0，如果不为 0 速度会从当前速度突变到 0 然后开始正常规划运动

1、指令格式

图形表现：



ST 表现：

```
FB_CenterAngleCircleInterpolation_0(
    AxisX:= ,
    AxisY:= ,
    Execute:= ,
    Pausestop:= ,
    bCW:= ,
    ICentreCircle:= ,
    JCentreCircle:= ,
    Angle:= ,
    VelMode:= ,
    Velocity:= ,
    Acceleration:= ,
    Deceleration:= ,
    Status=> ,
    Done=> ,
    Busy=> ,
    CommandAbort=> ,
    Error=> ,
    ErrorID=> );
```

2、 相关变量

变量名	名称	数据类型	输入输出属性	描述
AxisX	插补轴 X	AXIS_REF_SM3	输入输出变量	插补 X 轴
AxisY	插补轴 Y	AXIS_REF_SM3	输入输出变量	插补 Y 轴
Execute	运行	BOOL	输入变量	上升沿有效

Pausestop	暂停	BOOL	输入变量	=TRUE 暂停
bCw	运动方向	BOOL	输入变量	=0 CCW，=1CW
ICentreCircle	圆心坐标 X	LREAL	输入变量	圆心的 X 坐标
JCentreCircle	圆心坐标 Y	LREAL	输入变量	圆心的 Y 坐标
Angle	圆弧角度	LREAL	输入变量	圆弧的角度，需要设置大于 0
VelMode	速度模式设置	INT	输入变量	=0 梯形加速，=1 S 型加速
Velocity	速度值	LREAL	输入变量	速度
Acceleration	加速度值	LREAL	输入变量	加速度
Deceleration	加速度值	LREAL	输入变量	减速度
Status	状态输出	INT	输出变量	0:无状态，1:加速,2:匀速，3 减速，4 结束，5 暂停减速中，6 暂停完成
Done	完成	BOOL	输出变量	=TRUE 完成
Busy	运行中	BOOL	输出变量	=TRUE 执行中
CommandAbort	中断执行	BOOL	输出变量	=TRUE 指令中断
Error	报错	BOOL	输出变量	=True 错误
ErrorID	错误代码	INT	输出变量	0 无错误， 1 圆弧终点或圆心数据错误 或角度 Angle<=0 2 速度类参数错误 3 轴未准备好

3、详细使用

（1） 声明与实例化

```
PROGRAM POU_2
VAR
    FB_CenterAngleCircleInterpolation_0: FB_CenterAngleCircleInterpolation;
    GetR:REAL;

    ANG: LREAL := 90;
    V: LREAL := 100;
    ACC: LREAL := 100;
    DEC: LREAL := 100;
END_VAR
```

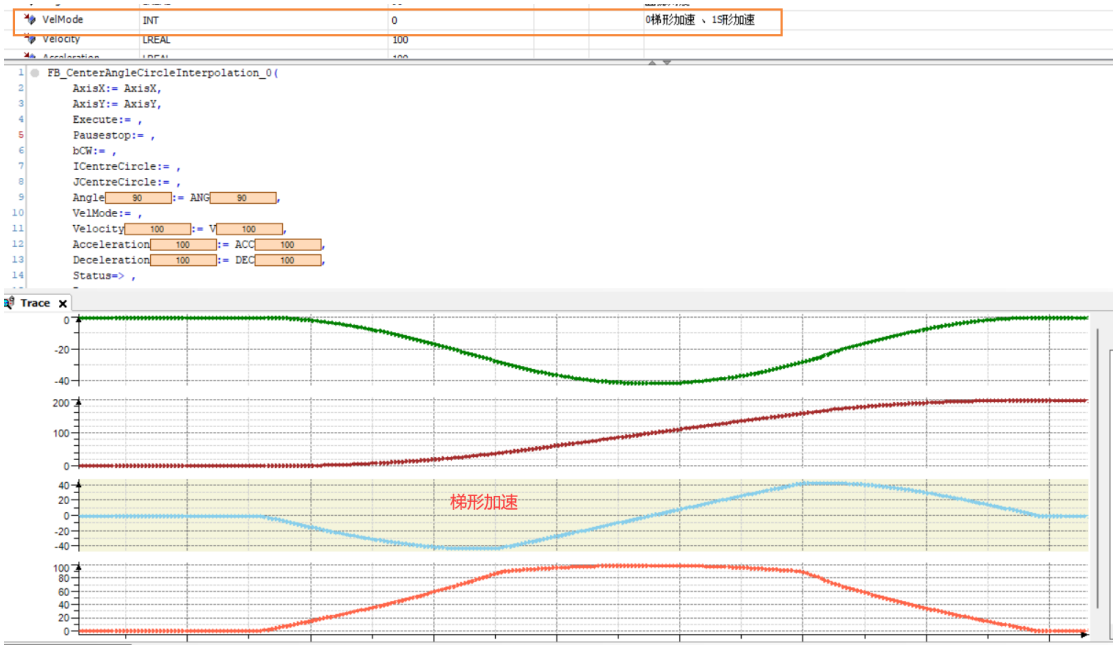
```
FB_CenterAngleCircleInterpolation_0(
    AxisX:= AxisX,
    AxisY:= AxisY,
    Execute:= ,
    Pausestop:= ,
    bCW:= ,
    ICentreCircle:= ,
    JCentreCircle:= ,
    Angle:= ANG,
    VelMode:= ,
    Velocity:= V,
    Acceleration:= ACC,
    Deceleration:= DEC,
    Status=> ,
    Done=> ,
    Busy=> ,
    CommandAbort=> ,
    Error=> ,
    ErrorID=> );
```

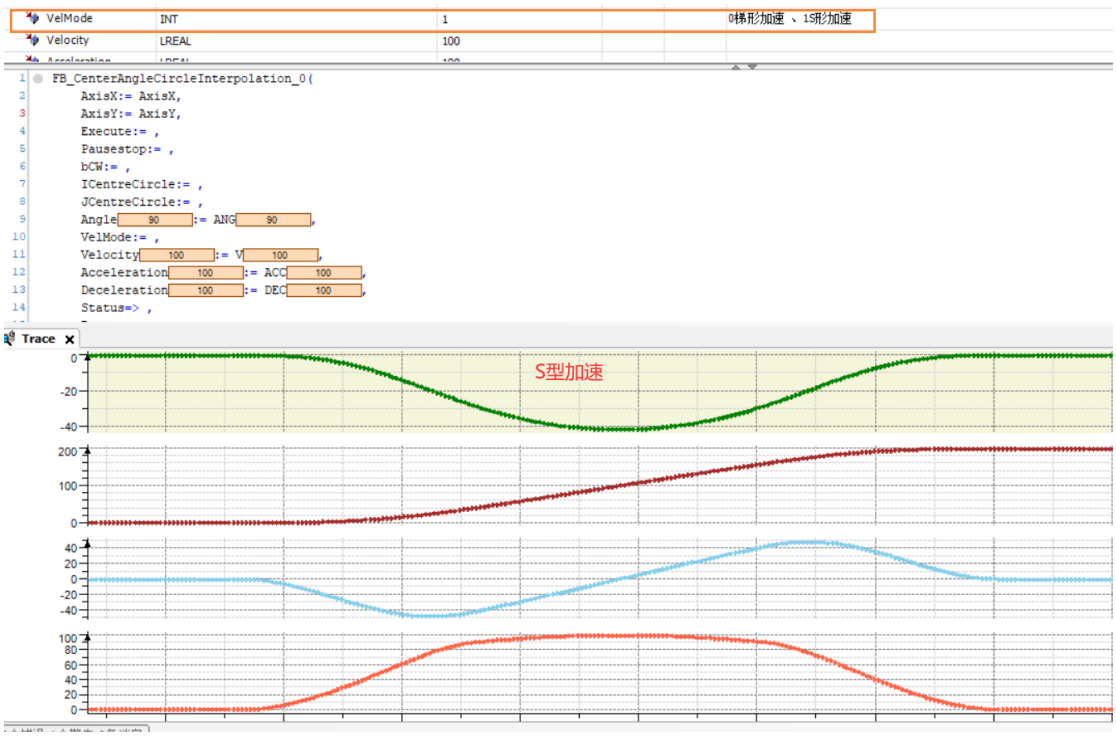
1 设定哪两个轴进行插补运动

2 设定圆心坐标和圆弧角度

3 运行速度设置

（2） 梯形加速和 S 型加速；通过 VelMode 设定。=0 为梯形加速，=1 为 S 型加速。





（3） 圆弧方向 bCW=0 为 CCW 方向，=1 为 CW 方向



（4） 暂停后再启动。使用 Pausestop 暂停和暂停后再次启动。如下所示：



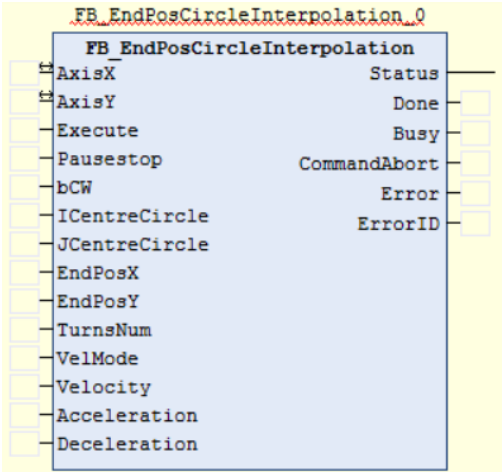
FB_EndPosCircleInterpolation 两轴圆弧插补（圆心、结束位置、圈数）

功能块说明：两轴圆弧插补 ,当前位置为起点，根据圆心、结束位置、圈数，规划路径 。如果 end 位置就在当前位置，则为一圈

注意：目前还不支持速度衔接，触发运动该指令时 需要确保轴速度为 0，如果不为 0 速度会从当前速度突变到 0 然后开始正常规划运动

1、指令格式

图形表现：　　ST 表现：




```
FB_EndPosCircleInterpolation_0(
    AxisX:= ,
    AxisY:= ,
    Execute:= ,
    Pausestop:= ,
    bCW:= ,
    ICentreCircle:= ,
    JCentreCircle:= ,
    EndPosX:= ,
    EndPosY:= ,
    TurnsNum:= ,
    VelMode:= ,
    Velocity:= ,
    Acceleration:= ,
    Deceleration:= ,
    Status=> ,
    Done=> ,
    Busy=> ,
    CommandAbort=> ,
    Error=> ,
    ErrorID=> );
```

2、相关变量

变量名	名称	数据类型	输入输出属性	描述
AxisX	插补轴 X	AXIS_REF_SM3	输入输出变量	插补轴 X
AxisY	插补轴 Y	AXIS_REF_SM3	输入输出变量	插补轴 Y
Execute	运行	BOOL	输入变量	上升沿有效
Pausestop	暂停	BOOL	输入变量	=TRUE 暂停
bCw	运动方向	BOOL	输入变量	=0 CCW，=1CW
ICentreCircle	圆心坐标 X	LREAL	输入变量	圆心的 X 坐标
JCentreCircle	圆心坐标 Y	LREAL	输入变量	圆心的 Y 坐标
EndPosX	结束点 X 坐标	LREAL	输入变量	结束点 X 坐标
EndPosY	结束点 Y 坐标	LREAL	输入变量	结束点 Y 坐标
TurnsNum	圆弧圈数	UINT	输入变量	圆弧圈数
VelMode	速度模式设置	INT	输入变量	=0 梯形加速，=1 S 型加速
Velocity	速度值	LREAL	输入变量	速度
Acceleration	加速度值	LREAL	输入变量	加速度
Deceleration	加速度值	LREAL	输入变量	减速度
Status	状态输出	INT	输出变量	0:无状态，1:加速,2:匀速，3 减速，4 结束，5 暂停减速中，6 暂停完成
Done	完成	BOOL	输出变量	=TRUE 完成
Busy	运行中	BOOL	输出变量	=TRUE 执行中
CommandAbort	中断执行	BOOL	输出变量	=TRUE 指令中断
Error	报错	BOOL	输出变量	=TRUE 错误
ErrorID	错误代码	INT	输出变量	0 无错误，

				1 圆弧终点或圆心数据错误 2 速度类参数错误 3 轴未准备好

3、具体使用

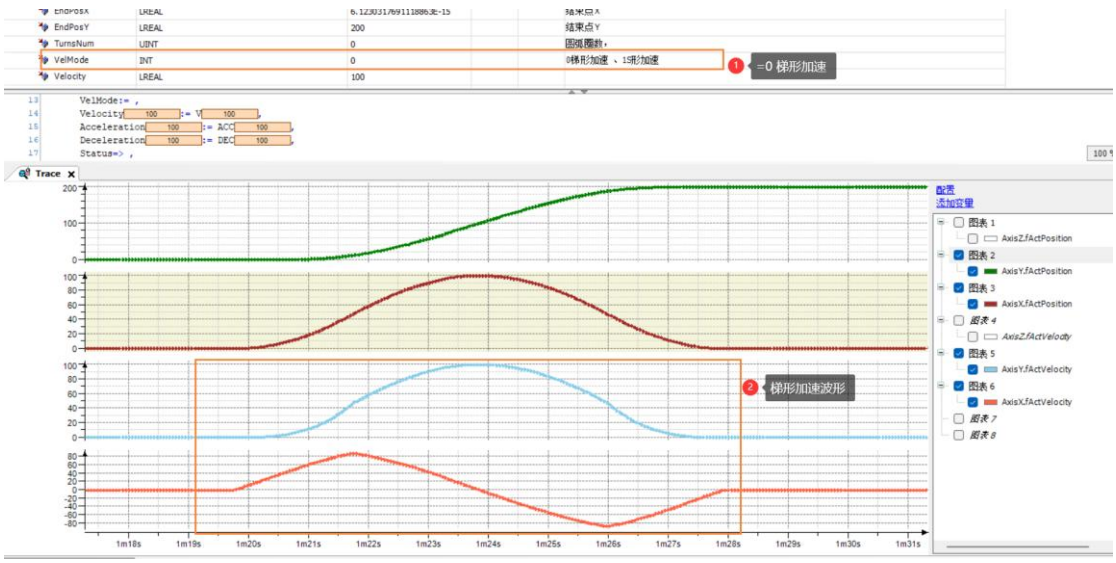
（1） 声明与写值

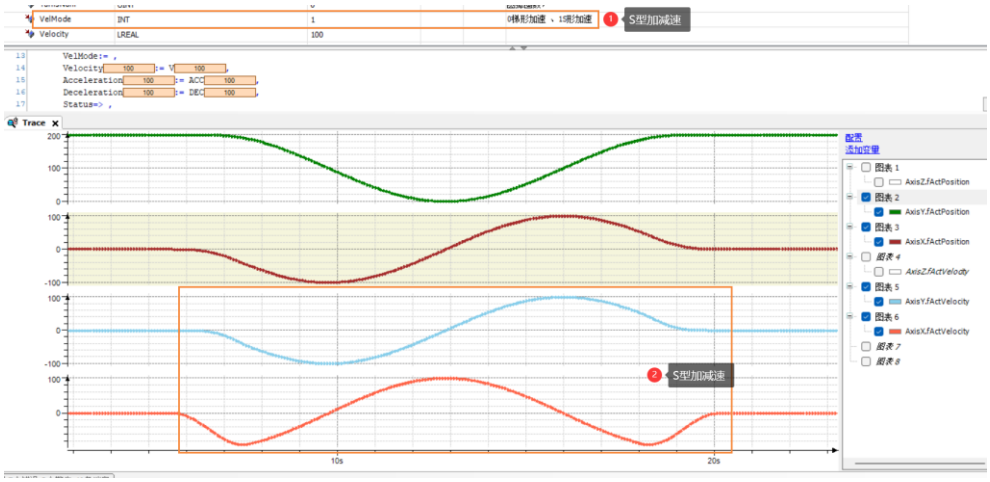
```
PROGRAM POU_3
VAR
    FB_EndPosC: FB_EndPosCircleInterpolation ;

    V: LREAL := 100;
    ACC: LREAL := 100;
    DEC: LREAL := 100;

    EndX:LREAL;
    EndY:LREAL;
    Angle:LREAL;
END_VAR
```

```
FC_Circle_AngleGteEndXY(In_I:= 0, In_J:= 100, In_R:= 100, In_Angle:= Angle, Out_EndX=> EndX, Out_EndY=>EndY );
FB_EndPosC(
    AxisX:= AxisX,
    AxisY:= AxisY,
    Execute:= ,
    Pausestop:= ,
    bCW:= ,
    ICentreCircle:= 0,
    JCentreCircle:=100 ,
    EndPosX:= EndX,
    EndPosY:= EndY,
    TurnsNum:= ,
    VelMode:= ,
    Velocity:= V,
    Acceleration:= ACC,
    Deceleration:= DEC,
    Status=> ,
    Done=> ,
    Busy=> ,
    CommandAbort=> ,
    Error=> ,
    ErrorID=> );
```

