雷赛智能
Leadshine
稳定可靠的运动控制专家

2025|01

雷赛大中型 PLC 指令手册



www.leisai.com

前言

资料简介

感谢您选用深圳市雷赛智能控制股份有限公司大中型 PLC 产品。本手册提供了雷赛智能大中型 PLC 的编程指令说明。对于初次使用的用户，请认真阅读本手册。若对产品的功能应用和性能方面有所疑惑，请咨询我司技术支持人员以获得帮助。

由于产品的改进，手册内容可能持续更新。

技术热线：400-885-5501

版权说明

本手册版权归深圳市雷赛控制技术有限公司所有，未经本公司书面许可，任何人不得翻印、翻译和抄袭本手册中的任何内容。

本手册中的信息资料仅供参考。由于改进设计和功能等原因，雷赛公司保留对本资料的最终解释权，内容如有更改，恕不另行通知。

版本变更记录

修订日期	发布版本	变更内容
2023/07	V0.1	初版发行
2025/01	V0.2	细小勘误

目录

前言 2
目录 3
1 指令概述 17
1.1 概述 17
1.2 标准数据类型 17
2 指令一览表 18
3 基本指令 28
3.1 比较 28
3.1.1 指令列表 28
3.1.2 GT 28
3.1.3 LT 29
3.1.4 GE 31
3.1.5 LE 32
3.1.6 EQ 33
3.1.7 NE 35
3.2 选择 37
3.2.1 指令列表 37
3.2.2 SEL 37
3.2.3 MUX 38
3.2.4 MAX 40
3.2.5 MIN 41
3.2.6 LIMIT 43
3.3 计数 45
3.3.1 指令列表 45
3.3.2 CTD 45
3.3.3 CTU 47
3.3.4 CTUD 49
3.4 定时器 53
3.4.1 指令列表 53
3.4.2 TP 53

3.4.3 TON 55
3.4.4 TOF 56
3.4.5 RTC 59
3.5 位和逻辑字 61
3.5.1 指令列表 61
3.5.2 AND 61
3.5.3 OR 63
3.5.4 NOT 64
3.5.5 XOR 66
3.5.6 SR 67
3.5.7 RS 69
3.5.8 R_TRIG 70
3.5.9 F_TRIG 72
3.6 数据移位 75
3.6.1 指令列表 75
3.6.2 SHR 75
3.6.3 SHL 76
3.6.4 ROR 78
3.6.5 ROL 79
3.7 数据类型转换指令 82
3.7.1 指令列表 82
3.7.2 BOOL_TO_<TYPE> 82
3.7.3 BYTE_TO_<TYPE> 84
3.7.4 〈整型数据〉_TO_<TYPE> 85
3.7.5 〈实数型数据〉_TO_〈TYPE〉 86
3.8 数据处理指令 89
3.8.1 指令列表 89
3.8.2 MOVE 89
3.8.3 HEXinASCII_TO_BYTE 90
3.8.4 BYTE_TO_HEXinASCII 91
3.8.5 WORD_AS_STRING 93
3.8.6 BYTE_TO_HEXSTRING 94

3.8.7 WORD_TO_HEXSTRING 95
3.8.8 DWORD_TO_HEXSTRING 96
3.9 数学运算指令 98
3.9.1 指令列表 98
3.9.2 ADD 98
3.9.3 SUB 99
3.9.4 MUL 101
3.9.5 DIV 102
3.9.6 MOD 104
3.9.7 ABS 105
3.9.8 SQRT 106
3.9.9 LN 107
3.9.10 LOG 108
3.9.11 EXP 110
3.9.12 EXPT 111
3.9.13 SIN 112
3.9.14 COS 113
3.9.15 TAN 114
3.9.16 ASIN 115
3.9.17 ACOS 116
3.9.18 ATAN 118
3.9.19 SIZEOF 119
3.10 字符串 121
3.10.1 指令列表 121
3.10.2 LEN 121
3.10.3 LEFT 122
3.10.4 RIGHT 123
3.10.5 MID 125
3.10.6 CONCAT 126
3.10.7 INSERT 127
3.10.8 DELETE 129
3.10.9 FIND 130
3.10.10 REPLACE 131

3.11 地址操作 133
3.11.1 指令列表 133
3.11.2 ADR 133
3.11.3 ^ 134
3.11.4 BITADR 136
3.12 文件操作 138
3.12.1 指令列表 138
3.12.2 SysFileOpen 138
3.12.3 SysFileClose 140
3.12.4 SysFileWrite 141
3.12.5 SysFileRead 143
3.12.6 SysFileDelete 145
3.12.7 SysFileCopy 146
3.12.8 SysFileRename 147
3.12.9 SysFileSetPos 148
3.12.10 SysFileGetPos 150
3.12.11 SysFileGetSize 151
3.13 模拟量计算 153
3.13.1 指令列表 153
3.13.2 PD 153
3.13.3 PID 155
3.13.4 PID_FIXCYCLE 158
3.14 BCD 码转换 161
3.14.1 指令列表 161
3.14.2 BCD_TO_INT 161
3.14.3 INT_TO_BCD 162
3.14.4 BCD_TO_BYTE 163
3.14.5 BYTE_TO_BCD 165
3.14.6 BCD_TO_WORD 166
3.14.7 WORD_TO_BCD 167
3.14.8 BCD_TO_DWORD 168
3.14.9 DWORD_TO_BCD 170

3.15 模拟波形 172
3.15.1 指令列表 172
3.15.2 BLINK 172
3.15.3 GEN 173
3.15.4 FREQ_MEASURE 176
4 运动指令 179
4.1 单轴状态监控 179
4.1.1 指令列表 179
4.1.2 MC_Power 179
4.1.3 MC_Reset 183
4.1.4 MC_ReadStatus 185
4.1.5 MC_ReadAxisError 190
4.1.6 MC_ReadParameter 193
4.1.7 MC_ReadBoolParameter 195
4.1.8 MC_WriteParameter 198
4.1.9 MC_WriteBoolParameter 200
4.1.10 MC_ReadActualPosition 203
4.1.11 MC_ReadActualVelocity 205
4.1.12 MC_ReadActualTorque 207
4.1.13 MC_SetPosition 210
4.1.14 SMC_ReadSetPosition 212
4.1.15 SMC_ReadFBError 215
4.1.16 SMC_ClearFBError 218
4.2 单轴运动控制 220
4.2.1 指令列表 220
4.2.2 MC_Home 220
4.2.3 MC_MoveAbsolute 223
4.2.4 MC_MoveRelative 228
4.2.5 MC_MoveVelocity 233
4.2.6 MC_Stop 238
4.2.7 MC_Halt 242
4.2.8 MC_Jog 245
4.2.9 MC_MoveAdditive 248

4.2.10 MC_MoveSuperImposed 253
4.2.11 MC_PositionProfile 257
4.2.12 MC_VelocityProfile 263
4.2.13 MC_AccelerationProfile 266
4.2.14 SMC_Homing 270
4.2.15 SMC_Inch 278
4.3 同步运动功能 283
4.3.1 指令列表 283
4.3.2 MC_GearIn 283
4.3.3 MC_GearInPos 287
4.3.4 MC_GearOut 293
4.3.5 MC_CamTableSelect 296
4.3.6 MC_CamIn 299
4.3.7 MC_CamOut 304
4.3.8 SMC_GetTappetValue 306
4.4 插补指令 309
4.4.1 指令列表 309
4.4.2 LS_2AxisLine 309
4.4.3 LS_3AxisLine 312
4.4.4 LS_4AxisLine 316
4.4.5 LS_5AxisLine 319
4.4.6 LS_6AxisLine 323
4.4.7 LS_2AxisLine_Ratio 326
4.4.8 LS_LineFollow 329
4.4.9 LS_2AxisCircle 332
4.4.10 LS_3AxisCircle 338
4.4.11 LS_2AxisEllipses 342
4.4.12 LS_2AxisCircle_Helical 346
4.4.13 LS_3AxisMoveSequence 351
4.4.14 LS_4AxisMoveSequence 356
4.4.15 LS_6AxisMoveSequence 361
5 通信指令 367
5.1 自由 TCP 通信指令 367

5.1.1 指令列表 367
5.1.2 TCP_Client 367
5.1.3 TCP_Write 371
5.1.4 TCP_Read 374
5.1.5 TCP_Connection 378
5.1.6 TCP_Server 381
5.2 自由 UDP 通信指令 385
5.2.1 指令列表 385
5.2.2 UDP_Peer 385
5.2.3 UDP_Receive 388
5.2.4 UDP_Send 392
5.3 EtherCAT 通信指令 397
5.3.1 指令列表 397
5.3.2 ETC_CO_SdoReadDWord 397
5.3.3 ETC_CO_SdoRead4 401
5.3.4 ETC_CO_SdoRead 405
5.3.5 ETC_CO_SdoWriteDword 409
5.3.6 ETC_CO_SdoWrite4 413
5.3.7 ETC_CO_SdoWrite 417
5.3.8 IoDrvEtherCAT_Diag 421
5.3.9 ETCSlave 423
5.4 Ethernet-IP 通信指令 426
5.4.1 指令列表 426
5.4.2 IoDrvEtherNetIP_Diag 426
5.4.3 RemoteAdapter 428
5.4.4 Generic_Service 430
5.4.5 Get_Attributes_All 440
5.4.6 Get_Attribute_Single 449
5.4.7 Set_Attributes_All 459
5.4.8 Set_Attribute_Single 468
6 高速 IO 指令 478
6.1 脉冲轴控制指令 478

6.1.1 指令列表 478
6.1.2 LS_Home_P 478
6.1.3 LS_MotionControl_P 480
6.1.4 LS_ReadAxisPara_P 482
6.1.5 LS_ResetAxis_P 485
6.2 高速计数器指令 487
6.2.1 指令列表 487
6.2.2 LS_Counter 487
6.2.3 LS_PresetValue 488
6.3 LC1000 本地高速计数器 490
6.3.1 指令列表 490
6.3.2 LS_Counter 490
6.3.3 LS_PresetValue 492
6.3.4 LS_TouchProbe 495
6.3.5 LS_Compare 500
6.3.6 LS_CompareStep 503
6.3.7 LS_CompareFIFO 507
6.3.8 LS_PWM 512
6.3.9 LS_EnableInterrupt 514
7 PMC600 专用指令 516
7.1 PWM 输出指令 516
7.1.1 指令列表 516
7.1.2 LS_PWM_SetVal 516
7.1.3 LS_PWM_GetVal 519
7.2 系统时间指令 522
7.2.1 指令列表 522
7.2.2 LS_DateAndTime_ReadDint 522
7.2.3 LS_DateAndTime_ReadString 524
7.2.4 LS_DateAndTime_SetDint 527
7.2.5 LS_DateAndTime_SetString 530
7.3 固件版本指令 534
7.3.1 指令列表 534

7.3.2 LS_PLC_ControllerVer 534
7.3.3 LS_PLC_CPUID 536
7.3.4 LS_PLC_HardwareVerInformation 537
7.3.5 LS_PLC_SoftwareVerInformation 539
7.3.6 LS_PLC_VerInformation 541
7.4 UsrData 内文件操作指令 545
7.4.1 指令列表 545
7.4.2 LS_FileWrite 545
7.4.3 LS_FileWriteAppend 548
7.4.4 LS_FileRead 550
7.4.5 LS_FileCopy 554
7.4.6 LS_FileRename 556
7.4.7 LS_FileDelete 559
7.4.8 LS_FileDeleteAll 561
7.4.9 LS_GetDirectoryFile 563
7.4.10 LS_GetFileInformation 566
7.4.11 LS_UDisk_CopyFromUDisk 569
7.4.12 LS_UDisk_CopyToUDisk 571
7.4.13 LS_UDisk_GetDirectoyFile 574
7.5 UsrConfig 内文件操作指令 578
7.5.1 指令列表 578
7.5.2 LS_ConfigFile_Write 578
7.5.3 LS_ConfigFile_WriteAppend 580
7.5.4 LS_ConfigFile_Read 583
7.5.5 LS_ConfigFile_Del 586
7.5.6 LS_ConfigFile_GetDirectoryFile 588
7.5.7 LS_ConfigFile_CopyFromUDisk 591
7.5.8 LS_ConfigFile_CopyToUDisk 594
7.6 HMI 库变量批量操作 597
7.6.1 指令列表 597
7.6.2 LS_Modbus_SetM 597
7.6.3 LS_Modbus_ResetM 599
7.6.4 LS_Modbus_SetByteToM 601

7.6.5 LS_Modbus_SetDWS 603
7.6.6 LS_Modbus_SetDWU 605
7.6.7 LS_Modbus_SetDDS 608
7.6.8 LS_Modbus_SetDDU 610
7.6.9 LS_Modbus_SetDR 612
7.6.10 LS_Modbus_SetDS 614
7.7 RS232 无协议通信 617
7.7.1 指令列表 617
7.7.2 LS_RS232_Open 617
7.7.3 LS_RS232_Write 620
7.7.4 LS_RS232_Read 622
7.7.5 LS_RS232_Close 625
7.8 RS485 无协议通信 627
7.8.1 指令列表 627
7.8.2 LS_RS485_Open 627
7.8.3 LS_RS485_Write 630
7.8.4 LS_RS485_Read 632
7.8.5 LS_RS485_Close 635
7.9 以太网通信 637
7.9.1 指令列表 637
7.9.2 LS_TCP_SetIpAddress 637
7.9.3 LS_TCP_GetIpAddress 640
7.9.4 LS_TCP_SetServer 643
7.9.5 LS_TCP_SetClient 646
7.9.6 LS_TCP_WriteData 649
7.9.7 LS_TCP_ReadData 652
7.9.8 LS_UDP_SetConnect 654
7.9.9 LS_UDP_WriteData 657
7.9.10 LS_UDP_ReadData 660
7.10 雷赛插补指令 663
7.10.1 指令列表 663
7.10.2 LS_2AxisLineAbs 664
7.10.3 LS_2AxisLineRel 669

7.10.4 LS_3AxisLineAbs 673
7.10.5 LS_3AxisLineRel 678
7.10.6 LS_4AxisLineAbs 683
7.10.7 LS_4AxisLineRel 688
7.10.8 LS_5AxisLineAbs 693
7.10.9 LS_5AxisLineRel 699
7.10.10 LS_6AxisLineAbs 705
7.10.11 LS_6AxisLineRel 711
7.10.12 LS_2AxisLineAbs_Ratio 717
7.10.13 LS_2AxisLineRel_Ratio 722
7.10.14 LS_LineFollowAbs 727
7.10.15 LS_LineFollowRel 732
7.10.16 LS_2AxisCircleAbs 737
7.10.17 LS_2AxisCircleRel 745
7.10.18 LS_3AxisCircleAbs 752
7.10.19 LS_3AxisCircleRel 758
7.10.20 LS_2AxisEllipsesAbs 763
7.10.21 LS_2AxisEllipsesRel 769
7.10.22 LS_2AxisCircleAbs_Helical 775
7.10.23 LS_2AxisCircleRel_Helical 782
7.10.24 LS_3AxisGCode 789
7.10.25 LS_3AxisGCode_File 794
7.10.26 LS_4AxisGCode 800
7.10.27 LS_4AxisGCode_File 806
7.10.28 LS_4AxisGCodeAxisP 812
7.10.29 LS_4AxisGCodeAxisP_File 818
7.10.30 LS_6AxisGCodeAxisUVW_File 825
7.10.31 LS_6Axis_ZeroOffset 836
7.10.32 LS_3AxisMoveSequence(PMC600) 839
7.10.33 LS_4AxisMoveSequence(PMC600) 845
7.10.34 LT_6AxisMoveSequence(PMC600) 852
7.11 雷赛单轴运动控制指令 860
7.11.1 指令列表 860

7.11.2 LS_EtherCATAxis_HomeMove 860
7.11.3 LS_MoveAbsChangePosVel 863
7.11.4 LS_MoveChangeVel 866
7.11.5 LS_MoveAbs 869
7.11.6 LS_MoveRel 872
7.11.7 LS_MoveVel 875
7.11.8 LS_ResetVirtualAxis 878
7.12 脉冲轴控制 880
7.12.1 指令列表 880
7.12.2 LS_PulseAxis_MachineState 881
7.12.3 LS_PulseAxis_SetPulseMode 883
7.12.4 LS_PulseAxis_GetPulseMode 886
7.12.5 LS_PulseAxis_SetSpecialIOLogicalState 888
7.12.6 LS_PulseAxis_GetSpecialIOLogicalState 892
7.12.7 LS_PulseAxis_GetSpecialIOState 895
7.12.8 LS_PulseAxis_SetRatio 897
7.12.9 LS_PulseAxis_ReadRatio 899
7.13 编码器 902
7.13.1 指令列表 902
7.13.2 LS_Encoder_EzClearSet 902
7.13.3 LS_Encoder_EzClearRead 904
7.13.4 LS_Encoder_EzSetFilterTime 906
7.13.5 LS_Encoder_EzGetFilterTime 908
7.13.6 LS_Encoder_SetVal 910
7.13.7 LS_Encoder_GetVal 912
7.13.8 LS_Encoder_SetWorkMode 915
7.13.9 LS_Encoder_SetAxisMove 918
7.14 高速比较 922
7.14.1 指令列表 922
7.14.2 LS_HighSpeedCmp_ClearPara 922
7.14.3 LS_HighSpeedCmp_SetWorkPara 925
7.14.4 LS_HighSpeedCmp_GetWorkPara 929
7.14.5 LS_HighSpeedCmp_SetCmpPos 932

7.14.6 LS_HighSpeedCmp_SetLinearPara 935
7.14.7 LS_HighSpeedCmp_GetLinearPara 938
7.14.8 LS_HighSpeedCmp_SetFIFOPos 941
7.14.9 LS_HighSpeedCmp_SetFIFOPosType2 943
7.14.10 LS_HighSpeedCmp_GetCurState 946
7.15 锁存功能 949
7.15.1 指令列表 949
7.15.2 LS_ZeroLatch_SetPara 949
7.15.3 LS_ZeroLatch_GetState 951
7.15.4 PMC_Controller.EzLatch_SetPara 955
7.15.5 PMC_Controller.EzLatch_GetState 957
7.15.6 LS_HighSpeedLatch_Clear 958
7.15.7 LS_HighSpeedLatch_SetSource 960
7.15.8 LS_HighSpeedLatch_SetWorkMode 963
7.15.9 LS_HighSpeedLatch_ReadState 966
7.15.10 LS_HighSpeedLatch_ReadVal 968
7.15.11 LS_HighSpeedLatch_FIFOReadVal 971
7.15.12 LS_HighSpeedLatch_FIFOReadNumber 974
8 特殊指令 977
8.1 SD 卡文件操作指令 977
8.1.1 指令列表 977
8.1.2 LS_CopyFromSDCard 977
8.1.3 LS_CopyToSDCard 979
8.1.4 LS_GetSDCardInfomation 981
8.2 系统指令 984
8.2.1 指令列表 984
8.2.2 LS_GetIpAddress 984
8.2.3 LS_SetIpAddress 986
8.2.4 GetPLCLoad 988
8.2.5 GetPLCVersion 990
8.2.6 ColdResetApp 992
8.2.7 WarmResetApp 993
8.2.8 LS_ReadDintDT 994

8.2.9 LS_ReadStringDT 996
8.2.10 LS_SetDintDT 998
8.2.11 LS_SetStringDT 1000

1 指令概述

1.1概述

本手册面向雷赛大中型 PLC，覆盖 LC 系列，MC 系列所有产品。本手册主要介绍雷赛大中型 PLC 所需要的指令、程序等内容，提供相关指令使用示范，指导客户快速使用。

1.2标准数据类型

数据大类	数据类型	关键字	占用内存位数	取值范围
布尔类型	布尔	BOOL	8	FALSE(0)或 TRUE(1)
整型	字节	BYTE	8	0～255
	字	WORD	16	0～65535
	双字	DWORD	32	0～4294967295
	长字	LWORD	64	0～(2^64-1)
	有符号短整型	SINT	8	-128～127
	无符号短整型	USINT	8	0～255
	整型	INT	16	-32768～32767
	无符号整型	UINT	16	0～65535
	双整型	DINT	32	-2147483648～2147483647
	无符号双整型	UDINT	32	0～4294967295
	长整型	LINT	64	-2^63～(2^63-1)
	无符号长整型	ULINT	64	0～(2^64-1)
浮点	单精度	REAL	32	1.175494351e-38～ 3.402823466e+38
	双精度	LREAL	64	2.2250738585072014e-308～ 1.7976931348623158e+308
字符串	字符串	STRING	8*N	

2 指令一览表

指令类别	名称	FB/FC	功能
比较	GT	FC	大于
	LT	FC	小于
	GE	FC	大于等于
	LE	FC	小于等于
	EQ	FC	等于
	NE	FC	不等于
选择指令	SEL	FC	二选一指令
	MUX	FC	多选一指令
	MAX	FC	取最大值指令
	MIN	FC	取最小值指令
	LIMIT	FC	极限值指令
计数器指令	CTD	FB	减法计数器组
	CTU	FB	加法计数器组
	CTUD	FB	可逆计数器组
定时器指令	TP	FB	普通定时器
	TON	FB	通电延时定时器
	TOF	FB	断电延时定时器
	RTC	FB	实时时钟
位和逻辑字	AND	FC	与
	OR	FC	或
	NOT	FC	非
	XOR	FC	异或
	SR	FB	置位优先
	RS	FB	复位优先
	R_TRIG	FC	上升沿检测
	F_TRIG	FC	下降沿检测
数据移位	SHR	FC	按位右移
	SHL	FC	按位左移
	ROR	FC	循环右移
	ROL	FC	循环左移
数据类型转换	BOOL_TO_<TYPE>	FC	BOOL 类型转换指令
	BYTE_TO_<TYPE>	FC	字节类型转换指令
	WORD_TO_<TYPE>	FC	字类型转换指令

指令类别	名称	FB/FC	功能
	DWORD_TO_<TYPE>	FC	双字类型转换指令
	INT_TO_<TYPE>	FC	整数型类型转换指令
	SINT_TO_<TYPE>	FC	短整型转换指令
	DINT_TO_<TYPE>	FC	长整形转换指令
	UDINT_TO_<TYPE>	FC	无符号长整型转换指令
	REAL_TO_<TYPE>	FC	实数类型转换指令
	STRING_TO_<TYPE>	FC	字符类型转换指令
	TIME_TO_<TYPE>	FC	时钟类型转换指令
	TOD_TO_<TYPE>	FC	时间类型转换指令
	DATE_TO_<TYPE>	FC	日期类型转换指令
	DT_TO_<TYPE>	FC	日期时间类型转换指令
数据处理指令	MOVE	FC	赋值
	HEXinASCII_TO_BYTE	FC	ASCII 转数字指令
	BYTE_TO_HEXinASCII	FC	数字转 ASCII 指令
	WORD_AS_STRING	FC	数字转字符串指令
	BYTE_TO_HEXSTRING	FC	BYTE 类型转 16 进制字符串
	WORD_TO_HEXSTRING	FC	WORD 类型转 16 进制字符串
	DWORD_TO_HEXSTRING	FC	DWORD 类型转 16 进制字符串
数学运算指令	ADD	FC	加法
	SUB	FC	减法
	MUL	FC	乘法
	DIV	FC	除法
	MOD	FC	取余
	ABS	FC	绝对值
	SQRT	FC	平方根
	LN	FC	自然对数
	LOG	FC	常用对数
	EXP	FC	指数
	EXPT	FC	幂指数
	SIN	FC	正弦
	COS	FC	余弦
	TAN	FC	正切
	ASIN	FC	反正弦
	ACOS	FC	反余弦

指令类别	名称	FB/FC	功能
	ATAN	FC	反正切
	SIZEOF	FC	获取数据大小
字符串指令	LEN	FC	获取字符串长度
	LEFT	FC	从左边取字符串
	RIGHT	FC	从右边取字符串
	MID	FC	从中间取字符串
	CONCAT	FC	字符串连接
	INSERT	FC	字符串插入
	DELETE	FC	字符串删除
	FIND	FC	字符串查找
	REPLACE	FC	字符串替换
地址操作	ADR	FC	取地址指令
	^	FC	取地址内容指令
	BITADR	FC	位地址
文件操作	SysFileOpen	FB	打开文件
	SysFileClose	FB	关闭文件
	SysFileWrite	FB	写入文件
	SysFileRead	FB	读取文件
	SysFileDelete	FB	删除文件
	SysFileCopy	FB	复制文件
	SysFileRename	FB	重命名文件
	SysFileSetPos	FB	设置文件读写位置
	SysFileGetPos	FB	读取文件读写位置
	SysFileGetSize	FB	获取文件大小
模拟量计算	PD	FB	比例微分控制
	PID	FB	比例积分微分控制指令
	PID_FIXCYCLE	FB	可手动设置循环周期的比例积分微分控制
BCD 码转换指令	BCD_TO_INT	FC	BCD码转整型指令
	INT_TO_BCD	FC	整型转BCD码指令
	BCD_TO_BYTE	FC	BCD码转字节指令
	BYTE_TO_BCD	FC	字节转BCD码指令
	BCD_TO_WORD	FC	BCD码转字指令
	WORD_TO_BCD	FC	字转BCD指令
	BCD_TO_DWORD	FC	BCD码转双字指令

指令类别	名称	FB/FC	功能
	DWORD_TO_BCD	FC	双字转BCD码指令
模拟波形	BLINK	FB	脉冲信号发生器
	GEN	FB	周期性信号发生器
	FREQ_MEASURE	FB	频率测量指令
单轴状态监控	MC_Power	FB	轴使能
	MC_Reset	FB	复位轴
	MC_ReadStatus	FB	读轴状态
	MC_ReadAxisError	FB	读轴错误
	MC_ReadParameter	FB	读参数
	MC_ReadBoolParameter	FB	读布尔参数
	MC_WriteParameter	FB	写参数
	MC_WriteBoolParameter	FB	写布尔参数
	MC_ReadActualPosition	FB	读实际位置
	MC_ReadActualVelocity	FB	读实际速度
	MC_ReadActualTorque	FB	读实际转矩
	MC_SetPosition	FB	设置位置
	SMC_ReadSetPosition	FB	读取设置位置
	SMC_ReadFBError	FB	读取历史错误信息
	SMC_ClearFBError	FB	清除历史错误信息
	SMC_ErrorString	FB	错误码到错误信息
单轴运动控制	MC_Home	FB	回零
	MC_MoveAbsolute	FB	绝对运动
	MC_MoveRelative	FB	相对运动
	MC_MoveVelocity	FB	定速运动
	MC_Stop	FB	停止
	MC_Halt	FB	暂停
	MC_Jog	FB	点动运动
	MC_MoveAdditive	FB	附加运动
	MC_MoveSuperImposed	FB	叠加运动
	MC_PositionProfile	FB	位置规划
	MC_VelocityProfile	FB	速度规划
	MC_AccelerationProfile	FB	加速度规划
	SMC_Homing	FB	上位机回原
	SMC_Inch	FB	寸动
单轴运同步运	MC_GearIn	FB	电子齿轮耦合

指令类别	名称	FB/FC	功能
动功能动控制	MC_GearInPos	FB	电子齿轮平滑耦合
	MC_GearOut	FB	电子齿轮脱离
	MC_CamTableSelect	FB	凸轮挺杆关联
	MC_CamIn	FB	电子凸轮关联
	MC_CamOut	FB	电子凸轮脱离
	SMC_GetTappetValue	FB	读取挺杆状态
插补指令	LS_2AxisLine	FB	两轴直线插补
	LS_3AxisLine	FB	三轴直线插补
	LS_4AxisLine	FB	四轴直线插补
	LS_5AxisLine	FB	五轴直线插补
	LS_6AxisLine	FB	六轴直线插补
	LS_2AxisLineA_Ratio	FB	两轴可调速直线插补
	LS_LineFollow	FB	跟随运动
	LS_2AxisCircle	FB	两轴圆弧插补
	LS_3AxisCircle	FB	三轴圆弧插补
	LS_2AxisEllipses	FB	两轴椭圆插补
	LS_2AxisCircle_Helical	FB	螺旋插补
	LS_3AxisMoveSequence	FB	三轴连续插补运动
	LS_4AxisMoveSequence	FB	四轴连续插补运动
	LS_6AxisMoveSequence	FB	六轴连续插补运动
自由 TCP 通信指令	TCP_Client	FB	创建 TCP 客户端通信服务
	TCP_Write	FB	TCP 通信数据发送
	TCP_Read	FB	TCP 通信数据接收
	TCP_Connection	FB	创建 TCP 连接，并连接到服务器
	TCP_Server	FB	创建 TCP 服务器端通信服务
自由 UDP 通信指令	UDP_Peer	FB	创建 UDP 通信连接
	UDP_Receive	FB	UDP 通信数据接收
	UDP_Send	FB	UDP 通信数据发送
EtherCAT 通信指令	ETC_CO_SdoReadDWord	FB	EtherCAT 从站 SDO 读取
	ETC_CO_SdoRead4	FB	EtherCAT 从站 SDO 读取
	ETC_CO_SdoRead	FB	EtherCAT 从站 SDO 读取
	ETC_CO_SdoWrite_Dword	FB	EtherCAT 从站 SDO 写入
	ETC_CO_SdoWrite4	FB	EtherCAT 从站 SDO 写入
	ETC_CO_SdoWrite	FB	EtherCAT 从站 SDO 写入

指令类别	名称	FB/FC	功能
	IoDrvEtherCAT_Diag	FB	EtherCAT 主站实例
	ETCSlave	FB	EtherCAT 从站实例
Ethernet-IP 通信指令	IoDrvEtherNetIP_Diag	FB	Ethernet/IP 主站实例
	RemoteAdapter	FB	Ethernet/IP 从站实例
	Generic_Service	FB	CIP 通用显示服务
	Get_Attributes_All	FB	获取对象实例全部属性
	Get_Attribute_Single	FB	获取对象实例单个属性
	Set_Attributes_All	FB	设置对象实例所有属性
	Set_Attribute_Single	FB	设置对象实例单个属性
脉冲轴控制	LS_Home_P	FB	脉冲轴回原
	LS_MotionControl_P	FB	脉冲轴绑定
	LS_ReadAxisPara_P	FB	获取脉冲轴的脉冲当量值
	LS_ResetAxis_P	FB	脉冲轴复位
高速计数器	LS_Counter	FB	高速计数器指令
	LS_PresetValue	FB	预置计数值指令
PWM输出指令	LS_PWM_SetVal	FC	PWM输出指令
	LS_PWM_GetVal	FC	PWM参数读取指令
系统时间指令	LS_DateAndTime_ReadDint	FB	读取系统时间指令
	LS_DateAndTime_ReadString	FB	读取系统时间指令
	LS_DateAndTime_SetDint	FB	设置系统时间指令
	LS_DateAndTime_SetString	FB	设置系统时间指令
固件版本指令	LS_PLC_ControllerVer	FC	读取硬件版本指令
	LS_PLC_CPUID	FC	读取CPU ID号指令
	LS_PLC_HardwareVerInformation	FC	读取固件FPGA版本指令
	LS_PLC_SoftwareVerInformation	FC	读取固件ARM版本指令
	LS_PLC_VerInformation	FC	读取控制器的所有版本信息指令
UsrData内文件操作指令	LS_FileWrite	FB	文件写入指令
	LS_FileWriteAppend	FB	文件追加写入指令
	LS_FileRead	FB	文件读取指令
	LS_FileCopy	FB	文件拷贝指令
	LS_FileRename	FB	文件重命名指令
	LS_FileDelete	FB	文件删除指令
	LS_FileDeleteAll	FB	文件批量删除指令
	LS_GetDirectoryFile	FB	获取目录内文件信息指令

指令类别	名称	FB/FC	功能
	LS_GetFileInformation	FB	获取指定文件信息指令
	LS_UDisk_CopyFromUDisk	FB	从U盘拷贝文件指令
	LS_UDisk_CopyToUDisk	FB	拷贝文件至U盘指令
	LS_UDisk_GetDirectoyFile	FB	获取U盘根目录内文件信息指令
UsrConfig内文件操作指令	LS_ConfigFile_Write	FB	文件写入指令
	LS_ConfigFile_WriteAppend	FB	文件追加写入指令
	LS_ConfigFile_Read	FB	文件读取指令
	LS_ConfigFile_Del	FB	文件删除指令
	LS_ConfigFile_GetDirectoryFile	FB	获取目录内所有文件信息指令
	LS_ConfigFile_CopyFromUDisk	FB	从U盘拷贝文件指令
	LS_ConfigFile_CopyToUDisk	FB	拷贝文件至U盘指令
HMI库变量批量操作	LS_Modbus_SetM	FB	BOOL变量批量置位指令
	LS_Modbus_ResetM	FB	BOOL变量批量复位指令
	LS_Modbus_SetByteToM	FB	BYTE变量批量置位指令
	LS_Modbus_SetDWS	FB	INT变量批量置值指令
	LS_Modbus_SetDWU	FB	UINT变量批量置值指令
	LS_Modbus_SetDDS	FB	DINT变量批量置值指令
	LS_Modbus_SetDDU	FB	UDINT变量批量置值指令
	LS_Modbus_SetDR	FB	REAL变量批量置值指令
	LS_Modbus_SetDS	FB	STRING变量批量置值指令
RS232 无协议通信	LS_RS232_Open	FB	打开指定的 232 端口指令
	LS_RS232_Write	FB	从 232 端口进行无协议数据发送指令
	LS_RS232_Read	FB	从 232 端口读取无协议的接收数据指令
	LS_RS232_Close	FB	关闭指定的 232 端口指令
RS485 无协议通信	LS_RS485_Open	FB	打开指定的RS485 端口指令
	LS_RS485_Write	FB	从RS485 端口进行无协议数据发送指令
	LS_RS485_Read	FB	从RS485 端口进行无协议的接收数据指令
	LS_RS485_Close	FB	关闭指定的RS485 端口指令
以太网通信	LS_TCP_SetIpAddress	FB	设置网口IP指令
	LS_TCP_GetIpAddress	FB	读取网口IP指令
	LS_TCP_SetServer	FB	将控制器设为TCP服务器指令
	LS_TCP_SetClient	FB	将控制器设为TCP客户端指令

指令类别	名称	FB/FC	功能
	LS_TCP_WriteData	FB	TCP发送数据指令
	LS_TCP_ReadData	FB	TCP接收数据指令
	LS_UDP_SetConnect	FB	启动UDP/IP服务指令
	LS_UDP_WriteData	FB	UDP发送数据指令
	LS_UDP_ReadData	FB	UDP接收数据指令
雷赛插补指令	LS_2AxisLineAbs	FB	两轴绝对直线插补
	LS_2AxisLineRel	FB	两轴相对直线插补
	LS_3AxisLineAbs	FB	三轴绝对直线插补
	LS_3AxisLineRel	FB	三轴相对直线插补
	LS_4AxisLineAbs	FB	四轴绝对直线插补
	LS_4AxisLineRel	FB	四轴相对直线插补
	LS_5AxisLineAbs	FB	五轴绝对直线插补
	LS_5AxisLineRel	FB	五轴相对直线插补
	LS_6AxisLineAbs	FB	六轴绝对直线插补
	LS_6AxisLineRel	FB	六轴相对直线插补
	LS_2AxisLineAbs_Ratio	FB	两轴可调速绝对直线插补
	LS_2AxisLineRel_Ratio	FB	两轴可调速相对直线插补
	LS_LineFollowAbs	FB	绝对跟随
	LS_LineFollowRel	FB	相对跟随
	LS_2AxisCircleAbs	FB	两轴绝对圆弧插补
	LS_2AxisCircleRel	FB	两轴相对圆弧插补
	LS_3AxisCircleAbs	FB	三轴绝对圆弧插补
	LS_3AxisCircleRel	FB	三轴相对圆弧插补
	LS_2AxisEllipsesAbs	FB	两轴绝对椭圆插补
	LS_2AxisEllipsesRel	FB	两轴相对椭圆插补
	LS_2AxisCircleAbs_Helical	FB	绝对螺旋插补
	LS_2AxisCircleRel_Helical	FB	相对螺旋插补
	LS_3AxisGCode	FB	三轴G代码插补
	LS_3AxisGCode_File	FB	三轴G代码插补-文件
	LS_4AxisGCode	FB	四轴G代码插补
	LS_4AxisGCode_File	FB	四轴G代码插补-文件
	LS_4AxisGCodeAxisP	FB	四轴G代码插补-P
	LS_4AxisGCodeAxisP_File	FB	四轴G代码插补-文件-P
	LS_6AxisGCodeAxisUVW_file	FB	六轴G代码插补-文件
	LS_6Axis_ZeroOffset	FB	6轴零点偏移

指令类别	名称	FB/FC	功能
	LS_3AxisMoveSequence	FB	三轴连续插补运动
	LS_4AxisMoveSequence	FB	四轴连续插补运动
	LS_6AxisMoveSequence	FB	六轴连续插补运动
雷赛单轴运动控制指令	LS_EtherCATAxis_HomeMove	FB	回零（驱动器回零）
	LS_MoveAbsChangePosVel	FB	在线变速变位置
	LS_MoveChangeVel	FB	在线变速
	LS_MoveAbs	FB	单轴起跳速度绝对运动
	LS_MoveRel	FB	单轴起跳速度相对运动
	LS_MoveVel	FB	单轴起跳速度恒速运动
	LS_ResetVirtualAxis	FB	单轴复位功能
脉冲控制指令	LS_PulseAxis_MachineState	FC	轴脉冲输出控制
	LS_PulseAxis_SetPulseMode	FC	设置轴脉冲模式
	LS_PulseAxis_GetPulseMode	FC	读取轴脉冲模式
	LS_PulseAxis_SetSpecialIOLogicalState	FC	设置轴特殊IO信号
	LS_PulseAxis_GetSpecialIOLogicalState	FC	读取轴特殊IO信号
	LS_PulseAxis_GetSpecialIOState	FC	获取轴特殊IO状态
	LS_PulseAxis_SetRatio	FC	设置轴电子齿轮比
	LS_PulseAxis_ReadRatio	FC	读取轴电子齿轮比
编码器指令	LS_Encoder_EzClearSet	FC	EZ清零参数设置
	LS_Encoder_EzClearRead	FC	EZ清零配置读取
	LS_Encoder_EzSetFilterTime	FC	EZ滤波参数设置
	LS_Encoder_EzGetFilterTime	FC	EZ滤波参数读
	LS_Encoder_SetVal	FC	设置编码器数值
	LS_Encoder_GetVal	FC	读取编码器数值
	LS_Encoder_SetWorkMode	FC	设置编码器工作模式
	LS_Encoder_SetAxisMove	FC	编码器数值转换轴运动
SD 卡文件操作	LS_CopyFromSDCard	FB	从SD卡拷贝文件至本地
	LS_CopyToSDCard	FB	将本地文件拷贝至SD卡
	LS_GetSDCardInformation	FB	获取SD卡下的文件信息
系统指令	LS_GetIpAddress	FB	获取PLC IP地址
	LS_SetIpAddress	FB	设置PLC IP地址
	GetPLCLoad	FC	获取PLC负载率
	GetPLCVersion	FC	获取PLC固件版本
	ColdResetApp	FC	冷复位PLC

指令类别	名称	FB/FC	功能
	WarmResetApp	FC	热复位PLC
	LS_ReadDintDT	FB	获取系统时间（Dint类型）
	LS_ReadStringDT	FB	获取系统时间（String类型）
	LS_SetDintDT	FB	设置系统时间（Dint类型）
	LS_SetStringDT	FB	设置系统时间（String类型）

3 基本指令

3.1 比较

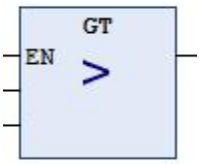
3.1.1 指令列表

指令类别	名称	FB/FC	功能
比较	GT	FC	大于
	LT	FC	小于
	GE	FC	大于等于
	LE	FC	小于等于
	EQ	FC	等于
	NE	FC	不等于

3.1.2 GT

判断两个输入值的大小，GT指令等同于数学符号中的大于号，即input1和input2相比较，当input1>input2时，输出output为TRUE，反之则为FAlSE。

- **指令格式**

指令	名称	FB/FC	LD表现	ST表现	库
GT	大于 指令	FC		>	-

- **相关变量**

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
input1	输入 1	-	遵从数据类型	0	数据 1
input2	输入 2	-	遵从数据类型	0	数据 2

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
output	返回值	BOOL	TRUE-FALSE	FALSE	比较结果

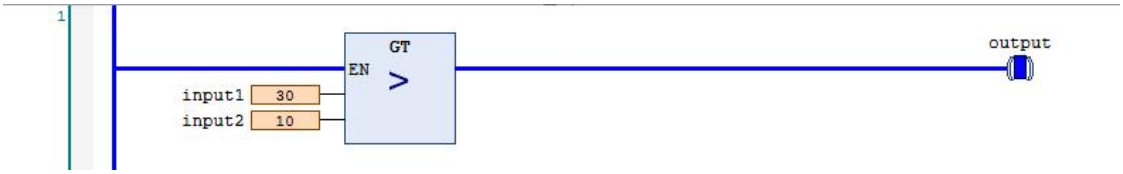
	布尔	位串				整数								实数		时刻、持续时间、日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In1	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√
In2	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√
Out	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

- **程序示例**

ST：

```
1  output TRUE :=input1 50 >input2 10 ;RETURN
```

LD：



- **注意事项**

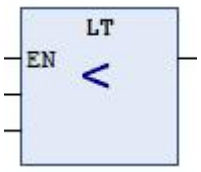
➢ 当两个输入变量数据类型不一致时编译会报错。

3.1.3 LT

判断两个输入值的大小，当第一个输入值小于第二个输入值时输出 TRUE，否则输出 FALSE。

- **指令格式**

指令	名称	FB/FC	LD 表现	ST 表现	库

LT	小于指令	FC		Out：=In1<In2；	—

- **相关变量**

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
In1	输入 1	—	遵从数据类型	0	数据 1
In2	输入 2	—	遵从数据类型	0	数据 2

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
Out	返回值	BOOL	FALSE,TRUE	FALSE	比较结果

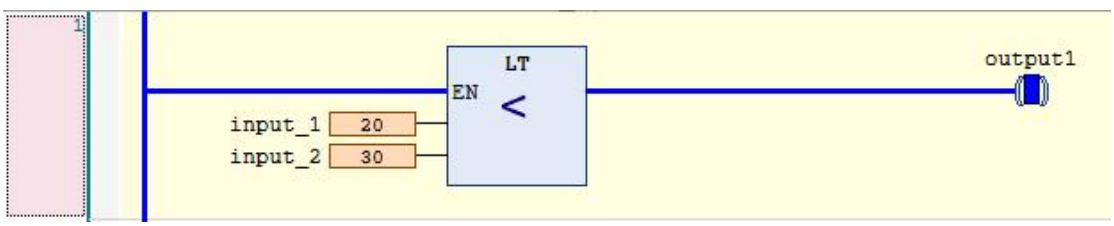
	布尔	位串				整数								实数		时刻、持续时间 日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In1	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√
In2	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√
Out	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

- **程序示例**

ST：当 Input_1 小于 Input_2 时输出 TRUE，反之输出 FALSE。

```
1  POU();
2  output1 TRUE :=input_1 20 <input_2 30 ;RETURN
```

LD：当 Input_1 小于 Input_2 时输出TRUE，反之输出FALSE。



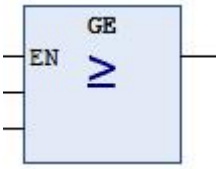
- **注意事项**

➢ 当两个输入变量数据类型不一致时编译会报错。

3.1.4 GE

判断两个输入值的大小，当第一个输入值大于等于第二个输入值时输出 TRUE，否则输出 FALSE。

- **指令格式**

指令	名称	FB/FC	LD 表现	ST 表现	库
GE	大于等于指令	FC		OUT:=IN1 >= IN2	-

- **相关变量**

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
In1	输入 1	—	遵从数据类型	0	数据 1
In2	输入 2	—	遵从数据类型	0	数据 2

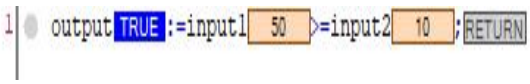
输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
Out	返回值	BOOL	TRUE-FALSE	FALSE	比较结果

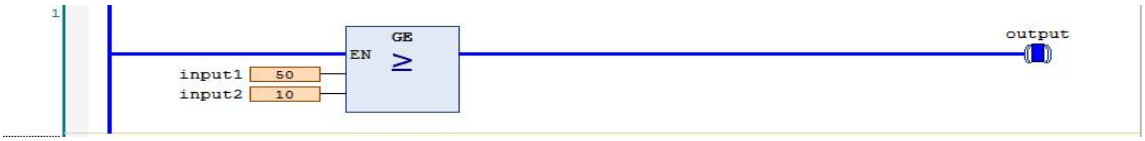
	布尔	位串				整数								实数		时刻、持续时间 日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In1	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√
In2	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√
Out	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

- **程序示例**

ST：当 Input_1 大于等于 Input_2 时输出 TRUE，反之输出 FALSE。



LD：当 Input_1 大于等于 Input_2 时输出 TRUE，反之输出 FALSE。



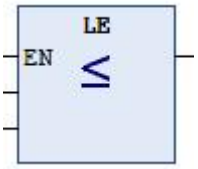
- **注意事项**

➢ 当两个输入变量数据类型不一致时编译会报错。

3.1.5 LE

判断两个输入值的大小，当第一个输入值小于等于第二个输入值时输出 TRUE，否则输出 FALSE。

- **指令格式**

指令	名称	FB/FC	LD 表现	ST 表现	库
LE	小于等于指令	FC		Out：In1<=In2	—

- **相关变量**

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
In1	输入 1	—	遵从数据类型	0	数据 1
In2	输入 2	—	遵从数据类型	0	数据 2

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
Out	返回值	BOOL	TRUE-FALSE	FALSE	比较结果

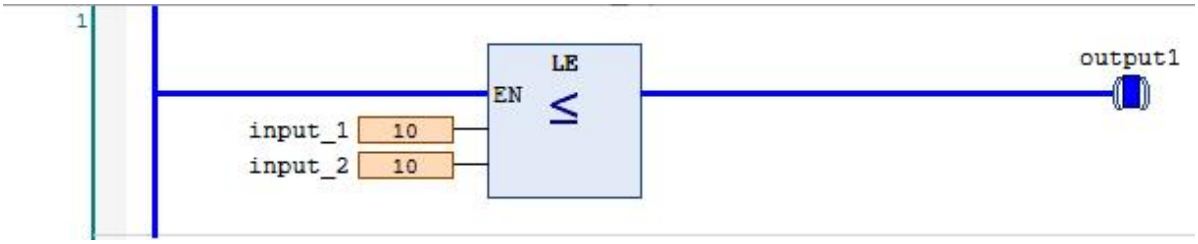
	布尔	位串				整数								实数		时刻、持续时间 日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In1	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√
In2	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√
Out	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

- **程序示例**

ST：当 Input_1 小于等于 Input_2 时输出 TRUE，反之输出 FALSE。

```
1  POU();
2  output1 TRUE :=input_1 10 <=input_2 10 ;RETURN
```

LD：当 Input_1 小于等于 Input_2 时输出 TRUE，反之输出 FALSE。



- **注意事项**

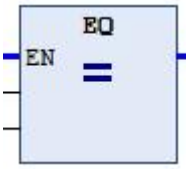
➢ 当两个输入变量数据类型不一致时编译会报错。

3.1.6 EQ

判断两个输入值的大小，当第一个输入值等于第二个输入值时输出 TRUE，否则输出 FALSE。

- **指令格式**

指令	名称	FB/FC	LD 表现	ST 表现	库

EQ	等于	FC		Out：In1=In2	—

- **相关变量**

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
In1	输入 1	—	遵从数据类型	0	数据 1
In2	输入 2	—	遵从数据类型	0	数据 2

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
Out	返回值	BOOL	TRUE-FALSE	FALSE	比较结果

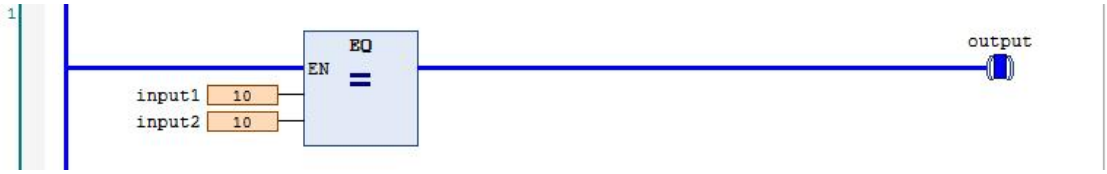
	布尔	位串				整数								实数		时刻、持续时间 日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In1	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√
In2	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√
Out	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

- **程序示例**

ST：当 Input1 等于 Input2 时输出 TRUE，反之输出 FALSE。

```
1  output TRUE :=input1 10 =input2 10 ;RETURN
```

LD：当 Input1 等于 Input2 时输出 TRUE，反之输出 FALSE。



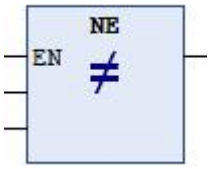
- **注意事项**

➢ 当两个输入变量数据类型不一致时编译会报错。

3.1.7 NE

判断两个输入值的大小，当第一个输入值不等于第二个输入值时输出 TRUE，否则输出 FALSE。

- **指令格式**

指令	名称	FB/FC	LD 表现	ST 表现	库
NE	不等于	FC		Out：In1<>In2	—

- **相关变量**

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
In1	输入 1	—	遵从数据类型	0	数据 1
In2	输入 2	—	遵从数据类型	0	数据 2

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
Out	返回值	BOOL	TRUE-FALSE	FALSE	比较结果

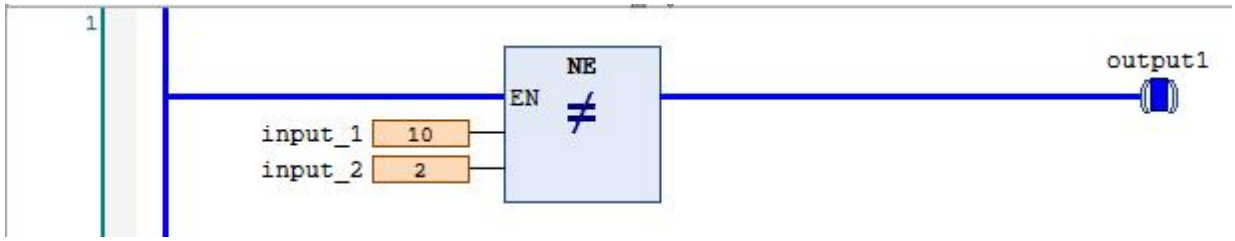
	布尔	位串				整数								实数		时刻、持续时间 日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In1	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√
In2	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√
Out	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

- **程序示例**

ST：当 Input_1 不等于 Input_2 时输出TRUE，反之输出FALSE。

```
1  POU();
2  output1 TRUE :=input_1  10  <>input_2  2  ;RETURN
```

LD：当 Input_1不等于Input_2 时输出TRUE，反之输出FALSE。



- **注意事项**

➢ 当两个输入变量数据类型不一致时编译会报错。

3.2选择

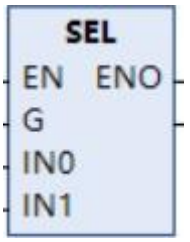
3.2.1 指令列表

指令类别	名称	FB/FC	功能
选择指令	SEL	FC	二选一指令
	MUX	FC	多选一指令
	MAX	FC	取最大值指令
	MIN	FC	取最小值指令
	LIMIT	FC	极限值指令

3.2.2 SEL

从两个操作数中选择一个。由G决定IN0还是IN1为输出。

- **指令格式**

指令	名称	FB/FC	LD表现	ST表现	库
SEL	二选一指令	FC		SEL	-

- **相关变量**

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
G	输出控制	BOOL	TRUE-FALSE	-	G=FLASE 时，输出 IN0 ；当 G=True 时，输出 IN1
IN0	输入0	任何类型	-	-	数据 1
IN1	输入1	任何类型	-	-	数据 2

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
output	选择结果	-	-	0	选择后输出的数字结果

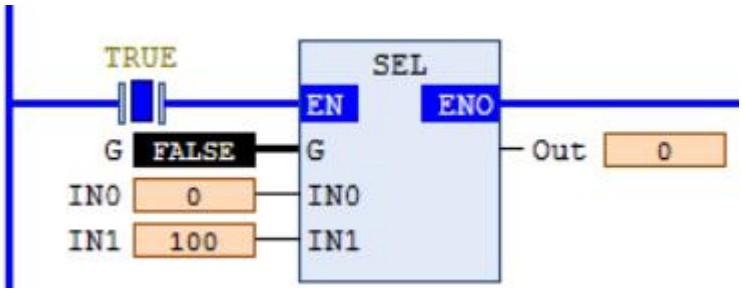
	布尔	位串				整数								实数		时刻、持续时间、日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
G	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
In1	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√
In2	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√
OUT	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√

- **程序示例**

ST：设置一组数据IN0=0、IN0=100。当G=True的时候，输出的结果为100

```
iOut 100 := SEL( G TRUE , IN0 0 , IN1 100 );
```

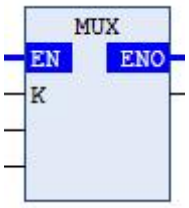
LD：



3.2.3 MUX

从多个输入值中取一个值输出，从一组值中选择第K个值。第一个值是 K=0。如果K比其他输入的值大，那么output传递最后的一个值。

- **指令格式**

指令	名称	FB/FC	LD表现	ST表现	库
MUX	多选一指令	FC		MUX	-

- **相关变量**

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
K	选择位	任何类型	-	-	选择的数字位
IN1	比较数1	任何类型	-	-	数据 1
IN 2	比较数2	任何类型	-	-	数据 2

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
Output	选择结果	-	-	0	选择后输出的数字结果

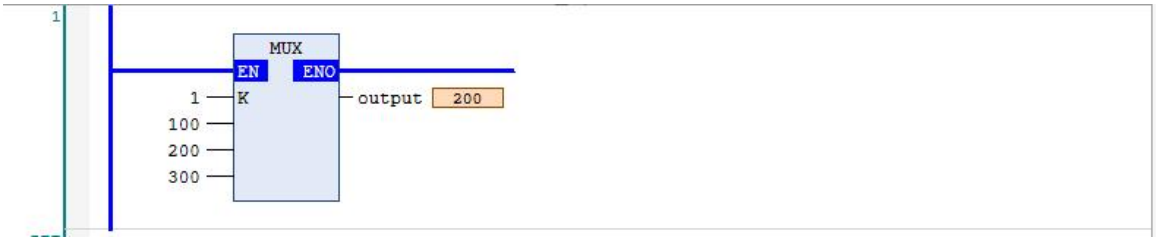
	布尔	位串				整数								实数		时刻、持续时间、日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
K	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√
In1	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√
In2	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√
Output	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√

- **程序示例**

ST：设置一组数据100、200、300。当K=1的时候，输出的结果为200

```
1  output 200 :=MUX(1,100,200,300);
2  RETURN
```

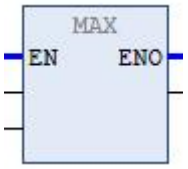
LD：



3.2.4 MAX

取输入值中的最大值，MAX指令实在两个输入之间比较之后，输出取最大值，即input1和input2相比较，output取两者之间的最大值。

- **指令格式**

指令	名称	FB/FC	LD表现	ST表现	库
MAX	取最大值指令	FC		MAX	-

- **相关变量**

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
IN1	比较数1	-	-	0	比较数据 1
IN2	比较数2	-	-	0	比较数据 2

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
output	最大数	-	-	0	取两个输入数字中最大

					的数

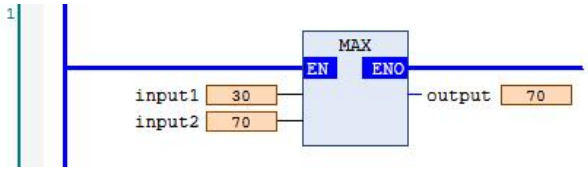
	布尔	位串				整数								实数		时刻、持续时间、日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In1	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√
In2	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√
Output	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√

- **程序示例**

ST：

```
1  output 70 :=(max (input1 30 ,input2 70 ));RETURN
```

LD：

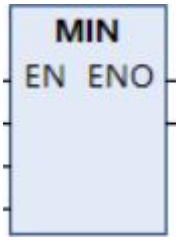


3.2.5 MIN

取输入值中的最小值，MIN指令实在两个输入之间比较之后，输出取最大值，即IN1和IN2相比较，output取两者之间的最大值。

- **指令格式**

指令	名称	FB/FC	LD表现	ST表现	库

MIN	取最小值指令	FC		MIN	-

- **相关变量**

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
IN1	比较数1	-	-	0	比较数据 1
IN2	比较数2	-	-	0	比较数据 2

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
output	最小数	-	-	0	取两个输入数字中最小的数

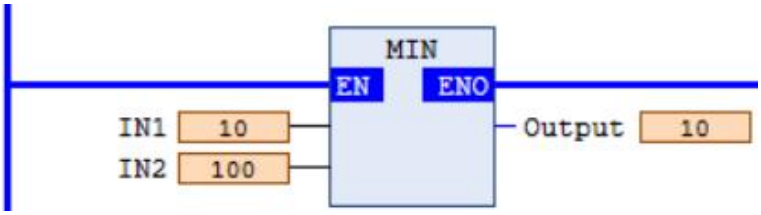
	布尔	位串				整数								实数		时刻、持续时间、日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In1	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√
In2	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√
Output	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√

- **程序示例**

ST：

```
Output 10 := MIN(IN1 10 ,IN2 100 );RETURN
```

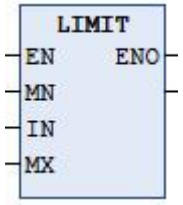
LD：



3.2.6 LIMIT

取极限值指令，MX是结果的上限值并且MN是结果的下限值。如果IN得值超过了MX的上限值，则输出运算结果为MX。如果IN的值小于MN的下限值，则输出运算结果为MN。当输入IN的值在MN和MX范围之内的时候，则输出的值为输入的IN的值。

- **指令格式**

指令	名称	FB/FC	LD表现	ST表现	库
LIMIT	极限值指令	FC		LIMIT	-

- **相关变量**

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
MN	输入信号	所有类型	-	-	数字下限
IN	输入信号	所有类型	-	-	比较中间值
MX	输入信号	所有类型	-	-	数字上限

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
output	输出结果	-	-	0	选择后输出的数字结果

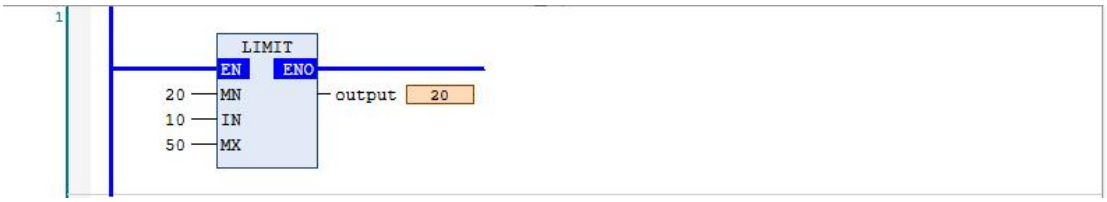
	布尔	位串				整数								实数		时刻、持续时间、日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
MN	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√
IN	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√
MX	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√
Output	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√

- **程序示例**

ST：

```
1  output 20 :=LIMIT(20,10,50);RETURN
```

LD：



3.3计数

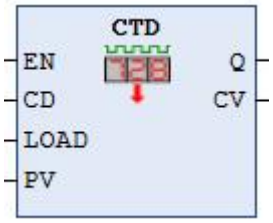
3.3.1 指令列表

指令类别	名称	FB/FC	功能
计数器指令	CTD	FB	减法计数器组
	CTU	FB	加法计数器组
	CTUD	FB	可逆计数器组

3.3.2 CTD

该功能块用作一个递减计数器。通过预设值代入变量，每次检测到上升沿便减 1，到 0 时输出状态 true。

- **指令格式**

指令	名称	FB/FC	LD表现	ST表现	库
CTD	递减计数器指令	FB		CTD(CD:=, LOAD:=, PV:=, Q=>, CV=>);	Standard

- **相关变量**

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
CD	计数器输入	BOOL	TRUE-FALSE	FALSE	计数器输入
LOAD	计数器复位	BOOL	TRUE-FALSE	FALSE	用PV给参数CV赋值
PV	预设值	WORD	0~65535	0	计数器预设值

输出变量

输入变量	名称	数据类型	有效范围	初始值	描述
Q	计数器输出	BOOL	TRUE-FALSE	FALSE	计数器输出：TRUE 表示 ON，FALSE 表示 OFF
CV	计数实时值	WORD	0~65535	0	计数实时值

	布尔	位串				整数								实数		时刻、持续时间 日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
CD	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
LOAD	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
PV	-	-	√	-	-	-	√	-	-	-	√	-	-	-	-	-	-	-	-	-
Q	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
CV	-	-	√	-	-	-	√	-	-	-	√	-	-	-	-	-	-	-	-	-

- **功能说明**

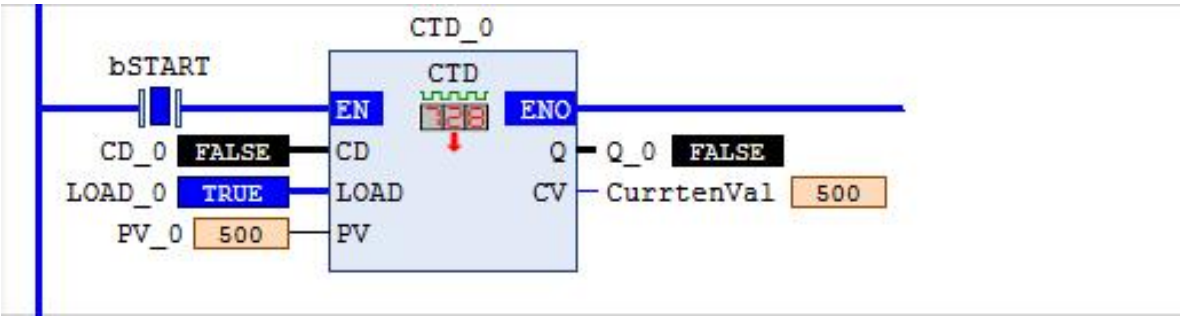
  - 减法计数器。预设值、计数值的数据类型WORD。
  - 当LOAD为TRUE时，预设值PV的值将载入计数值CV中，计数器输出Q变为FALSE。LOAD为TRUE期间，将忽略CD，CV不减少。当LOAD为FALSE时，当CD端有一个从FALSE变为TRUE的上升沿时，若CV大于0时，它将减1，当CV等于0时，Q返回TRUE。计数器输入信号“CD”启动时，“CV”不断减少。“CV”的值小于 0 时，“Q”的值变为TRUE。“CV”的值小于 0 时，即使“CD”处于上升沿，“CV”不进行减法运算。

- **程序示例**

ST:

```
CTD_0(CD FALSE :=CD_0 FALSE , LOAD TRUE :=LOAD_0 TRUE , PV 500 :=pv_0 500 , Q FALSE =>Q_0 FALSE , CV 500 =>CV_0 500 );RETURN
```

LD:

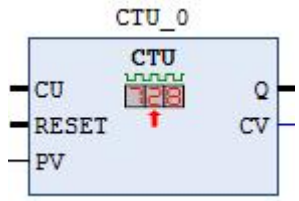


3.3.3 CTU

每次输入信号，每次增加一次的加法计数器，递增的计数器。预设值、计数值的数据类型 WORD。

若复位信号“RESET”设置为 TRUE，计数值“CV”的值将变为 0，计数器输出“Q”变为 FALSE。计数器输入信号“CU”每启动一次时，“CV”增加一次。“CV”的值大于预设值“PV”的值时，“Q” 的值变为TRUE。“CV”的值大于“PV”的值时，即使输入更大的“CU”的值，“CV”也不会变化，“RESET”为 TRUE 期间，忽略“CU”。“CV”不进行加法运算。

- **指令格式**

指令	名称	FB/FC	LD表现	ST表现	库
CTU	加法计数器	FB		CTU_0(CU:=start, RESET:= , PV:=, Q=> , CV=>);	Standard

- **相关变量**

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
CU	计数器输入	BOOL	TRUE-FALSE	FALSE	计数器上输入
RESET	计数器复位	BOOL	TRUE-FALSE	FALSE	TRUE:将CV复位为0

PV	预设值	WORD	0~65535	0	计数器预设值

输出变量

输入变量	名称	数据类型	有效范围	初始值	描述
Q	计数器输出	BOOL	TRUE-FALSE	FALSE	计数器输出：TRUE 表示 ON，FALSE 表示 OFF
CV	计数实时值	WORD	0~65535	0	计数实时值

	布尔	位串				整数								实数		时刻、持续时间 日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
CU	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
RESET	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
PV	-	-	√	-	-	-	√	-	-	-	√	-	-	-	-	-	-	-	-	-
Q	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
CV	-	-	√	-	-	-	√	-	-	-	√	-	-	-	-	-	-	-	-	-

- **功能说明**

  - 每次输入信号，每次增加一次的加法计数器，递增的计数器。预设值、计数值的数据类型 WORD。

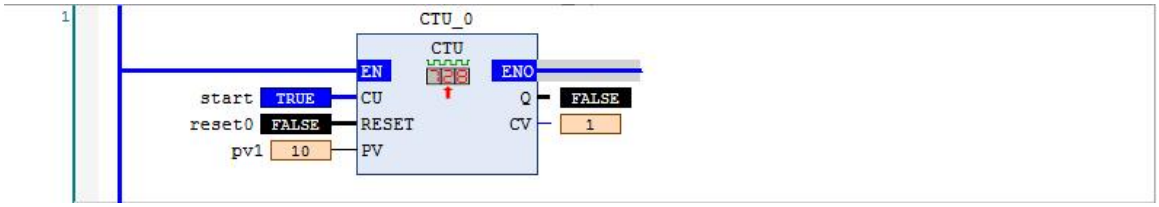
  - 若复位信号“RESET”设置为 TRUE，计数值“CV”的值将变为 0，计数器输出“Q”变为 FALSE。计数器输入信号“CU”每启动一次时，“CV”增加一次。“CV”的值大于预设值“PV”的值时，“Q” 的值变为TRUE。“CV”的值大于“PV”的值时，即使输入更大的“CU”的值，“CV”也不会变化，“RESET”为 TRUE 期间，忽略“CU”。“CV”不进行加法运算。

- **程序示例**

ST：

```
CTU_0(
    CU TRUE :=start TRUE,
    RESET FALSE :=reset0 FALSE,
    PV 10 :=pv1 10,
    Q=> ,
    CV=> );RETURN
```

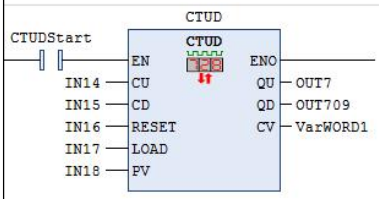
LD：



3.3.4 CTUD

根据加法计数器输入及减法计数器输入进行加减运算的计数器。预设值、计数值的数据类型为WORD。

- **指令格式**

指令	名称	FB/FC	LD表现	ST表现	库
CTUD	可逆计数器	FB		CTUD(CU:= , CD:= , RESET:= , LOAD:= , PV:= , QU=> , QD=>, CV=>);	Standard

- **相关变量**

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
CU	加法计数器输入	BOOL	TRUE-FALSE	FALSE	加法计数器输入

CD	减法器输入	BOOL	TRUE-FALSE	FALSE	减法计数器输入
RESET	计数器复位	BOOL	TRUE-FALSE	FALSE	当其为TRUE时，CV复位为0
LOAD	加载信号	BOOL	TRUE-FALSE	FALSE	当其为TRUE时，CV被置为PV
PV	预设值	WORD	0~65535	0	计数器的预设值

输出变量

输入变量	名称	数据类型	有效范围	初始值	描述
QU	加法计数器输出	BOOL	TRUE-FALSE	FALSE	计数器输出：TRUE表示加法计数器输出为ON，FALSE表示加法计数器输出为OFF
QD	减法计数器输出	BOOL	TRUE-FALSE	FALSE	计数器输出：TRUE表示减法计数器输出为ON，FALSE表示减法法计数器输出为OFF
CV	计数值	WORD	0-65535	0	计数器的当前值

	布尔	位串				整数								实数		时刻、持续时间 日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
CU	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
CD	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
RESET	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
LOAD	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
PV	-	-	√	-	-	-	√	-	-	-	√	-	-	-	-	-	-	-	-	-
QU	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
QD	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

CV	-	-	√	-	-	-	√	-	-	-	√	-	-	-	-	-	-	-	-

- **功能说明**

通过检测输入信号的上升沿来进行加减法计数器运算，实现加法与减法计数器的功能。

➢ 加法计数器功能：

复位信号“RESET”置为 TRUE 时，计数器实时值“CV”变为 0，加法计数器输出“QU”变为 FALSE。当加法计数器输入信号“CU”处于上升沿时，“CV”值加一，“CV”值大于等于计数上限的预设值“PV”时，“QU”值变为 TRUE。

➢ 减法计数器功能：

加载信号“LOAD”置为 TRUE 时，将计数上限的预设值“PV”值赋值到计数器实时值“CV”中，减法计数器输出“QD”变为 FALSE。当减法计数器输入信号“CD”处于上升沿时，“CV”值减一，“CV”值等于 0 时，“QD”的值变为 TRUE。“CV”的值等于 0 时，即使输入再大的“CD”值，“CV”也不会变化。

➢ 加法计数器、减法计数器共通的功能：

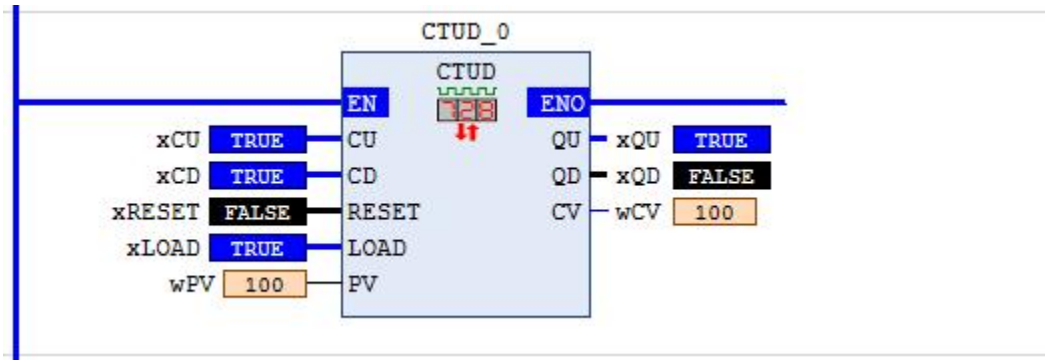
“LOAD”或“RESET”为 TRUE 期间，忽略“CU”和“CD”的变化。“CV”不进行加减法运算。“CU”和“CD”同时处于上升沿时，“CV”不会变化。“RESET”和“LOAD”均为 TRUE 时，“Reset”参数影响优先，“CV”的值变为 0。“RESET”设置为 TRUE 时，“CV”的值变为 0，因此“QD”值变为 TRUE。“LOAD”设置为 TRUE 时，“CV”的值与“PV”相等，因此“QU”值变为 TRUE。

- **程序示例**

ST：

```
CTUD_0(
CU TRUE :=xCU TRUE,
CD TRUE :=xCD TRUE,
RESET FALSE :=xRESET FALSE,
LOAD TRUE :=xload TRUE,
PV 100 :=wPV 100,
QU TRUE =>xQU TRUE,
QD FALSE =>xQD FALSE,
CV 100 =>wCV 100);
```

LD：



3.4定时器

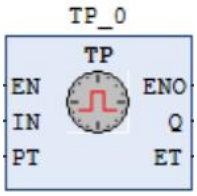
3.4.1 指令列表

指令类别	名称	FB/FC	功能
定时器指令	TP	FB	普通定时器
	TON	FB	通电延时定时器
	TOF	FB	断电延时定时器
	RTC	FB	实时时钟

3.4.2 TP

输入计时时间，输出当前计时时间与计时状态信号。

- **指令格式**

指令	名称	FB/FC	LD表现	ST表现	库
TP	定时器指令	FB		TP(IN:= , PT:= , Q=> , ET=>);	Standard

- **相关变量**

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
IN	输入信号	BOOL	TRUE-FALSE	FALSE	FALSE-TRUE开始计时
PT	计时时间	TIME	0-4294967295	0	设定时间

输出变量

输入变量	名称	数据类型	有效范围	初始值	描述
Q	计时完成	BOOL	TRUE-FALSE	FALSE	TRUE->FALSE计时完成

ET	当前计时时间	TIME	0-4294967295	0	设定时间

	布尔	位串				整数								实数		时刻、持续时间 日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
IN	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
PT	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-
Q	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
ET	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-

- **功能说明**

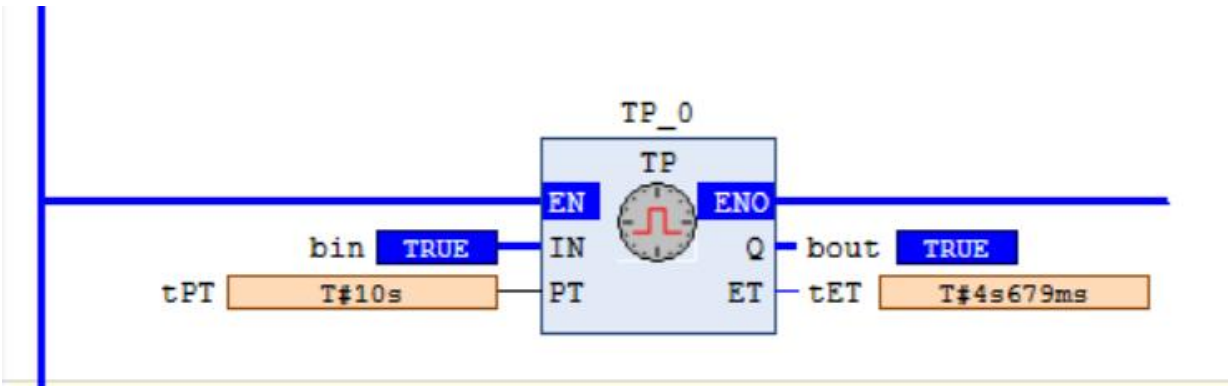
  - 当IN为TRUE时，Q输出PT时间的高电平信号。ET显示当前时间，当ET=PT时，Q变为FALSE。

- **程序示例**

ST：

```
TP_2(
IN TRUE :=bIn TRUE,
PT T#5s999ms :=tTime T#5s999ms,
Q TRUE =>xout TRUE,
ET T#2s300ms =>tTime_1 T#2s300ms );RETURN
```

LD：



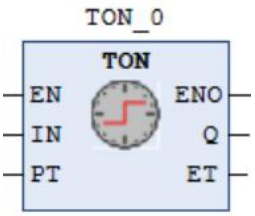
- **注意事项**

➢ 在计时中，可以改变PT值，当改变后的PT值大于ET值，则继续计时，当改变后的PT值小于等于ET值，则停止计时，Q变为FALSE。

3.4.3 TON

通电延时一段时间之后输出计时完成信号。

- **指令格式**

指令	名称	FB/FC	LD表现	ST表现	库
TON	通电延时定时器	FB		TON(IN:= , PT:= , Q=> , ET=>);	Standard

- **相关变量**

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
IN	输入信号	BOOL	TRUE-FALSE	FALSE	FALSE-TRUE开始计时
PT	计时时间	TIME	0-4294967295	0	设定时间

输出变量

输入变量	名称	数据类型	有效范围	初始值	描述
Q	计时完成	BOOL	TRUE-FALSE	FALSE	FALSE->TRUE 计时完成
ET	当前计时时间	TIME	0-4294967295	0	设定时间

	布	位串	整数	实数	时刻、持续时间

	尔															日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
IN	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
PT	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-
Q	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
ET	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-

- **功能说明**

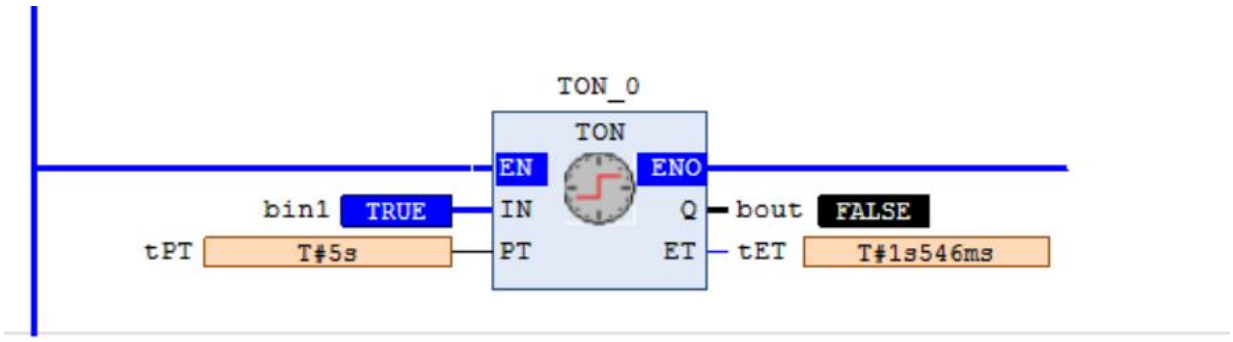
  - 当IN为TRUE时，ET显示IN导通的时间，当ET=PT时，Q变TRUE。

- **程序示例**

ST：

```
TON_1(
    IN TRUE :=bIn1 TRUE,
    PT T#5s999ms :=tTime T#5s999ms,
    Q FALSE =>xOut FALSE,
    ET T#2s656ms =>tTime_1 T#2s656ms );RETURN
```

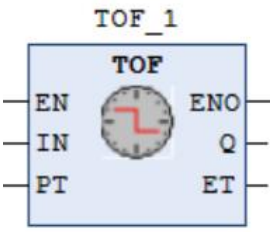
LD：



3.4.4 TOF

断电延时一段时间之后输出计时完成信号。

- **指令格式**

指令	名称	FB/FC	LD表现	ST表现	库
TOF	断电延时定时器	FB		TOF(IN:= , PT:= , Q=> , ET=>);	Standard

- **相关变量**

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
IN	输入信号	BOOL	TRUE-FALSE	FALSE	TRUE-FALSE开始计时
PT	计时时间	TIME	0-4294967295	0	设定时间

输出变量

输入变量	名称	数据类型	有效范围	初始值	描述
Q	计时完成	BOOL	TRUE-FALSE	FALSE	FALSE->TRUE 计时完成
ET	当前计时时间	TIME	0-4294967295	0	设定时间

	布尔	位串				整数								实数		时刻、持续时间 日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
IN	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
PT	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-
Q	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
ET	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-