注：EndX 和 EndY 可以用通过半径用函数设定

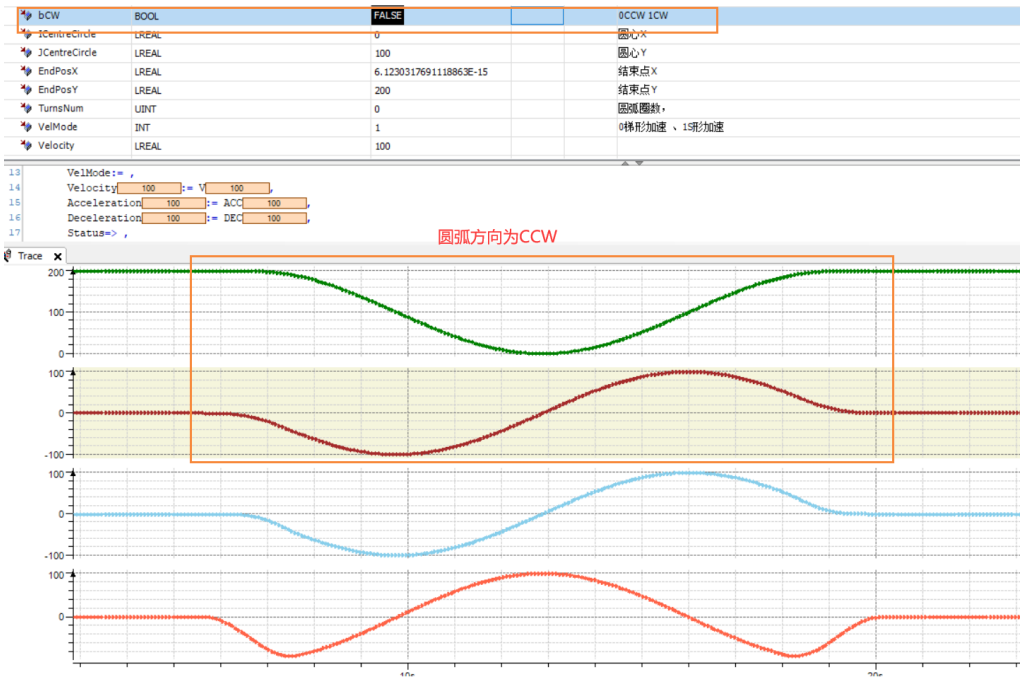
（2） T 型和 S 型加速



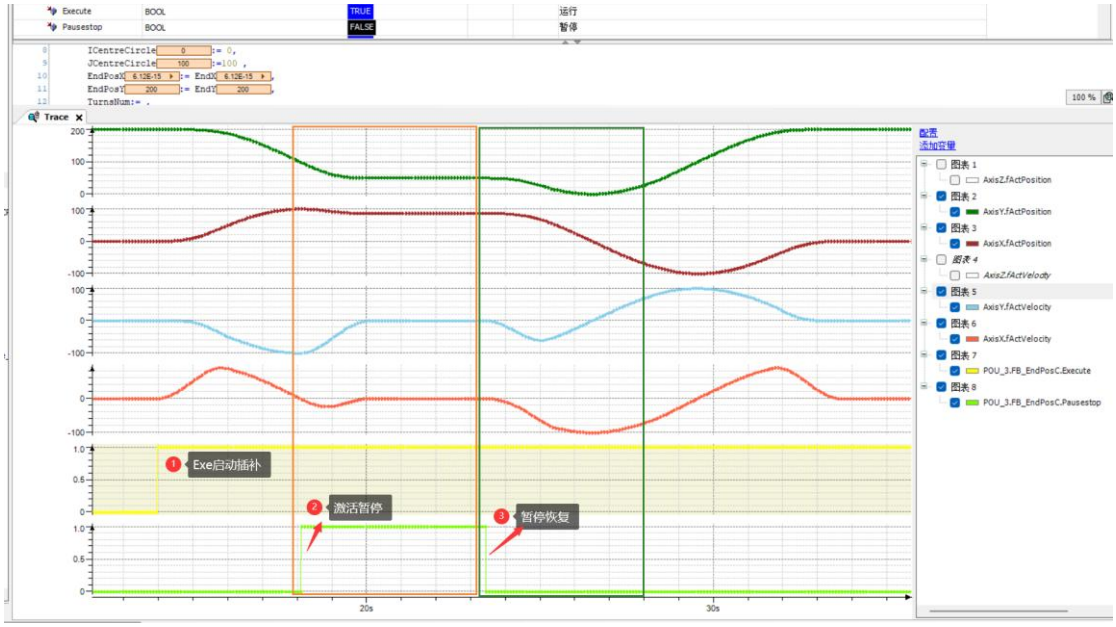


（3） 圆弧方向通过 bCW 引脚设定





（4） 插补暂停后再启动



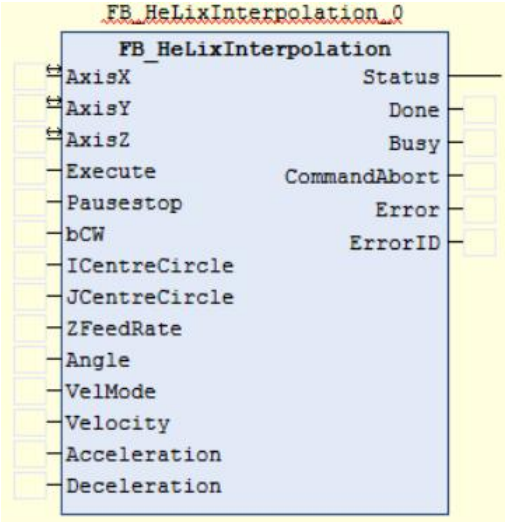
FB_HeLixInterpolation 三轴螺旋插补

功能块说明：三轴螺旋插补 。当前位置为起点、根据圆心，螺旋角度 Z 增量，规划路径

注意：目前还不支持速度衔接，触发运动该指令时 需要确保轴速度为 0，如果不为 0 速度会从当前速度突变到 0 然后开始正常规划运动

1、指令格式

图形表现：



ST 表现：

```
FB_HeLixInterpolation_0(
    AxisX:= ,
    AxisY:= ,
    AxisZ:= ,
    Execute:= ,
    Pausestop:= ,
    bCW:= ,
    ICentreCircle:= ,
    JCentreCircle:= ,
    ZFeedRate:= ,
    Angle:= ,
    VelMode:= ,
    Velocity:= ,
    Acceleration:= ,
    Deceleration:= ,
    Status=> ,
    Done=> ,
    Busy=> ,
    CommandAbort=> ,
    Error=> ,
    ErrorID=> );
```

2、相关变量

变量名	名称	数据类型	输入输出属性	描述
AxisX	插补轴 X	AXIS_REF_SM3	输入输出变量	插补轴 X
AxisY	插补轴 Y	AXIS_REF_SM3	输入输出变量	插补轴 Y
AxisZ	插补轴 Z	AXIS_REF_SM3	输入输出变量	插补轴 Z
Execute	运行	BOOL	输入变量	上升沿有效
Pausestop	暂停	BOOL	输入变量	=TRUE 暂停
bCw	运动方向	BOOL	输入变量	=0 CCW，=1CW
ICentreCircle	圆心坐标 X	LREAL	输入变量	圆心的 X 坐标
JCentreCircle	圆心坐标 Y	LREAL	输入变量	圆心的 Y 坐标
ZFeedRate	Z 轴增量	LREAL	输入变量	Z 轴增量
Angle	旋转角度	LREAL	输入变量	旋转角度
VelMode	速度模式设置	INT	输入变量	=0 梯形加速，=1 S 型加速
Velocity	速度值	LREAL	输入变量	速度
Acceleration	加速度值	LREAL	输入变量	加速度
Deceleration	加速度值	LREAL	输入变量	减速度
Status	状态输出	INT	输出变量	0:无状态，1:加速,2:匀速，3 减速，4 结束，5 暂停减速中，6 暂停完成
Done	完成	BOOL	输出变量	=TRUE 完成
Busy	运行中	BOOL	输出变量	=TRUE 执行中
CommandAbort	中断执行	BOOL	输出变量	=TRUE 指令中断
Error	报错	BOOL	输出变量	=TRUE 错误

ErrorID	错误代码	INT	输出变量	0 无错误， 1 圆弧终点或圆心数据错误，或者 Angle<=0 2 速度类参数错误 3 轴未准备好

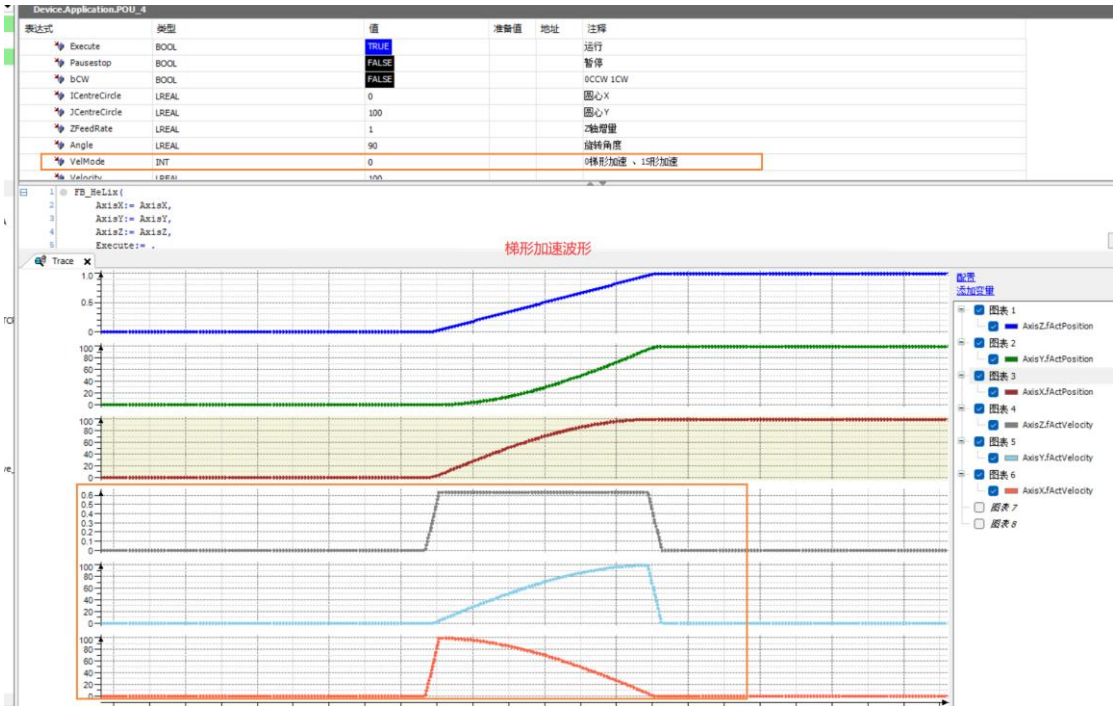
3、 详细使用

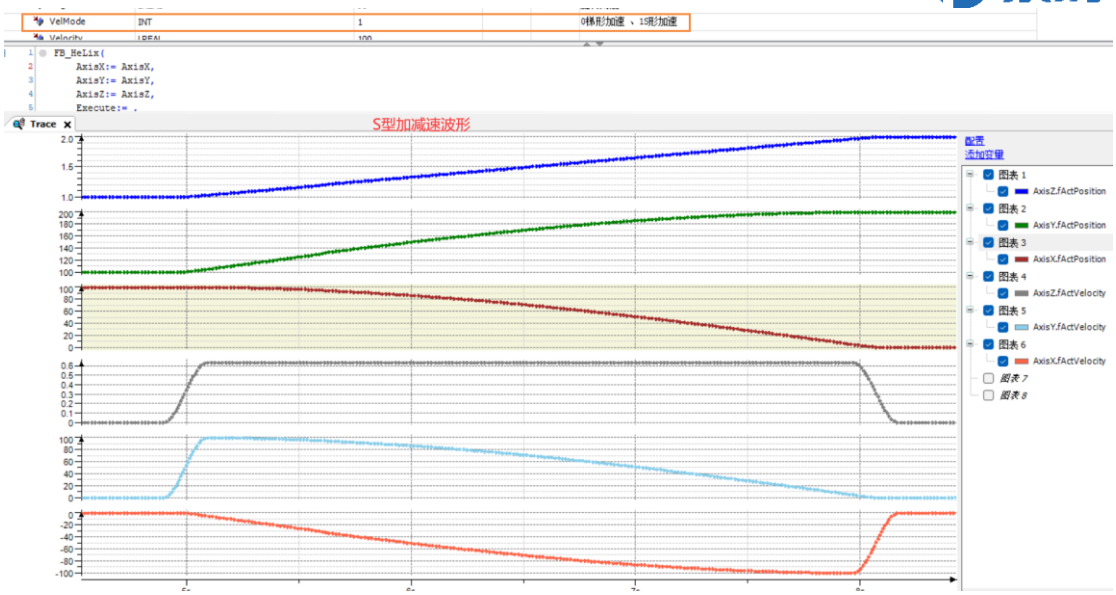
（1） 声明与实例化

```
PROGRAM POU_4
VAR
    FB_HeLix: FB_HeLixInterpolation ;
END_VAR
```

```
FB_HeLix(
    AxisX:= AxisX,
    AxisY:= AxisY,
    AxisZ:= AxisZ,
    Execute:= ,
    Pausestop:= ,
    bCW:= ,
    ICentreCircle:= 0,
    JCentreCircle:= 100,
    ZFeedRate:= ,
    Angle:= ,
    VelMode:= ,
    Velocity:= 100,
    Acceleration:= 1000,
    Deceleration:= 1000,
    Status=> ,
    Done=> ,
    Busy=> ,
    CommandAbort=> ,
    Error=> ,
    ErrorID=> );
```

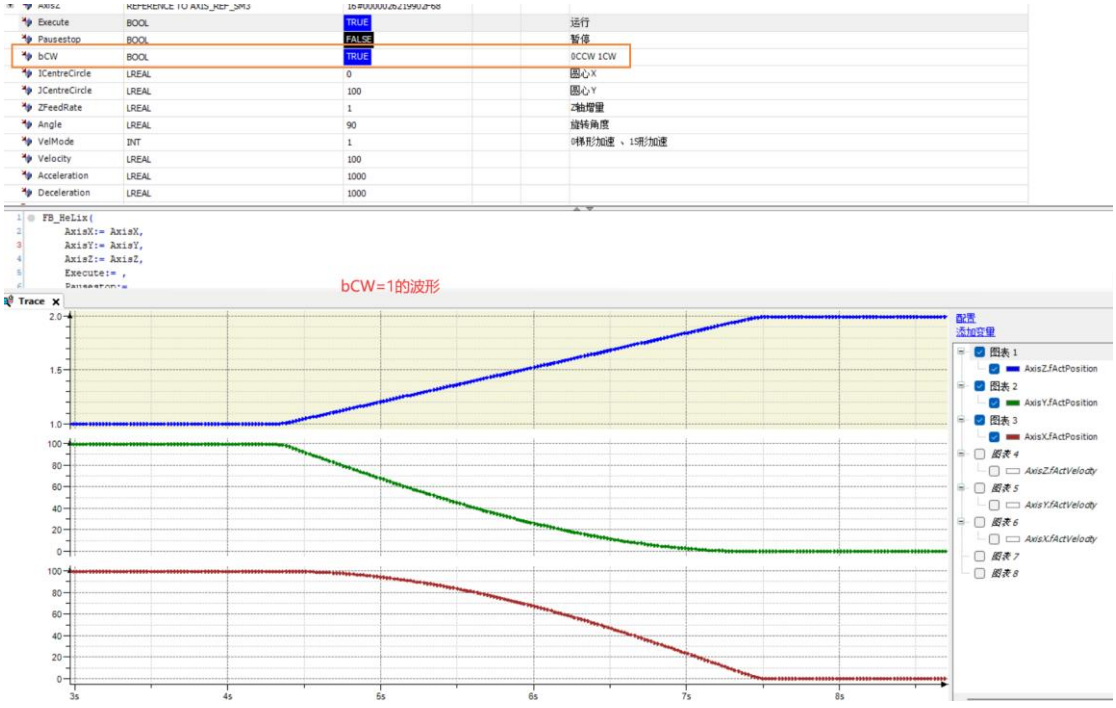
（2） T 型加速和 S 型加速



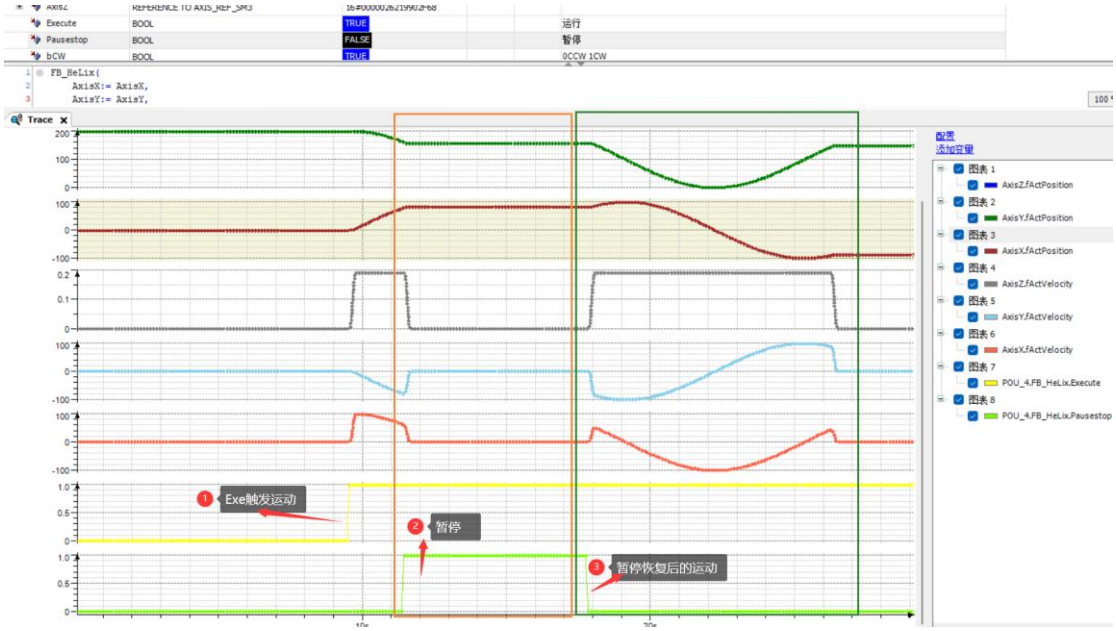

VelMode
INT
1
0梯形加速 、1S形加速
FB_HeLix(
AxisX:= AxisX,
AxisY:= AxisY,
AxisZ:= AxisZ,
Execute:= ,
Trace
S型加减速波形
配置
添加容量
图表 1
AxisZ.fActPosition
图表 2
AxisY.fActPosition
图表 3
AxisX.fActPosition
图表 4
AxisZ.fActVelocity
图表 5
AxisY.fActVelocity
图表 6
AxisX.fActVelocity
图表 7
图表 8

（3）圆弧方向设置 bCW


表达式
类型
值
准备值
地址
注释
AxisZ
REFERENCE TO AXIS_REF_SM3
16#0000026219902F68
Execute
BOOL
TRUE
运行
Pausestop
BOOL
FALSE
暂停
bCW
BOOL
FALSE
0CCW 1CW
iCentreCircle
LREAL
0
圆心X
jCentreCircle
LREAL
100
圆心Y
ZFeedRate
LREAL
1
Z轴增量
Angle
LREAL
90
旋转角度
VelMode
INT
1
0梯形加速 、1S形加速
Velocity
LREAL
100
Acceleration
LREAL
1000
Deceleration
LREAL
1000
FB_HeLix(
AxisX:= AxisX,
AxisY:= AxisY,
AxisZ:= AxisZ,
Execute:= ,
Trace
bCW=0波形
配置
添加容量
图表 1
AxisZ.fActPosition
图表 2
AxisY.fActPosition
图表 3
AxisX.fActPosition
图表 4
AxisZ.fActVelocity
图表 5
AxisY.fActVelocity
图表 6
AxisX.fActVelocity
图表 7
图表 8
17s
18s
19s
20s
21s
22s
23s



（4）暂停和暂停后再启动



01_凸轮控制切入齿轮

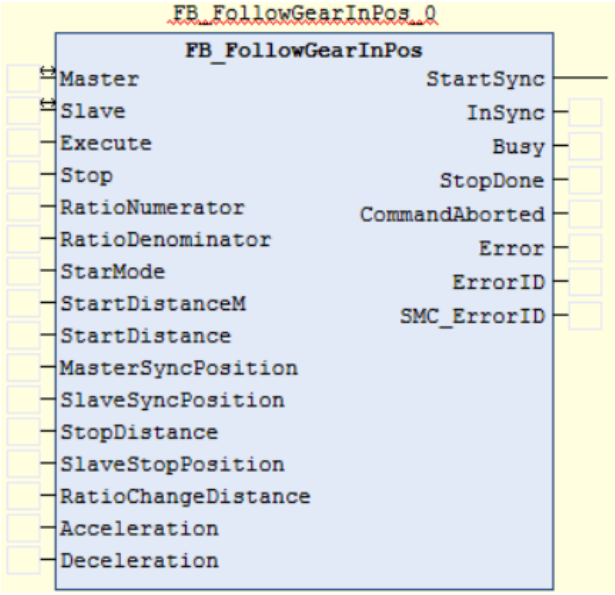
FB_FollowGearInPos 特定位置齿轮耦合

功能说明：主从轴需要都是线性轴，以当前位置为起点。MasterSyncPosition，

SlaveSyncPosition 为加速终点后开始以齿轮比运行主从轴耦合。位置可设定，加速的位置主从可设定，停止位置可设定，停止减速距离可设

1、指令格式

图形表现：



ST 表现：

```
FB_FollowGearInPos_0(
    Master:= ,
    Slave:= ,
    Execute:= ,
    Stop:= ,
    RatioNumerator:= ,
    RatioDenominator:= ,
    StarMode:= ,
    StartDistanceM:= ,
    StartDistance:= ,
    MasterSyncPosition:= ,
    SlaveSyncPosition:= ,
    StopDistance:= ,
    SlaveStopPosition:= ,
    RatioChangeDistance:= ,
    Acceleration:= ,
    Deceleration:= ,
    StartSync=> ,
    InSync=> ,
    Busy=> ,
    StopDone=> ,
    CommandAborted=> ,
    Error=> ,
    ErrorID=> ,
    SMC_ErrorID=> );
```

2、 相关变量

变量名	名称	数据类型	输入输出属性	描述
Master	主轴	AXIS_REF_SM3	输入输出变量	主轴
Slave	从轴	AXIS_REF_SM3	输入输出变量	从轴
Execute	触发运行	BOOL	输入变量	触发运行
Stop	停止	BOOL	输入变量	停止
RatioNumerator	分子	LREAL	输入变量	分子
RatioDenominator	分母	LREAL	输入变量	分母
StarMode	启动模式	INT	输入变量	0:从当前位置启动，主轴以 StartDistanceM 从轴以 StartDistance 加速 1:给定位置启动，主轴以 StartDistance 从轴以 StartDistance 加速 2:从当前位置启动，按 StartDistance 齿轮比加速 3:按给定位置启动，按 StartDistance 齿轮比加速

StartDistanceM	起始位置	LREAL	输入变量	StarMode=0 有效，0 则按 ACC 加速耦合主轴位移 StartDistanceM，从轴位置 StartDistance
StartDistance	起始位置	LREAL	输入变量	起始位置
MasterSyncPosition	主轴开始同步位置	LREAL	输入变量	以当前位置为起点，到达该参数位置 + StartDistance 后开始同步 ,StarMode=1 会提前 StartDistance 开始耦合使加速更平滑
SlaveSyncPosition	从轴开始同步的位置	LREAL	输入变量	以当前位置为起点，到达该参数位置+ StartDistance 后开始同步
StopDistance	停止时的从轴运动距离	LREAL	输入变量	停止时 从轴的运动距离
SlaveStopPosition	停止时从轴位置	LREAL	输入变量	停止时从轴位置
RatioChangeDistance	齿轮比变更，加速距离	LREAL	输入变量	齿轮比变更，加速距离
Acceleration	加速度	LREAL	输入变量	加速度
Deceleration	减速度	LREAL	输入变量	减速度
StartSync	开始同步	BOOL	输出变量	开始同步
InSync	同步中	BOOL	输出变量	同步中
Busy	运行中	BOOL	输出变量	运行中
StopDone	停止完成	BOOL	输出变量	停止完成
CommandAborted	功能块被打断	BOOL	输出变量	功能块被打断
Error	产生错误	BOOL	输出变量	产生错误
ErrorID	功能块报错	WORD	输出变量	1:轴未准备好，2：ACC DEC 错误，3：分子分母错误，4 MC 错误 ，5：iMovementType 主或从轴不是线性轴 ，
SMC_ErrorID	SMC 错误信息	SM3_Basic.SMC_ERROR	输出变量	SMC 错误信息

02_旋切/追剪

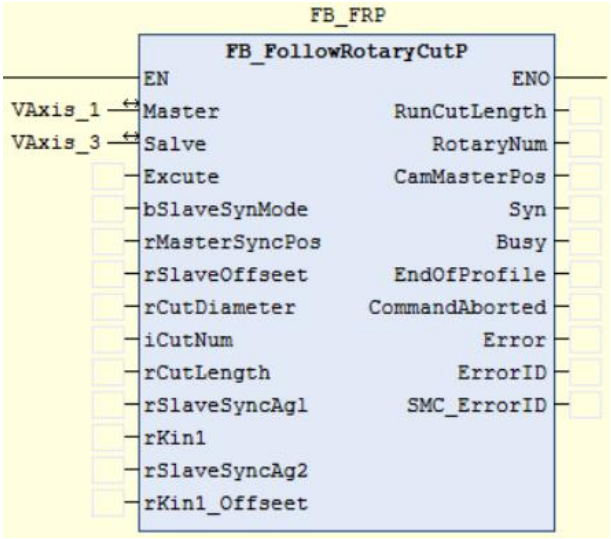
021 旋切

FB_FollowRotaryCutP 飞剪跟随

功能块说明：本功能块适用于飞剪类型动作 可根据需求指定主轴周期（裁切长度）、从轴周期、同步比例。

1、指令格式

图形表现



ST 表现

```
FB_FollowRotaryCutP(
    Master:=VAxis_1 ,
    Salve:=VAxis_2 ,
    Excute:= ,
    bSlaveSynMode:= ,
    rMasterSyncPos:= ,
    rSlaveOffseet:= ,
    rCutDiameter:= ,
    iCutNum:= ,
    rCutLength:= ,
    rSlaveSyncAgl:= ,
    rKin1:= ,
    rSlaveSyncAg2:= ,
    rKin1_Offseet:= ,
    RunCutLength=> ,
    RotaryNum=> ,
    CamMasterPos=> ,
    Syn=> ,
    Busy=> ,
    EndOfProfile=> ,
    CommandAborted=> ,
    Error=> ,
    ErrorID=> ,
    SMC_ErrorID=> );
```

1、 相关变量

输入输出变量

输入输出变量	名称	数据类型	有效范围	描述
Master	主轴	AXIS_REF_SM3		映射到轴，即 AXIS_REF_SM3 的一个实例,线性轴
Salve	从轴	AXIS_REF_SM3		映射到轴，即 AXIS_REF_SM3 的一个实例，模态轴 360°

输入变量

输入变量	名称	数据类型	有效范围	描述
Excute	旋切启动	BOOL	TRUE,FALSE	上升沿，执行旋切
bSlaveSynMode	同步模式	BOOL		0:直接以当前速比耦合， 1:当前到下一点为加速耦合
rMasterSyncPos	主轴启动位置	LREAL		Master 经过该位置后开始运行凸轮
rSlaveOffseet	从轴偏移	LREAL	>=0 <= 360	从旋切该角度切入运行
rCutDiameter	切刀直径	LREAL	>0.0	切刀的直径
iCutNum	切刀数量	Int	>0	切刀数量
rCutLength	主轴周期	LREAL	>0	旋切主轴周期
rSlaveSyncAgl	同步角度 1	LREAL	>0	切刀咬合区间主从轴线速度同步区间（度）
rKin1	同步角度 1 的速比	LREAL	>0	主轴与从轴同步区间线速度同步比例
rSlaveSyncAg2	同步角度 2	LREAL	>0<rSlaveSyncAgl	该角度区间 rKin1_Offseet<>0 时表现为五次曲线
rKin1_Offseet	同步速比 1 偏移	LREAL		在同步的速度下，调整 rSlaveSyncAg2 区间的速度

输出变量

输出变量	名称	数据类型	有效范围	描述
RunCutLength	当前裁切长度	LREAL		当前运行裁切长度
RotaryNum	当前切刀	Int		当前裁切的切刀
CamMasterPos	当前凸轮主轴位置	LREAL	0- rCutLength	当前旋切主轴位置
Syn	同步区	BOOL	TRUE,FALSE	切刀线速度与主轴线速度同步时该引脚置 true 当离开同步区该引脚置 false
Busy	运行	BOOL	TRUE,FALSE	Execute 输入上升沿时，置位 TRUE，示旋切关系耦合中，需用外部打断指令
EndOfProfile	凸轮曲线完成	BOOL	TRUE,FALSE	凸轮曲线执行完一次， 该引脚输出一次
CommandAborted	打断	BOOL	TRUE,FALSE	如果旋切从轴被外部打断，该位置 TRUE 当指令的执行条件 OFF 时，Error 位被复位。
Error	报错	BOOL	TRUE,FALSE	如果检测到有错误，Error 位被置位; 当指令的执行条件 OFF 时，Error 位被复位。
ErrorID	功能块错误 ID	WORD		1:轴未准备好，2: 工艺参数错误，

				3：跟随同步错误，4:当前轴为非模态轴
SMC_ErrorID	轴错误 ID	SMC_ERROR		当前轴控功能块错误 ID

2、 使用说明

4.1、bSlaveSynMode：

0：耦合时从轴不做加速处理 耦合当前周期直接根据凸轮关系到达目标速度，

1：耦合时从轴按 S 形加速跟随到同步。注意:为 1 时 第一个周期不可反转 否则会突变，加速过程规划在当前位置到下一关键点

4.2、rMasterSyncPos：主轴到了该位置 开始跟随动作(相当于绝对位置启动凸轮，如不需要 则直接给 Master.fsetPosition)，

注意:需要是正向越过该位置才会开始执行旋切

4.3、同步区间（rSlaveSyncAgl） 是以切刀在零点左右划分

比如设定值为 30 度 那么主从轴的同步区间在 345-15 度

4.4、从轴偏移（rSlaveOffseet）

当旋切启动切刀不是在 0 点的位置，需要给出当前切刀的位置 以保证凸轮正常运行

4.5、运行过程中参数变更

iCutNum 为 Excute 时有效，如需修改 需打断凸轮重新执行

运行过程中 rCutDiameter、rCutLength、rSlaveSyncAgl、rKin1、rSlaveSyncAg2、rKin1_Offseet 参数变更后 会在下一周期有效

FB_Rotary_Cut_Cam 切原点旋切曲线设置

- **功能块解释说明：**

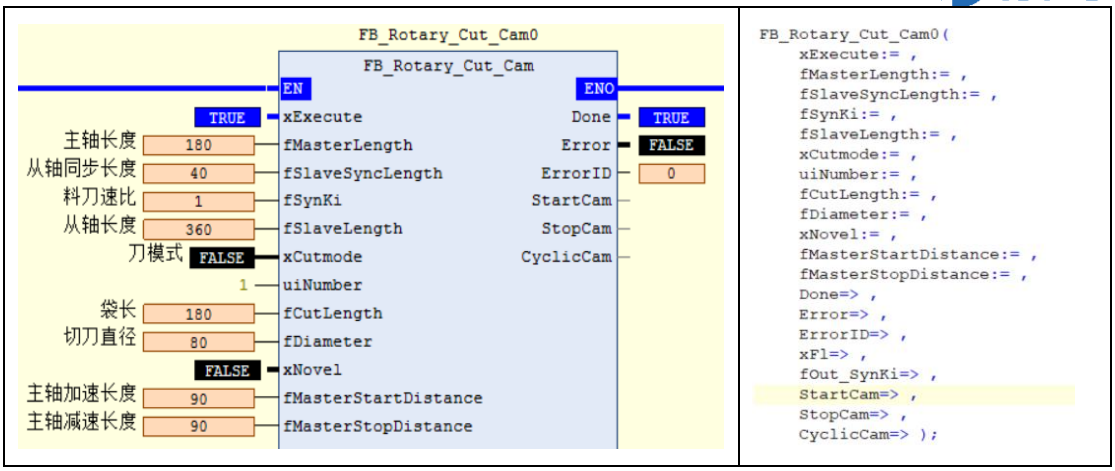
  此功能块可以形成三条凸轮曲线，分别是启动、连续、停止凸轮曲线。相关参数设置包含了主轴线性或旋转，从轴线性或旋转等不同的场景。若主轴线性，则主轴长度=袋长，否则主轴长度<>袋长；若从轴线性，则刀模式=true，从轴长度为从轴周长，切刀直径不设置，若从轴为模态，则从轴从轴长度为刀转一圈 360 度，刀模式=false，切刀直径按照实际设置。一般主轴加、减速长度设为主轴长度的一半。

- **使用场景描述：**

  运动控制中需要知道飞剪曲线，并自己构建相关控制逻辑的场合。

1、指令格式

图形表现 ST 表现



2、功能块变量说明

（1）输入输出变量（Var In_Out）

变量名称	数据类型	默认值	有效范围	描述
/	/	/	/	/

（2）输入变量（VAR_Input）

变量名称	数据类型	默认值	有效范围	描述
xExecute	BOOL	/	/	使能功能块有效
fMasterLength	LREAL	/	/	主轴模态时，为主轴模值，线性时为裁切长度
fSlaveSyncLength	LREAL	/	/	同步长度(角度)，旋转刀时为从轴角度。
fSynKi	LREAL			料刀线速度比
fSlaveLength	LREAL			刀轴一圈的周长(角度)，xCutmode 时为周长，否则为角度。
xCutmode	BOOL			切刀模式,false 为旋转刀(一圈是 360 度)，true 为线性刀(一圈是周长)
uiNumber	UINT			旋切刀头数, xCutmode =false 时有效
fCutLength	LREAL			裁切长度, xCutmode =false 时有效
fDiameter	LREAL			切刀直径, xCutmode =false 时有效
xNovel	BOOL			长料停止区跟踪
fMasterStartDistance	LREAL			启动阶段主轴长度
fMasterStopDistance	LREAL			停止阶段主轴长度

（3）输出变量（VAR_Output）

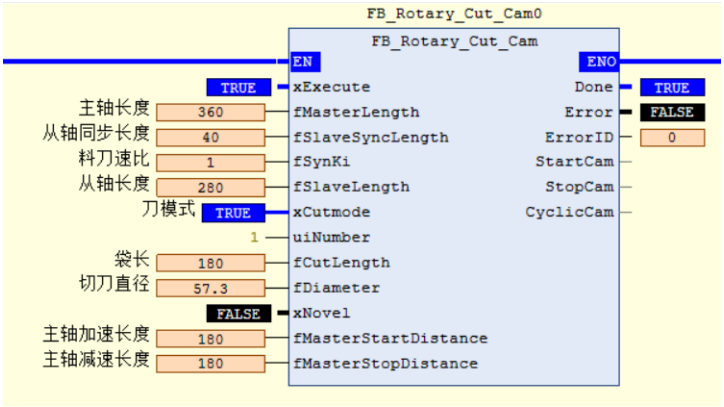
输出变量	数据类型	默认值	有效范围	描述
Done	BOOL	/		
Error	BOOL	/		
ErrorID	INT			报警： 0:无错误，1：规划从轴最大值超限, 3:非繁易控制器

xFl	BOOL			繁易产品 ID
fOut_SynKi	LREAL			输出材料速度与刀切线速度的比例
StartCam	MC_Cam_Ref			启动凸轮表
StopCam	MC_Cam_Ref			停止凸轮表
CyclicCam	MC_Cam_Ref			连续运行凸轮表