- **功能说明**

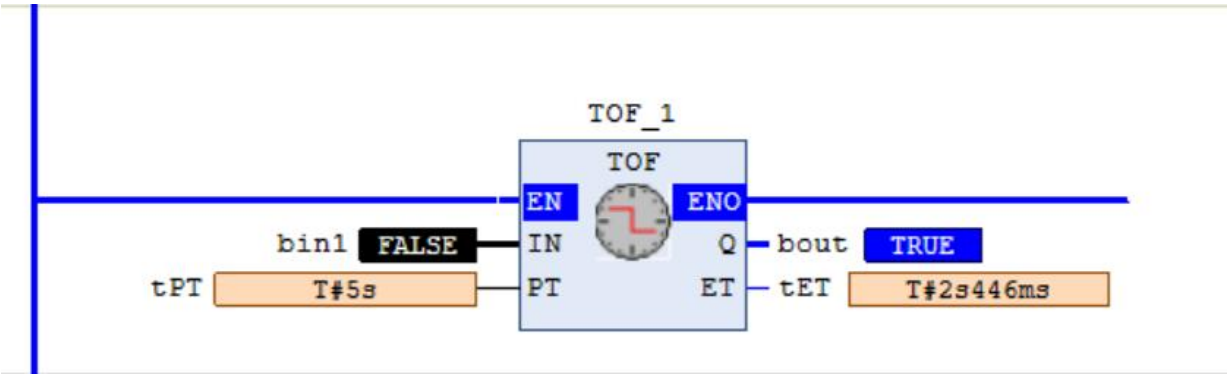
  - 输入信号PT设定计时时间，输入信号IN从TRUE变为FALSE时开始计时，计时完成后Q置为FALSE。

- **程序示例**

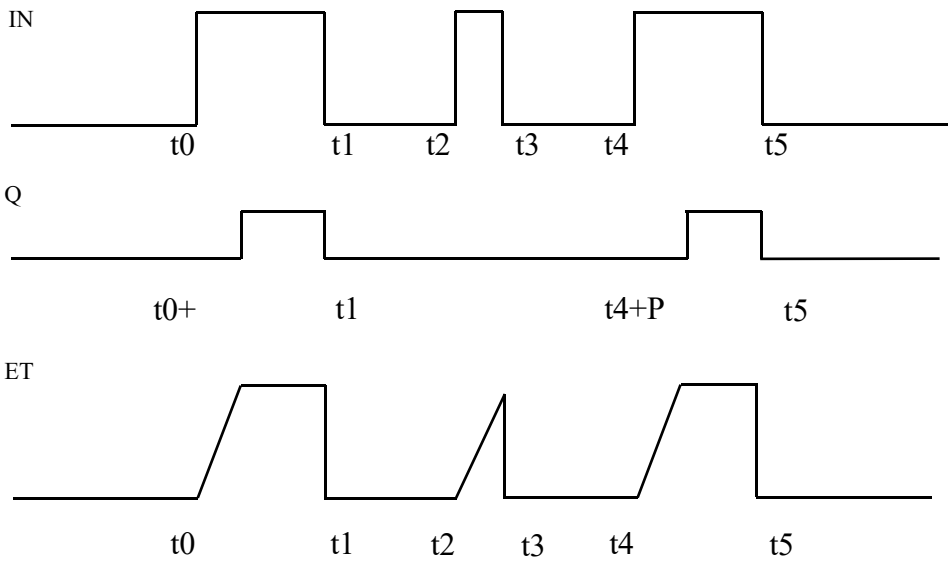
ST：

```
TOF_1(
    IN FALSE :=bin1 FALSE,
    PT T#5s999ms :=tTime T#5s999ms,
    Q TRUE =>xOut TRUE,
    ET T#3s761ms =>tTime_1 T#3s761ms
);RETURN
```

LD：



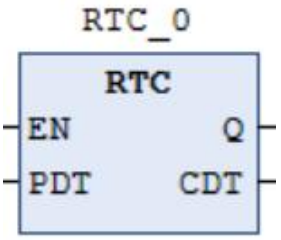
- **时序图**



3.4.5 RTC

时钟功能

- **指令格式**

指令	名称	FB/FC	LD表现	ST表现	库
RTC	实时时钟	FB		RTC(EN:= , PDT:= , Q=> , CDT=>);	Standard

- **相关变量**

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
PDT	计时起始时间	DATE_AND_TIME	0(1970-01-01,00:00:00) – 4294967295 (210602-07,06:28:15	FALSE	开始计时

输出变量

输入变量	名称	数据类型	有效范围	初始值	描述
Q	计时完成	BOOL	TRUE-FALSE	FALSE	FALSE->TRUE 计时有效
CDT	当前计时时间	DATE_AND_TIME	0(1970-01-01,00:00:00) – 4294967295 (210602-07,06:28:15	1970-01-01,00:00:00	开始计时

	布尔	位串				整数								实数		时刻、持续时间 日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
PDT	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-

Q	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
CDT	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-

- **功能说明**

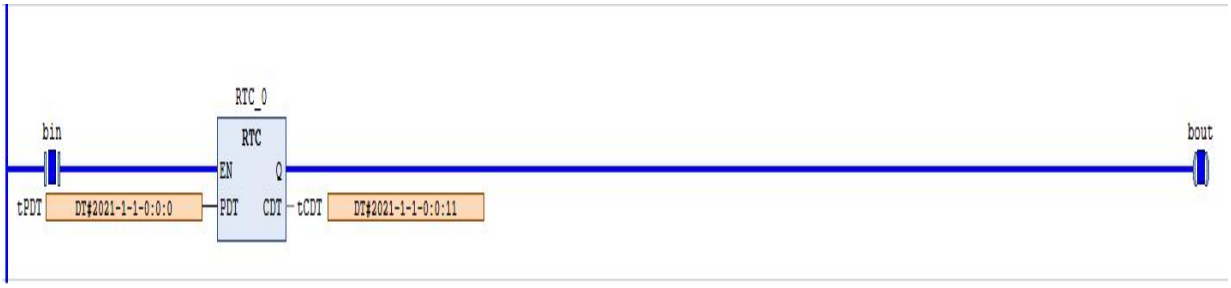
  - 输入EN为TRUE时，以设定时间PDT开始计时，输出Q为TRUE，输出CDT显示当前计时时间。

- **程序示例**

ST：

```
RTC_1(
    EN TRUE :=bIn TRUE,
    PDT DT#2021-1-1-0:0:0 :=tPDT DT#2021-1-1-0:0:0,
    Q TRUE =>xOut TRUE,
    CDT DT#2021-1-1-0:0:16 =>tCDT DT#2021-1-1-0:0:16 ); RETURN
```

LD：



3.5位和逻辑字

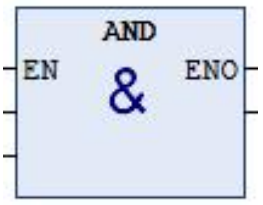
3.5.1 指令列表

指令类别	名称	FB/FC	功能
位和逻辑字指令	AND	FC	与
	OR	FC	或
	NOT	FC	非
	XOR	FC	异或
	SR	FB	置位优先
	RS	FB	复位优先
	R_TRIG	FC	上升沿检测
	F_TRIG	FC	下降沿检测

3.5.2 AND

与运算，可以分为：按位逻辑运算以及布尔逻辑运算。

- **指令格式**

指令	名称	FB/FC	LD表现	ST表现	库
AND	与指令	FC		AND	-

- **相关变量**

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
IN1	输入1	-	-	0	数据1
IN2	输入2	-	-	0	数据2

输出变量

输入变量	名称	数据类型	有效范围	初始值	描述
OUT	输出结果	-	-	0	输出结果

	布尔	位串				整数								实数		时刻、持续时间 日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
IN1	√	√	√	√	√	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-
IN2	√	√	√	√	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
OUT	√	√	√	√	√	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-

- **功能说明**

  - 按位“与”：按位“与”运算指令是比较两个整数的相应位。当两个数的相应位都是“1”时，返回相应的结果位是“1”；当两个整数的相应位都是“0”或者其中一个位是“0”时，则返回相应的结果位是“0”。

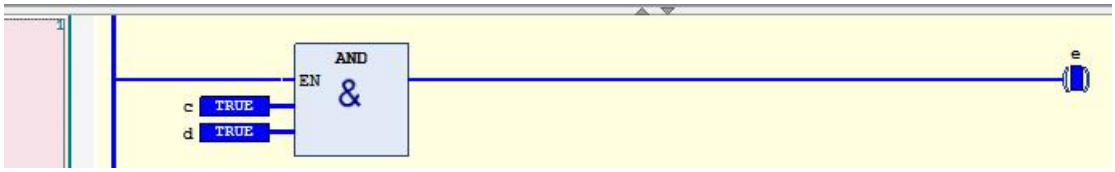
  - 布尔“与”：布尔“与”运算用于计算连个布尔表达式的“与”结果。当两个布尔表达式的结果都为真时，则返回为真，其中只要有一个为假，则返回为假。

- **程序示例**

ST：

```
1  e TRUE :=c TRUE AND d TRUE;
2
```

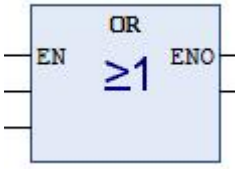
LD：



3.5.3 OR

或运算，可以分为：按位逻辑运算以及布尔逻辑运算。

- **指令格式**

指令	名称	FB/FC	LD表现	ST表现	库
OR	或指令	FC		OR	-

- **相关变量**

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
IN1	输入1	-	-	0	数据1
IN2	输入2	-	-	0	数据2

输出变量

输入变量	名称	数据类型	有效范围	初始值	描述
OUT	输出结果	-	-	0	输出结果

	布尔	位串				整数								实数		时刻、持续时间 日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
IN1	√	√	√	√	√	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-
IN2	√	√	√	√	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
OUT	√	√	√	√	√	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-

- **功能说明**

  - 按位“或”：按位“或”运算指令是比较两个整数的相应位。当两个整数的相应位有一个是“1”时，返回相应的结果位为“1”。当两个整数的相应位都是“0”时，则返回相应的结果位是“0”。

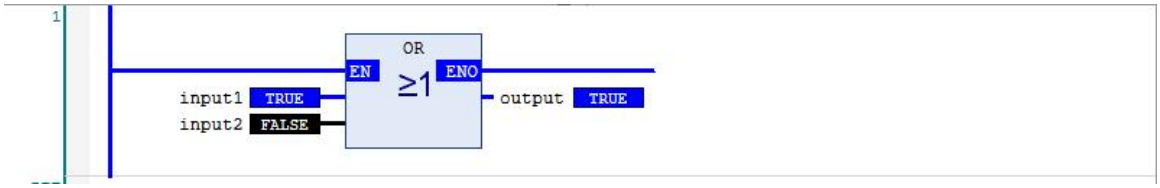
  - 布尔“或”：布尔“或”运算指令是计算两个布尔表达式的“或”结果。当两个布尔表达式中有一个表达式返回为真时，则结果位真；当两个表达式的结果都是假时，则结果为假。

- **程序示例**

ST：



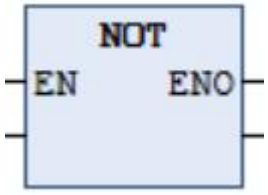
LD：



3.5.4 NOT

非运算，可以分为：按位逻辑运算以及布尔逻辑运算。

- **指令格式**

指令	名称	FB/FC	LD表现	ST表现	库
NOT	非指令	FC		NOT	-

- **相关变量**

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
IN1	输入1	-	-	0	数据1

输出变量

输入变量	名称	数据类型	有效范围	初始值	描述
OUT	输出结果	-	-	0	输出结果

	布尔	位串				整数								实数		时刻、持续时间 日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
IN1	√	√	√	√	√	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-
OUT	√	√	√	√	√	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-

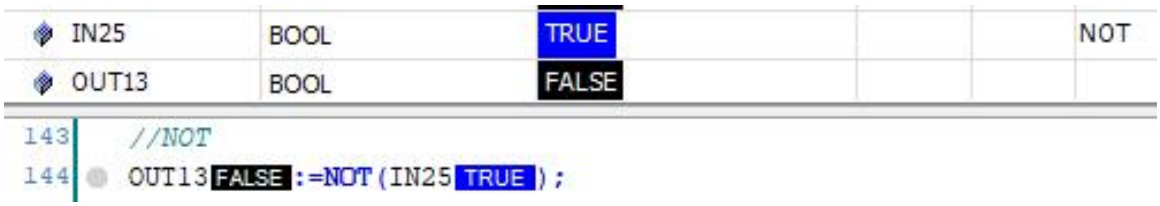
- **功能说明**

  - 按位“非”：按逻辑进行取反，将当前的值由“0”变为“1”，或由“1”变为“0”。按位“非”运算指令是将变量或常亮逐一取非。

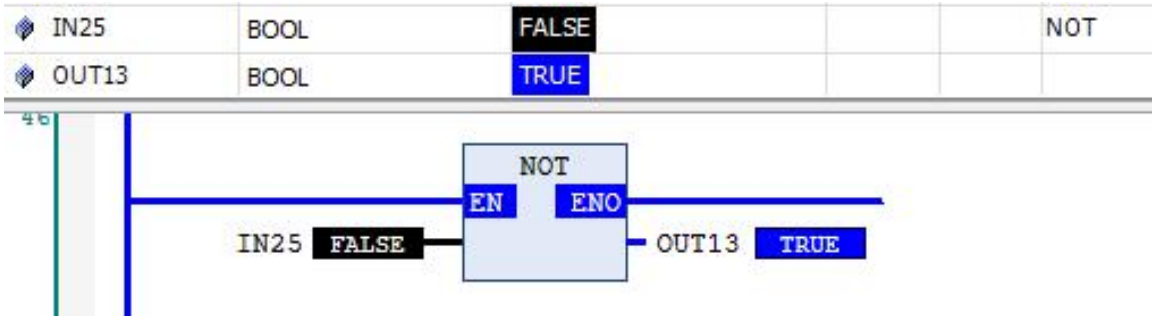
  - 布尔“非”：布尔“非”运算指令用于计算单个布尔表达式的结果。当输入为真时，结果为假；当输入为假时，结果为真。

- **程序示例**

ST：



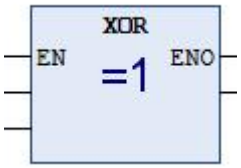
LD：



3.5.5 XOR

按位“异或”运算指令比较两个整数的相应位。

- **指令格式**

指令	名称	FB/FC	LD表现	ST表现	库
XOR	异或指令	FC		XOR	-

- **相关变量**

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
IN1	输入1	-	-	0	数据1
IN2	输入2	-	-	0	数据2

输出变量

输入变量	名称	数据类型	有效范围	初始值	描述
OUT	输出结果	-	-	0	输出结果

	布尔	位串				整数								实数		时刻、持续时间 日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
IN1	√	√	√	√	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
IN2	√	√	√	√	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
OUT	√	√	√	√	√	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-

- **功能说明**

  - 当两个整数的相应位一个是“1”而另一个是“0”时，返回相应的结果位是“1”。当两个整数的相应位都是“1”或者都是“0”时，则返回相应的结果位是“0”。

- **程序示例**

ST：

```
output:=input1 XOR input2;
```

LD：

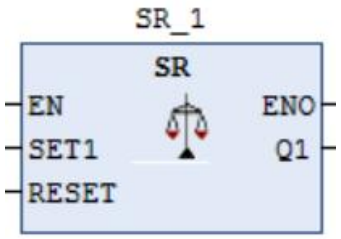


3.5.6 SR

SR是“置位优先”型触发器指令，当置位信号（SET1）和复位信号（RESET）同时为1时，触发器最终为置位状态。

- **指令格式**

指令	名称	FB/FC	LD表现	ST表现	库

SR	置位优先指令	FB		SR_1(SET1:=, RESET:=, Q1=>);	-

- **相关变量**

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
SET1	置位	BOOL	TRUE - FALSE	FALSE	数据1
RESET	复位	BOOL	TRUE - FALSE	FALSE	数据2

输出变量

输入变量	名称	数据类型	有效范围	初始值	描述
Q1	输出值	BOOL	TRUE - FALSE	FALSE	置位优先输出

	布尔	位串				整数								实数		时刻、持续时间 日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
SET1	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
RESET	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Q1	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-

- **功能说明**

  - 当SET1为“1”时，无论RESET是否为“1”，Q1输出值都为“1”。当SET1为“0”时，如果Q1输出为“1”，一旦RESET为“1”，Q1输出立刻复位为“0”；如果Q1输出为“0”，无论RESET为“1”还是“0”，Q1输出保持为“0”。

- **程序示例**

ST：

```
//----------SR
SR_0(SET1 TRUE := set1 TRUE , RESET TRUE :=reset1 TRUE , Q1 TRUE =>Q3 TRUE );
```

LD：



3.5.7 RS

RS是“复位优先”型触发器指令，当置位信号（SET1）和复位信号（RESET）同时为1时，触发器最终为复位状态。

- **指令格式**

指令	名称	FB/FC	LD表现	ST表现	库
RS	复位优先指令	FB	RS EN SET RESET1 ENO Q1	RS_1(SET1:=, RESET:=, Q1=>);	-

- **相关变量**

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
SET	置位	BOOL	TRUE - FALSE	FALSE	数据1
RESET1	复位	BOOL	TRUE - FALSE	FALSE	数据2

输出变量

输入变量	名称	数据类型	有效范围	初始值	描述

Q1	输出值	BOOL	TRUE - FALSE	FALSE	复位优先输出

	布尔	位串				整数								实数		时刻、持续时间 日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
SET1	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
RESET	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Q1	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-

- **功能说明**

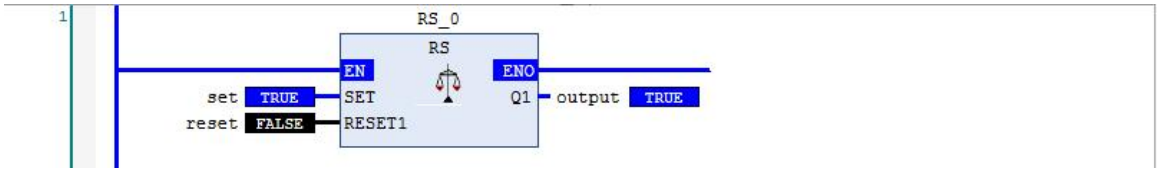
  - 当RESET1为“1”时，无论SET是否为“1”，Q1输出值都为“0”。当RESET1为“0”时，如果Q1输出为“0”，一旦SET为“1”，Q1输出立刻复位为“1”；如果Q1输出为“1”，无论RESET为“1”还是“0”，Q1输出保持为“0”。

- **程序示例**

ST：

```
1  RS_0(SET TRUE :=set TRUE, RESET1 FALSE :=reset FALSE, Q1 TRUE =>output TRUE );RETURN
```

LD：

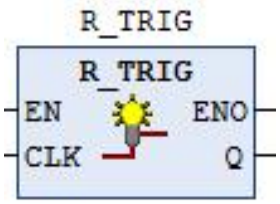


3.5.8 R_TRIG

边沿检测指令用来检测BOOL信号的上升沿（信号由0→1）和下降沿信号（信号由1→0）的变化。在每个扫描周期中，把信号状态和它再前一个扫描周期的状态进行比较，若不

同，则表明有一个跳变沿。R_TRIG指令用于检测上升沿，F_TRIG指令用于检测下降沿。

- **指令格式**

指令	名称	FB/FC	LD表现	ST表现	库
R_TRIG	上升沿检测	FB		R_TRIG(CLK:=, Q=>);	-

- **相关变量**

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
CLK	输入信号	BOOL	TRUE - FALSE	FALSE	设置为布尔量，检测其上升沿

输出变量

输入变量	名称	数据类型	有效范围	初始值	描述
Q	状态输出	BOOL	TRUE - FALSE	FALSE	如果检测到输入变量的上升沿，输出变量将在一个扫描周期内置 TRUE。

	布尔	位串				整数								实数		时刻、持续时间 日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
CLK	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Q	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-

- **功能说明**

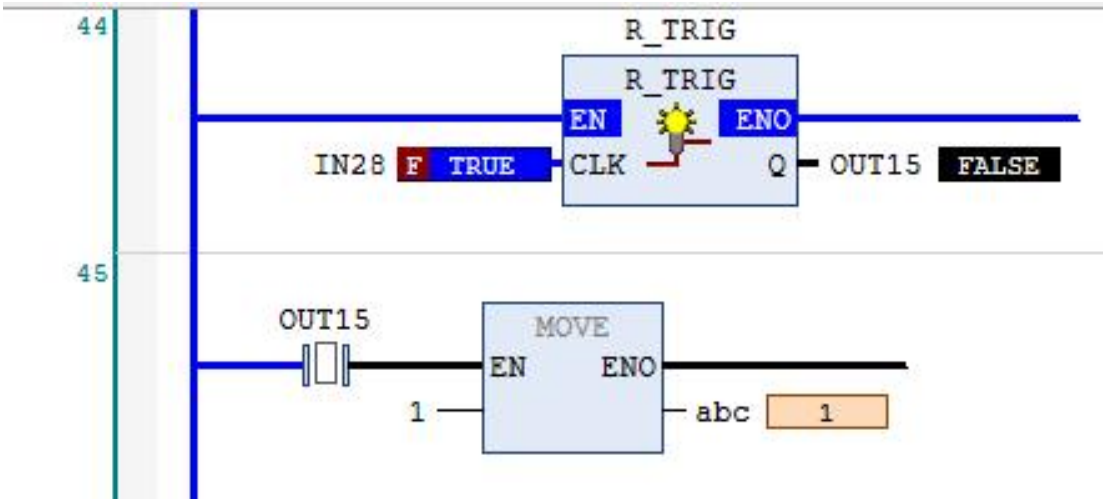
➢ 当CLK从“0”变为“1”时，该上升沿检测器开始启动，Q输出先为“1”然后输出变为“0”，持续一个PLC运算周期；如果CLK持续保持为“1”或者“0”，Q输出一直保持为“0”。

- **程序示例**

ST：

```
1  //----------R_TRIG
2  R_TRIG1(CLK TRUE :=VarBool5 TRUE , Q FALSE =>QBool FALSE );
3  IF QBool FALSE THEN
4      abc 1 :=1;
5  END_IF
```

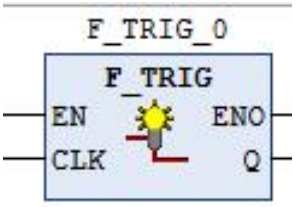
LD：



3.5.9 F_TRIG

边沿检测指令用来检测BOOL信号的上升沿（信号由0→1）和下降沿信号（信号由1→0）的变化。在每个扫描周期中，把信号状态和它再前一个扫描周期的状态进行比较，若不同，则表明有一个跳变沿。R_TRIG指令用于检测上升沿，F_TRIG指令用于检测下降沿。

- **指令格式**

指令	名称	FB/FC	LD表现	ST表现	库
F_TRIG	下降沿检测	FB		F_TRIG(CLK:=, Q=>);	-

- **相关变量**

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
CLK	输入信号	BOOL	TRUE - FALSE	FALSE	设置为布尔量，检测其上升沿

输出变量

输入变量	名称	数据类型	有效范围	初始值	描述
Q	状态输出	BOOL	TRUE - FALSE	FALSE	如果检测到输入变量的下降沿，输出变量将在一个扫描周期内置 TRUE。

	布尔	位串				整数								实数		时刻、持续时间 日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
CLK	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Q	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-

- **功能说明**

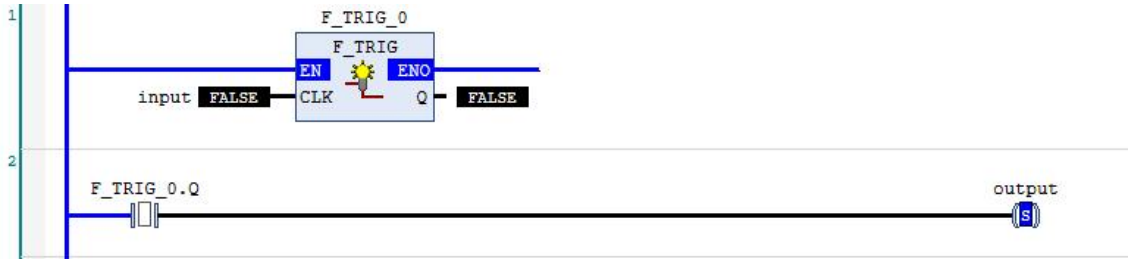
  - 当CLK从“1”变为“0”时，该下降沿检测器开始启动，Q输出先为“1”然后输出变为“0”，持续一个PLC运算周期；如果CLK持续保持为“1”或者“0”，Q输出一直保持为“0”。

- **程序示例**

ST：

```
F_TRIG_0(CLK:=input, Q=>);
IF F_TRIG_0.Q THEN
    output:=TRUE;
END_IF
```

LD：


1
F_TRIG_0
F_TRIG
EN
ENO
input
FALSE
CLK
Q
FALSE
2
F_TRIG_0.Q
output
S

3.6数据移位

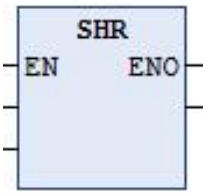
3.6.1 指令列表

指令类别	名称	FB/FC	功能
数据移位指令	SHR	FC	按位右移
	SHL	FC	按位左移
	ROR	FC	循环右移
	ROL	FC	循环左移

3.6.2 SHR

对操作数进行按位右移，右边移出位不做处理，左边空位自动补 0。

- **指令格式**

指令	名称	FB/FC	LD表现	ST表现	库
SHR	按位右移	FC		SHR(IN,N);	-

- **相关变量**

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
IN	数据来源	-	-	0	数据来源
N	移位位数	-	-	0	右移位位数

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
OUT	偏移后结果数据	-	-	0	转换后的数据

	布尔	位串				整数								实数		时刻、持续时间 日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
IN	-	√	√	√	√	√	√	√	√	√	√	√	√	-	-	-	-	-	-	-
N		√	√	√	√	√	√	√	√	√	√	√	√							
OUT	-	√	√	√	√	√	√	√	√	√	√	√	√	-	-	-	-	-	√	-

- **功能说明**

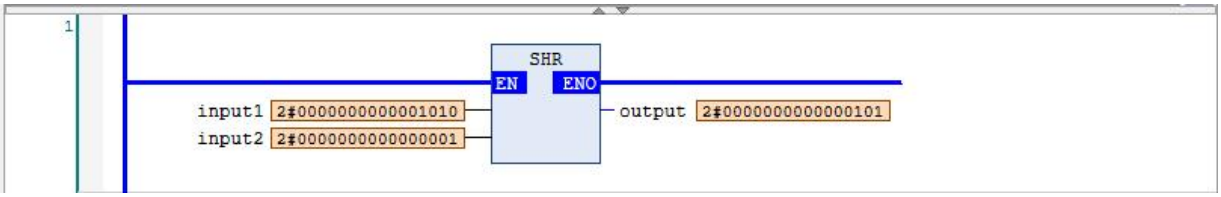
  - SHR指令可以将输入IN中的数据右移n位，输出结果赋值至OUT，二进制右移一位相当于将原数除以2.如果n大于数据类型宽度，那么BYTE、WORD、DWORD值将填为0。如果使用带符号数据类型，则最高位补符号位。

- **程序示例**

ST：

```
1  output 2#0000000000000101 :=SHR(input1 2#0000000000001010 ,input2 2#0000000000000001 );RETURN
```

LD：

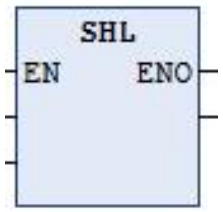


3.6.3 SHL

对操作数进行按位左移，左边移出位不做处理，右边空位自动补0。

- **指令格式**

指令	名称	FB/FC	LD表现	ST表现	库

SHL	按位左移	FC		SHL(IN,N);	-

- **相关变量**

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
IN	数据来源	-	-	0	数据来源
N	移位位数	-	-	0	左移位位数

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
OUT	偏移后结果数据	-	-	0	转换后的数据

	布尔	位串				整数								实数		时刻、持续时间日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
IN	-	√	√	√	√	√	√	√	√	√	√	√	√	-	-	-	-	-	-	-
N		√	√	√	√	√	√	√	√	√	√	√	√							
OUT	-	√	√	√	√	√	√	√	√	√	√	√	√	-	-	-	-	-	√	-

- **功能说明**

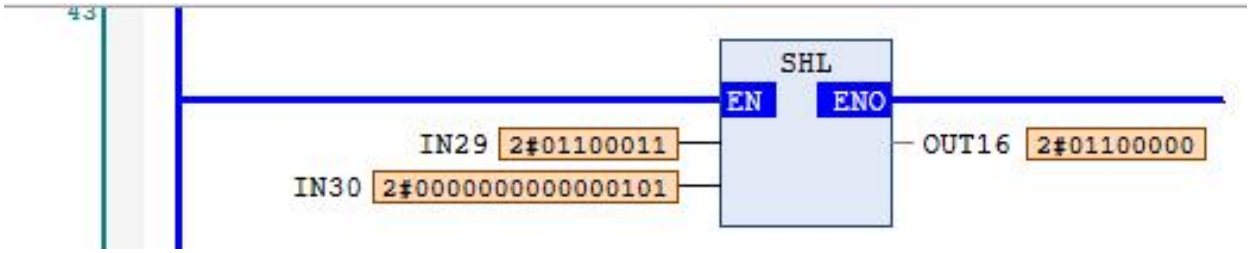
  - SHL指令可以将输入IN中的数据左移n位，输出结果赋值至OUT，二进制左移一位相当于将原数乘以2.如果n大于数据类型宽度，那么BYTE、WORD、DWORD值将填为0。

- **程序示例**

ST：

```
137   //SHL
138   OUT16 2#01100000 :=SHL(IN29 2#01100011 ,IN30 2#0000000000000101 );
```

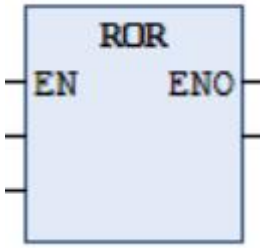
LD：



3.6.4 ROR

对操作数进行循环右移，右边移出的位直接补充到左边最高位。

- **指令格式**

指令	名称	FB/FC	LD表现	ST表现	库
ROR	循环右移	FC		ROR(IN,N);	-

- **相关变量**

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
IN	数据来源	-	-	0	数据来源
N	移位位数	-	-	0	右移位位数

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述

OUT	偏移后结果数据	-	-	0	转换后的数据

	布尔	位串				整数								实数		时刻、持续时间 日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
IN	-	√	√	√	√	√	√	√	√	√	√	√	√	-	-	-	-	-	-	-
N		√	√	√	√	√	√	√	√	√	√	√	√	-	-	-	-	-	-	-
OUT	-	√	√	√	√	√	√	√	√	√	√	√	√	-	-	-	-	-	-	-

- **功能说明**

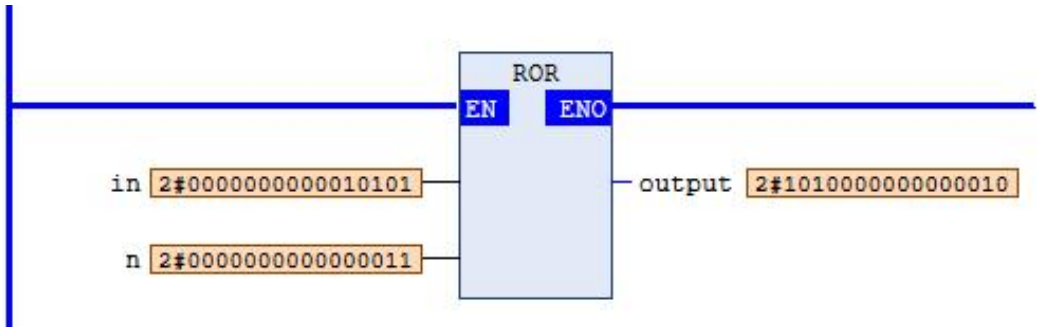
  - 使用该指令可以将输入IN中的全部内容循环地逐位右移，空出的位用移出位的信号状态填充。输入参数n提供的数值表示循环移动的位数，OUT是循环移位的操作结果。

- **程序示例**

ST：

```
output 2#1010000000000010 :=ROR(in 2#0000000000010101 ,n 2#0000000000000011 );
```

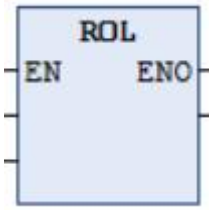
LD：



3.6.5 ROL

对操作数进行循环左移，左边移出的位直接补充到右边最高位。

- **指令格式**

指令	名称	FB/FC	LD表现	ST表现	库
ROL	循环右移	FC		ROL(IN,N);	-

- **相关变量**

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
IN	数据来源	-	-	0	数据来源
N	移位位数	-	-	0	左移位位数

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
OUT	偏移后结果数据	-	-	0	转换后的数据

	布尔	位串				整数								实数		时刻、持续时间 日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
IN	-	√	√	√	√	√	√	√	√	√	√	√	√	-	-	-	-	-	-	-
N		√	√	√	√	√	√	√	√	√	√	√	√		-	-	-	-	-	-
OUT	-	√	√	√	√	√	√	√	√	√	√	√	√	-	-	-	-	-	–	-

- **功能说明**

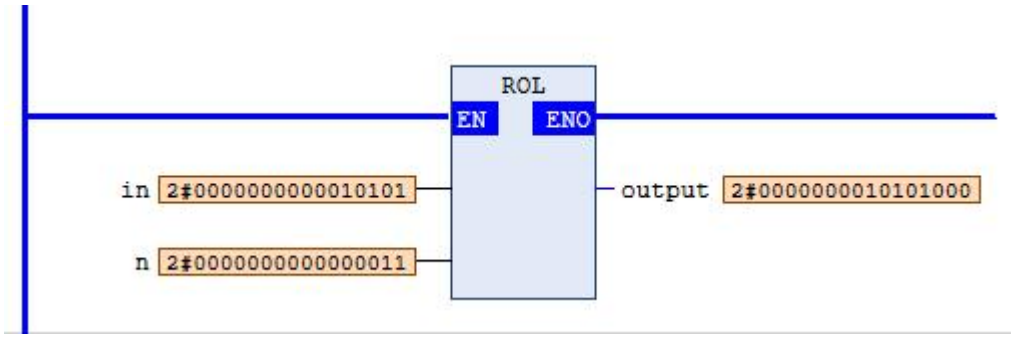
  - 使用该指令可以将输入IN中的全部内容循环地逐位左移，空出的位用移出位的信号状态填充。输入参数n提供的数值表示循环移动的位数，OUT是循环移位的操作结果。

- **程序示例**

ST：

```
output 2#0000000010101000 :=ROL(in 2#0000000000010101 ,n 2#0000000000000011 );
```

LD：



3.7数据类型转换指令

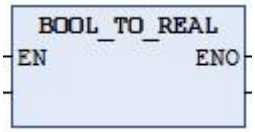
3.7.1 指令列表

指令类别	名称	FB/FC	功能
数据类型转换	BOOL_TO_<TYPE>	FC	BOOL 类型转换指令
	BYTE_TO_<TYPE>	FC	字节类型转换指令
	WORD_TO_<TYPE>	FC	字类型转换指令
	DWORD_TO_<TYPE>	FC	双字类型转换指令
	INT_TO_<TYPE>	FC	整数型类型转换指令
	SINT_TO_<TYPE>	FC	短整型转换指令
	DINT_TO_<TYPE>	FC	长整形转换指令
	UDINT_TO_<TYPE>	FC	无符号长整型转换指令
	REAL_TO_<TYPE>	FC	实数类型转换指令
	STRING_TO_<TYPE>	FC	字符类型转换指令
	TIME_TO_<TYPE>	FC	时钟类型转换指令
	TOD_TO_<TYPE>	FC	时间类型转换指令
	DATE_TO_<TYPE>	FC	日期类型转换指令
	DT_TO_<TYPE>	FC	日期时间类型转换指令

3.7.2 BOOL_TO_<TYPE>

把布尔数据类型转换为其他数据类型。

- **指令格式**

指令	名称	FB/FC	LD表现	ST表现	库
BOOL_TO_<TYPE>	布尔类型转换指令	FC		Output:= BOOL_TO_REAL(var1)	-

- **相关变量**

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
IN	数据来源	BOOL	TRUE - FALSE	FALSE	数据来源

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
OUT	结果数据	-	-	0	转换后的数据

	布尔	位串				整数								实数		时刻、持续时间 日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
IN	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
OUT	-	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√

- **功能说明**

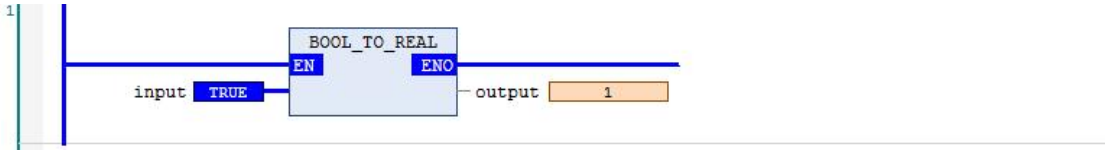
  - 支持的数据类型：BYTE、WORD、DWORD、SINT、USINT、INT、UINT、DINT、UDINT、REAL、TIME、DATE、TOD、DT和STRING.
  - 在输出为数字类型时，如果输入是TRUE，则输出为1;如果输入是FALSE，则输出为0。在输出为字符串类型时，如果输入是TRUE，则输出字符串“TRUE”；如果输入是FALSE，则输出字符串‘FALSE’。

- **程序示例**

ST：

```
output := BOOL_TO_real(input := TRUE);
RETURN
```

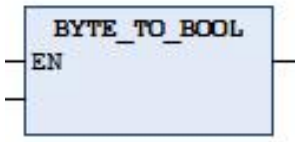
LD：



3.7.3 BYTE_TO_<TYPE>

把字节数据类型转换为其他数据类型。

- **指令格式**

指令	名称	FB/FC	LD表现	ST表现	库
BYTE_TO_<TYPE>	字节类型转换指令	FC		Output:= BYTE_TO_TYPE(var1)	-

- **相关变量**

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
IN	数据来源	BYTE	0 - 255	0	数据来源

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
OUT	结果数据	-	-	0	转换后的数据

	布尔	位串				整数								实数		时刻、持续时间 日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
IN	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
OUT	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√

- **功能说明**

➢ 支持的数据类型：BOOL、WORD、DWORD、SINT、USINT、INT、UINT、DINT、UDINT、REAL、TIME、DATE、TOD、DT和STRING.

➢ 在输出为BOOL时：如果输入不等于0，输出为TRUE；如果输入等于0，输出为FALSE。输出为TIME或TOD时：输入将以毫秒为单位进行替换。输出为DATE或DT时：输入将以

秒为单位进行转换。

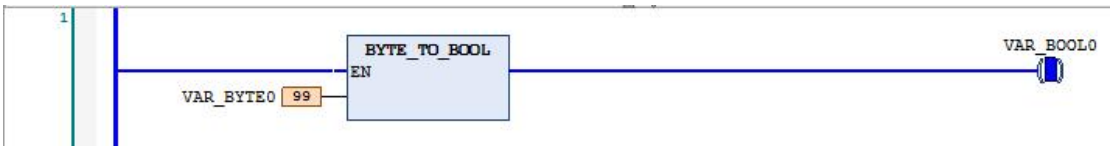
- **程序示例**

以BYTE_TO_BOOL为例

ST：

```
//----------BYTE_TO_BOOL
VAR_BOOL0 TRUE := BYTE_TO_BOOL(VAR_BYTE0 10);
```

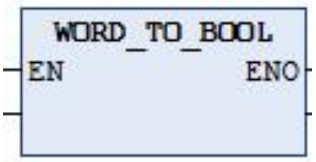
LD：



3.7.4 〈整型数据〉_TO_<TYPE>

把整型数据类型转换为其他数据类型。由于整型数据类型很多，且转换过程相似，故以WORD_TO_BOOL举例。

- **指令格式**

指令	名称	FB/FC	LD表现	ST表现	库
WORD_TO_BOOL	字节类型转换指令	FC		OUT:= BYTE_TO_BOOL(IN)	-

- **相关变量**

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
IN	数据来源	WORD	0 - 65535	0	数据来源

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
OUT	结果数据	BOOL	0 - 255	0	转换后的数据

	布尔	位串				整数								实数		时刻、持续时间 日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
IN	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
OUT	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

- **程序示例**

以WORD_TO_BOOL为例

ST：



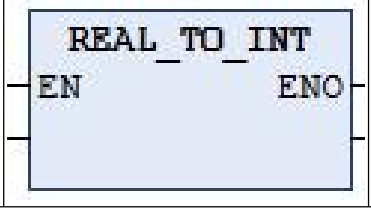
LD：



3.7.5 〈实数型数据〉_TO_〈TYPE〉

把实数型数据类型转换为其他数据类型。由于可转换的数据类型较多，且转换过程相似，故以REAL_TO_INT举例。

- **指令格式**

指令	名称	FB/FC	LD表现	ST表现	库
REAL_TO_INT	REAL 类型转换为 INT 型	FC		OUT:= REAL_TO_INT(IN)	-

- **相关变量**

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
IN	数据来源	REAL			数据来源

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
OUT	结果数据	INT			转换后的数据

	布尔	位串				整数								实数		时刻、持续时间 日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
IN	-	-	√	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-
OUT	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-

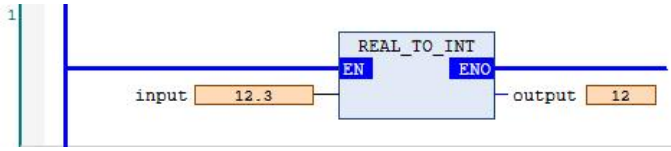
- **程序示例**

以REAL_TO_INT为例

ST：



LD：


1
REAL_TO_INT
EN
ENO
input
12.3
output
12

3.8数据处理指令

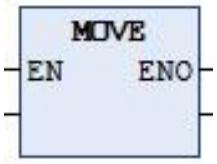
3.8.1 指令列表

指令类别	名称	FB/FC	功能
数据处理指令	MOVE	FC	赋值
	HEXinASCII_TO_BYTE	FC	ASCII 转数字指令
	BYTE_TO_HEXinASCII	FC	数字转 ASCII 指令
	WORD_AS_STRING	FC	数字转字符串指令
	BYTE_TO_HEXSTRING	FC	BYTE 类型转 16 进制字符串
	WORD_TO_HEXSTRING	FC	WORD 类型转 16 进制字符串
	DWORD_TO_HEXSTRING	FC	DWORD 类型转 16 进制字符串

3.8.2 MOVE

此运算符用于将一个变量的值分配到另一个相同类型的变量中。

- **指令格式**

指令	名称	FB/FC	LD表现	ST表现	库
MOVE	赋值	FC		A2:=MOVE(A1)	-

- **相关变量**

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
IN	传送数据	-	-	0	输入数据

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
OUT	接收数据	-	-	0	输出数据

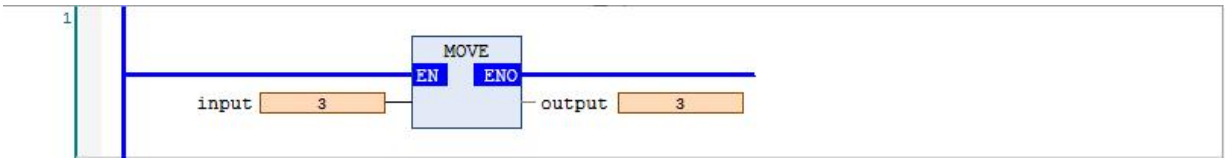
	布尔	位串				整数								实数		时刻、持续时间 日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
IN	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√
OUT	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√

- **程序示例**

ST：

```
1  output 3 :=MOVE(input 3 );RETURN
```

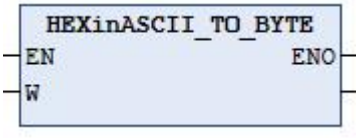
LD：



3.8.3 HEXinASCII_TO_BYTE

此运算符将16进制的ASCII码转换成BYTE类型数据。

- **指令格式**

指令	名称	FB/FC	LD表现	ST表现	库
HEXinASCII_TO_BYTE	ASCII码转BYTE字符	FC		output:= HEXinASCII_TO_BYTE(var)	-

- **相关变量**

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
IN	输入数据	WORD	0 - FFFF	0	输入数据

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
OUT	输出数据	BYTE	0 - 255	0	输出数据

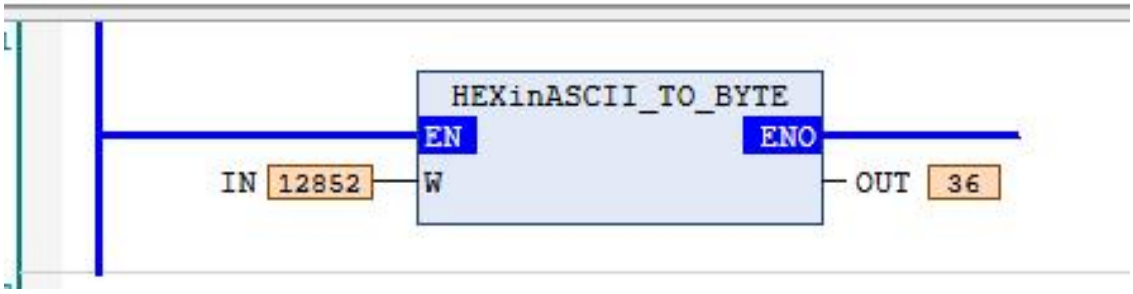
	布尔	位串				整数								实数		时刻、持续时间 日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
W	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
OUT	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

- 程序示例

ST：

```
1  out 36 := HEXinASCII_TO_BYTE(IN 12852);RETURN
```

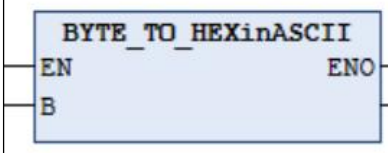
LD：



3.8.4 BYTE_TO_HEXinASCII

此运算符将BYTE类型数据转换成16进制的ASCII码。

- 指令格式

指令	名称	FB/FC	LD表现	ST表现	库
BYTE_TO_HEXinASCII	BYTE 字符转 ASCII 码	FC		output:= BYTE_TO_HEXinASCII (B:=var)	Util

- 相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
IN	输入数据	BYTE	0 - 255	0	输入数据

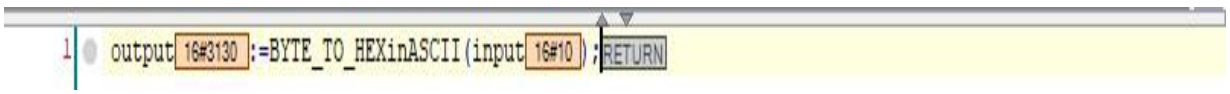
输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
OUT	输出数据	WORD	0 - FFFF	0	输出数据

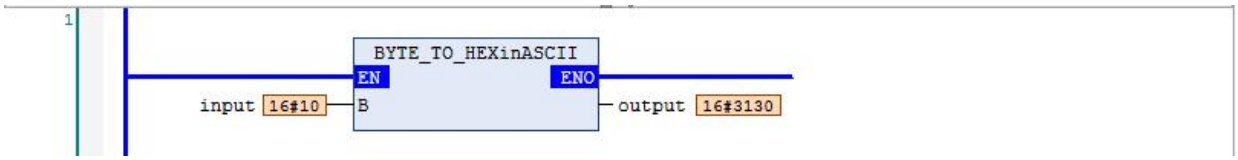
	布尔	位串				整数								实数		时刻、持续时间 日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
W	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
OUT	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

- 程序示例

ST：



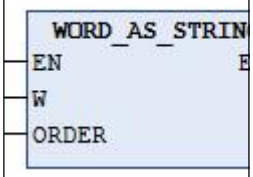
LD：



3.8.5 WORD_AS_STRING

此运算符将BYTE类型数据转换成16进制的ASCII码。

- **指令格式**

指令	名称	FB/FC	LD表现	ST表现	库
WORD_AS_STRING	WORD字符转STRING码	FC		STR:=WORD_AS_STRING(val_1, ORDER)	Util

- **相关变量**

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
W	输入数据	WORD	0 – FFFF	0	输入数据
ORDER	高低字节转换	BOOL	FALSE-TRUE	FALSE	用于转换变量的高低字节

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
OUT	输出数据	STRING	-	‘’	输出数据

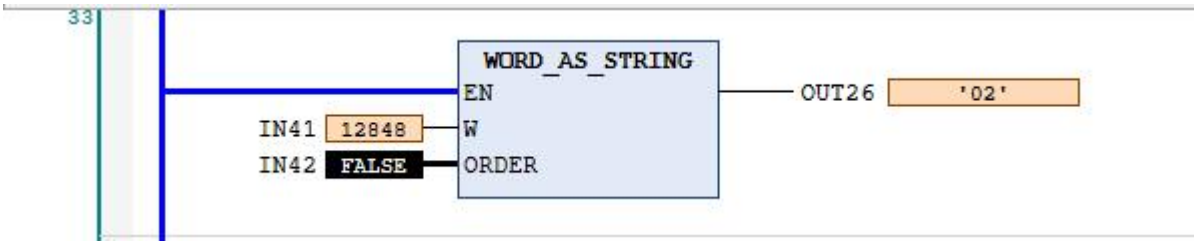
	布尔	位串				整数								实数		时刻、持续时间 日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
W	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
ORDER	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
OUT	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√

- **程序示例**

ST：

```
17  //WORD_AS_STRING
18  out57 '1' :=WORD_AS_STRING(IN57 49 ,IN58 FALSE );
```

LD：



- **注意事项**

➢ 该指令在不使用 ORDER 引脚切换高低位的情况下，不修改数值。使用 ORDER 引脚为 True 时，则切换数值的高低位。

➢ STRING默认80字符量，可定义变量为STRING（具体字符量）。

3.8.6 BYTE_TO_HEXSTRING

此运算符将BYTE类型数据转换成16进制的ASCII码。

- **指令格式**

指令	名称	FB/FC	LD表现	ST表现	库
BYTE_TO_HEXSTRING	BYTE类型转换为16进制字符串指令	FC	Byte_To_HexString EN ENO B	STR:=BYTE_TO_HEXSTRING(val)	Util

- **相关变量**

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述

B	输入数据	BYTE	0 – 255	0	输入数据

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
OUT	输出数据	STRING	-	‘’	输出数据

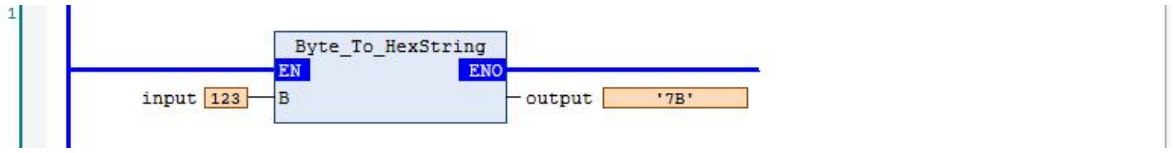
	布尔	位串				整数								实数		时刻、持续时间 日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
B	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
OUT	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√

- **程序示例**

ST：

```
1  output 'B' :=Byte_To_HexString(input 123 );RETURN
```

LD：



3.8.7 WORD_TO_HEXSTRING

此运算符将WORD类型数据转换成16进制的ASCII码。

- **指令格式**

指令	名称	FB/FC	LD表现	ST表现	库
WORD_TO_HEXSTRING	BYTE类型转换为16进制字符串指令	FC	Word_To_HexStrin EN W	STR:=WORD_TO_HEXSTRING(W)	Util

- **相关变量**

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
W	输入数据	WORD	0 – FFFF	0	输入数据

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
OUT	输出数据	STRING	-	‘’	输出数据

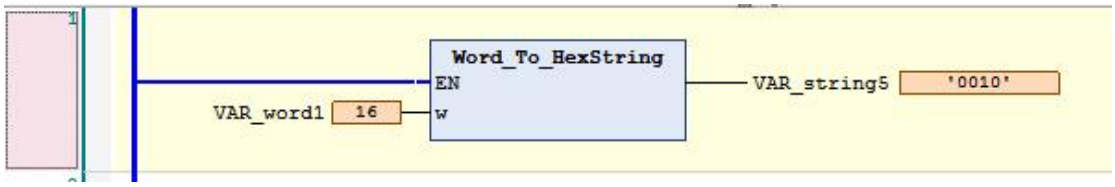
	布尔	位串				整数								实数		时刻、持续时间 日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
W	-	√	√	-	-	√	√	-	-	√	√	-	-	-	-	-	-	-	-	-
OUT	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√

- **程序示例**

ST：

```
1  //Word_To_HexString
2  VAR_string5 '0010' :=Word_To_HexString(VAR_word1 16);
```

LD：

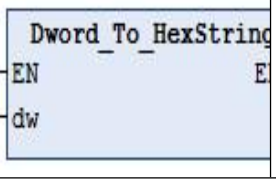


3.8.8 DWORD_TO_HEXSTRING

此运算符将DWORD类型数据转换成16进制的ASCII码。

- **指令格式**

指令	名称	FB/FC	LD表现	ST表现	库

DWORD_TO_HEXSTRING	BYTE 类型转换为 16 进制字符串指令	FC		STR:=DWORD_TO_HEXSTRING(DW)	Util

- **相关变量**

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
DW	输入数据	DWORD	0 – FFFF FFFF	0	输入数据

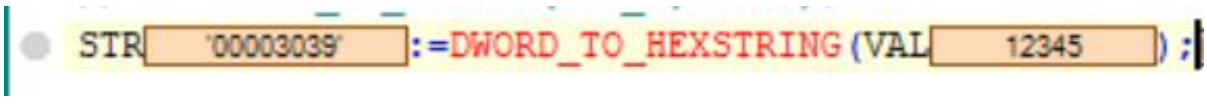
输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
OUT	输出数据	STRING	-	‘’	输出数据

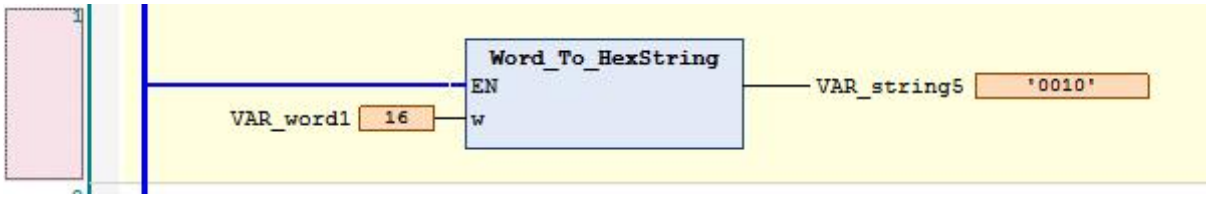
	布尔	位串				整数								实数		时刻、持续时间 日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
DW	-	√	√	-	-	√	√	-	-	√	√	-	-	-	-	-	-	-	-	-
OUT	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√

- **程序示例**

ST：



LD：



3.9数学运算指令

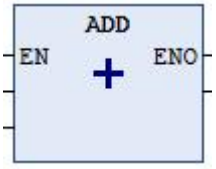
3.9.1 指令列表

指令类别	名称	FB/FC	功能
数学运算指令	ADD	FC	加法
	SUB	FC	减法
	MUL	FC	乘法
	DIV	FC	除法
	MOD	FC	取余
	ABS	FC	绝对值
	SQRT	FC	平方根
	LN	FC	自然对数
	LOG	FC	常用对数
	EXP	FC	指数
	EXPT	FC	幂指数
	SIN	FC	正弦
	COS	FC	余弦
	TAN	FC	正切
	ASIN	FC	反正弦
	ACOS	FC	反余弦
	ATAN	FC	反正切

3.9.2 ADD

将输入的变量相加，输出相加的结果

- **指令格式**

指令	名称	FB/FC	LD表现	ST表现	库
ADD	加法指令	FC		OUT:=IN1 + IN2	-

- **相关变量**

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
IN1	相加值	-	通用数据类型	-	输入
IN2	相加值	-	通用数据类型	-	输入

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
OUT	输出值	-	通用数据类型	-	输出

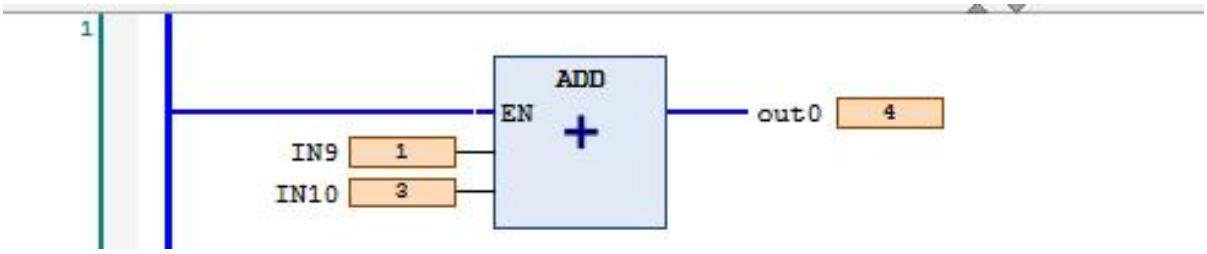
	布尔	位串				整数								实数		时刻、持续时间 日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
IN1	-	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	-	-	-	-
IN2	-	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	-	-	-	-
OUT	-	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	-	-	-	-

- 程序示例

ST：

```
1  //ADD
2  out0 := IN9 + IN10;
```

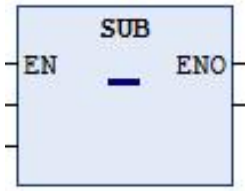
LD：



3.9.3 SUB

将输入的变量相减，输出相减的结果

- **指令格式**

指令	名称	FB/FC	LD表现	ST表现	库
SUB	减法指令	FC		OUT:=IN1 - IN2	-

- **相关变量**

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
IN1	相减值	-	通用数据类型	-	输入
IN2	相减值	-	通用数据类型	-	输入

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
OUT	输出值	-	通用数据类型	-	相减后的结果

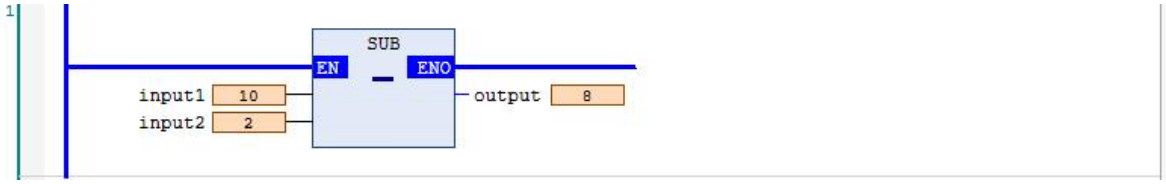
	布尔	位串				整数								实数		时刻、持续时间 日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
IN1	-	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	-	-	-	-
IN2	-	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	-	-	-	-
OUT	-	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	-	-	-	-

- **程序示例**

ST：



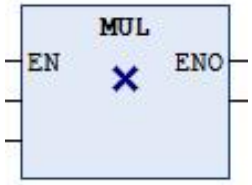
LD：



3.9.4 MUL

变量或常量的乘法运算

- **指令格式**

指令	名称	FB/FC	LD表现	ST表现	库
MUL	乘法指令	FC		OUT:=IN1 * IN2	-

- **相关变量**

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
IN1	相乘值	-	通用数据类型	-	输入
IN2	相乘值	-	通用数据类型	-	输入

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
OUT	输出值	-	通用数据类型	-	相减后的结果

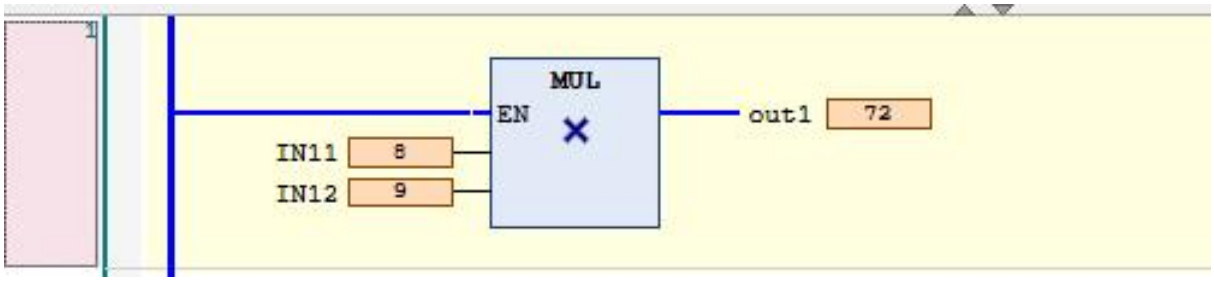
	布尔	位串				整数								实数		时刻、持续时间 日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
IN1	-	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	-	-	-	-
IN2	-	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	-	-	-	-
OUT	-	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	-	-	-	-

- **程序示例**

ST：

```
1   //MUL
2   out2 18 :=IN11 2 *IN12 9 ;
3   //ADD
```

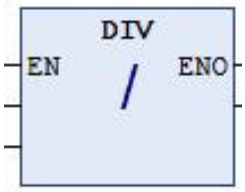
LD：



3.9.5 DIV

两个输入相除

- **指令格式**

指令	名称	FB/FC	LD表现	ST表现	库
DIV	除法指令	FC		OUT:=IN1 / IN2	-

- **相关变量**

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
IN1	被除数	-	通用数据类型	-	输入
IN2	除数	-	通用数据类型	-	输入

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述

OUT	输出值	-	通用数据类型	-	相减后的结果

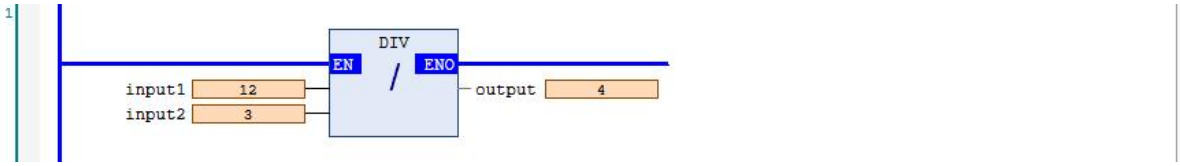
	布尔	位串				整数								实数		时刻、持续时间 日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
IN1	-	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	-	-	-	-
IN2	-	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	-	-	-	-
OUT	-	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	-	-	-	-

- **程序示例**

ST：

```
1  output:=input1 / input2;RETURN
```

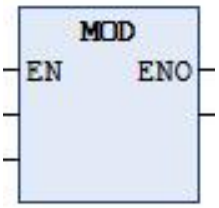
LD：



3.9.6 MOD

计算两个输入相除后非负额整数余数。

- **指令格式**

指令	名称	FB/FC	LD表现	ST表现	库
MOD	取余指令	FC		OUT:=IN1 MOD IN2	-

- **相关变量**

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
IN1	被除值	-	通用数据类型	-	输入
IN2	除值	-	通用数据类型	-	输入

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
OUT	余数	-	通用数据类型	-	输出

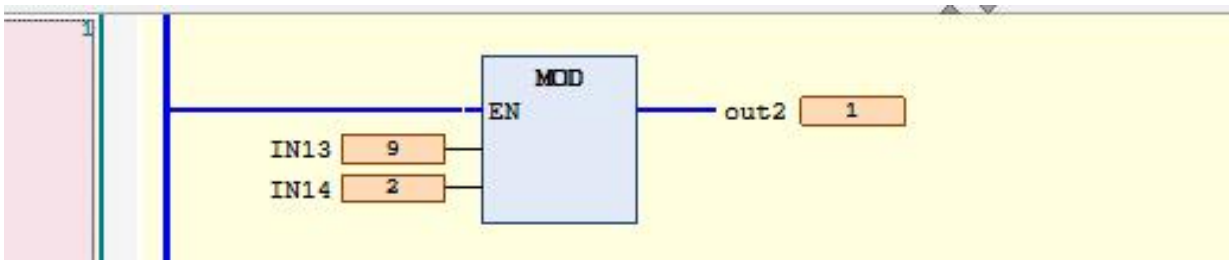
	布尔	位串				整数								实数		时刻、持续时间 日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
IN1	-	√	√	√	√	√	√	√	√	√	√	√	√	-	-	-	-	-	-	-
IN2	-	√	√	√	√	√	√	√	√	√	√	√	√	-	-	-	-	-	-	-
OUT	-	√	√	√	√	√	√	√	√	√	√	√	√	-	-	-	-	-	-	-

- **程序示例**

ST：

```
9   //MOD
10  out3 1 :=IN13 9 MOD IN14 2 ;
```

LD：



3.9.7 ABS

将输入数据取绝对值后赋值给输出变量

- **指令格式**

指令	名称	FB/FC	LD表现	ST表现	库
ABS	绝地址指令	FC	ABS EN ENO	OUT:=ABS(IN)	-

- **相关变量**

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
IN	输入	-	通用数据类型	-	输入

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
OUT	输出	-	通用数据类型	-	输出

	布尔	位串	整数	实数	时刻、持续时间 日期、字符串

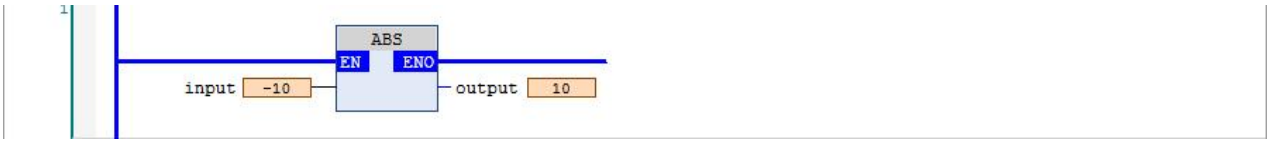
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
IN1	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√
OUT	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√

- **程序示例**

ST：

```
output:=ABS(input);RETURN
```

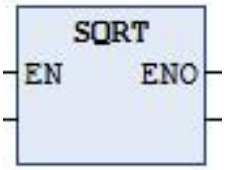
LD：



3.9.8 SQRT

将输入数据进行平方根运算，输出平方根运算结果

- **指令格式**

指令	名称	FB/FC	LD表现	ST表现	库
SQRT	计算平方根值	FC		OUT:=SQRT(IN)	-

- **相关变量**

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
IN	输入	-	通用数据类型	-	输入

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
OUT	输出	-	通用数据类型	-	输出

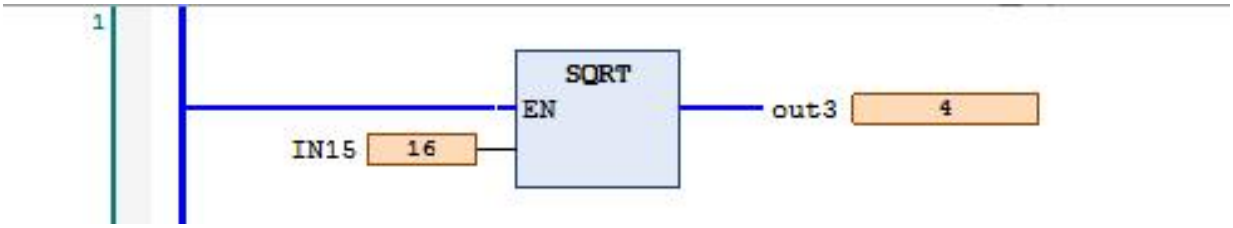
	布尔	位串				整数								实数		时刻、持续时间 日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
IN	-	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√
OUT	-	-	-	-	-	-	-	-	-	-	-	-	-	√	√	-	-	-	-	-

- **程序示例**

ST：

```
1  //SQRT
2  out4 3 :=SQRT(IN15 9);
```

LD：



3.9.9 LN

将输入数据进行对数运算，输出对数运算结果

- **指令格式**

指令	名称	FB/FC	LD表现	ST表现	库
LN	对数指令	FC	LN EN ENO	OUT:=LN(IN)	-

- **相关变量**

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
IN	输入	-	通用数据类型	-	输入

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
OUT	输出	-	通用数据类型	-	输出

	布尔	位串				整数								实数		时刻、持续时间 日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
IN	-	√	√	√	√	√	√	√	√	√	√	√	√	√	-	-	-	-	-	-
OUT	-	-	-	-	-	-	-	-	-	-	-	-	-	√	√	-	-	-	-	-

- **程序示例**

ST：

```
1  output:=LN(10);RETURN
```

LD：



3.9.10 LOG

将输入数据进行以 10 为底的对数运算，输出对数运算结果

- **指令格式**

指令	名称	FB/FC	LD表现	ST表现	库

LOG	计算以 10 为底的对数指令	FC		OUT:=LOG(IN)	-

- **相关变量**

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
IN	输入	-	通用数据类型	-	输入

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
OUT	输出	-	通用数据类型	-	输出

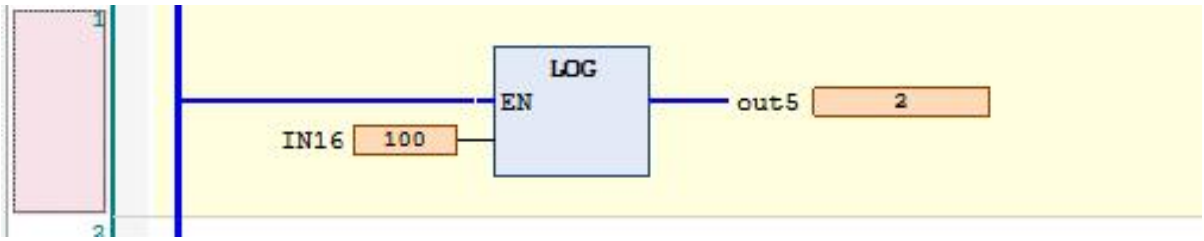
	布尔	位串				整数								实数		时刻、持续时间 日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
IN	-	√	√	√	√	√	√	√	√	√	√	√	√	√	-	-	-	-	-	-
OUT	-	-	-	-	-	-	-	-	-	-	-	-	-	√	√	-	-	-	-	-

- **程序示例**

ST：

```
1  //log
2  out5 3 :=LOG(IN16 1000 );
```

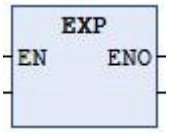
LD：



3.9.11 EXP

将输入数据进行指数运算，输出运算结果

- **指令格式**

指令	名称	FB/FC	LD表现	ST表现	库
EXP	计算以10为底的对数指令	FC		OUT:=EXP(IN)	-

- **相关变量**

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
IN	输入	-	通用数据类型	-	输入

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
OUT	输出	-	通用数据类型	-	输出

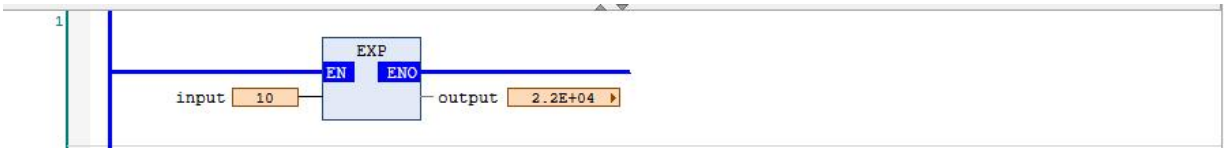
	布尔	位串				整数								实数		时刻、持续时间 日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
IN	-	√	√	√	√	√	√	√	√	√	√	√	√	√	-	-	-	-	-	-
OUT	-	-	-	-	-	-	-	-	-	-	-	-	-	√	√	-	-	-	-	-

- **程序示例**

ST：

```
1  output:=EXP(input);RETURN
```

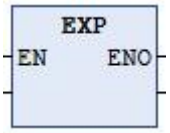
LD：



3.9.12 EXPT

将输入数据 1 作为底数，输入数据 2 作为次方数，输出幂运算的结果

- **指令格式**

指令	名称	FB/FC	LD表现	ST表现	库
EXP	幂运算指令	FC		OUT:=EXP(IN1，IN2)	-

- **相关变量**

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
IN1	输入1：底数	-	通用数据类型	-	输入
IN2	输入2：底数	-	通用数据类型	-	输入

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
OUT	输出	-	通用数据类型	-	输出

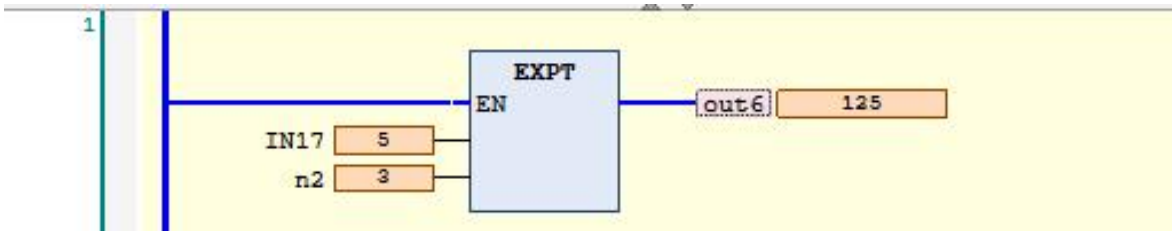
	布尔	位串				整数								实数		时刻、持续时间 日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
IN1	-	√	√	√	√	√	√	√	√	√	√	√	√	√	-	-	-	-	-	-
IN2		√	√	√	√	√	√	√	√	√	√	√	√	√						
OUT	-	-	-	-	-	-	-	-	-	-	-	-	-	√	√	-	-	-	-	-

- **程序示例**

ST：

```
1  //EXPT
2  out6 125 :=EXPT(IN17 5 ,n2 3 );
```

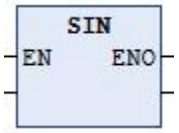
LD：



3.9.13 SIN

将输入数据作为正弦运算，输出正弦运算的结果

- **指令格式**

指令	名称	FB/FC	LD表现	ST表现	库
SIN	正弦指令	FC		OUT:=SIN(IN)	-

- **相关变量**

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
IN	输入数据	-	通用数据类型	-	输入

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
OUT	输出	-	通用数据类型	-	输出

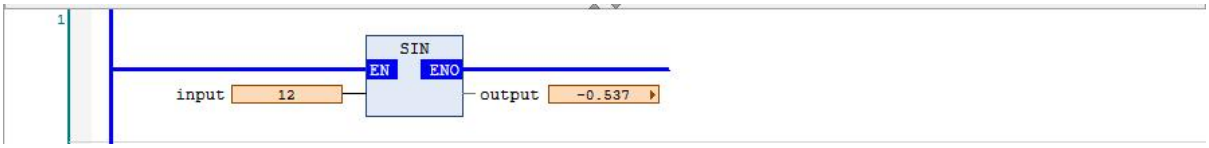
	布尔	位串				整数								实数		时刻、持续时间 日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
IN1	-	√	√	√	√	√	√	√	√	√	√	√	√	√	-	-	-	-	-	-
OUT	-	-	-	-	-	-	-	-	-	-	-	-	-	√	√	-	-	-	-	-

- **程序示例**

ST：

```
1  output:=SIN(input);RETURN
```

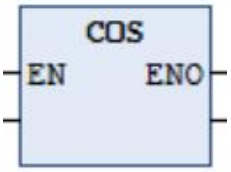
LD：



3.9.14 COS

将输入数据作为余弦运算，输出余弦运算的结果

- **指令格式**

指令	名称	FB/FC	LD表现	ST表现	库
COS	余弦指令	FC		OUT:=COS(IN)	-

- **相关变量**

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
IN	输入数据	-	通用数据类型	-	输入

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
OUT	输出	-	通用数据类型	-	输出

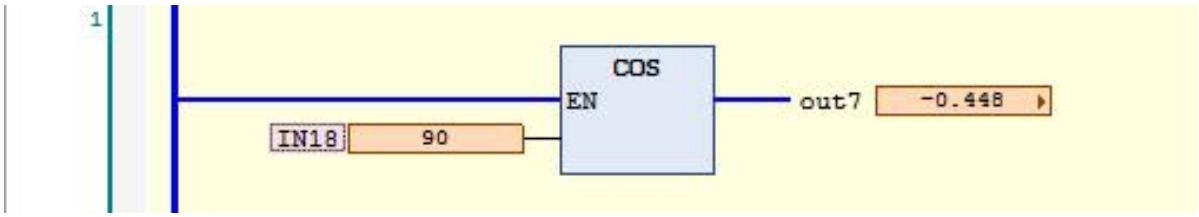
	布尔	位串				整数								实数		时刻、持续时间 日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
IN	-	√	√	√	√	√	√	√	√	√	√	√	√	√	-	-	-	-	-	-
OUT	-	-	-	-	-	-	-	-	-	-	-	-	-	√	√	-	-	-	-	-

- **程序示例**

ST：

```
1  //cos
2  out7 -0.448 :=COS(IN18 90);
```

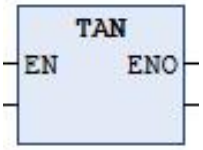
LD：



3.9.15 TAN

将输入数据作为正切运算，输出正切运算的结果

- **指令格式**

指令	名称	FB/FC	LD表现	ST表现	库
TAN	正切指令	FC		OUT:=TAN(IN)	-

- **相关变量**

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
IN	输入数据	-	通用数据类型	-	输入

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
OUT	输出	-	通用数据类型	-	输出

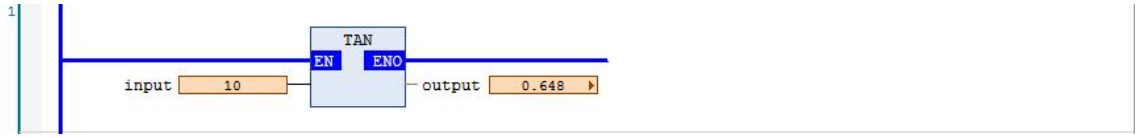
	布尔	位串				整数								实数		时刻、持续时间 日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
IN	-	√	√	√	√	√	√	√	√	√	√	√	√	√	-	-	-	-	-	-
OUT	-	-	-	-	-	-	-	-	-	-	-	-	-	√	√	-	-	-	-	-

- **程序示例**

ST：

```
1  output:=TAN(input);RETURN
```

LD：

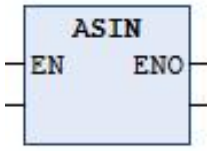


3.9.16 ASIN

将输入数据作为反正弦运算，输出反正弦运算的结果

- **指令格式**

指令	名称	FB/FC	LD表现	ST表现	库

ASIN	反正弦指令	FC		OUT:=ASIN(IN)	-

- **相关变量**

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
IN	输入数据	-	通用数据类型	-	输入

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
OUT	输出	-	通用数据类型	-	输出

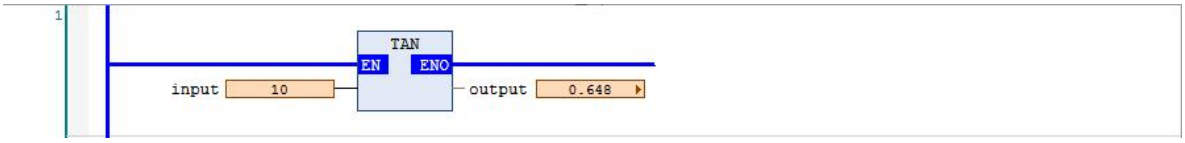
	布尔	位串				整数								实数		时刻、持续时间 日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
IN	-	√	√	√	√	√	√	√	√	√	√	√	√	√	-	-	-	-	-	-
OUT	-	-	-	-	-	-	-	-	-	-	-	-	-	√	√	-	-	-	-	-

- **程序示例**

ST：

```
1  output:=TAN(input);RETURN
```

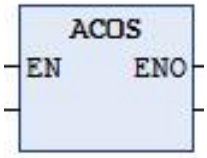
LD：



3.9.17 ACOS

将输入数据作为反余弦运算，输出反余弦运算的结果

- **指令格式**

指令	名称	FB/FC	LD表现	ST表现	库
ACOS	反余弦指令	FC		OUT:=ACOS(IN)	-

- **相关变量**

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
IN	输入数据	-	通用数据类型	-	输入

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
OUT	输出	-	通用数据类型	-	输出

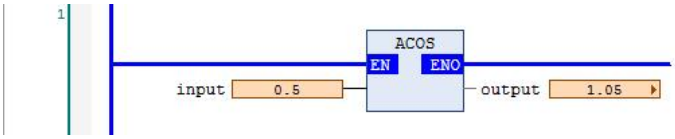
	布尔	位串				整数								实数		时刻、持续时间 日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
IN	-	√	√	√	√	√	√	√	√	√	√	√	√	√	-	-	-	-	-	-
OUT	-	-	-	-	-	-	-	-	-	-	-	-	-	√	√	-	-	-	-	-