3、使用样例

（1）声明与调用

```
PROGRAM EtcCTL
VAR
    //FB_Rotary_Cut_Cam0 :FB_Rotary_Cut_Cam;
    FB_Flying_Saw_Cam0 :FB_Flying_Saw_Cam;

END_VAR
```



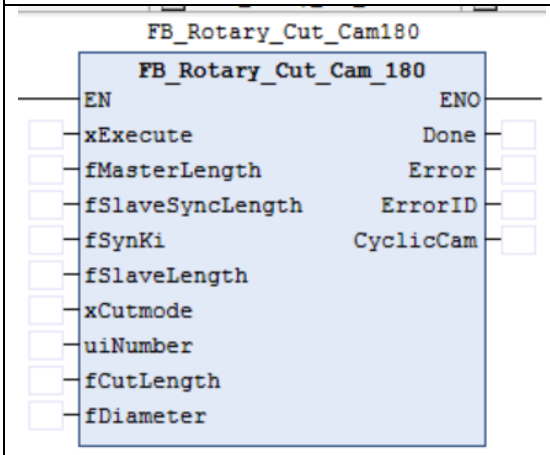
（2）功能演示部分



FB_Rotary_Cut_Cam_180 停原点旋切曲线设置

根据设定的参数，自动生成连续运行的飞剪曲线，飞剪曲线从待料点启动，待料点在切割点对称的背面。运动控制中需要用到绝对凸轮的飞剪切割场合。

1、指令格式

图形表现：	ST 表现：
	FB_Rotary_Cut_Cam180(xExecute:= , fMasterLength:= , fSlaveSyncLength:= , fSynKi:= , fSlaveLength:= , xCutmode:= , uiNumber:= , fCutLength:= , fDiameter:= , Done=> , Error=> , ErrorID=> , CyclicCam=>);

2、功能块变量说明

（1）输入变量（VAR_Input）

变量名称	数据类型	默认值	有效范围	描述
xExecute	BOOL	/	/	使能功能块有效
fMasterLength	LREAL	/	/	主轴长度
fSlaveSyncLength	LREAL	/	/	同步长度(角度)，一般为从轴长度的十分之一。
fSynKi	LREAL			料刀线速度比,外部计算，兼容(线型,旋转)切刀轴
fSlaveLength	LREAL			从轴长度(角度)，xCutmode =false 时自动算出
xCutmode	BOOL			切刀模式,0 为旋转刀，1 为线性刀
uiNumber	UINT			旋切刀头数, xCutmode =false 时有效
fCutLength	LREAL			裁切长度, xCutmode =false 时有效
fDiameter	LREAL			切刀直径, xCutmode =false 时有效

（2）输出变量（VAR_Output）

输出变量	数据类型	默认值	有效范围	描述
Done	BOOL	/		
Error	BOOL	/		
ErrorID	INT			报警： 0:无错误，1：规划从轴最大值超限
CyclicCam	MC_Cam_Ref			连续运行凸轮表

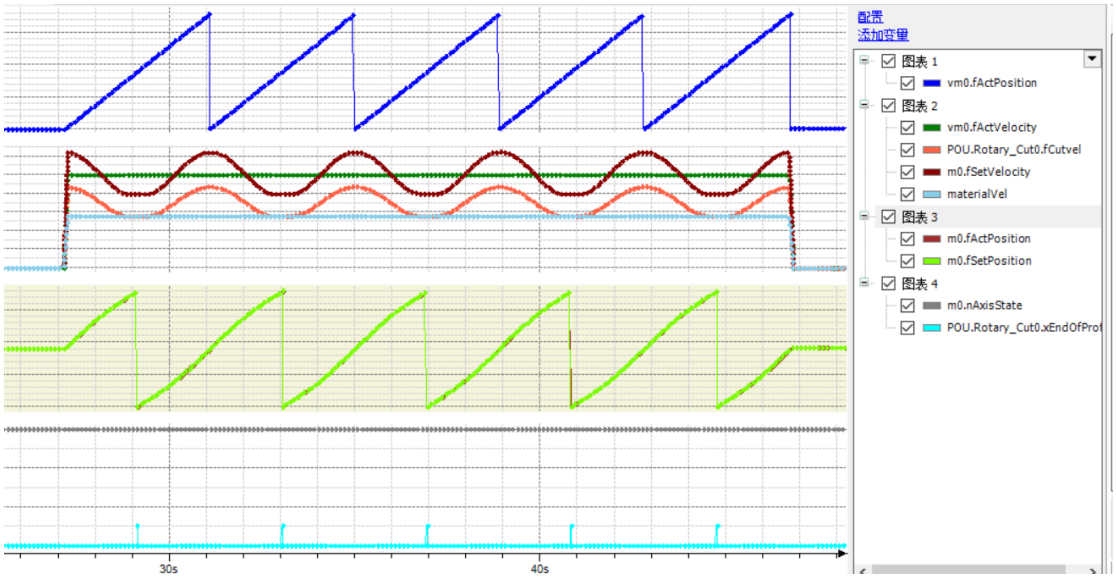
3、使用样例

（1）声明与调用

```
PROGRAM EtcCTL
VAR
    //FB_Rotary_Cut_Cam0 :FB_Rotary_Cut_Cam;
    //FB_Flying_Saw_Cam0 :FB_Flying_Saw_Cam;
    FB_Rotary_Cut_Cam180 :FB_Rotary_Cut_Cam_180;

END_VAR
```

```
FB_Rotary_Cut_Cam180(
    xExecute:= ,
    fMasterLength:= ,
    fSlaveSyncLength:= ,
    fSynKi:= ,
    fSlaveLength:= ,
    xCutmode:= ,
    uiNumber:= ,
    fCutLength:= ,
    fDiameter:= ,
    Done=> ,
    Error=> ,
    ErrorID=> ,
    CyclicCam=> );
```

（2）功能演示部分



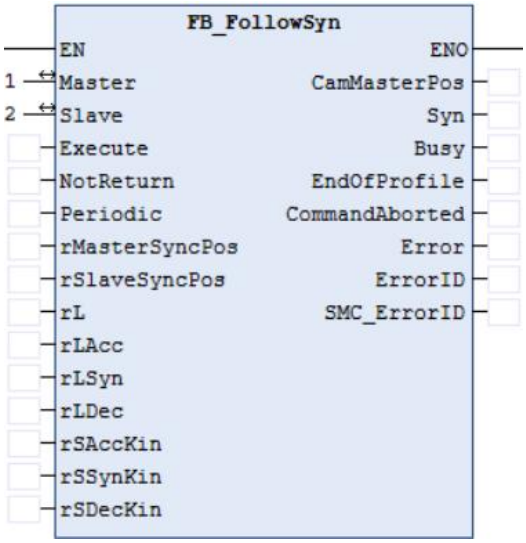
022 追剪

FB_FollowSyn 追剪跟随

指令名称： FB_FollowSyn 追剪功能块
功能块说明：本功能块适用于追剪类型动作 可根据设定追剪参数实现定长追剪

1、指令格式

图形表现



ST 表现

```
FB_FollowSyn(
    Master:=Master ,
    Salve:=Salve ,
    Excute:= ,
    NotReturn:= ,
    Periodic:= ,
    MasterSyncPos:= ,
    SlaveSyncPos:= ,
    rL:= ,
    rLAcc:= ,
    rLSyn:= ,
    rLDec:= ,
    rSAccKin:= ,
    rSSynKin:= ,
    rSDecKin:= ,
    CamMasterPos=> ,
    Sync=> ,
    Busy=> ,
    EndOfProfile=> ,
    CommandAborted=> ,
    Error=> ,
    ErrorID=> ,
    SMC_ErrorID=> );
```

2、相关变量

输入输出变量

输入输出变量	名称	数据类型	有效范围	描述
Master	主轴	AXIS_REF_SM3		映射到轴，即 AXIS_REF_SM3 的一个实例,线性轴
Salve	从轴	AXIS_REF_SM3		映射到轴，即 AXIS_REF_SM3 的一个实例，模态轴 360°

输入变量

输入变量	名称	数据类型	有效范围	描述
Excute	启动	BOOL	TRUE,FALSE	上升沿，执行旋切
NotReturn	追剪模式	BOOL	TRUE,FALSE	FALSE:带返回路径，TRUE:单次凸轮减速后无返回动作
Periodic	重复模式	BOOL	TRUE,FALSE	追剪动作是否重复 TRUE:重复，FALSE:不重复
rMasterSyncPos	主轴开始跟随位置	LREAL		Master 经过该位置后开始运行凸轮
rSlaveSyncPos	从轴开始跟随位置	LREAL		从轴从该位置开始追剪
rL	追剪主轴总长	LREAL	>0.0	

rLAcc	加速主轴距离	Int	>0	
rLSyn	同步主轴距离	LREAL	>0	
rLDec	减速主轴距离	LREAL	>0	
rSAccKin	从轴加速速比	LREAL	>0	
rSSynKin	从轴同步速比	LREAL		
rSDecKin	从轴减速速比	LREAL		

输出变量

输出变量	名称	数据类型	有效范围	描述
CamMasterPos	当前凸轮主轴位置	LREAL	0- rL	当前旋切主轴位置
Syn	同步区	BOOL	TRUE,FALSE	切刀线速度与主轴线速度同步时该引脚置 true，当离开同步区该引脚置 false
Busy	运行	BOOL	TRUE,FALSE	Execute 输入上升沿时，置位 TRUE，示旋切关系耦合中，需用外部打断指令
EndOfProfile	凸轮曲线完成	BOOL	TRUE,FALSE	凸轮曲线执行完一次， 该引脚输出一次
CommandAborted	打断	BOOL	TRUE,FALSE	如果旋切从轴被外部打断，该位置 TRUE 当指令的执行条件 OFF 时，Error 位被复位。
Error	报错	BOOL	TRUE,FALSE	如果检测到有错误，Error 位被置位; 当指令的执行条件 OFF 时，Error 位被复位。
ErrorID	功能块错误 ID	WORD		1:轴未准备好，2: 工艺参数错误，3: 跟随同步错误，4:当前轴为非模态轴
SMC_ErrorID	轴错误 ID	SMC_ERROR		当前轴控功能块错误 ID

3、使用说明

3.1、规划的轨迹为绝对路径

当 Master.fsetposition 经过 rMasterSyncPos 位置后开始按追剪轨迹运动，从轴从 rSlaveSyncPos 开始运行

3.5、运行过程中参数变更

运行过程中 rL、rLAcc、rLDec、rSAccKin、rSSynKin、 rSDecKin 参数变更后 会在下一周期有效

FB_Flying_Saw_Cam 追剪曲线设置

- **功能块解释说明：**

  通过参数设置，生成追剪的曲线。

  1:主轴线性，主轴长度等于袋长，同步长度小于袋长一半，从轴最大长度限值按照刀台可移动的最大距离设置即可。

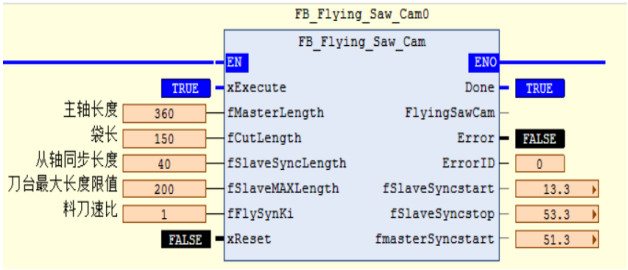
  2:主轴模态，主轴长度等于模值，同步长度小于袋长一半，从轴最大长度限值按照刀台可移动的最大距离设置即可。有时候也会把通过 PLC 软件，将主轴模值设置成袋长，那么功能块这里也设成袋长即可。

- **使用场景描述：**

  运动控制中的需要自定义凸轮关系的追剪场合

1、指令格式

图形表现



ST 表现

```
FB_Flying_Saw_Cam0(
    xExecute:= ,
    fMasterLength:= ,
    fCutLength:= ,
    fSlaveSyncLength:= ,
    fSlaveMAXLength:= ,
    fFlySynKi:= ,
    xReset:= ,
    Done=> ,
    Error=> ,
    ErrorID=> ,
    FlyingSawCam=> ,
    fmasterSyncstart=> ,
    fmasterSyncstop=> ,
    fSlaveSyncstart=> ,
    fSlaveSyncstop=> ,
    xFl=> );
```

2、功能块变量说明

（1）输入变量（VAR_Input）

变量名称	数据类型	默认值	有效范围	描述
xExecute	BOOL	/	/	使能功能块有效
fMasterLength	DINT	/	/	主轴长度,如果主轴是线性轴，这就是剪切的长度。
fCutLength	LREAL	/	/	裁剪长度
fSlaveSyncLength	LREAL	/	/	从轴同步长度
fSlaveMAXLength	LREAL	/	/	从轴最大可达长度
fFlySynKi	LREAL	/	/	同步速比
xReset	BOOL	/	/	复位故障使用

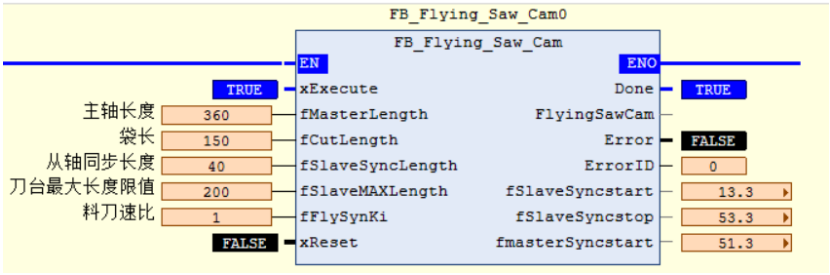
（2）输出变量（VAR_Output）

输出变量	数据类型	默认值	有效范围	描述
Done	BOOL	/	/	求解完成
Error	BOOL	/	/	
ErrorID	INT	/	/	//0:无错误，1: 规划从轴最大值超限,2:同步区过长，3:非繁易控制
FlyingSawCam	MC_Cam_Ref	/	/	运行凸轮表
fSlaveSyncstart	REAL	/	/	主轴同步起点
fSlaveSyncstop	REAL	/	/	主轴同步终点
fSlaveSyncstart	REAL	/	/	从轴同步起点
fSlaveSyncstop	REAL	/	/	从轴同步终点
xFl	BOOL	/	/	繁易产品 ID

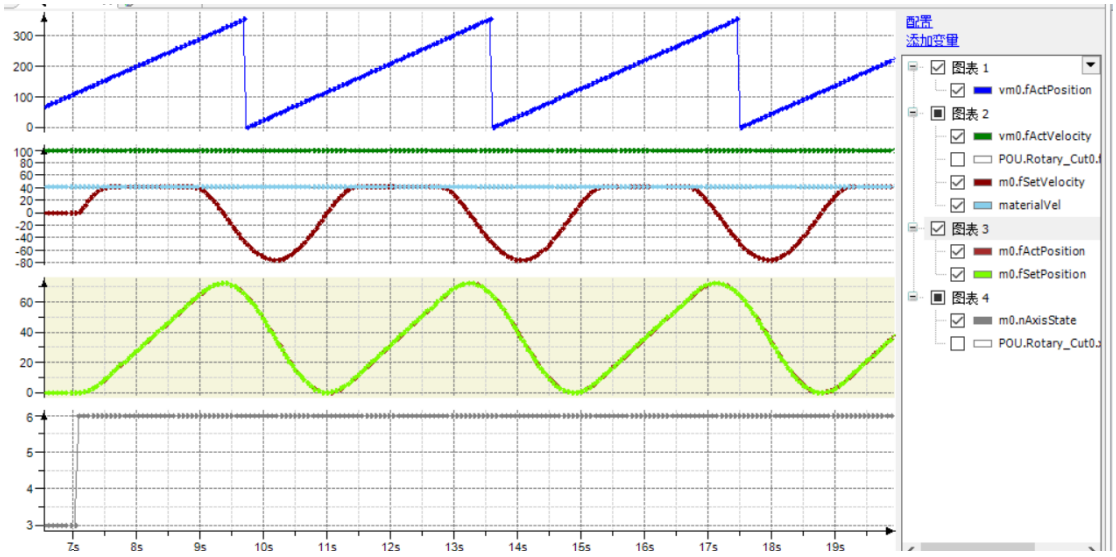
3、使用样例

（1）声明与调用

```
PROGRAM EtcCTL
VAR
    //FB_Rotary_Cut_Cam0 :FB_Rotary_Cut_Cam;
    FB_Flying_Saw_Cam0 :FB_Flying_Saw_Cam;

END_VAR
```



（2）功能演示部分



03_CNC 文件处理

待发

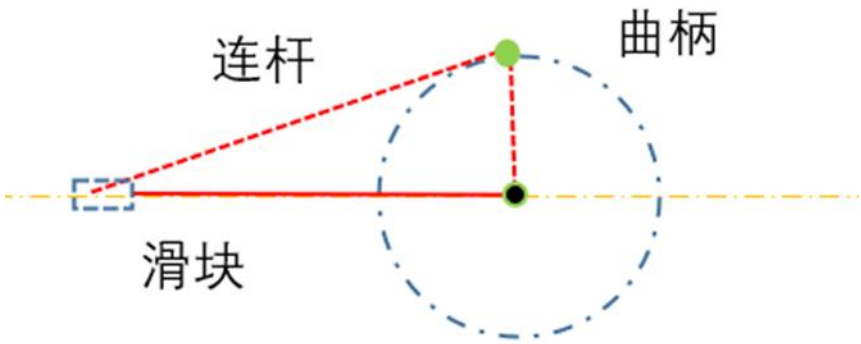
04_曲柄滑块结构

FB_CenteredCrankSlider 对心曲柄滑块机构的运算

功能块解释说明:

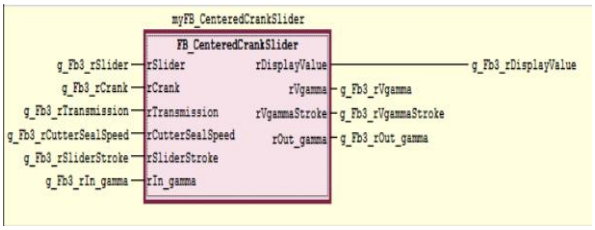
通过滑块、曲柄、传动的初始位置，结合刀封速度、给定行程值、减速机末端转动值，显示滑块位置值并计算当前末端转动值和速度值。

示意图:



1、指令格式

图形表现



ST 表现

```
myFB_CenteredCrankSlider(
    rSlider:= g_Fb3_rSlider,
    rCrank:= g_Fb3_rCrank,
    rTransmission:= g_Fb3_rTransmission,
    rCutterSealSpeed:= g_Fb3_rCutterSealSpeed,
    rSliderStroke:= g_Fb3_rSliderStroke,
    rIn_gamma:= g_Fb3_rIn_gamma,
    rDisplayValue=> g_Fb3_rDisplayValue,
    rVgamma=> g_Fb3_rVgamma,
    rVgammaStroke=> g_Fb3_rVgammaStroke,
    rOut_gamma=> g_Fb3_rOut_gamma);
```

2、功能块变量说明

（1）输入变量（VAR_Input）

变量名称	数据类型	默认值	有效范围	描述
rSlider	LREAL	300	/	滑块的初始位置
rCrank	LREAL	400	/	曲柄的初始位置
rTransmission	LREAL	500	/	传动连杆的初始位置
rCutterSealSpeed	LREAL	/	/	刀封速度
rSliderStroke	LREAL	/	/	上位机给的行程
rIn_gamma	LREAL	/	/	减速机末端转动值

（2）输出变量（VAR_Output）

输出变量	数据类型	默认值	有效范围	描述
rDisplayValue	LREAL	/	/	显示滑块的位置
rVgamma	LREAL	/	/	点动时减速机速度
rVgammaStroke	LREAL	/	/	绝对运动时减速机速度
rOut_gamma	LREAL	/	/	减速机末端转动值

3、使用样例

（1）声明与调用

```
g_Fb2_rOut_theta:LREAL;//减速机末端转动值
g_Fb2_rInitial_Angle_theta1: LREAL;  //初始弧度
g_Fb2_rStroke_Angle_theta2: LREAL;  //行程弧度

g_Fb3_rSlider:REAL:=300;//滑块初始长度
g_Fb3_rCrank: REAL := 400;//曲柄初始长度
g_Fb3_rTransmission:REAL:=500;//传动初始长度
g_Fb3_rCutterSealSpeed:REAL;//刀封速度
g_Fb3_rSliderStroke:REAL;    //上位机给的行程
g_Fb3_rIn_gamma:REAL;        //减速机末端转动值

g_Fb3_rDisplayValue:REAL;  //显示滑块的位置
g_Fb3_rVgamma:REAL;//点动时减速机速度
g_Fb3_rVgammaStroke:REAL;//绝对运动时减速机速度
g_Fb3_rOut_gamma:REAL;//减速机末端转动值


myFB_CenteredCrankSlider(
    rSlider:= g_Fb3_rSlider,
    rCrank:= g_Fb3_rCrank,
    rTransmission:= g_Fb3_rTransmission,
    rCutterSealSpeed:= g_Fb3_rCutterSealSpeed,
    rSliderStroke:= g_Fb3_rSliderStroke,
    rIn_gamma:= g_Fb3_rIn_gamma,
    rDisplayValue=> g_Fb3_rDisplayValue,
    rVgamma=> g_Fb3_rVgamma,
    rVgammaStroke=> g_Fb3_rVgammaStroke,
    rOut_gamma=> g_Fb3_rOut_gamma);
```

（2）功能详细演示部分

```
myFB_CenteredCrankSlider(
    rSlider 300 := g_Fb3_rSlider 300,
    rCrank 400 := g_Fb3_rCrank 400,
    rTransmission 500 := g_Fb3_rTransmission 500,
    rCutterSealSpeed 10 := g_Fb3_rCutterSealSpeed 10,
    rSliderStroke 350 := g_Fb3_rSliderStroke 350,
    rIn_gamma 300 := g_Fb3_rIn_gamma 300,
    rDisplayValue -200 => g_Fb3_rDisplayValue -200,
    rVgamma 0.628 => g_Fb3_rVgamma 0.628,
    rVgammaStroke -0.032 => g_Fb3_rVgammaStroke -0.032,
    rOut_gamma 0.694 => g_Fb3_rOut_gamma 0.694 );
```

fb_UpperCrankSlider_Offset 上偏心曲柄滑块机构的运算

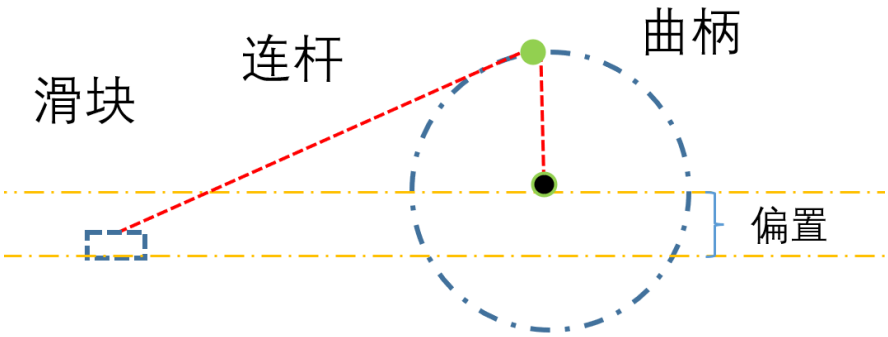
功能块解释说明：

通过滑块、曲柄、传动的初始位置，结合偏置值，计算当前滑块位置下的末端转动值。

使用场景描述：

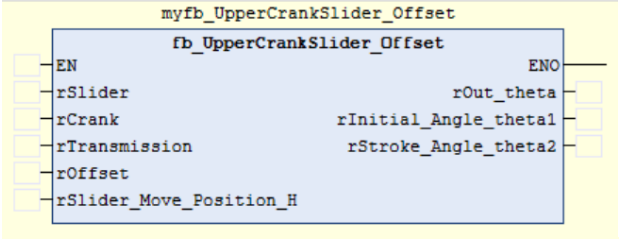
偏置在上曲柄滑块。曲柄旋转中心在滑块移动的水平线以上。

示意图：



1、指令格式

图形表现



ST 表现

```
myfb_UpperCrankSlider_Offset(
    rSlider:=  ,
    rCrank:= ,
    rTransmission:=  ,
    rOffset:=  ,
    rSlider_Move_Position_H:=  ,
    rOut_theta=>  ,
    rInitial_Angle_theta1=>  ,
    rStroke_Angle_theta2=> );
```

2、功能块变量说明

（1）输入变量（VAR_Input）

变量名称	数据类型	默认值	有效范围	描述
rSlider	LREAL	300	/	滑块的初始位置
rCrank	LREAL	200	/	曲柄的初始位置
rTransmission	LREAL	500	/	传动的初始位置
rOffset	LREAL	200	/	偏置值
rSlider_Move_Position_H	LREAL	/	/	滑块移动位置

（2）输出变量（VAR_Output）

输出变量	数据类型	默认值	有效范围	描述
rOut_theta	LREAL	/	/	减速机末端转动值
rInitial_Angle_theta1	LREAL	/	/	初始弧度
rStroke_Angle_theta2	LREAL	/	/	行程弧度

3、使用样例

（1）声明与调用

```
g_Fbl_rSlider:LREAL :=300;     //滑块的初始位置
g_Fbl_rCrank:LREAL :=200;       //曲柄的初始位置
g_Fbl_rTransmission:LREAL :=500;        //传动的初始位置
g_Fbl_rOffset: LREAL:= 200;     //偏置值
g_Fbl_rSlider_Move_Position_H: LREAL;//滑块移动位置

g_Fbl_rOut_theta:LREAL;//减速机末端转动值
g_Fbl_rInitial_Angle_theta1: LREAL;  //初始弧度
g_Fbl_rStroke_Angle_theta2: LREAL;  //行程弧度


myfb_UpperCrankSlider_Offset(
    rSlider:= g_Fbl_rSlider,
    rCrank:= g_Fbl_rCrank,
    rTransmission:= g_Fbl_rTransmission,
    rOffset:= g_Fbl_rOffset,
    rSlider_Move_Position_H:= g_Fbl_rSlider_Move_Position_H,
    rOut_theta=> g_Fbl_rOut_theta,
    rInitial_Angle_theta1=> g_Fbl_rInitial_Angle_theta1,
    rStroke_Angle_theta2=> g_Fbl_rStroke_Angle_theta2);
```

（2）功能演示部分

```
myfb_UpperCrankSlider_Offset(
    rSlider 300 := g_Fbl_rSlider 300,
    rCrank 200 := g_Fbl_rCrank 200,
    rTransmission 500 := g_Fbl_rTransmission 500,
    rOffset 200 := g_Fbl_rOffset 200,
    rSlider_Move_Position_H 10 := g_Fbl_rSlider_Move_Position_H 10,
    rOut_theta 2.78 => g_Fbl_rOut_theta 2.78,
    rInitial_Angle_theta1 1.57 => g_Fbl_rInitial_Angle_theta1 1.57,
    rStroke_Angle_theta2 1.62 => g_Fbl_rStroke_Angle_theta2 1.62 );
```

FB_LowerCrankSlider_Offset 下偏心曲柄滑块机构的运算

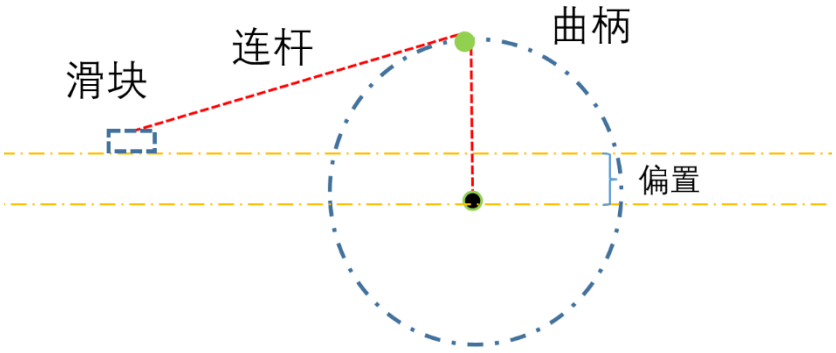
功能块解释说明：

通过滑块、曲柄、传动的初始位置，结合偏置值，计算当前滑块位置下的末端转动值。

使用场景描述：

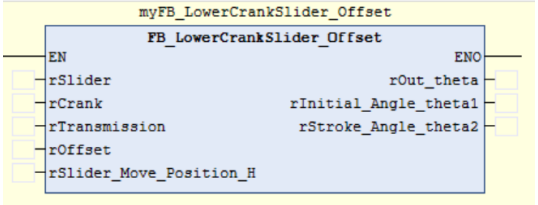
偏置在下曲柄滑块。滑块移动的水平线，与曲柄旋转中心不在一起，旋转中心在水平线以下。

示意图：



1、指令格式

图形表现



ST 表现

```
myFB_LowerCrankSlider_Offset(
    rSlider:=  ,
    rCrank:=  ,
    rTransmission:=  ,
    rOffset:=  ,
    rSlider_Move_Position_H:=  ,
    rOut_theta=>  ,
    rInitial_Angle_theta1=>  ,
    rStroke_Angle_theta2=>  );
```

2、功能块变量说明

（1）输入变量（VAR_Input）

变量名称	数据类型	默认值	有效范围	描述
rSlider	LREAL	300	/	滑块的初始位置
rCrank	LREAL	200	/	曲柄的初始位置
rTransmission	LREAL	500	/	传动的初始位置
rOffset	LREAL	200	/	偏置值
rSlider_Move_Position_H	LREAL	/	/	滑块移动位置

（2）输出变量（VAR_Output）

输出变量	数据类型	默认值	有效范围	描述
rOut_theta	LREAL	/	/	减速机末端转动值
rInitial_Angle_theta1	LREAL	/	/	初始弧度
rStroke_Angle_theta2	LREAL	/	/	行程弧度

3、使用样例

（1）声明与调用

```
g_Fb1_rSlider:LREAL :=300;    //滑块的初始位置
g_Fb1_rCrank:LREAL :=200;      //曲柄的初始位置
g_Fb1_rTransmission:LREAL :=500;        //传动的初始位置
g_Fb1_rOffset: LREAL:= 200;     //偏置值
g_Fb1_rSlider_Move_Position_H: LREAL;//滑块移动位置

g_Fb1_rOut_theta:LREAL;//减速机末端转动值
g_Fb1_rInitial_Angle_theta1: LREAL;  //初始弧度
g_Fb1_rStroke_Angle_theta2: LREAL;  //行程弧度

g_Fb2_rSlider:LREAL :=300;    //滑块的初始位置
g_Fb2_rCrank:LREAL :=200;      //曲柄的初始位置
g_Fb2_rTransmission:LREAL :=500;        //传动的初始位置
g_Fb2_rOffset: LREAL:= 200;     //偏置值
g_Fb2_rSlider_Move_Position_H: LREAL;//滑块移动位置
```

```
myFB_LowerCrankSlider_Offset(
    rSlider:= g_Fb2_rSlider,
    rCrank:= g_Fb2_rCrank,
    rTransmission:= g_Fb2_rTransmission,
    rOffset:= g_Fb2_rOffset,
    rSlider_Move_Position_H:= g_Fb2_rSlider_Move_Position_H,
    rOut_theta=> g_Fb2_rOut_theta,
    rInitial_Angle_theta1=> g_Fb2_rInitial_Angle_theta1,
    rStroke_Angle_theta2=> g_Fb2_rStroke_Angle_theta2);
```

（2）功能演示部分

```
myFB_LowerCrankSlider_Offset(
    rSlider 300 := g_Fb2_rSlider 300,
    rCrank 200 := g_Fb2_rCrank 200,
    rTransmission 500 := g_Fb2_rTransmission 500,
    rOffset 200 := g_Fb2_rOffset 200,
    rSlider_Move_Position_H 10 := g_Fb2_rSlider_Move_Position_H 10,
    rOut_theta 4.5 => g_Fb2_rOut_theta 4.5,
    rInitial_Angle_thetal 0.395 => g_Fb2_rInitial_Angle_thetal 0.395,
    rStroke_Angle_theta2 0.473 => g_Fb2_rStroke_Angle_theta2 0.473 );
```

05_自定义凸轮

051 样条曲线

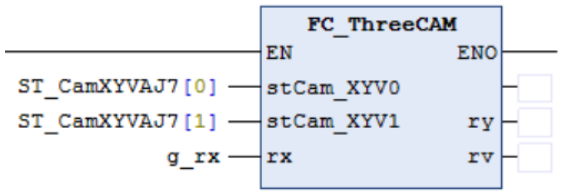
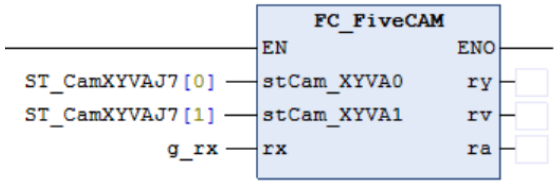
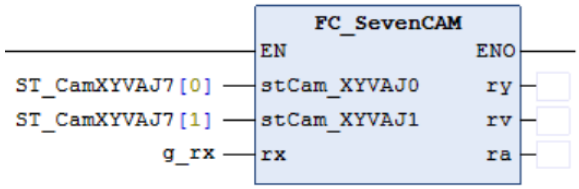
FC_Three/Five/SevenCAM 两点 XYVA 三次/五次/七次曲线运算

功能块解释说明：

已知首点和末点的 XYVAJ，求解 3 次，5 次，7 次函数关系下，X 对应的 Y，V。

使用场景描述：

运动控制中的跟随曲线拟合、曲线速度求解等。

图形表现：	ST 表现：
	FC_ThreeCAM(stCam_XYV0:= ST_CamXYVAJ7[0], stCam_XYV1:=ST_CamXYVAJ7[1] , rx:= g_rx, ry=> g_ry[3] , rv=>g_rv[3]);
	FC_FiveCAM(stCam_XYVa0:= ST_CamXYVAJ7[0], stCam_XYVa1:=ST_CamXYVAJ7[1] , rx:= g_rx, ry=> g_ry[4] , rv=>g_rv[4]);
	FC_SevenCAM(stCam_XYVaj0:= ST_CamXYVAJ7[0], stCam_XYVaj1:=ST_CamXYVAJ7[1] , rx:= g_rx, ry=> g_ry[5] , rv=>g_rv[5]);

2、功能块变量说明

（1）输入输出变量（Var In_Out）

无

（2）输入变量（VAR_Input）

变量名称	数据类型	默认值	有效范围	描述
stCam_XYVAJ0	ST_CamXYVAJ	/	/	曲线的首点
stCam_XYVAJ1	ST_CamXYVAJ	/	/	曲线的末点
rx	LREAL	/	/	X 坐标位置，要在首末点之间