- **程序示例**

ST：



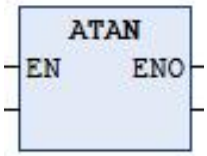
LD：



3.9.18 ATAN

将输入数据作为反正切运算，输出反正切运算的结果

- **指令格式**

指令	名称	FB/FC	LD表现	ST表现	库
ATAN	反正切指令	FC		OUT:=ATAN(IN)	-

- **相关变量**

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
IN	输入数据	-	通用数据类型	-	输入

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
OUT	输出	-	通用数据类型	-	输出

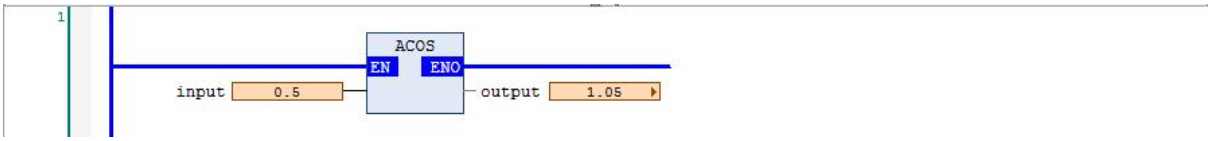
	布尔	位串				整数								实数		时刻、持续时间 日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
IN	-	√	√	√	√	√	√	√	√	√	√	√	√	√	-	-	-	-	-	-
OUT	-	-	-	-	-	-	-	-	-	-	-	-	-	√	√	-	-	-	-	-

- **程序示例**

ST：

```
97    //ATAN
98    OUT36 0.785 :=ATAN(IN56 1);
```

LD：



3.9.19 SIZEOF

用来确定输入变量所占用的字节数。SIZEOF 操作符通常返回一个无符号数。返回值的大小为输入变量的类型所占的字节数。

- **指令格式**

指令	名称	FB/FC	LD表现	ST表现	库
SIZEOF	数据类型大小	FC	SIZEOF EN ENO	OUT:=SIZEOF(IN)	-

- **相关变量**

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
IN	输入数据	-	通用数据类型	-	输入

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
OUT	输出	UINT	通用数据类型	-	输出

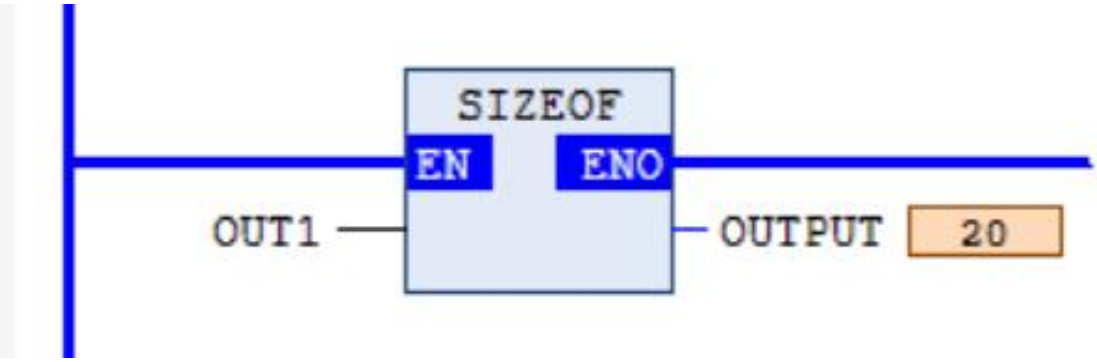
	布尔	位串				整数								实数		时刻、持续时间 日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
IN	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√
OUT	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-

- **程序示例**

ST：

```
OUTPUT1 20 :=SIZEOF(IN1);
```

LD：



3.10 字符串

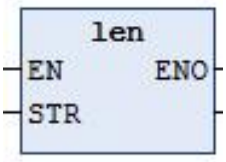
3.10.1 指令列表

指令类别	名称	FB/FC	功能
字符串指令	LEN	FC	获取字符串长度
	LEFT	FC	从左边取字符串
	RIGHT	FC	从右边取字符串
	MID	FC	从中间取字符串
	CONCAT	FC	字符串连接
	INSERT	FC	字符串插入
	DELETE	FC	字符串删除
	FIND	FC	字符串查找
	REPLACE	FC	字符串替换

3.10.2 LEN

获取字符串的长度指令

- **指令格式**

指令	名称	FB/FC	LD表现	ST表现	库
LEN	获取字符串长度指令	FC		OUT:=LEN(STR)	-

- **相关变量**

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
STR	输入	STRING	-	‘’	输入

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述

LEN	返回值	INT	0 - 65535	0	目标字符串个数

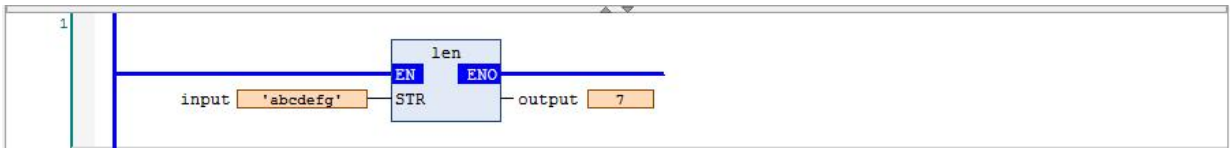
	布尔	位串				整数								实数		时刻、持续时间 日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
STR	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√
LEN	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-

- **程序示例**

ST：

```
1  output:=LEN(input);RETURN
```

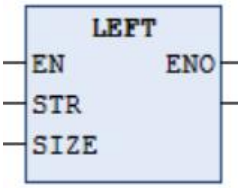
LD：



3.10.3 LEFT

从字符串左侧开始向右提取 size 个字符，返回所取长度的字符串

- **指令格式**

指令	名称	FB/FC	LD表现	ST表现	库
LEFT	取左边字符串	FC		STR1:= LEFT(STR,SIZE)	-

- **相关变量**

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述

STR	源字符串	STRING	-	‘’	输入字符串
SIZE	取字符的个数	INT	0 - 65535	0	

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
STR1	返回值	STRING	-	‘’	目标字符串个数

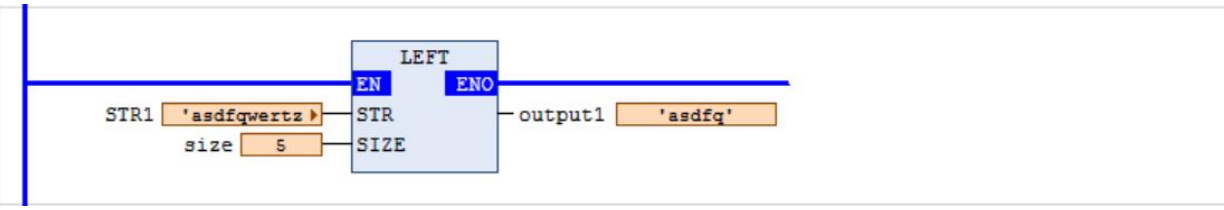
	布尔	位串				整数								实数		时刻、持续时间 日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
STR	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√
SIZE	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-
STR1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√

- **程序示例**

ST：

```
OUTPUT2 'lead' :=LEFT(STR1 'leadshine', SIZE 4);
```

LD：

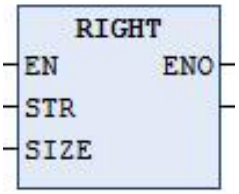


3.10.4 RIGHT

从字符串右侧开始向左提取 size 个字符，返回所取长度的字符串

- **指令格式**

指令	名称	FB/FC	LD表现	ST表现	库

RIGHT	取右边字符串	FC		STR1:= RIGHT(STR,SIZE)	-

- **相关变量**

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
STR	源字符串	STRING	-	‘’	输入字符串
SIZE	取字符的个数	INT	0 - 65535	0	

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
STR1	返回值	STRING	-	‘’	目标字符串个数

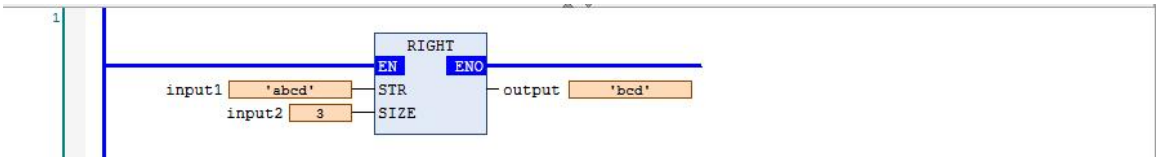
	布尔	位串				整数								实数		时刻、持续时间 日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
STR	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√
SIZE	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-
STR1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√

- **程序示例**

ST：

```
1  output:=RIGHT(input1,input2);RETURN
```

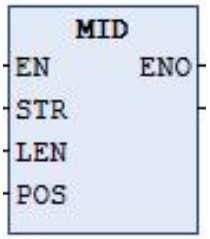
LD：



3.10.5 MID

从字符串指定位置取指定长度的字符串，返回所取长度的字符串

- **指令格式**

指令	名称	FB/FC	LD表现	ST表现	库
MID	取中间字符串	FC		STR1:= MID(STR,LEN,POS)	-

- **相关变量**

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
STR	源字符串	STRING	-	‘’	输入字符串
LEN	取字符的个数	INT	0 – 65535	0	数据长度
POS	取字符串的位置	INT	0 – 65535	0	

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
STR1	返回值	STRING	-	‘’	目标字符串个数

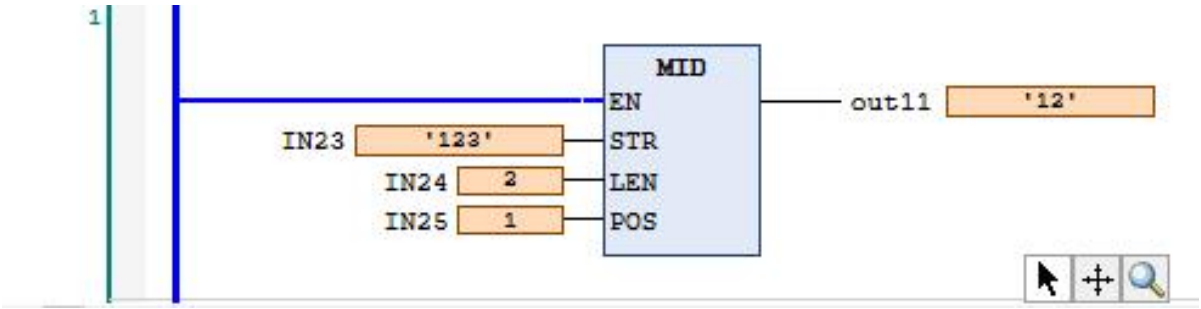
	布尔	位串				整数								实数		时刻、持续时间 日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
STR	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√
LEN	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-
POS	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-
STR1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√

- **程序示例**

ST：

```
1  //MID
2  out11 := MID (IN23 '123', IN24 2, IN25 1);
```

LD：



3.10.6 CONCAT

将输入的两字符串连接并从右侧输出

- **指令格式**

指令	名称	FB/FC	LD表现	ST表现	库
CONCAT	字符串连接	FC	CONCAT EN ENO STR1 STR2	STR:= MID(STR1,STR2)	-

- **相关变量**

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
STR1	源字符串1	STRING	-	‘’	输入字符串 1
STR2	源字符串2	STRING	-	‘’	输入字符串2

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
STR	返回值	STRING	-	‘’	输出合成的字符串

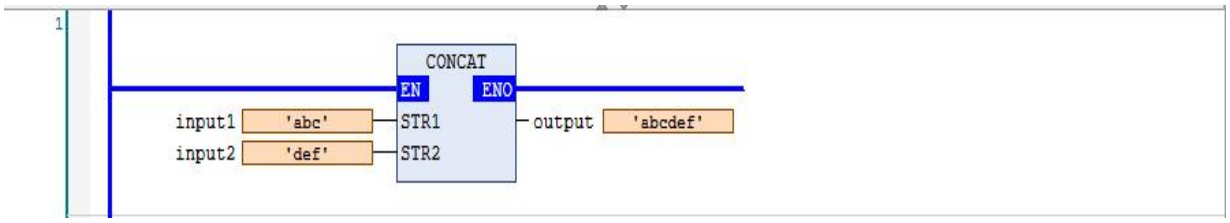
	布尔	位串				整数								实数		时刻、持续时间 日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
STR1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√
STR2	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
STR	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√

- **程序示例**

ST：

```
output:=CONCAT(input1,input2);RETURN
```

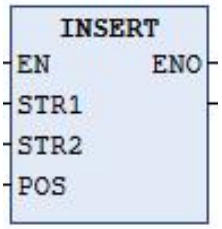
LD：



3.10.7 INSERT

在输入源字符串中插入新的字符串，合并成新的字符串输出

- **指令格式**

指令	名称	FB/FC	LD表现	ST表现	库
INSERT	插入字符串	FC		STR:= INSERT(STR1,STR2,POS)	-

- 相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
STR1	源字符串1	STRING	-	‘’	输入字符串 1
STR2	源字符串2	STRING	-	‘’	输入字符串2
POS	数据插入位置	INT	0 - 65535	0	数据插入位置

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
STR	返回值	STRING	-	‘’	输出新的字符串

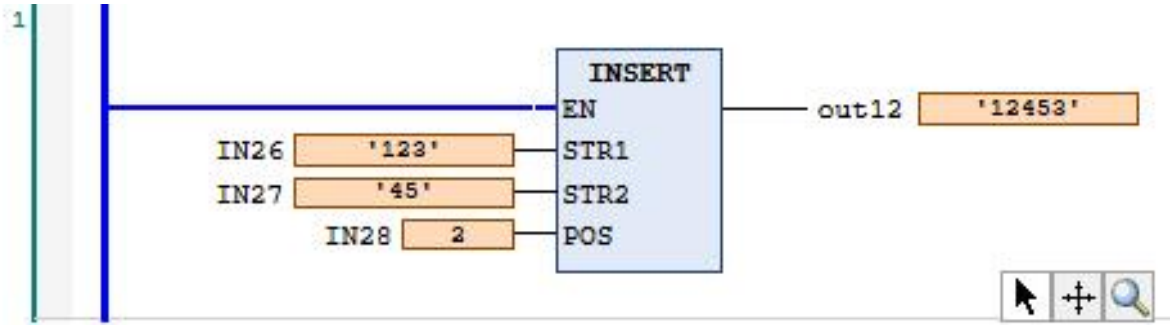
	布尔	位串				整数								实数		时刻、持续时间 日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
STR1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√
STR2	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√
POS	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-
STR	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√

- 程序示例

ST：

```
1  //INSERT
2  out12 := INSERT (IN26 '123', IN27 '45', IN28 2);
```

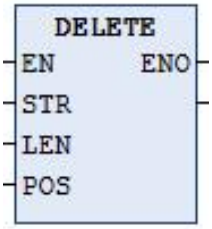
LD：



3.10.8 DELETE

在输入源字符串中删除指定位置指定长度的字符串，合并成新的字符串输出

- **指令格式**

指令	名称	FB/FC	LD表现	ST表现	库
DELETE	删除字符串	FC		STR:= DELETE(STR,LEN,POS)	-

- **相关变量**

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
STR	输入的字符串	STRING	-	‘’	输入字符串
LEN	数据长度	INT	0 - 65535	0	数据长度
POS	数据位置	INT	0 - 65535	0	删除的字符位置

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
STR	返回值	STRING	-	‘’	输出新的字符串

	布尔	位串				整数								实数		时刻、持续时间 日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
STR	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√
LEN	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-
POS	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-

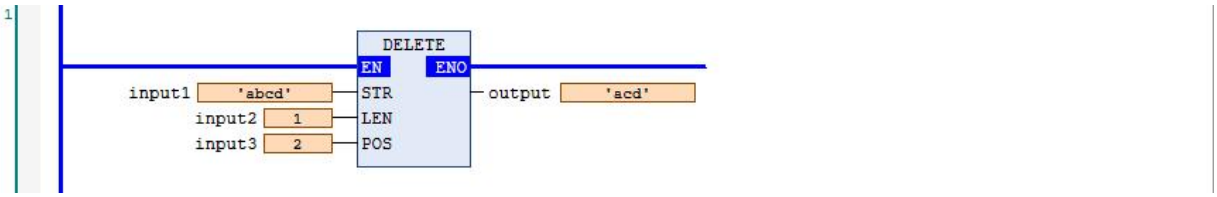
STR	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√

- **程序示例**

ST：

```
output:=DELETE(input1,input2,input3);RETURN
```

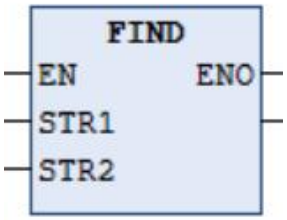
LD：



3.10.9 FIND

检测目标字符串在源字符串中的位置

- **指令格式**

指令	名称	FB/FC	LD表现	ST表现	库
FIND	查找字符串	FC		STR:= FIND(STR1,STR2)	-

- **相关变量**

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
STR1	源字符串	STRING	-	‘’	输入字符串
STR2	源字符串	STRING	-	‘’	输入字符串

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述

Output	返回值	INT	0 - 65535	0	目标字符串在源字符串中的位置

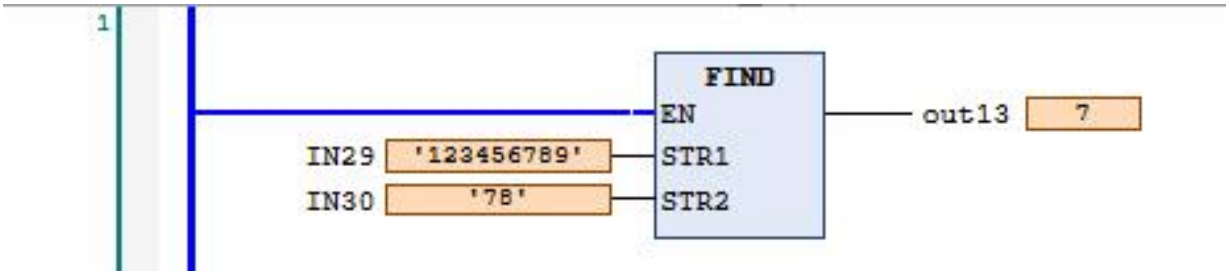
	布尔	位串				整数								实数		时刻、持续时间 日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
STR1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√
STR2	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√
Output	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-

- **程序示例**

ST：

```
//FIND
out13 7 := FIND (IN29 '123456789',IN30 '78');
```

LD：



3.10.10 REPLACE

用目标字符串替换掉原字符串中的一部分，将替换后生成的新字符串作为返回值。

- **指令格式**

指令	名称	FB/FC	LD表现	ST表现	库
REPLACE	替换字符串	FC	REPLACE EN ENO STR1 STR2 L P	STR:= REPLACE(STR1,STR2,L,P)	-

- **相关变量**

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
STR1	输入的字符串	STRING	-	‘’	输入字符串
STR2	数据长度	STRING	-	‘’	输入字符串
L	数据长度	INT	0 - 65535	0	数据长度
P	数据位置	INT	0 - 65535	0	数据位置

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
Output	返回值	STRING	-	‘’	目标字符串

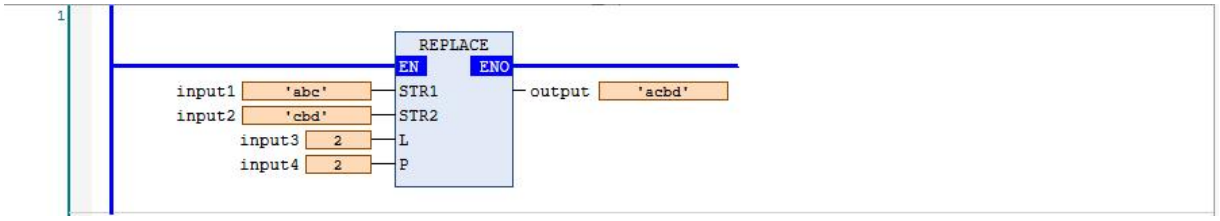
	布尔	位串				整数								实数		时刻、持续时间 日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
STR1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√
STR2	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√
L	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-
P	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-
Output	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√

- **程序示例**

ST：

```
1  output:=REPLACE(input1,input2,input3,input4);RETURN
```

LD：



3.11 地址操作

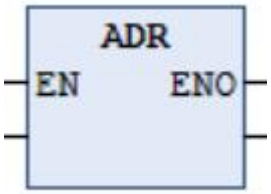
3.11.1 指令列表

指令类别	名称	FB/FC	功能
地址操作	ADR	FC	取地址指令
	^	FC	取地址内容指令
	BITADR	FC	位地址

3.11.2 ADR

取得输入变量的内存地址，结果赋值给输出变量

- **指令格式**

指令	名称	FB/FC	LD表现	ST表现	库
ADR	取地址指令	FC		OUT:=ADR(IN)	-

- **相关变量**

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
IN	目标数据	-	-	-	目标数据

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
OUT	目标数据地址	POINTER_TO_<TYPE>	0 - 65535	0	目标数据地址

	布尔	位串	整数	实数	时刻、持续时间 日期、字符串

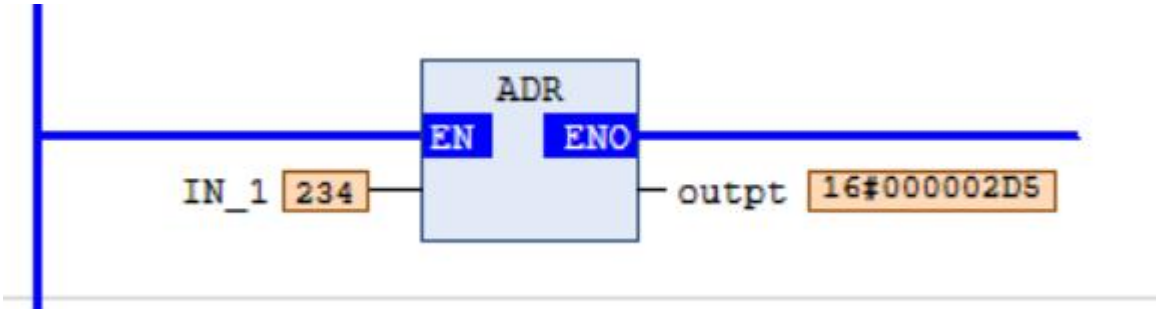
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
IN	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√
OUT	POINTER_TO_<TYPE>																			

- **程序示例**

ST：



LD：



- **注意事项：**

➢ ADR以DWORD格式返回一个引用的地址， 这个地址可以用于修改函数的操作指针或者可以用于工程内的指针操作。

3.11.3 ^

取地址内容指令

- **指令格式**

指令	名称	FB/FC	LD表现	ST表现	库
^	取地址内容	FC	^	^	-

- **相关变量**

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
OUT	指向地址的内容	-	-	‘’	指向地址内容

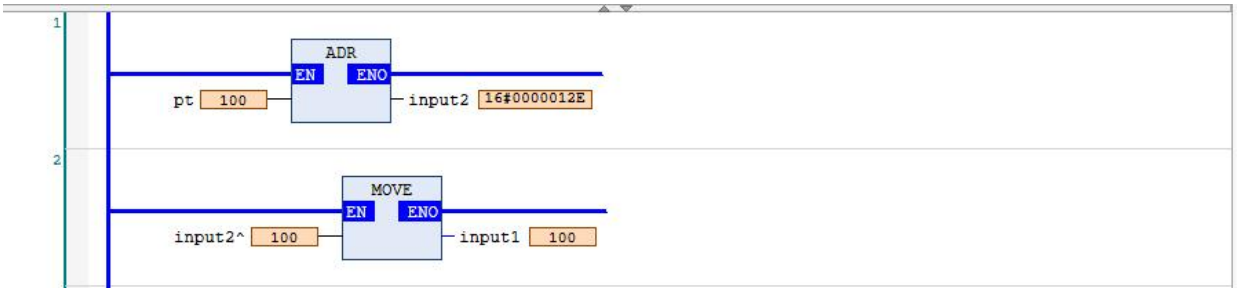
	布尔	位串				整数								实数		时刻、持续时间 日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
OUT	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

- **程序示例**

ST：

```
input2:=ADR(pt);
input1:=input2^;
```

LD：



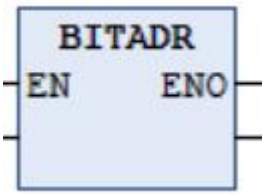
- **功能说明**

  - 程序运行后指针会自动生成参数地址，在指针后加^代表指针指向地址的内容。
  - 当进行在线更改时，地址内容会改变，将导致POINTER变量指向不可用的内存区域。为了避免问题，请在使用指针变量前，使用ADR更新一次指针值。
  - 不要将函数的POINTER_TO 变量和方法返回到调用函数或将他们分配给全局变量。

3.11.4 BITADR

取 BOOL 变量的内存地址，将结果赋值给输出变量

- **指令格式**

指令	名称	FB/FC	LD表现	ST表现	库
BITADR	位地址指令	FC		OUT:= BITADR(IN)	-

- **相关变量**

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
IN	目标数据	BOOL	-	-	目标数据

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
OUT	指向目标地址	DWORD	-	-	指向目标地址

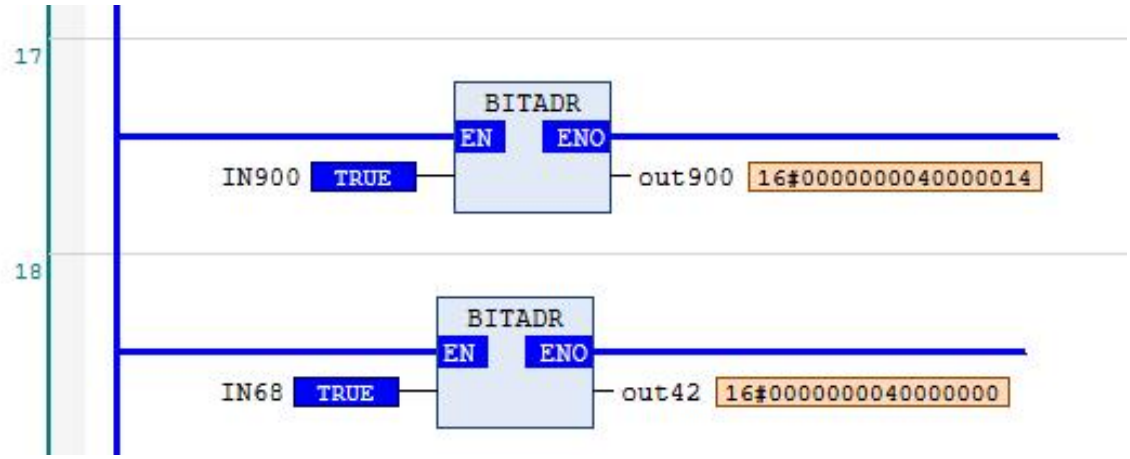
	布尔	位串				整数								实数		时刻、持续时间 日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
IN	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
OUT	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

- **程序示例**

ST：

```
85  //BITADR
86  out42 16#0000000040000000 := BITADR(IN68 TRUE);
87  //BITADR
88  out900 16#0000000040000014 := BITADR(IN900 TRUE);
```

LD：



3.12 文件操作

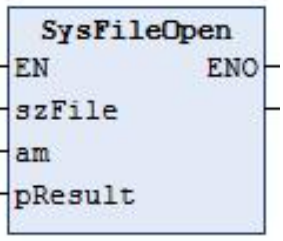
3.12.1 指令列表

指令类别	名称	FB/FC	功能
文件操作	SysFileOpen	FB	打开文件
	SysFileClose	FB	关闭文件
	SysFileWrite	FB	写入文件
	SysFileRead	FB	读取文件
	SysFileDelete	FB	删除文件
	SysFileCopy	FB	复制文件
	SysFileRename	FB	重命名文件
	SysFileSetPos	FB	设置文件读写位置
	SysFileGetPos	FB	读取文件读写位置
	SysFileGetSize	FB	获取文件大小
	File.Open	FB	按目录打开文件
	File.Close	FB	按目录关闭文件
	File.Copy	FB	按目录复制
	File.Create	FB	按目录创建
	File.Delete	FB	按目录删除

3.12.2 SysFileOpen

打开现有文件或者创建新文件

- **指令格式**

指令	名称	FB/FC	LD表现	ST表现	库
SysFileOpen	打开文件	FC		SysFileOpen(szFile:=, am:=, pResult:=);	SysFile

- **相关变量**

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
szFile	文件名	STRING	-	-	需要打开或者创建文件的路径
am	文件访问模式	ACCESS_MODE	-	-	请求的文件访问模式
pResult	错误代码的指针	POINTER TO RTS_IEC_RESULT	-		指向运行时系统错误的指针

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
hFile	返回值	RTS_IEC_HANDLE	-	-	文件句柄

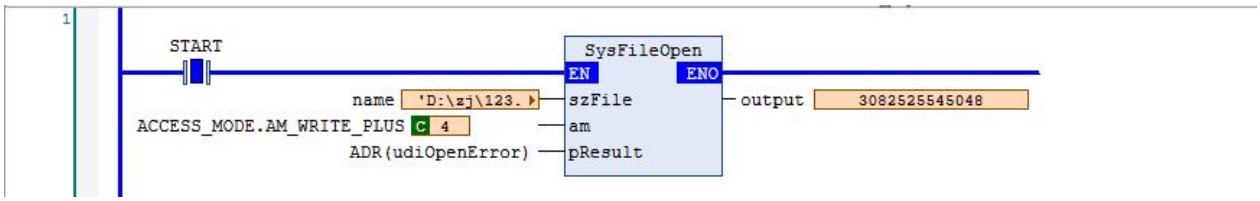
	布尔	位串				整数								实数		时刻、持续时间 日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
SZFILE	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√
AM	ACCESS_MODE,详见功能说明																			
PRESULT	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
hFile	RTS_IEC_HANDLE																			

- **程序示例**

ST：

```
IF start THEN
    SysFileOpen(szFile:=name, am:=SysFile.AM_WRITE_PLUS, pResult:=ADR(udiopenerror));
    //start:=FALSE;
END IF
```

LD：



- **功能说明**

  - “AM”的数据类型为 ACCESS_MODE，其含义如下所示：

枚举值	含义
AM_READ	打开具有读取权限的现有文件。如果文件不存在，则打开失败
AM_WRITE	创建具有写访问权限的新文件。如果文件确实存在，则丢弃内容
AM_APPEND	使用Append（仅写）访问权限打开现有文件。如果文件不存在，则打开失败
AM_READ_PLUS	打开具有读/写访问权限的现有文件。如果文件不存在，则打开失败
AM_WRITE_PLUS	创建具有读/写访问权限的新文件。如果文件确实存在，则丢弃内容
AM_APPEND_PLUS	使用Append（读/写）访问权限打开现有文件。如果文件不存在，“打开”将创建新文件

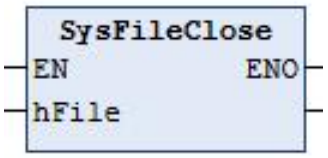
  - 请先添加“sysfile”库。
  - 本指令用于打开或创建指定路径下文件，文件路径及文件名由szFile指定；文件打开模式由am指定；返回文件句柄，作为写文件等功能块的输入；打开文件错误码请查看库CmpErrors,3.3.1.40(System)。
  - 有两个函数库中支持文件相关的操作，分别为SYSFILE和CAA.FILE，两者之间的区别在于：
    （1）文件访问机制差异，SYSFILE 为同步访问实现而 CAA .FIFLE 为异步访问。
    （2）仿真差异，SYSFILE 可进行仿真而 CAA .FIFLE 不能进行仿真。
    （3）SysFile 函数类型，File Access 为功能块类型。
    （4）文件路径差异，SYSFILE 禁止使用反斜杠。

3.12.3 SysFileClose

关闭现有文件

- **指令格式**

指令	名称	FB/FC	LD表现	ST表现	库

SysFileClose	关闭文件	FC		SysFileClose(hFile);	Sysfile

- **相关变量**

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
hFile	文件句柄	RTS_IEC_HANDLE	-	-	文件句柄

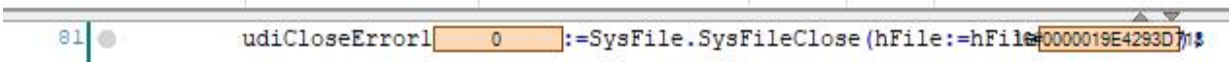
输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
FileCloseErrorID	返回值	RTS_IEC_RESULT	-	-	关闭文件错误码

	布尔	位串				整数								实数		时刻、持续时间 日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
hFile	RTS_IEC_HANDLE																			
FileCloseErrorID	RTS_IEC_RESULT																			

- **程序示例**

ST：



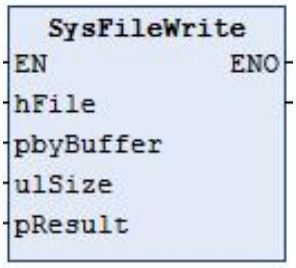
LD：



3.12.4 SysFileWrite

将字节数写入文件

- **指令格式**

指令	名称	FB/FC	LD表现	ST表现	库
SysFileWrite	文件写入指令	FC		SysFileWrite(hFile,pbyBuffer,ulSize,pResult);	SysFile

- **相关变量**

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
hFile	文件句柄	RTS_IEC_HANDLE	-	-	文件句柄
pbyBuffer	要写入的文件内容指针	POINTER TO STRING	-	-	要写入的文件内容指针
ulSize	要写入内容的字节数	__XWORD	-	-	要写入内容的字节数
pResult	写入文件内容错误码指针	POINTER TO RTS_IEC_RESULT	-	-	写入文件内容错误码指针

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
RelSentSize	返回值	__XWORD	-	-	实际写入文件数据大小，单位字节

	布尔	位串				整数								实数		时刻、持续时间 日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
hFile	RTS_IEC_HANDLE																			
pbyBuffer	POINTER TO STRING																			

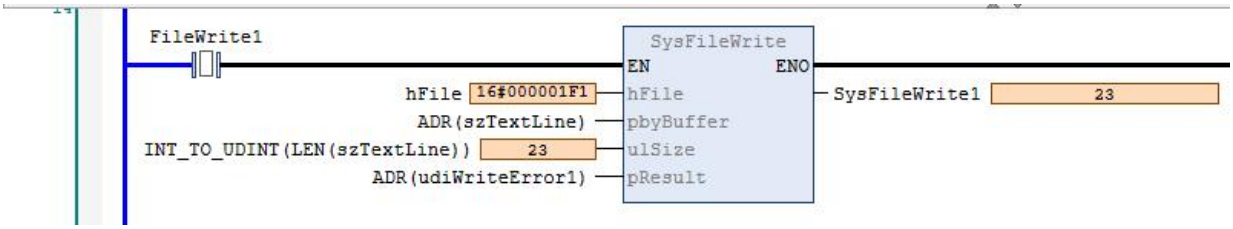
ulSize	-	-	√	√	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-
pResult	POINTER TO RTS_IEC_RESULT																		
RelSentSize	POINTER TO RTS_IEC_RESULT																		

- **程序示例**

ST：

```
77    udiWrite 23 :=SysFileWrite(hFile:=hFile 16#000019E4293D71 pbyBuffer:=ADR(szTextLine This is th ),
78    ulSize:= INT_TO_UDINT(LEN(szTextLine This is th )), pResult:= ADR(udiWriteError 16#0000000000000000);
```

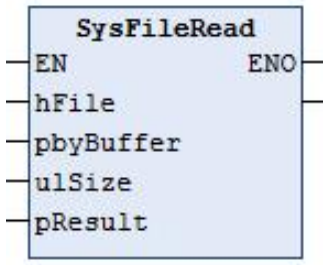
LD：



3.12.5 SysFileRead

读取文件的数据内容

- **指令格式**

指令	名称	FB/FC	LD表现	ST表现	库
SysFileRead	读取文件数据内容	FC		SysFileRead(hFile,pbyBuffer,ulSize,pResult);	SysFile

- **相关变量**

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
hFile	文件句柄	RTS_IEC_HANDLE	-	-	文件句柄
pbyBuffer	读取到的	POINTER TO	–	–	要读取的文件内容指针

	文件内容指针	STRING			
ulSize	要读取文件的字节数	__XWORD	-	-	要读取内容的字节数
pResult	读取文件内容错误码指针	POINTER TO RTS_IEC_RESULT	-	-	读取文件内容错误码指针

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
RelSentSize	返回值	__XWORD	-	-	实际读取文件数据大小，单位字节

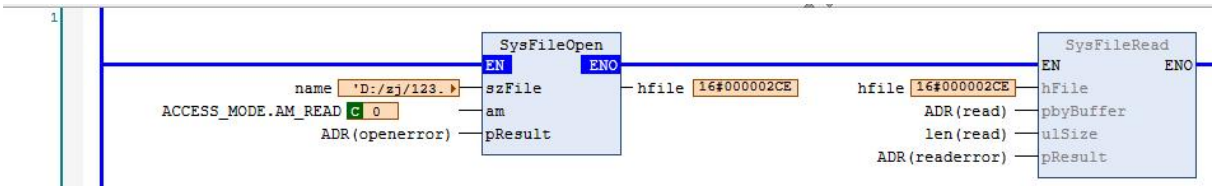
	布尔	位串				整数								实数		时刻、持续时间 日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
hFile	RTS_IEC_HANDLE																			
pbyBuffer	POINTER TO STRING																			
ulSize	-	-	√	√	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
pResult	POINTER TO RTS_IEC_RESULT																			
RelSentSize	POINTER TO RTS_IEC_RESULT																			

- **程序示例**

ST：

```
SysFileOpen(szFile:=name, am:=SysFile.AM_READ, pResult:=ADR(openerror));
hfile:=SysFileOpen(szFile:=name, am:=SysFile.AM_READ, pResult:=ADR(openerror));
SysFileRead(hFile:=hfile, pbyBuffer:=ADR(read), ulSize:=LEN(read), pResult:=ADR(readerror));
```

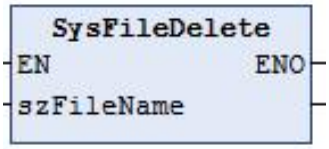
LD：



3.12.6 SysFileDelete

删除文件操作

- **指令格式**

指令	名称	FB/FC	LD表现	ST表现	库
SysFileDelete	删除文件指令	FC		SysFileDelete(szFileName);	SysFile

- **相关变量**

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
szFileName	指定文件名	STRING	-	-	指定文件路径及文件名

输出变量

输入变量	名称	数据类型	有效范围	初始值	描述
FileDeleteErrorID	返回值	RTS_IEC_RESULT	-	-	删除文件错误码

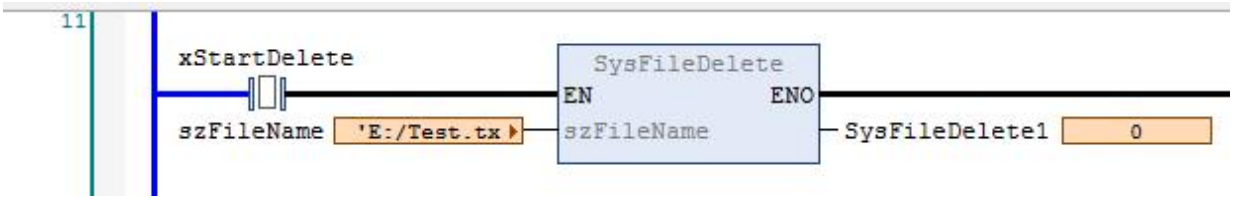
	布尔	位串				整数								实数		时刻、持续时间 日期、字符串					
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
szFileName	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√
FileDeleteErrorID	POINTER TO RTS_IEC_RESULT																			

- **程序示例**

ST：

```
SysFileDelete(szFileName:= 'E:/Test.txt');
```

LD：



3.12.7 SysFileCopy

将一个文件复制到另一个文件

- **指令格式**

指令	名称	FB/FC	LD表现	ST表现	库
SysFileCopy	复制文件指令	FC	sysfilecopy EN ENO szDestFileName szSourceFileName pulCopied	SysFileCopy(szDestFileName,szsourceFileName,pulCopied);	SysFile

- **相关变量**

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
szDestFileName	目标文件	STRING	-	-	目标文件文件路径及文件名
szSourceFileName	源文件	STRING	-	-	源文件路径及文件名
pulCopied	实际读取数据大小指针	POINTER TO _XWORD	-	-	实际读取数据大小指针；单位：字节

输出变量

输入变量	名称	数据类型	有效范围	初始值	描述
FileCopyErrorID	返回值	RTS_IEC_RESULT	-	-	复制文件错误码

	布尔	位串	整数	实数	时刻、持续时间 日期、字符串

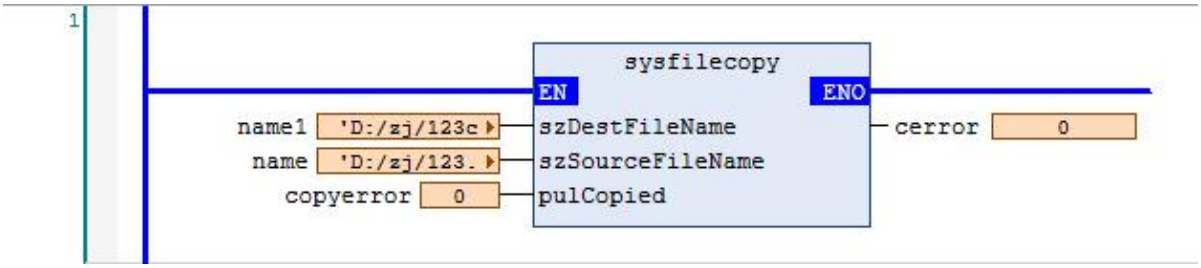
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
szDestFileName	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√
szSourceFileName	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√
pulCopied	POINTER TO _XWORD																			

- **程序示例**

ST：

```
1  SysFileCopy(szDestFileName:=name1, szSourceFileName:=name, pulCopied:=copyerror);
2  RETURN
```

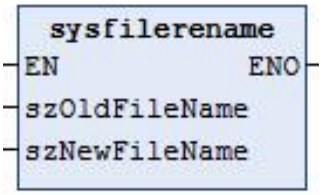
LD：



3.12.8 SysFileRename

重命名文件操作

- **指令格式**

指令	名称	FB/FC	LD表现	ST表现	库
SysFileRename	重命名文件	FC		SysFileCopy(szOldFileName,szNewFileName);	SysFile

- **相关变量**

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
szOldFileN	源文件	STRING	-	-	旧文件名

ame					
szNewFileName	目标文件	STRING	-	-	新文件名

输出变量

输入变量	名称	数据类型	有效范围	初始值	描述
sysRename	返回值	RTS_IEC_RESULT	-	-	重命名文件错误码

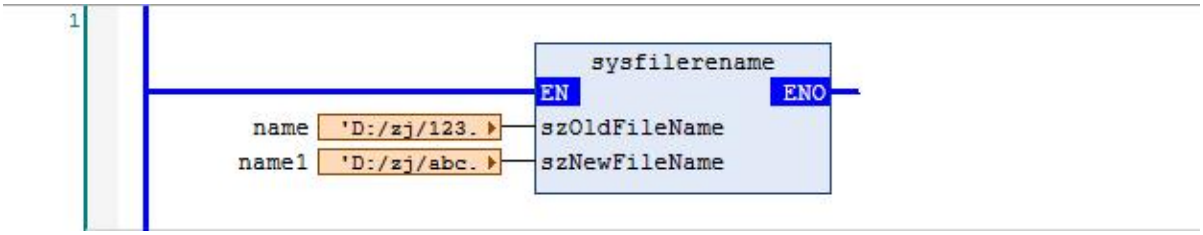
	布尔	位串				整数								实数		时刻、持续时间 日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
szOldFileName	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√
szNewFileName	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√

- **程序示例**

ST：

```
1  SysFileRename(szOldFileName:=name, szNewFileName:=name1);
```

LD：

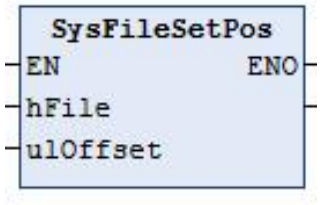


3.12.9 SysFileSetPos

设置文件指针到指定位置

- **指令格式**

指令	名称	FB/FC	LD表现	ST表现	库

SysFileSetPos	设置文件指针到指定位置	FC		SysFileSetpos(hFile,ulOffset);	SysFile

- **相关变量**

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
hFile	文件句柄	RTS_IEC_HANDLE	-	-	文件句柄
uloffset	偏移位置	WORD	-	-	文件指针偏移量

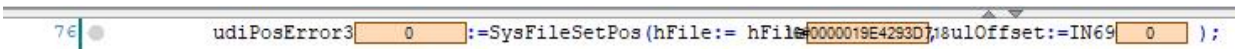
输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
FiileSetPosErrorID	返回值	RTS_IEC_RESULT	-	-	设置文件指针到指定位置错误码

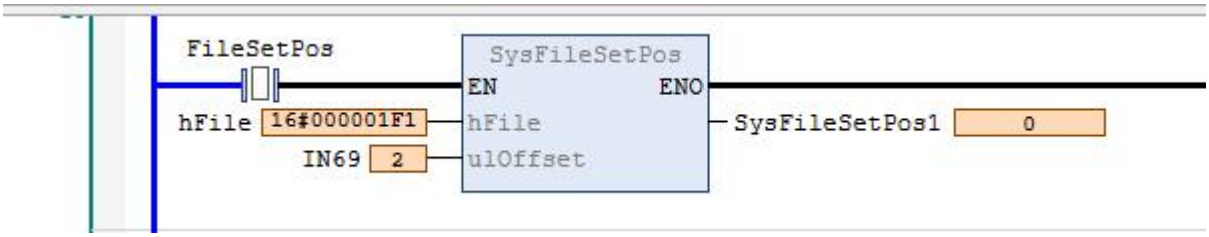
	布尔	位串				整数								实数		时刻、持续时间 日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
hFile	RTS_IEC_HANDLE																			
uloffset	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
FiileSetPosErrorID	RTS_IEC_RESULT																			

- **程序示例**

ST：



LD：



3.12.10 SysFileGetPos

获取文件指针的位置指令

- **指令格式**

指令	名称	FB/FC	LD表现	ST表现	库
SysFileGetPos	获取文件指针的位置	FC	SysFileGetPos EN ENO hFile pulPos	SysFileGetpos(hFile,pulPos);	SysFile

- **相关变量**

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
hFile	文件句柄	RTS_IEC_HANDLE	-	-	文件句柄
pulPos	偏移位置	POINTER TO WORD	-	-	获取文件指针偏移量

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
FiileGetPosErrorID	返回值	RTS_IEC_RESULT	-	-	获取文件指针位置错误码

	布尔	位串				整数								实数		时刻、持续时间 日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING

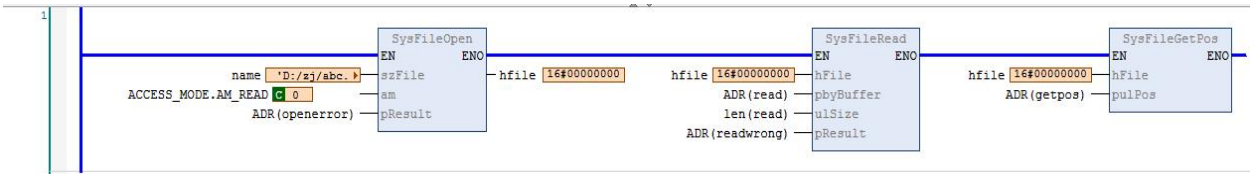
hFile	RTS_IEC_HANDLE																			
uloffset	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
FiileGetPosErrorID	RTS_IEC_RESULT																			

- **程序示例**

ST：

```
SysFileOpen(szFile:=name , am:=SysFile.AM_READ , pResult:=ADR(openerror));
hfile:=SysFileOpen(szFile:=name , am:=SysFile.AM_READ , pResult:=ADR(openerror));
SysFileRead(hFile:=hfile, pbyBuffer:=ADR(read), ulSize:=len(read), pResult:=ADR(readwrong));
SysFileGetPos(hFile:=hfile, pulPos:=ADR(getpos));RETURN
```

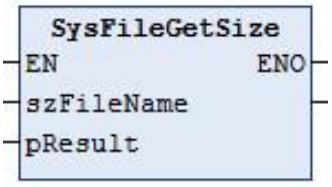
LD：



3.12.11 SysFileGetSize

获取文件大小指令

- **指令格式**

指令	名称	FB/FC	LD表现	ST表现	库
SysFileGetSize	获取文件大小的指令	FC		SysFileGetSize(szFileName,pResult);	SysFile

- **相关变量**

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
szFileName	文件名称	STRING	–	–	文件名
pResult	偏移位置	POINTER TO RTS_IEC_RESULT	–	–	获取文件大小指令错误代码的指针

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
SIZE	返回值	_XWORD	-	-	获取文件指针到指定位置错误码

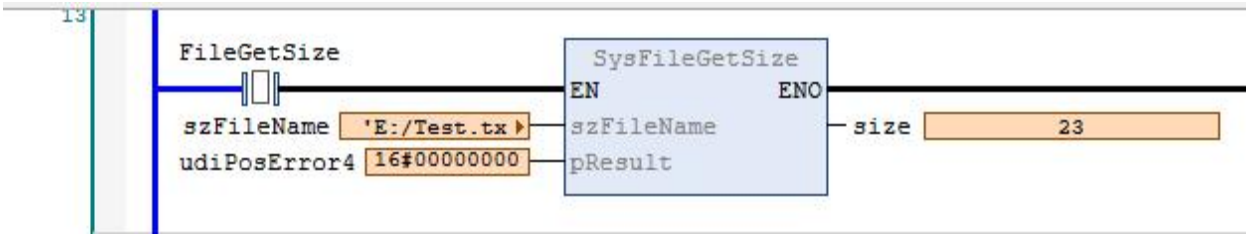
	布尔	位串				整数								实数		时刻、持续时间 日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
szFileName	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√
pResult	POINTER TO RTS_IEC_RESULT																			
SIZE			√																	

- **程序示例**

ST：

```
size := SysFileGetSize(szFileName:='E:/Test.txt' , pResult:=udiPosError4);
```

LD：



3.13 模拟量计算

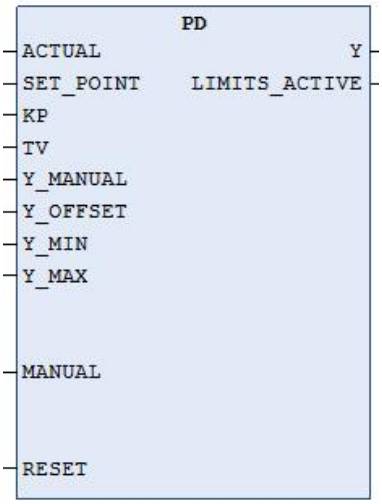
3.13.1 指令列表

指令类别	名称	FB/FC	功能
模拟量计算	PD	FB	比例微分控制
	PID	FB	比例积分微分控制指令
	PID_FIXCYCLE	FB	可手动设置循环周期的比例积分微分控制

3.13.2 PD

比例微分控制器

- **指令格式**

指令	名称	FB/FC	LD表现	ST表现	库
PD	比例微分控制指令	FB		PD (ACTUAL:=, SET_POINT:=, KP:=, TV:=, Y_MANUAL:=, Y_OFFSET:=, Y_MIN:= , Y_MAX:=, MANUAL:=, RESET:= , Y=>, LIMITS_ACTIVE=>)	Util

- **相关变量**

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
ACTUAL	控制变量当前值	REAL	-	0	控制变量当前值
SET_POINT	设置值	REAL	-	0	设置值
KP	比例系数	REAL	-	0	比例系数

TV	微分时间	REAL	-	0	微分时间
Y_MANUAL	手动输出值	REAL	-	0	手动输出值
Y_OFFSET	输出Y的偏移量	REAL	-	0	输出Y的偏移量
Y_MIN	输出Y的最小值	REAL	-	0	输出Y的最小值
Y_MAX	输出Y的最大值	REAL	-	0	输出Y的最大值
MANUAL	手动输出	BOOL	FALSE - TRUE	FALSE	手动输出
RESET	复位	BOOL	FALSE - TRUE	FALSE	复位

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
Y	输出值	REAL	-	0	输出值
LIMITS_ACTIVE	到达给定限制值	BOOL	FALSE-TRUE	FALSE	到达给定限制值

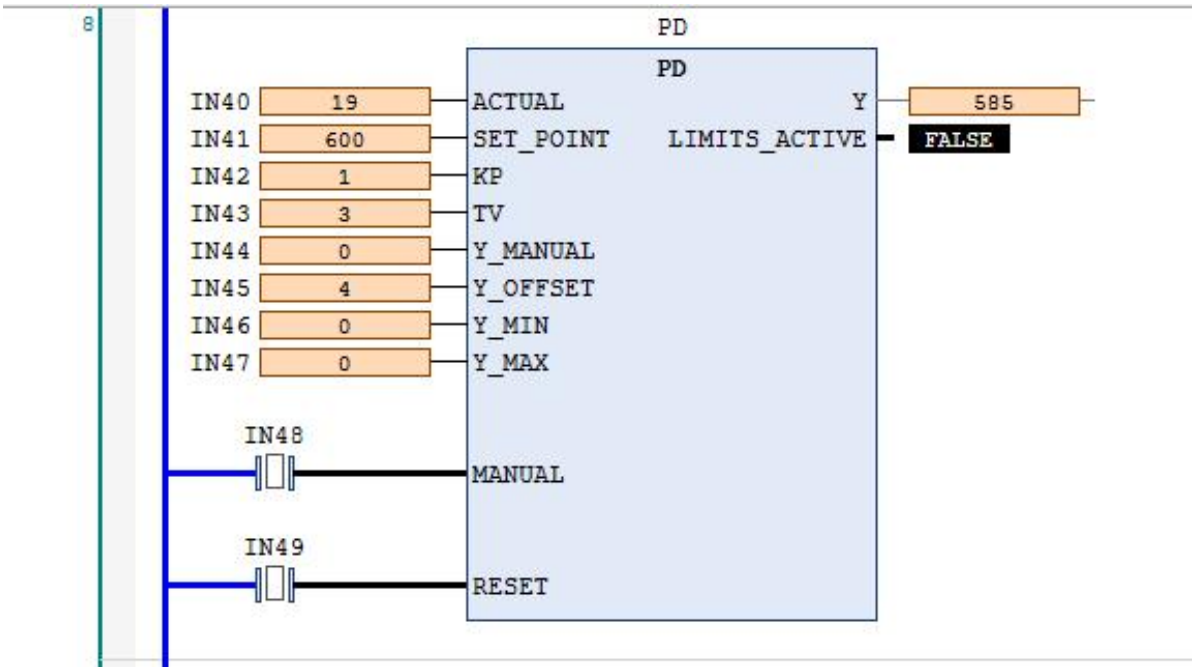
	布尔	位串				整数								实数		时刻、持续时间 日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
ACTUAL	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-
SET_POINT	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-
KP	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-
TV	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-
Y_MANUAL	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-
Y_OFFSET	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-
Y_MIN	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-
Y_MAX	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-
MANUAL	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
RESET	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Y	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-
LIMITS_ACTIVE	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

- **程序示例**

ST：

```
//PD
PD(ACTUAL 19 :=IN40 19 , SET_POINT 600 :=IN41 600 , KP 1 := IN42 1 , TV 3 := IN43 3 ,
  Y_MANUAL 0 := IN44 0 , Y_OFFSET 4 := IN45 4 , Y_MIN 0 :=IN46 0 , Y_MAX 0 := IN47 0 ,
MANUAL FALSE :=IN48 FALSE , RESET FALSE :=IN49 FALSE , Y 585 =>Y1 585 , LIMITS_ACTIVE=> );
```

LD：

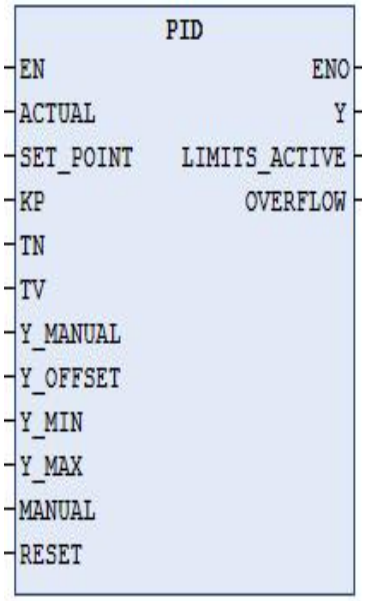


3.13.3 PID

比例积分微分控制器

- **指令格式**

指令	名称	FB/FC	LD表现	ST表现	库

PID	比例积分微分控制指令	FB		PD (ACTUAL:=, SET_POINT:=, KP:=, TN:=, TV:=, Y_MANUAL:=, Y_OFFSET:=, Y_MIN:= , Y_MAX:=, MANUAL:=, RESET:= , Y=>, LIMITS_ACTIVE=> OVERFLOW)	Util

- **相关变量**

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
ACTUAL	控制变量当前值	REAL	-	0	当前值，过程变量
SET_POINT	设置值	REAL	-	0	所需值，设定值
KP	比例系数	REAL	-	0	比例系数
TN	积分时间	REAL	-	0	重置时间，I形部件的相互单位增益。
TV	微分时间	REAL	-	0	微分作用时间，D部分的单位增益
Y_MANUAL	手动输出值	REAL	-	0	定义手动
Y_OFFSET	输出Y的偏移量	REAL	-	0	操纵变量Y的偏移量
Y_MIN	输出Y的最小值	REAL	-	0	操纵变量的最小值
Y_MAX	输出Y的最大值	REAL	-	0	操纵变量的最大值
MANUAL	手动输出	BOOL	FALSE - TRUE	FALSE	激活手动操作
RESET	复位	BOOL	FALSE - TRUE	FALSE	重置控制器

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
Y	输出值	REAL	-	0	输出值
LIMITS_ACTIVE	到达给定限制值	BOOL	FALSE-TRUE	FALSE	到达给定限制值
OVERFLOW	溢出标志	BOOL	FALSE-TRUE	FALSE	溢出标志

	布尔	位串				整数								实数		时刻、持续时间 日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
ACTUAL	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-
SET_POINT	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-
KP	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-
TN	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-
TV	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-
Y_MANUAL	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-
Y_OFFSET	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-
Y_MIN	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-
Y_MAX	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-
MANUAL	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
RESET	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Y	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-
LIMITS_ACTIVE	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
OVERFLOW	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

- **程序示例**

ST：

```
PID_0(
    ACTUAL:= input1 ,
    SET_POINT:=spoint ,
    KP:=kp ,
    TN:=tn ,
    TV:=tv ,
    Y_MANUAL:=ymanual,
    Y_OFFSET:=yoffest,
    Y_MIN:=ymin,
    Y_MAX:=ymax,
    MANUAL:=man,
    RESET:=reset,
    Y=> ,
    LIMITS_ACTIVE=> ,
    OVERFLOW=> );RETURN
```

LD：



3.13.4 PID_FIXCYCLE

比例积分微分控制器

- **指令格式**

指令	名称	FB/FC	LD表现	ST表现	库
PID_FIXCYCLE	比例积分微分控制指令	FB	PID_FIXCYCLE PID_FIXCYCLE ACTUAL Y SET_POINT LIMITS_ACTIVE KP OVERFLOW TN TV Y_MANUAL Y_OFFSET Y_MIN Y_MAX MANUAL RESET CYCLE	PD (ACTUAL:=, SET_POINT:=, KP:=, TN:=, TV:=, Y_MANUAL:=, Y_OFFSET:=, Y_MIN:= , Y_MAX:=, MANUAL:=, RESET:= , CYCLE:=, Y=>, LIMITS_ACTIVE=> OVERFLOW=>)	Util

- **相关变量**

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
ACTUAL	控制变量当前值	REAL	-	0	当前值，过程变量

SET_POINT	设置值	REAL	-	0	所需值，设定值
KP	比例系数	REAL	-	0	比例系数
TN	积分时间	REAL	-	0	重置时间，I形部件的相互单位增益。
TV	微分时间	REAL	-	0	微分作用时间，D部分的单位增益
Y_MANUAL	手动输出值	REAL	-	0	定义手动
Y_OFFSET	输出Y的偏移量	REAL	-	0	操纵变量Y的偏移量
Y_MIN	输出Y的最小值	REAL	-	0	操纵变量的最小值
Y_MAX	输出Y的最大值	REAL	-	0	操纵变量的最大值
MANUAL	手动输出	BOOL	FALSE - TRUE	FALSE	激活手动操作
RESET	复位	BOOL	FALSE - TRUE	FALSE	重置控制器
CYCLE	采样周期时间	REAL	-	0	采样周期时间

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
Y	输出值	REAL	-	0	输出值
LIMITS_ACTIVE	到达给定限制值	BOOL	FALSE-TRUE	FALSE	到达给定限制值
OVERFLOW	溢出标志	BOOL	FALSE-TRUE	FALSE	溢出标志

	布尔	位串				整数								实数		时刻、持续时间 日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
ACTUAL	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-
SET_POINT	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-
KP	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-
TN	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-
TV	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-
Y_MANUAL	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-

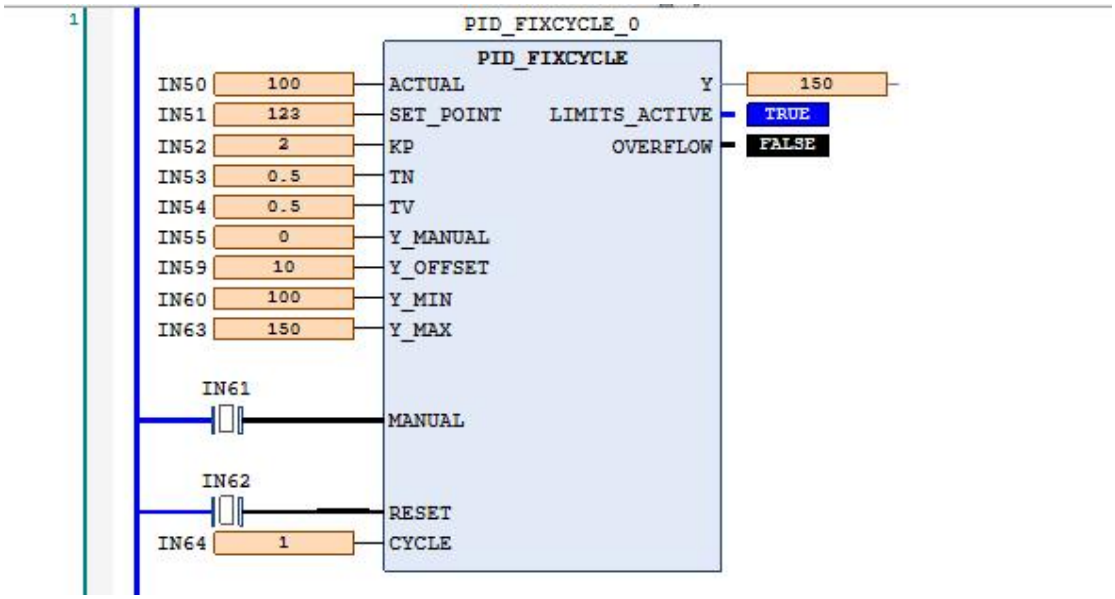
Y_OFFSET	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-
Y_MIN	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-
Y_MAX	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-
MANUAL	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
RESET	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
CYCLE	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-
Y	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-
LIMITS_ACTIVE	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
OVERFLOW	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

- **程序示例**

ST：

```
14  //PID_FIXCYCLE
15  PID_FIXCYCLE(ACTUAL 19 := IN50 19 , SET_POINT 21 :=IN51 21 , KP 1 :=IN52 1 ,
16   TN 10 :=IN53 10 , TV 0 := IN54 0 , Y_MANUAL 0 := IN55 0 , Y_OFFSET 0 :=IN59 0 ,
17   Y_MIN 0 := IN60 0 , Y_MAX 0 := IN61 0 , MANUAL FALSE :=IN62 FALSE , RESET FALSE :=IN63 FALSE ,
18  CYCLE 0.05 :=IN64 0.05 , Y 49.2 => Y2 49.2 , LIMITS_ACTIVE=> , OVERFLOW=> );
```

LD：



3.14 BCD码转换

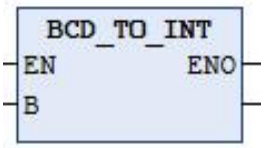
3.14.1 指令列表

指令类别	名称	FB/FC	功能
BCD 码转换指令	BCD_TO_INT	FC	BCD 码转整型指令
	INT_TO_BCD	FC	整型转 BCD 码指令
	BCD_TO_BYTE	FC	BCD 码转字节指令
	BYTE_TO_BCD	FC	字节转 BCD 码指令
	BCD_TO_WORD	FC	BCD 码转字指令
	WORD_TO_BCD	FC	字转 BCD 指令
	BCD_TO_DWORD	FC	BCD 码转双字指令
	DWORD_TO_BCD	FC	双字转 BCD 码指令

3.14.2 BCD_TO_INT

将源数据的 BCD 码转换成 INT 类型的数据

- **指令格式**

指令	名称	FB/FC	LD表现	ST表现	库
BCD_TO_INT	BCD 码转整型	FC		Output： =BCD_TO_INT(B)	Util

- **相关变量**

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
B	输入数据	BYTE	0 – 255	0	输入数据

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
output	输出数据	INT	-32768-32768	0	输出数据

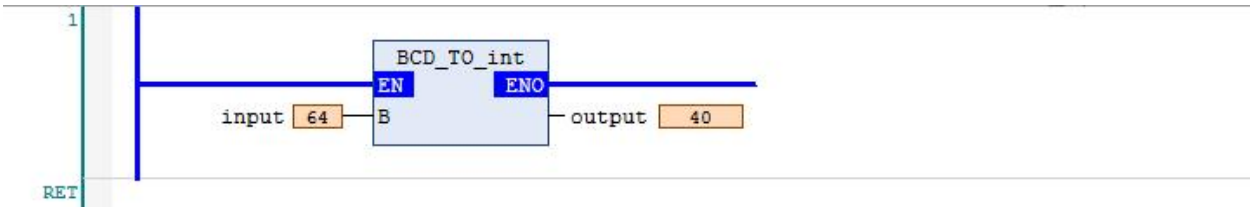
	布尔	位串				整数								实数		时刻、持续时间 日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
B	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-
output	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

- **程序示例**

ST：

```
1  output:=BCD_TO_INT(input);RETURN
```

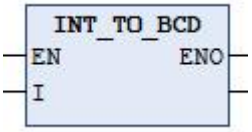
LD：



3.14.3 INT_TO_BCD

将源数据的 INT 型数据转换成 BCD 码

- **指令格式**

指令	名称	FB/FC	LD表现	ST表现	库
INT_TO_BCD	整型转BCD码	FC		Output：=INT_TO_BCD(I)	Util

- **相关变量**

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
I	输入数据	INT	0 – 99	0	输入数据

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
output	输出数据	BYTE	0 - 255	0	输出数据

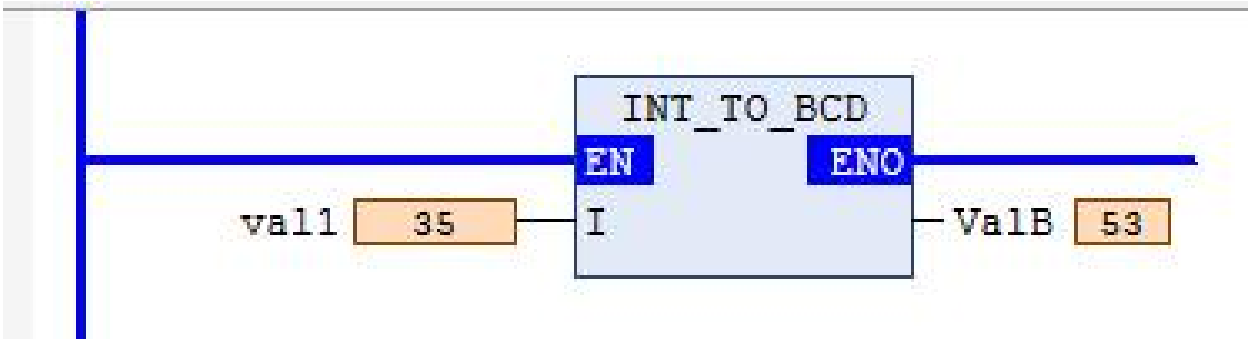
	布尔	位串				整数								实数		时刻、持续时间 日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
I	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	√
output	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

- 程序示例

ST：

```
1  OUTPUT 53 :=INT_TO_BCD(I 35 );RETURN
```

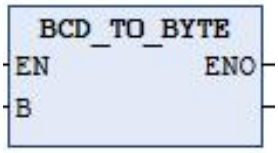
LD：



3.14.4 BCD_TO_BYTE

将源数据的 BCD 码转换成 BYTE 类型的数据

- **指令格式**

指令	名称	FB/FC	LD表现	ST表现	库
BCD_TO_BYTE	BCD 码转 BYTE 型	FC		Output：=BCD_TO_BYTE(B)	Util

- **相关变量**

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
B	输入数据	BYTE	0 – 255	0	输入数据

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
output	输出数据	BYTE	0 – 255	0	输出数据

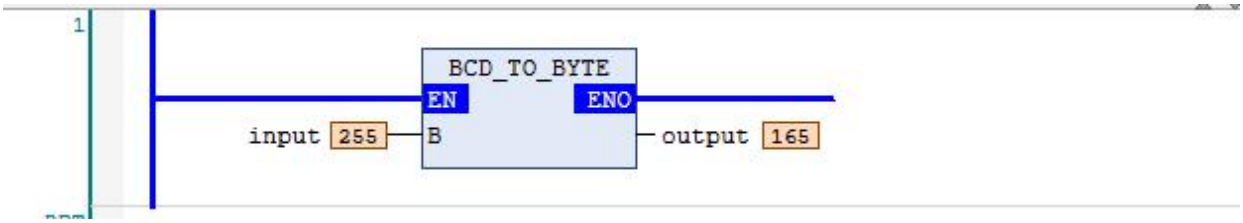
	布尔	位串				整数								实数		时刻、持续时间 日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
B	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
output	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

- **程序示例**

ST：

```
1  output:=BCD_TO_BYTE(input);RETURN
```

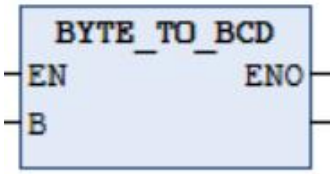
LD：



3.14.5 BYTE_TO_BCD

将源数据的 BYTE 型数据转换成 BCD 码

- **指令格式**

指令	名称	FB/FC	LD表现	ST表现	库
BYTE_TO_BCD	BYTE型转BCD码	FC		Output：=BYTE_TO_BCD(B)	Util

- **相关变量**

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
B	输入数据	BYTE	0 – 255	0	输入数据

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
output	输出数据	BYTE	0 - 255	0	输出数据

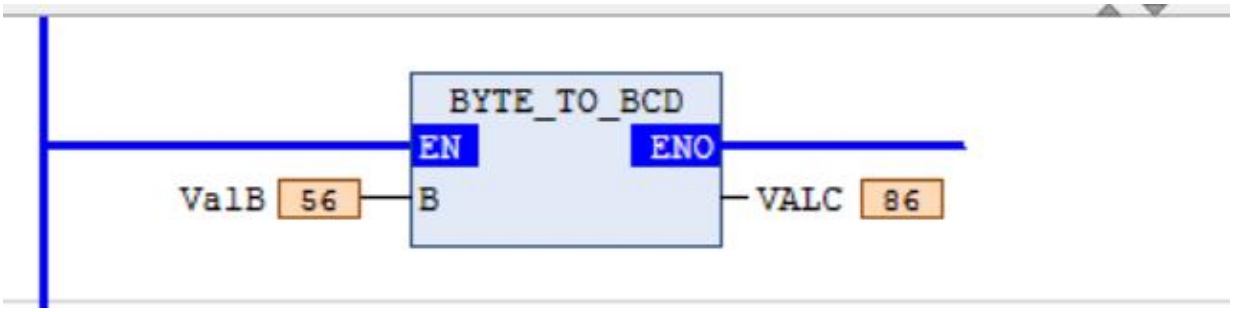
	布尔	位串				整数								实数		时刻、持续时间 日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
B	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
output	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

- 程序示例

ST：

```
OUTPUT 86 :=BYTE_TO_BCD(B 56 );RETURN
```

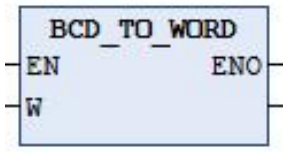
LD：



3.14.6 BCD_TO_WORD

将源数据的 BCD 码转换成 WORD 类型的数据

- 指令格式

指令	名称	FB/FC	LD表现	ST表现	库
BCD_TO_WORD	BCD 码转 WORD 型	FC		Output：=BCD_TO_WORD(W)	Util

- 相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
B	输入数据	WORD	0 – FFFF	0	输入数据

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
output	输出数据	WORD	0 – FFFF	0	输出数据

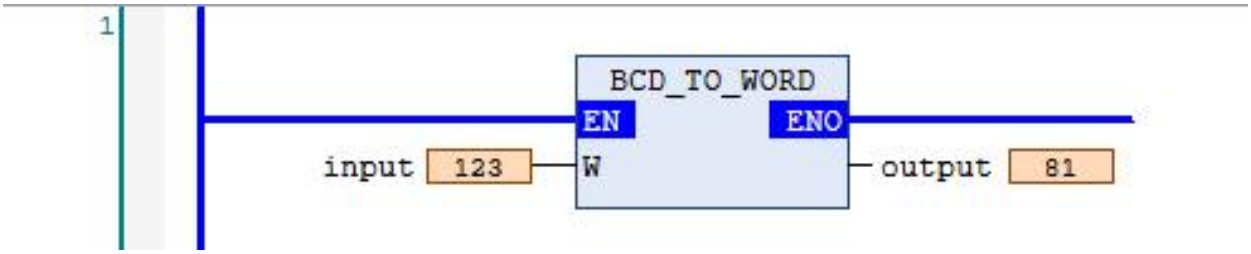
	布尔	位串				整数								实数		时刻、持续时间 日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
B	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
output	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

- **程序示例**

ST：

```
1  output:=BCD_TO_WORD(input);
2  RETURN
```

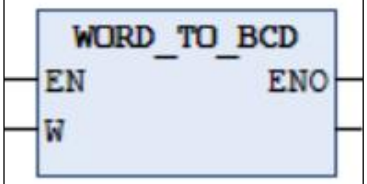
LD：



3.14.7 WORD_TO_BCD

将源数据的 WORD 型数据转换成 BCD 码

- **指令格式**

指令	名称	FB/FC	LD表现	ST表现	库
WORD_TO_BCD	WORD 型 转BCD 码	FC		Output：= WORD_TO_BCD(W)	Util

- 相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
W	输入数据	WORD	0 – FFFF	0	输入数据

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
output	输出数据	WORD	0 - FFFF	0	输出数据

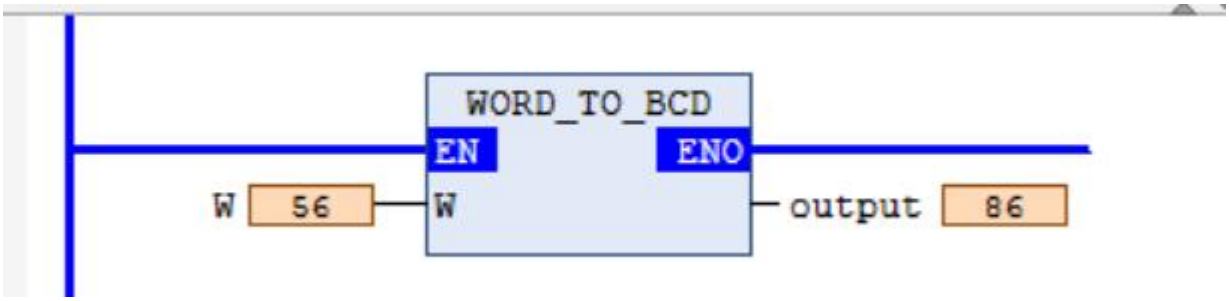
	布尔	位串				整数								实数		时刻、持续时间 日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
W	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
output	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

- 程序示例

ST：

```
wOUTPUT 86 :=WORD_TO_BCD(W 56);RETURN
```

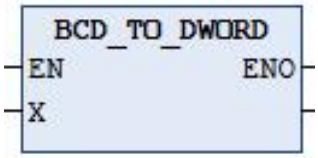
LD：



3.14.8 BCD_TO_DWORD

将源数据的 BCD 码转换成 DWORD 类型的数据

- 指令格式

指令	名称	FB/FC	LD表现	ST表现	库
BCD_TO_DWORD	BCD 码转 DWORD 型	FC		Output：= BCD_TO_DWORD(W)	Util

- 相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
X	输入数据	DWORD	0 – FFFF FFFF	0	输入数据

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
output	输出数据	DWORD	0 – FFFF FFFF	0	输出数据

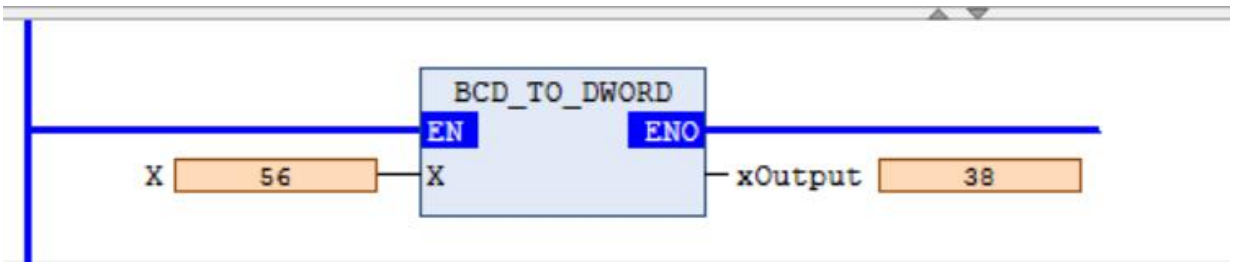
	布尔	位串				整数								实数		时刻、持续时间 日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
B	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
output	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

- 程序示例

ST：

```
xOUTPUT 38 :=BCD_TO_DWORD(X 56);
```

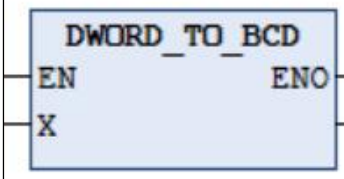
LD：



3.14.9 DWORD_TO_BCD

将源数据的 DWORD 型数据转换成 BCD 码

- **指令格式**

指令	名称	FB/FC	LD表现	ST表现	库
DWORD_TO_BCD	DWORD型转BCD码	FC		Output：= DWORD_TO_BCD(X)	Util

- **相关变量**

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
X	输入数据	DWORD	0 – FFFF FFFF	0	输入数据

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
output	输出数据	DWORD	0 – FFFF FFFF	0	输出数据

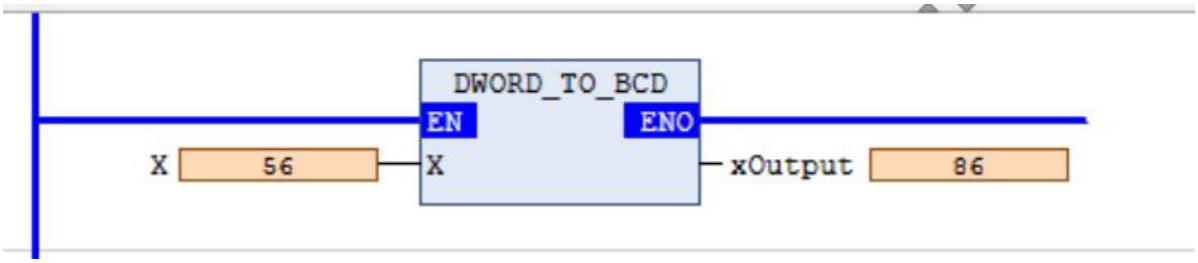
	布尔	位串				整数								实数		时刻、持续时间 日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
W	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
output	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

- **程序示例**

ST：

```
xOUTPUT 86 :=DWORD_TO_BCD(X 56 );
```

LD：


DWORD_TO_BCD
EN
ENO
X
56
X
xOutput
86

3.15 模拟波形

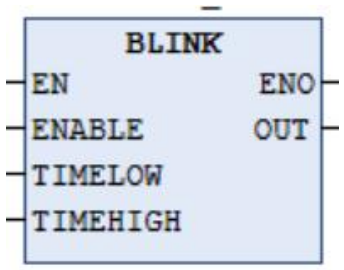
3.15.1 指令列表

指令类别	名称	FB/FC	功能
模拟波形	BLINK	FB	脉冲信号发生器
	GEN	FB	周期性信号发生器
	FREQ_MEASURE	FB	频率测量指令

3.15.2 BLINK

脉冲信号发生器，在时间周期 TIMEHIGH 内将输出设置为 TRUE，随后在 TIMELOW 周期内将输出设置为 FALSE。

- **指令格式**

指令	名称	FB/FC	LD表现	ST表现	库
BLINK	BCD 码转整型	FB		BLINK_0(ENABLE:=, TIMELOW:=, TIMEHIGH:=, OUT=>);	Util

- **相关变量**

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
ENABLE	使能	BOOL	0 – 255	0	TRUE 时，指令开始工作
TIMELOW	低电平时间	TIME	-	-	脉冲信号保持低电平的时间长短
TIMEHIGH	高电平时间	TIME	-	-	脉冲信号保持高电平的时间长短

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
OUT	脉冲信号输出	BOOL	TRUE-FALSE	FALSE	脉冲信号的输出值

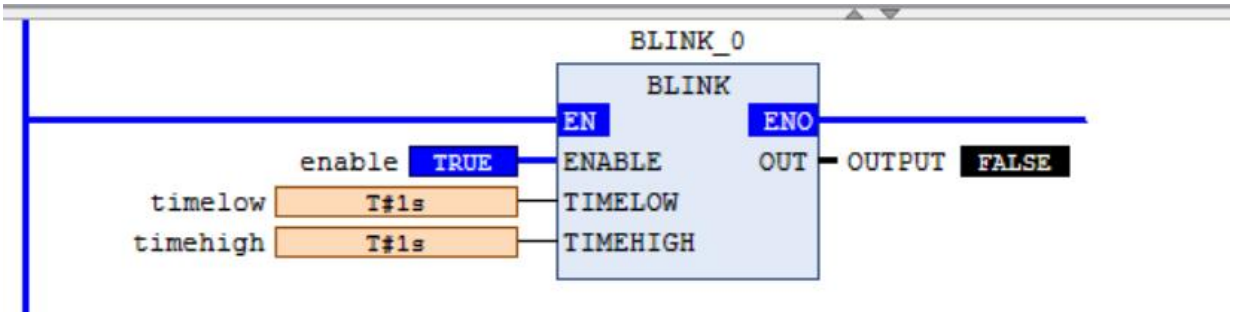
	布尔	位串				整数								实数		时刻、持续时间 日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
ENABLE	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
TIMELOW	-		-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-
TIMEHIGH	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-
OUT	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

- **程序示例**

ST：

```
BLINK_0(ENABLE:=enable,TIMELOW:=timelow,TIMEHIGH:=timehigh,OUT=>OUTPUT);
```

LD：

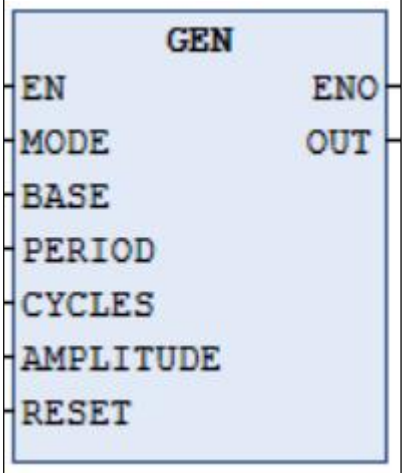


3.15.3 GEN

根据输入参数生成指定的函数波形

- **指令格式**

指令	名称	FB/FC	LD表现	ST表现	库

GEN	周期信号发生器	FB		GEN_0(MODE:=, BASE:=, PERIOD:=, CYCLES:=, AMPLITUDE:=, RESET:=, OUT=>);	Util

- **相关变量**

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
MODE	信号类型	GEN_MODE	-	TRIANGLE	GEN_MODE 类型的枚举变量
BASE	循环方式选择	BOOL	FALSE-TRUE	FALSE	TURE:以PERIOD为循环周期输出信号 FALSE:以CYCLES为循环次数输出信号
PERIOD	循环周期	TIME	-	TIME#1s0ms	BASE为TRUE时有效，输出信号的循环周期
CYCLES	循环次数	INT	0-65535	1000	BASE为FALSE时有效，输出信号的循环次数
AMPLITUDE	信号振幅	INT	0-65535	0	输出信号的振幅值
RESET	复位	BOOL	FALSE-TRUE	FALSE	TRUE时输出信号复位为0，为FALSE时方可正常输出信号

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
OUT	信号输出值	INT	0-65535	0	周期信号的输出值

	布	位串	整数	实数	时刻、持续时间

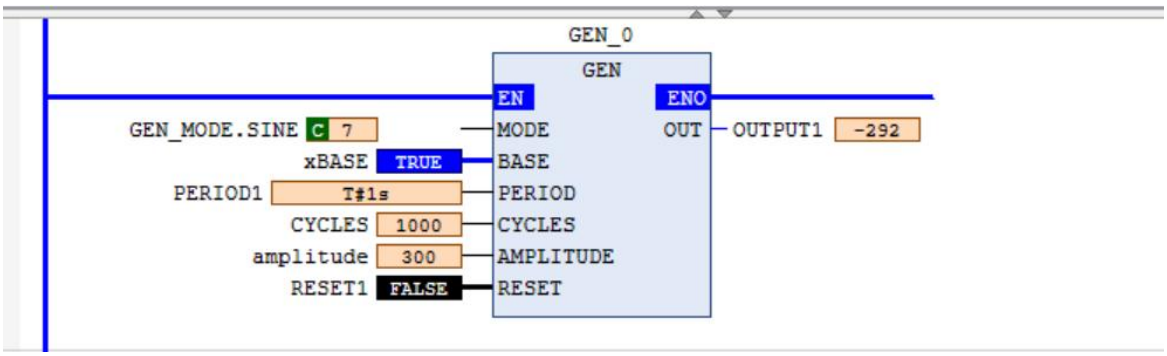
	尔															日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
MODE	GEN_MODE																			
BASE	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
PERIOD	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-
CYCLES	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-
AMPLITUDE	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-
RESET	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
OUT	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-

- **程序示例**

ST：

```
GEN_0(
MODE SINE :=GEN_MODE.SINE,
BASE FALSE :=BASE FALSE,
PERIOD T#1s :=PERIOD1 T#1s,
CYCLES 1000 :=CYCLES 1000,
AMPLITUDE 300 :=amplitude 300,
RESET FALSE :=RESET FALSE,
OUT -146 =>OUTPUT1 -146 );RETURN
```

LD：



MODE为GEN_MODE类型的一个枚举变量，有7个枚举变量,如：

0-TRIANGLE：输出三角波函数，幅值：-Amplitude~+Amplitude。

1-TRIANGLE_POS：输出三角波函数，幅值：0~+Amplitude。

2-SAWTOOTH_RISE：上升锯齿波函数，幅值：-Amplitude~+Amplitude。

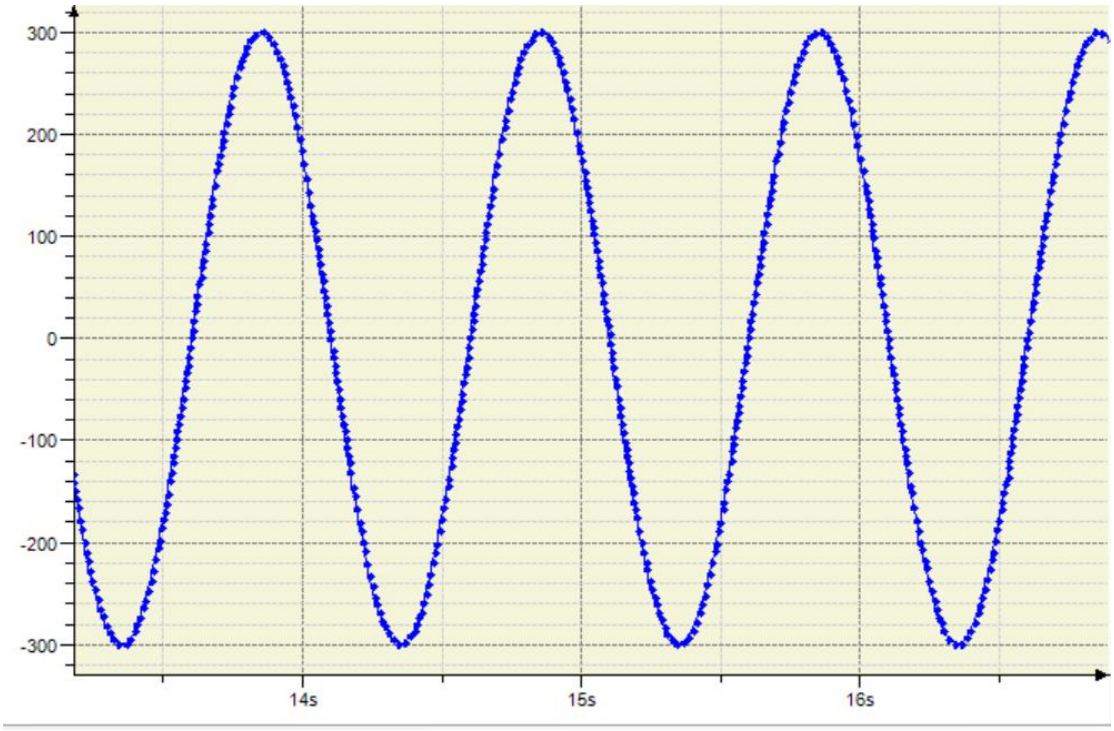
3-SAWTOOTH_FALL：下降锯齿波函数，幅值：-Amplitude~+Amplitude。

4-RECTANGLE：矩形锯齿波函数转换，幅值：-Amplitude~+Amplitude。

5-SINE：正弦函数

6-COSINE：余弦函数

MODE:=SINE



3.15.4 FREQ_MEASURE

测量输入的布尔类型信号的（平均）频率值（Hz）

- **指令格式**

指令	名称	FB/FC	LD表现	ST表现	库

FREQ_MEASURE	频率测量指令	FB	FREQ_MEASURE_0 FREQ_MEASURE EN ENO IN OUT PERIODS VALID RESET	FREQ_MEASURE_0(IN:=, PERIODS:=, RESET:=, OUT=>, VALID=>);	Util

- **相关变量**

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
IN	输入信号	BOOL	TRUE-FALSE	FALSE	IN 置 TURE，指令开始工作
PERIODS	测量周期	INT	0-65535	1	范围 1 到 10，默认值为1输入信号两个上升沿的时间间隔为一个周期，将 N 个周期的频率值求平均值就得到输入信号的频率，此参数即为求平均值运算的周期次数。
RESET	复位	BOOL	TRUE-FALSE	FALSE	为TRUE时重新测量

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
OUT	频率输出	REAL	-2147483648～2147483647	0	输入信号的频率值，单位 Hz
VALID	运算标志	BOOL	TRUE-FALSE	FALSE	第一次运算完成后或运算错误时置TRUE，其余时刻为FALSE

	布尔	位串				整数								实数		时刻、持续时间 日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING

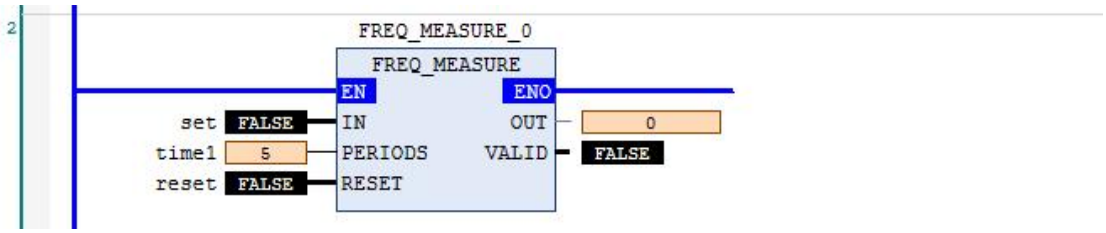
IN	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
PERIODS	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-
RESET	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
OUT	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-
VALID	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

- **程序示例**

ST：

```
1  FREQ_MEASURE_0(
2      IN:=set,
3      PERIODS:=time1,
4      RESET:=reset,
5      OUT=> ,
6      VALID=> );
```

LD：



4 运动指令

4.1 单轴状态监控

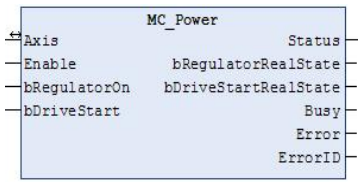
4.1.1 指令列表

指令类别	名称	FB/FC	功能
单轴状态监控	MC_Power	FB	轴使能
	MC_Reset	FB	复位轴
	MC_ReadStatus	FB	读轴状态
	MC_ReadAxisError	FB	读轴错误
	MC_ReadParameter	FB	读参数
	MC_ReadBoolParameter	FB	读布尔参数
	MC_WriteParameter	FB	写参数
	MC_WriteBoolParameter	FB	写布尔参数
	MC_ReadActualPosition	FB	读实际位置
	MC_ReadActualVelocity	FB	读实际速度
	MC_ReadActualTorque	FB	读实际转矩
	MC_SetPosition	FB	设置位置
	SMC_ReadSetPosition	FB	读取设置位置
	SMC_ReadFBError	FB	读取历史错误信息
	SMC_ClearFBError	FB	清除历史错误信息
	SMC_ErrorString	FB	错误码到错误信息

4.1.2 MC_Power

本指令用于使能指定轴，使轴进入可运行状态或退出可运行状态，也叫轴使能。

- **指令格式**

指令	名称	FB/FC	LD表现	ST表现	库
MC_Power	轴使能	FB		MC_Power(Axis:=, Enable:=, bRegulatorOn:=, bDriveStart:=, Status=>, bRegulatorRealState =>, bDriveStartRealState =>, Busy=>, Error=>, ErrorID=>);	SM3_Basic

- **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	初始值	描述
Axis	轴	AXIS_REF_SM3	–	–	指定轴

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Enable	有效	BOOL	TRUE-FALSE	FALSE	必须设置为 TRUE，以激活功能块的处理。
bRegulatorOn	使能	BOOL	TRUE-FALSE	FALSE	必须设置为 TRUE，以激活轴的使能状态。
bDriveStart	驱动启用	BOOL	TRUE-FALSE	FALSE	必须设置为 TRUE，以关闭轴的紧急停止处理。

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
Status	可运行	BOOL	TRUE-FALSE	FALSE	轴准备好则为 TRUE。
bRegulatorRealState	使能有效	BOOL	TRUE-FALSE	FALSE	轴使能的有效状态。

bDriveStartRealState	驱动可使用	BOOL	TRUE-FALSE	FALSE	驱动器没有被快速停止机制中断为 TRUE。
Busy	执行中	BOOL	TRUE-FALSE	FALSE	功能块的处理没有完成为 TRUE。
Error	错误	BOOL	TRUE-FALSE	FALSE	发生异常时为 TRUE
ErrorID	错误代码	SMC_ERROR	–	0	正常时数值为 0，发生异常时，输出报错代码。

	布尔	位串				整数								实数		时刻、持续时间 日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Axis	AXIS_REF_SM3																			
Enable	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
bRegulatorOn	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
bDriveStart	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Status	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
bRegulatorRealState	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
bDriveStartRealState	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Busy	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Error	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
ErrorID	SMC_ERROR																			

- **功能说明：**

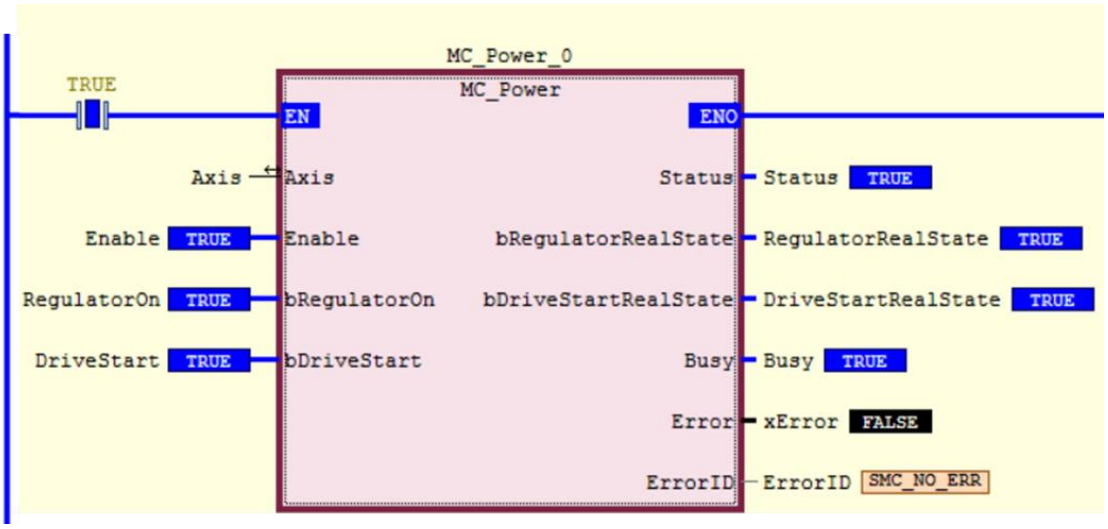
  - 将Enable设为TRUE时，该功能块进入可运行状态；将Enable设为FALSE，功能块不运行，但是可以执行MC_Power指令、MC_Reset（轴错误复位）指令。
  - 将bRegulatorOn设为TRUE，如果轴没有错误则轴状态为standstill。如果轴有错误，轴状态为errorstop。
  - 将bRegulatorOn设为FALSE，轴状态为Disabled，表明轴没有做好运动准备。
  - 将bDriveStart设为FALSE，轴急停。

- **程序示例**

ST：当Enable，RegulatorOn，DriveStart为TRUE时，Status变为TRUE，表示轴Axis已使能。

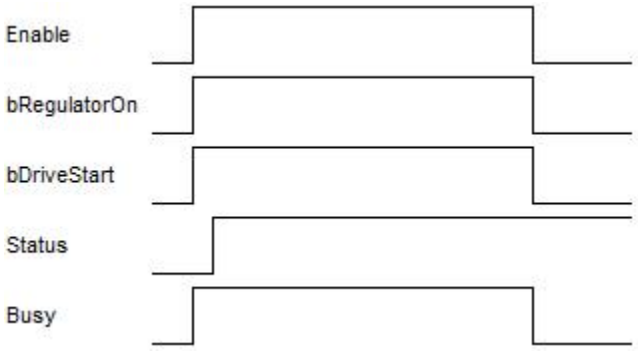
```
1  MC_Power_0(
2      Axis:= Axis,
3      Enable TRUE := Enable TRUE,
4      bRegulatorOn TRUE := RegulatorOn TRUE,
5      bDriveStart TRUE := DriveStart TRUE,
6      Status TRUE => Status TRUE,
7      bRegulatorRealState TRUE => RegulatorRealState TRUE,
8      bDriveStartRealState TRUE => DriveStartRealState TRUE,
9      Busy TRUE => Busy TRUE,
10     Error FALSE => xError FALSE,
11     ErrorID SMC_NO_ERR => ErrorID SMC_NO_ERR ); RETURN
```

LD：当Enable，RegulatorOn，DriveStart为TRUE时，Status变为TRUE，表示轴Axis已使能。



- **时序图：**

将 Enable、bRegulatorOn 和 bDriveStart 分别置为 TRUE，功能块的 Busy 状态会变成 TRUE，然后 Status 信号也变为 TRUE，该轴变成使能状态。如图所示。

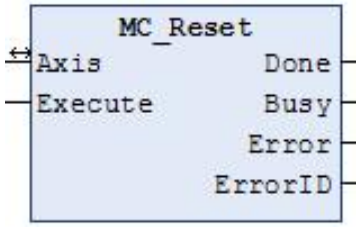


MC_Power 的时序图

4.1.3 MC_Reset

本指令用于复位（清除）轴的错误。

- **指令格式**

指令	名称	FB/FC	LD表现	ST表现	库
MC_Reset	轴复位	FB		Axis:=, Execute:=, Done=>, Busy=>, Error=>, ErrorID=>);	SM3_Basic

- **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	初始值	描述
Axis	轴	AXIS_REF_SM3	-	-	指定轴

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Execute	启动	BOOL	TRUE-FALSE	FALSE	输入值的上升沿将启动该功能块的执行。

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
Done	完成	BOOL	TRUE-FALSE	FALSE	如果复位被执行则为TRUE。
Busy	执行中	BOOL	TRUE-FALSE	FALSE	当功能块执行还没结束时为TRUE。
Error	错误	BOOL	TRUE-FALSE	FALSE	功能块执行错误
ErrorID	错误代码	SMC_ERROR	–	0	错误指示，见SMC_Error。

	布尔	位串				整数								实数		时刻、持续时间 日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Axis	AXIS_REF_SM3																			
Execute	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Done	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Busy	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Error	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
ErrorID	SMC_ERROR																			

- **功能说明：**

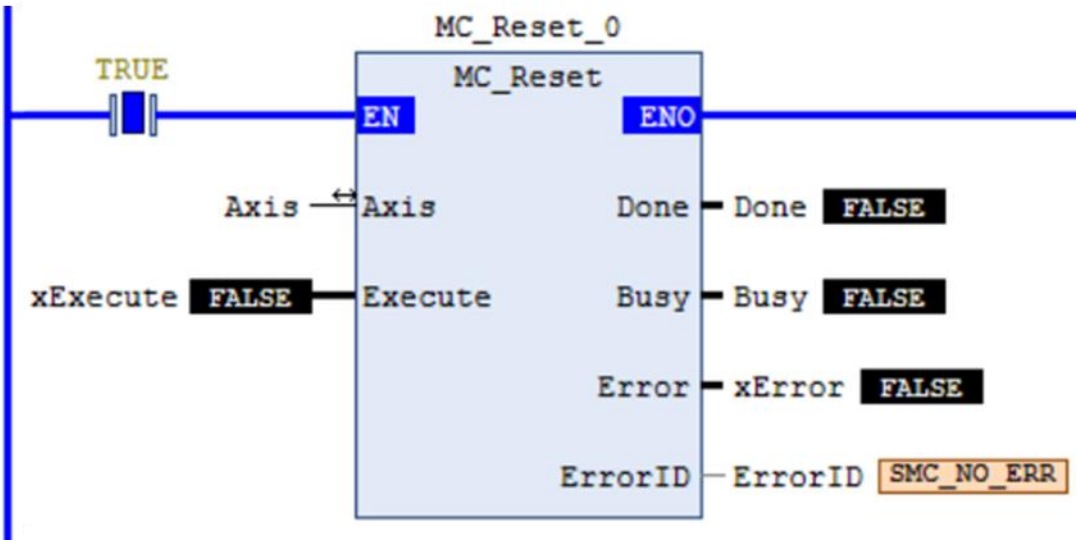
  - 轴在报错后，将不能继续执行运动指令，只有调用MC_Reset指令清除错误之后才能操作轴。MC_Reset便是用于将轴的状态从ErrorStop状态转换成StandStill 状态，消除轴报错状态，变成可执行状态。

- **程序示例**

ST：当xExecute变为TRUE时，Done变为TRUE，表示轴Axis错误已清除。

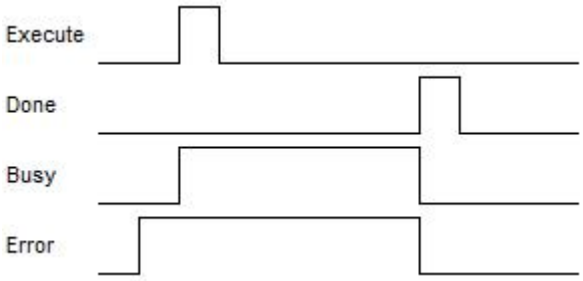
```
MC_Reset_0(
Axis:= Axis ,
Execute FALSE := xExecute FALSE ,
Done FALSE => Done FALSE,
Busy FALSE => Busy FALSE,
Error FALSE => xError FALSE,
ErrorID SMC_NO_ERR => ErrorID SMC_NO_ERR );
```

LD：当xExecute变为TRUE时，Done变为TRUE，表示轴Axis错误已清除。



- **时序图：**

  当 Error 变为 TRUE，将 Excute 置为 TRUE，功能块的 Busy 状态会变成 TRUE，然后 Done 信号也变为 TRUE，该轴清除错误状态。如图所示。



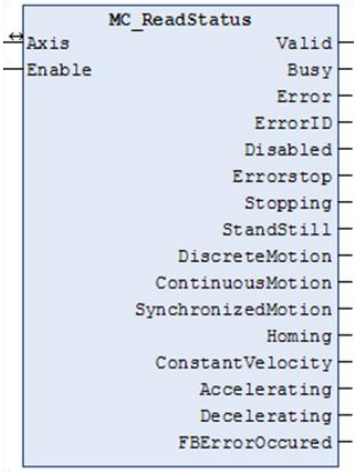
MC_Reset 的时序图

4.1.4 MC_ReadStatus

本指令用于读取轴的详细状态。

- **指令格式**

指令	名称	FB/FC	LD表现	ST表现	库

MC_ReadStatus	读轴状态	FB		MC_ReadStatus(Axis：=， Enable：=， Valid=>， Busy=>， Error=>， ErrorID=>， Disabled=>， Errorstop=>， Stopping=>， StandStill=>， DiscreteMotion=>， ContinuousMotion=>， SynchronizedMotion=>， Homing=>， ConstantVelocity=>， Accelerating=>， Decelerating=>， FBErrorOccured=>);	SM3_Basic

- **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	初始值	描述
Axis	轴	AXIS_REF_SM3	-	-	指定轴

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Enable	有效	BOOL	TRUE-FALSE	FALSE	必须设置为TRUE，以激活功能块的处理。

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
Valid	有效	BOOL	TRUE-FALSE	FALSE	如果轴准备好则为TRUE。
Busy	执行中	BOOL	TRUE-FALSE	FALSE	当功能块执行还没结束时为TRUE。

Error	错误	BOOL	TRUE-FALSE	FALSE	功能块执行错误
ErrorID	错误代码	SMC_ERROR	–	0	错误指示，见SMC_Error。
Disabled	未使能	BOOL	TRUE-FALSE	FALSE	如果轴状态为Disabled 则为TRUE
Errorstop	报错	BOOL	TRUE-FALSE	FALSE	如果轴状态为Errorstop 则为TRUE
Stopping	停止中	BOOL	TRUE-FALSE	FALSE	如果轴状态为Stopping 则为TRUE
StandStill	准备好	BOOL	TRUE-FALSE	FALSE	如果轴状态为Standstill 则为TRUE
DiscreteMotion	点位运动	BOOL	TRUE-FALSE	FALSE	如果轴状态为DiscreteMotion 则为TRUE
ContinuousMotion	连续运动	BOOL	TRUE-FALSE	FALSE	如果轴状态为ContinuousMotion 则为TRUE
SynchronizedMotion	同步运动	BOOL	TRUE-FALSE	FALSE	如果轴状态为 SynchronizedMotion 则为TRUE
Homing	回零中	BOOL	TRUE-FALSE	FALSE	如果轴状态为Homing 则为TRUE
ConstantVelocity	恒速运动	BOOL	TRUE-FALSE	FALSE	如果电机以恒定速度移动则为TRUE
Accelerating	加速中	BOOL	TRUE-FALSE	FALSE	如果电机正在加速则为TRUE
Decelerating	减速中	BOOL	TRUE-FALSE	FALSE	如果电机正在减速则为TRUE
FBErrorOccured	FB报错	BOOL	TRUE-FALSE	FALSE	如果功能块错误被探测到并且还没有由SMC_ClearFBError清除，则为TRUE

	布尔	位串				整数								实数		时刻、持续时间 日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Axis	AXIS_REF_SM3																			
Enable	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Valid	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Busy	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

Error	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
ErrorID	SMC_ERROR																		
Disabled	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Errorstop	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Stopping	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
StandStill	SMC_ERROR																		
DiscreteMotion	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
ContinuousMotion	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
SynchronizedMotion	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Homing	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
ConstantVelocity	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Accelerating	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Decelerating	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
FBErrorOccured	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

- **功能说明：**

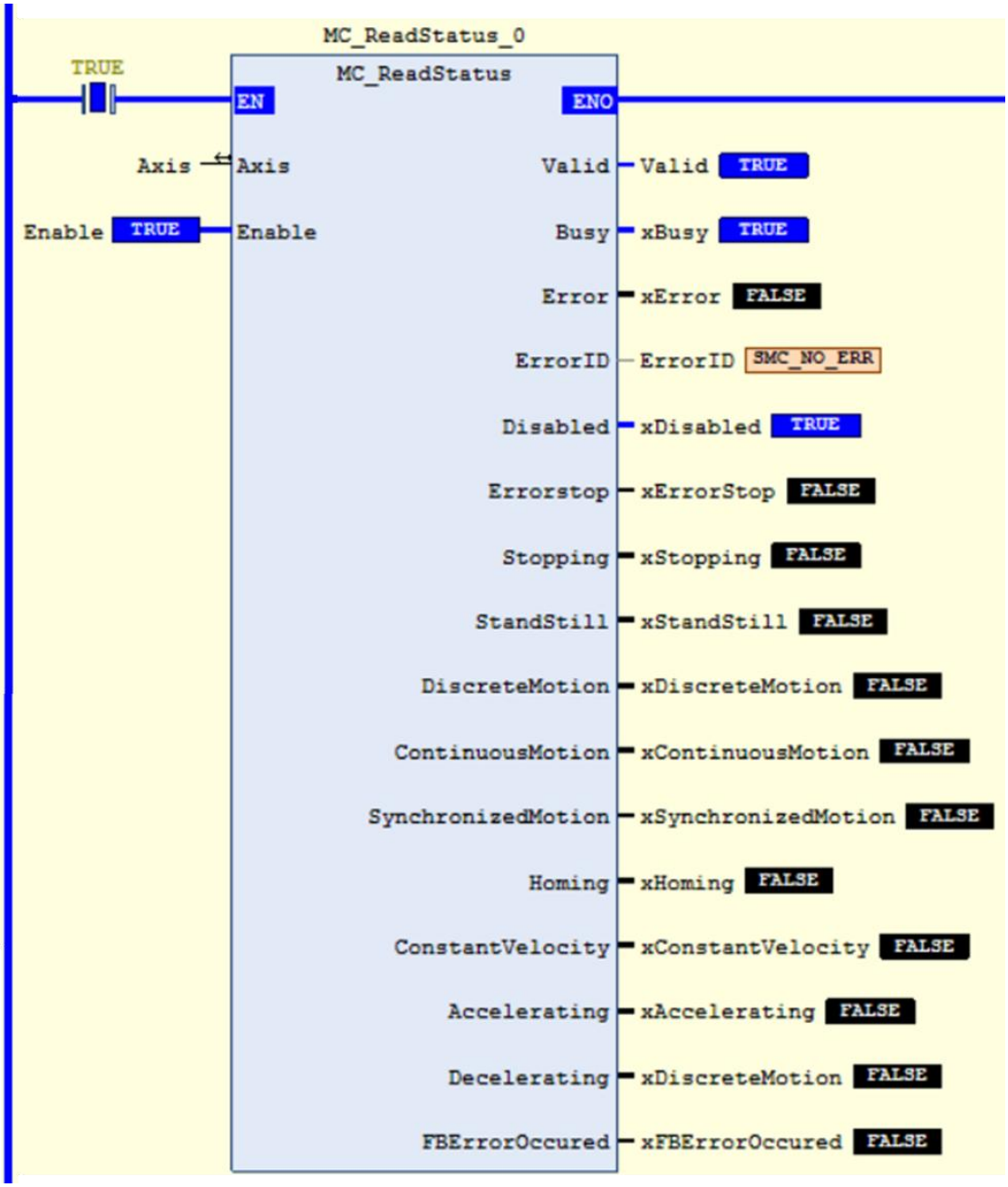
  - 可以用来读取轴的状态，并将状态引用到对应的程序步骤之中。Enable为false，则所有状态输出将被置为false；Enable为true时，持续读取轴的状态。

- **程序示例**

ST：当Enable为TRUE时，持续读取轴的当前状态。

```
MC_ReadStatus_0(
    Axis:=Axis ,
    Enable TRUE :=Enable TRUE ,
    Valid TRUE =>Valid TRUE ,
    Busy TRUE =>xBusy TRUE ,
    Error FALSE =>xError FALSE ,
    ErrorID SMC_NO_ERR =>ErrorID SMC_NO_ERR ,
    Disabled TRUE =>xDisabled TRUE ,
    Errorstop FALSE => xErrorStop FALSE,
    Stopping FALSE => xStopping FALSE,
    StandStill FALSE => xStandStill FALSE,
    DiscreteMotion FALSE => xDiscreteMotion FALSE,
    ContinuousMotion FALSE =>xContinuousMotion FALSE ,
    SynchronizedMotion FALSE =>xSynchronizedMotion FALSE ,
    Homing FALSE =>xHoming FALSE ,
    ConstantVelocity FALSE =>xConstantVelocity FALSE ,
    Accelerating FALSE =>xAccelerating FALSE ,
    Decelerating FALSE =>xDecelerating FALSE ,
    FBErrorOccured FALSE =>xFBErrorOccured FALSE );RETURN
```

LD： 当Enable为TRUE时，持续读取轴的当前状态。

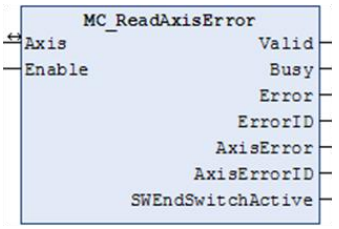


4.1.5 MC_ReadAxisError

本指令用于读取轴的错误。

- **指令格式**

指令	名称	FB/FC	LD表现	ST表现	库

MC_ReadAxisError	读轴错误	FB		MC_ReadAxisError(Axis：=， Enable：=， Valid=>， Busy=>， Error=>， ErrorID=>， AxisError=>， AxisErrorID=>， SWEndSwitchActive= >);	SM3_Basic

- **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	初始值	描述
Axis	轴	AXIS_REF_SM3	-	-	指定轴

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Enable	有效	BOOL	TRUE-FALSE	FALSE	必须设置为 TRUE，以激活功能块的处理。

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
Valid	有效	BOOL	TRUE-FALSE	FALSE	如果轴准备好则为 TRUE。
Busy	执行中	BOOL	TRUE-FALSE	FALSE	当功能块执行还没结束时为 TRUE。
Error	错误	BOOL	TRUE-FALSE	FALSE	功能块执行错误
ErrorID	错误代码	SMC_ERROR	-	0	错误指示，见 SMC_Error。
AxisError	轴报错	BOOL	TRUE-FALSE	FALSE	轴错误的标志
AxisErrorID	轴错误值	DWORD	0 正数	0	轴错误值
SWEndSwitchActive	软限位	BOOL	TRUE-FALSE	FALSE	如果超过软限位则为 TRUE。

	布尔	位串				整数								实数		时刻、持续时间 日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Axis	AXIS_REF_SM3																			
Enable	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Valid	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Busy	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Error	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
ErrorID	SMC_ERROR																			
AxisError	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
AxisErrorID	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
SWEndSwitch Active	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

- **功能说明：**

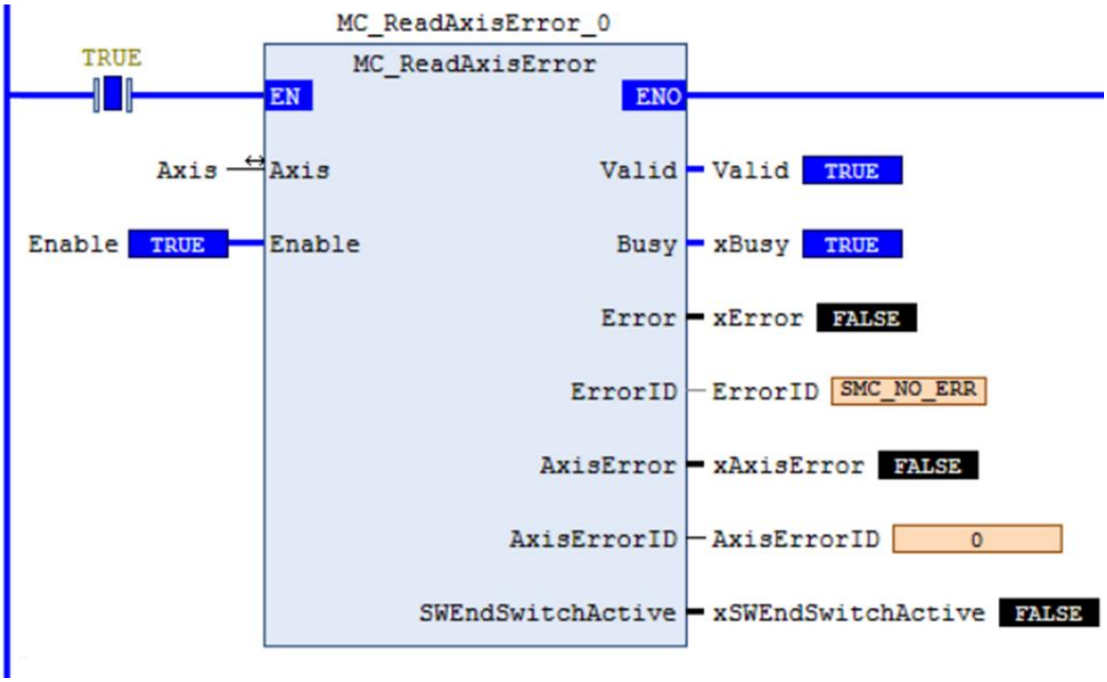
  ➢ 可以用来读取轴的错误，并将错误状态引用到对应的程序步骤之中。Enable为false，则所有输出将被置为false；Enable为true时，持续读取轴的错误。

- **程序示例**

ST：当Enable为TRUE时，持续读取轴Axis的错误。

```
MC_ReadAxisError_0(
    Axis:= Axis ,
    Enable TRUE := Enable TRUE ,
    Valid TRUE => Valid TRUE,
    Busy TRUE => xBusy TRUE,
    Error FALSE => xError FALSE,
    ErrorID SMC_NO_ERR => ErrorID SMC_NO_ERR,
    AxisError FALSE => xAxisError FALSE,
    AxisErrorID 0 => AxisErrorID 0,
    SWEndSwitchActive FALSE => xSWEndSwitchActive FALSE);RETURN
```

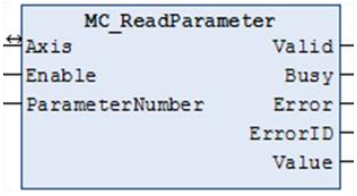
LD：当Enable为TRUE时，持续读取轴Axis的错误。



4.1.6 MC_ReadParameter

本指令用于读取指定的参数值。

- **指令格式**

指令	名称	FB/FC	LD表现	ST表现	库
MC_ReadParameter	读轴参数	FB		MC_ReadParameter(Axis：=, Enable：=, ParameterNumber：=, Valid=>, Busy=>, Error=>, ErrorID=>, Value=>);	SM3_Basic

- **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	初始值	描述
Axis	轴	AXIS_REF_SM3	-	-	指定轴

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Enable	启动	BOOL	TRUE-FALSE	FALSE	必须设置为 TRUE，以激活功能块的处理。
ParameterNumber	轴参数号	DINT	0 正数	0	访问轴参数的索引和子索引和序号

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
Valid	获取标志	BOOL	TRUE-FALSE	FALSE	如果参数已读取则为TRUE。
Busy	执行中	BOOL	TRUE-FALSE	FALSE	当功能块执行还没结束时为TRUE。
Error	错误	BOOL	TRUE-FALSE	FALSE	功能块执行错误
ErrorID	错误代码	SMC_ERROR	–	0	错误指示，见SMC_Error。
Value	参数值	LREAL	负数，0， 正数	0	读取参数的值

	布尔	位串				整数								实数		时刻、持续时间 日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Axis	AXIS_REF_SM3																			
Enable	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
ParameterNumber	_	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–
Valid	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Busy	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Error	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
ErrorID	SMC_ERROR																			
Value	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–

- **功能说明：**

  ➢ 可以在 SM3_Basic 库的 AXIS_REF_SM3 之中，找到ParameterNumber参数具体的参

数序号所代表的参数用途。

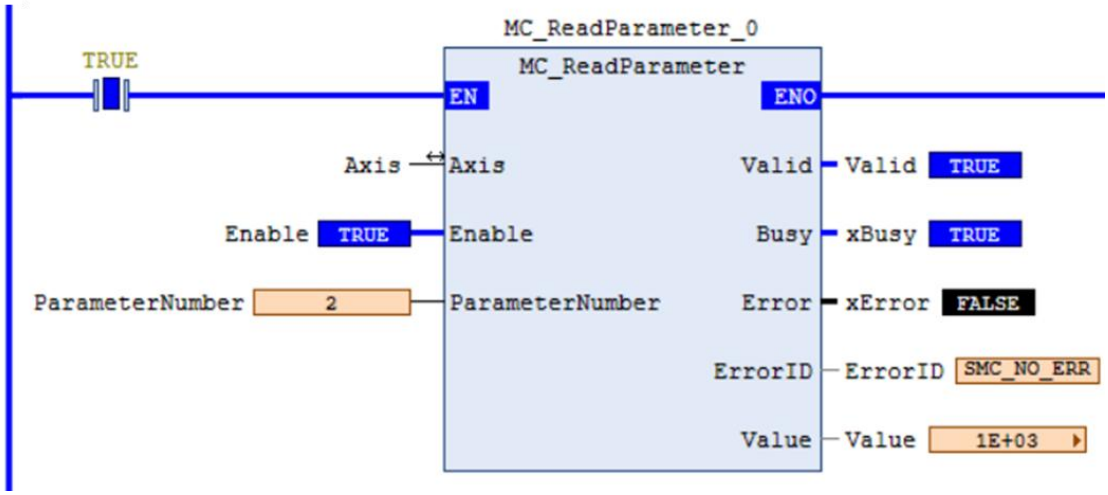
➢ Enable为false，则所有状态输出将被置为false；Enable为true时，持续读取轴的参数。

- **程序示例**

ST：当Enable为TRUE时，持续读取轴Axis的参数。

```
MC_ReadParameter_0(
    Axis:= Axis,
    Enable TRUE := Enable TRUE,
    ParameterNumber 2 := ParameterNumber 2,
    Valid TRUE => Valid TRUE,
    Busy TRUE => xBusy TRUE,
    Error FALSE => xError FALSE,
    ErrorID SMC_NO_ERR => ErrorID SMC_NO_ERR,
    Value 1E+03 => Value 1E+03 );RETURN
```

LD：当Enable为TRUE时，持续读取轴Axis的参数。

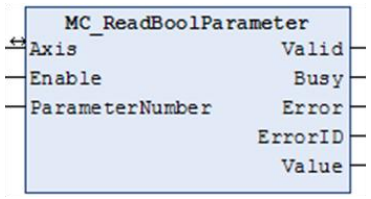


4.1.7 MC_ReadBoolParameter

本指令用于读取指定的 BOOL 型变量的值。

- **指令格式**

指令	名称	FB/FC	LD表现	ST表现	库

MC_ReadBoolParameter	读布尔参数	FB		MC_ReadBoolParameter(Axis：=， Enable：=， ParameterNumber：=， Valid=>， Busy=>， Error=>， ErrorID=>， Value=>);	SM3_Basic

- **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	初始值	描述
Axis	轴	AXIS_REF_SM3	-	-	指定轴

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Enable	启动	BOOL	TRUE-FALSE	FALSE	必须设置为 TRUE，以激活功能块的处理。
ParameterNumber	轴参数号	DINT	0 正数	0	访问轴参数的索引和子索引和序号

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
Valid	获取标志	BOOL	TRUE-FALSE	FALSE	如果参数已读取则为 TRUE。
Busy	执行中	BOOL	TRUE-FALSE	FALSE	当功能块执行还没结束时为 TRUE。
Error	错误	BOOL	TRUE-FALSE	FALSE	功能块执行错误
ErrorID	错误代码	SMC_ERROR	-	0	错误指示，见 SMC_Error。
Value	参数值	BOOL	TRUE-FALSE	FALSE	读取参数的值

	布	位串	整数	实数	时刻、持续时间

	尔															日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Axis	AXIS_REF_SM3																			
Enable	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
ParameterNumber	_	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–
Valid	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Busy	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Error	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
ErrorID	SMC_ERROR																			
Value	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

- **功能说明：**

  - 可以在 SM3_Basic 库的 AXIS_REF_SM3 之中，找到ParameterNumber参数具体的参数序号所代表的参数用途。

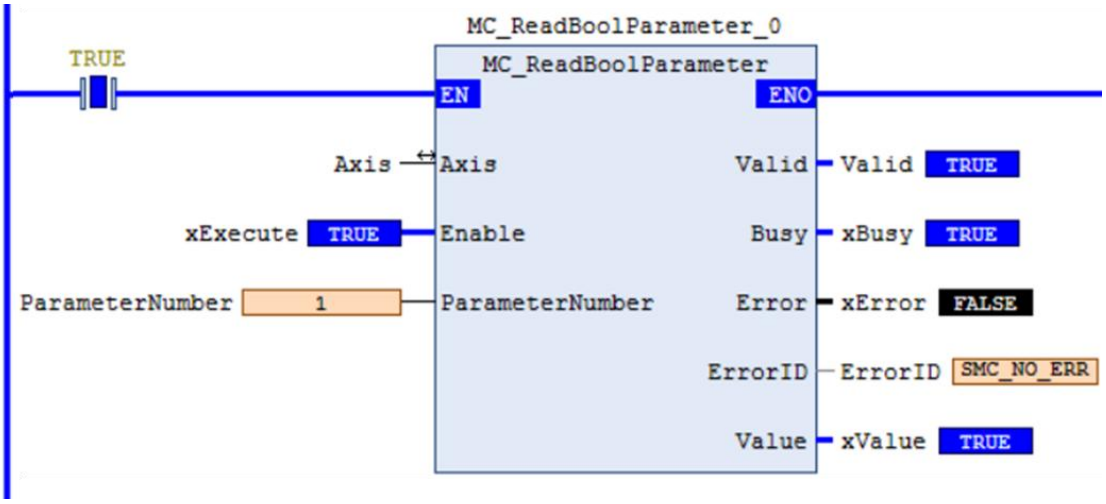
  - Enable为false，则所有状态输出将被置为false；Enable为true时，持续读取轴的参数。

- **程序示例**

ST：当Enable为TRUE时，持续读取轴的参数。

```
MC_ReadBoolParameter_0(
    Axis:= Axis,
    Enable TRUE := Enable TRUE,
    ParameterNumber 1 := ParameterNumber 1,
    Valid TRUE => Valid TRUE,
    Busy TRUE => Busy TRUE,
    Error FALSE => xError FALSE,
    ErrorID SMC_NO_ERR => ErrorID SMC_NO_ERR,
    Value TRUE => xValue TRUE );RETURN
```

LD：当Enable为TRUE时，持续读取轴的参数。



4.1.8 MC_WriteParameter

本指令用于写入指定的参数值。

- **指令格式**

指令	名称	FB/FC	LD表现	ST表现	库
MC_WriteParameter	写轴参数	FB	MC_WriteParameter Axis Done Execute Busy ParameterNumber Error Value ErrorID	MC_WriteParameter(Axis：=， Execute：=， ParameterNumber：=， Value：=， Done=>， Busy=>， Error=>， ErrorID=>);	SM3_Basic

- **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	初始值	描述
Axis	轴	AXIS_REF_SM3	-	-	指定轴

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述

Enable	启动	BOOL	TRUE-FALSE	FALSE	必须设置为 TRUE，以激活功能块的处理。
ParameterNumber	轴参数号	DINT	0 正数	0	访问轴参数的索引和子索引和序号
Value	参数值	LREAL	负数，0， 正数	0	需要写入的值。

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
Valid	获取标志	BOOL	TRUE-FALSE	FALSE	如果参数已写入则为 TRUE。
Busy	执行中	BOOL	TRUE-FALSE	FALSE	当功能块执行还没结束时为 TRUE。
Error	错误	BOOL	TRUE-FALSE	FALSE	功能块执行错误
ErrorID	错误代码	SMC_ERROR	-	0	错误指示，见 SMC_Error。

	布尔	位串				整数								实数		时刻、持续时间 日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Axis	AXIS_REF_SM3																			
Enable	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Value	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–
ParameterNumber	_	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–
Valid	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Busy	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Error	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
ErrorID	SMC_ERROR																			

- **功能说明：**

  - 可以在 SM3_Basic 库的 AXIS_REF_SM3 之中，找到ParameterNumber参数具体的参数序号所代表的参数用途。

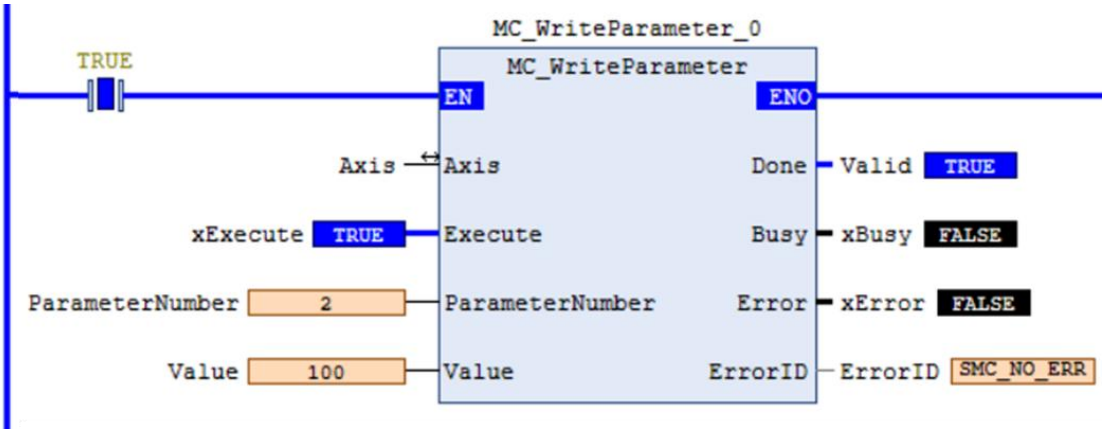
➢ Enable为true，写入指定的轴参数。

- **程序示例**

ST：当xExecute变为TRUE时，写入指定的轴参数。

```
MC_WriteParameter_0(
    Axis:= Axis,
    Execute TRUE := xExecute TRUE,
    ParameterNumber 2 := ParameterNumber 2,
    Value 100 := Value 100,
    Done TRUE => xDone TRUE,
    Busy FALSE => xBusy FALSE,
    Error FALSE => xError FALSE,
    ErrorID SMC_NO_ERR => ErrorID SMC_NO_ERR );RETURN
```

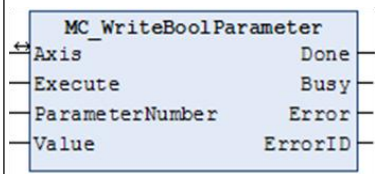
LD：当xExecute变为TRUE时，写入指定的轴参数。



4.1.9 MC_WriteBoolParameter

本指令用于写入 BOOL 类型的参数值。

- **指令格式**

指令	名称	FB/FC	LD表现	ST表现	库
MC_WriteBoolParameter	写布尔参数	FB		MC_WriteBoolParameter(Axis：=， Execute：=， ParameterNumber：=， Value：=，	SM3_Basic

				Done=>, Busy=>, Error=>, ErrorID=>);	

- **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	初始值	描述
Axis	轴	AXIS_REF_SM3	-	-	指定轴

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Enable	启动	BOOL	TRUE-FALSE	FALSE	必须设置为TRUE，以激活功能块的处理。
ParameterNumber	轴参数号	DINT	0 正数	0	访问轴参数的索引和子索引和序号
Value	参数值	BOOL	TRUE-FALSE	FALSE	需要写入的值。

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
Valid	获取标志	BOOL	TRUE-FALSE	FALSE	如果参数已写入则为 TRUE。
Busy	执行中	BOOL	TRUE-FALSE	FALSE	当功能块执行还没结束时为 TRUE。
Error	错误	BOOL	TRUE-FALSE	FALSE	功能块执行错误
ErrorID	错误代码	SMC_ERROR	-	0	错误指示，见 SMC_Error。

	布尔	位串	整数	实数	时刻、持续时间 日期、字符串

	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Axis	AXIS_REF_SM3																			
Enable	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Value	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
ParameterNumber	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–
Valid	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Busy	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Error	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
ErrorID	SMC_ERROR																			

- **功能说明：**

  - 可以在 SM3_Basic 库的 AXIS_REF_SM3 之中，找到ParameterNumber参数具体的参数序号所代表的参数用途。

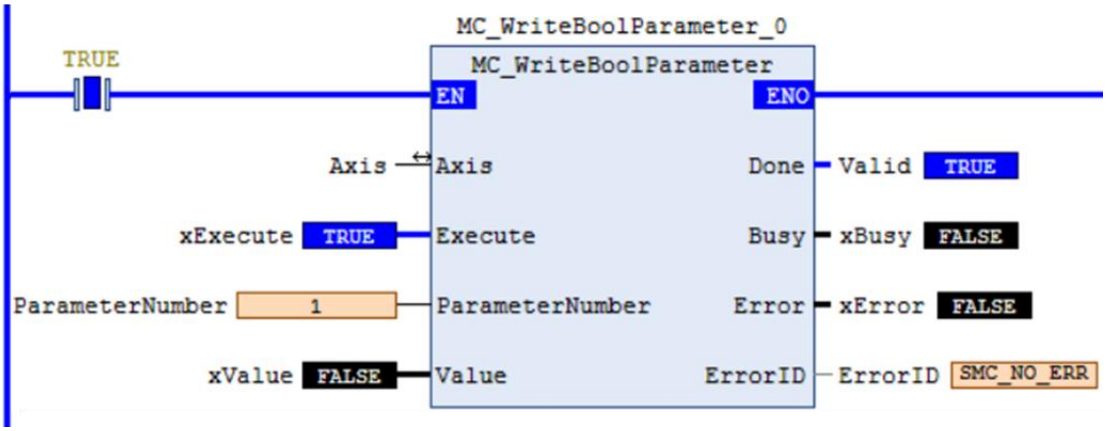
  - Enable为true，写入指定的轴参数。

- **程序示例**

ST：当xExecute变为TRUE时，写入指定的轴参数。

```
MC_WriteBoolParameter_0(
    Axis:= Axis,
    Execute TRUE := xExecute TRUE,
    ParameterNumber 1 := ParameterNumber 1,
    Value FALSE := xValue FALSE,
    Done TRUE => xDone TRUE,
    Busy FALSE => xBusy FALSE,
    Error FALSE => xError FALSE,
    ErrorID SMC_NO_ERR => ErrorID SMC_NO_ERR );RETURN
```

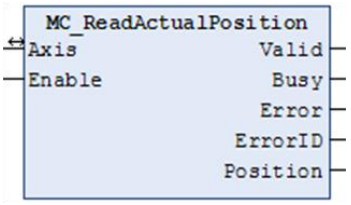
LD：当xExecute变为TRUE时，写入指定的轴参数。



4.1.10 MC_ReadActualPosition

本指令用于读取轴当前实际位置值。

- **指令格式**

指令	名称	FB/FC	LD表现	ST表现	库
MC_ReadActualPosition	读轴实际位置	FB		MC_ReadActualPosition(Axis：=， Enable：=， Valid=>， Busy=>， Error=>， ErrorID=>， Position=>);	SM3_Basic

- **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	初始值	描述
Axis	轴	AXIS_REF_SM3	-	-	指定轴

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Enable	启动	BOOL	TRUE-FALSE	FALSE	必须设置为TRUE，以激活功能块的处理。

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
Valid	获取标志	BOOL	TRUE-FALSE	FALSE	如果输出值有效，则为 TRUE。
Busy	执行中	BOOL	TRUE-FALSE	FALSE	当功能块执行还没结束时为 TRUE。
Error	错误	BOOL	TRUE-FALSE	FALSE	功能块执行错误
ErrorID	错误代码	SMC_ERROR	-	0	错误指示，见 SMC_Error。
Position	位置	LREAL	负数，0，正数	0	新的绝对位置（以轴的单位表达[u]）。

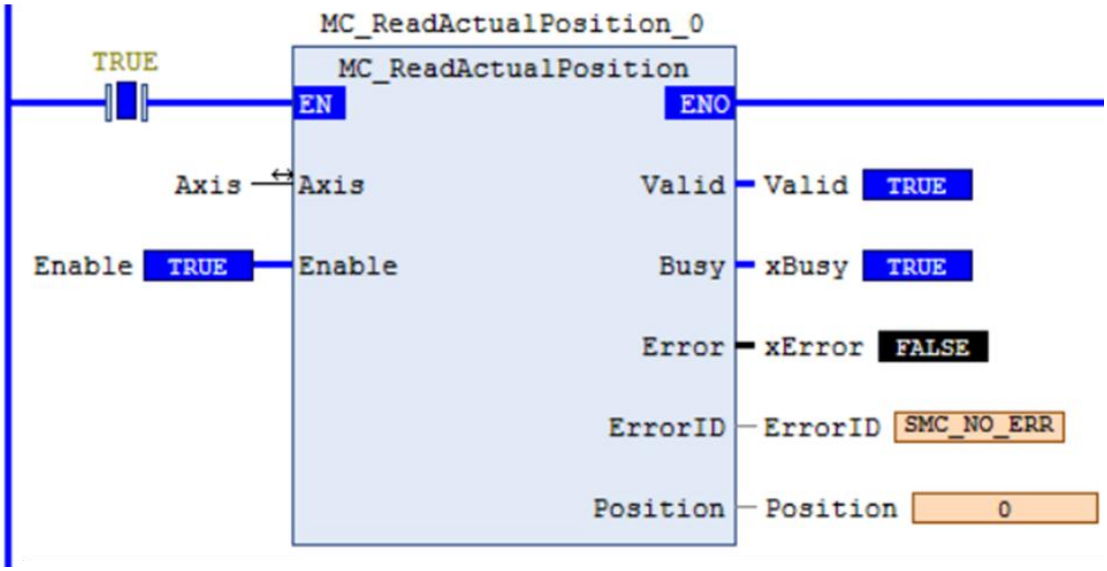
	布尔	位串				整数								实数		时刻、持续时间 日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Axis	AXIS_REF_SM3																			
Enable	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Valid	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Busy	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Error	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
ErrorID	SMC_ERROR																			
Position	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-

- **功能说明：**
  - 这个指令用来读取驱动器中轴的实际位置，即fActPosition。
  - Enable为false，则所有状态输出将被置为false；Enable为true时，持续读取轴的实际位置。
- **程序示例**

ST：当Enable为TRUE时，持续读取轴的实际位置。

```
MC_ReadActualPosition_0(
    Axis:= Axis,
    Enable TRUE := Enable TRUE,
    Valid TRUE => Valid TRUE,
    Busy TRUE => xBusy TRUE,
    Error FALSE => xError FALSE,
    ErrorID SMC_NO_ERR => ErrorID SMC_NO_ERR,
    Position 0 => Position 0 );RETURN
```

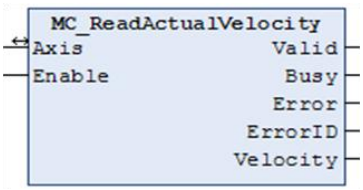
LD：当Enable为TRUE时，持续读取轴的实际位置。



4.1.11 MC_ReadActualVelocity

本指令用于读取轴的实际速度值。

- **指令格式**

指令	名称	FB/FC	LD表现	ST表现	库
MC_ReadActualVelocity	读实际速度	FB		MC_ReadActualVelocity(Axis：=， Enable：=， Valid=>， Busy=>， Error=>， ErrorID=>， Velocity=>);	SM3_Basic

- **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	初始值	描述
Axis	轴	AXIS_REF_SM3	-	-	指定轴

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Enable	启动	BOOL	TRUE-FALSE	FALSE	必须设置为TRUE，以激活功能块的处理。

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
Valid	获取标志	BOOL	TRUE-FALSE	FALSE	如果输出值有效，则为 TRUE。
Busy	执行中	BOOL	TRUE-FALSE	FALSE	当功能块执行还没结束时为 TRUE。
Error	错误	BOOL	TRUE-FALSE	FALSE	功能块执行错误
ErrorID	错误代码	SMC_ERROR	-	0	错误指示，见 SMC_Error。
Velocity	速度	LREAL	负数，0，正数	0	实际速度的值（以[计数单位/秒]表达）。

	布尔	位串				整数								实数		时刻、持续时间 日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Axis	AXIS_REF_SM3																			
Enable	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Valid	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Busy	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Error	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
ErrorID	SMC_ERROR																			
Velocity	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-

- **功能说明：**

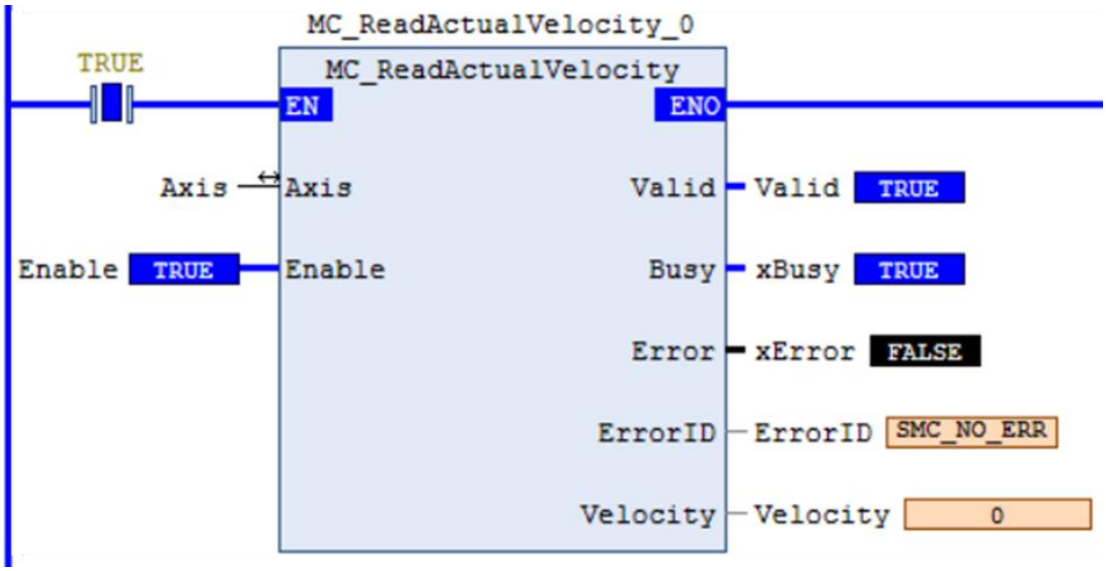
  - 这个指令用来读取驱动器中轴的实际速度，即fActVelocity。
  - Enable为false，则所有状态输出将被置为false；Enable为true时，持续读取轴的当前速度。

- **程序示例**

ST：当Enable为TRUE时，持续读取轴的当前速度。

```
MC_ReadActualVelocity_0(
    Axis:= Axis,
    Enable TRUE := Enable TRUE,
    Valid TRUE => Valid TRUE,
    Busy TRUE => xBusy TRUE,
    Error FALSE => xError FALSE,
    ErrorID SMC_NO_ERR => ErrorID SMC_NO_ERR,
    Velocity 0 => Velocity 0 );RETURN
```

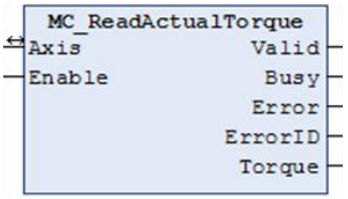
LD：当Enable为TRUE时，持续读取轴的当前速度。



4.1.12 MC_ReadActualTorque

本指令用于读取轴实际转矩值。

- **指令格式**

指令	名称	FB/FC	LD表现	ST表现	库
MC_ReadActualTorque	读实际力矩	FB		MC_ReadActualTorque(Axis：=， Enable：=， Valid=>， Busy=>， Error=>， ErrorID=>， Torque=>);	SM3_Basic

- **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	初始值	描述
Axis	轴	AXIS_REF_SM3	-	-	指定轴

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Enable	启动	BOOL	TRUE-FALSE	FALSE	必须设置为 TRUE，以激活功能块的处理。

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
Valid	获取标志	BOOL	TRUE-FALSE	FALSE	如果输出值有效，则为 TRUE。
Busy	执行中	BOOL	TRUE-FALSE	FALSE	当功能块执行还没结束时为 TRUE。
Error	错误	BOOL	TRUE-FALSE	FALSE	功能块执行错误
ErrorID	错误代码	SMC_ERROR	-	0	错误指示，见 SMC_Error。
Torque	力矩	LREAL	负数，0，正数	0	实际转矩或力矩的值（以计数单位表达）

	布	位串	整数	实数	时刻、持续时间

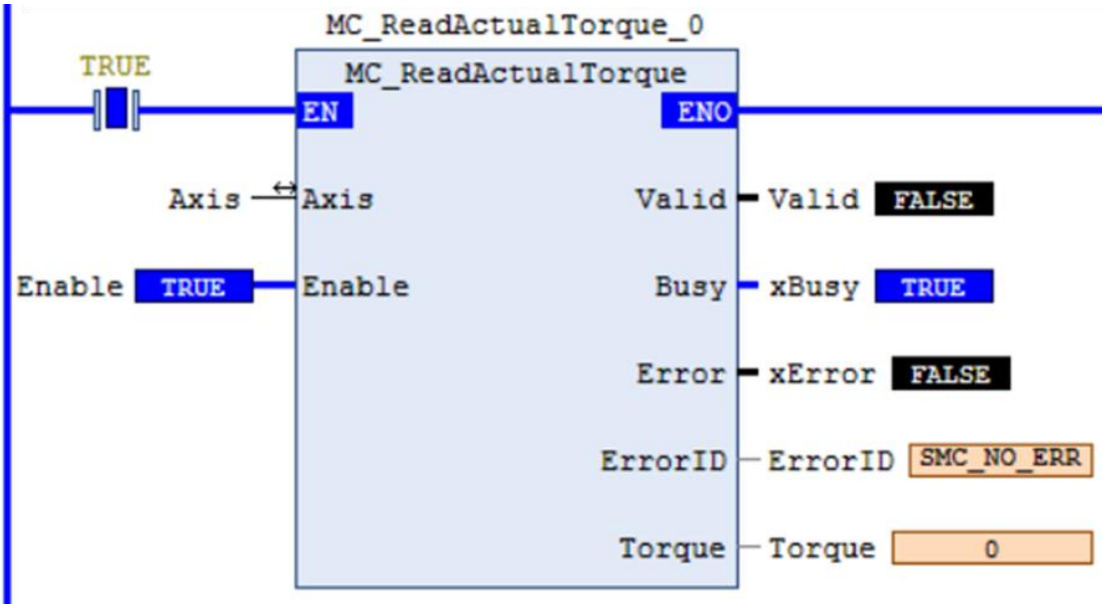
	尔															日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Axis	AXIS_REF_SM3																			
Enable	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Valid	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Busy	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Error	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
ErrorID	SMC_ERROR																			
Torque	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–

- **功能说明：**
  - 这个指令用来读取驱动器中轴的实际力矩。
  - Enable为false，则所有状态输出将被置为false；Enable为true时，持续读取轴的当前转矩。
- **程序示例**

ST：当Enable为TRUE时，持续读取轴的当前转矩。

```
MC_ReadActualTorque_0(
    Axis:= Axis,
    Enable TRUE := Enable TRUE,
    Valid FALSE => Valid FALSE,
    Busy TRUE => xBusy TRUE,
    Error FALSE => xError FALSE,
    ErrorID SMC_NO_ERR => ErrorID SMC_NO_ERR,
    Torque 0 => Torque 0 );RETURN
```

LD：当Enable为TRUE时，持续读取轴的当前转矩。



4.1.13 MC_SetPosition

本指令用于设置轴当前的位置（不会引起轴的运动）。

- **指令格式**

指令	名称	FB/FC	LD表现	ST表现	库
MC_SetPosition	设置轴位置	FB	MC_SetPosition Axis Done Execute Busy Position Error Mode ErrorID	MC_SetPosition(Axis：=， Execute：=， Position， Mode：=， Done=>， Busy=>， Error=>， ErrorID=>);	SM3_Basic

- **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	初始值	描述
Axis	轴	AXIS_REF_SM3	-	-	指定轴

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Execute	启动	BOOL	TRUE-FALSE	FALSE	输入值的上升沿将启动该功能块的执行。
Position	位置	LREAL	负数，0，正数	0	位置单位[u]（表示“距离”如果 Mode = RELATIVE）
Mode	模式	BOOL	TRUE-FALSE	FALSE	TRUE =相对， FALSE =绝对

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
Done	完成	BOOL	TRUE-FALSE	FALSE	功能块执行完成为 TRUE。
Busy	执行中	BOOL	TRUE-FALSE	FALSE	当功能块执行还没结束时为 TRUE。
Error	错误	BOOL	TRUE-FALSE	FALSE	功能块执行错误
ErrorID	错误代码	SMC_ERROR	–	0	错误指示，见 SMC_Error。

	布尔	位串				整数								实数		时刻、持续时间 日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Axis	AXIS_REF_SM3																			
Execute	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Position	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–
Mode	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Done	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Busy	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Error	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
ErrorID	SMC_ERROR																			

- **功能说明：**

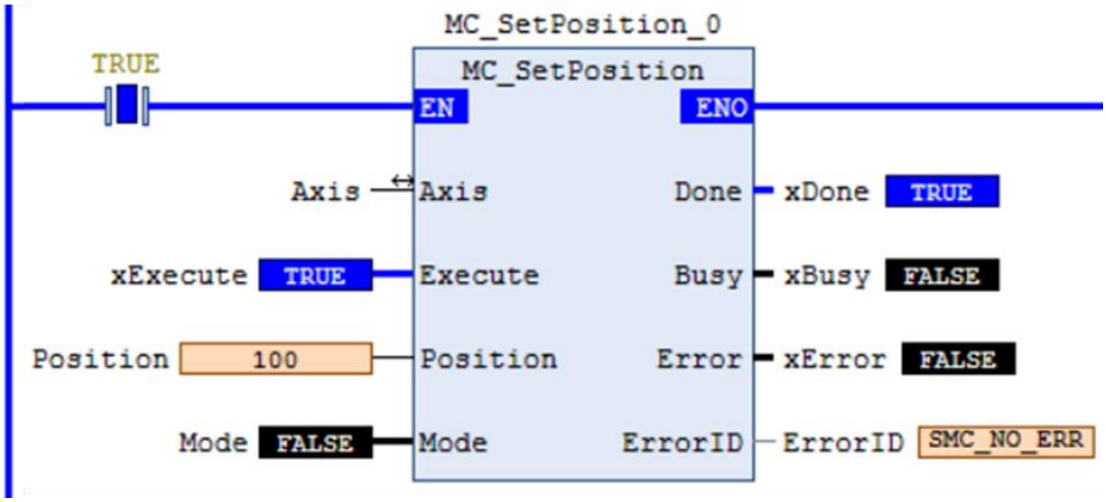
  - Execute的上升沿将触发这个指令的执行，指令执行后会根据Mode设置模式，将Position参数设置到轴的当前位置，但不会引起轴的运动。

- **程序示例**

ST：当xExecute变为TRUE时，将Position参数设置到轴的当前位置。

```
MC_SetPosition_0(
    Axis:= Axis,
    Execute TRUE := xExecute TRUE,
    Position 100 := Position 100,
    Mode FALSE := Mode FALSE,
    Done TRUE => xDone TRUE,
    Busy FALSE => xBusy FALSE ,
    Error FALSE => xError FALSE,
    ErrorID SMC_NO_ERR => ErrorID SMC_NO_ERR );RETURN
```

LD：当xExecute变为TRUE时，将Position参数设置到轴的当前位置。



4.1.14 SMC_ReadSetPosition

本指令用于读取当前轴的设置位置。

- **指令格式**

指令	名称	FB/FC	LD表现	ST表现	库
SMC_ReadSetPosition	读取设置位置	FB	SMC_ReadSetPosition Axis — Valid Enable — Busy Error ErrorID Position	SMC_ReadSetPosition (Axis：=, Enable：=, Valid=>, Busy=>,	SM3_Basic

				Error=>， ErrorID=>， Position=>);	

- **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	初始值	描述
Axis	轴	AXIS_REF_SM3	-	-	指定轴

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Enable	有效	BOOL	TRUE-FALSE	FALSE	必须被设为 TRUE 以使功能块能够执行。

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
Valid	获取标志	BOOL	TRUE-FALSE	FALSE	如果参数有效则为 TRUE。
Busy	执行中	BOOL	TRUE-FALSE	FALSE	当功能块执行还没结束时为 TRUE。
Error	错误	BOOL	TRUE-FALSE	FALSE	功能块执行错误
ErrorID	错误代码	SMC_ERROR	-	0	错误指示，见 SMC_Error。
Position	位置	LREAL	负数，0，正数	0	设置的位置值

	布尔	位串				整数								实数		时刻、持续时间 日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Axis	AXIS_REF_SM3																			
Enable	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

Valid	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Busy	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Error	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
ErrorID	SMC_ERROR																		
Position	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–

- **功能说明：**

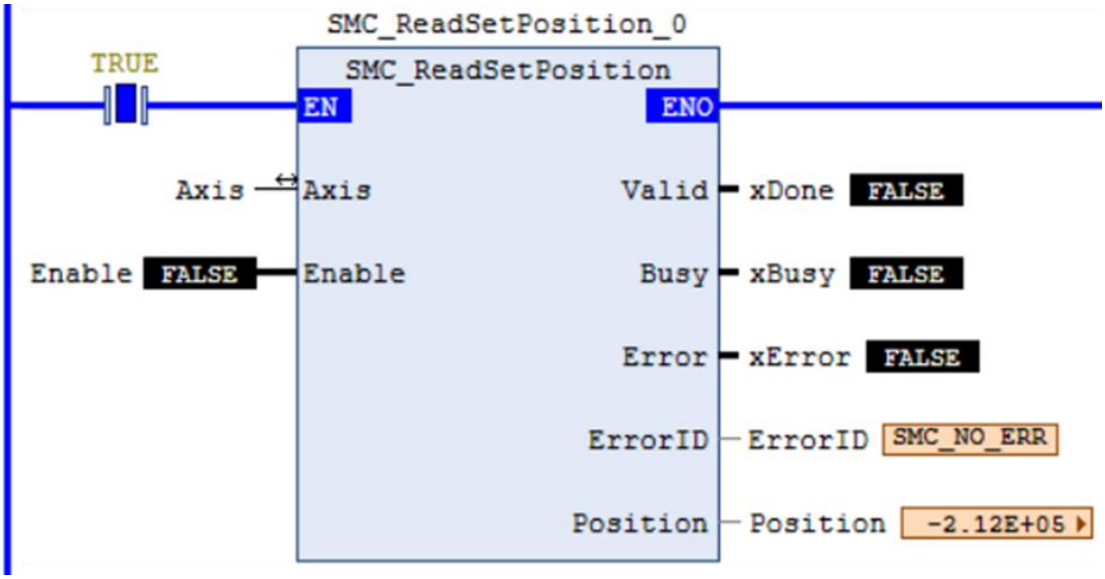
  - 这个指令用来读取轴的设置位置。
  - Enable为false，则所有状态输出将被置为false；Enable为true时，持续读取轴的设置位置。

- **程序示例**

ST：当Enable为TRUE时，持续读取轴的设置位置。

```
SMC_ReadSetPosition_0(
    Axis:= Axis,
    Enable TRUE := Enable TRUE,
    Valid TRUE => Valid TRUE,
    Busy TRUE => xBusy TRUE,
    Error FALSE => xError FALSE,
    ErrorID SMC_NO_ERR => ErrorID SMC_NO_ERR,
    Position -2.12E+05 => Position -2.12E+05 );RETURN
```

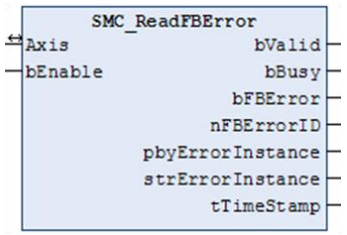
LD：当Enable为TRUE时，持续读取轴的设置位置。



4.1.15 SMC_ReadFBError

本指令用于读取轴功能块的错误信息。

- **指令格式**

指令	名称	FB/FC	LD表现	ST表现	库
SMC_ReadFBError	读功能块错误信息	FB		SMC_ReadFBError(Axis：=， bEnable：=， bValid=>， bBusy=>， bFBError=>， nFBErrorID=>， pbyErrorInstance=>， strErrorInstance=>， tTimeStamp=>);	SM3_Basic

- **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	初始值	描述
Axis	轴	AXIS_REF_SM3	–	–	指定轴

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
bEnable	有效	BOOL	TRUE-FALSE	FALSE	必须被设为 TRUE 以使功能块能够执行。

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
bValid	获取标志	BOOL	TRUE-FALSE	FALSE	如果参数有效则为 TRUE。
bBusy	执行中	BOOL	TRUE-FALSE	FALSE	当功能块执行还没结束时为 TRUE。
bFBError	错误	BOOL	TRUE-FALSE	FALSE	功能块执行错误
nFBErrorID	错误代码	SMC_ERROR	-	0	错误指示，见 SMC_Error。
pbyErrorInstance	错误指针	POINTER TO BYTE	-	-	指针指向报错的 FB
strErrorInstance		STRING	-	-	指向错误功能块（程序，子程序，功能块）
tTimeStamp	时间戳	TIME	-	0	错误发生的时间戳

	布尔	位串				整数								实数		时刻、持续时间 日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Axis	AXIS_REF_SM3																			
bEnable	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
bValid	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
bBusy	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
bFBError	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
nFBErrorID	SMC_ERROR																			
pbyErrorInsta	POINTER TO BYTE																			

nce																				
strErrorInstance	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√
tTimeStamp	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–