（3）输出变量（VAR_Output）

输出变量	数据类型	默认值	有效范围	描述
ry	LREAL	/		根据表达式和 X 求出的 Y
rv	LREAL	/		根据表达式和 X 求出的比例

（4）自定义数据类型解释（结构体、枚举等）

结构体名称：ST_CamXYVAJ

变量名称	数据类型	初始值	说明
dX	LREAL	0	X 坐标位置
dY	LREAL	0	Y 坐标位置
dV	LREAL	0	y/x 的比例
dA	LREAL	0	v/x 的比例
dJ	LREAL	0	a/x 的比例

3、使用样例

（1）声明与调用

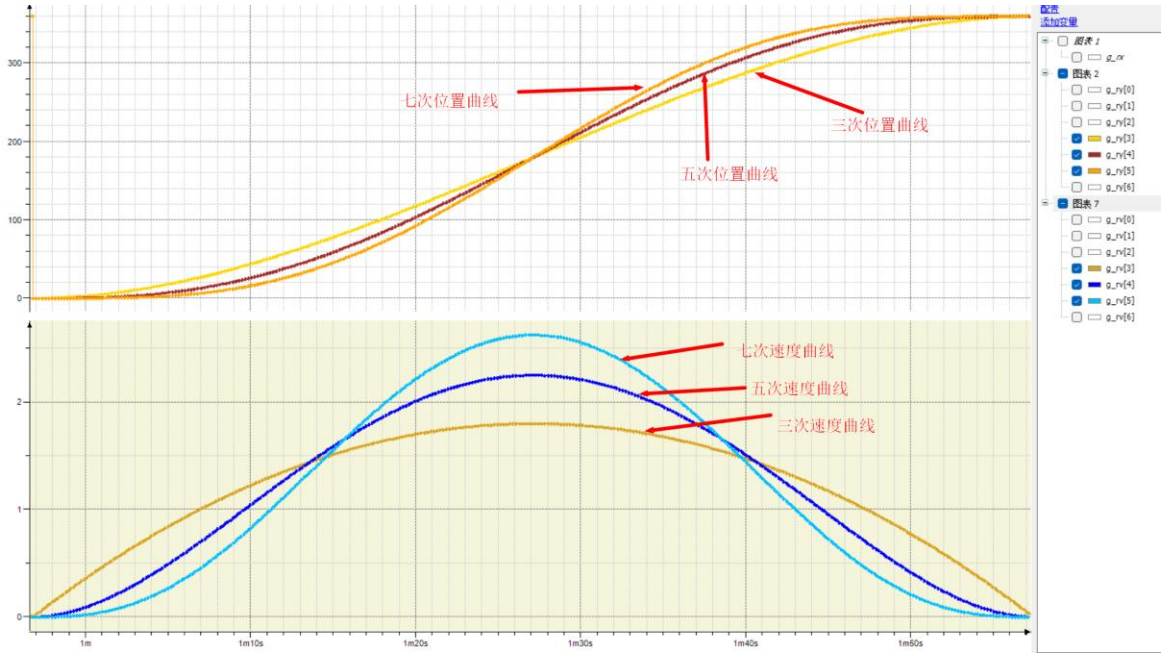
PLC_PRG

```
PROGRAM PLC_PRG
VAR
    ST_CamXYVAJ7 : ARRAY[0..1] OF ST_CamXYVAJ;
    x_runexe     : BOOL;
END_VAR
```

```
//样条曲线函数测试

ST_CamXYVAJ7[0].dX:=0 ;ST_CamXYVAJ7[0].dY:=0 ;
ST_CamXYVAJ7[0].dV:=0 ;ST_CamXYVAJ7[0].dA:=0 ;
ST_CamXYVAJ7[0].dj:=0 ;

ST_CamXYVAJ7[1].dX:=300 ;ST_CamXYVAJ7[1].dY:=360 ;
ST_CamXYVAJ7[1].dV:=0 ;ST_CamXYVAJ7[1].dA:=0 ;
ST_CamXYVAJ7[1].dj:=0 ;

FC_ThreeCAM(
stCam_XYV0:= ST_CamXYVAJ7[0],
 stCam_XYV1:=ST_CamXYVAJ7[1] ,
rx:= g_rx, ry=> g_ry[3] , rv=>g_rv[3] );

FC_FiveCAM(
stCam_XYVa0:= ST_CamXYVAJ7[0],
stCam_XYVa1:=ST_CamXYVAJ7[1] ,
rx:= g_rx, ry=> g_ry[4] , rv=>g_rv[4] );

FC_SevenCAM(
stCam_XYVaj0:= ST_CamXYVAJ7[0],
 stCam_XYVaj1:=ST_CamXYVAJ7[1] ,
 rx:= g_rx, ry=> g_ry[5] , rv=>g_rv[5] );
```

（2）功能详细演示部分

```
//样条曲线函数测试

ST_CamXYVAJ7[0].dX 0 :=0 ;ST_CamXYVAJ7[0].dY 0 :=0 ;
ST_CamXYVAJ7[0].dV 0 :=0 ;ST_CamXYVAJ7[0].dA 0 :=0 ;
ST_CamXYVAJ7[0].dj 0 :=0 ;

ST_CamXYVAJ7[1].dX 300 :=300 ;ST_CamXYVAJ7[1].dY 360 :=360 ;
ST_CamXYVAJ7[1].dV 0 :=0 ;ST_CamXYVAJ7[1].dA 0 :=0 ;
ST_CamXYVAJ7[1].dj 0 :=0 ;

FC_ThreeCAM(
stCam_XYV0:= ST_CamXYVAJ7[0],
stCam_XYV1:=ST_CamXYVAJ7[1] ,
rx:= g_rx 13.9 , ry=> g_ry[3] 2.23 ,rv=>g_rv[3] 0.317 );

FC_FiveCAM(
stCam_XYVa0:= ST_CamXYVAJ7[0],
stCam_XYVa1:=ST_CamXYVAJ7[1] ,
rx:= g_rx 13.9 , ry=> g_ry[4] 0.33 , rv=>g_rv[4] 0.0698 );

FC_SevenCAM(
stCam_XYVaj0:= ST_CamXYVAJ7[0],
stCam_XYVaj1:=ST_CamXYVAJ7[1] ,
rx:= g_rx 13.9 , ry=> g_ry[5] 0.0511 ,rv=>g_rv[5] 0.0143 );



IF x_runexe TRUE THEN
    IF g_rx 13.9 <300 THEN
        g_rx 13.9 :=g_rx 13.9 +0.05;
    ELSE
        g_rx 13.9 :=0;
    END_IF
END_IF
```



FC_CosAcc/SinAcc/TvCAM 加速曲线规划（Sin/Cos/T 型）

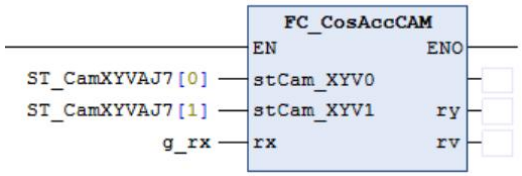
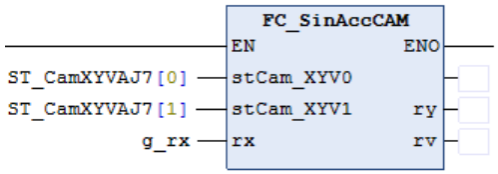
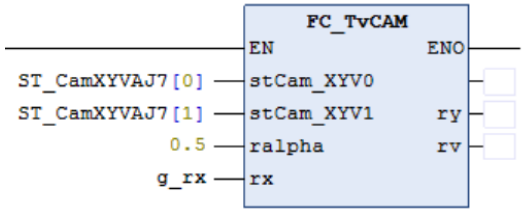
- **功能块解释说明：**

  已知首点和末点的 XYVAJ，求解三角函数加速关系下，X 对应的 Y。

  已知首点和末点的 XYVAJ，以及匀速区占比，求解梯形速度函数关系下，X 对应的 Y，V。

- **使用场景描述：**

  运动控制中的跟随曲线拟合、曲线速度求解等。

图形表现：	ST 表现：
	FC_CosAccCAM(stCam_XYV0:= ST_CamXYVAJ7[0], stCam_XYV1:=ST_CamXYVAJ7[1] , rx:= g_rx, ry=> g_ry[1] , rv=>g_rv[1]);
	FC_SinAccCAM(stCam_XYV0:= ST_CamXYVAJ7[0], stCam_XYV1:=ST_CamXYVAJ7[1] , rx:= g_rx, ry=> g_ry[2] , rv=>g_rv[2]);
	FC_TvCAM(stCam_XYV0:= ST_CamXYVAJ7[0], stCam_XYV1:=ST_CamXYVAJ7[1] , ralpha:=0.5, rx:= g_rx, ry=> g_ry[6] , rv=>g_rv[6]);

2、功能块变量说明

（1）输入变量（VAR_Input）

变量名称	数据类型	默认值	有效范围	描述
stCam_XYVAJ0	ST_CamXYVAJ	/	/	曲线的首点
stCam_XYVAJ1	ST_CamXYVAJ	/	/	曲线的末点
rx	LREAL	/	/	X 坐标位置，要在首末点之间
ralpha	LREAL	0.5	/	T 型速度曲线匀速区比例

（2）输出变量（VAR_Output）

输出变量	数据类型	默认值	有效范围	描述

ry	LREAL	/	/	根据表达式和 X 求出的 Y
rv	LREAL	/	/	根据表达式和 X 求出的比例

（3）自定义数据类型解释（结构体、枚举等）

结构体名称：ST_CamXYVAJ

变量名称	数据类型	初始值	说明
dX	LREAL	0	X 坐标位置
dY	LREAL	0	Y 坐标位置
dV	LREAL	0	y/x 的比例
dA	LREAL	0	v/x 的比例
dJ	LREAL	0	a/x 的比例
ralpha	LREAL	0.5	T 型速度曲线匀速区比例

3、使用样例

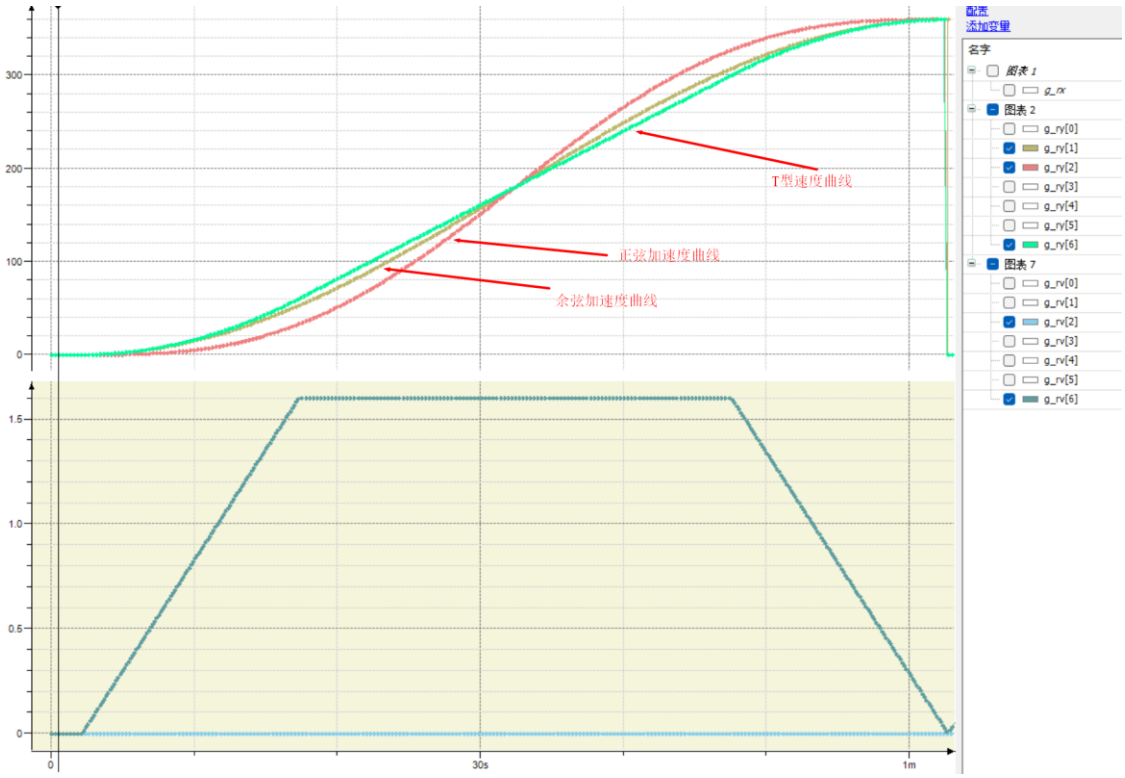
（1）声明与调用

```
PROGRAM PLC_PRG
VAR
    ST_CamXYVAJ7 : ARRAY[0..1] OF ST_CamXYVAJ;
    x_runexe       : BOOL;
END_VAR
```

```
//样条曲线函数测试

ST_CamXYVAJ7[0].dX:=0 ;ST_CamXYVAJ7[0].dY:=0 ;
ST_CamXYVAJ7[0].dV:=0 ;ST_CamXYVAJ7[0].dA:=0 ;
ST_CamXYVAJ7[0].dj:=0 ;

ST_CamXYVAJ7[1].dX:=300 ;ST_CamXYVAJ7[1].dY:=360 ;
ST_CamXYVAJ7[1].dV:=0 ;ST_CamXYVAJ7[1].dA:=0 ;
ST_CamXYVAJ7[1].dj:=0 ;

FC_CosAccCAM(stCam_XYV0:= ST_CamXYVAJ7[0], stCam_XYV1:=ST_CamXYVAJ7[1] ,
 rx:= g_rx, ry=> g_ry[1] , rv=>g_rv[1] );

FC_SinAccCAM(stCam_XYV0:= ST_CamXYVAJ7[0], stCam_XYV1:=ST_CamXYVAJ7[1] ,
rx:= g_rx, ry=> g_ry[2] , rv=>g_rv[2] );

FC_TvCAM(stCam_XYV0:= ST_CamXYVAJ7[0], stCam_XYV1:=ST_CamXYVAJ7[1] ,
ralpha:=0.5, rx:= g_rx, ry=> g_ry[6] , rv=>g_rv[6] );

IF x_runexe THEN
    IF g_rx<300 THEN
        g_rx :=g_rx+0.05;
    ELSE
        g_rx:=0;
    END_IF
END_IF
```

（2）功能详细演示部分

```
FC_CosAccCAM(
stCam_XYV0:= ST_CamXYVAJ7[0],
stCam_XYV1:=ST_CamXYVAJ7[1] ,
rx:= g_rx 29.5 , ry=> g_ry[1] 8.52 ,
rv=>g_rv[1] 54.7 );

FC_SinAccCAM(
stCam_XYV0:= ST_CamXYVAJ7[0],
stCam_XYV1:=ST_CamXYVAJ7[1] ,
rx:= g_rx 29.5 , ry=> g_ry[2] 2.21 ,
rv=>g_rv[2] 0 );

FC_TvCAM(
stCam_XYV0:= ST_CamXYVAJ7[0],
stCam_XYV1:=ST_CamXYVAJ7[1] ,
ralpha:=0.5, rx:= g_rx 29.5 ,
ry=> g_ry[6] 9.28 , rv=>g_rv[6] 0.629 );

IF x_runexe TRUE THEN
    IF g_rx 29.5 <300 THEN
        g_rx 29.5 :=g_rx 29.5 +0.05;
    ELSE
        g_rx 29.5 :=0;
    END_IF
END_IF
```



FC_LineCAM 两点 XYVA 直线运算

待发

052 跟随实现

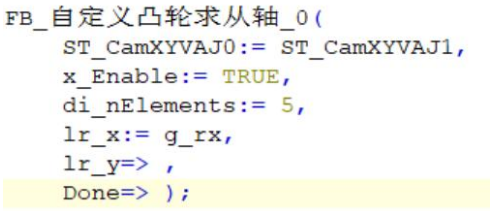
FB_FollowCam_fl 自定义凸轮主轴求从轴

- **功能块解释说明：**

  已知 FollowCam 凸轮表的各关键点，通过 X 求解 Y。

- **使用场景描述：**

  运动控制中的已知主轴位置，需要求解从轴位置的场合。

图形表现：	ST 表现：
	


```
FB_自定义凸轮求从轴_0(
    ST_CamXYVAJ0:= ST_CamXYVAJ1,
    x_Enable:= TRUE,
    di_nElements:= 5,
    lr_x:= g_rx,
    lr_y=> ,
    Done=> );
```

2、功能块变量说明

（1）输入输出变量（Var In_Out）

变量名称	数据类型	默认值	有效范围	描述
ST_CamXYVAJ0	ARRAY[*] OF ST_CamXYVAJ	/	/	凸轮表中的各关键点。

（2）输入变量（VAR_Input）

变量名称	数据类型	默认值	有效范围	描述
x_Enable	BOOL	/	/	使能功能块有效
di_nElements	DINT	/	/	凸轮表关键点个数
lr_x	LREAL	/	/	X 坐标值

（3）输出变量（VAR_Output）

输出变量	数据类型	默认值	有效范围	描述
lr_y	LREAL	/	/	求得的凸轮表中的 Y
Done	BOOL	/	/	求解完成

（4）自定义数据类型解释（结构体、枚举等）

结构体名称：ST_CamXYVAJ

变量名称	数据类型	初始值	说明
dX	LREAL	0	X 坐标位置
dY	LREAL	0	Y 坐标位置
dV	LREAL	0	y/x 的比例
dA	LREAL	0	v/x 的比例
dJ	LREAL	0	a/x 的比例