- **功能说明：**

  - Enable 为 TRUE时，无错误则 Valid，Busy 输出为 TURE；有功能块报警则bFBError 输出为 true。

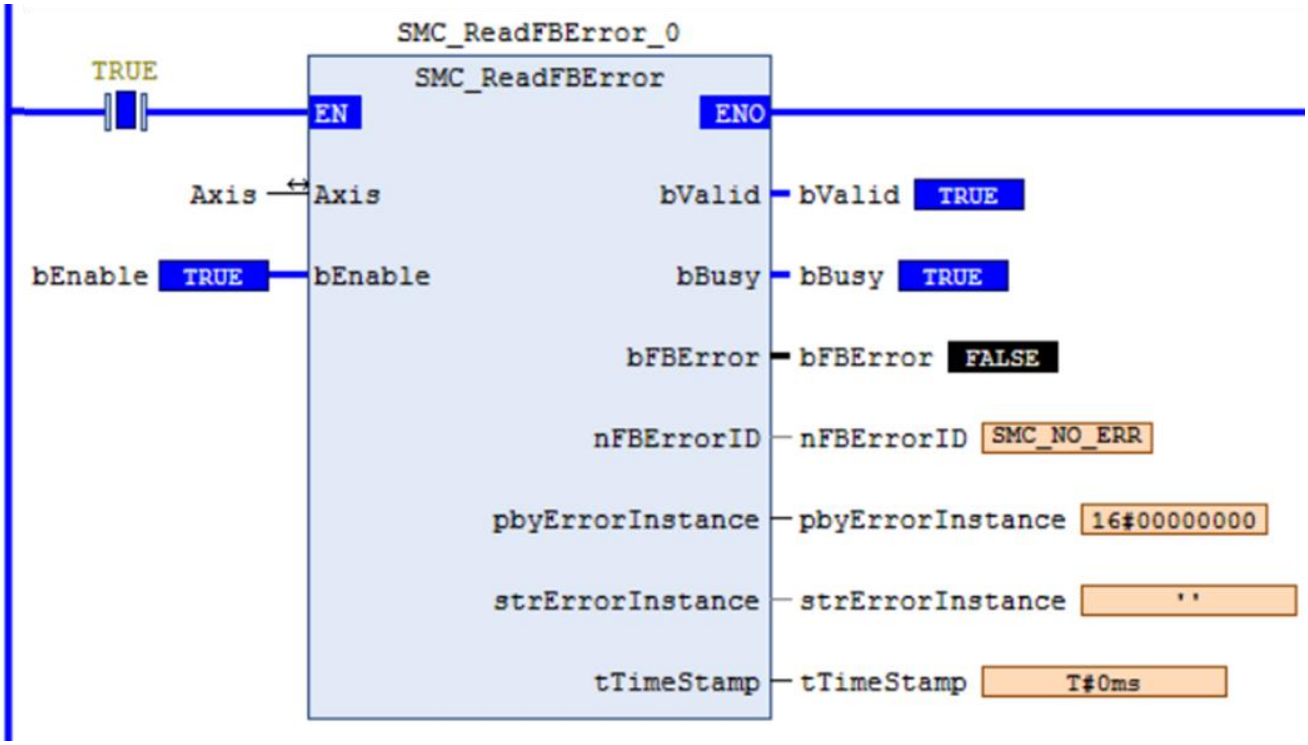
  - Enable 变为 FALSE，，则 Valid，Busy 输出为 FALSE。

- **程序示例**

ST：当Enable为TRUE时，读取轴功能块错误信息。

```
SMC_ReadFBError_0(
    Axis:= Axis,
    bEnable TRUE := bEnable TRUE,
    bValid TRUE => bValid TRUE,
    bBusy TRUE => bBusy TRUE,
    bFBError FALSE => bFBError FALSE,
    nFBErrorID SMC_NO_ERR => nFBErrorID SMC_NO_ERR,
    pbyErrorInstance 16#00000000 => pbyErrorInstance 16#00000000,
    strErrorInstance '' => strErrorInstance '',
    tTimeStamp T#0ms => tTimeStamp T#0ms);RETURN
```

LD：当Enable为TRUE时，读取轴功能块错误信息。



- **注意事项**

➢ 只有当Enable为True时，功能块进入运行状态。

4.1.16 SMC_ClearFBError

本指令用于清除功能块的历史错误信息。

- **指令格式**

指令	名称	FB/FC	LD表现	ST表现	库
SMC_ClearFBError	清除功能块错误	FC	SMC_ClearFBError pDrive	SMC_ClearFBError(pDrive：=);	SM3_Basic

- **相关变量**

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述

pDrive	轴指针	POINTER TO AXIS_REF_SM3	–	–	该输入为指向轴结构体的地址指针。

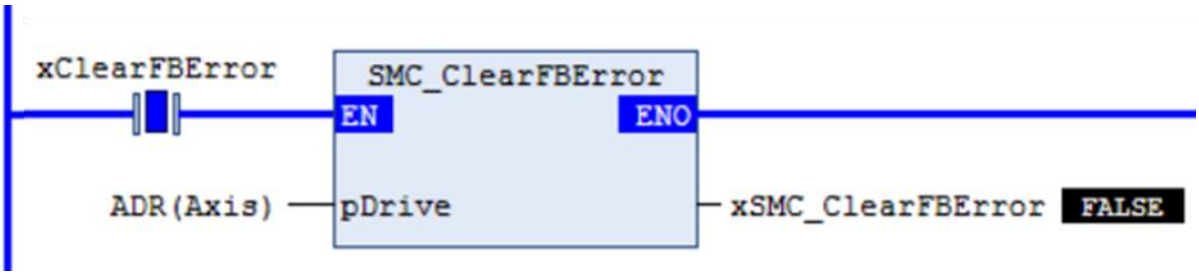
	布尔	位串				整数								实数		时刻、持续时间 日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
pDrive	POINTER TO AXIS_REF_SM3																			

- **功能说明：**
  - 当轴出现错误，调用复位功能块将轴复位后，需要调用该功能块清除轴的历史错误状态。
  - Execute的上升沿将触发这个指令的执行。
- **程序示例**

ST：当xClearFBError变为TRUE时，清除轴Axis的历史错误状态。

```
IF xClearFBError TRUE THEN
    xSMC_ClearFBError FALSE := SMC_ClearFBError(pDrive:= ADR(Axis) );
END_IF
RETURN
```

LD：当xClearFBError变为TRUE时，清除轴Axis的历史错误状态。



4.2 单轴运动控制

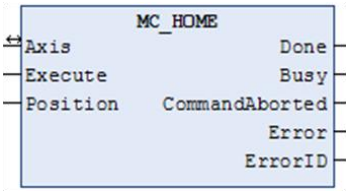
4.2.1 指令列表

指令类别	名称	FB/FC	功能
单轴运动控制	MC_Home	FB	回零
	MC_MoveAbsolute	FB	绝对运动
	MC_MoveRelative	FB	相对运动
	MC_MoveVelocity	FB	定速运动
	MC_Stop	FB	停止
	MC_Halt	FB	暂停
	MC_Jog	FB	点动运动
	MC_MoveAdditive	FB	附加运动
	MC_MoveSuperImposed	FB	叠加运动
	MC_PositionProfile	FB	位置规划
	MC_VelocityProfile	FB	速度规划
	MC_AccelerationProfile	FB	加速度规划
	SMC_Homing	FB	上位机回原
	SMC_Inch	FB	寸动

4.2.2 MC_Home

本指令用于执行总线电机回零动作，具体回零过程由总线驱动设计的回零模式决定。

- **指令格式**

指令	名称	FB/FC	LD表现	ST表现	库
MC_Home	总线轴回原	FB		MC_Home(Axis：=， Execute：=， Position：=， Done=>， Busy=>， CommandAborted=>， Error=>， ErrorID=>);	SM3_Basic

- **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	初始值	描述
Axis	轴	AXIS_REF_SM3	-	-	指定轴

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Execute	启动	BOOL	TRUE-FALSE	FALSE	上升沿触发指令运行
Position	位置	LREAL	负数，0，正数	0	驱动器的回零偏置 16#607C

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
Done	完成	BOOL	TRUE-FALSE	FALSE	如果回零已完成则为 TRUE。
Busy	执行中	BOOL	TRUE-FALSE	FALSE	当功能块执行还没结束时为 TRUE。
Command Aborted	命令中断	BOOL	TRUE-FALSE	FALSE	如果该命令已被其他命令终止则为 TRUE。
Error	错误	BOOL	TRUE-FALSE	FALSE	在功能块内部发生错误的信号
ErrorID	错误代码	SMC_ERROR	-	0	错误指示，见 SMC_Error。

	布尔	位串				整数								实数		时刻、持续时间 日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Axis	AXIS_REF_SM3																			
Execute	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Position	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-
Done	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Busy	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

CommandAborted	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Error	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
ErrorID	SMC_ERROR																			

- **功能说明：**

  - 这是总线驱动的回零运动指令，当所控制的运动轴是一个总线轴时，需要调用这个指令，以实现运动轴的回零。
  - 此回零指令的回零模式由连接在总线上的从站决定，控制端只是将需要的回零参数发给驱动器，具体的回零动作，在驱动器端完成。
  - 一般驱动器的回零偏移地址为16#607C，各驱动器的处理不同，具体请参考该驱动器的回零流程及回零参数说明。
  - 在执行回零之前，需要配置总线驱动的回零参数，如回零模式、速度、加速度等。总线驱动回零需要配置哪些参数，请参考所使用驱动的手册。
  - 一般的总线驱动回零，需要设置索引和子索引的数据如下表所示。

总线回零设置

参数	索引	子索引	描述
回零方式	0x6098		各家驱动器回零方式有所差异，需要根据具体的驱动厂家手册，选择相应的回零方式
回零高速	0x6099	0x01	开始回零到找到零点过程的速度，数值较高 ， 以减少回零时间
回零低速	0x6099	0x02	找到零点到回零完成过程的速度，数值较低，以提高精度
回零加减速	0x609A		在原点回归时的加减速变化

修改回零模式的三种方式：

1) 控制器通过SDO服务向驱动器写入回零参数；

2) 通过配置PDO参数并映射相关回零参数进行回零设置；

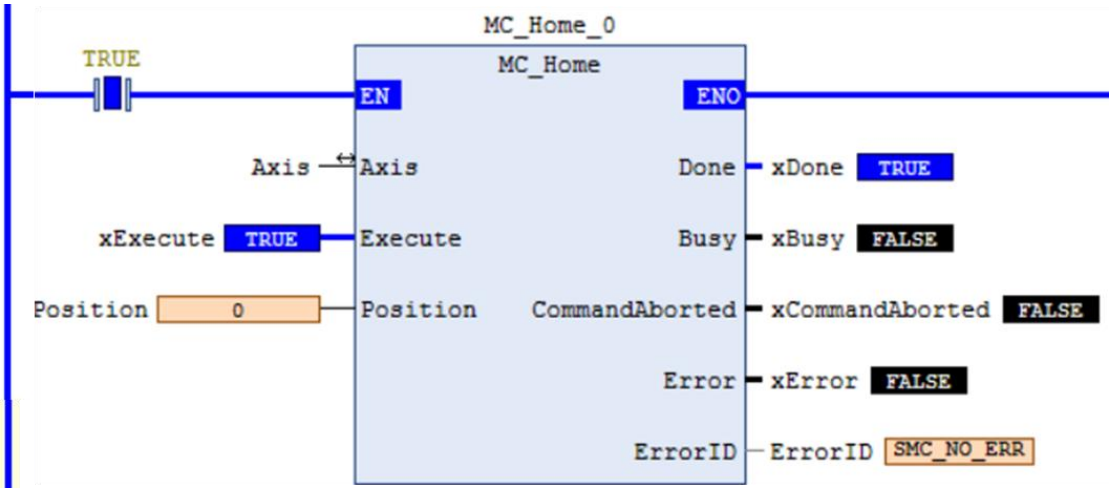
3) 通过配置启动参数向驱动器写入回零参数。

- **程序示例**

ST：当xExecute变为TRUE时，执行回零指令。

```
MC_Home_0(
    Axis:= Axis,
    Execute TRUE := xExecute TRUE,
    Position 0 := Position 0,
    Done TRUE => xDone TRUE,
    Busy FALSE => xBusy FALSE,
    CommandAborted FALSE => xCommandAborted FALSE,
    Error FALSE => xError FALSE,
    ErrorID SMC_NO_ERR => ErrorID SMC_NO_ERR );RETURN
```

LD：当xExecute变为TRUE时，执行回零指令。

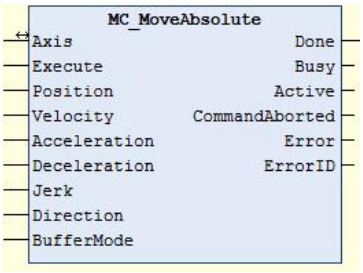


4.2.3 MC_MoveAbsolute

本指令用于实现将控制轴按照设定参数，运动到指定的绝对位置。

- **指令格式**

指令	名称	FB/FC	LD表现	ST表现	库

MC_Move Absolute	绝对位置	FB		MC_MoveAbsolute(Axis：=， Execute：=， Position：=， Velocity：=， Acceleration：=， Deceleration：=， Jerk：=， Direction：=， BufferMode：=， Done=>， Busy=>， Active=>， CommandAborted=>， Error=>， ErrorID=>);	SM3_Basic

- **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	初始值	描述
Axis	轴	AXIS_REF_SM3	-	-	指定轴

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Execute	启动	BOOL	TRUE-FALSE	FALSE	上升沿触发指令运动。
Position	位置	LREAL	负数，0，正数	0	运动的目标位置 [u]（负或正）。
Velocity	目标速度	LREAL	0，正数	0	速度最大值（总是为正） [u/s]
Acceleration	目标加速度	LREAL	0，正数	0	加速度值（总是为正）[u/s2]

Deceleration	目标减速度	LREAL	0，正数	0	减速度值（总是为正）[u/s2]
Jerk	目标加加速度	LREAL	0，正数	0	加加速度值（总是为正）[u/s3]
Direction	方向	MC_DIRECTION	-1,1	Positive	1，Positive，正； -1，Negative，负
BufferMode	缓存模式	MC_BUFFER_MODE	0-5	0	指定多重启动运动指令时的动作。 0： mcAborting，中断； 1： mcBuffered，等待； 2： mcBlendingLow，以低速合并； 3： mcBlendingPrevious，以前一个速度合并； 4： mcBlendingNext，以后一个速度合并； 5： mcBlendingHigh，以高速合并

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
Done	执行完成	BOOL	TRUE-FALSE	FALSE	指令执行完成时为 TRUE
Busy	执行中	BOOL	TRUE-FALSE	FALSE	功能块的处理没有完成为 TRUE
Active		BOOL			该轴在工作
CommandAborted	指令被中断	BOOL	TRUE-FALSE	FALSE	指令执行被中断时为 TRUE
Error	错误	BOOL	TRUE-FALSE	FALSE	发生异常时为 TRUE
ErrorID	错误代码	SMC_ERROR	–	0	正常时数值为 0，发生异常时，输出报错代码。

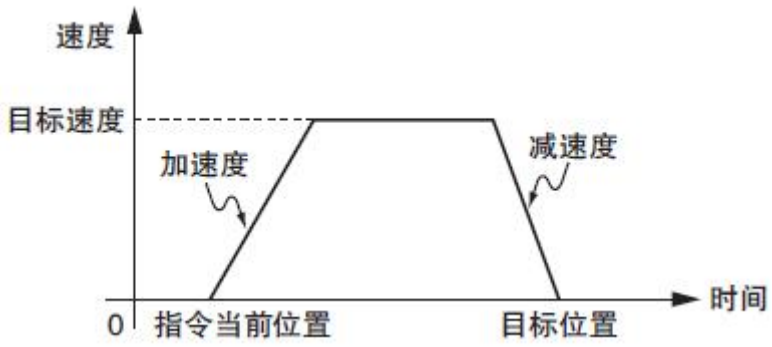
	布尔	位串				整数								实数		时刻、持续时间 日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Axis	AXIS_REF_SM3																			

Execute	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Position	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–
Velocity	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–
Acceleration	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–
Deceleration	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–
Jerk	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–
Direction	MC_DIRECTION																		
BufferMode	MC_BUFFER_MODE																		
Done	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Busy	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Active	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
CommandAborted	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Error	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
ErrorID	SMC_ERROR																		

- **功能说明：**

  - 如果是线性轴的绝对点位运动，方向值将被忽略。

  - 当速度曲线是梯形曲线时，指令执行时的“速度-时间”曲线如下图所示。



  - 如果距离过短，可能达不到最大速度。

  - 将加减速度设定为0后，将不做加减速而直接达到目标速度。

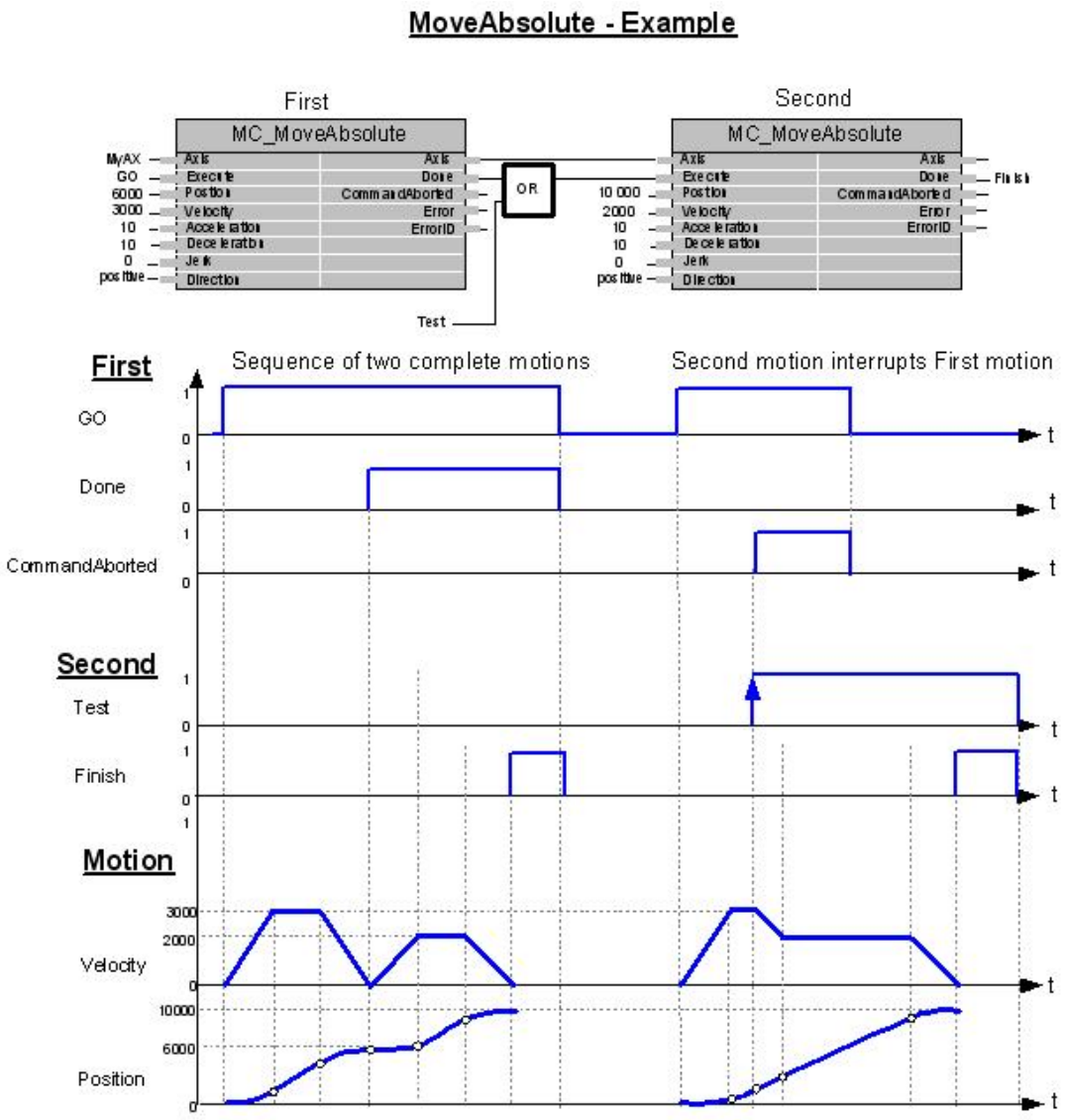
- **例 MC_MoveAbsolute**

  实现功能：

下图是两个 MC_MoveAbsolute 功能块的实例“First”和“Second”连接的例程：

1) 时序图的前段画出了“Second”在“First”之后调用的情况。当“First”达到要求的位置6000（速度为0）时，输出Done将引发“Second”向位置10000移动。

2) 时序图的后段画出了当“First”还在执行时“Second”被启动的情况。这时，“First”的运动在速度恒定期间被Test信号终止。“Second”功能块将直接移向位置10000，虽然并没有到达位置6000。



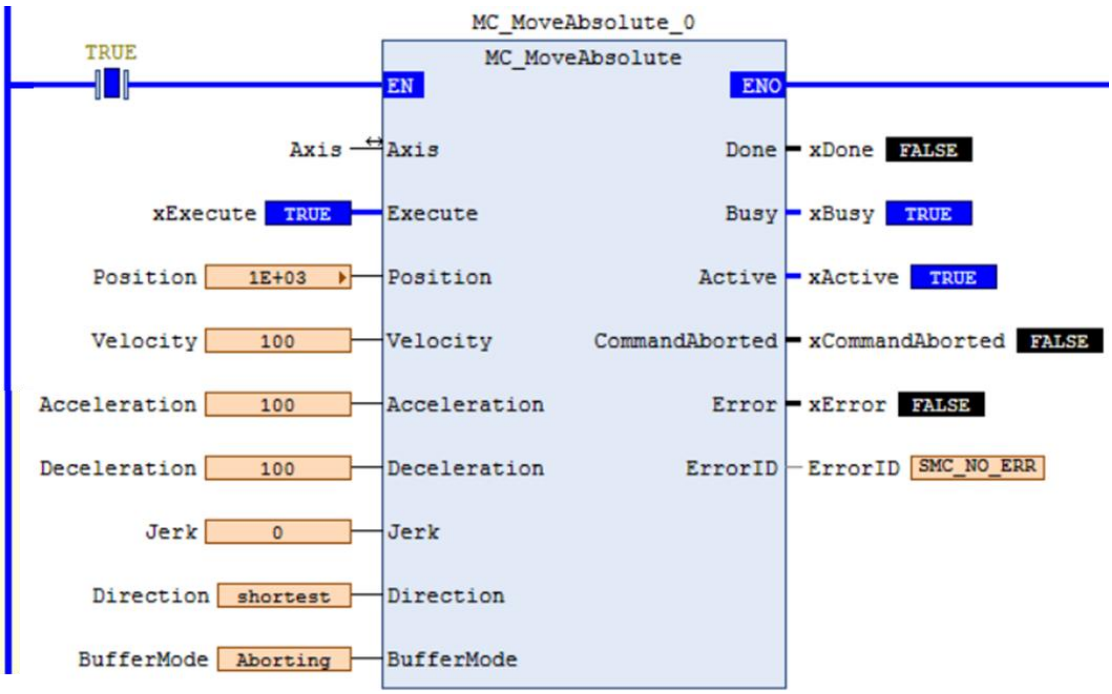
绝对位置运动

- **程序示例**

ST：当xExecute变为TRUE时，执行绝对定位指令。

```
MC_MoveAbsolute_0(
    Axis:= Axis,
    Execute TRUE := xExecute TRUE,
    Position 1E+03 := Position 1E+03,
    Velocity 10 := Velocity 10,
    Acceleration 10 := Acceleration 10,
    Deceleration 10 := Deceleration 10,
    Jerk 0 := Jerk 0,
    Direction shortest := Direction shortest,
    BufferMode Aborting := BufferMode Aborting ,
    Done FALSE => xDone FALSE,
    Busy TRUE => xBusy TRUE,
    Active TRUE => xActive TRUE,
    CommandAborted FALSE => xCommandAborted FALSE,
    Error FALSE => xError FALSE,
    ErrorID SMC_NO_ERR => ErrorID SMC_NO_ERR );RETURN
```

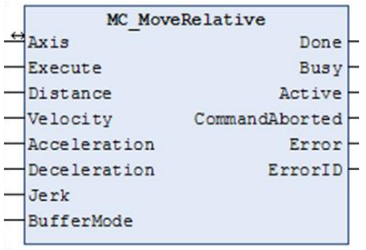
LD：当xExecute变为TRUE时，执行绝对定位指令。



4.2.4 MC_MoveRelative

本指令用于实现将控制轴按照设定参数，运动一段相对距离。

- **指令格式**

指令	名称	FB/FC	LD表现	ST表现	库
MC_Move Relative	相对定位	FB		MC_MoveRelative(Axis：=， Execute：=， Distance：=， Velocity：=， Acceleration：=， Deceleration：=， Jerk：=， BufferMode：=， Done=>， Busy=>， Active=>， CommandAborted=>， Error=>， ErrorID=>);	SM3_Basic

- **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	初始值	描述
Axis	轴	AXIS_REF_SM3	-	-	指定轴

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Execute	启动	BOOL	TRUE-FALSE	FALSE	上升沿触发指令运动。
Distance	距离	LREAL	负数，0，正数	0	目标位置与当前位置的相对距离[u]（负或正）。
Velocity	目标速度	LREAL	0，正数	0	速度最大值（总是为正） [u/s]
Acceleration	目标加速度	LREAL	0，正数	0	加速度值（总是为正）[u/s2]
Deceleration	目标减速度	LREAL	0，正数	0	减速度值（总是为正）[u/s2]

Jerk	目标加加速度	LREAL	0， 正数	0	加加速度值（总是为正）[u/s3]
Direction	方向	MC_DIRECTION	-1,1	Positive	1，Positive，正； -1，Negative，负
BufferMode	缓存模式	MC_BUFFER_MODE	0-5	0	指定多重启动运动指令时的动作。 0： mcAborting，中断； 1： mcBuffered，等待； 2： mcBlendingLow，以低速合并； 3： mcBlendingPrevious，以前一个速度合并； 4： mcBlendingNext，以后一个速度合并； 5： mcBlendingHigh，以高速合并

输出变量

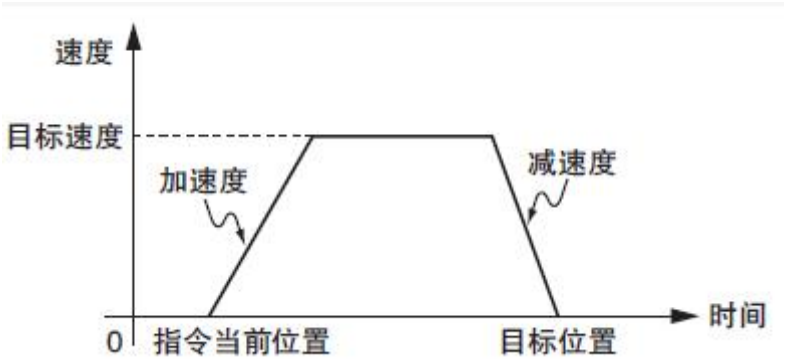
输出变量	名称	数据类型	有效范围	初始值	描述
Done	执行完成	BOOL	TRUE-FALSE	FALSE	指令执行完成时为TRUE
Busy	执行中	BOOL	TRUE-FALSE	FALSE	功能块的处理没有完成为TRUE
Active		BOOL			该轴在工作
CommandAborted	指令被中断	BOOL	TRUE-FALSE	FALSE	指令执行被中断时为TRUE
Error	错误	BOOL	TRUE-FALSE	FALSE	发生异常时为TRUE
ErrorID	错误代码	SMC_ERROR	–	0	正常时数值为0，发生异常时，输出报错代码

	布尔	位串				整数								实数		时刻、持续时间 日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Axis	AXIS_REF_SM3																			
Execute	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Distance	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–

Velocity	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–
Acceleration	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–
Deceleration	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–
Jerk	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–
Direction	MC_DIRECTION																			
BufferMode	MC_BUFFER_MODE																			
Done	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Busy	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Active	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
CommandAborted	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Error	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
ErrorID	SMC_ERROR																			

- **功能说明：**

  - 以当前位置为原点做相对运动，终点坐标即为起点到终点的距离。
  - 当速度曲线是梯形曲线时，指令执行时的“速度-时间”如下图所示。

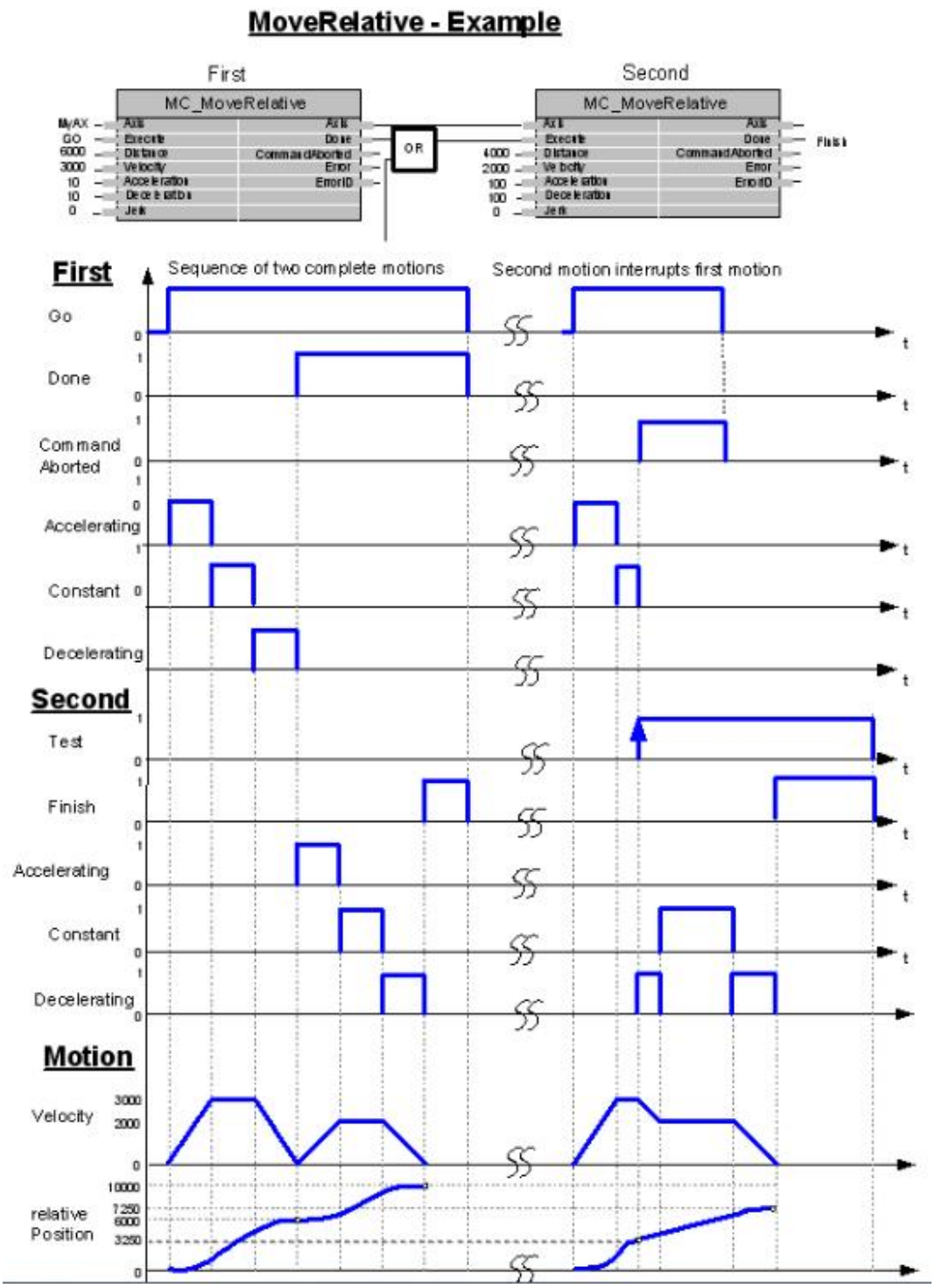


- **例 MC_MoveRelative：**

下图是两个“MC_MoveRelative”功能块的实例“First”和“Second”连接的例程。

1) 时序图的前段为“Second”在“First”之后调用的情况。当“First”达到要求的位置6000（速度为0）时，输出Done将引发“Second”向位置10000移动。

2) 时序图的后段为“First”还在执行时“Second”被启动的情况。这时，“First”在速度恒定期间被Test信号终止。将距离4000加上实际位置3250后，“Second”功能块将把轴移向得到的新位置7250。



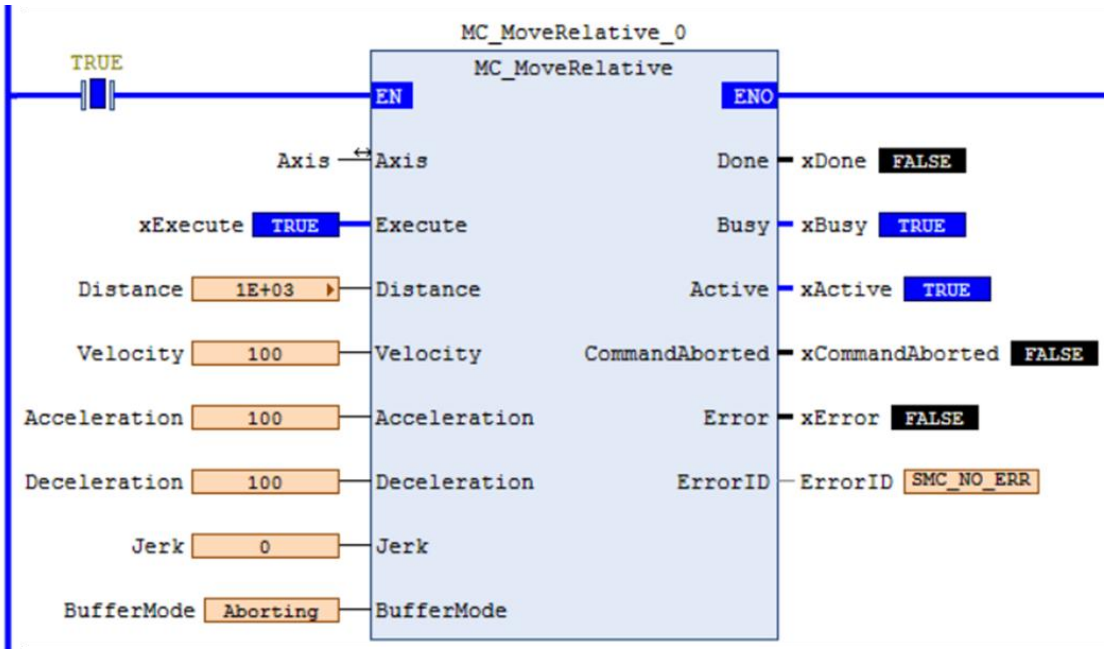
相对位置运动

- **程序示例**

ST：当xExecute变为TRUE时，执行相对定位指令。

```
MC_MoveRelative_0(
    Axis:= Axis,
    Execute TRUE := xExecute TRUE,
    Distance 1E+03 := Distance 1E+03 ,
    Velocity 100 := Velocity 100 ,
    Acceleration 100 := Acceleration 100 ,
    Deceleration 100 := Deceleration 100 ,
    Jerk 0 := Jerk 0 ,
    BufferMode Aborting := BufferMode Aborting ,
    Done FALSE => xDone FALSE ,
    Busy TRUE => xBusy TRUE ,
    Active TRUE => xActive TRUE ,
    CommandAborted FALSE => xCommandAborted FALSE ,
    Error FALSE => xError FALSE ,
    ErrorID SMC_NO_ERR => ErrorID SMC_NO_ERR );
```

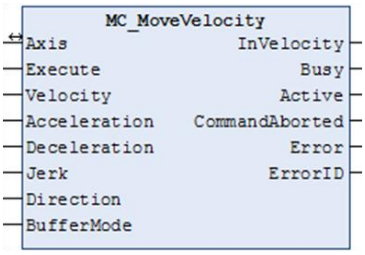
LD：当xExecute变为TRUE时，执行相对定位指令。



4.2.5 MC_MoveVelocity

本指令用于实现将控制轴按照设定参数，保持恒速运动。

- **指令格式**

指令	名称	FB/FC	LD表现	ST表现	库
MC_MoveVelocity	轴恒速运动	FB		MC_MoveVelocity(Axis：=， Execute：=， Velocity：=， Acceleration：=， Deceleration：=， Jerk：=， Direction：=， BufferMode：=， InVelocity=>， Busy=>， Active=>， CommandAborted=>， Error=>， ErrorID=>);	SM3_Basic

- **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	初始值	描述
Axis	轴	AXIS_REF_SM3	-	-	指定轴

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Execute	启动	BOOL	TRUE-FALSE	FALSE	上升沿触发指令运动。
Velocity	目标速度	LREAL	负数，0，正数	0	速度最大值（总是为正） [u/s]
Acceleration	目标加速度	LREAL	0，正数	0	加速度值（总是为正）[u/s2]
Deceleration	目标减速度	LREAL	0，正数	0	减速度值（总是为正）[u/s2]
Jerk	目标加加速度	LREAL	0，正数	0	加加速度值（总是为正）[u/s3]

Direction	方向	MC_DIRECTION	-1,1	Positive	1，Positive，正； -1，Negative，负
BufferMode	缓存模式	MC_BUFFER_MODE	0-5	0	指定多重启动运动指令时的动作。 0： mcAborting，中断； 1： mcBuffered，等待； 2： mcBlendingLow，以低速合并； 3： mcBlendingPrevious，以前一个速度合并； 4： mcBlendingNext，以后一个速度合并； 5： mcBlendingHigh，以高速合并

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
InVelocity	速度到达	BOOL	TRUE-FALSE	FALSE	设定速度到达之后则为 TRUE
Busy	执行中	BOOL	TRUE-FALSE	FALSE	当功能块执行还没结束时为 TRUE。
Active		BOOL			该轴在工作
CommandAborted	指令被中断	BOOL	TRUE-FALSE	FALSE	当功能块执行被中断时为 TRUE。
Error	错误	BOOL	TRUE-FALSE	FALSE	发生异常时为 TRUE
ErrorID	错误代码	SMC_ERROR	-	0	正常时数值为 0，发生异常时，输出报错代码。

	布尔	位串				整数								实数		时刻、持续时间 日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Axis	AXIS_REF_SM3																			
Execute	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Velocity	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–
Acceleration	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–
Deceleration	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–

Jerk	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–
Direction	MC_DIRECTION																			
BufferMode	MC_BUFFER_MODE																			
InVelocity	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Busy	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Active	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
CommandAborted	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Error	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
ErrorID	SMC_ERROR																			

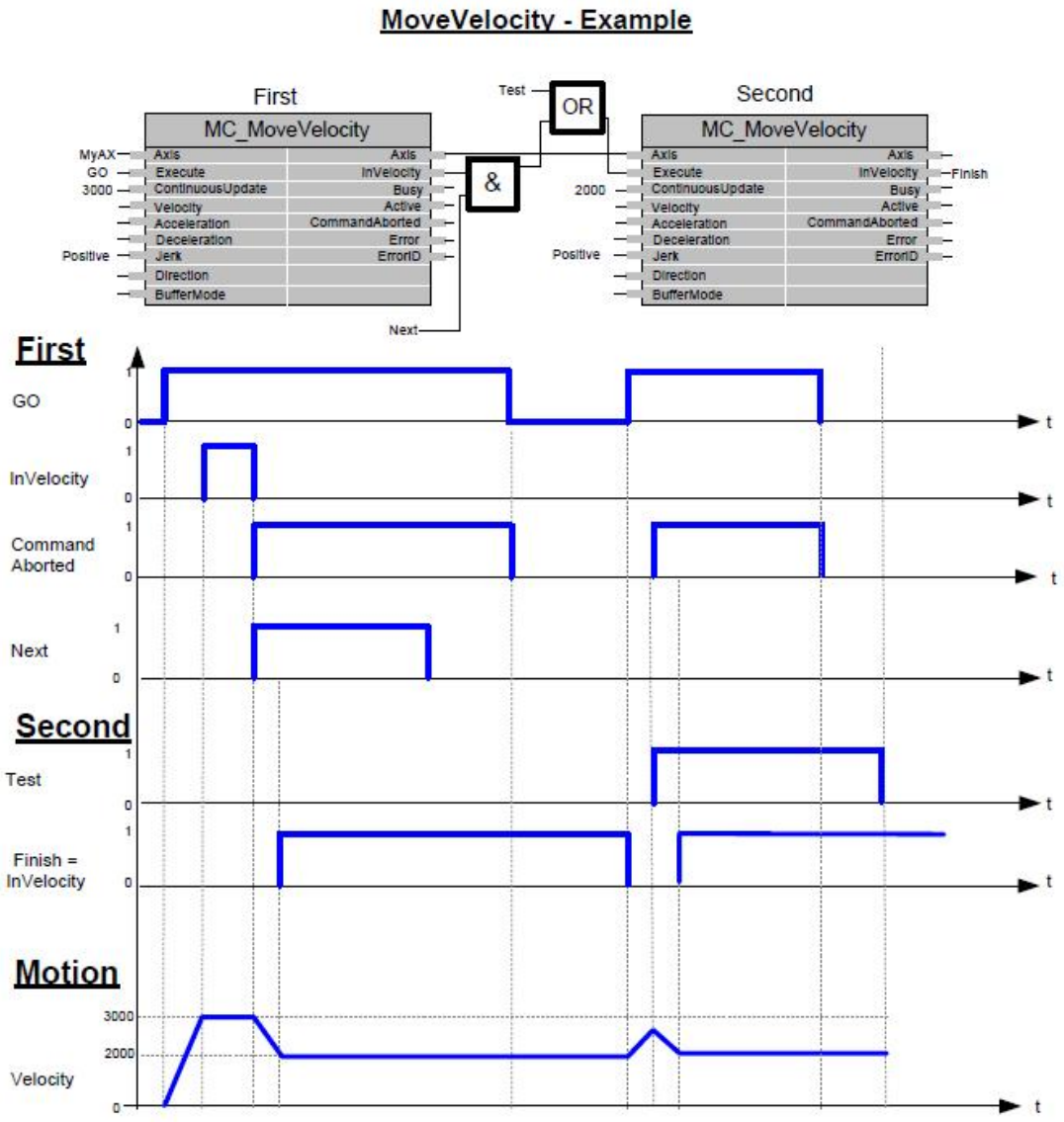
- **功能说明：**
  - Execute的上升沿启动恒速运动，而后，只能由其他的命令来中止恒速运动。
  - 当恒速运动到达设定速度后，InVelocity信号将变为TRUE。
  - 当恒速运动被其它运动中断之后，InVelocity信号将被重置为FALSE。
  - 负速度×负方向=正速度。
  - 控制电机以指定的加速度加速到最大速度，然后以该最大速度一直运行，直到调用停止指令或者其他的中断指令中断该指令。

- **例 MC_MoveVelocity**

下图以梯形图的形式，展示了“MC_MoveVelocity”功能块的两个实例“First”和“Second”连接的例子：

1) 时间图的左部画出了“Second”在“First”之后调用的情况。当“First”达到设置速度3000时，第一个指令输出了InVelocity信号，与下一个指令进行“与”运算，触发了第二个指令以2000的速度运动。

2) 时间图的右部画出了当“First”还在执行时“Second”被启动的情况。这时，“First”的运动被中断，并被在“First”速度恒定期间传递的Test信号终止。虽然第一个指令仍在向着3000的速度加速过程中，但由于中途执行了第二个实例，第一个的加速被中断并中止。然后第二个实例将会减速到2000，之后第二个实例的InVelocity置TRUE。



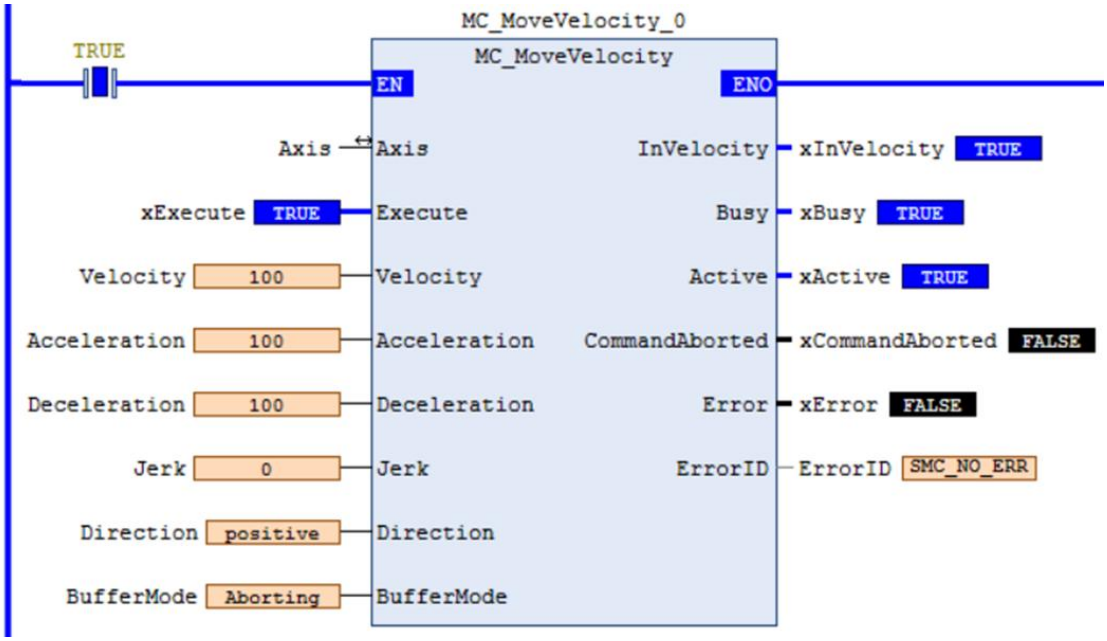
恒速运动

- **程序示例**

ST：当xExecute变为TRUE时，执行恒速运动指令。

```
MC_MoveVelocity_0(
    Axis:= Axis,
    Execute TRUE := xExecute TRUE,
    Velocity 100 := Velocity 100,
    Acceleration 100 := Acceleration 100,
    Deceleration 100 := Deceleration 100,
    Jerk 0 := Jerk 0,
    Direction positive := Direction positive,
    BufferMode Aborting := BufferMode Aborting ,
    InVelocity TRUE => InVelocity TRUE,
    Busy TRUE => xBusy TRUE,
    Active TRUE => xActive TRUE,
    CommandAborted FALSE => xCommandAborted FALSE,
    Error FALSE => xError FALSE,
    ErrorID SMC_NO_ERR => ErrorID SMC_NO_ERR );
```

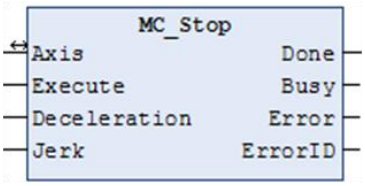
LD：当xExecute变为TRUE时，执行恒速运动指令。



4.2.6 MC_Stop

本指令用于中断轴正在进行的运动，对轴进行减速停止。

- **指令格式**

指令	名称	FB/FC	LD表现	ST表现	库
MC_Stop	停止运动	FB		MC_Stop(Axis：=， Execute：=， Deceleration：=， Jerk：=， Done=>， Busy=>， Error=>， ErrorID=>);	SM3_Basic

- **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	初始值	描述
Axis	轴	AXIS_REF_SM3	-	-	指定轴

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Execute	启动	BOOL	TRUE-FALSE	FALSE	上升沿触发指令运动。
Deceleration	目标减速度	LREAL	0，正数	0	减速度值（总是为正）[u/s2]
Jerk	目标加加速度	LREAL	0，正数	0	加加速度值（总是为正）[u/s3]

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
Done	执行完成	BOOL	TRUE-FALSE	FALSE	如果指令执行完成则为TRUE。
Busy	执行中	BOOL	TRUE-FALSE	FALSE	当功能块执行还没结束时为TRUE。
Error	错误	BOOL	TRUE-FALSE	FALSE	功能块执行错误
ErrorID	错误代码	SMC_ERROR	-	0	错误指示，见SMC_Error

	布尔	位串				整数								实数		时刻、持续时间 日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Axis	AXIS_REF_SM3																			
Execute	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Deceleration	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–
Jerk	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–
Done	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Busy	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Error	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
ErrorID	SMC_ERROR																			

- **功能说明：**

  - Execute的上升沿处开始停止运动，控制轴减速，由当前速度变为“0”。本指令运行时，不能被其它任何指令终止。因启动MC_Stop(强制停止)指令，处于运行中的指令执行CommandAborted(执行中断)。

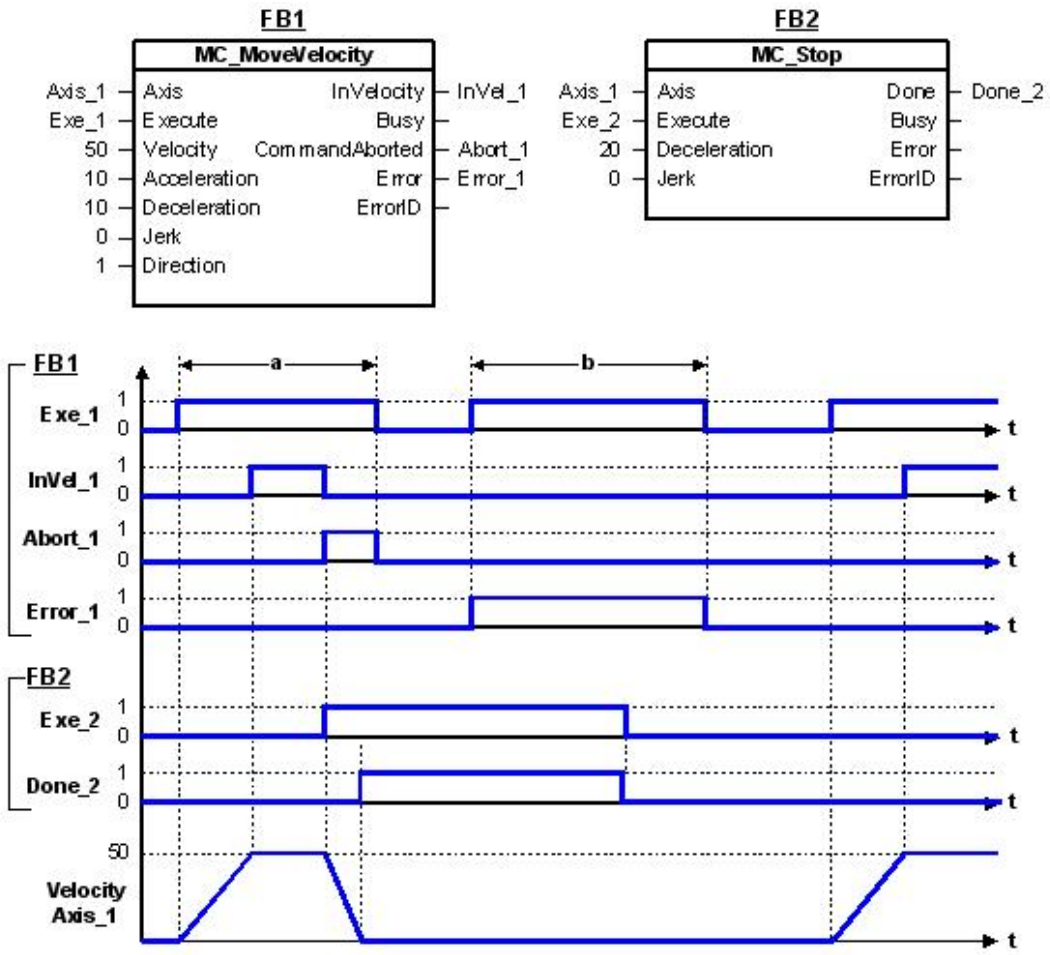
- **例 MC_Stop**

  下图为 FB2（MC_Stop）实例与 FB1（MC_MoveVelocity）实例相结合时的使用方法。

  1) 由图中时序图可以看到，在FB1的运动过程中，启动了FB2，轴的速度会呈现倾斜向下的趋势，直到为0。

  2) 只要FB2的Execute信号为TRUE，对应的轴就不会执行任何运动命令。

  3) 在FB2启动后，FB1的Error信号置为TRUE，只是当前轴处于停止状态。



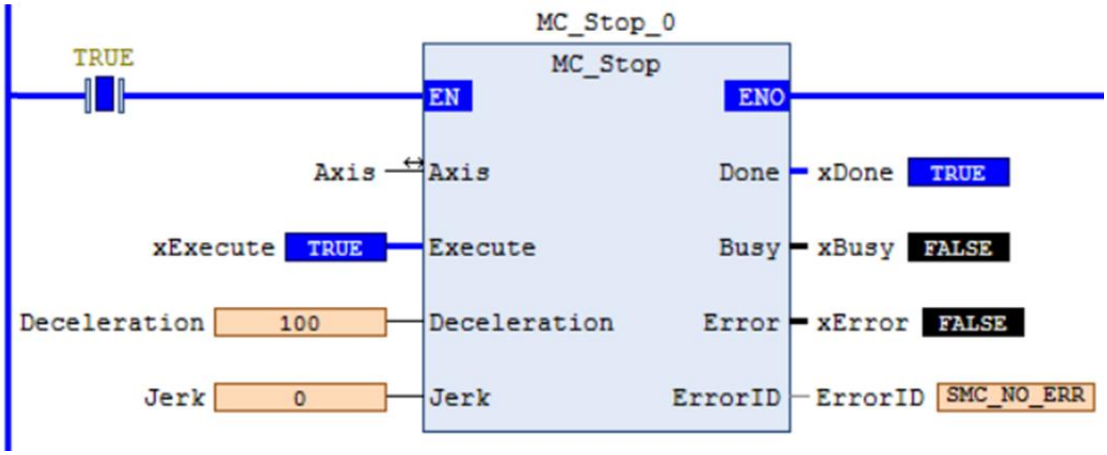
停止运动

- **程序示例**

ST：当xExecute变为TRUE时，对轴进行减速停止。

```
MC_Stop_0(
    Axis:= Axis,
    Execute TRUE := xExecute TRUE,
    Deceleration 100 := Deceleration 100,
    Jerk 0 := Jerk 0,
    Done TRUE => xDone TRUE,
    Busy FALSE => xBusy FALSE,
    Error FALSE => xError FALSE,
    ErrorID SMC_NO_ERR => ErrorID SMC_NO_ERR );
```

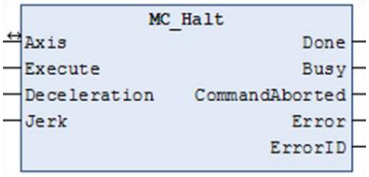
LD：当xExecute变为TRUE时，对轴进行减速停止。



4.2.7 MC_Halt

本指令用于减速停止轴正在执行的运动，停止的运动可恢复执行未完成部分。

- **指令格式**

指令	名称	FB/FC	LD表现	ST表现	库
MC_Halt	暂停运动	FB		MC_Halt(Axis：=， Execute：=， Deceleration：=， Jerk：=， Done=>， Busy=>， CommandAborted=>， Error=>， ErrorID=>);	SM3_Basic

- **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	初始值	描述
Axis	轴	AXIS_REF_SM3	–	–	指定轴

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Execute	启动	BOOL	TRUE-FALSE	FALSE	上升沿触发指令运动。
Deceleration	目标减速度	LREAL	0，正数	0	停止的减速度值（总是为正）
Jerk	目标减减速度	LREAL	0，正数	0	停止的减减速度值（总是为正）

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
Done	执行完成	BOOL	TRUE-FALSE	FALSE	如果指令执行完成则为 TRUE。
Busy	执行中	BOOL	TRUE-FALSE	FALSE	当功能块执行还没结束时为 TRUE。
bCommandAborted	指令被中断	BOOL	TRUE-FALSE	FALSE	如果该命令已被其他命令终止则为 TRUE。
Error	错误	BOOL	TRUE-FALSE	FALSE	功能块执行错误
ErrorID	错误代码	SMC_ERROR	–	0	错误指示，见 SMC_Error。

	布尔	位串				整数								实数		时刻、持续时间 日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Axis	AXIS_REF_SM3																			
Execute	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Deceleration	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–
Jerk	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–
Done	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Busy	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
bCommandA	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

borted																				
Error	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
ErrorID	SMC_ERROR																			

- **功能说明：**

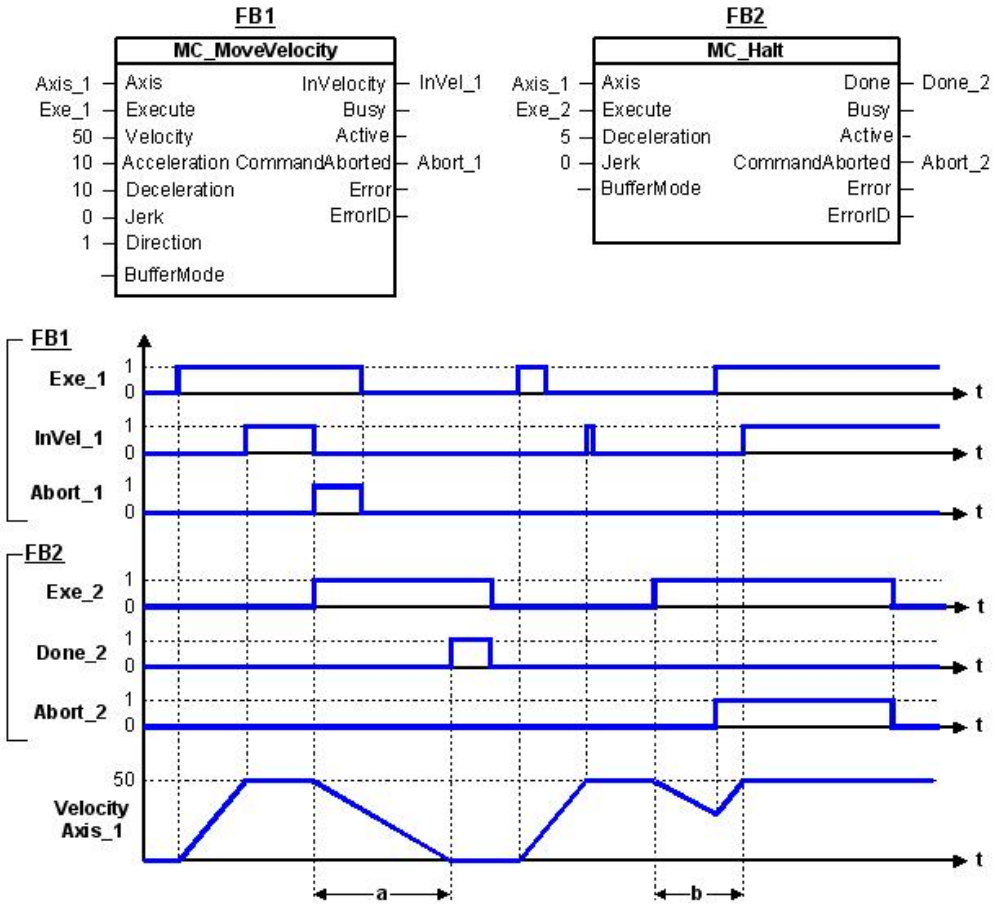
  - 对于设定轴，在Execute的上升沿处开始暂停运动。
  - 重新执行原来运动指令可以恢复原运动的执行。

例 MC_Halt

下图显示了 MC_MoveVelocity 指令正在执行运动的过程中，被 MC_Halt 指令中断的行为。

1) 与MC_Stop相比，MC_Halt只是暂停运动，运动在后续还能够继续执行直到指令完成。

2) 在被MC_Halt指令暂停后，轴可以在速度没有为0时，重新调用运动指令，进行再次加速。



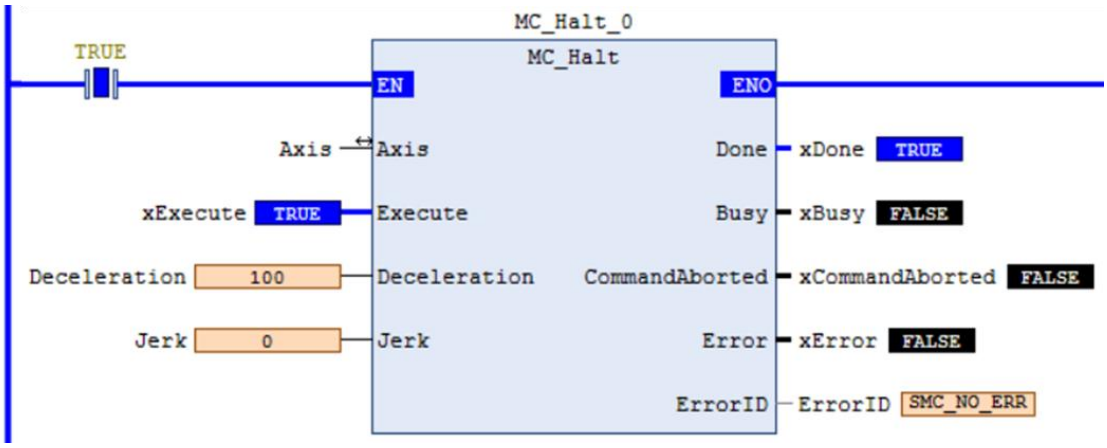
暂停运动

- **程序示例**

ST：当xExecute变为TRUE时，对轴执行减速暂停指令。

```
MC_Halt_0(
    Axis:= Axis,
    Execute TRUE := xExecute TRUE,
    Deceleration 100 := Deceleration 100,
    Jerk 0 := Jerk 0,
    Done TRUE => xDone TRUE,
    Busy FALSE => xBusy FALSE,
    CommandAborted FALSE => xCommandAborted FALSE,
    Error FALSE => xError FALSE,
    ErrorID SMC_NO_ERR => ErrorID SMC_NO_ERR );
```

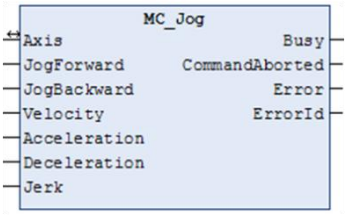
LD：当xExecute变为TRUE时，对轴执行减速暂停指令。



4.2.8 MC_Jog

本指令用于手动控制轴朝指定方向运动。

- **指令格式**

指令	名称	FB/FC	LD表现	ST表现	库
MC_Jog	点动运动	FB		MC_Jog(Axis：=， JogForward：=， JogBackward：=， Velocity：=， Acceleration：=，	SM3_Basic

				Deceleration：=， Jerk：=， Busy=>， CommandAborted=>， Error=>， ErrorId=>);	

- **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	初始值	描述
Axis	轴	AXIS_REF_SM3	-	-	指定轴

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
JogForward	向前点动	BOOL	TRUE-FALSE	FALSE	如果JogForward为TRUE，轴将按给定参数（Velocity、Acceleration、Deceleration）朝正向运动，如果JogBackward同时为TRUE，轴不动。
JogBackward	向后点动	BOOL	TRUE-FALSE	FALSE	如果JogBackward为TRUE，轴将按给定参数（Velocity、Acceleration、Deceleration）朝负向运动，如果JogForward同时为TRUE，轴不动。
Velocity	目标速度	LREAL	负数，0，正数	0	速度最大值 [u/s]
Acceleration	目标加速度	LREAL	负数，0，正数	0	加速度值[u/s2]
Deceleration	目标减速度	LREAL	负数，0，正数	0	减速度值[u/s2]
Jerk	目标加加速度	LREAL	负数，0，正数	0	加加速度值[u/s3]

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
Busy	执行中	BOOL	TRUE-FALSE	FALSE	当功能块执行还没结束时为 TRUE。
Command Aborted	指令被中断	BOOL	TRUE-FALSE	FALSE	如果该命令已被其他命令终止则为 TRUE。
Error	错误	BOOL	TRUE-FALSE	FALSE	功能块执行错误
ErrorID	错误代码	SMC_ERROR	–	0	错误指示，见 SMC_Error。

	布尔	位串				整数								实数		时刻、持续时间 日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Axis	AXIS_REF_SM3																			
JogForward	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
JogBackward	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Velocity	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–
Acceleration	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–
Deceleration	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–
Jerk	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Busy	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
CommandAborted	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Error	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
ErrorID	SMC_ERROR																			

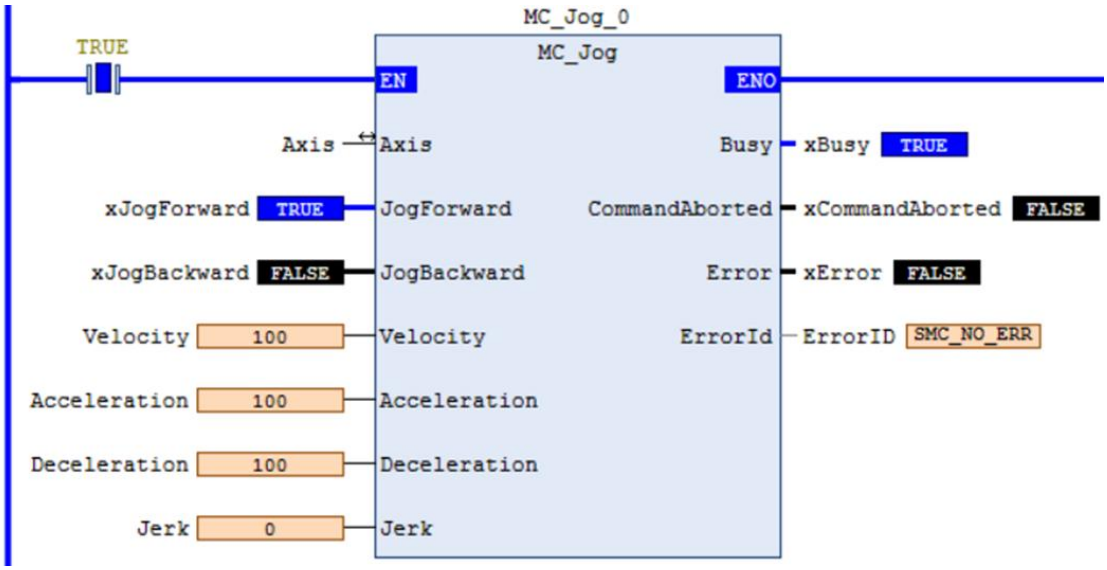
- **功能说明：**
  - 当“JogForward”或“JogBackward”为TRUE时，指定轴分别往正方向或负方向执行恒速运动。
  - 同时将JogForward和JogBackward置为TRUE，将不会有运动发生。
  - 如果MC_Jog指令的指令速度设置值超过轴参数中的点动最高速度，则以点动最高速度执行。

- **程序示例**

ST：当JogForward为TRUE时，指定轴往正方向执行恒速运动。

```
MC_Jog_0(
    Axis:= Axis,
    JogForward TRUE := xJogForward TRUE,
    JogBackward FALSE := xJogBackward FALSE,
    Velocity 100 := Velocity 100,
    Acceleration 100 := Acceleration 100,
    Deceleration 100 := Deceleration 100,
    Jerk 0 := Jerk 0,
    Busy TRUE => xBusy TRUE,
    CommandAborted FALSE => xCommandAborted FALSE,
    Error FALSE => xError FALSE,
    ErrorId SMC_NO_ERR => ErrorId SMC_NO_ERR );
```

LD：当JogForward为TRUE时，指定轴往正方向执行恒速运动。

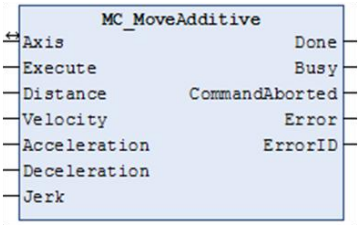


4.2.9 MC_MoveAdditive

本指令用于控制终端执行机构在运动过程中做位置叠加运动。

- **指令格式**

指令	名称	FB/F	LD表现	ST表现	库

		C			
MC_MoveAdditive	位置叠加	FB		MC_MoveAdditive(Axis：=， Execute：=， Distance：=， Velocity：ion：=， Deceleration：=， Jerk：=， Done=>， Busy=>， CommandAborted=> ， Error=>， ErrorID=>);	SM3_Basic

- **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	初始值	描述
Axis	轴	AXIS_REF_SM3	-	-	指定轴

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Execute	启动	BOOL	TRUE-FALSE	FALSE	上升沿触发指令运动。
Distance	距离	LREAL	负数，0，正数	0	运动的相对距离
Velocity	目标速度	LREAL	负数，0，正数	0	最大速度
Acceleration	目标加速度	LREAL	负数，0，正数	0	加速度值
Deceleration	目标减速度	LREAL	负数，0，正数	0	减速度值

Jerk	目标加加速度	LREAL	负数，0，正数	0	加加速度值

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
Done	执行完成	BOOL	TRUE-FALSE	FALSE	如果指令执行完成则为 TRUE。
Busy	执行中	BOOL	TRUE-FALSE	FALSE	当功能块执行还没结束时为 TRUE。
CommandAborted	指令被中断	BOOL	TRUE-FALSE	FALSE	当功能块执行被中断时为 TRUE。
Error	错误	BOOL	TRUE-FALSE	FALSE	功能块执行错误
ErrorID	错误代码	SMC_ERROR	–	0	错误指示，见 SMC_Error。

	布尔	位串				整数								实数		时刻、持续时间 日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Axis	AXIS_REF_SM3																			
Execute	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Distance	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–
Velocity	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–
Acceleration	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–
Deceleration	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–
Jerk	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–
Done	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Busy	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
CommandAborted	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Error	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
ErrorID	SMC_ERROR																			

- **功能说明：**

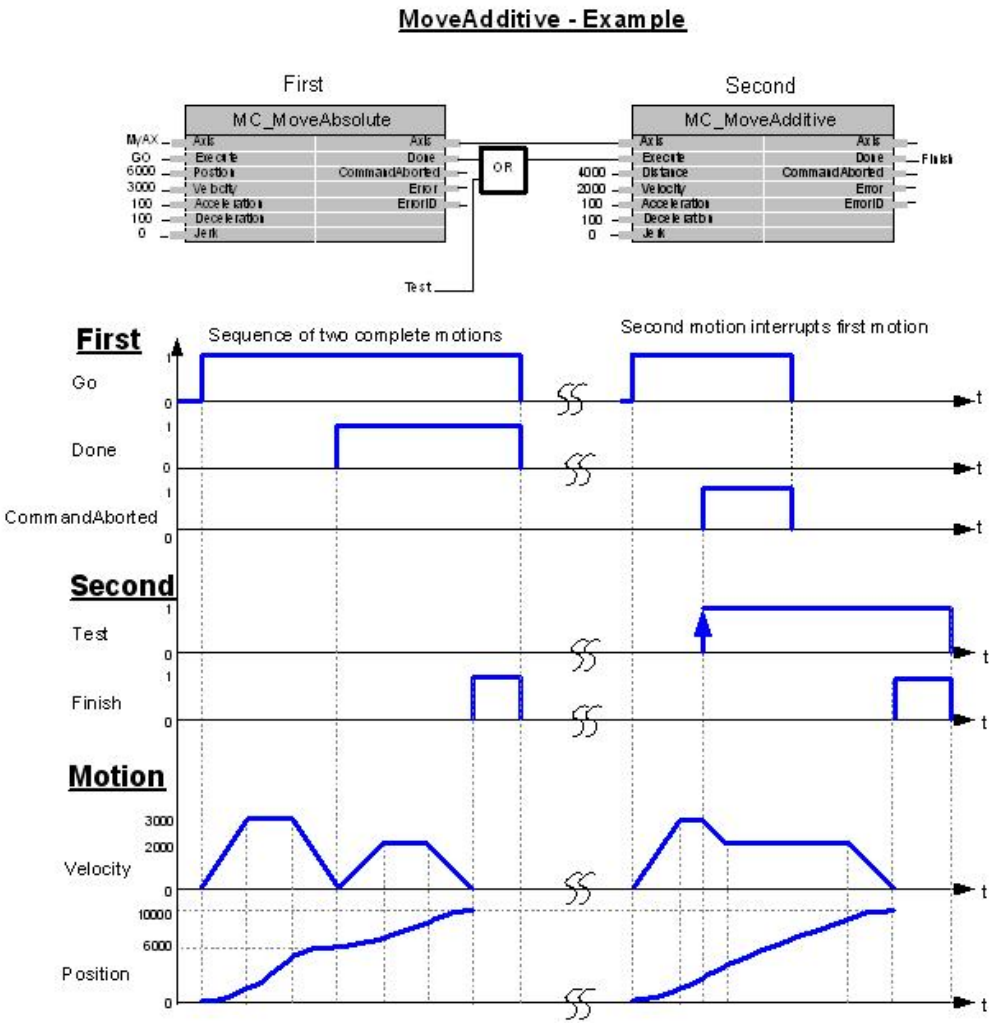
- 此指令用于控制终端执行机构按给定的速度、加速度移动一段附加的距离。
- 终端执行机构的最终位置为前一个位移指令和此指令给定的距离总和。
- 打断其它指令的运动时，轴的速度为此功能块的速度。
- 当前一个指令为速度指令时，此指令执行时会终止速度指令执行，并按给定的速度，加减速移动给定的距离后停止。

- **例 MC_MoveAdditive**

1) 下图展示了第一个实例“First”（MC_MoveAbsolute）和第二个实例“Second”（MC_MoveAdditive）组合的运动。

2) 在图的前段，Second实例在First实例之后被调用。当First实例运动到指定的位置6000，则速度为0并且Done信号为True，开始执行实例Second，使轴移动到位置10000。

3) 在图的后段，在实例First执行的过程中，启动Second实例，First实例的运动此刻正处在4000的位置，此时First的运动被打断，开始减速，当到达位置6000时，以Second 指定的速度，将轴移动到最终的位置10000。



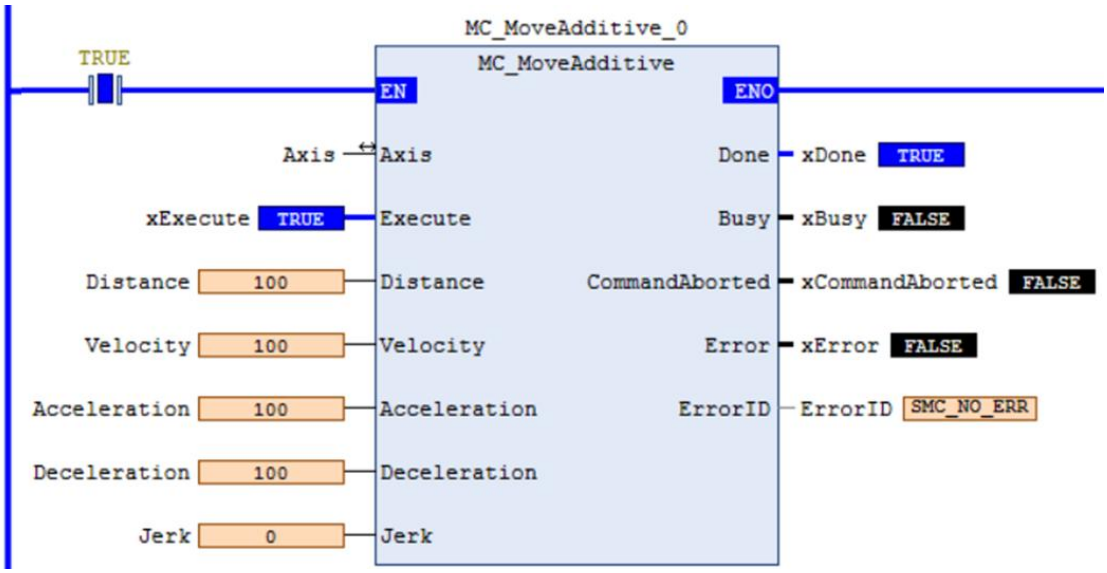
位置叠加运动

- **程序示例**

ST：当xExecute变为TRUE时，执行位置叠加运动指令。

```
MC_MoveAdditive_0(
    Axis:= Axis,
    Execute TRUE := xExecute TRUE,
    Distance 100 := Distance 100,
    Velocity 100 := Velocity 100 ,
    Acceleration 100 := Acceleration 100,
    Deceleration 100 := Deceleration 100,
    Jerk 0 := Jerk 0,
    Done FALSE => xDone FALSE,
    Busy TRUE => xBusy TRUE,
    CommandAborted FALSE => xCommandAborted FALSE,
    Error FALSE => xError FALSE,
    ErrorID SMC_NO_ERR => ErrorID SMC_NO_ERR );
```

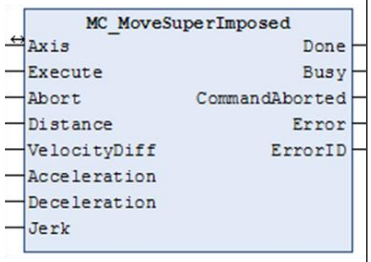
LD：当xExecute变为TRUE时，执行位置叠加运动指令。



4.2.10 MC_MoveSuperImposed

本指令用于控制终端执行机构在运动过程中叠加运动。

- **指令格式**

指令	名称	FB/FC	LD表现	ST表现	库
MC_MoveSuperImposed	位置速度叠加	FB		MC_MoveSuperImposed(Axis：=， Execute：=， Abort：=， Distance：=， VelocityDiff：=， Acceleration：=， Deceleration：=， Jerk：=， Done=>， Busy=>， CommandAborted=>， Error=>， ErrorID=>);	SM3_Basic

- **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	初始值	描述
Axis	轴	AXIS_REF_SM3	-	-	指定轴

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Execute	启动	BOOL	TRUE-FALSE	FALSE	上升沿触发指令运动。
Abort	终止	BOOL	TRUE-FALSE	FALSE	终止运动，输出参数复位
Distance	距离	LREAL	负数，0，正数	0	运动的相对距离
VelocityDiff	叠加速度	LREAL	负数，0，正数	0	叠加的速度
Acceleration	目标加速度	LREAL	负数，0，正数	0	加速度值
Deceleration	目标减速度	LREAL	负数，0，正数	0	减速度值
Jerk	目标加加速度	LREAL	负数，0，正数	0	加加速度值

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
Done	执行完成	BOOL	TRUE-FALSE	FALSE	如果指令执行完成则为 TRUE。
Busy	执行中	BOOL	TRUE-FALSE	FALSE	当功能块执行还没结束时为 TRUE。
CommandAborted	指令被中断	BOOL	TRUE-FALSE	FALSE	当功能块执行被中断时为 TRUE。

Error	错误	BOOL	TRUE-FALSE	FALSE	功能块执行错误
ErrorID	错误代码	SMC_ERROR	-	0	错误指示，见 SMC_Error。

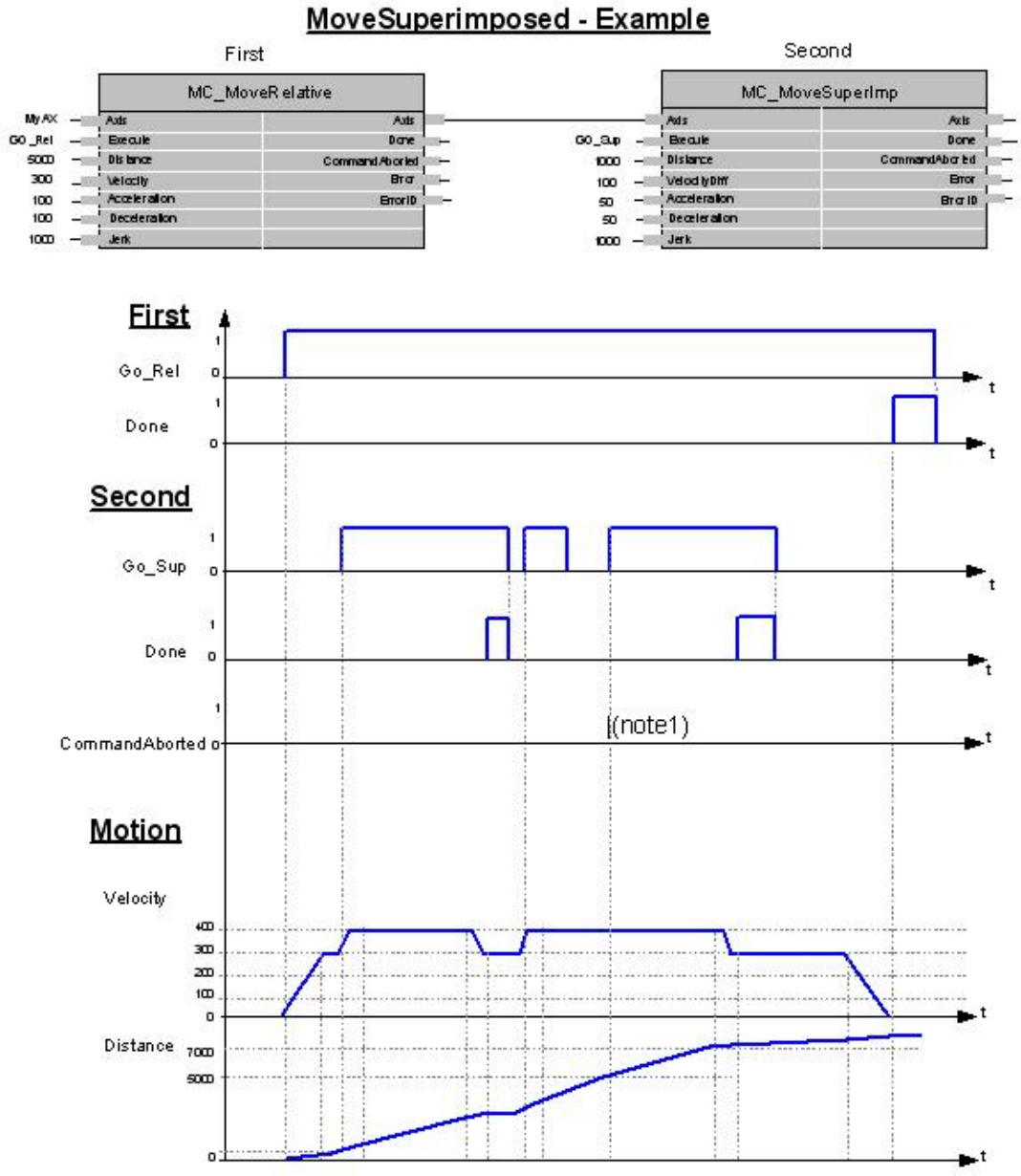
	布尔	位串				整数								实数		时刻、持续时间 日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Axis	AXIS_REF_SM3																			
Execute	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Abort	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Distance	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-
VelocityDiff	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-
Acceleration	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-
Deceleration	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-
Jerk	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-
Done	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Busy	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
CommandAborted	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Error	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
ErrorID	SMC_ERROR																			

- **功能说明：**
  - 此指令用于控制终端执行机构按给定的速度、加速度移动一段叠加的距离。
  - 终端执行机构的最终位置为前一个位移指令和此指令给定的距离总和。
  - 第一段运动未完成时，速度为两段运动之和；第一段运动完成时，速度为此功能块的速度。
  - 当前一个指令为速度指令时，此指令执行时会终止速度指令执行，并按给定的速度，加减速度移动给定的距离后停止。
- **例 MC_MoveSuperImposed**

1) 下图展示了MC_MoveRelative功能块的实例“First”和MC_MoveSuperImposed功能块的实例“Second”连接的例子：

2) 可以看到CommandAborted信号的曲线一直都是零，这是因为“Second”实例所作用的，是和“First”命令在同一个实例上的。

3) 最终运动结束的位置是在7000-8000之间，具体取决于“Second”实例的实际启停时间。



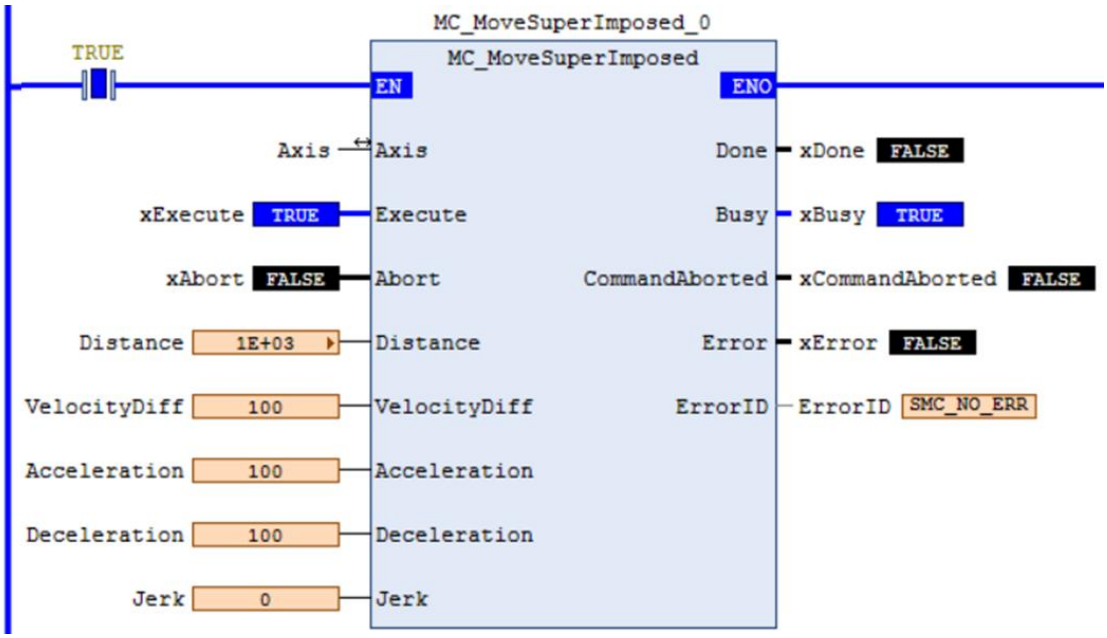
位置速度叠加运动

- **程序示例**

ST：当xExecute变为TRUE时，执行位置速度叠加运动指令。

```
MC_MoveSuperImposed_0(
    Axis:= Axis,
    Execute TRUE := xExecute TRUE,
    Abort FALSE := xAbort FALSE,
    Distance 1E+03 := Distance 1E+03,
    VelocityDiff 100 := VelocityDiff 100,
    Acceleration 100 := Acceleration 100,
    Deceleration 100 := Deceleration 100,
    Jerk 0 := Jerk 0,
    Done FALSE => xDone FALSE,
    Busy TRUE => xBusy TRUE,
    CommandAborted FALSE => xCommandAborted FALSE,
    Error FALSE => xError FALSE,
    ErrorID SMC_NO_ERR => ErrorID SMC_NO_ERR);
```

LD： 当xExecute变为TRUE时，执行位置速度叠加运动指令。

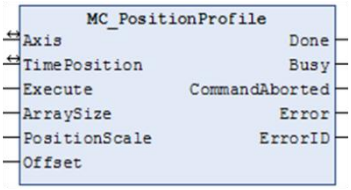


4.2.11 MC_PositionProfile

本指令使用户可以自己规划“时间-位置”数据表，控制器将按照规划的数据完成运动。

- **指令格式**

指令	名称	FB/FC	LD表现	ST表现	库

MC_PositionProfile	时间-位置规划	FB		MC_PositionProfile(Axis：=， TimePosition：=， Execute：=， ArraySize：=， PositionScale：=， Offset：=， Done=>， Busy=>， CommandAborted=>， Error=>， ErrorID=>);	SM3_Basic

- **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	初始值	描述
Axis	轴	AXIS_REF_SM3	-	-	指定轴
TimePosition	数据表	MC_TP_REF	-	-	用户规划的时间-位置数据表

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Execute	启动	BOOL	TRUE-FALSE	FALSE	上升沿触发指令运动。
ArraySize	数据点数	INT	0 正数	0	数据表大小
PositionScale	比例	LREAL	负数，0， 正数	1	整个数据表位置的比例系数
Offset	偏移	LREAL	负数，0， 正数	0	位置偏置

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
Done	执行完成	BOOL	TRUE-FALSE	FALSE	如果指令执行完成则为 TRUE。
Busy	执行中	BOOL	TRUE-FALSE	FALSE	当功能块执行还没结束时为 TRUE。
CommandAborted	指令被中断	BOOL	TRUE-FALSE	FALSE	当功能块执行被中断时为 TRUE。
Error	错误	BOOL	TRUE-FALSE	FALSE	功能块执行错误
ErrorID	错误代码	SMC_ERROR	–	0	错误指示，见 SMC_Error。

	布尔	位串				整数								实数		时刻、持续时间 日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Axis	AXIS_REF_SM3																			
TimePosition	MC_TP_REF																			
Execute	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
ArraySize	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–
PositionScale	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–
Offset	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–
Done	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Busy	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
CommandAborted	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Error	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
ErrorID	SMC_ERROR																			

- **功能说明：**
  - 用户可以自己规划“时间-位置”数据表，控制器将按照规划的数据完成运动。
  - PVT运动是指通过定义各个时刻点轴达到的位置、速度、加速度等参数来规划一个运动。

➢ 控制器支持的PVT运动有三种，如下表所示。

PVT 运动指令

指令名	功能说明
MC_PositionProfile	通过定义时间-位置表，规划轴的运动
MC_VelocityProfile	通过定义时间-速度表，规划轴的运动
MC_AccelerationProfile	通过定义时间-加速度表，规划轴的运动

MC_PositionProfile 运动所用到的数据结构定义如下：

1) “时间-位置”数据表的结构体：

```
TYPE MC_TP_TABLE
STRUCT
Number_of_pairs ： INT；        //位置-时间对的个数
IsAbsolute ： BOOL；            //位置值绝对相对选择
MC_TP_Array ： ARRAY [1..N] of MC_TP；      //位置-时间数据
END_STRUCT
END_TYPE
```

2) “时间-位置”数据的结构体：

```
TYPE MC_TP
STRUCT
delta_time ： TIME；      //位置-时间数据之时间
position ： REAL：        //位置-时间数据之位置（绝对值或相对值）
END_STRUCT
END_TYPE
```

- **例 MC_PositionProfile**

1) MC_TP_REF支持一个特殊的数据类型。下图中的代码给出的是这个数据结构的一个示例。

2) 一个时间/位置的内容可以被表述为 DeltaTime/Pos，这里的DeltaTime是指连续两点间的时间差：

```
//PT运动参数的赋值
    arPT.IsAbsolute:=FALSE ;     //相对位置
    arPT.MC_TP_Array:=arNUM ;    //PT点的数据
    arPT.Number_of_pairs:=4 ;    //PT点的个数

//PT点的数据赋值
    arNUM[1].delta_time:=T#1S;   //第一个PT点的时间数据
    arNUM[1].position:=10000;    //第一个PT点的位置数据
    arNUM[2].delta_time:=T#2S;
    arNUM[2].position:=40000;
    arNUM[3].delta_time:=T#1S;
    arNUM[3].position:=-10000;
    arNUM[4].delta_time:=T#0.5S;
    arNUM[4].position:=-5000;
```

程序 PT 的数据

本例实现了 AXIS_0 的 PT 运动，分 4 个阶段，各阶段之间运动连续：

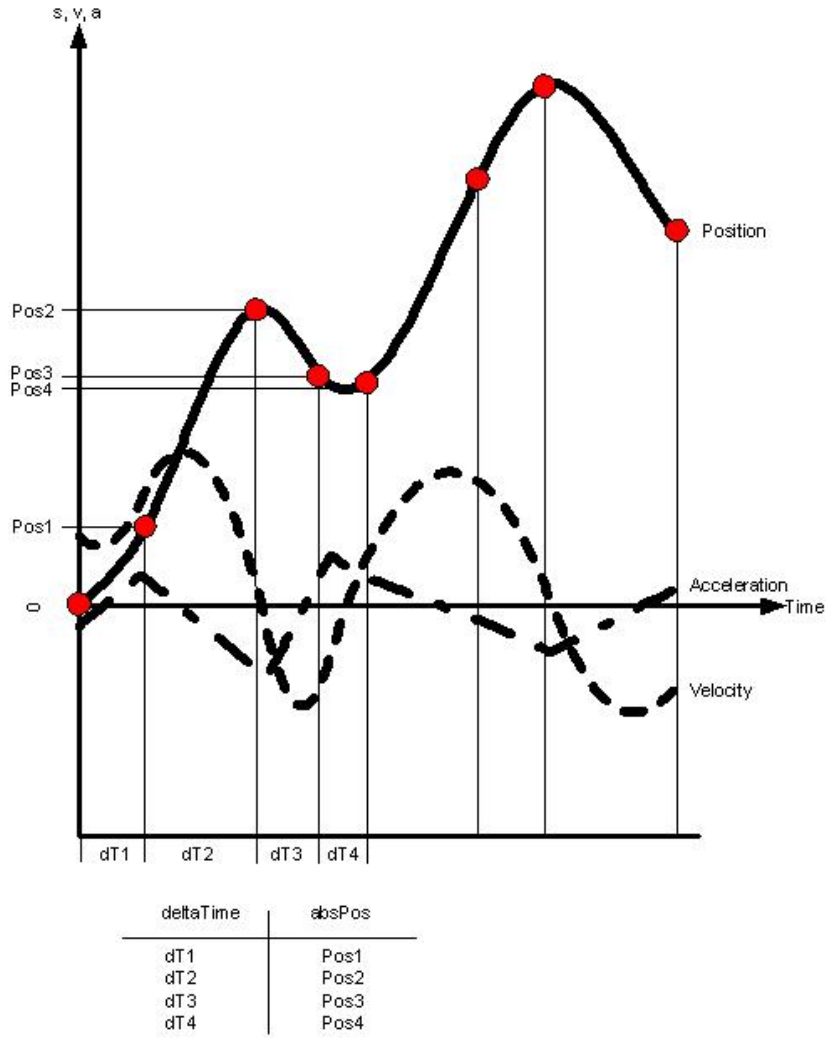
第一阶段：1 秒，运动 10000，

第二阶段：2 秒，运动 40000，

第三阶段：1 秒，运动-10000，

第四阶段：0.5 秒，运动-5000。

例程“PT”的运行结果如下图所示。



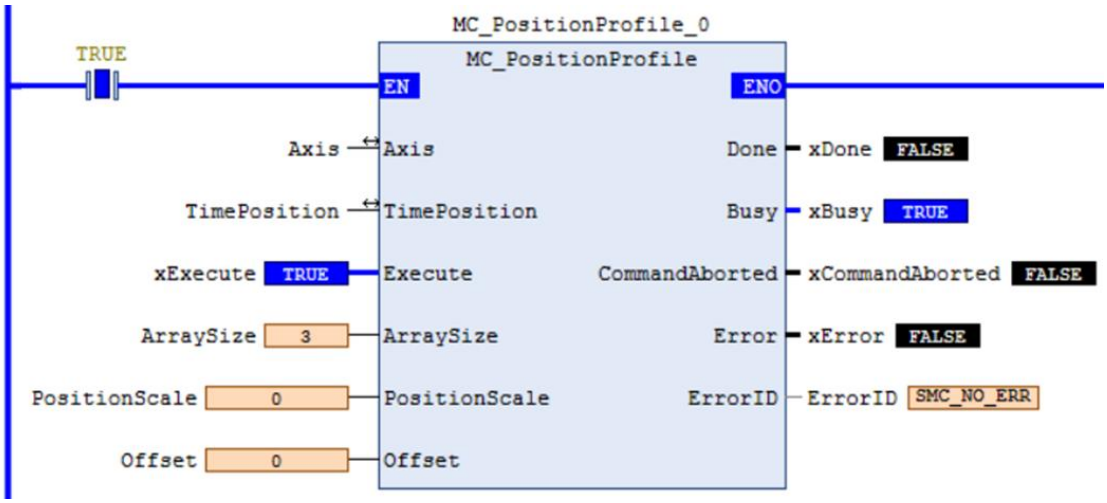
PT 运行结果

- **程序示例**

ST：当xExecute变为TRUE时，执行时间-位置规划运动指令。

```
MC_PositionProfile_0(
    Axis:=Axis ,
    TimePosition:= TimePosition,
    Execute TRUE := xExecute TRUE,
    ArraySize 3 := ArraySize 3,
    PositionScale 0 := PositionScale 0,
    Offset 0 := Offset 0,
    Done FALSE => xDone FALSE,
    Busy TRUE => xBusy TRUE,
    CommandAborted FALSE => xCommandAborted FALSE,
    Error FALSE => xError FALSE,
    ErrorID SMC_NO_ERR => ErrorID SMC_NO_ERR );
```

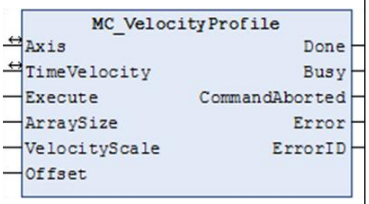
LD：当xExecute变为TRUE时，执行时间-位置规划运动指令。



4.2.12 MC_VelocityProfile

本指令与 MC_PositionProfile 指令相似，MC_VelocityProfile 通过定义“时间-速度”数据来规划运动。

- **指令格式**

指令	名称	FB/FC	LD表现	ST表现	库
MC_VelocityProfile	时间-速度规划	FB		MC_VelocityProfile(Axis：=， TimeVelocity：=， Execute：=， ArraySize：=， VelocityScale：=， Offset：=， Done=>， Busy=>， CommandAborted=>， Error=>， ErrorID=>);	SM3_Basic

- **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	初始值	描述
Axis	轴	AXIS_REF_SM3	-	-	指定轴
TimeVelocity	数据表	MC_TV_REF	-	-	用户规划的时间-速度数据表

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Execute	启动	BOOL	TRUE-FALSE	FALSE	上升沿触发指令运动。
ArraySize	数据点数	INT	0 正数	0	数据表大小
VelocityScale	比例	LREAL	负数，0，正数	1	比例系数
Offset	偏移	LREAL	负数，0，正数	0	位置偏置

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
Done	执行完成	BOOL	TRUE-FALSE	FALSE	如果指令执行完成则为 TRUE。
Busy	执行中	BOOL	TRUE-FALSE	FALSE	当功能块执行还没结束时为 TRUE。
CommandAborted	指令被中断	BOOL	TRUE-FALSE	FALSE	当功能块执行被中断时为 TRUE。
Error	错误	BOOL	TRUE-FALSE	FALSE	功能块执行错误
ErrorID	错误代码	SMC_ERROR	-	0	错误指示，见 SMC_Error。

	布尔	位串	整数	实数	时刻、持续时间 日期、字符串

	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Axis	AXIS_REF_SM3																			
TimeVelocity	MC_TV_REF																			
Execute	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
ArraySize	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–
VelocityScale	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–
Offset	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–
Done	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Busy	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
CommandAborted	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Error	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
ErrorID	SMC_ERROR																			

- **功能说明：**

  - MC_VelocityProfile 功能块为时间和速度的轮廓运动模型 ， 按照用户在“时间-速度”表变量中设定的数据执行运动。

  - VT运动所用到的数据表：

1) “时间-速度”数据表的结构体如下：

```
TYPE MC_TV_TABLE
STRUCT
Number_of_pairs ： INT；          //速度-时间对的个数
IsAbsolute ： BOOL；              //速度值绝对相对选择
MC_TV_Array ： ARRAY [1..N] of MC_TV；      //速度-时间数据
END_STRUCT
END_TYPE
```

2) “时间-速度”数据的类型

```
TYPE MC_TV
STRUCT
delta_time ： TIME；     //速度-时间数据之时间
```

velocity ： REAL： //速度-时间数据之速度（绝对值或相对值）

END_STRUCT

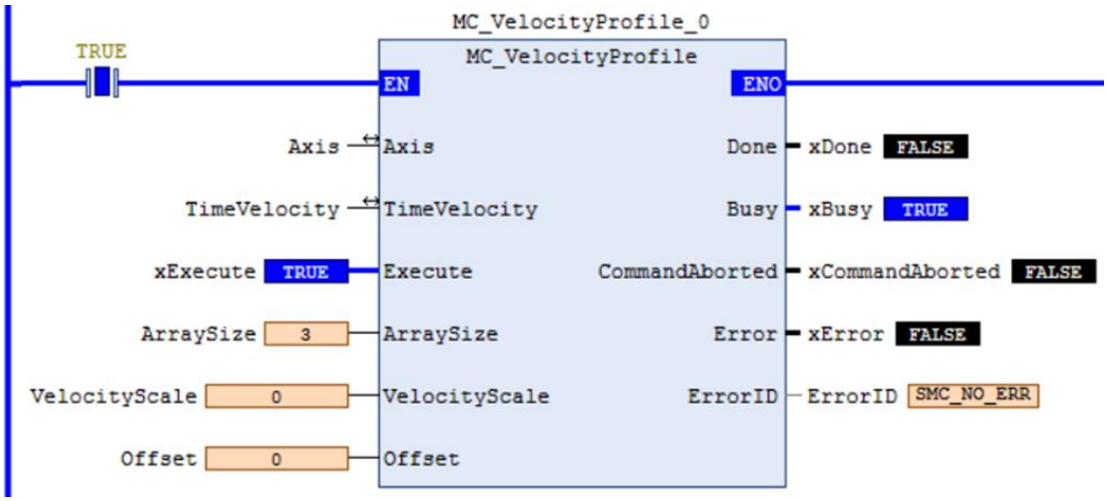
END_TYPE

- **程序示例**

ST：当xExecute变为TRUE时，执行时间-速度规划运动指令。

```
MC_VelocityProfile_0(
    Axis:=Axis ,
    TimeVelocity:= TimeVelocity,
    Execute TRUE := xExecute TRUE,
    ArraySize 3 := ArraySize 3,
    VelocityScale 0 := VelocityScale 0,
    Offset 0 := Offset 0,
    Done FALSE => xDone FALSE,
    Busy TRUE => xBusy TRUE,
    CommandAborted FALSE => xCommandAborted FALSE,
    Error FALSE => xError FALSE,
    ErrorID SMC_NO_ERR => ErrorID SMC_NO_ERR );
```

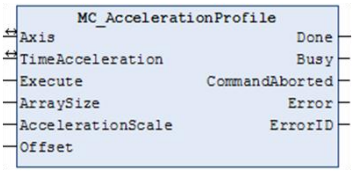
LD：当xExecute变为TRUE时，执行时间-速度规划运动指令。



4.2.13 MC_AccelerationProfile

本指令与 MC_PositionProfile 指令相似，MC_AccelerationProfile 通过定义“时间-加速度”数据来规划运动。

- **指令格式**

指令	名称	FB/FC	LD表现	ST表现	库
MC_AccelerationProfile	时间-加速度规划	FB		MC_AccelerationProfile(Axis：=， TimeAcceleration：=， Execute：=， ArraySize：=， AccelerationScale：=， Offset：=， Done=>， Busy=>， CommandAborted=>， Error=>， ErrorID=>);	SM3_Basic

- **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	初始值	描述
Axis	轴	AXIS_REF_SM3	–	–	指定轴
TimeAcceleration	数据表	MC_TA_REF	–	–	用户规划的时间-加速度数据表

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Execute	启动	BOOL	TRUE-FALSE	FALSE	上升沿触发指令运动。
ArraySize	数据点数	INT	0 正数	0	数据表大小

AccelerationScale	比例	LREAL	负数，0，正数	1	比例系数
Offset	偏移	LREAL	负数，0，正数	0	位置偏置

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
Done	执行完成	BOOL	TRUE-FALSE	FALSE	如果指令执行完成则为 TRUE。
Busy	执行中	BOOL	TRUE-FALSE	FALSE	当功能块执行还没结束时为 TRUE。
CommandAborted	指令被中断	BOOL	TRUE-FALSE	FALSE	当功能块执行被中断时为 TRUE。
Error	错误	BOOL	TRUE-FALSE	FALSE	功能块执行错误
ErrorID	错误代码	SMC_ERROR	–	0	错误指示，见 SMC_Error。

	布尔	位串				整数								实数		时刻、持续时间 日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Axis	AXIS_REF_SM3																			
TimeAcceleration	MC_TA_REF																			
Execute	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
ArraySize	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–
AccelerationScale	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–
Offset	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–
Done	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Busy	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
CommandAborted	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Error	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

ErrorID	SMC_ERROR

- **功能说明：**

  - MC_AccelerationProfile 功能块为时间和加速度的轮廓运动模型 ， 按照用户在“时间-加速度”表变量中设定的数据执行运动。

  - AT运动所用到的数据表：

1) “时间-加速度”数据表的结构体如下：

```
TYPE MC_TA_TABLE
STRUCT
Number_of_pairs ： INT；        //加速度-时间对的个数
IsAbsolute ： BOOL；            //加速度值绝对相对选择
MC_TA_Array ： ARRAY [1..N] of MC_TA；      //加速度-时间数据
END_STRUCT
END_TYPE
```

2) “时间-加速度”数据的类型

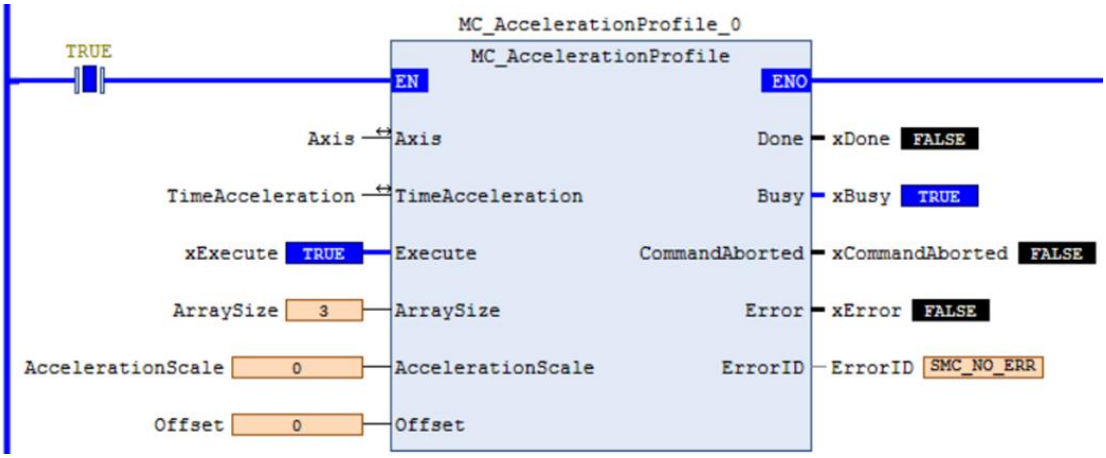
```
TYPE MC_TA
STRUCT
delta_time ： TIME；     //加速度-时间数据之时间
acceleration： REAL：        //加速度-时间数据之加速度（绝对值或相对值）
END_STRUCT
END_TYPE
```

- **程序示例**

ST：当xExecute变为TRUE时，执行时间-加速度规划运动指令。

```
MC_AccelerationProfile_0(
    Axis:=Axis ,
    TimeAcceleration := TimeAcceleration,
    Execute TRUE := xExecute TRUE,
    ArraySize 3 := ArraySize 3,
    AccelerationScale 0 := AccelerationScale 0,
    Offset 0 := Offset 0,
    Done FALSE => xDone FALSE,
    Busy TRUE => xBusy TRUE,
    CommandAborted FALSE => xCommandAborted FALSE,
    Error FALSE => xError FALSE,
    ErrorID SMC_NO_ERR => ErrorID SMC_NO_ERR );
```

LD：当xExecute变为TRUE时，执行时间-加速度规划运动指令。

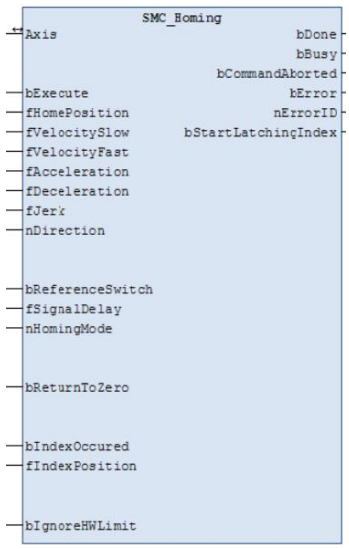


4.2.14 SMC_Homing

轴回零指令，本指令是通过控制器控制回原，与 MC_Home 有区别。

- **指令格式**

指令	名称	FB/FC	LD表现	ST表现	库

SMC_Homing	轴回零	FB		SMC_Homing(Axis:= , bExecute:=, fHomePosition:=, fVelocitySlow:=, fVelocityFast:=, fAcceleration:=, fDeceleration:=, fJerk:=, nDirection:=, bReferenceSwitch:=, fSignalDelay:=, nHomingMode:=, bReturnToZero:=, bIndexOccured:=, fIndexPosition:=, bIgnoreHWLimit:=, bDone=>, bBusy=> , bCommandAborted=>, bError=> , nErrorID=>, bStartLatchingIndex=>);	SM3_Basic

- **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	初始值	描述
Axis	轴	AXIS_REF_SM3	-	-	指定轴

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
bExecute	执行	BOOL	TRUE-FALSE	FALSE	True功能块执行，false不执行功能块
fHomePosition	原点设定位置	LREAL	负数，0，正数	0	回零后原点设定位置，单位为用户标定后的单位

fVelocitySlow	慢速	LREAL	负数，0，正数	0	离开参考开关后慢速设定速度
fVelocityFast	快速	LREAL	负数，0，正数	0	离开参考开关置位时，快速设定速度
fAcceleration	加速度	LREAL	负数，0，正数	0	加速度设定值
fDeceleration	减速度	LREAL	负数，0，正数	0	减速度设定值
fJerk	加加速度	LREAL	负数，0，正数	0	Jerk [u/s3]
nDirection	回零方向	MC_DIRECTION	负数，0，正数	negative	回零开始方向，参考MC_DIRECTION
bReferenceSwitch	参考开关	BOOL	TRUE-FALSE	FALSE	连接参考开关，TRUE:参考开关触发，FALSE:参考开关闭合
fSignalDelay	延迟	LREAL	负数，0，正数	0	参考开关的传输时间，用来补偿死区时间，单位为秒。
nHomingMode	回零模式	SMC_HOMING_MODE	-	-	参考SMC_HOMING_MODE
bReturnTozero	返回零位	BOOL	TRUE-FALSE	FALSE	TRUE:回零完成后轴运行到位置零（注意：如果fHomePosition=10，则回零完成后轴位置变为10，bReturnTozero为ture则回零完成后轴反向走10个单位到0位）
bIndexOccured	脉冲信号	BOOL	TRUE-FALSE	FALSE	True：标志脉冲记录，回零模式为FAST_BSLOW_I_S_STOP，FAST_SLOW_I_S_STOP时生效
fIndexPosition	记录位置	LREAL	负数，0，正数	0	标志脉冲时记录的位置
bIgnoreHWLimit	忽略硬限位	BOOL	TRUE-FALSE	FALSE	TRUE设定硬件限位开关使能为false，如果相同的物理开关用于硬件限位开关和参考开关，那么硬件控制将被设置为假

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
Done	执行完成	BOOL	TRUE-FALSE	FALSE	如果指令执行完成则为TRUE。
Busy	执行中	BOOL	TRUE-FALSE	FALSE	当功能块执行还没结束时为TRUE。
Command Aborted	指令被中断	BOOL	TRUE-FALSE	FALSE	当功能块执行被中断时为TRUE。
Error	错误	BOOL	TRUE-FALSE	FALSE	功能块执行错误
ErrorID	错误代码	SMC_ERROR	–	0	错误指示，见SMC_Error。
bStartLatchingIndex	启动锁存	BOOL	TRUE-FALSE	FALSE	由“bIndexOccured”和“fIndexPosition” 共同作用产生

	布尔	位串				整数								实数		时刻、持续时间 日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Axis	AXIS_REF_SM3																			
bExecute	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
fHomePosition	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
fVelocitySlow	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–
fVelocityFast	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–
fAcceleration	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–
fDeceleration	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–
fJerk	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
nDirection	MC_DIRECTION																			
bReferenceSwitch	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
fSignalDelay	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–
nHomingMode	SMC_HOMING_ MODE																			
bReturnTozero	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
bIndexOccured	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
fIndexPosition	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–
bIgnoreHWLimit	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Done	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Busy	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

CommandAborted	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Error	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
ErrorID	SMC_ERROR																			
bStartLatchingIndex	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

- **功能说明：**

  - SMC_HOMING 通过 bExecute 的上升沿启动之后，轴将会按照速度 fVelocityFast 并以 nDirection 定义的方向开始运动，直到 bReferenceSwitch = FALSE。然后轴将会缓慢停止并按照相反的方向以速度 fVelocitySlow 离开参考开关。bReferenceSwitch = TRUE 后回零完成，使能回零指令后 bReferenceSwitch 的状态为 ON->OFF->ON，在 OFF->ON 的上升沿回零完成，设置参考位置。

  - 参考位置 =fHomePostion+（（fSignalDelay*1000+1 个 DC 时钟周期）/1000*fVelocitySlow 实际就是补偿了设置的 bReferenceSwitch 采样延迟和一个通讯周期位移延迟。

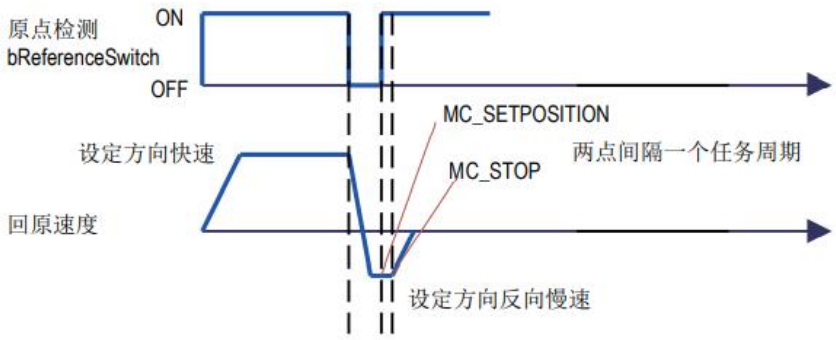
  - 如果 bReturnToZero=TRUE，bReferenceSwitch 的状态在 OFF->ON 的上升沿将参考位置设置为

  - fHomePostion+（（fSignalDelay*1000+1 个 DC 时钟周期）/1000*fVelocitySlow，然后按照速度fVelocityFast 运行到 0 位置。

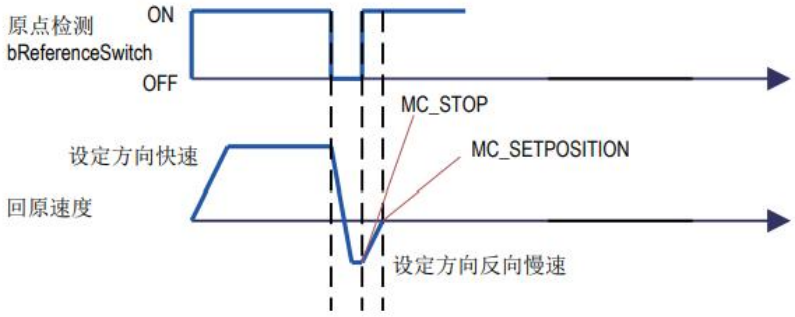
  - 注意：Done 完成信号后，轴位置设定为：fHomePosition。设定的时机跟 nHomingMode 有关（详情参考SMC_HOMING_MODE）。

  下图为几种回零模式：

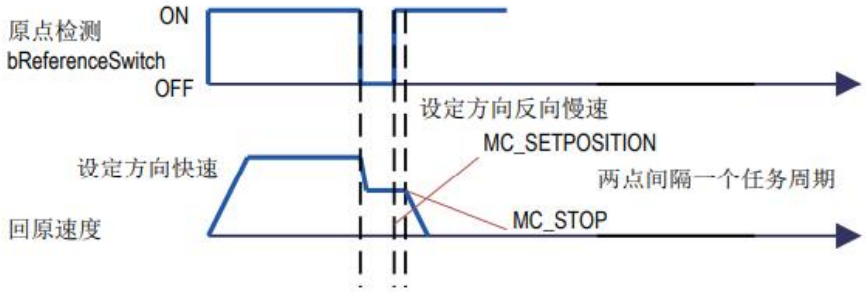
  1) 回零模式“0”时



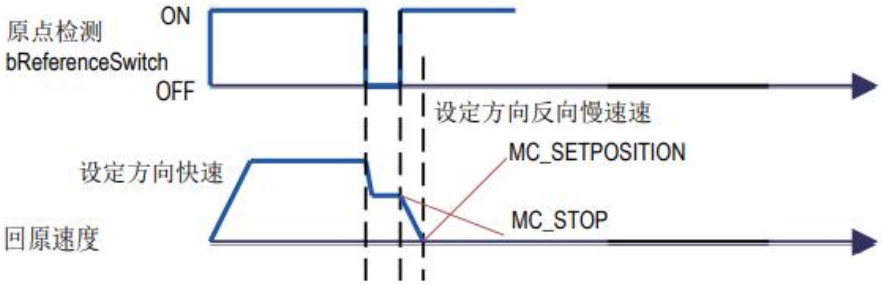
2) 回零模式“1”时



3) 回零模式“4”时



4) 回零模式“5”时

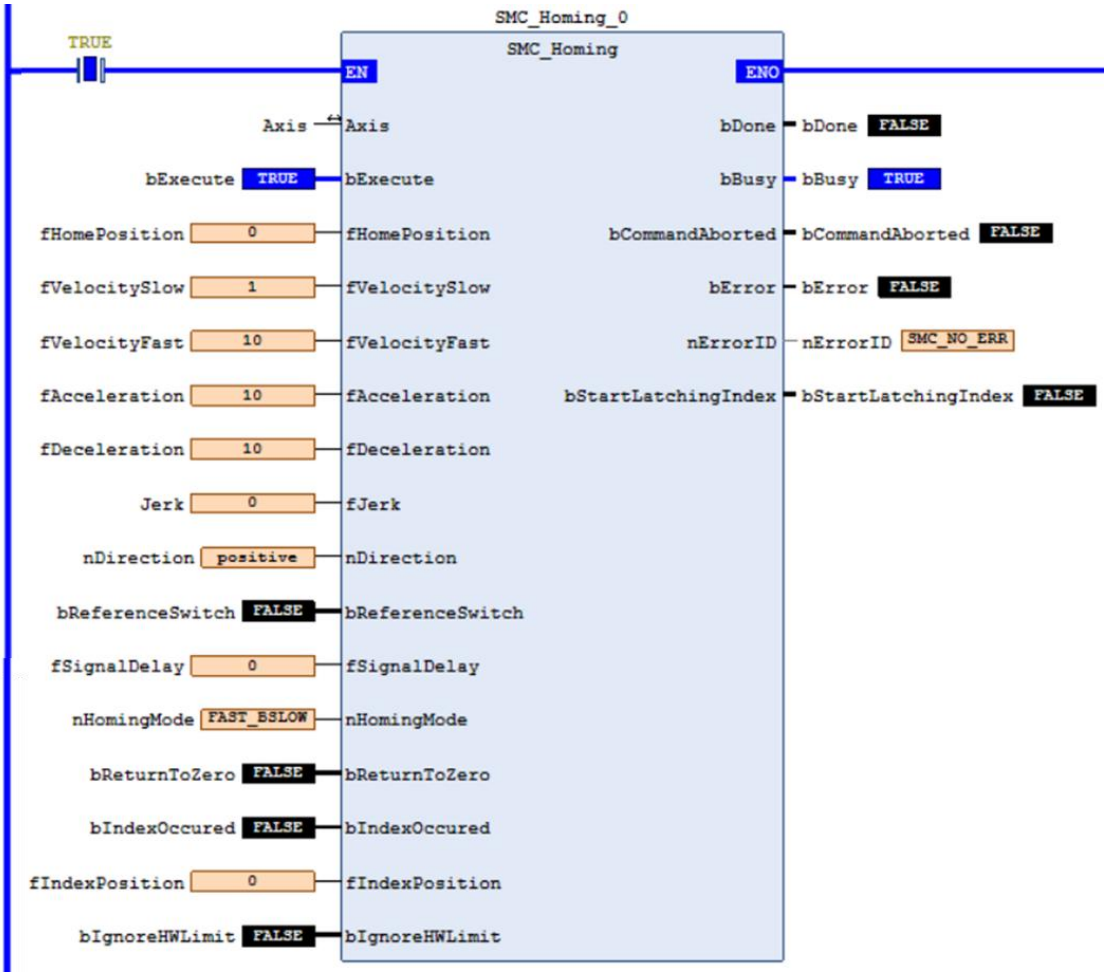


- **程序示例**

ST：当bExecute变为TRUE时，执行轴回零指令。

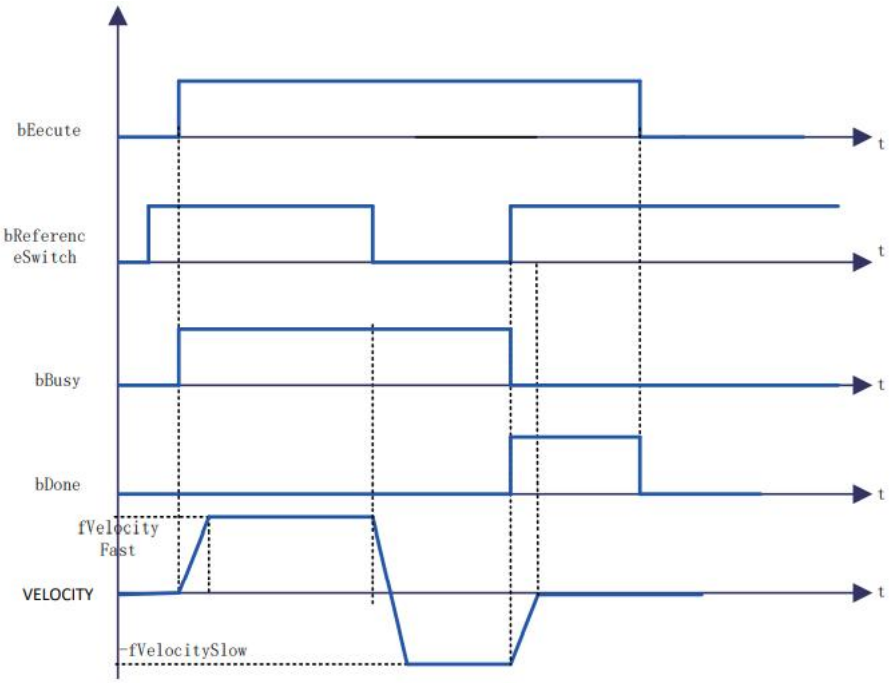
```
SMC_Homing_0(
    Axis:= Axis,
    bExecute TRUE := bExecute TRUE,
    fHomePosition 0 := fHomePosition 0,
    fVelocitySlow 1 := fVelocitySlow 1,
    fVelocityFast 10 := fVelocityFast 10,
    fAcceleration 10 := fAcceleration 10,
    fDeceleration 10 := fDeceleration 10,
    fJerk 0 := fJerk 0,
    nDirection positive := nDirection positive,
    bReferenceSwitch FALSE := bReferenceSwitch FALSE,
    fSignalDelay 0 := fSignalDelay 0,
    nHomingMode FAST_BSLOW := nHomingMode FAST_BSLOW,
    bReturnToZero FALSE := bReturnToZero FALSE,
    bIndexOccured FALSE := bIndexOccured FALSE,
    fIndexPosition 0 := fIndexPosition 0,
    bIgnoreHWLimit FALSE := bIgnoreHWLimit FALSE,
    bDone FALSE => bDone FALSE,
    bBusy TRUE => bBusy TRUE,
    bCommandAborted FALSE => bCommandAborted FALSE,
    bError FALSE => bError FALSE,
    nErrorID SMC_NO_ERR => nErrorID SMC_NO_ERR,
    bStartLatchingIndex FALSE => bStartLatchingIndex FALSE );
```

LD：当bExecute变为TRUE时，执行轴回零指令。

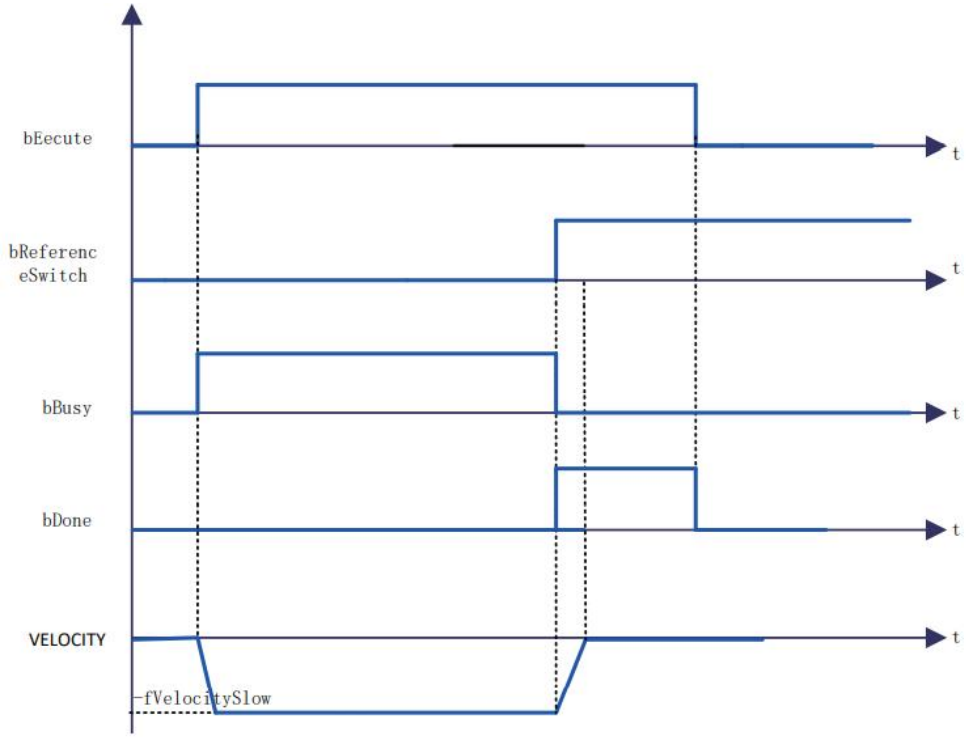


- **时序图：**

  指令执行时 bReferenceSwitch = TRUE 时



指令执行时 bReferenceSwitch = FALSE 时



4.2.15 SMC_Inch

本指令用于手动控制轴按指定的距离单位，一段一段的朝指定方向运动。

- **指令格式**

指令	名称	FB/FC	LD表现	ST表现	库
SMC_Inch	寸动	FB	SMC_Inch Axis — Busy InchForward — CommandAborted InchBackward — Error Distance — ErrorId Velocity Acceleration Deceleration Jerk	SMC_Inch(Axis：=， InchForward：=， InchBackward：=， Distance：=， Velocity：=， Acceleration：=， Deceleration：=， Jerk：=， Busy=>， CommandAborted=>， Error=>， ErrorId=>);	SM3_Basic

- **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	初始值	描述
Axis	轴	AXIS_REF_SM3	-	-	指定轴

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
InchForward	向前寸动	BOOL	TRUE-FALSE	FALSE	如果InchForward为TRUE，轴朝正向运动设定的距离，如果InchForward再次设定FALSE到TRUE，则轴再移动一段设定的距离。如果轴还没有运动到设定的距离，InchForward就变为FALSE，轴立即减速到0，Busy输出为FALSE。 如果InchBackward同时为真，轴不动。
InchBackward	向后寸动	BOOL	TRUE-FALSE	FALSE	如果InchBackward为TRUE，轴朝反向运动设定的距离，如果InchBackward再次设定FALSE到TRUE，则轴再移动一段设定的距离。如果轴还没有运动到设定的距离，InchBackward就变为FALSE，轴立即减速到0，Busy输出为FALSE。如果InchForward同时为真，轴不动。
Distance	寸动距离	LREAL	负数，0，正数	0	定义一次输入轴要运动的距离
Velocity	目标速度	LREAL	负数，0，正数	0	速度最大值 [u/s]
Acceleration	目标加速度	LREAL	负数，0，正数	0	加速度值[u/s2]
Deceleration	目标减速度	LREAL	负数，0，正数	0	减速度值[u/s2]

Jerk	目标加加速度	LREAL	负数，0，正数	0	加加速度值

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
Busy	执行中	BOOL	TRUE-FALSE	FALSE	当功能块执行还没结束时为 TRUE
Command Aborted	指令被中断	BOOL	TRUE-FALSE	FALSE	如果该命令已被其他命令终止则为 TRUE
Error	错误	BOOL	TRUE-FALSE	FALSE	功能块执行错误
ErrorID	错误代码	SMC_ERROR	-	0	错误指示，见 SMC_Error

	布尔	位串				整数								实数		时刻、持续时间 日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Axis	AXIS_REF_SM3																			
InchForward	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
InchBackward	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Distance	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–
Velocity	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–
Acceleration	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–
Deceleration	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–
Jerk	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–
Busy	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
CommandAborted	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Error	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
ErrorID	SMC_ERROR																			

- **功能说明：**
  - 一次运动的最大距离是固定的，由参数Distance定义。

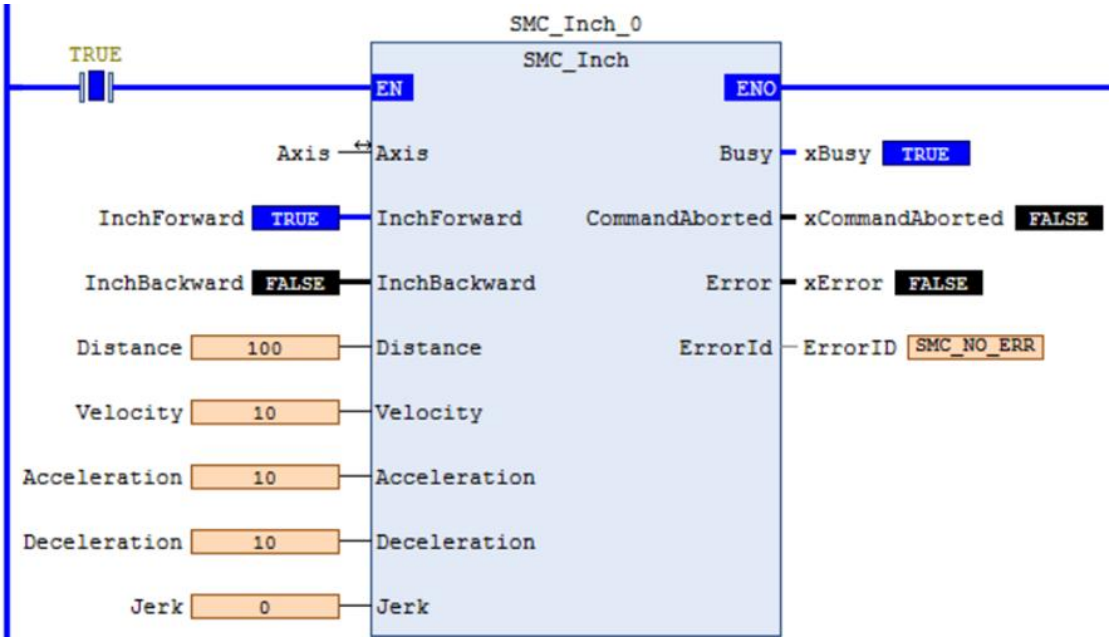
- 如果需要再次运行Distance的距离，需要重置输入（InchForward OR InchBackward）。
- 如果距离Distance还没有到达，输入（InchForward OR InchBackward）就重置为FALSE，运动立即减速停止。
- 输入InchForward和InchBackward都为TRUE时，轴不会运动。此时，当其中一个信号变成FALSE，则轴将执行还为TRUE的信号所指定的运动。
- 每次正向或反向运动的速度、加减速度等参数由Velocity、Acceleration、Deceleration和Jerk决定。

- **程序示例**

ST：当InchForward变为TRUE时，指定轴往正方向运动指定距离。

```
SMC_Inch_0(
    Axis:= Axis,
    InchForward TRUE := InchForward TRUE,
    InchBackward FALSE := InchBackward FALSE ,
    Distance 100 := Distance 100,
    Velocity 10 := Velocity 10,
    Acceleration 10 := Acceleration 10,
    Deceleration 10 := Deceleration 10,
    Jerk 0 := Jerk 0,
    Busy TRUE => xBusy TRUE,
    CommandAborted FALSE => xCommandAborted FALSE,
    Error FALSE => xError FALSE,
    ErrorId SMC_NO_ERR => ErrorId SMC_NO_ERR );
```

LD：当InchForward变为TRUE时，指定轴往正方向运动指定距离。


SMC_Inch_0
SMC_Inch
TRUE
EN
ENO
Axis
Axis
Busy
xBusy
TRUE
InchForward
TRUE
InchForward
CommandAborted
xCommandAborted
FALSE
InchBackward
FALSE
InchBackward
Error
xError
FALSE
Distance
100
Distance
ErrorId
ErrorID
SMC_NO_ERR
Velocity
10
Velocity
Acceleration
10
Acceleration
Deceleration
10
Deceleration
Jerk
0
Jerk

4.3同步运动功能

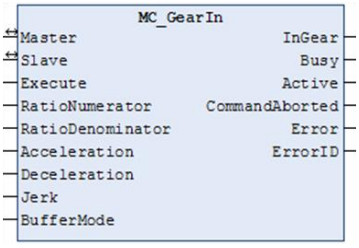
4.3.1 指令列表

指令类别	名称	FB/FC	功能
单轴运同步运动功能动控制	MC_GearIn	FB	电子齿轮耦合
	MC_GearInPos	FB	电子齿轮平滑耦合
	MC_GearOut	FB	电子齿轮脱离
	MC_CamTableSelect	FB	凸轮挺杆关联
	MC_CamIn	FB	电子凸轮关联
	MC_CamOut	FB	电子凸轮脱离
	SMC_GetTappetValue	FB	读取挺杆状态

4.3.2 MC_GearIn

本指令用于设置主从轴齿轮比并启动电子齿轮。

- **指令格式**

指令	名称	FB/FC	LD表现	ST表现	库
MC_GearIn	电子齿轮	FB		MC_GearIn(Master：= ， Slave：= ， Execute：= ， RatioNumerator:=， RatioDenominator:= ， Acceleration：= ， Deceleration：= ， Jerk：= ， BufferMode：=， InGear=>， Busy=>， Active=>， CommandAborted=> ， Error=>， ErrorID=>);	SM3_Basic

- **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	初始值	描述
Master	主轴	AXIS_REF_SM3	-	-	指定主轴
Slave	从轴	AXIS_REF_SM3	-	-	指定从轴

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Execute	启动	BOOL	TRUE-FALSE	FALSE	此输入的一个上升沿将启动功能块的处理
RatioNumerator	分子	DINT	正数	1	齿轮比分子
RatioDenominator	分母	UDINT	正数	1	齿轮比分母
Acceleration	目标加速度	LREAL	负数，0，正数	0	电子齿轮比加速度
Deceleration	目标减速度	LREAL	负数，0，正数	0	电子齿轮比减速度
Jerk	目标加加速度	LREAL	负数，0，正数	0	加加速度
BufferMode	缓存模式	MC_BUFFER_MODE	-	-	缓冲模式

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
InGear	齿轮啮合	BOOL	TRUE-FALSE	FALSE	TRUE 表示齿轮比的处理已经完成
Busy	执行中	BOOL	TRUE-FALSE	FALSE	TRUE 表示功能块的处理没有完成
Active	工作中	BOOL	TRUE-FALSE	FALSE	该轴在工作

Command Aborted	指令被中断	BOOL	TRUE-FALSE	FALSE	TRUE 表示命令被另一个命令中断
Error	错误	BOOL	TRUE-FALSE	FALSE	功能块内部发生错误信号
ErrorID	错误码	SMC_ERROR	-	0	错误 ID，参阅“SMC_ERROR”

	布尔	位串				整数								实数		时刻、持续时间 日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Master	AXIS_REF_SM3																			
Slave	AXIS_REF_SM3																			
Execute	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
RatioNumerator	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–
RatioDenominator	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–
Acceleration	–	–	–	–	–	–		–	–	–	–	–	–	–	√	–	–	–	–	–
Deceleration	–	–	–	–	–	–		–	–	–	–	–	–	–	√	–	–	–	–	–
Jerk	–	–	–	–	–	–		–	–	–	–	–	–	–	√	–	–	–	–	–
BufferMode	MC_BUFFER_MODE																			
InGear	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Busy	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Active	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
CommandAborted	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Error	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
ErrorID	SMC_ERROR																			

- **功能说明：**
  - 这个指令由“SM3_Basic”库实现。
  - 执行电子齿轮后，从轴将处于同步状态，要解除耦合，必须调用 GearOut 指令。
  - 指令属于速度同步类型，当启动指令后，从轴速度将上升至给定的齿轮比。在完成这个过程时，从轴耦合完成。在耦合期间造成的同步距离丢失不会得到补偿。

- 在MC_GearIn指令处于运行状态时，可以通过重新触发MC_GearIn指令，去改变主从轴的电子齿轮比，且中间不需要先调用MC_GearOut指令去暂停原来的电子齿轮运动。
- 到达目标速度，InGear 信号为 TRUE，此后，从轴移动量 = 主轴移动量 ×RatioNumerator / RatioDenominator



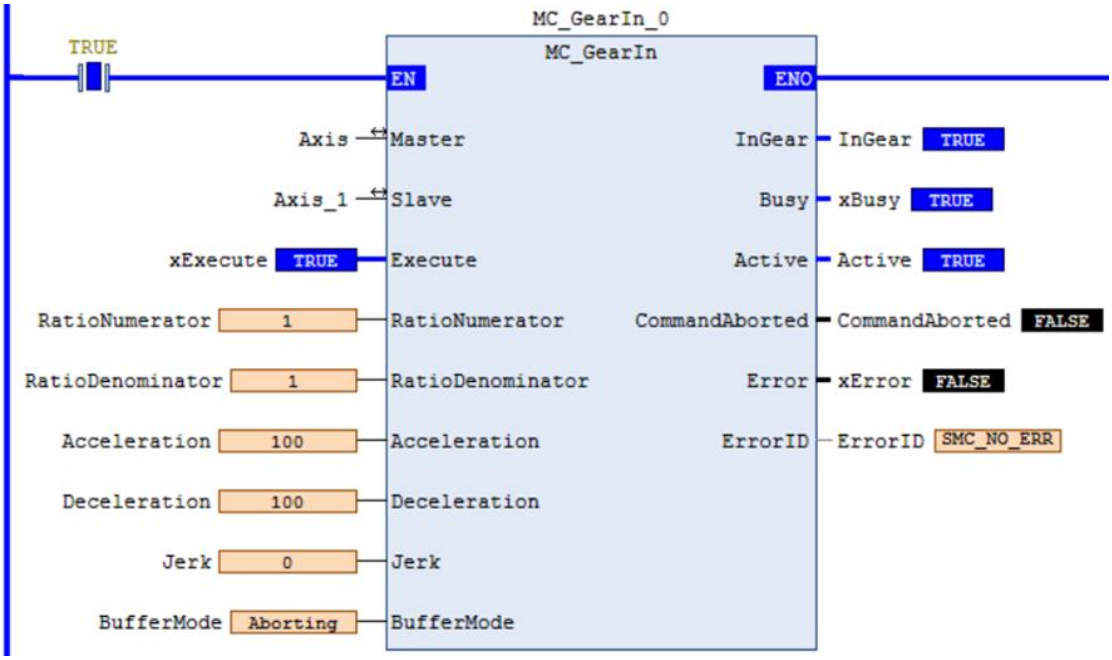
- 该指令一般适用于主轴速度稳定的情况，如果主轴速度实时变化的情况下，请注意慎重使用该指令。

- **程序示例**

ST：当xExecute变为TRUE时，设定主从轴齿轮比并启动电子齿轮。

```
MC_GearIn_0(
    Master:= Axis,
    Slave:= Axis_1,
    Execute TRUE := xExecute TRUE,
    RatioNumerator 1 := RatioNumerator 1,
    RatioDenominator 1 := RatioDenominator 1,
    Acceleration 100 := Acceleration 100,
    Deceleration 100 := Deceleration 100,
    Jerk 0 := Jerk 0,
    BufferMode Aborting := BufferMode Aborting,
    InGear TRUE => InGear TRUE,
    Busy TRUE => Busy TRUE,
    Active TRUE => Active TRUE,
    CommandAborted FALSE => CommandAborted FALSE,
    Error FALSE => xError FALSE,
    ErrorID SMC_NO_ERR => ErrorID SMC_NO_ERR );
```

LD：当xExecute变为TRUE时，设定主从轴齿轮比并启动电子齿轮。



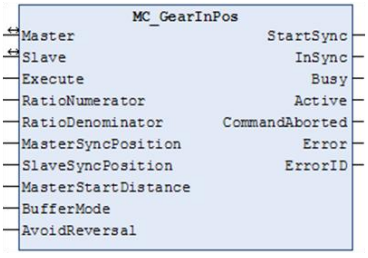
- **注意事项：**

➢ 执行指令过程中请不要使用 MC_SetPosition 指令以免引起电机急速运转产生事故。

4.3.3 MC_GearInPos

设定主从轴电子齿轮比，执行电子齿轮运动。与 MC_GearIn 指令不同的，是该指令需要指定开始同步的主轴位置、从轴位置、主轴开始同步距离，并以此来完成切入电子齿轮动作。

- **指令格式**

指令	名称	FB/FC	LD表现	ST表现	库
MC_GearInPos	齿轮平滑耦合	FB		MC_GearInPos(Master：=, Slave：=, Execute：=, RatioNumerator：=, RatioDenominator:=, MasterSyncPosition:=, SlaveSyncPosition:=,	SM3_Basic

				MasterStartDistance:=, BufferMode：=, AvoidReversal:= , StartSync=>, InSync=>, Busy=>, Active=>, CommandAborted=>, Error=>, ErrorID=>);	

- **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	初始值	描述
Master	主轴	AXIS_REF_SM3	-	-	指定主轴
Slave	从轴	AXIS_REF_SM3	-	-	指定从轴

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Execute	启动	BOOL	TRUE-FALSE	FALSE	此输入的一个上升沿将启动功能块的处理
RatioNumerator	分子	DINT	正数	1	齿轮比分子
RatioDenominator	分母	UDINT	正数	1	齿轮比分母
MasterSyncPosition	主轴同步位置	REAL	负数，0，正数	0	主从轴达到同步时主轴的位置
SlaveSyncPosition	从轴同步位置	REAL	负数，0，正数	0	主从轴达到同步时从轴的位置

MasterStartDistance	啮合段主轴运动距离	REAL	正数	0	从轴启动到和主轴同步时主轴运动的距离
BufferMode	缓存模式	MC_BUFFER_MODE	–	–	仅支持BufferModes Aborting、Buffered和BlendingPrevious。
AvoidReversal	禁止反转	BOOL	TRUE-FALSE	FALSE	设置为FALSE表示从轴物理位置超前的情况下进行反转；设置为TRUE表示从轴物理上不能实现反转或者导致危险，只在模态轴下适用。如果反转不能被避免，那么轴将错误停止。

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
StartSync	开始同步	BOOL	TRUE-FALSE	FALSE	TRUE 表示电子齿轮开始进行处理
InSync	到达同步	BOOL	TRUE-FALSE	FALSE	TRUE 表示电子齿轮命令完成
Busy	执行中	BOOL	TRUE-FALSE	FALSE	TRUE 表示功能块的处理没有完成
Active		BOOL			该轴在工作
CommandAborted	指令被中断	BOOL	TRUE-FALSE	FALSE	TRUE 表示命令被另一个命令中断
Error	错误	BOOL	TRUE-FALSE	FALSE	功能块内部发生错误信号
ErrorID	错误代码	SMC_ERROR	–	0	错误 ID，参阅“SMC_ERROR”

	布尔	位串				整数								实数		时刻、持续时间 日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Master	AXIS_REF_SM3																			
Slave	AXIS_REF_SM3																			
Execute	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
RatioNumerator	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–

RatioDenominator	_	_	_	_	_	_	_	√	_	_	_	_	_	_	_	_	_	_	_
MasterSyncPosition	_	_	_	_	_	_		_	_	_	_	_	_	_	√	_	_	_	_
SlaveSyncPosition	_	_	_	_	_	_		_	_	_	_	_	_	_	√	_	_	_	_
MasterStartDistance	_	_	_	_	_	_		_	_	_	_	_	_	_	√	_	_	_	_
BufferMode	MC_BUFFER_MODE																		
AvoidReversal	√	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_
StartSync	√	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_
InSync	√	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_
Busy	√	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_
Active	√	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_
CommandAborted	√	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_
Error	√	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_
ErrorID	SMC_ERROR																		

- **功能说明：**
  - 这个指令由“SM3_Basic”库实现。
  - 开始动作后，从轴以主轴速度乘以齿轮比得到的速度为目标速度，进行加减速动作。
  - 该功能块同步开始到同步结束的过程，本质为同步区间内从轴跟随主轴的一个电子凸轮，此时根据主轴范围（MasterSyncPosition- MasterStartDistance，MasterSyncPosition），从轴范围（当前位置，SlaveSyncPosition），指令会根据设置齿轮比以及上述三个参数自动设计一条凸轮曲线，执行同步时从轴跟随主轴完成凸轮动作。
  - 注意，如果主从轴工作在线性模式，需保证上述几个参数设置合理否则齿轮动作无法正确进行，故而建议使用该指令时主从轴为模数轴模式。比如主从轴线性工作模式都向正向运动，如果执行指令时主轴位置MasterSyncPosition-MasterStartDistance，或者从轴位置 > SlaveSyncPosition 则无法切入电子齿轮动作。
  - 同步完成（InSync为TRUE）的同时达到目标速度，
  - 此后从轴移动量 = 主轴移动量 × RatioNumerator / RatioDenominator

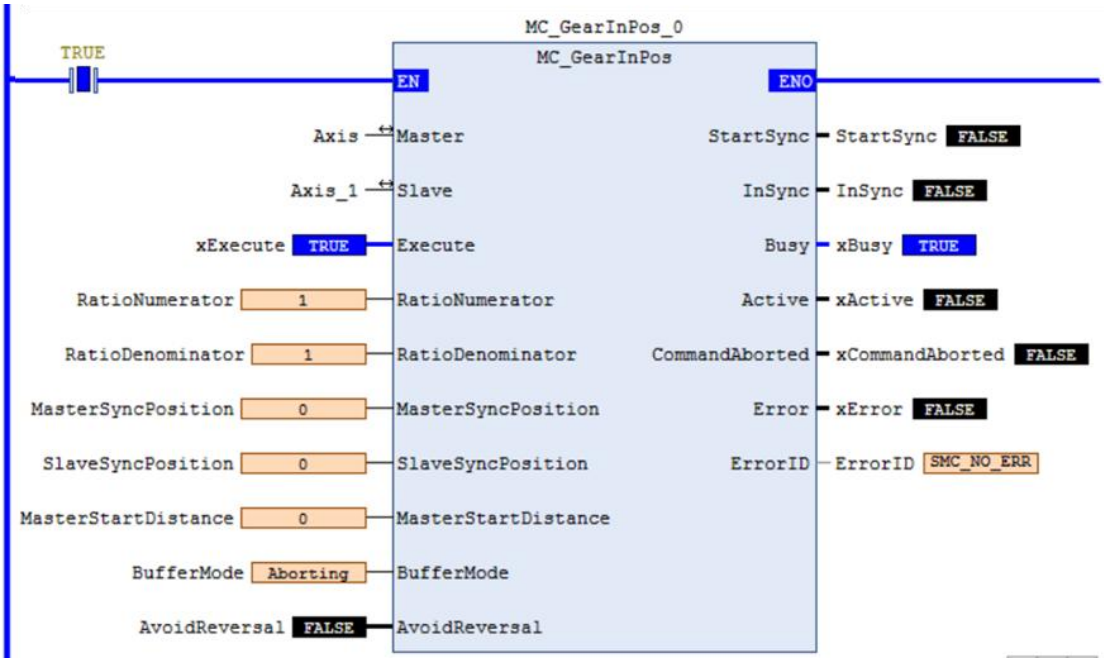
- 对于AvoidReversal：如果从轴是模态轴并且主轴速度（齿轮比的倍数关系）不是相对于从轴的速度关系，那么MC_GearInPos 会尝试着避免从轴的反转。它试图通过增加 5 个从轴周期“拉伸”从轴的运动。如果这个“拉伸”无效，那么会有错误出现并且从轴错误停止。如果从轴速度关联主轴速度（是齿轮比的倍数），那么会有错误出现，并且轴错误停止。如果从轴是线性模式轴，那么一个错误会产生在 Execute 输入上升沿处理时。

- **程序示例**

ST：当xExecute变为TRUE时，执行齿轮平滑耦合指令。

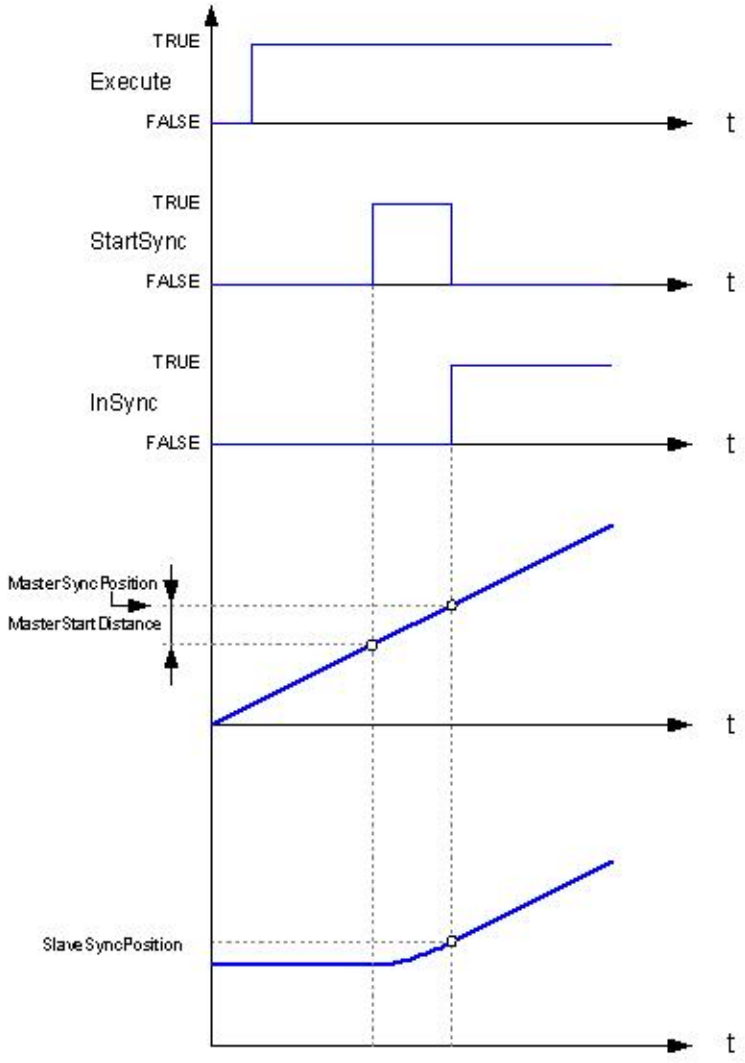
```
MC_GearInPos_0(
    Master:= Axis,
    Slave:= Axis_1,
    Execute TRUE := xExecute TRUE,
    RatioNumerator 1 := RatioNumerator 1,
    RatioDenominator 1 := RatioDenominator 1,
    MasterSyncPosition 0 := MasterSyncPosition 0,
    SlaveSyncPosition 0 := SlaveSyncPosition 0,
    MasterStartDistance 0 := MasterStartDistance 0,
    BufferMode Aborting := BufferMode Aborting,
    AvoidReversal FALSE := AvoidReversal FALSE,
    StartSync FALSE => StartSync FALSE,
    InSync FALSE => InSync FALSE,
    Busy TRUE => xBusy TRUE,
    Active FALSE => xActive FALSE,
    CommandAborted FALSE => xCommandAborted FALSE,
    Error FALSE => xError FALSE,
    ErrorID SMC_NO_ERR => ErrorID SMC_NO_ERR );
```

LD：当xExecute变为TRUE时，执行齿轮平滑耦合指令。



- **时序图：**

下图的时序图画出了在 Execute 信号置 True 后，MC_GearInPos 指令的各个主要信号的时序。



MC_GearInPos 时序图

4.3.4 MC_GearOut

本指令用于将从轴与主轴的电子齿轮耦合断开。

- **指令格式**

指令	名称	FB/FC	LD表现	ST表现	库
MC_GearOut	齿轮脱离	FB	MC_GearOut Slave Done Execute Busy Error ErrorID	MC_GearOut(Slave：=， Execute：=， Done=>， Busy=>， Error=>， ErrorID=>);	SM3_Basic

- **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	初始值	描述
Slave	从轴	AXIS_REF_SM3	-	-	指定从轴

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Execute	启动	BOOL	TRUE-FALSE	FALSE	此输入的一个上升沿将启动功能块的处理

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
Done	执行完成	BOOL	TRUE-FALSE	FALSE	TRUE 表示电子凸轮断开
Busy	执行中	BOOL	TRUE-FALSE	FALSE	TRUE 表示功能块的处理没有完成
Error	错误	BOOL	TRUE-FALSE	FALSE	功能块内部发生错误信号
ErrorID	错误代码	SMC_ERROR	-	0	错误 ID，参阅“SMC_ERROR”

	布尔	位串				整数								实数		时刻、持续时间 日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Slave	AXIS_REF_SM3																			
Execute	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Done	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Busy	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Error	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
ErrorID	SMC_ERROR																			

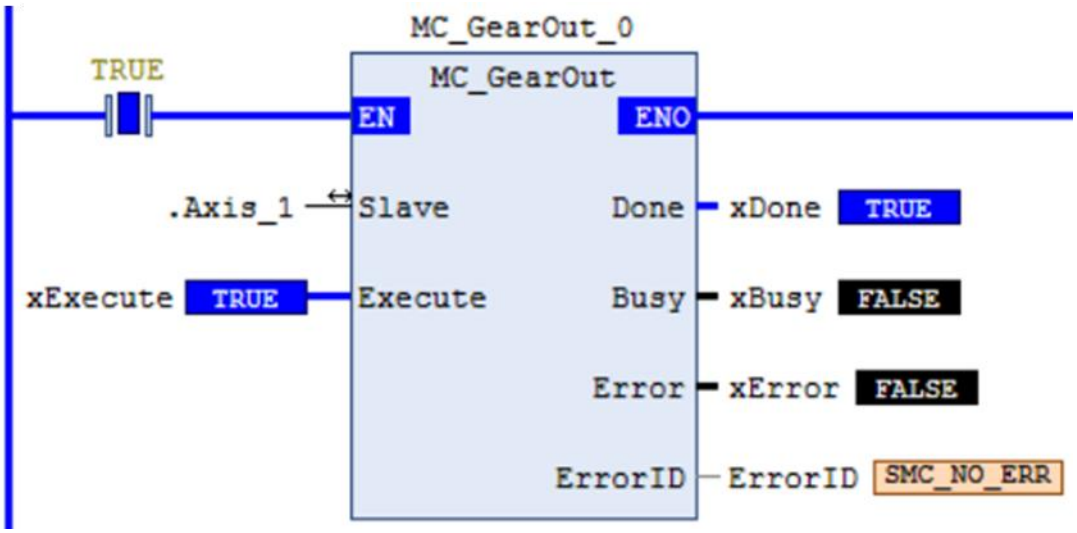
- **功能说明：**
  - 这个指令由“SM3_Basic”库实现。
  - 切出电子齿轮动作完成后，此时从轴的速度为切出前的速度，所以需配合以 MC_Stop 指令停止从轴。
  - 通过Slave(从轴)指定动作对象轴，指定Deceleration(减速度)，中止执行中的MC_GearIn(齿轮动作开始)指令、MC_GearInPos(位置指定齿轮动作)指令。
  - 本指令对MC_GearIn(齿轮动作开始)指令、MC_GearInPos(位置指定齿轮动作)指令的主轴动作没有影响。

- **程序示例**

ST：当xExecute变为TRUE时，执行齿轮脱离指令。

```
MC_GearOut_0(
Slave:= Axis_1 ,
Execute TRUE := xExecute TRUE,
Done TRUE => xDone TRUE,
Busy FALSE => xBusy FALSE,
Error FALSE => xError FALSE,
ErrorID SMC_NO_ERR => ErrorID SMC_NO_ERR );
```

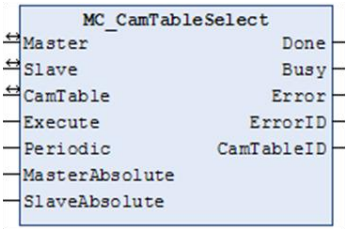
LD：当xExecute变为TRUE时，执行齿轮脱离指令。



4.3.5 MC_CamTableSelect

本指令用于选择需要执行的 cam 表，需要和 MC_CamIn 指令配合使用。

- **指令格式**

指令	名称	FB/FC	LD表现	ST表现	库
MC_CamTableSelect	凸轮表选择	FB		MC_CamTableSelect(Master：=, Slave：=, CamTable：= , Execute：=, Periodic：= , MasterAbsolute：= , SlaveAbsolute：= , Done=>, Busy=>, Error=>, ErrorID=>, CamTableID=>);	SM3_Basic

- **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	初始值	描述
Master	主轴	AXIS_REF	-	-	指定主轴
Slave	从轴	AXIS_REF	-	-	指定从轴
CamTable	凸轮表	MC_CAM_REF	-	-	指定凸轮表

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Execute	启动	BOOL	TRUE-FALSE	FALSE	此变量的一个上升沿将会启动功能块的处理

Periodic	重复模式	BOOL	TRUE-FALSE	FALSE	True：周期、重复地执行所指定的凸轮表 FALSE：仅执行一次凸轮表
MasterAbsolute	主轴绝对模式	BOOL	TRUE-FALSE	FALSE	TRUE表示绝对坐标，FALSE表示相对坐标
SlaveAbsolute	从轴绝对模式	BOOL	TRUE-FALSE	FALSE	TRUE表示绝对坐标，FALSE表示相对坐标

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
Done	执行完成	BOOL	TRUE-FALSE	FALSE	TRUE表示cam表格选择结束
Busy	执行中	BOOL	TRUE-FALSE	FALSE	TRUE表示功能块的处理没有完成
Error	错误	BOOL	TRUE-FALSE	FALSE	功能块内部发生错误信号
ErrorID	错误代码	SMC_ERROR	–	0	错误ID，参阅SMC_Error
CamTableID	挺杆ID	MC_CAM_ID	–	–	cam表格的定义，用于功能块MC_CamID

	布尔	位串				整数								实数		时刻、持续时间 日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Master	AXIS_REF_SM3																			
Slave	AXIS_REF_SM3																			
CamTable	MC_CAM_REF																			
Execute	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Periodic	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
MasterAbsolute	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
SlaveAbsolute	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Done	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Busy	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Error	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
ErrorID	SMC_ERROR																			