3、使用样例

（1）声明与调用

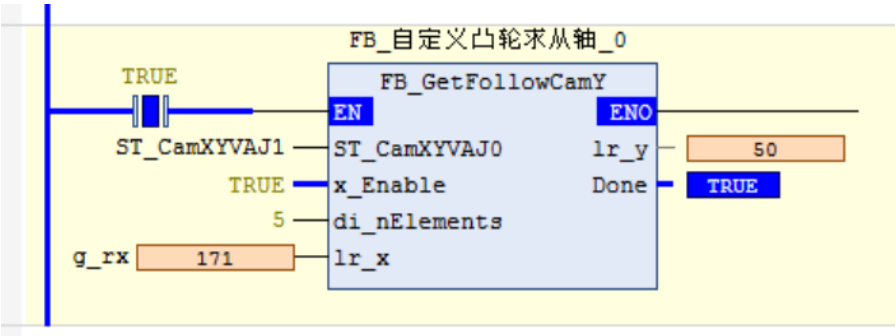
曲线显示：

```
//第一条曲线

ST_CamXYVAJ1[0].dX:=0 ;
ST_CamXYVAJ1[0].dY:=0 ;
ST_CamXYVAJ1[0].dV:=0 ;
ST_CamXYVAJ1[0].dA:=0 ;
ST_CamXYVAJ1[0].eCamType:=E_CamType.eFivePos ;

ST_CamXYVAJ1[1].dX:=100 ;
ST_CamXYVAJ1[1].dY:=50 ;
ST_CamXYVAJ1[1].dV:=0 ;
ST_CamXYVAJ1[1].dA:=0 ;
ST_CamXYVAJ1[1].eCamType:=E_CamType.eFivePos ;

ST_CamXYVAJ1[2].dX:=200 ;
ST_CamXYVAJ1[2].dY:=50 ;
ST_CamXYVAJ1[2].dV:=0 ;
ST_CamXYVAJ1[2].dA:=0 ;
ST_CamXYVAJ1[2].eCamType:=E_CamType.eFivePos ;

ST_CamXYVAJ1[3].dX:=300 ;
ST_CamXYVAJ1[3].dY:=0 ;
ST_CamXYVAJ1[3].dV:=0 ;
ST_CamXYVAJ1[3].dA:=0 ;
ST_CamXYVAJ1[3].eCamType:=E_CamType.eFivePos ;

ST_CamXYVAJ1[4].dX:=360 ;
ST_CamXYVAJ1[4].dY:=0 ;
ST_CamXYVAJ1[4].dV:=0 ;
ST_CamXYVAJ1[4].dA:=0 ;
ST_CamXYVAJ1[4].eCamType:=E_CamType.eLine ;
```

```
    g_rx         :LREAL :=0;
    ST_CamXYVAJ1 : ARRAY[0..4] OF ST_CamXYVAJ;
    FB_自定义凸轮求从轴_0: FB_GetFollowCamY;
END_VAR
```

```
FB_自定义凸轮求从轴_0(
    ST_CamXYVAJ0:= ST_CamXYVAJ1,
    x_Enable:= TRUE,
    di_nElements:= 5,
    lr_x:= g_rx,
    lr_y=> ,
    Done=> );
```

（2）功能详细演示部分



FB_GetFollowCamY 自定义凸轮控制（follow）

- **功能块解释说明：**

  FB_FollowCam_fl 功能块采用基础的的位置跟随功能，用在非常规场合下，凸轮点间连接曲线需要用到三角函数加速度，多次方曲线的，功能块包含了：凸轮表间的实时动态或末点切换功能，非末端凸轮点数据实时修改，末端凸轮点周期末点修改，凸轮表主轴启动距离，以及其他常规的缩放，偏移等功能。

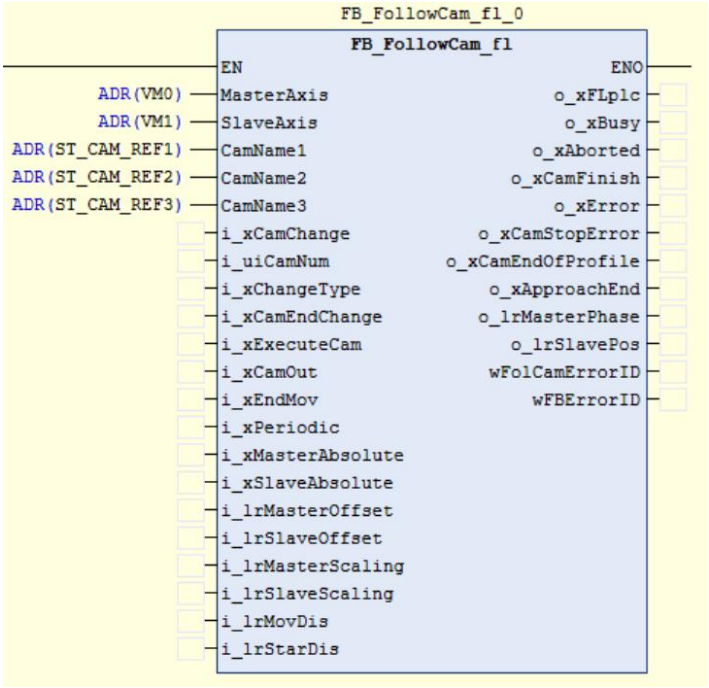
- **使用场景描述：**

  用在非常规场合下，凸轮点间连接曲线需要用到三角函数加速度，多次方曲线的。

- **其他注释：**

  注意，主轴不可以反转，梯形速度曲线目前屏蔽。

1、指令格式

图形表现：	ST 表现：
	FB_FollowCam_fl_0(MasterAxis:= , SlaveAxis:= , CamName1:= , CamName2:= , CamName3:= , i_xCamChange:= , i_uiCamNum:= , i_xChangeType:= , i_xCamEndChange:= , i_xExecuteCam:= , i_xCamOut:= , i_xEndMov:= , i_xPeriodic:= , i_xMasterAbsolute:= , i_xSlaveAbsolute:= , i_lrMasterOffset:= , i_lrSlaveOffset:= , i_lrMasterScaling:= , i_lrSlaveScaling:= , i_lrMovDis:= , i_lrStarDis:= , o_xFLplc=> , o_xBusy=> , o_xAborted=> , o_xCamFinish=> , o_xError=> , o_xCamStopError=> , o_xCamEndOfProfile=> , o_xApproachEnd=> , o_lrMasterPhase=> , o_lrSlavePos=> , wFolCamErrorID=> , wFBErrorID=>);

2、功能块变量说明

（1）输入变量（VAR_Input）

变量名称	数据类型	默认值	有效范围	描述
MasterAxis	POINTER TO AXIS_REF_SM3	/	/	主轴
SlaveAxis	POINTER TO AXIS_REF_SM3	/	/	从轴
CamName1	POINTER TO ST_CAM_REF	/		凸轮表 1
CamName2	POINTER TO ST_CAM_REF	/	/	凸轮表 2
CamName3	POINTER TO ST_CAM_REF	/	/	凸轮表 3
i_xCamChange	BOOL	false	/	ture:允许切换凸轮，false：不允许
i_uiCamNum	Uint	1	/	允许切换凸轮时， 编号对应上述凸轮
i_xChangeType	BOOL	false	/	ture:立即切换凸轮，false：周期末点切换凸轮
i_xCamEndChange	BOOL	/	/	更新凸轮表，修改凸轮表末点，且只能在凸轮末点的位置进行触发
i_xExecuteCam	BOOL	/	/	凸轮拟合
i_xCamOut	BOOL	/	/	凸轮脱离
i_xEndMov	BOOL	false	/	ture:非周期凸轮结束会走一段定位，false:非周期凸轮结束直接停止。多用在从轴单向运动的场合。

i_xPeriodic	BOOL	false	/	周期模式 < ture：周期>-
i_xMasterAbsolute	BOOL	false	/	主坐标系-< ture ： 绝对坐标>
i_xSlaveAbsolute	BOOL	false	/	从坐标系-< ture ： 绝对坐标>
i_lrMasterOffset	LREAL	0	/	主轴偏移
i_lrSlaveOffset	LREAL	0	/	从轴偏移
i_lrMasterScaling	LREAL	1	/	主轴缩放
i_lrSlaveScaling	LREAL	1	/	从轴缩放
i_lrMovDis	LREAL	10	/	非周期凸轮结束时定位距离
i_lrStarDis	LREAL	0	/	非周期凸轮启动时，主轴先走的距离,不能超过主轴模值

（2）输出变量（VAR_Output）

输出变量	数据类型	默认值	有效范围	描述
O_xFLplc	BOOL	/	/	繁易 PLC 标志
o_xBusy	BOOL	/	/	凸轮运行
o_xAborted	BOOL	/	/	凸轮被中断
o_xCamFinish	BOOL	/	/	凸轮结束
o_xError	BOOL	/	/	凸轮故障
o_xCamStopError	BOOL	/	/	凸轮停止故障（MC_Stop）
o_xCamEndOfProfile	BOOL	/	/	凸轮的周期终点
o_xApproachEnd	BOOL	/	/	接近曲线末点，在末点前 1-2 周期
o_lrMasterPhase	LREAL	/	/	凸轮主轴周期位置
o_lrSlavePos	LREAL	/	/	从轴在凸轮表里的位置。
wFolCamErrorID	FL_ERRORID			自定义功能块内部故障代码
wFBErrorID	SMC_ERROR	/	/	其他功能块故障代码

（3）自定义数据类型解释（结构体、枚举等）

结构体名称：ST_CamXYVAJ

变量名称	数据类型	初始值	说明
dX	LREAL	0	X 坐标位置
dY	LREAL	0	Y 坐标位置
dV	LREAL	0	y/x 的比例
dA	LREAL	0	v/x 的比例
dJ	LREAL	0	a/x 的比例
eCamType	E_CamType	0	关键点链接方式

结构体名称：ST_CAM_REF

变量名称	数据类型	初始值	说明
lrStart	LREAL	0	X 坐标位置
lrEnd	LREAL	0	Y 坐标位置
nElements	Dint	0	y/x 的比例
pce	POINTER TO BYTE	0	v/x 的比例

枚举名称：E_CamType

变量名称	值	说明
eLine	0	0 加速度曲线，速度是直线。
eCosAcc	1	余弦加速度曲线（*由于三角函数的特殊性，我们约定起末点速度为 0*）
eSinAcc	2	正弦加速度曲线　（*由于三角函数的特殊性，我们约定起末点速度为 0*）
eThreePos	3	3 次方位置曲线
eFivePos	4	5 次方位置曲线
eSevenPos	5	7 次方位置曲线

3、使用样例

（1）声明与调用

```
PROGRAM ETC_FLcam
VAR
    ST_CAM_REF1 :ST_CAM_REF;
    ST_CAM_REF2 :ST_CAM_REF;
    ST_CAM_REF3 :ST_CAM_REF;
    ST_CamXYVAJ1 : ARRAY[0..4] OF ST_CamXYVAJ;
    ST_CamXYVAJ2 : ARRAY[0..4] OF ST_CamXYVAJ;
    ST_CamXYVAJ3 : ARRAY[0..4] OF ST_CamXYVAJ;
    FB_FollowCam_fl_0: FB_FollowCam_fl;
END_VAR
```

（2）功能详细演示部分

演示：非周期下的主轴启动距离，定位停止，缩放，偏移功能。周期模式下的凸轮切换功能。

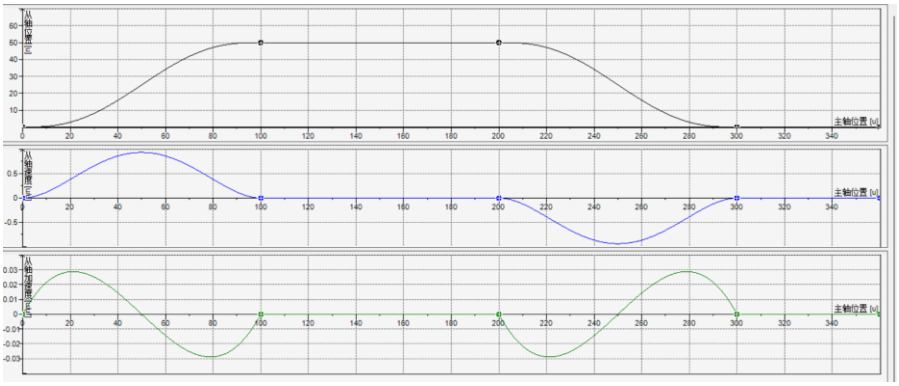
一：建立凸轮表：第一条为 5 次曲线，第二条为 3 次曲线，　第三条为 7 次曲线。

```
//第一条曲线
ST_CamXYVAJ1[0].dX:=0 ;
ST_CamXYVAJ1[0].dY:=0 ;
ST_CamXYVAJ1[0].dV:=0 ;
ST_CamXYVAJ1[0].dA:=0 ;
ST_CamXYVAJ1[0].eCamType:=E_CamType.eFivePos ;

ST_CamXYVAJ1[1].dX:=100 ;
ST_CamXYVAJ1[1].dY:=50 ;
ST_CamXYVAJ1[1].dV:=0 ;
ST_CamXYVAJ1[1].dA:=0 ;
ST_CamXYVAJ1[1].eCamType:=E_CamType.eFivePos ;

ST_CamXYVAJ1[2].dX:=200 ;
ST_CamXYVAJ1[2].dY:=50 ;
ST_CamXYVAJ1[2].dV:=0 ;
ST_CamXYVAJ1[2].dA:=0 ;
ST_CamXYVAJ1[2].eCamType:=E_CamType.eFivePos ;

ST_CamXYVAJ1[3].dX:=300 ;
ST_CamXYVAJ1[3].dY:=0 ;
ST_CamXYVAJ1[3].dV:=0 ;
ST_CamXYVAJ1[3].dA:=0 ;
ST_CamXYVAJ1[3].eCamType:=E_CamType.eFivePos ;

ST_CamXYVAJ1[4].dX:=360 ;
ST_CamXYVAJ1[4].dY:=0 ;
ST_CamXYVAJ1[4].dV:=0 ;
ST_CamXYVAJ1[4].dA:=0 ;
ST_CamXYVAJ1[4].eCamType:=E_CamType.eLine ;

ST_CAM_REF1.lrStart:=0;
ST_CAM_REF1.lrEnd:=360;
ST_CAM_REF1.nElements :=5;
ST_CAM_REF1.pce:=ADR(ST_CamXYVAJ1);
```

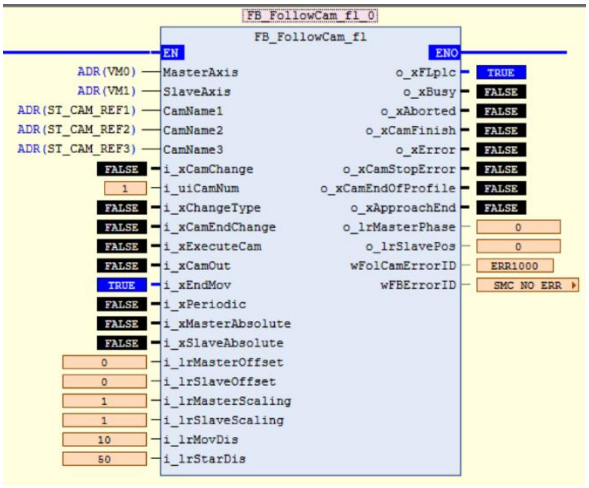
```
//第二条曲线
ST_CamXYVAJ2[0].dX:=0 ;
ST_CamXYVAJ2[0].dY:=0 ;
ST_CamXYVAJ2[0].dV:=0 ;
ST_CamXYVAJ2[0].dA:=0 ;
ST_CamXYVAJ2[0].eCamType:=E_CamType.eThreePos ;

ST_CamXYVAJ2[1].dX:=100 ;
ST_CamXYVAJ2[1].dY:=50 ;
ST_CamXYVAJ2[1].dV:=0 ;
ST_CamXYVAJ2[1].dA:=0 ;
ST_CamXYVAJ2[1].eCamType:=E_CamType.eLine ;

ST_CamXYVAJ2[2].dX:=200 ;
ST_CamXYVAJ2[2].dY:=50 ;
ST_CamXYVAJ2[2].dV:=0 ;
ST_CamXYVAJ2[2].dA:=0 ;
ST_CamXYVAJ2[2].eCamType:=E_CamType.eThreePos ;

ST_CamXYVAJ2[3].dX:=300 ;
ST_CamXYVAJ2[3].dY:=0 ;
ST_CamXYVAJ2[3].dV:=0 ;
ST_CamXYVAJ2[3].dA:=0 ;
ST_CamXYVAJ2[3].eCamType:=E_CamType.eLine ;

ST_CamXYVAJ2[4].dX:=360 ;
ST_CamXYVAJ2[4].dY:=0 ;
ST_CamXYVAJ2[4].dV:=0 ;
ST_CamXYVAJ2[4].dA:=0 ;
ST_CamXYVAJ2[4].eCamType:=E_CamType.eLine ;

ST_CAM_REF2.lrStart:=0;
ST_CAM_REF2.lrEnd:=360;
ST_CAM_REF2.nElements :=5;
ST_CAM_REF2.pce:=ADR(ST_CamXYVAJ2);
```

```
//第三条曲线
ST_CamXYVAJ3[0].dX:=0 ;
ST_CamXYVAJ3[0].dY:=0 ;
ST_CamXYVAJ3[0].dV:=0 ;
ST_CamXYVAJ3[0].dA:=0 ;
ST_CamXYVAJ3[0].eCamType:=E_CamType.eSevenPos ;

ST_CamXYVAJ3[1].dX:=100 ;
ST_CamXYVAJ3[1].dY:=50 ;
ST_CamXYVAJ3[1].dV:=0 ;
ST_CamXYVAJ3[1].dA:=0 ;
ST_CamXYVAJ3[1].eCamType:=E_CamType.eLine ;

ST_CamXYVAJ3[2].dX:=200 ;
ST_CamXYVAJ3[2].dY:=50 ;
ST_CamXYVAJ3[2].dV:=0 ;
ST_CamXYVAJ3[2].dA:=0 ;
ST_CamXYVAJ3[2].eCamType:=E_CamType.eSevenPos ;

ST_CamXYVAJ3[3].dX:=300 ;
ST_CamXYVAJ3[3].dY:=0 ;
ST_CamXYVAJ3[3].dV:=0 ;
ST_CamXYVAJ3[3].dA:=0 ;
ST_CamXYVAJ3[3].eCamType:=E_CamType.eLine ;

ST_CamXYVAJ3[4].dX:=360 ;
ST_CamXYVAJ3[4].dY:=0 ;
ST_CamXYVAJ3[4].dV:=0 ;
ST_CamXYVAJ3[4].dA:=0 ;
ST_CamXYVAJ3[4].eCamType:=E_CamType.eLine ;

ST_CAM_REF3.lrStart:=0;
ST_CAM_REF3.lrEnd:=360;
ST_CAM_REF3.nElements :=5;
ST_CAM_REF3.pce:=ADR(ST_CamXYVAJ3);
```