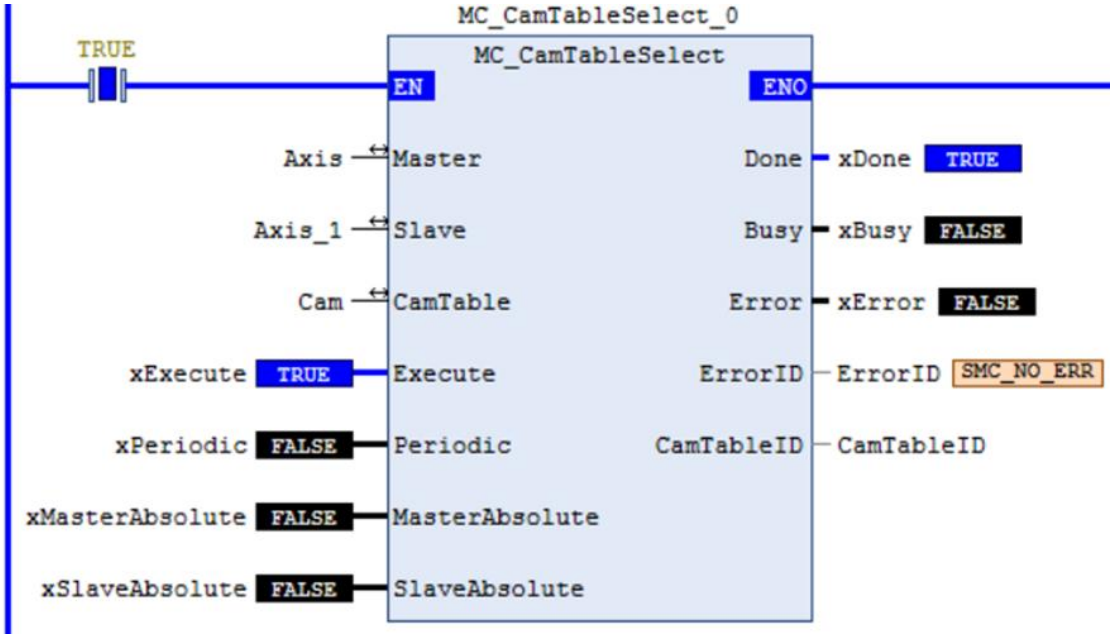
CamTableID	MC_CAM_ID

- **功能说明：**
  - 这个指令由“SM3_Basic”库实现。
  - 这个指令用于指定电子凸轮运行所需凸轮表，所以在使用本指令之前先要将凸轮表编辑好（凸轮编辑器编辑或者在线编辑好）。
  - Execute上升沿，执行指定凸轮表，亦可凸轮表更新后刷新指定的凸轮表。
  - Done信号输出为TRUE时，则输出变量“CamTableID”生成并且生效。
  - 主轴和从轴不能指定为同一轴，否则会有报错输出。
- **程序示例**

ST：当xExecute变为TRUE时，执行指定凸轮表指令。

```
MC_CamTableSelect_0(
    Master:= Axis,
    Slave:= Axis_1,
    CamTable:= Cam,
    Execute TRUE := xExecute TRUE,
    Periodic FALSE := xPeriodic FALSE,
    MasterAbsolute FALSE := xMasterAbsolute FALSE,
    SlaveAbsolute FALSE := xSlaveAbsolute FALSE,
    Done TRUE => xDone TRUE,
    Busy FALSE => xBusy FALSE,
    Error FALSE => xError FALSE,
    ErrorID SMC_NO_ERR => ErrorID SMC_NO_ERR,
    CamTableID=> CamTableID);
```

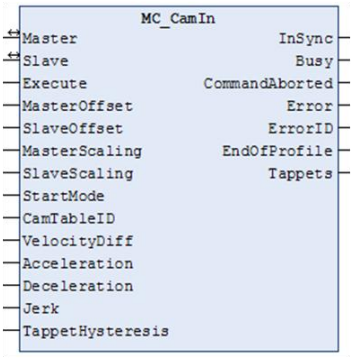
LD：当xExecute变为TRUE时，执行指定凸轮表指令。



4.3.6 MC_CamIn

使用指定的凸轮表开始执行电子凸轮动作。

- **指令格式**

指令	名称	FB/FC	LD表现	ST表现	库
MC_CamIn	电子凸轮耦合	FB		MC_CamIn(Master：=, Slave：=， Execute：=， MasterOffset：=， SlaveOffset：=, MasterScaling：=， SlaveScaling：=， StartMode：=， CamTableID：=， VelocityDiff：=， Acceleration：=， Deceleration：=， Jerk：=， TappetHysteresis:=, InSync=>, Busy=>, CommandAborted=>, Error=>, ErrorID=>,	SM3_Basic

				EndOfProfile=>， Tappets=>);	

- 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	初始值	描述
Master	主轴	AXIS_REF_SM3	-	-	指定主轴
Slave	从轴	AXIS_REF_SM3	-	-	指定从轴

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Execute	启动	BOOL	TRUE-FALSE	FALSE	此变量的一个上升沿将会启动功能块的处理
MasterOffset	主轴偏移	LREAL	负数，0，正数	0	主轴表格的偏移
SlaveOffset	从轴偏移	LREAL	负数，0，正数	0	从轴列表的偏移
MasterScaling	主轴比例	LREAL	正数	1	主轴缩放比例
SlaveScaling	从轴比例	LREAL	正数	1	从轴缩放比例
StartMode	从轴啮合方式	MC_StartMode	0-4	0：absolute	0：absolute 绝对位置 ： 1：relative 相对位置 ： 2：ramp_in (斜坡切入) 3：ramp_in_pos（正向斜坡切入） 4：ramp_in_neg 反向斜坡切入
CamTableID	凸轮表	MC_CAM_ID	-	-	定义cam表格的使用，是MC_CamID的输出

VelocityDiff	速度	LREAL	负数，0，正数	0	与ramp_in不同的最大速度
Acceleration	加速度	LREAL	0，正数	0	对ramp_in的最大加速度
Deceleration	减速度	LREAL	0，正数	0	对ramp_in的最大减速度
Jerk	目标加加速度	LREAL	负数，0，正数	0	加加速度
TappetHysteresis	挺杆阻尼	LREAL	0，正数	0	挺杆滞后的尺寸大小

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
InSync	同步中	BOOL	TRUE-FALSE	FALSE	TRUE表示从轴根据cam表格与主轴同步
Busy	执行中	BOOL	TRUE-FALSE	FALSE	TRUE表示功能块的处理没有完成
CommandAborted	指令被中断	BOOL	TRUE-FALSE	FALSE	TRUE表示命令被另一个命令中断
Error	错误	BOOL	TRUE-FALSE	FALSE	功能块内部发生错误
ErrorID	错误代码	SMC_ERROR	–	0	错误ID，参阅SMC_Error
EndOfProfile	曲线完成	BOOL	TRUE-FALSE	FALSE	在cam编译周期结束输出的脉冲信号
Tappets	挺杆表	SMC_TappetData	–	–	用于SMC_GetTappetValue处理的挺杆信号

	布尔	位串				整数								实数		时刻、持续时间 日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Master	AXIS_REF_SM3																			
Slave	AXIS_REF_SM3																			
Execute	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

MasterOffset	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–
SlaveOffset	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–
MasterScaling	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–
SlaveScaling	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–
StartMode	MC_StartMode																			
CamTableID	MC_CAM_ID																			
VelocityDiff	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–
Acceleration	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–
Deceleration	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–
Jerk	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–
TappetHysteresis	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–
InSync	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Busy	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
CommandAborted	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Error	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
ErrorID	SMC_ERROR																			
EndOfProfile	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Tappets	SMC_TappetData																			

- **功能说明：**
  - 这个指令由“SM3_Basic”库实现。
  - 本指令可使相位(主轴)和位移(从轴)按照凸轮表进行同步凸轮动作。
  - 由本指令指定的凸轮表有两种制作方式：

1) 使用凸轮编辑器编制；

2) 通过编程自建凸轮表数据结构。

  - 在一个凸轮系统中，要调用一条凸轮曲线，先调用 MC_CamTableSelect 指令选择相应的凸轮表，再执行MC_CamIn；如要更换凸轮曲线，则再调用 MC_CamTableSelect 指令重新选择凸轮表。

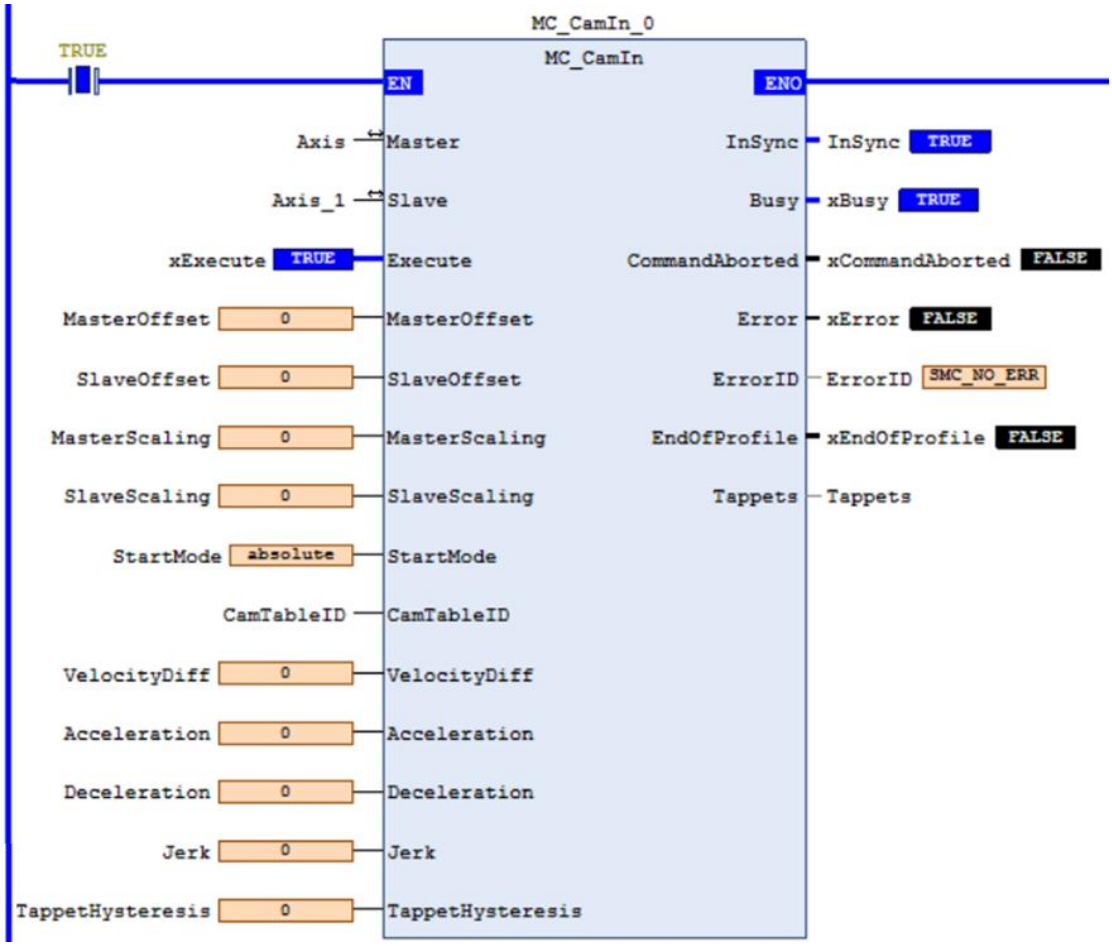
- 需使用 MC_CamOut 指令断开主从轴的凸轮耦合关系。
- 该指令执行时，该指令的从轴再执行其它运动指令时，从轴和主轴之间的凸轮关系会解除，并且MC_CamIn的Command-Aborted变量输出为TRUE。

- **程序示例**

ST：当xExecute变为TRUE时，执行电子凸轮耦合指令。

```
MC_CamIn_0(
Master:= Axis,
Slave:= Axis_1,
Execute TRUE := xExecute TRUE,
MasterOffset 0 := MasterOffset 0,
SlaveOffset 0 := SlaveOffset 0,
MasterScaling 0 := MasterScaling 0,
SlaveScaling 0 := SlaveScaling 0,
StartMode absolute := StartMode absolute,
CamTableID:= CamTableID,
VelocityDiff 0 := VelocityDiff 0,
Acceleration 0 := Acceleration 0,
Deceleration 0 := Deceleration 0,
Jerk 0 := Jerk 0,
TappetHysteresis 0 := TappetHysteresis 0,
InSync TRUE => xInSync TRUE,
Busy TRUE => xBusy TRUE,
CommandAborted FALSE => xCommandAborted FALSE,
Error FALSE => xError FALSE,
ErrorID SMC_NO_ERR => ErrorID SMC_NO_ERR,
EndOfProfile FALSE => xEndOfProfile FALSE,
Tappets=> Tappets);
```

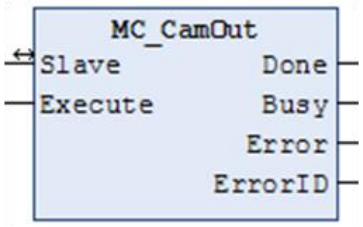
LD：当xExecute变为TRUE时，执行电子凸轮耦合指令。



4.3.7 MC_CamOut

解除指定的从轴与其对应主轴的凸轮耦合关系。

- **指令格式**

指令	名称	FB/FC	LD表现	ST表现	库
MC_CamOut	电子凸轮脱离	FB		MC_CamOut(Slave：=, Execute：= , Done=>, Busy=>, Error=>, ErrorID=>);	SM3_Basic

- **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	初始值	描述
Slave	从轴	AXIS_REF	-	-	指定从轴

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Execute	启动	BOOL	TRUE-FALSE	FALSE	此变量的一个上升沿将会启动功能块的处理

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
Done	执行完成	BOOL	TRUE-FALSE	FALSE	TRUE，如果cam已经被分离
Busy	执行中	BOOL	TRUE-FALSE	FALSE	TRUE，如果功能块的处理没有完成
Error	错误	BOOL	TRUE-FALSE	FALSE	功能块内有错误信号发生
ErrorID	错误码	SMC_ERROR	-	0	错误ID，参阅SMC_Error

	布尔	位串				整数								实数		时刻、持续时间 日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Slave	AXIS_REF_SM3																			
Execute	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Done	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Busy	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Error	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
ErrorID	SMC_ERROR																			

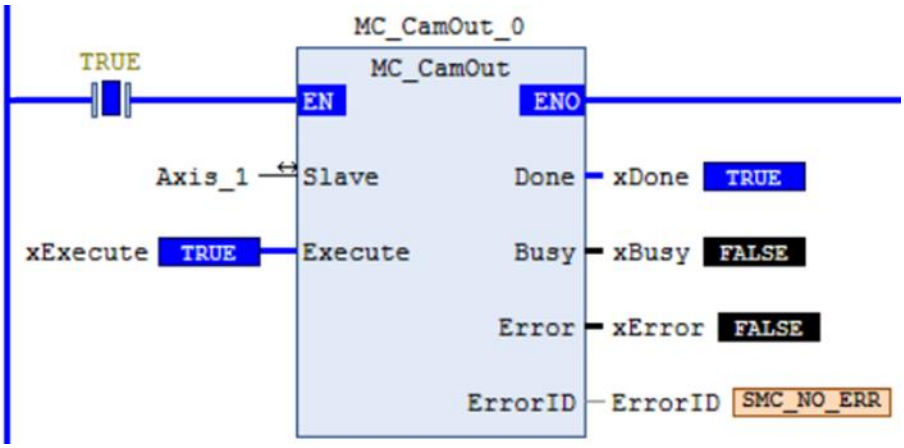
- **功能说明：**
  - 这个指令由“SM3_Basic”库实现。
  - 本指令用于解除指定的从轴与其对应主轴的凸轮耦合关系。

- **程序示例**

ST：当xExecute变为TRUE时，执行电子凸轮脱离指令。

```
MC_CamOut_0(
    Slave:= Axis_1,
    Execute:= xExecute,
    Done=> xDone,
    Busy=> xBbusy,
    Error=> xError,
    ErrorID=> ErrorID);
```

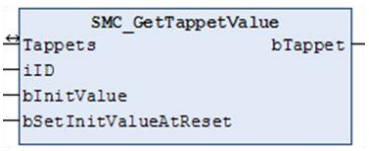
LD：当xExecute变为TRUE时，执行电子凸轮脱离指令。



4.3.8 SMC_GetTappetValue

读取当前的挺杆状态，需要和 MC_CamIn 指令配合使用。

- **指令格式**

指令	名称	FB/FC	LD表现	ST表现	库
SMC_GetTappetValue	读取挺杆状态	FB		SMC_GetTappetValue (Tappets：=, iID：=, bInitValue：= , bSetInitValueAtReset：=, bTappet=>);	SM3_Basic

- **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	初始值	描述
Tappets	挺杆	SMC_TappetData	–	–	通过MC_GetTappetValue评估的挺杆信号，参阅SMC_TappetData

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
iID	挺杆组号	INT	0，正数	0	要评估的挺杆组ID
bInitValue	初始值	BOOL	TRUE-FALSE	FALSE	第一次调用时分配的挺杆初始值
bSetInitValueAtReset	挺杆复位	BOOL	TRUE-FALSE	FALSE	如果为TRUE，在 CamIn 功能块重启的时候挺杆的输出值将会被设置为初始值。如果为FALSE，挺杆的值在CamIn 功能块重启的时候将会被保持。

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
bTappet	挺杆值	BOOL	TRUE /FALSE	FALSE	挺杆的值

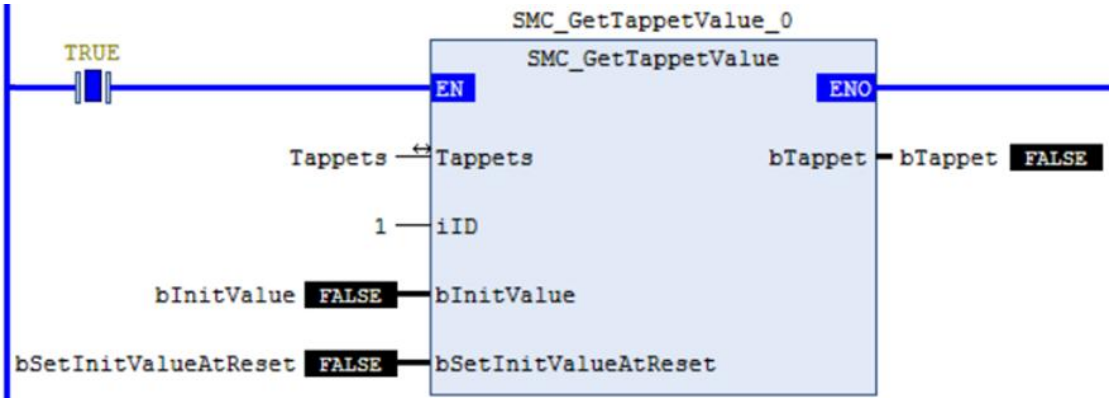
	布尔	位串				整数								实数		时刻、持续时间 日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Tappets	SMC_TappetData																			
iID	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–
bInitValue	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
bSetInitValueAtReset	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
bTappet	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
ErrorID	SMC_ERROR																			

- **功能说明：**
  - 这个指令由“SM3_Basic”库实现。
  - 在机械凸轮结构中，凸轮的旋转运动靠挺杆转换为垂直方向的运动。电子凸轮的挺杆，与机械挺杆类似，相当于电子化的机械挺杆。电子凸轮的挺杆就是凸轮主轴在规定的位置，给出个高电平或者低电平，作为开关去使用。
  - 该功能块需要和MC_CamIn指令配合使用，用来获取凸轮表管理的挺杆的数值。
- **程序示例**

ST：当xExecute变为TRUE时，执行读取挺杆状态指令。

```
SMC_GetTappetValue_0(
Tappets:= Tappets,
iID  1  := 1,
bInitValue FALSE := bInitValue FALSE,
bSetInitValueAtReset FALSE := bSetInitValueAtReset FALSE,
bTappet FALSE => bTappet FALSE);
```

LD：当xExecute变为TRUE时，执行读取挺杆状态指令。



4.4插补指令

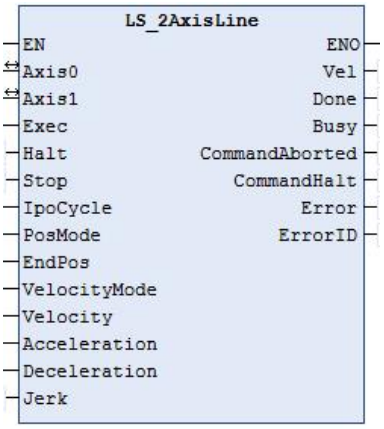
4.4.1 指令列表

指令类别	名称	FB/FC	功能
插补指令	LS_2AxisLine	FB	两轴直线插补
	LS_3AxisLine	FB	三轴直线插补
	LS_4AxisLine	FB	四轴直线插补
	LS_5AxisLine	FB	五轴直线插补
	LS_6AxisLine	FB	六轴直线插补
	LS_2AxisLineA_Ratio	FB	两轴可调速直线插补
	LS_LineFollow	FB	跟随运动
	LS_2AxisCircle	FB	两轴圆弧插补
	LS_3AxisCircle	FB	三轴圆弧插补
	LS_2AxisEllipses	FB	两轴椭圆插补
	LS_2AxisCircle_Helical	FB	螺旋插补
	LS_3AxisMoveSequence	FB	三轴连续插补运动
	LS_4AxisMoveSequence	FB	四轴连续插补运动
	LS_6AxisMoveSequence	FB	六轴连续插补运动

4.4.2 LS_2AxisLine

两轴直线插补指令。

- **指令格式**

指令	名称	FB/FC	LD表现	ST表现	库
LS_2AxisLine	两轴直线插补	FB		LS_2AxisLine(Axis0:=, Axis1:=, Exec:=, Halt:=, Stop:=, IpoCycle:=, PosMode:=, EndPos:=, VelocityMode:=, Velocity:=, Acceleration:=,	LS_IpoLib

				Deceleration:=, Jerk:=, Vel=>, Done=>, Busy=>, CommandAborted=>, CommandHalt=>, Error=>, ErrorID=>);	

- **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	初始值	描述
Axis0	轴0	AXIS_REF_VIRTUAL_SM3	-	-	参与插补的0轴。
Axis1	轴1	AXIS_REF_VIRTUAL_SM3	-	-	参与插补的1轴。

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Exec	启动	BOOL	TRUE，FALSE	FALSE	直线插补模块启动命令，上升沿有效。
Halt	暂停	BOOL	TRUE，FALSE	FALSE	暂停命令，当该输入量为True时，当前插补运动暂停；当状态再次切换到False时，插补模块继续之前的未完成的插补任务。
Stop	停止	BOOL	TRUE，FALSE	FALSE	停止命令，该命令为True时，插补模块停止。
IpoCycle	插补周期	DWORD	遵循数据类型	2000	插补周期，单位：us。
PosMode	位置模式	INT	[0,1]	0	位置模式： 0 绝对模式； 1 相对模式。

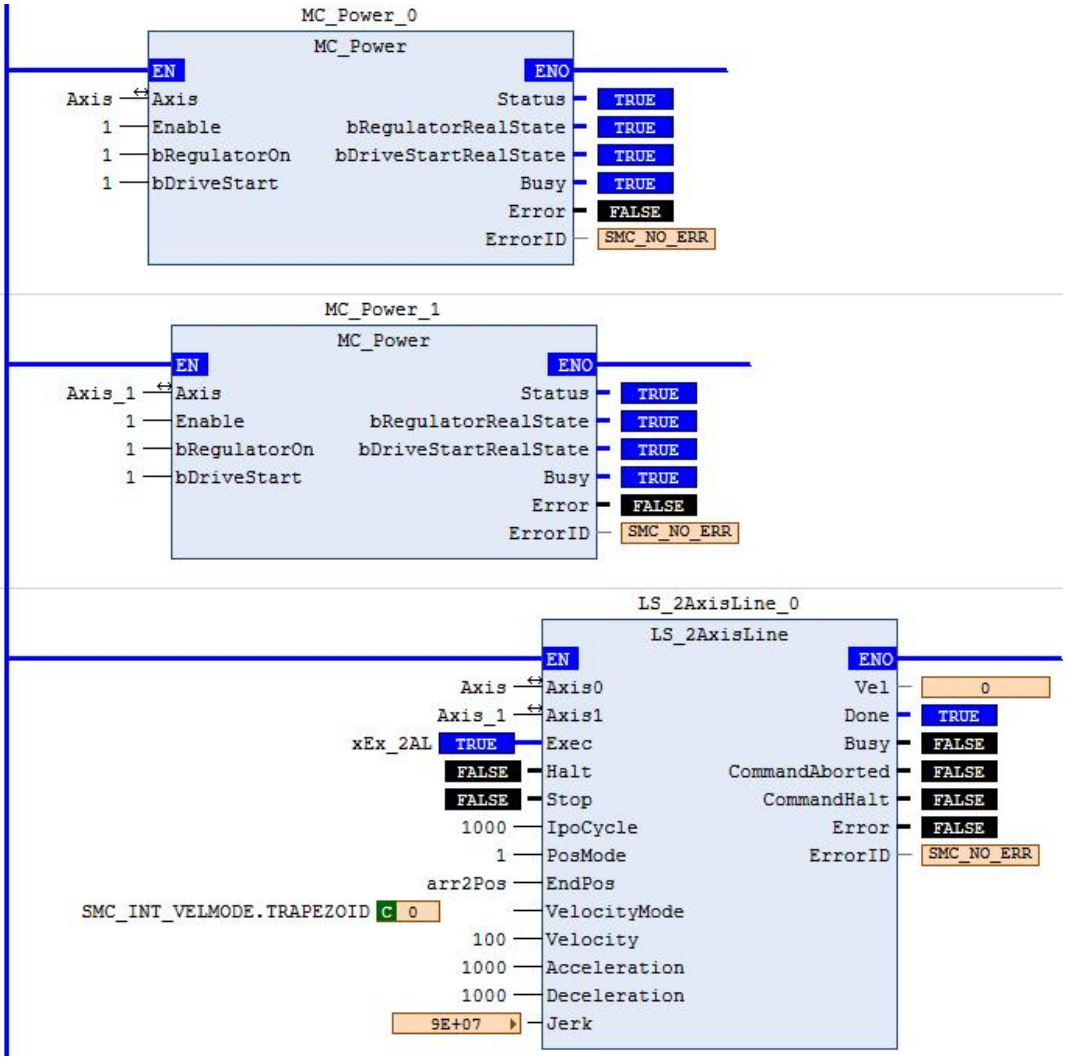
Endpos	目标位置	ARRAY [0..1] OF REAL	负数，0，正数	0	轴的目标位置，单位：pulse。
VelocityMode	速度模式	SMC_INT_VELMODE	0-3	SIGMOID	速度模式，梯形：0（TRAPEZOID）；S形：1（SIGMOID）；四次方：3（QUADRATIC）。
Velocity	速度	LREAL	0，正数	0	插补合成速度。
Acceleration	加速度	LREAL	0，正数	0	插补合成加速度。
Deceleration	减速度	LREAL	0，正数	0	插补合成减速度。
Jerk	加加速度	LREAL	负数，0 正数	90000000	加加速度，速度模式为3时候需要设置此参数，此参数值不能为0。

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
Vel	合速度	LREAL	0，正数	0	当前运动合成速度。
Done	完成	BOOL	TRUE，FALSE	FALSE	为 True 时，表示插补模块已完成插补任务。
Busy	运动中	BOOL	TRUE，FALSE	FALSE	为 True 时，表示插补运动正在进行之中。
CommandAborted	运动中断	BOOL	TRUE，FALSE	FALSE	当 Stop 有效时，该变量为 True，表示插补模块非正常中止。
CommandHalt	运动暂停	BOOL	TRUE，FALSE	FALSE	当 Halt 为 True 时，该变量为 True，表示插补模块处于暂停状态。
Error	报错	BOOL	TRUE，FALSE	FALSE	错误状态。
ErrorID	错误码	DINT	0，正数	0	错误码。

- **程序示例**

两个轴以相对模式，直线插补运动到目标位置 1000，1000。



- **注意事项**

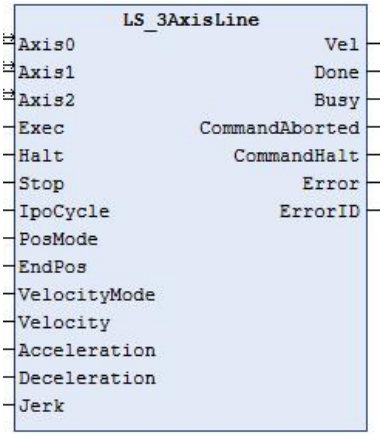
  - IpoCycle参数需要与运动指令所在的任务周期时间保持一致，否则指令执行时可能会出现Error报错。
  - 参数IpoCycle、Jerk、Velocity、Acceleration、Deceleration不可以设置为0。
  - 在指令运行期间，主从轴都不能再被其它运动指令所调用。

4.4.3 LS_3AxisLine

三轴直线插补指令。

- **指令格式**

指令	名称	FB/	LD表现	ST表现	库

		FC			
LS_3AxisLine	三轴直线插补	FB		LS_3AxisLine(Axis0:=, Axis1:=, Axis2:=, Exec:=, Halt:=, Stop:=, IpoCycle:=, PosMode:=, EndPos:=, VelocityMode:=, Velocity:=, Acceleration:=, Deceleration:=, Jerk:=, Vel=>, Done=>, Busy=>, CommandAborted=>, CommandHalt=>, Error=>, ErrorID=>);	LS_IpoLib

- **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	初始值	描述
Axis0	轴0	AXIS_REF_VIRTUAL_SM3	-	-	参与插补的0轴。
Axis1	轴1	AXIS_REF_VIRTUAL_SM3	-	-	参与插补的1轴。
Axis2	轴2	AXIS_REF_VIRTUAL_SM3	-	-	参与插补的2轴。

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述

Exec	启动	BOOL	TRUE，FALSE	FALSE	直线插补模块启动命令，上升沿有效。
Halt	暂停	BOOL	TRUE，FALSE	FALSE	暂停命令，当该输入量为True时，当前插补运动暂停；当状态再次切换到False时，插补模块继续之前的未完成的插补任务。
Stop	停止	BOOL	TRUE，FALSE	FALSE	停止命令，该命令为True时，插补模块停止。
IpoCycle	插补周期	DWORD	遵循数据类型	2000	插补周期，单位：us。
PosMode	位置模式	INT	[0,1]	0	位置模式： 0 绝对模式； 1 相对模式。
Endpos	目标位置	ARRAY [0..1] OF REAL	负数，0，正数	0	轴的目标位置，单位：pulse。
VelocityMode	速度模式	SMC_INT_VELMODE	0-3	SIGMOID	速度模式，梯形：0（TRAPEZOID）；S形：1（SIGMOID）；四次方：3（QUADRATIC）。
Velocity	速度	LREAL	0，正数	0	插补合成速度。
Acceleration	加速度	LREAL	0，正数	0	插补合成加速度。
Deceleration	减速度	LREAL	0，正数	0	插补合成减速度。
Jerk	加加速度	LREAL	负数，0 正数	90000000	加加速度，速度模式为3时候需要设置此参数，此参数值不能为0。

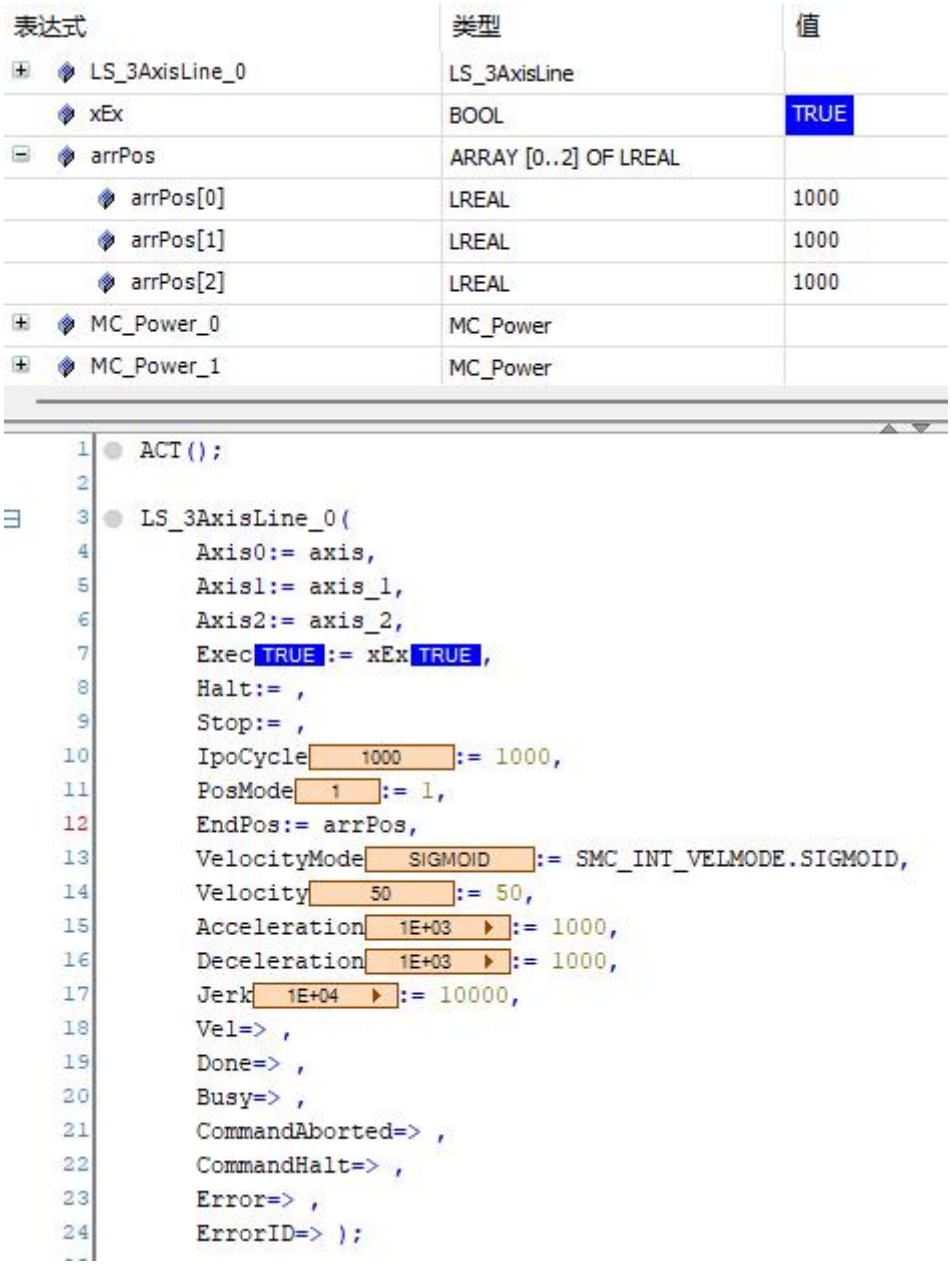
输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
Vel	合速度	LREAL	0，正数	0	当前运动合成速度。
Done	完成	BOOL	TRUE，FALSE	FALSE	为 True 时，表示插补模块已完成插补任务。
Busy	运动中	BOOL	TRUE，FALSE	FALSE	为 True 时，表示插补运动正在进行之中。

Command Aborted	运动中断	BOOL	TRUE，FALSE	FALSE	当 Stop 有效时，该变量为 True，表示插补模块非正常中止。
Command Halt	运动暂停	BOOL	TRUE，FALSE	FALSE	当 Halt 为 True 时，该变量为 True，表示插补模块处于暂停状态。
Error	报错	BOOL	TRUE，FALSE	FALSE	错误状态。
ErrorID	错误码	DINT	0，正数	0	错误码。

- **程序示例**

三个轴以相对模式，直线插补运动到目标位置 1000，1000，1000。



- **注意事项**

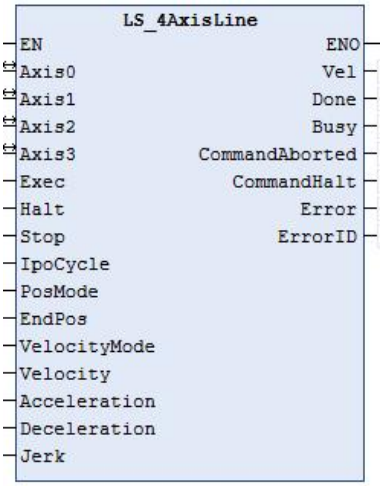
- IpoCycle参数需要与运动指令所在的任务周期时间保持一致，否则指令执行时可能会出现Error报错。

- 参数IpoCycle、Jerk、Velocity、Acceleration、Deceleration不可以设置为0。
  在指令运行期间，主从轴都不能再被其它运动指令所调用。

4.4.4 LS_4AxisLine

四轴直线插补指令。

- **指令格式**

指令	名称	FB/FC	LD表现	ST表现	库
MC_4AxisLine	四轴直线插补	FB		LS_4AxisLine(Axis0:=, Axis1:=, Axis2:=, Axis3:=, Exec:=, Halt:=, Stop:=, IpoCycle:=, PosMode:=, EndPos:=, VelocityMode:=, Velocity:=, Acceleration:=, Deceleration:=, Jerk:=, Vel=>, Done=>, Busy=>, CommandAborted=>, CommandHalt=>, Error=>, ErrorID=>);	LS_IpoLib

- **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	初始值	描述
Axis0	轴0	AXIS_REF_VIRTUAL_SM3	-	-	参与插补的0轴。
Axis1	轴1	AXIS_REF_VIRTUAL_SM3	-	-	参与插补的1轴。
Axis2	轴2	AXIS_REF_VIRTUAL_SM3	-	-	参与插补的2轴。
Axis3	轴3	AXIS_REF_VIRTUAL_SM3	-	-	参与插补的3轴。

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Exec	启动	BOOL	TRUE，FALSE	FALSE	直线插补模块启动命令，上升沿有效。
Halt	暂停	BOOL	TRUE，FALSE	FALSE	暂停命令，当该输入量为True时，当前插补运动暂停；当状态再次切换到False时，插补模块继续之前的未完成的插补任务。
Stop	停止	BOOL	TRUE，FALSE	FALSE	停止命令，该命令为True时，插补模块停止。
IpoCycle	插补周期	DWORD	遵循数据类型	2000	插补周期，单位：us。
PosMode	位置模式	INT	[0,1]	0	位置模式： 0 绝对模式； 1 相对模式。
Endpos	目标位置	ARRAY [0..1] OF REAL	负数，0，正数	0	轴的目标位置，单位：pulse。
VelocityMode	速度模式	SMC_INT_VELMOD	0-3	SIGMOID	速度模式，梯形：0（TRAPEZOID）；S形：1（SIGMOID）；四次方：3

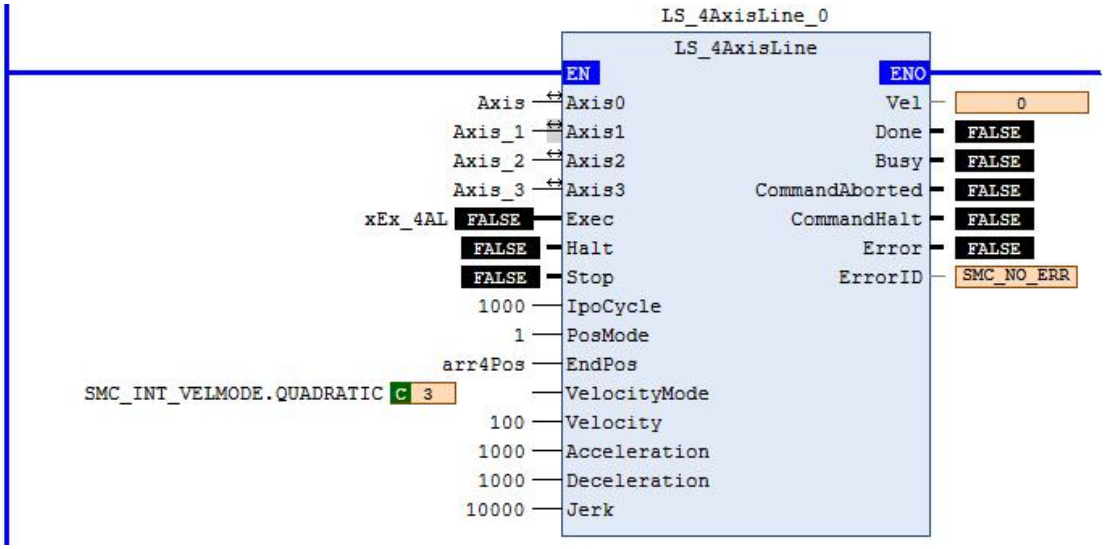
		E			（QUADRATIC）。
Velocity	速度	LREAL	0，正数	0	插补合成速度。
Acceleration	加速度	LREAL	0，正数	0	插补合成加速度。
Deceleration	减速度	LREAL	0，正数	0	插补合成减速度。
Jerk	加加速度	LREAL	负数，0 正数	90000000	加加速度，速度模式为3时候需要设置此参数，此参数值不能为0。

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
Vel	合速度	LREAL	0，正数	0	当前运动合成速度。
Done	完成	BOOL	TRUE， FALSE	FALSE	为 True 时，表示插补模块已完成插补任务。
Busy	运动中	BOOL	TRUE， FALSE	FALSE	为 True 时，表示插补运动正在进行之中。
Command Aborted	运动中断	BOOL	TRUE， FALSE	FALSE	当 Stop 有效时，该变量为 True，表示插补模块非正常中止。
Command Halt	运动暂停	BOOL	TRUE， FALSE	FALSE	当 Halt 为 True 时，该变量为 True，表示插补模块处于暂停状态。
Error	报错	BOOL	TRUE， FALSE	FALSE	错误状态。
ErrorID	错误码	DINT	0，正数	0	错误码。

- **程序示例**

  四轴以相对模式，直线插补运动到目标位置 1000，2000，3000，4000。变量定义：Arr4Pos: ARRAY [0..3] OF LREAL:=[1000,2000,3000,4000];。



- **注意事项**

➢ IpoCycle 参数需要与运动指令所在的任务周期时间保持一致，否则指令执行时可能会出现Error报错。

➢ 参数IpoCycle、Jerk、Velocity、Acceleration、Deceleration不可以设置为0。

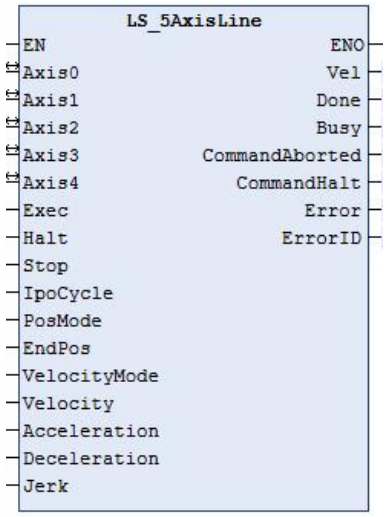
➢ 在指令运行期间，主从轴都不能再被其它运动指令所调用。

4.4.5 LS_5AxisLine

五轴直线插补指令。

- **指令格式**

指令	名称	FB/FC	LD表现	ST表现	库

LS_5AxisLine	五轴直线插补	FB		LS_5AxisLine(Axis0:=, Axis1:=, Axis2:=, Axis3:=, Axis4:=, Exec:=, Halt:=, Stop:=, IpoCycle:=, PosMode:=, EndPos:=, VelocityMode:=, Velocity:=, Acceleration:=, Deceleration:=, Jerk:=, Vel=>, Done=>, Busy=>, CommandAborted=>, CommandHalt=>, Error=>, ErrorID=>);	LS_IpoLib

- **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	初始值	描述
Axis0	轴0	AXIS_REF_VIRTUAL_SM3	-	-	参与插补的0轴。
Axis1	轴1	AXIS_REF_VIRTUAL_SM3	-	-	参与插补的1轴。
Axis2	轴2	AXIS_REF_VIRTUAL_SM3	-	-	参与插补的2轴。

Axis3	轴3	AXIS_REF_VIRTUAL_SM3	-	-	参与插补的3轴。
Axis4	轴4	AXIS_REF_VIRTUAL_SM3	-	-	参与插补的4轴。

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Exec	启动	BOOL	TRUE，FALSE	FALSE	直线插补模块启动命令，上升沿有效。
Halt	暂停	BOOL	TRUE，FALSE	FALSE	暂停命令，当该输入量为True时，当前插补运动暂停；当状态再次切换到False时，插补模块继续之前的未完成的插补任务。
Stop	停止	BOOL	TRUE，FALSE	FALSE	停止命令，该命令为True时，插补模块停止。
IpoCycle	插补周期	DWORD	遵循数据类型	2000	插补周期，单位：us。
PosMode	位置模式	INT	[0,1]	0	位置模式： 0 绝对模式； 1 相对模式。
Endpos	目标位置	ARRAY [0..1] OF REAL	负数，0，正数	0	轴的目标位置，单位：pulse。
VelocityMode	速度模式	SMC_INT_VELMODE	0-3	SIGMOID	速度模式，梯形：0（TRAPEZOID）；S形：1（SIGMOID）；四次方：3（QUADRATIC）。
Velocity	速度	LREAL	0，正数	0	插补合成速度。
Acceleration	加速度	LREAL	0，正数	0	插补合成加速度。
Deceleration	减速度	LREAL	0，正数	0	插补合成减速度。

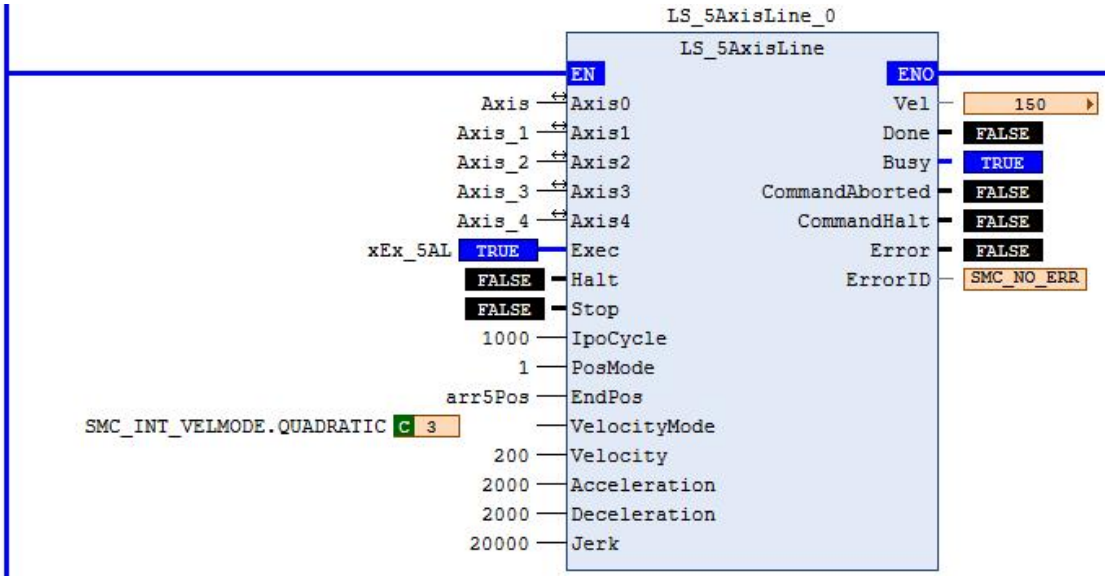
Jerk	加加速度	LREAL	负数，0 正数	90000000	加加速度，速度模式为3时候需要设置此参数，此参数值不能为0。

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
Vel	合速度	LREAL	0，正数	0	当前运动合成速度。
Done	完成	BOOL	TRUE， FALSE	FALSE	为 True 时，表示插补模块已完成插补任务。
Busy	运动中	BOOL	TRUE， FALSE	FALSE	为 True 时，表示插补运动正在进行之中。
Command Aborted	运动中断	BOOL	TRUE， FALSE	FALSE	当 Stop 有效时，该变量为 True，表示插补模块非正常中止。
Command Halt	运动暂停	BOOL	TRUE， FALSE	FALSE	当 Halt 为 True 时，该变量为 True，表示插补模块处于暂停状态。
Error	报错	BOOL	TRUE， FALSE	FALSE	错误状态。
ErrorID	错误码	DINT	0，正数	0	错误码。

- **程序示例**

五轴以相对模式，直线插补运行到目标位置 1000，2000，3000，4000，5000。变量定义：arr5Pos: ARRAY [0..4] OF LREAL:=[1000,2000,3000,4000,5000];。



- **注意事项**

➢ IpoCycle 参数需要与运动指令所在的任务周期时间保持一致，否则指令执行时可能会出

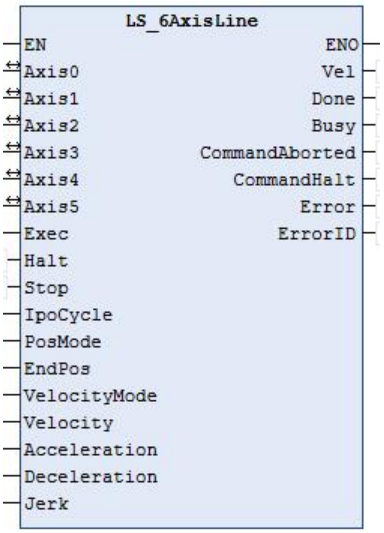
现Error报错。

- 参数IpoCycle、Jerk、Velocity、Acceleration、Deceleration不可以设置为0。
- 在指令运行期间，主从轴都不能再被其它运动指令所调用。

4.4.6 LS_6AxisLine

六轴直线插补指令。

- **指令格式**

指令	名称	FB/FC	LD表现	ST表现	库
LS_6AxisLine	六轴直线插补	FB		LS_6AxisLine(Axis0:=, Axis1:=, Axis2:=, Axis3:=, Axis4:=, Axis5:=, Exec:=, Halt:=, Stop:=, IpoCycle:=, PosMode:=, EndPos:=, VelocityMode:=, Velocity:=, Acceleration:=, Deceleration:=, Jerk:=, Vel=>, Done=>, Busy=>, CommandAborted=>, CommandHalt=>, Error=>, ErrorID=>);	LS_IpoLib

- **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	初始值	描述
Axis0	轴0	AXIS_REF_VIRTUAL_SM3	-	-	参与插补的0轴。
Axis1	轴1	AXIS_REF_VIRTUAL_SM3	-	-	参与插补的1轴。
Axis2	轴2	AXIS_REF_VIRTUAL_SM3	-	-	参与插补的2轴。
Axis3	轴3	AXIS_REF_VIRTUAL_SM3	-	-	参与插补的3轴。
Axis4	轴4	AXIS_REF_VIRTUAL_SM3	-	-	参与插补的4轴。
Axis5	轴5	AXIS_REF_VIRTUAL_SM3	-	-	参与插补的5轴。

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Exec	启动	BOOL	TRUE，FALSE	FALSE	直线插补模块启动命令，上升沿有效。
Halt	暂停	BOOL	TRUE，FALSE	FALSE	暂停命令，当该输入量为True时，当前插补运动暂停；当状态再次切换到False时，插补模块继续之前的未完成的插补任务。
Stop	停止	BOOL	TRUE，FALSE	FALSE	停止命令，该命令为True时，插补模块停止。
IpoCycle	插补周期	DWORD	遵循数据类型	2000	插补周期，单位：us。

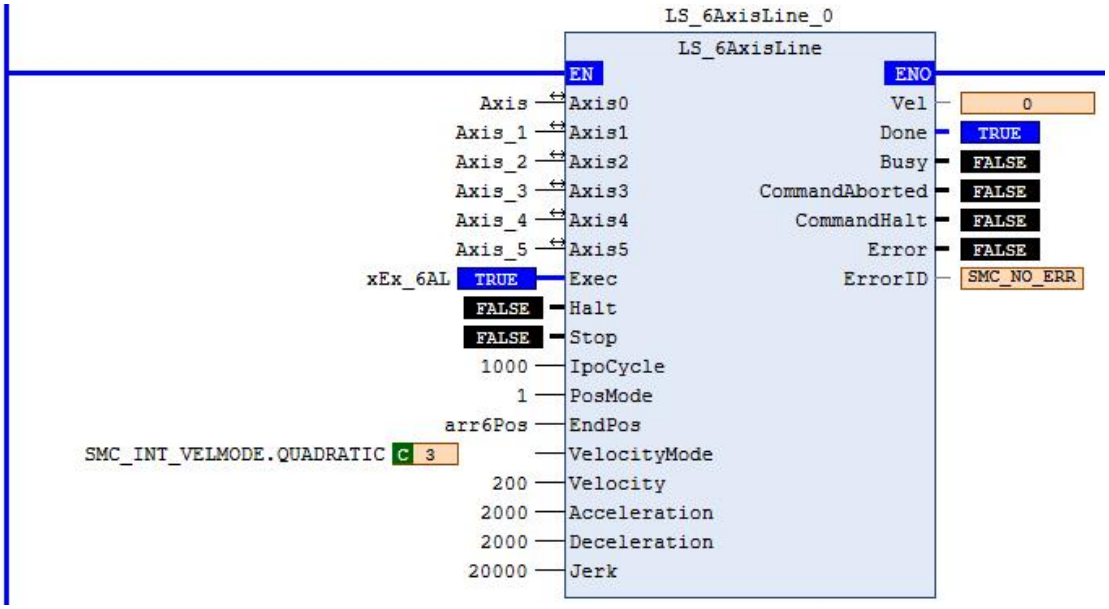
PosMode	位置模式	INT	[0,1]	0	位置模式： 0 绝对模式； 1 相对模式。
Endpos	目标位置	ARRAY [0..1] OF REAL	负数，0，正数	0	轴的目标位置，单位：pulse。
VelocityMode	速度模式	SMC_INT_VELMODE	0-3	SIGMOID	速度模式，梯形：0（TRAPEZOID）；S形：1（SIGMOID）；四次方：3（QUADRATIC）。
Velocity	速度	LREAL	0，正数	0	插补合成速度。
Acceleration	加速度	LREAL	0，正数	0	插补合成加速度。
Deceleration	减速度	LREAL	0，正数	0	插补合成减速度。
Jerk	加加速度	LREAL	负数，0 正数	90000000	加加速度，速度模式为3时候需要设置此参数，此参数值不能为0。

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
Vel	合速度	LREAL	0，正数	0	当前运动合成速度。
Done	完成	BOOL	TRUE， FALSE	FALSE	为 True 时，表示插补模块已完成插补任务。
Busy	运动中	BOOL	TRUE， FALSE	FALSE	为 True 时，表示插补运动正在进行之中。
CommandAborted	运动中断	BOOL	TRUE， FALSE	FALSE	当 Stop 有效时，该变量为 True，表示插补模块非正常中止。
CommandHalt	运动暂停	BOOL	TRUE， FALSE	FALSE	当 Halt 为 True 时，该变量为 True，表示插补模块处于暂停状态。
Error	报错	BOOL	TRUE， FALSE	FALSE	错误状态。
ErrorID	错误码	DINT	0，正数	0	错误码。

- **程序示例**

六轴以相对模式，直线插补运行到目标位置 1000，2000，3000，4000，5000，6000。变量定义：arr6Pos: ARRAY [0..5] OF LREAL:=[1000,2000,3000,4000,5000,6000];。



- **注意事项**

➢ IpoCycle 参数需要与运动指令所在的任务周期时间保持一致，否则指令执行时可能会出现Error报错。

➢ 参数IpoCycle、Jerk、Velocity、Acceleration、Deceleration不可以设置为0。

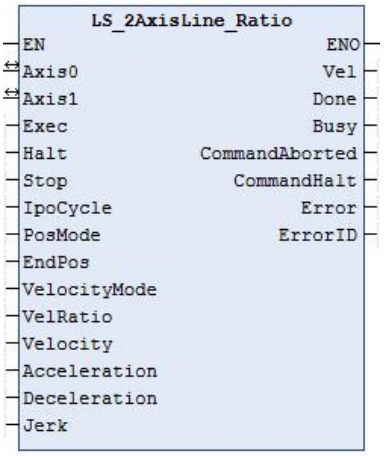
➢ 在指令运行期间，主从轴都不能再被其它运动指令所调用。

4.4.7 LS_2AxisLine_Ratio

两轴可调速直线插补指令。与 LS_2AxisLine 相比，增加了可调速功能，即在插补过程中可通过改变 VelRatio 的值来在线改变插补速度。

- **指令格式**

指令	名称	FB/FC	LD表现	ST表现	库

LS_2AxisLine_Ratio	两轴可调速直线插补	FB		LS_2AxisLine_Ratio(Axis0:= , Axis1:= , Exec:= , Halt:= , Stop:= , IpoCycle:= , PosMode:= , EndPos:= , VelocityMode:= , VelRatio:= , Velocity:= , Acceleration:= , Deceleration:= , Jerk:= , Vel=> , Done=> , Busy=> , CommandAborted=> , CommandHalt=> , Error=> , ErrorID=>);	LS_IpoLib

- **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	初始值	描述
Axis0	轴0	AXIS_REF_VIRTUAL_SM3	-	-	参与插补的0轴。
Axis1	轴1	AXIS_REF_VIRTUAL_SM3	-	-	参与插补的1轴。

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Exec	启动	BOOL	TRUE，FALSE	FALSE	直线插补模块启动命令，上升沿有效。

Halt	暂停	BOOL	TRUE，FALSE	FALSE	暂停命令，当该输入量为True时，当前插补运动暂停；当状态再次切换到False时，插补模块继续之前的未完成的插补任务。
Stop	停止	BOOL	TRUE，FALSE	FALSE	停止命令，该命令为True时，插补模块停止。
IpoCycle	插补周期	DWORD	-	2000	插补周期，单位：us。
PosMode	位置模式	INT	[0,1]	0	位置模式： 0 绝对模式； 1 相对模式。
Endpos	目标位置	ARRAY [0..2] OF REAL	负数，0，正数	0	轴的目标位置，单位：pulse。
VelocityMode	速度模式	SMC_INT_VELMODE	0-3	SIGMOID	速度模式，梯形：0（TRAPEZOID）；S形：1（SIGMOID）；四次方：3（QUADRATIC）。
VelRatio	速度倍率	LREAL	0.01-2	1	速度倍率，最小值0.01，最大值2。
Velocity	速度	LREAL	0，正数	0	插补合成速度。
Acceleration	加速度	LREAL	0，正数	0	插补合成加速度。
Deceleration	减速度	LREAL	0，正数	0	插补合成减速度。
Jerk	加加速度	LREAL	负数，0 正数	90000000	加加速度，速度模式为3时候需要设置此参数，此参数值不能为0。

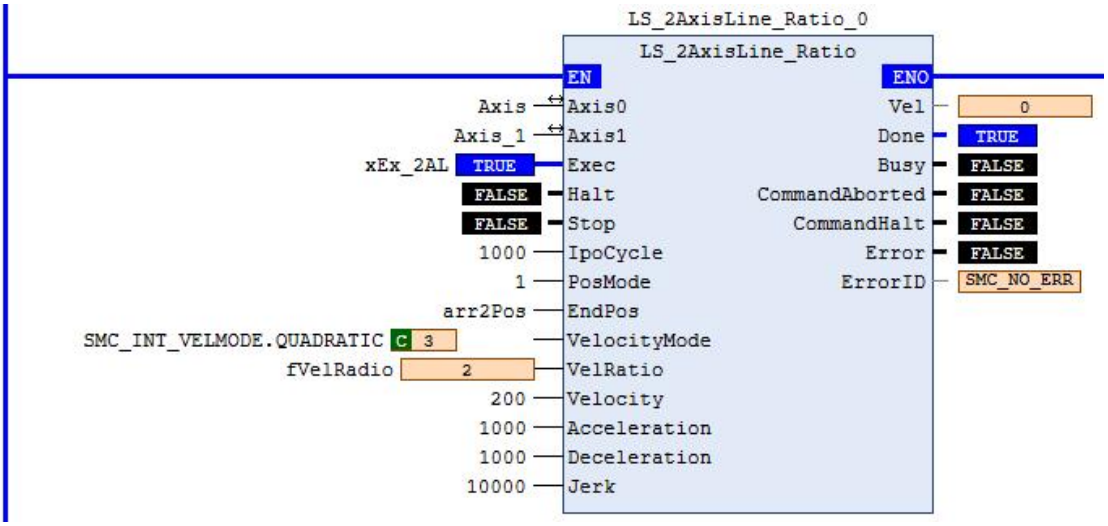
输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
Vel	合速度	LREAL	0，正数	0	当前运动合成速度。
Done	完成	BOOL	TRUE，FALSE	FALSE	为 True 时，表示插补模块已完成插补任务。
Busy	运动中	BOOL	TRUE，	FALSE	为 True 时，表示插补运动正在进

			FALSE		行之中。
Command Aborted	运动中断	BOOL	TRUE，FALSE	FALSE	当 Stop 有效时，该变量为 True，表示插补模块非正常中止。
Command Halt	运动暂停	BOOL	TRUE，FALSE	FALSE	当 Halt 为 True 时，该变量为 True，表示插补模块处于暂停状态。
Error	报错	BOOL	TRUE，FALSE	FALSE	错误状态。
ErrorID	错误码	DINT	0，正数	0	错误码。

- **程序示例**

两个轴以相对模式，插补速度为 2*200，直线插补运动到目标位置 2000，2000。



- **注意事项**

➢ IpoCycle 参数需要与运动指令所在的任务周期时间保持一致，否则指令执行时可能会出现Error报错。

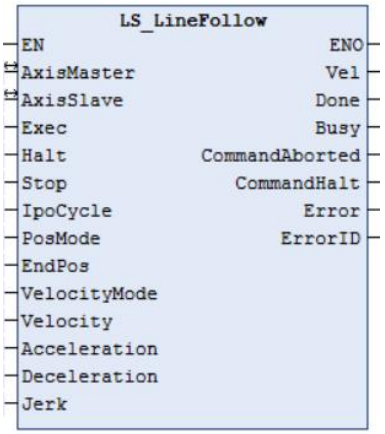
➢ 参数IpoCycle、Jerk、Velocity、Acceleration、Deceleration不可以设置为0。

➢ 在指令运行期间，主从轴都不能再被其它运动指令所调用。

4.4.8 LS_LineFollow

一轴点位，另一轴跟随运动指令。

- **指令格式**

指令	名称	FB/FC	LD表现	ST表现	库
LS_LineFollow	位置跟随	FB		LS_LineFollow(AxisMaster:= , AxisSlave:= , Exec:= , Halt:= , Stop:= , IpoCycle:= , PosMode:= , EndPos:= , VelocityMode:= , Velocity:= , Acceleration:= , Deceleration:= , Jerk:= , Vel=> , Done=> , Busy=> , CommandAborted=> , CommandHalt=> , Error=> , ErrorID=>);	LS_IpoLib

- **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	初始值	描述
AxisMaster	主轴	AXIS_REF_VIRTUAL_SM3	-	-	主轴
AxisSlave	从轴	AXIS_REF_VIRTUAL_SM3	-	-	跟随轴

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Exec	启动	BOOL	TRUE FALSE	FALSE	启动

Halt	暂停	BOOL	TRUE FALSE	FALSE	暂停
Stop	停止	BOOL	TRUE FALSE	FALSE	停止
IpoCycle	周期	DWORD	-	2000	周期，单位：us
PosMode	位置模式	INT	[0,1]	0	位置模式： 0 绝对模式； 1 相对模式。
EndPos	目标位置	ARRAY [0..1] OF LREAL	负数，0正数	0	轴的目标位置，单位：Pulse
VelocityMode	速度模式	SMC_INT_VELMODE	0-3	SIGMOID	速度模式，梯形：0（TRAPEZOID）；S形：1（SIGMOID）；四次方：3（QUADRATIC）
Velocity	速度	LREAL	负数，0正数	10	主轴速度 Pulse/s
Acceleration	加速度	LREAL	0，正数	10	主轴加速度 Pulse/s2
Deceleration	减速度	LREAL	0，正数	10	主轴减速度 Pulse/s2
Jerk	加加速度	LREAL	0，正数	90000000	加加速度 Pulse/s3，速度模式设置为3时需要设置该参数，该参数值不能为0

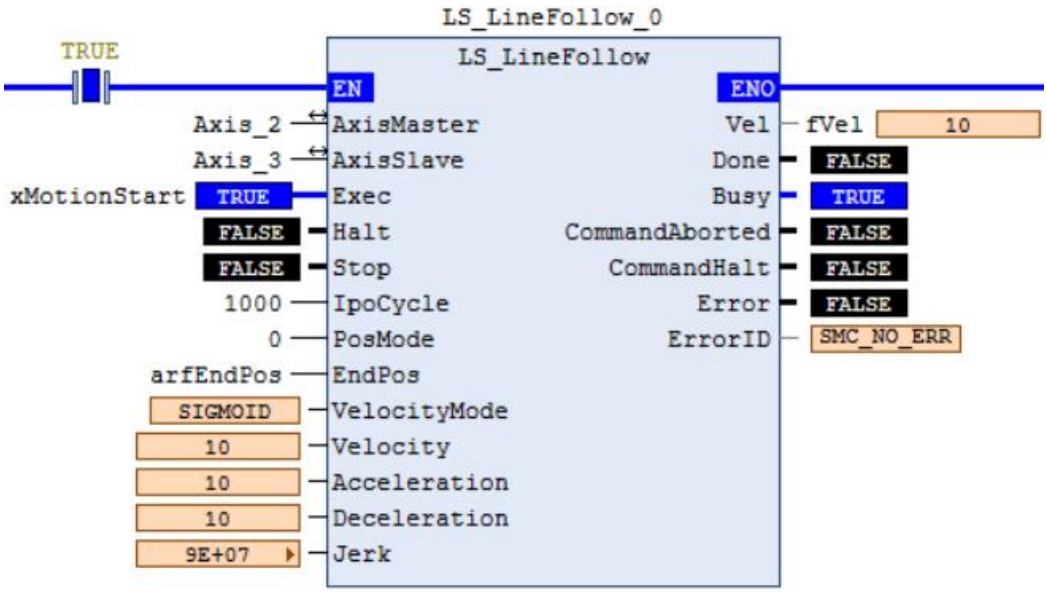
输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
Vel	合速度	LREAL	0，正数	0	当前运动合成速度。
Done	完成	BOOL	TRUE， FALSE	FALSE	为 True 时，表示插补模块已完成插补任务。
Busy	运动中	BOOL	TRUE， FALSE	FALSE	为 True 时，表示插补运动正在进行之中。
CommandAborted	运动中断	BOOL	TRUE， FALSE	FALSE	当 Stop 有效时，该变量为 True，表示插补模块非正常中止。

Command Halt	运动暂停	BOOL	TRUE，FALSE	FALSE	当 Halt 为 True 时，该变量为 True，表示插补模块处于暂停状态。
Error	报错	BOOL	TRUE，FALSE	FALSE	错误状态。
ErrorID	错误码	DINT	0，正数	0	错误码。

- **程序示例**

EndPos 设置[100,100]，主轴从轴同步运动到指定位置。



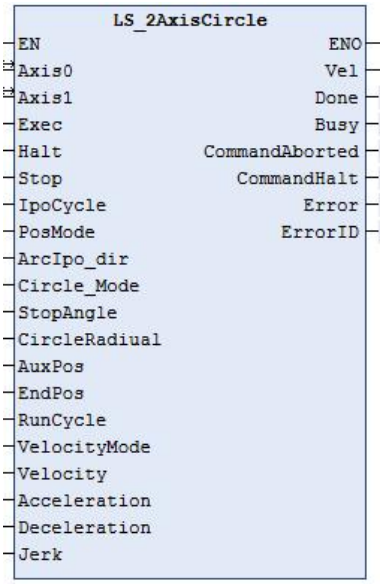
- **注意事项**

- IpoCycle 参数需要与运动指令所在的任务周期时间保持一致，否则指令执行时可能会出现Error报错。
- 参数IpoCycle、Jerk、Velocity、Acceleration、Deceleration不可以设置为0。
- 跟随运动指令仅在指令启动到结束这段时间有效，对主从轴有跟随控制效果，到达指令位置后，主从轴不再同步，都恢复standstill状态。
- 在指令运行期间，主从轴都不能再被其它运动指令所调用。

4.4.9 LS_2AxisCircle

两轴平面圆弧插补指令，支持三点圆弧、圆心圆弧、半径圆弧、角度圆弧四种模式。

- **指令格式**

指令	名称	FB/FC	LD表现	ST表现	库
LS_2AxisCircle	两轴圆弧插补	FB		LS_2AxisCircle(Axis0:= , Axis1:= , Exec:= , Halt:= , Stop:= , IpoCycle:= , PosMode:= , ArcIpo_dir:= , Circle_Mode:= , StopAngle:= , CircleRadiual:= , AuxPos:= , EndPos:= , RunCycle:= , VelocityMode:= , Velocity:= , Acceleration:= , Deceleration:= , Jerk:= , Vel=> , Done=> , Busy=> , CommandAborted=> , CommandHalt=> , Error=> , ErrorID=>);	LS_IpoLib

- **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	初始值	描述
Axis0	轴0	AXIS_REF_VIRTUAL_SM3	-	-	参与插补的0轴。

Axis1	轴1	AXIS_REF_VIRTUAL_SM3	-	-	参与插补的1轴。

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Exec	启动	BOOL	TRUE，FALSE	FALSE	平面圆弧插补模块启动命令，上升沿有效。
Halt	暂停	BOOL	TRUE，FALSE	FALSE	暂停命令，当该输入量为True时，当前插补运动暂停；当状态再次切换到False时，插补模块继续之前的未完成的插补任务。
Stop	停止	BOOL	TRUE，FALSE	FALSE	停止命令，该命令为True时，插补模块停止。
IpoCycle	插补周期	DWORD	-	2000	插补周期，单位：us。
PosMode	位置模式	INT	[0,1]	0	位置模式： 0 绝对模式； 1 相对模式。
ArcIpo_dir	插补方向	INT	0，1	0	圆弧插补的方向，0表示逆时针方向，1表示顺时针方向。圆弧模式1、2、3需要设置该值。
Circle_Mode	插补模式	INT	0-3	1	圆弧模式：0表示三点圆弧模式；1表示圆心位置与终点确定圆弧；2表示终点半径模式；3表示目标角度模式。
StopAngle	停止角度	REAL	0-360	0	停止的目标角度，相对于起点的角度，单位为度，取值范围为[0，360.0]；圆弧模式3需要设置该值。
CircleRadiual	圆弧半径	REAL	正数	0	圆弧半径，圆弧模式2需要设置该值。该值为负表示圆弧角小于180度，半径为正表示圆弧角大于180度。
AuxPos	辅助位置	ARRAY [0..1] OF REAL	负数，0 正数	0	辅助点，单位：Pulse。圆弧模式0为三点圆弧的经过点。圆弧模式1、3为圆心位置。

Endpos	停止位置	ARRAY [0..1] OF REAL	负数，0 正数	0	终点位置，单位：Pulse。
RunCycle	循环次数	UINT	0，正数	0	圆弧运动的循环次数（即圆弧旋转的圈数）。
VelocityMode	速度模式	SMC_INT_VELMODE	0-3	SIGMOID	速度模式，梯形：0（TRAPEZOID）；S形：1（SIGMOID）；四次方：3（QUADRATIC）。
Velocity	速度	LREAL	0，正数	0	插补合成速度。
Acceleration	加速度	LREAL	0，正数	0	插补合成加速度。
Deceleration	减速度	LREAL	0，正数	0	插补合成减速度。
Jerk	加加速度	LREAL	负数，0 正数	90000000	加加速度，速度模式为3时候需要设置此参数，此参数值不能为0。

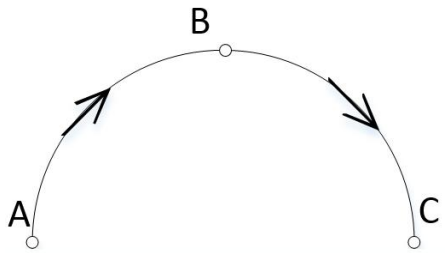
输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
Vel	合速度	LREAL	0，正数	0	当前运动合成速度。
Done	完成	BOOL	TRUE，FALSE	FALSE	为 True 时，表示插补模块已完成插补任务。
Busy	运动中	BOOL	TRUE，FALSE	FALSE	为 True 时，表示插补运动正在进行之中。
CommandAborted	运动中断	BOOL	TRUE，FALSE	FALSE	当 Stop 有效时，该变量为 True，表示插补模块非正常中止。
CommandHalt	运动暂停	BOOL	TRUE，FALSE	FALSE	当 Halt 为 True 时，该变量为 True，表示插补模块处于暂停状态。
Error	报错	BOOL	TRUE，FALSE	FALSE	错误状态。
ErrorID	错误码	DINT	0，正数	0	错误码。

- **功能说明**

  - “Circle_Mode”参数用来设置圆弧插补的模式，分别如下：

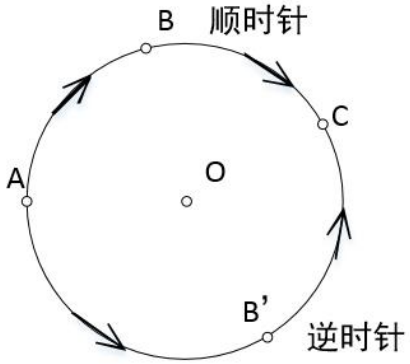
1) Circle_Mode=0，表示三点圆弧模式，如图所示。



三点圆弧插补

在使用三点圆弧模式时，需要在圆弧轨迹上确定 3 个不重合的点，如图中的 A、B、C 三点。在指令中，A 代表圆弧插补轴 0 和 1 的当前位置，不需要设置；B 代表圆弧经过的点，需要设置为参数 AuxPos；C 代表圆弧的终点位置，需要设置为参数 Endpos。

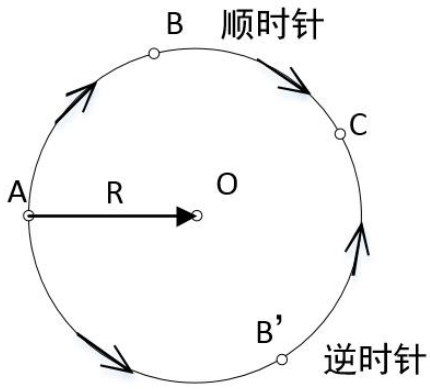
2) Circle_Mode=1，表示圆心与终点模式的圆弧，如图所示。



圆心终点圆弧插补模式

在使用圆心终点圆弧模式时，需要设置圆心 O、终点 C 和运动方向等参数。在指令中 O 代表圆心坐标，需要设置为参数 AuxPos；C 代表圆弧的终点坐标，需要设置为参数 Endpos；运动方向需要设置参数 ArcIpo_dir。

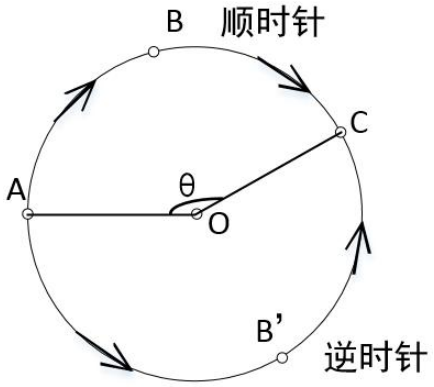
3) Circle_Mode=2，表示终点半径模式，如图所示。



终点半径圆弧插补模式

在使用终点半径模式时，需要设置终点 C，半径 R 和运动方向等参数。在指令中 C 代表圆弧的终点位置，需要设置为参数 Endpos；半径 R 需要设置为参数 CircleRadiual；运动方向需要设置为参数 ArcIpo_dir。

4）Circle_Mode=3，表示目标角度模式，如图所示。



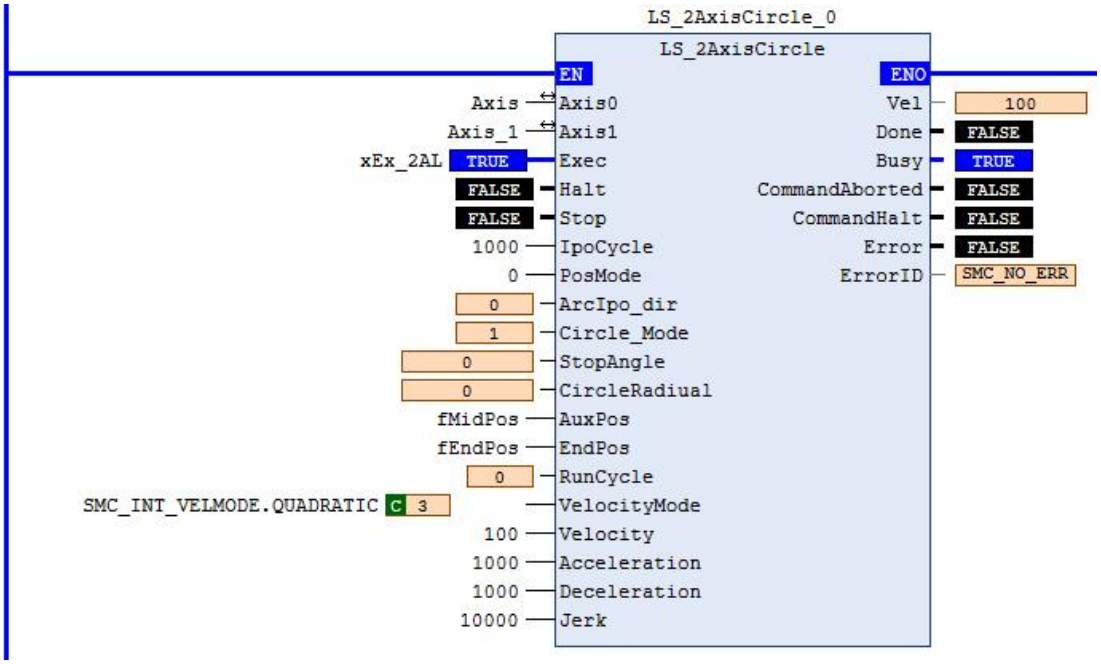
目标角度圆弧插补模式

在使用目标角度模式时，需要设置圆心 O，角度 θ 和运动方向等参数。在指令中 O 代表圆心位置，需要设置为参数 Auxpos；目标角度 θ 需要设置为参数 StopAngle；运动方向需要设置为参数 ArcIpo_dir。

- **程序示例**

两轴平面圆弧执行三点画半圆，起始点为 0，0，三点圆弧经过点为 1000，1000，目标点为 2000，0。

fMidPos	ARRAY [0..1] OF LREAL	
fMidPos[0]	LREAL	1000
fMidPos[1]	LREAL	1000
fEndPos	ARRAY [0..1] OF LREAL	
fEndPos[0]	LREAL	2000
fEndPos[1]	LREAL	0



- **注意事项**

➢ IpoCycle 参数需要与运动指令所在的任务周期时间保持一致，否则指令执行时可能会出现Error报错。

➢ 参数IpoCycle、Jerk、Velocity、Acceleration、Deceleration不可以设置为0。

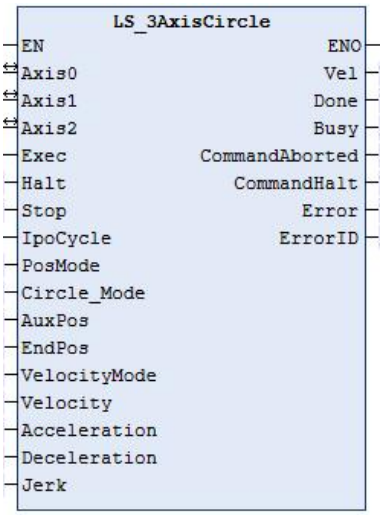
➢ 在指令运行期间，主从轴都不能再被其它运动指令所调用。

4.4.10 LS_3AxisCircle

三轴空间圆弧插补指令，仅支持三点圆弧模式。

- **指令格式**

指令	名称	FB/FC	LD表现	ST表现	库

LS_3AxisCircle	三轴圆弧插补	FB		LS_2AxisCircle(Axis0:= , Axis1:= , Axis2:= , Exec:= , Halt:= , Stop:= , IpoCycle:= , PosMode:= , Circle_Mode:= , AuxPos:= , EndPos:= , VelocityMode:= , Velocity:= , Acceleration:= , Deceleration:= , Jerk:= , Vel=> , Done=> , Busy=> , CommandAborted=> , CommandHalt=> , Error=> , ErrorID=>);	LS_IpoLib