第一条曲线展示：

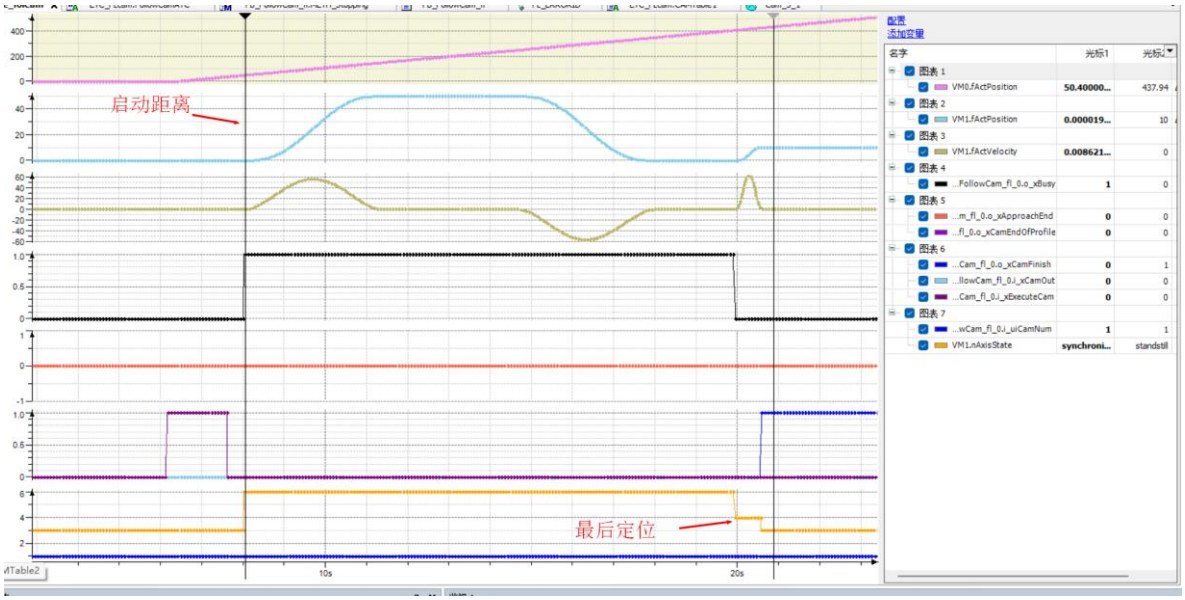


二：关联凸轮表

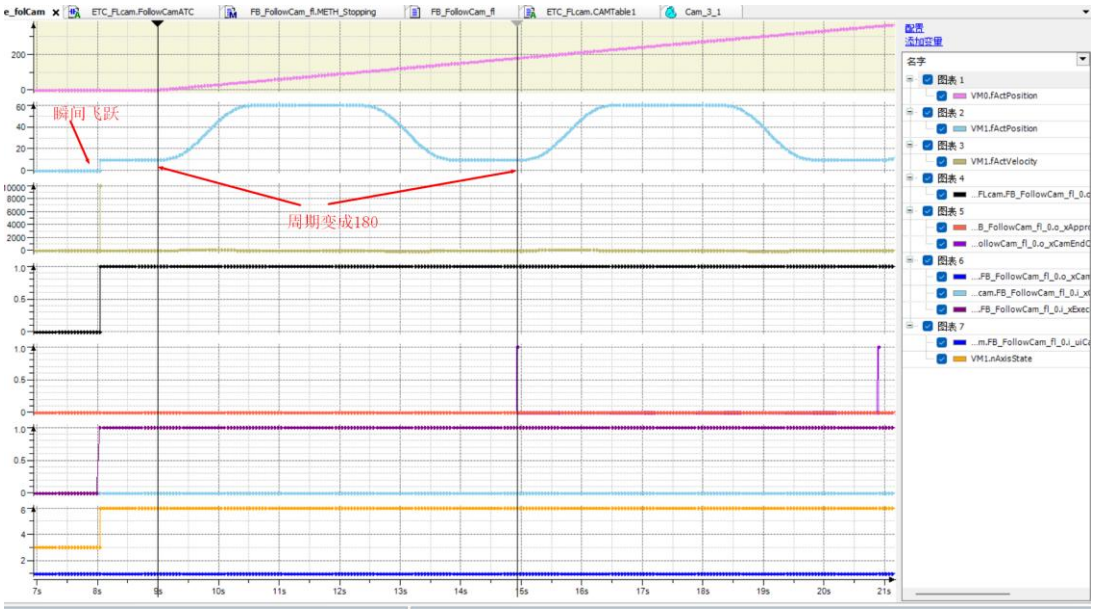


三：进行测试：

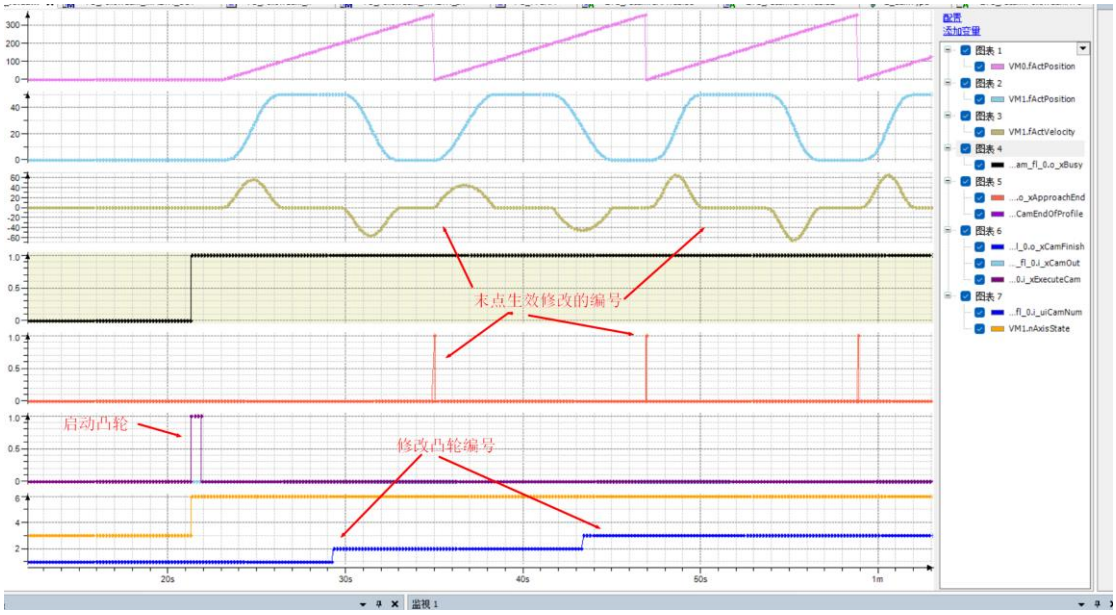
启动距离，定位停止功能：



缩放，偏移，主轴缩放 2 倍，从轴偏移 10。



周期模式下的凸轮切换：先跑 5 次曲线，后面切换 3 次曲线，最后切到 7 次曲线。



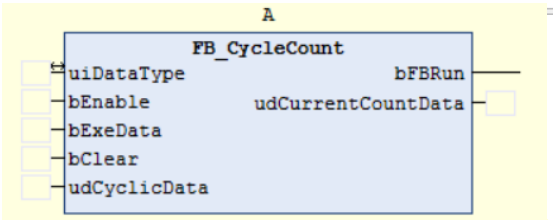
11_光伏行业

FB_CycleCount 循环计数

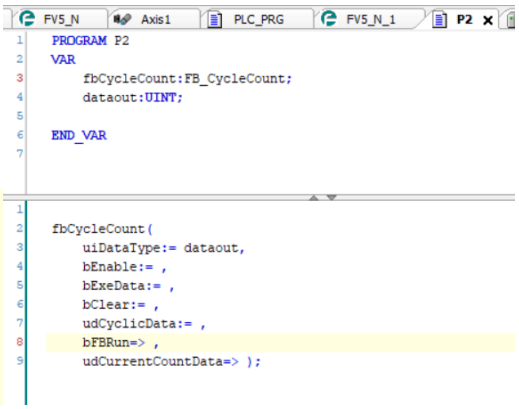
功能块说明：光伏行业，串焊拉丝叠片循环计数。
拉丝长度分为正常丝和串间丝，叠片位置也分为正常放片和串间放片。通过对计数和当前值的判断，输出目前是用于正常还是串间。=1 正常，=2 串间

1、指令格式

图形表现



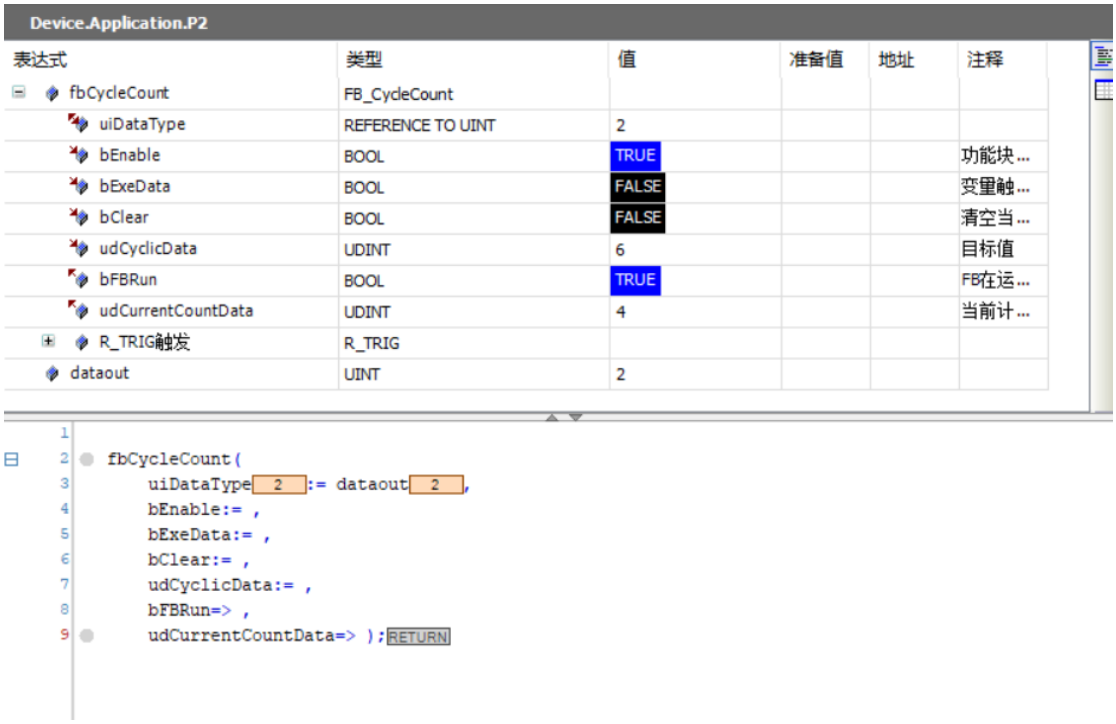
ST 表现



2、相关变量说明

变量名称	名称	数据类型	变量属性	描述
uiDataType	输出类型	UINT	输入输出变量	=1 正常，=2 串间
bEnable	使能功能块	BOOL	输入变量	功能块开启
bExeData	计数信号	BOOL	输入变量	计数信号变量
bClear	清除	BOOL	输入变量	清除当前计数值和输出类型数据
udCyclicData	计数目标值	UDINT	输入变量	计数目标值
bFBRun	功能块运行标志	BOOL	输出变量	功能块正在运行
udCurrentCountData	当前计数值	UDINT	输出变量	bExeData 从 FALSE-TRUE，计数+1

3、调用举例：



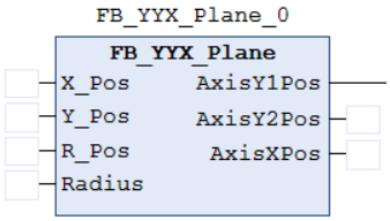
FB_YYX_Plane -XYR 坐标变换

指令名称 FB_YYX_Plane （XYR 坐标转换 YYX 坐标功能块）

功能块说明：本功能块适用于 YYX 轴机构和相机配合，做物料纠偏补偿等工况。将相机 XYR 坐标通过计算转换成 YYX 轴补偿值，实现相同的位置补偿。

1、指令格式

图形表现：



ST 表现：

```
YYX_Plane_0(
    X_Pos:= ,
    Y_Pos:= ,
    R_Pos:= ,
    Radius:= ,
    AxisY1Pos=> ,
    AxisY2Pos=> ,
    AxisXPos=> );
```

2、相关变量

输入变量

输入变量	名称	数据类型	有效范围	描述

X_Pos	补偿 X 坐标	REAL		填写需要补偿的 X 坐标
Y_Pos	补偿 Y 坐标	REAL		填写需要补偿的 Y 坐标
R_Pos	补偿 R 坐标	REAL		填写需要补偿的 R 坐标
Radius	平台转中心半径	REAL	>0	填写实际机构平台转心半径

输出变量

输出变量	名称	数据类型	有效范围	描述
AxisY1Pos	当前 Y1 轴补偿值	REAL		将补偿值相对当前 Y1 轴位置进行定位即可
AxisY2Pos	当前 Y2 轴补偿值	REAL		将补偿值相对当前 Y2 轴位置进行定位即可
AxisXPos	当前 X 轴补偿值	REAL		将补偿值相对当前 X 轴位置进行定位即可

3、使用说明

YYX_Plane：在当前周期可计算完成，在程序上端调用，在下方读取值，进行使用。

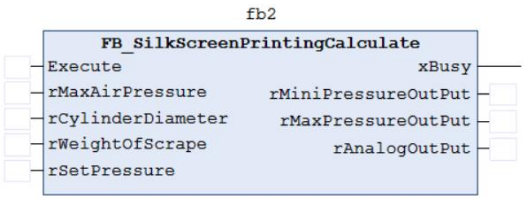
FB_SilkScreenPrintingCalculate 刮刀压力计算

指令名称：FB_SilkScreenPrintingCalculate

功能块说明：本功能块适用于丝网印刷气缸刮刀压力计算。

1、指令格式

图形表现



ST 表现

```
fb1(
    Execute:= ,
    rMaxAirPressure:= ,
    rCylinderDiameter:= ,
    rWeightOfScrape:= ,
    rSetPressure:= ,
    xBusy=> ,
    rMiniPressureOutPut=> ,
    rMaxPressureOutPut=> ,
    rAnalogOutPut=> );
```

2、相关变量

变量名称	名称	数据类型	变量属性	描述
Execute	触发功能	BOOL	输入变量	触发功能块运行，上升沿有效
rMaxAirPressure	进气最大气压	REAL	输入变量	填写刮刀连接调压阀的进气大小
rCylinderDiameter	气缸缸圈直径	REAL	输入变量	填写刮刀连接气缸的缸圈直径
rWeightOfScrape	刮刀模块重量	REAL	输入变量	填写刮刀模块的重量
rSetPressure	印刷设定压力	REAL	输入变量	填写需要设定的印刷压力

xBusy	功能块运行中	BOOL	输出变量	功能块运行中
rMiniPressureOutPut	最小输出压力	REAL	输出变量	输出值刮刀模块重量*9.8
rMaxPressureOutPut	最大输出压力	REAL	输出变量	输出值刮刀模块重量*9.8+进气最大气压*(3.1415*气缸缸圈直径*气缸缸圈直径/4)
rAnalogOutPut	模拟量输出值	REAL	输出变量	输出值给到模拟量模块的调节调压阀的压力

3、使用说明

最小输出压力:=刮刀模块重量*9.8;

最大输出压力:=刮刀模块重量*9.8+进气最大气压*(3.1415*气缸缸圈直径*气缸缸圈直径/4);

当 印刷设定压力<=最小输出压力 模拟量输出值:=0;

当 印刷设定压力>=最大输出压力 模拟量输出值:=20000;

当 最小输出压力<印刷设定压力<最大输出压力 模拟量输出值:=REAL_TO_INT((印刷设定压力-最小输出压力)/(3.1415*气缸缸圈直径*气缸缸圈直径/4)*20000/进气最大气压);