- **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	初始值	描述
Axis0	轴0	AXIS_REF_VIRTUAL_SM3	-	-	参与插补的0轴。
Axis1	轴1	AXIS_REF_VIRTUAL_SM3	-	-	参与插补的1轴。
Axis2	轴2	AXIS_REF_VIRTUAL_SM3	-	-	参与插补的2轴。

输入变量

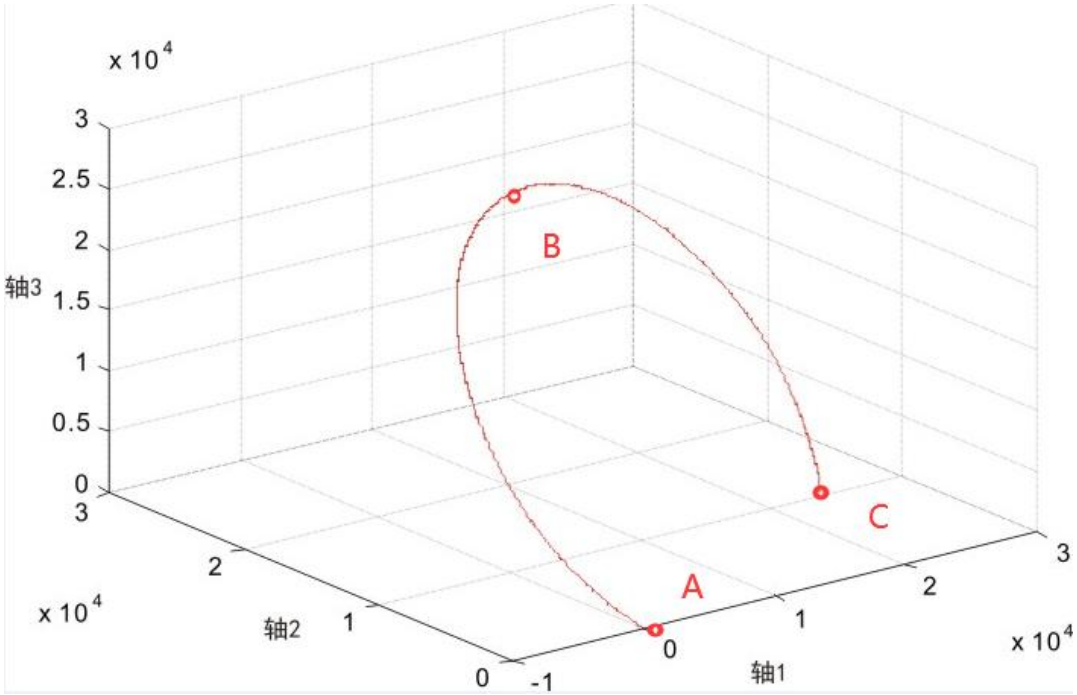
输入变量	名称	数据类型	有效范围	初始值	描述
Exec	启动	BOOL	TRUE，FALSE	FALSE	空间圆弧插补模块启动命令，上升沿有效。
Halt	暂停	BOOL	TRUE，FALSE	FALSE	暂停命令，当该输入量为True时，当前插补运动暂停；当状态再次切换到False时，插补模块继续之前的未完成的插补任务。
Stop	停止	BOOL	TRUE，FALSE	FALSE	停止命令，该命令为True时，插补模块停止。
IpoCycle	插补周期	DWORD	-	2000	插补周期，单位：us。
PosMode	位置模式	INT	[0,1]	0	位置模式： 0 绝对模式； 1 相对模式。
Circle_Mode	插补模式	INT	0	0	圆弧模式：0表示三点圆弧模式，仅支持该模式。
AuxPos	辅助位置	ARRAY [0..2] OF REAL	负数，0 正数	0	辅助点，单位：Pulse。圆弧模式0为三点圆弧的经过点。圆弧模式1、3为圆心位置。
Endpos	停止位置	ARRAY [0..2] OF REAL	负数，0 正数	0	终点位置，单位：Pulse。
VelocityMode	速度模式	SMC_INT_VELMODE	0-3	SIGMOID	速度模式，梯形：0（TRAPEZOID）；S形：1（SIGMOID）；四次方：3（QUADRATIC）。
Velocity	速度	LREAL	0，正数	0	插补合成速度。
Acceleration	加速度	LREAL	0，正数	0	插补合成加速度。
Deceleration	减速度	LREAL	0，正数	0	插补合成减速度。
Jerk	加加速度	LREAL	负数，0 正数	90000000	加加速度，速度模式为3时候需要设置此参数，此参数值不能为0。

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
Vel	合速度	LREAL	0，正数	0	当前运动合成速度。
Done	完成	BOOL	TRUE，FALSE	FALSE	为 True 时，表示插补模块已完成插补任务。
Busy	运动中	BOOL	TRUE，FALSE	FALSE	为 True 时，表示插补运动正在进行之中。
Command Aborted	运动中断	BOOL	TRUE，FALSE	FALSE	当 Stop 有效时，该变量为 True，表示插补模块非正常中止。
Command Halt	运动暂停	BOOL	TRUE，FALSE	FALSE	当 Halt 为 True 时，该变量为 True，表示插补模块处于暂停状态。
Error	报错	BOOL	TRUE，FALSE	FALSE	错误状态。
ErrorID	错误码	DINT	0，正数	0	错误码。

- **功能说明**

  - “Circle_Mode”参数用来设置三轴圆弧的圆弧模式，默认模式为三点圆弧，对应参数值为0，且仅支持这种圆弧模式，圆弧轨迹如图所示，A点为轴0、1、2的当前位置；B点位圆弧经过的点，C点为圆弧的终点。



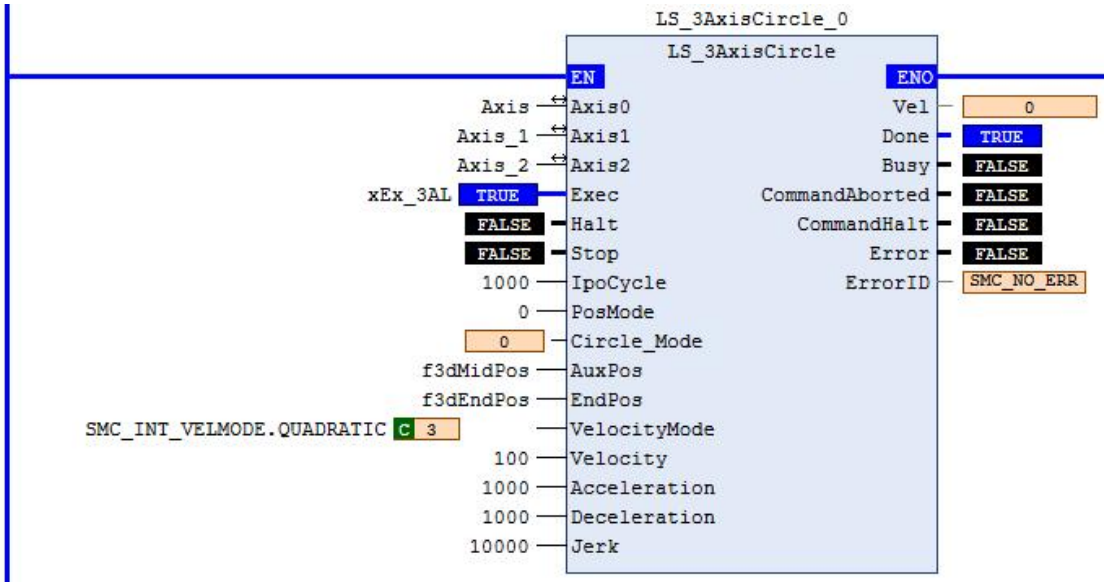
空间三轴圆弧插补轨迹

- **程序示例**

三轴空间圆弧执行三点画圆弧，起始点为 0，0，0，三点圆弧经过点 1000，1000，100，

目标点 2000，0，0。

表达式	类型	值
f3dMidPos	ARRAY [0..2] OF LREAL	
f3dMidPos[0]	LREAL	1000
f3dMidPos[1]	LREAL	1000
f3dMidPos[2]	LREAL	100
f3dEndPos	ARRAY [0..2] OF LREAL	
f3dEndPos[0]	LREAL	2000
f3dEndPos[1]	LREAL	0
f3dEndPos[2]	LREAL	0



- **注意事项**

➢ IpoCycle 参数需要与运动指令所在的任务周期时间保持一致，否则指令执行时可能会出现Error报错。

➢ 参数IpoCycle、Jerk、Velocity、Acceleration、Deceleration不可以设置为0。

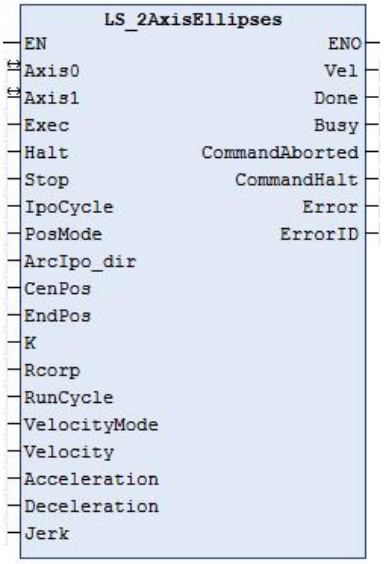
➢ 在指令运行期间，主从轴都不能再被其它运动指令所调用。

4.4.11 LS_2AxisEllipses

两轴椭圆插补指令。

- **指令格式**

指令	名称	FB/FC	LD表现	ST表现	库

LS_2AxisEllipses	两轴椭圆插补	FB		LS_2AxisEllipses(Axis0:= , Axis1:= , Exec:= , Halt:= , Stop:= , IpoCycle:= , PosMode:= , ArcIpo_dir:= , CenPos:= , EndPos:= , K:= , Rcorp:= , RunCycle:= , VelocityMode:= , Velocity:= , Acceleration:= , Deceleration:= , Jerk:= , Vel=> , Done=> , Busy=> , CommandAborted=> , CommandHalt=> , Error=> , ErrorID=>);	LS_IpoLib

- **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	初始值	描述
Axis0	轴0	AXIS_REF_VIRTUAL_SM3	-	-	参与插补的0轴。
Axis1	轴1	AXIS_REF_VIRTUAL_SM3	-	-	参与插补的1轴。

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述

Exec	启动	BOOL	TRUE，FALSE	FALSE	椭圆插补模块启动命令，上升沿有效。
Halt	暂停	BOOL	TRUE，FALSE	FALSE	暂停命令，当该输入量为True时，当前插补运动暂停；当状态再次切换到False时，插补模块继续之前的未完成的插补任务。
Stop	停止	BOOL	TRUE，FALSE	FALSE	停止命令，该命令为True时，插补模块停止。
IpoCycle	插补周期	DWORD	-	2000	插补周期，单位：us。
PosMode	位置模式	INT	[0,1]	0	位置模式： 0 绝对模式； 1 相对模式。
ArcIpo_dir	插补方向	INT	0，1	0	椭圆插补的方向：0表示逆时针方向；1表示顺时针方向。
CenPos	中心位置	ARRAY [0..1] OF REAL	负数，0正数	0	椭圆的中心位置，绝对坐标，单位：Pulse。
Endpos	终点位置	ARRAY [0..1] OF REAL	负数，0正数	0	终点位置，绝对坐标，单位：Pulse。
K	主轴角度	REAl	0-360	0	主轴的角度，主轴相对于X轴的角度；长轴与X轴的夹角，用来确定主轴的方向。
Rcorp	短轴长轴比	REAL	0，1	0	短轴与长轴的比，取值范围为[0，1]。
RunCycle	循环次数	UINT	0，正数	0	椭圆运动的循环次数（即椭圆的圈数）。
VelocityMode	速度模式	SMC_INT_VELMODE	0-3	SIGMOID	速度模式，梯形：0（TRAPEZOID）；S形：1（SIGMOID）；四次方：3（QUADRATIC）。
Velocity	速度	LREAL	0，正数	0	插补合成速度。
Acceleration	加速度	LREAL	0，正数	0	插补合成加速度。

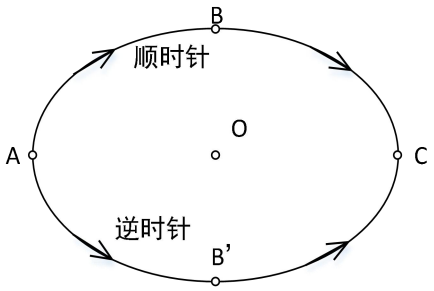
Deceleration	减速度	LREAL	0，正数	0	插补合成减速度。
Jerk	加加速度	LREAL	负数，0 正数	90000000	加加速度，速度模式为3时候需要设置此参数，此参数值不能为0。

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
Vel	合速度	LREAL	0，正数	0	当前运动合成速度。
Done	完成	BOOL	TRUE， FALSE	FALSE	为 True 时，表示插补模块已完成插补任务。
Busy	运动中	BOOL	TRUE， FALSE	FALSE	为 True 时，表示插补运动正在进行之中。
Command Aborted	运动中断	BOOL	TRUE， FALSE	FALSE	当 Stop 有效时，该变量为 True，表示插补模块非正常中止。
Command Halt	运动暂停	BOOL	TRUE， FALSE	FALSE	当 Halt 为 True 时，该变量为 True，表示插补模块处于暂停状态。
Error	报错	BOOL	TRUE， FALSE	FALSE	错误状态。
ErrorID	错误码	DINT	0，正数	0	错误码。

- **功能说明**

  - 指令指定的两个轴做椭圆插补运动，椭圆插补的轨迹，如图所示。



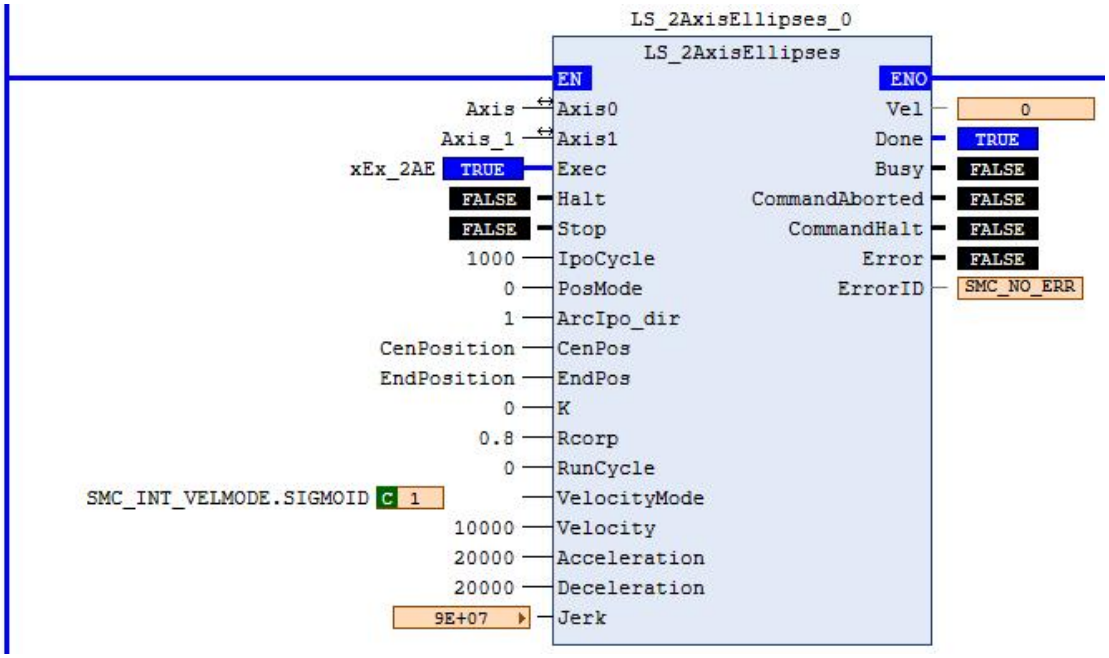
椭圆插补

  - 在使用椭圆插补运动时，需要设置圆心O、终点C和运动方向等参数。在指令中，A代表圆弧插补轴0和1的当前位置，不需要设置；O代表圆心位置，需要设置为参数CenPos；C代表圆弧的终点位置，需要设置为参数Endpos；运动方向需要设置为参数ArcIpo_dir；椭圆轨迹执行圈数，则需要设置为参数RunCycle。

- **程序示例**

XY 轴椭圆插补运动，当前位置（0，0）顺时针运动到（10000，0），X 轴为长轴，长 10000，Y 为短轴，长 8000，中心点（5000，0），单位为 Pulse，运行速度 10000Pulse/s，加减速度 20000Pulse/s^2 。

表达式	类型	值
CenPosition	ARRAY [0..1] OF LREAL	
CenPosition[0]	LREAL	5000
CenPosition[1]	LREAL	0
EndPosition	ARRAY [0..1] OF LREAL	
EndPosition[0]	LREAL	10000
EndPosition[1]	LREAL	0



- **注意事项**

➢ IpoCycle 参数需要与运动指令所在的任务周期时间保持一致，否则指令执行时可能会出现Error报错。

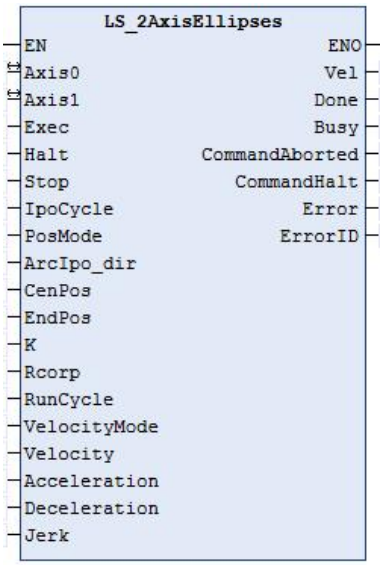
➢ 参数IpoCycle、Jerk、Velocity、Acceleration、Deceleration不可以设置为0。

➢ 在指令运行期间，主从轴都不能再被其它运动指令所调用。

4.4.12 LS_2AxisCircle_Helical

螺旋插补指令。

- **指令格式**

指令	名称	FB/FC	LD表现	ST表现	库
LS_2AxisCircle_Helical	螺旋线插补	FB		LS_2AxisCircle_Helical(Axis0:= , Axis1:= , Axis2:= , Exec:= , Halt:= , Stop:= , IpoCycle:= , PosMode:= , ArcIpo_dir:= , Circle_Mode:= , StopAngle:= , CircleRadiual:= , AuxPos:= , EndPos:= , HelicalPos:= , RunCycle:= , VelocityMode:= , Velocity:= , Acceleration:= , Deceleration:= , Jerk:= , Vel=> , Done=> , Busy=> , CommandAborted=> , CommandHalt=> , Error=> , ErrorID=>);	LS_IpoLib

- **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	初始值	描述
Axis0	轴0	AXIS_REF_VIRTUAL_SM3	-	-	参与插补的0轴。
Axis1	轴1	AXIS_REF_VIRTUAL_	-	-	参与插补的1轴。

		SM3			

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Exec	启动	BOOL	TRUE，FALSE	FALSE	螺旋线插补模块启动命令，上升沿有效。
Halt	暂停	BOOL	TRUE，FALSE	FALSE	暂停命令，当该输入量为True时，当前插补运动暂停；当状态再次切换到False时，插补模块继续之前的未完成的插补任务。
Stop	停止	BOOL	TRUE，FALSE	FALSE	停止命令，该命令为True时，插补模块停止。
IpoCycle	插补周期	DWORD	-	2000	插补周期，单位：us。
PosMode	位置模式	INT	[0,1]	0	位置模式： 0 绝对模式； 1 相对模式。
ArcIpo_dir	插补方向	INT	0，1	0	圆弧插补的方向：0表示逆时针方向；1表示顺时针方向。圆弧模式1、2、3需要设置该值。
Circle_Mode	插补模式	INT	0-3	1	圆弧模式：0表示三点圆弧模式；1表示圆心位置与终点确定圆弧；2表示终点半径模式；3表示目标角度模式。
StopAngle	停止角度	REAL	0-360	0	停止的目标角度，相对于起点的角度，单位为度，取值范围为[0，360.0]。圆弧模式3需要设置该值。
CircleRadiual	圆弧半径	REAL	正数	0	圆弧半径，圆弧模式2需要设置该值。该值半径为负表示大圆弧，半径为正表示小圆弧。
AuxPos	辅助位置	ARRAY [0..1] OF REAL	负数，0 正数	0	辅助点，单位：Pulse。
Endpos	终点位置	ARRAY [0..1] OF REAL	负数，0正数	0	终点位置，绝对坐标，单位：Pulse。

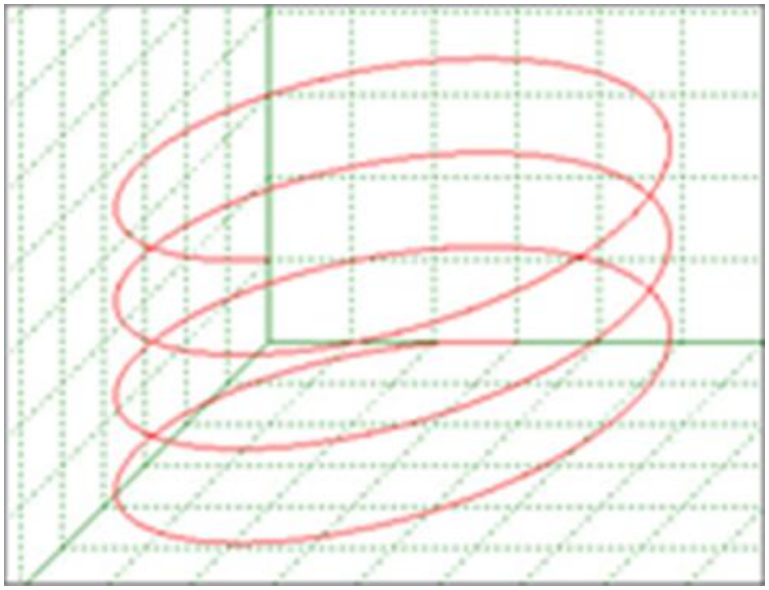
HelicalPos	螺旋轴位置	LREAl	负数，0 正数	0	螺旋轴位置，单位：Pulse。
RunCycle	循环次数	UINT	0，正数	0	圆弧运动的循环次数（即圆弧旋转的圈数）。
VelocityMode	速度模式	SMC_INT_VELMODE	0-3	SIGMOID	速度模式，梯形：0（TRAPEZOID）；S形：1（SIGMOID）；四次方：3（QUADRATIC）。
Velocity	速度	LREAL	0，正数	0	插补合成速度。
Acceleration	加速度	LREAL	0，正数	0	插补合成加速度。
Deceleration	减速度	LREAL	0，正数	0	插补合成减速度。
Jerk	加加速度	LREAL	负数，0 正数	90000000	加加速度，速度模式为3时候需要设置此参数，此参数值不能为0。

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
Vel	合速度	LREAL	0，正数	0	当前运动合成速度。
Done	完成	BOOL	TRUE，FALSE	FALSE	为 True 时，表示插补模块已完成插补任务。
Busy	运动中	BOOL	TRUE，FALSE	FALSE	为 True 时，表示插补运动正在进行之中。
CommandAborted	运动中断	BOOL	TRUE，FALSE	FALSE	当 Stop 有效时，该变量为 True，表示插补模块非正常中止。
CommandHalt	运动暂停	BOOL	TRUE，FALSE	FALSE	当 Halt 为 True 时，该变量为 True，表示插补模块处于暂停状态。
Error	报错	BOOL	TRUE，FALSE	FALSE	错误状态。
ErrorID	错误码	DINT	0，正数	0	错误码。

- **功能说明**

  - 指令指定的三个轴做空间螺旋插补运动。螺旋插补的轨迹，如图所示。



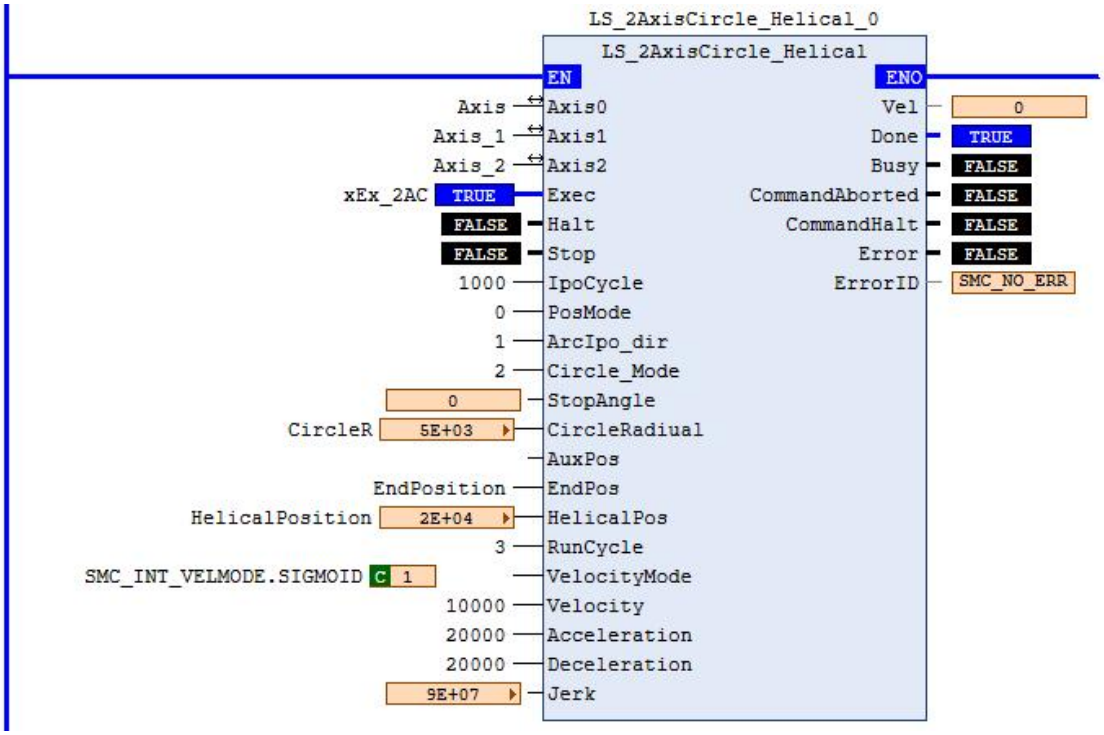
螺旋插补轨迹

- 在螺旋插补运动之中，轴0和轴1用于实现圆弧运动，“Circle_Mode”参数用来设置轴0、轴1的圆弧插补的模式，定义与两轴圆弧插补指令相同。

- **程序示例**

XYZ轴做三轴圆弧螺旋线插补运动。其中XY轴做圆弧插补，采用模式2 — 终点半径模式，半径5000，终点（10000，0），Z轴做跟随运动，距离为10000，单位为Pulse，圆弧循环次数为3次，设置运行速度10000Pulse/s，加减速度20000Pulse/s^2。

表达式	类型	值
CircleR	LREAL	5000
EndPosition	ARRAY [0..1] OF LREAL	
EndPosition[0]	LREAL	10000
EndPosition[1]	LREAL	0
HelicalPosition	LREAL	20000



- **注意事项**

➢ IpoCycle 参数需要与运动指令所在的任务周期时间保持一致，否则指令执行时可能会出现Error报错。

➢ 参数IpoCycle、Jerk、Velocity、Acceleration、Deceleration不可以设置为0。

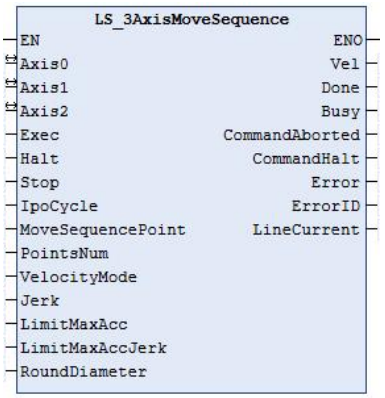
➢ 在指令运行期间，主从轴都不能再被其它运动指令所调用。

4.4.13 LS_3AxisMoveSequence

三轴连续插补指令，速度连续。

- **指令格式**

指令	名称	FB/FC	LD表现	ST表现	库

LS_3Axis MoveSequence	三轴连续插补	FB		LS_3AxisMoveSequence(Axis0:= , Axis1:= , Axis2:= , Exec:= , Halt:= , Stop:= , IpoCycle:= , MoveSequencePoint:= , PointsNum:= , VelocityMode:= , Jerk:= , LimitMaxAcc:= , LimitMaxAccJerk:= , RoundDiameter:= , Vel=> , Done=> , Busy=> , CommandAborted=> , CommandHalt=> , Error=> , ErrorID=> , LineCurrent=>);	LS_IpoLib

- **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	初始值	描述
Axis0	轴0	AXIS_REF_VIRTUAL_SM3	-	-	参与插补的0轴。
Axis1	轴1	AXIS_REF_VIRTUAL_SM3	-	-	参与插补的1轴。
Axis2	轴2	AXIS_REF_VIRTUAL_SM3	-	-	参与插补的2轴。

- **输入变量**

输入变量	名称	数据类型	有效范围	初始值	描述

Exec	启动	BOOL	TRUE，FALSE	FALSE	连续插补模块启动命令，上升沿有效。
Halt	暂停	BOOL	TRUE，FALSE	FALSE	暂停命令，当该输入量为True时，当前插补运动暂停；当状态再次切换到False时，插补模块继续之前的未完成的插补任务。
Stop	停止	BOOL	TRUE，FALSE	FALSE	停止命令，该命令为True时，插补模块停止。
IpoCycle	插补周期	DWORD	-	2000	插补周期，单位：us。
MoveSequenceePoint	插补数据表	ARRAY [0..MoveSequence_MAX] OF MoveSequence	-	-	输入的数据点。
PointsNum	点数	UINT	正数	0	输入的数据点数。
VelocityMode	速度模式	SMC_INT_VELMODE	0，1，3	1	速度模式，0： 梯形（TRAPEZOID）；1：S型曲线（SIGMOID）；3：四次方（QUADRATIC）。
Jerk	加加速度	REAL	正数	90000000	加加速度，速度模式为3时候需要设置此参数。
LimitMaxAcc	最大加速度	REAL	正数	9000000	拐角过渡圆弧最大加速度限制。
LimitMaxAccJerk	最大减速度	REAL	正数	90000000	拐角过渡圆弧最大加加速度限制。
RoundDiameter	拐角半径	REAL	正数	100	拐角过渡圆弧半径，单位：Pulse。

- **输出变量**

输出变量	名称	数据类型	有效范围	初始值	描述
Vel	合速度	LREAL	0，正数	0	当前运动合成速度。
Done	完成	BOOL	TRUE，	FALSE	为 True 时，表示插补模块已完成

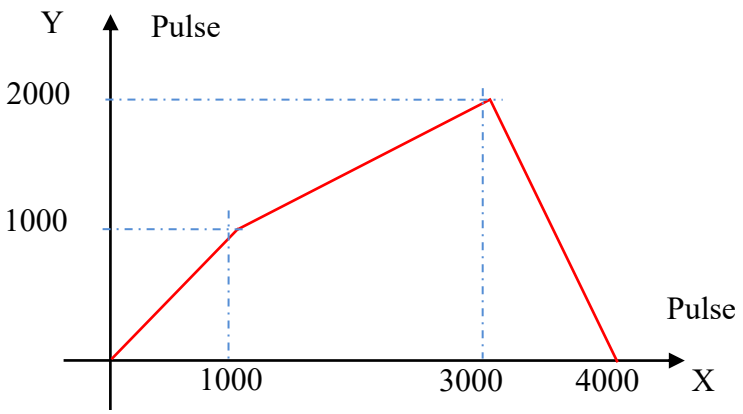
			FALSE		插补任务。
Busy	运动中	BOOL	TRUE，FALSE	FALSE	为 True 时，表示插补运动正在进行之中。
Command Aborted	运动中断	BOOL	TRUE，FALSE	FALSE	当 Stop 有效时，该变量为 True，表示插补模块非正常中止。
Command Halt	运动暂停	BOOL	TRUE，FALSE	FALSE	当 Halt 为 True 时，该变量为 True，表示插补模块处于暂停状态。
Error	报错	BOOL	TRUE，FALSE	FALSE	错误状态。
ErrorID	错误码	DINT	0，正数	0	错误码。
LineCuren	行号	UINT	遵循数据类型	0	当前执行的行号。

- **功能说明**

  - 指定3个轴做连续插补运动，位置坐标为绝对坐标，一次最多可以输入200个插补运动点。
  - 在连续插补运动中，每个运动点位，都是以MoveSequence 结构体的形式存在。
  - 当选择的插补类型为直线插补时，如果相邻两段插补运动的轨迹相切或拐角时，规划出来的合速度便会是连续的，如果拐角太大，则速度不连续；当选择的插补类型为圆弧插补时，相邻两段插补运动的轨迹在交接点处如果是相切的，则速度连续，如果不相切时，则速度不连续。
  - 启动连续插补指令后，指令内部需要先进行压缓存的操作，插补点数越多，压缓存的时间会越长。

- **程序示例**

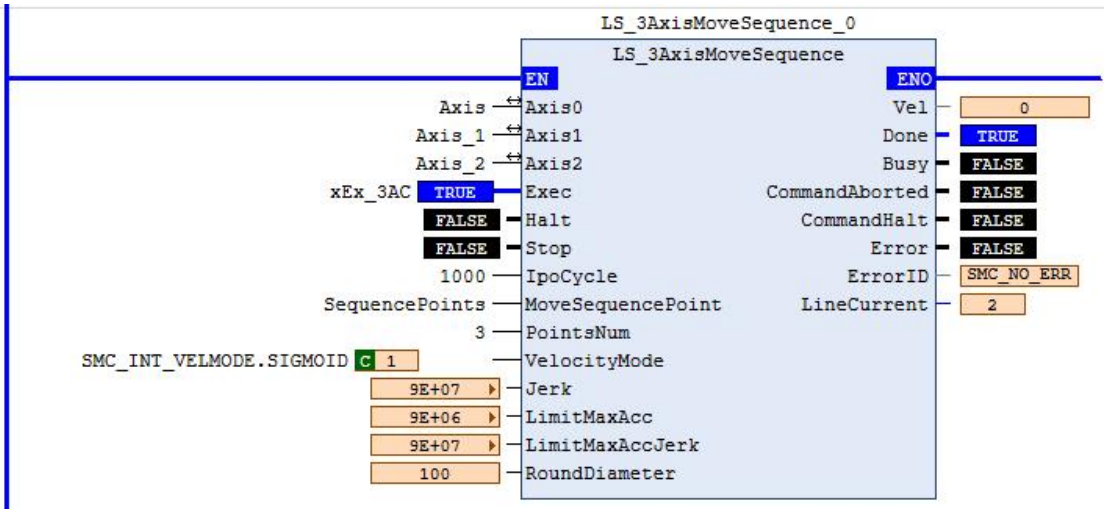
编写程序用连续插补指令实现 XYZ 轴进行三段直线连续插补，速度均为 1000Pulse/s，加减速度为 2000Pulse/s^2，第一段终点坐标为（1000，1000，0），第二段终点坐标为（3000，2000，0），第三段终点坐标为（4000，0，0），如图所示。



三段直线的连续插补轨迹

在主程序中添加数据点的相关参数。程序完成后，编译下载到控制器中执行，并将变量 iState 强制改为 1 启动连续插补。

```
CASE iState[0] OF
    1://配置连续插补数据点参数
        FOR i[0]:=0 TO 5 DO
        SequencePoints[i[0]].Acc[0]:=20000;
        SequencePoints[i[0]].DataType[0]:=1;
        SequencePoints[i[0]].LineNum[0]:=i[0];
        SequencePoints[i[0]].Vel[0]:=10000;
        SequencePoints[i[0]].Z_EndPos[0]:=0;
        END_FOR
        SequencePoints[0].X_EndPos[0]:=1000;
        SequencePoints[0].Y_EndPos[0]:=1000;
        SequencePoints[1].X_EndPos[0]:=3000;
        SequencePoints[1].Y_EndPos[0]:=2000;
        SequencePoints[2].X_EndPos[0]:=4000;
        SequencePoints[2].Y_EndPos[0]:=0;
        iState[0]:=2;
    2://启动运动
        xEx_3AC[FALSE]:=TRUE;
        iState[0]:=3;
    3://等待运动完成
        IF LS_3AxisMoveSequence_0.Done[FALSE] THEN
            xEx_3AC[FALSE]:=FALSE;
            iState[0]:=0;
        END_IF
END_CASE
ACT();RETURN
```



- **注意事项**

➢ IpoCycle参数需要与运动指令所在的任务周期时间保持一致，否则指令执行时可能会出现Error报错。

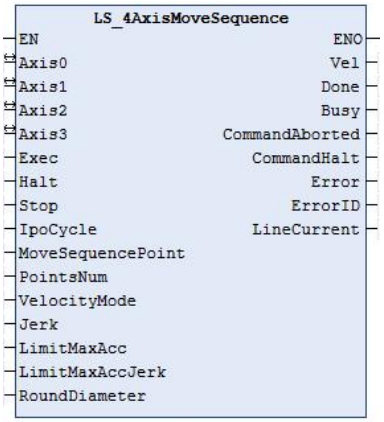
➢ 参数IpoCycle、Jerk、Velocity、Acceleration、Deceleration不可以设置为0。

➢ 在指令运行期间，主从轴都不能再被其它运动指令所调用。

4.4.14 LS_4AxisMoveSequence

四轴连续插补指令，速度连续。

- **指令格式**

指令	名称	FB/FC	LD表现	ST表现	库
LS_4AxisMoveSequence	四轴连续插补	FB		LS_4AxisMoveSequence(Axis0:= , Axis1:= , Axis2:= , Axis3:= , Exec:= , Halt:= , Stop:= , IpoCycle:= , MoveSequencePoint:= , PointsNum:= , VelocityMode:= ,	LS_IpoLib

				Jerk:= , LimitMaxAcc:= , LimitMaxAccJerk:= , RoundDiameter:= , Vel=> , Done=> , Busy=> , CommandAborted=> , CommandHalt=> , Error=> , ErrorID=> , LineCurrent=>);	

- **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	初始值	描述
Axis0	轴0	AXIS_REF_VIRTUAL_SM3	-	-	参与插补的0轴。
Axis1	轴1	AXIS_REF_VIRTUAL_SM3	-	-	参与插补的1轴。
Axis2	轴2	AXIS_REF_VIRTUAL_SM3	-	-	参与插补的2轴。
Axis3	轴3	AXIS_REF_VIRTUAL_SM3	-	-	跟随轴，速度不连续。

- **输入变量**

输入变量	名称	数据类型	有效范围	初始值	描述
Exec	启动	BOOL	TRUE，FALSE	FALSE	连续插补模块启动命令，上升沿有效。
Halt	暂停	BOOL	TRUE，FALSE	FALSE	暂停命令，当该输入量为True时，当前插补运动暂停；当状态再次切换到False时，插补模块继续之前的未完成的插补任务。

Stop	停止	BOOL	TRUE，FALSE	FALSE	停止命令，该命令为True时，插补模块停止。
IpoCycle	插补周期	DWORD	-	2000	插补周期，单位：us。
MoveSequencePoint	插补数据表	ARRAY [0..MoveSequence_MAX] OF MoveSequence	-	-	输入的数据点。
PointsNum	点数	UINT	正数	0	输入的数据点数。
VelocityMode	速度模式	SMC_INT_VELMODE	0，1，3	1	速度模式，0： 梯形（TRAPEZOID）；1：S型曲线（SIGMOID）；3：四次方（QUADRATIC）。
Jerk	加加速度	REAL	正数	90000000	加加速度，速度模式为3时候需要设置此参数。
LimitMaxAcc	最大加速度	REAL	正数	9000000	拐角过渡圆弧最大加速度限制。
LimitMaxAccJerk	最大减速度	REAL	正数	90000000	拐角过渡圆弧最大加加速度限制。
RoundDiameter	拐角半径	REAL	正数	100	拐角过渡圆弧半径，单位：Pulse。

- **输出变量**

输出变量	名称	数据类型	有效范围	初始值	描述
Vel	合速度	LREAL	0，正数	0	当前运动合成速度。
Done	完成	BOOL	TRUE，FALSE	FALSE	为 True 时，表示插补模块已完成插补任务。
Busy	运动中	BOOL	TRUE，FALSE	FALSE	为 True 时，表示插补运动正在进行之中。
CommandAborted	运动中断	BOOL	TRUE，FALSE	FALSE	当 Stop 有效时，该变量为 True，表示插补模块非正常中止。
CommandHalt	运动暂停	BOOL	TRUE，FALSE	FALSE	当 Halt 为 True 时，该变量为 True，

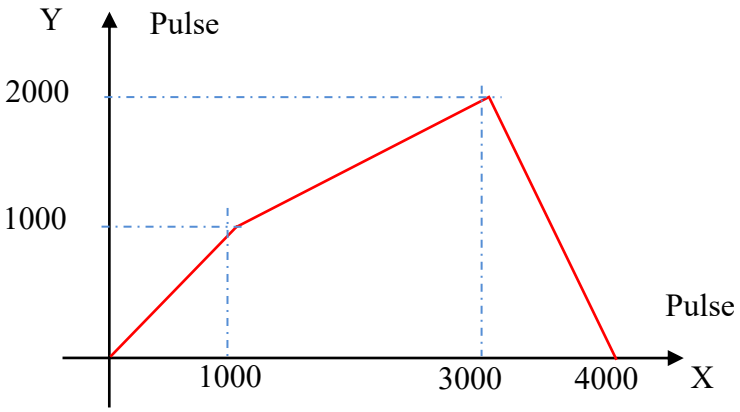
					表示插补模块处于暂停状态。
Error	报错	BOOL	TRUE，FALSE	FALSE	错误状态。
ErrorID	错误码	DINT	0，正数	0	错误码。
LineCuren	行号	UINT	遵循数据类型	0	当前执行的行号。

- **功能说明**

  - 指定Axis0-Axis2做连续插补运动，Axis3做跟随运动，位置坐标为绝对坐标，一次最多可以输入200个插补运动点。
  - 其他参数的设置和LS_3AxisMoveSequence指令相同。

- **程序示例**

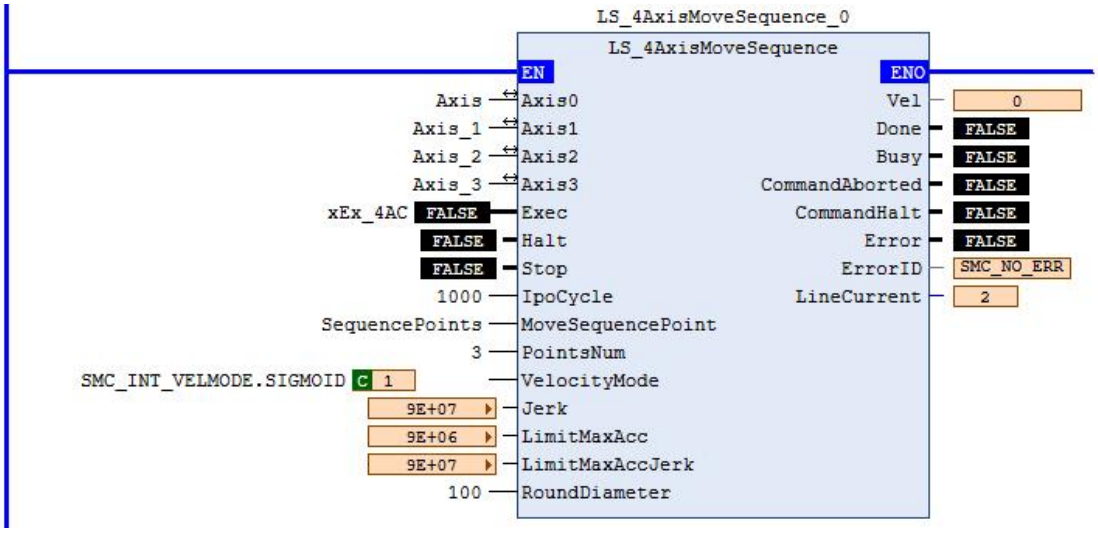
编写程序用连续插补指令实现 XYZU 轴进行三段直线连续插补，速度均为 1000Pulse/s，加减速度为 2000Pulse/s2，第一段终点坐标为（1000，1000，0，0），第二段终点坐标为（3000，2000，0，0），第三段终点坐标为（4000，0，0，0），如图所示。



三段直线的连续插补轨迹

在主程序中添加数据点的相关参数。程序完成后，编译下载到控制器中执行，并将变量 iState 强制改为 1 启动连续插补。

```
CASE iState 0 OF
    1://配置连续插补数据点参数
        FOR i 4 :=0 TO 3 DO
        SequencePoints[i 4 ].Acc 0 :=20000;
        SequencePoints[i 4 ].DataType 0 :=1;
        SequencePoints[i 4 ].LineNum 0 :=i 4 ;
        SequencePoints[i 4 ].U_EndPos 0 :=0;
        SequencePoints[i 4 ].Vel 0 :=10000;
        SequencePoints[i 4 ].Z_EndPos 0 :=0;
        END_FOR
        SequencePoints[0].X_EndPos 1E+03 :=1000;
        SequencePoints[0].Y_EndPos 1E+03 :=1000;
        SequencePoints[1].X_EndPos 3E+03 :=3000;
        SequencePoints[1].Y_EndPos 2E+03 :=2000;
        SequencePoints[2].X_EndPos 4E+03 :=4000;
        SequencePoints[2].Y_EndPos 0 :=0;
        iState 0 :=2;
    2://启动运动
        xEx_4AC FALSE :=TRUE;
        iState 0 :=3;
    3://等待运动完成
        IF LS_4AxisMoveSequence_0.Done FALSE THEN
            xEx_4AC FALSE :=FALSE;
            iState 0 :=0;
        END_IF
END_CASE
ACT();RETURN
```



- **注意事项**

➢ IpoCycle参数需要与运动指令所在的任务周期时间保持一致，否则指令执行时可能会出现Error报错。

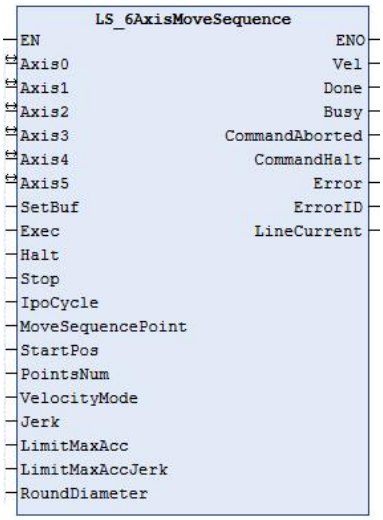
➢ 参数IpoCycle、Jerk、Velocity、Acceleration、Deceleration不可以设置为0。

➢ 在指令运行期间，主从轴都不能再被其它运动指令所调用。

4.4.15 LS_6AxisMoveSequence

六轴连续插补，三轴跟随运动。

- **指令格式**

指令	名称	FB/FC	LD表现	ST表现	库
LS_6AxisMoveSequence	六轴连续插补	FB		LS_6AxisMoveSequence(Axis0:= , Axis1:= , Axis2:= , Axis3:= , Axis4:= , Axis5:= , SetBuf:= , Exec:= , Halt:= , Stop:= , IpoCycle:= , MoveSequencePoint:= , StartPos:= , PointsNum:= , VelocityMode:= , Jerk:= , LimitMaxAcc:= , LimitMaxAccJerk:= , RoundDiameter:= , Vel=> , Done=> , Busy=> , CommandAborted=> , CommandHalt=> , Error=> , ErrorID=> , LineCurrent=>);	LS_IpoLib

- **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	初始值	描述

Axis0	轴0	AXIS_REF_VIRTUAL_SM3	-	-	参与插补的0轴。
Axis1	轴1	AXIS_REF_VIRTUAL_SM3	-	-	参与插补的1轴。
Axis2	轴2	AXIS_REF_VIRTUAL_SM3	-	-	参与插补的2轴。
Axis3	轴3	AXIS_REF_VIRTUAL_SM3	-	-	跟随轴，速度不连续。
Axis4	轴4	AXIS_REF_VIRTUAL_SM3	-	-	跟随轴，速度不连续。
Axis5	轴5	AXIS_REF_VIRTUAL_SM3	-	-	跟随轴，速度不连续。

- **输入变量**

输入变量	名称	数据类型	有效范围	初始值	描述
SetBuf	触发压入数据	BOOL	TRUE，FALSE	FALSE	插补数据压入缓存。
Exec	启动	BOOL	TRUE，FALSE	FALSE	连续插补模块启动命令，上升沿有效。
Halt	暂停	BOOL	TRUE，FALSE	FALSE	暂停命令，当该输入量为True时，当前插补运动暂停；当状态再次切换到False时，插补模块继续之前的未完成的插补任务。
Stop	停止	BOOL	TRUE，FALSE	FALSE	停止命令，该命令为True时，插补模块停止。
IpoCycle	插补周期	DWORD	-	2000	插补周期，单位：us。

MoveSequenceePoint	插补数据表	ARRAY [0..MoveSequence_MAX] OF MoveSequence	-	-	输入的数据点。
PointsNum	点数	UINT	正数	0	输入的数据点数。
VelocityMode	速度模式	SMC_INT_VELMODE	0，1，3	1	速度模式，0： 梯形（TRAPEZOID）；1：S型曲线（SIGMOID）；3：四次方（QUADRATIC）。
Jerk	加加速度	REAL	正数	90000000	加加速度，速度模式为3时候需要设置此参数。
LimitMaxAcc	最大加速度	REAL	正数	9000000	拐角过渡圆弧最大加速度限制。
LimitMaxAccJerk	最大减速度	REAL	正数	90000000	拐角过渡圆弧最大加加速度限制。
RoundDiameter	拐角半径	REAL	正数	100	拐角过渡圆弧半径，单位：Pulse。

- **输出变量**

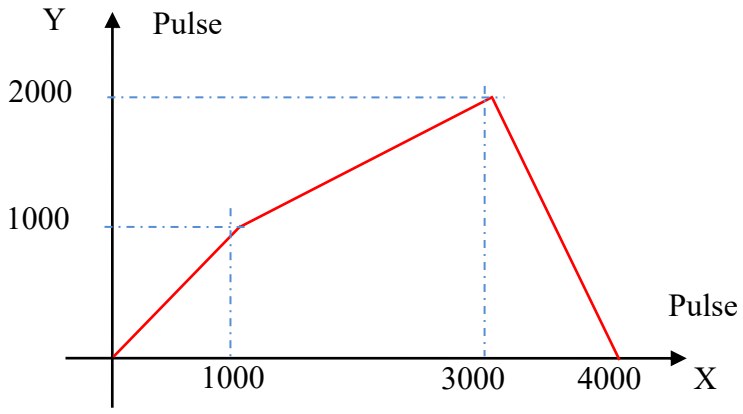
输出变量	名称	数据类型	有效范围	初始值	描述
Vel	合速度	LREAL	0，正数	0	当前运动合成速度。
Done	完成	BOOL	TRUE，FALSE	FALSE	为 True 时，表示插补模块已完成插补任务。
Busy	运动中	BOOL	TRUE，FALSE	FALSE	为 True 时，表示插补运动正在进行之中。
CommandAborted	运动中断	BOOL	TRUE，FALSE	FALSE	当 Stop 有效时，该变量为 True，表示插补模块非正常中止。
CommandHalt	运动暂停	BOOL	TRUE，FALSE	FALSE	当 Halt 为 True 时，该变量为 True，表示插补模块处于暂停状态。
Error	报错	BOOL	TRUE，FALSE	FALSE	错误状态。
ErrorID	错误码	DINT	0，正数	0	错误码。
LineCuren	行号	UINT	遵循数据类型	0	当前执行的行号。

- **功能说明**

  - 六轴连续插补，三轴跟随运动。一次最多可以输入MoveSequence_MAX个点。参数IpoCycle、Jerk、LimitMaxAcc、LimitMaxAccJerk、RoundDiameter不可以设置为0。LT_6AxisMoveSequence指令将Exec设置为TRUE后，触发压缓存和指令的执行。

- **程序示例**

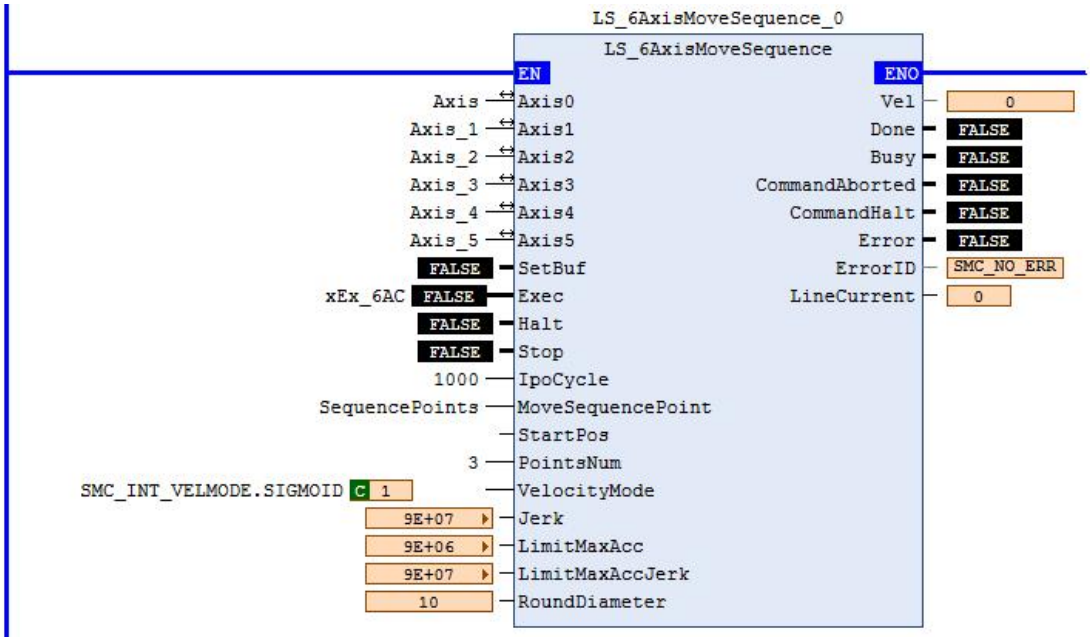
编写程序用连续插补指令实现 X/Y/Z/U/V/W 轴进行三段直线连续插补，速度均为1000Pulse/s，加减速度为 2000Pulse/s^2，第一段终点坐标为（1000，1000，0，0，0，0），第二段终点坐标为（3000，2000，0，0，0，0），第三段终点坐标为（4000，0，0，0，0，0），如图所示。



三段直线的连续插补轨迹

在主程序中添加数据点的相关参数。程序完成后，编译下载到控制器中执行，并将变量 Exec 强制改为1启动连续插补。

```
CASE iState[0] OF
    1://配置连续插补数据点参数
        FOR i[0]:=0 TO 5 DO
        SequencePoints[i[0]].Acc[0]:=20000;
        SequencePoints[i[0]].DataType[0]:=1;
        SequencePoints[i[0]].LineNum[0]:=i[0];
        SequencePoints[i[0]].Vel[0]:=10000;
        SequencePoints[i[0]].Z_EndPos[0]:=0;
        SequencePoints[i[0]].U_EndPos[0]:=0;
        SequencePoints[i[0]].V_EndPos[0]:=0;
        SequencePoints[i[0]].W_EndPos[0]:=0;
        END_FOR
        SequencePoints[0].X_EndPos[0]:=1000;
        SequencePoints[0].Y_EndPos[0]:=1000;
        SequencePoints[1].X_EndPos[0]:=3000;
        SequencePoints[1].Y_EndPos[0]:=2000;
        SequencePoints[2].X_EndPos[0]:=4000;
        SequencePoints[2].Y_EndPos[0]:=0;
        iState[0]:=2;
    2://启动运动
        xEx_6AC[FALSE]:=TRUE;
        iState[0]:=3;
    3://等待运动完成
        IF LS_6AxisMoveSequence_0.Done[FALSE] THEN
            xEx_6AC[FALSE]:=FALSE;
            iState[0]:=0;
        END_IF
END_CASE
ACT();RETURN
```



- **注意事项**

➢ IpoCycle参数需要与运动指令所在的任务周期时间保持一致，否则指令执行时可能会出现

Error报错。

- 参数IpoCycle、Jerk、Velocity、Acceleration、Deceleration不可以设置为0。

  在指令运行期间，主从轴都不能再被其它运动指令所调用。

5 通信指令

5.1 自由 TCP 通信指令

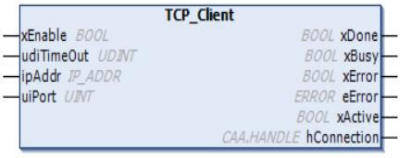
5.1.1 指令列表

指令类别	名称	FB/FC	功能
自由 TCP 通信指令	TCP_Client	FB	创建 TCP 客户端通信服务
	TCP_Write	FB	TCP 通信数据发送
	TCP_Read	FB	TCP 通信数据接收
	TCP_Connection	FB	创建 TCP 连接，并连接到服务器
	TCP_Server	FB	创建 TCP 服务器端通信服务

5.1.2 TCP_Client

创建 TCP 客户端通信服务

- **指令格式**

指令	名称	FB/FC	LD 表现	ST 表现	库
TCP_Client	创建TCP客户端	FB		TCP_Client(xEnable:= , xDone=> , xBusy=> , xError=> , udiTimeOut:= , ipAddr:= , uiPort:= , eError=> , xActive=> , hConnection=>)	CAA Net Base Services

- **相关变量**

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
xEnable	功能块使能	BOOL	TRUE-FALSE	-	若为TRUE，则启动功能块运行
udiTimeOut	超时时间	UDINT	0- 4294967295	0	请求建立连接的超时时间，单位：μs
ipAddr	服务器IP地址	IP_ADDR	-	-	客户端所连接的服务器IP地址，具体参考CAA Net Base Services库的结构体IP_ADDR
uiPort	端口号	UINT	0-65535	-	客户端所连接的服务器端口号

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
xDone	完成	BOOL	TRUE-FALSE	-	客户端通信完成
xBusy	功能块运行中	BOOL	TRUE-FALSE	-	功能块运行中标志
xError	错误标志	BOOL	TRUE-FALSE	-	错误状态
eError	错误	ERROR	--	-	具体错误请参考CAA Net Base Services库的枚举体ERROR
xActive	连接成功标志	BOOL	TRUE-FALSE	-	客户端与服务器连接成功标志
hConnection	连接句柄	CAA. HANDLE	--	_	客户端与服务器的连接句柄

	布尔	位串				整数								实数		时刻、持续时间 日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
xEnable	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
udiTim	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-

eOut																			
ipAddr	IP_ADDR																		
uiPort	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–
xDone	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
xBusy	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
xError	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
eError	ERROR																		
xActive	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
hConnec tion	CAA. HANDLE																		

- **功能说明**

  - 创建TCP客户端通信服务。

  - “ipAddr”的数据类型为结构体IP_ADDR，结构体成员的含义如下表所示。

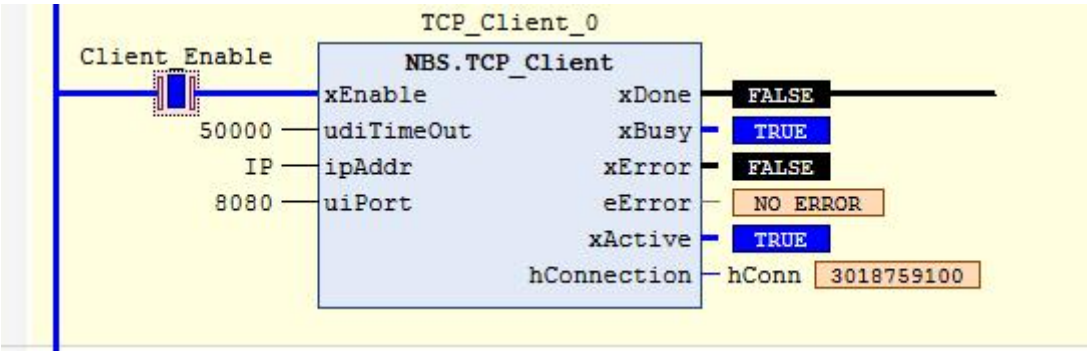
结构体成员	数据类型	描述
sAddr	STRING(80)	客户端所连接服务器的IP地址

- **程序示例**

ST： 创建TCP客户端通信服务。当TCP_Client指令xEnable为TRUE时，将创建本地服务器端与远端客户端TCP通信的有效句柄（hServer大于0），xBusy为TRUE。

```
TCP_Client_0(
    xEnable TRUE :=Client_Enable TRUE ,
    xDone=> ,
    xBusy TRUE => Client_Busy TRUE ,
    xError=> ,
    udiTimeOut 50000 := 50000,
    ipAddr:=IP ,
    uiPort 8080 :=8080 ,
    eError=> ,
    xActive=> ,
    hConnection 3018759100 => hConn 3018759100 );
```

LD： 创建TCP客户端通信服务。当TCP_Client指令xEnable为TRUE时，将创建本地服务器端与远端客户端TCP通信的有效句柄（hServer大于0），xBusy为TRUE。



- **注意事项**

➢ 如果功能块出错，xError将置TRUE，eError将被赋相关错误值（枚举型ERROR里的各种值），hConnection将无效（等于0）。详细请看以下错误码。

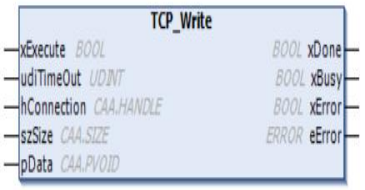
错误码	定义	描述
0	NO_ERROR	无错误
6000	FIRST_ERROR	保留
6001	TIME_OUT	保留
6002	INVALID_ADDR	客户端所连接服务器的IP地址无效
6003	INVALID_HANDLE	连接句柄无效
6004	INVALID_DATAPOINTER	数据指针无效
6005	INVALID_DATASIZE	数据大小无效
6006	UDP_RECEIVE_ERROR	UDP接收错误
6007	UDP_SEND_ERROR	UDP发送错误
6008	UDP_SEND_NOT_COMPLETE	UDP发送未完成
6009	UDP_OPEN_ERROR	UDP打开错误
6010	UDP_CLOSE_ERROR	UDP关闭错误
6011	TCP_SEND_ERROR	TCP发送错误
6012	TCP_RECEIVE_ERROR	TCP发送未完成
6013	TCP_OPEN_ERROR	TCP打开错误
6014	TCP_CONNECT_ERROR	TCP连接错误
6015	TCP_CLOSE_ERROR	TCP关闭错误
6016	TCP_SERVER_ERROR	TCP服务器错误
6017	WRONG_PARAMETER	参数错误
6018	ERROR_UNKNOWN	未知错误
6019	TCP_NO_CONNECTION	无TCP连接

6020	LOCTL_ERROR	内部错误（本机不支持）
6050	FIRST_MF	保留
6099	LAST_ERROR	保留

5.1.3 TCP_Write

将数据写入在 hConnection 所建立的连接。

- **指令格式**

指令	名称	FB/FC	LD 表现	ST表现	库
TCP_Write	TCP通信数据发送	FB		TCP_Write(xExecute:= , udiTimeOut:= , xDone=> , xBusy=> , xError=> , hConnection:= , szSize:= , pData:= , eError=>)	CAA Net Base Services

- **相关变量**

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
xExecute	功能块使能	BOOL	TRUE-FALSE	FALSE	若为TRUE（上升沿信号），则启动功能块运行
udiTimeOut	超时时间	UDINT	0- 4294967295	0	请求建立连接的超时时间，单位：μs
hConnection	连接句柄	CAA. HANDLE	-	-	客户端与服务器的连接句柄
szSize	数据大小	CAA. SIZE	-	-	发送数据区域大小，byte
pData	发送缓存	CAA. PVOID	-	-	指向属性数据的指针，发送该数据到

					目标设备

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
xDone	完成	BOOL	TRUE-FALSE	-	客户端通信完成
xBusy	功能块运行中	BOOL	TRUE-FALSE	-	功能块运行中标志
xError	错误标志	BOOL	TRUE-FALSE	-	错误状态
eError	错误	ERROR	-	-	具体错误请参考CAA Net Base Services库的枚举体ERROR

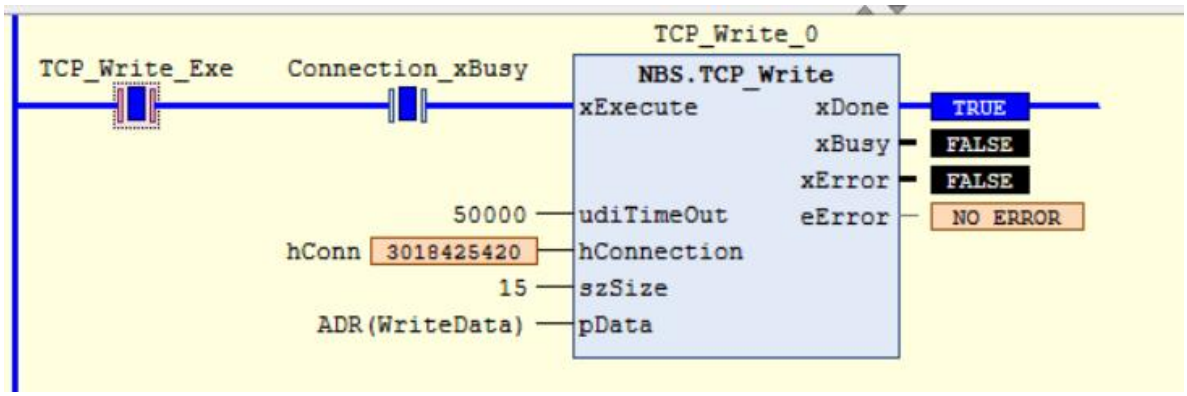
	布尔	位串				整数								实数		时刻、持续时间 日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
xExecute	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
udiTime Out	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-
hConnection	CAA. HANDLE																			
szSize	CAA. SIZE																			
pData	CAA. PVOID																			
xDone	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
xBusy	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
xError	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
eError	ERROR																			

- **程序示例**

ST: 此功能块的作用为将数据写入在 hConnection 所建立的连接。当 TCP_Write 指令 xExecute 为 TRUE 时，会将用户设置的发送缓冲区 pData 为首地址的长度为 szSize 的数据发送到目标设备。如果在超时时间内发送成功，xDone 置 TRUE。

```
1  TCP_Write_0(
2      xExecute TRUE := TCP_Write_Exe TRUE,
3      udiTimeOut 50000 := 50000,
4      xDone=> ,
5      xBusy=> ,
6      xError=> ,
7      hConnection 3018759840 :=hConn 3018759840 ,
8      szSize 15 :=15 ,
9      pData 3018760317 :=ADR(WriteData) ,
10     eError=> );
11
```

LD:此功能块的作用为将数据写入在 hConnection 所建立的连接。当 TCP_Write 指令 xExecute 为 TRUE 时，会将用户设置的发送缓冲区 pData 为首地址的长度为 szSize 的数据发送到目标设备。如果在超时时间内发送成功，xDone 置 TRUE。



- **注意事项**

➢ 如果功能块出错，xError将置TRUE，eError将被赋相关错误值（枚举型ERROR里的各种值），hConnection将无效（等于0）。详细请看以下错误码：

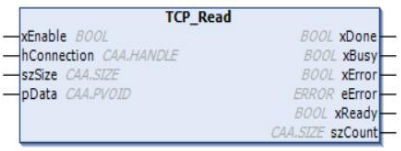
错误码	定义	描述
0	NO_ERROR	无错误
6000	FIRST_ERROR	保留
6001	TIME_OUT	保留
6002	INVALID_ADDR	客户端所连接服务器的IP地址无效
6003	INVALID_HANDLE	连接句柄无效
6004	INVALID_DATAPOINTER	数据指针无效
6005	INVALID_DATASIZE	数据大小无效

6006	UDP_RECEIVE_ERROR	UDP接收错误
6007	UDP_SEND_ERROR	UDP发送错误
6008	UDP_SEND_NOT_COMPLETE	UDP发送未完成
6009	UDP_OPEN_ERROR	UDP打开错误
6010	UDP_CLOSE_ERROR	UDP关闭错误
6011	TCP_SEND_ERROR	TCP发送错误
6012	TCP_RECEIVE_ERROR	TCP发送未完成
6013	TCP_OPEN_ERROR	TCP打开错误
6014	TCP_CONNECT_ERROR	TCP连接错误
6015	TCP_CLOSE_ERROR	TCP关闭错误
6016	TCP_SERVER_ERROR	TCP服务器错误
6017	WRONG_PARAMETER	参数错误
6018	ERROR_UNKNOWN	未知错误
6019	TCP_NO_CONNECTION	无TCP连接
6020	LOCTL_ERROR	内部错误（本机不支持）
6050	FIRST_MF	保留
6099	LAST_ERROR	保留

5.1.4 TCP_Read

从 hConnection 所建立的连接的通信缓冲区中读取数据。

- **指令格式**

指令	名称	FB/FC	LD 表现	ST表现	库
TCP_Read	TCP通信数据接收	FB		TCP_Read(xEnable:= , xDone=> , xBusy=> , xError=> , hConnection:= , szSize:= , pData:= , eError=> , xReady=> , szCount=>) ;	CAA Net Base Services

- **相关变量**

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
xEnable	功能块使能	BOOL	TRUE-FALSE	-	若为TRUE，则启动功能块运行
hConnection	连接句柄	CAA. HANDLE	-	-	客户端与服务器的连接句柄
szSize	数据大小	CAA. SIZE	-	-	接收数据区域大小，byte
pData	发送缓存	CAA. PVOID	-	-	指向属性数据的指针，从目标设备接收数据

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
xDone	完成	BOOL	TRUE-FALSE	-	客户端通信完成
xBusy	功能块运行中	BOOL	TRUE-FALSE	-	功能块运行中标志
xError	错误标志	BOOL	TRUE-FALSE	-	错误状态
eError	错误	ERROR	-	-	具体错误请参考CAA Net Base Services库的枚举体ERROR
xReady	成功建立连接标志位	BOOL	TRUE-FALSE	-	从缓冲区内读取数据，如果数据不为空，置位一个扫描周期的标志位
szCount	数据大小	CAA. SIZE	-	-	接收数据区域的实际大小

	布尔	位串				整数								实数		时刻、持续时间 日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING

xEnable	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
hConnection	CAA. HANDLE																		
szSize	CAA. SIZE																		
pData	CAA. PVOID																		
xDone	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
xBusy	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
xError	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
eError	ERROR																		
xReady	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
szCount	CAA. SIZE																		

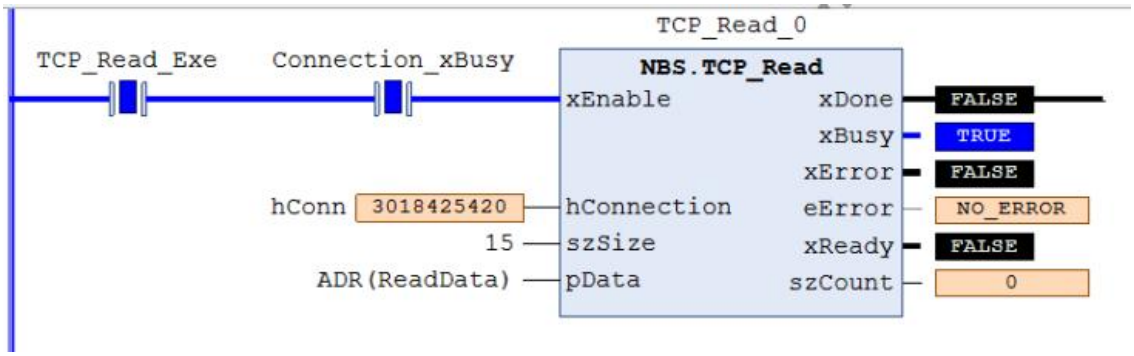
- **程序示例**

ST:从 hConnection 所建立的连接的通信缓冲区中读取数据。当 TCP_Read 指令xEnable 为 TRUE 时，将会从 TCP 通信缓冲区读取数据，xBusy 为 TRUE。若读取成功，xDone 置位一个扫描周期；读取的数据会被放置到地址为 pData 的变量里；同时 xReady 置位一个扫描周期;接收数据区域的实际大小值将会赋给 szCount,在一个扫描周期后 szCount 的值会被清零。

```
1
2  TCP_Read_0(
3      xEnable TRUE := TCP_Read_Exe TRUE,
4      xDone=> ,
5      xBusy=> ,
6      xError=> ,
7      hConnection 3017879200 := hConn 3017879200 ,
8      szSize 15 :=15 ,
9      pData 3017879692 :=ADR(ReadData) ,
10     eError=> ,
11     xReady=> ,
12     szCount=> );
13
14
15
```

LD:从 hConnection 所建立的连接的通信缓冲区中读取数据。当 TCP_Read 指令xEnable 为 TRUE 时，将会从 TCP 通信缓冲区读取数据，xBusy 为 TRUE。若读取成功，xDone

置位一个扫描周期；读取的数据会被放置到地址为 pData 的变量里；同时 xReady 置位一个扫描周期；接收数据区域的实际大小值将会赋给 szCount，在一个扫描周期后 szCount 的值会被清零。



- **注意事项**

➢ 如果功能块出错，xError将置TRUE，eError将被赋相关错误值（枚举型ERROR里的各种值），hConnection将无效（等于0）。详细请看以下错误码。

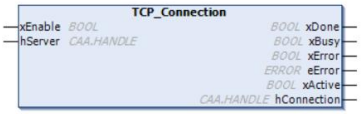
错误码	定义	描述
0	NO_ERROR	无错误
6000	FIRST_ERROR	保留
6001	TIME_OUT	保留
6002	INVALID_ADDR	客户端所连接服务器的IP地址无效
6003	INVALID_HANDLE	连接句柄无效
6004	INVALID_DATAPOINTER	数据指针无效
6005	INVALID_DATASIZE	数据大小无效
6006	UDP_RECEIVE_ERROR	UDP接收错误
6007	UDP_SEND_ERROR	UDP发送错误
6008	UDP_SEND_NOT_COMPLETE	UDP发送未完成
6009	UDP_OPEN_ERROR	UDP打开错误
6010	UDP_CLOSE_ERROR	UDP关闭错误
6011	TCP_SEND_ERROR	TCP发送错误
6012	TCP_RECEIVE_ERROR	TCP发送未完成
6013	TCP_OPEN_ERROR	TCP打开错误
6014	TCP_CONNECT_ERROR	TCP连接错误
6015	TCP_CLOSE_ERROR	TCP关闭错误

6016	TCP_SERVER_ERROR	TCP服务器错误
6017	WRONG_PARAMETER	参数错误
6018	ERROR_UNKNOWN	未知错误
6019	TCP_NO_CONNECTION	无TCP连接
6020	LOCTL_ERROR	内部错误（本机不支持）
6050	FIRST_MF	保留
6099	LAST_ERROR	保留

5.1.5 TCP_Connection

创建 TCP 连接并连接到服务器。

- **指令格式**

指令	名称	FB/FC	LD 表现	ST表现	库
TCP_Connect	创建TCP连接，并连 到服务器	FB		TCP_Connection(xEnable:= , xDone=> , xBusy=> , xError=> , hServer:= , eError=> , xActive=> , hConnection=>);	CAA Net Base Services

- **相关变量**

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
xEnable	功能块使能	BOOL	TRUE-FALSE	-	若为TRUE，则启动功能块运行
hServer	服务器端句柄	CAA. HANDLE	-	-	TCP服务器端的句柄

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述

xDone	完成	BOOL	TRUE-FALSE	-	客户端通信完成
xBusy	功能块运行中	BOOL	TRUE-FALSE	-	功能块运行中标志
xError	错误标志	BOOL	TRUE-FALSE	-	错误状态
eError	错误	ERROR	-	-	具体错误请参考CAA Net Base Services库的枚举体ERROR
xActive	连接成功标志	BOOL	TRUE-FALSE	-	客户端与服务器连接成功标志
hConnection	连接句柄	CAA. HANDLE	-	-	客户端与服务器的连接句柄

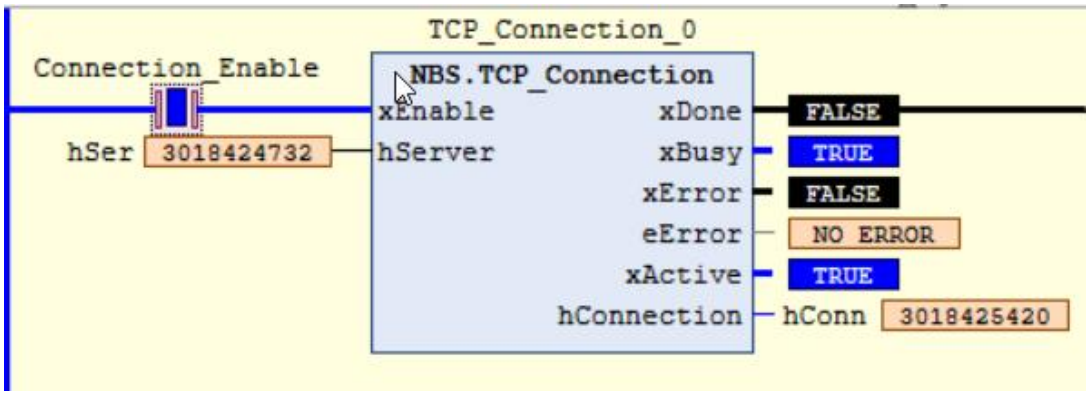
	布尔	位串				整数								实数		时刻、持续时间 日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
xEnable	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
hServer	CAA. HANDLE																			
xDone	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
xBusy	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
xError	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
eError	ERROR																			
xActive	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
hConnection	CAA. HANDLE																			

- **程序示例**

ST: 建立 TCP 连接并连接到服务器。当 TCP_Connection 指令 xEnable 为 TRUE 时，本地服务器端将监听远端客户端的连接请求，若客户端和服务器连接成功，将创建服务器与远端客户的通信连接句柄 hConnection。

```
TCP_Connection_0(
    xEnable TRUE := Connection_Enable TRUE,
    xDone=> ,
    xBusy TRUE => Connection_xBusy TRUE,
    xError=> ,
    hServer 3017878424 :=hSer 3017878424 ,
    eError=> ,
    xActive=> ,
    hConnection 3017879200 => hConn 3017879200 );
```

LD:建立 TCP 连接并连接到服务器。当 TCP_Connection 指令 xEnable 为 TRUE 时，本地服务器端将监听远端客户端的连接请求，若客户端和服务器连接成功，将创建服务器与远端客户的通信连接句柄 hConnection。



- **注意事项**

➢ 如果通信过程中出现错误，xError将置TRUE，eError将被赋相关错误值（枚举型ERROR里的各种值），hConnection将无效（等于0）。详细请看以下错误码。

错误码	定义	描述
0	NO_ERROR	无错误
6000	FIRST_ERROR	保留
6001	TIME_OUT	保留
6002	INVALID_ADDR	客户端所连接服务器的IP地址无效
6003	INVALID_HANDLE	连接句柄无效
6004	INVALID_DATAPOINTER	数据指针无效
6005	INVALID_DATASIZE	数据大小无效

6006	UDP_RECEIVE_ERROR	UDP接收错误
6007	UDP_SEND_ERROR	UDP发送错误
6008	UDP_SEND_NOT_COMPLETE	UDP发送未完成
6009	UDP_OPEN_ERROR	UDP打开错误
6010	UDP_CLOSE_ERROR	UDP关闭错误
6011	TCP_SEND_ERROR	TCP发送错误
6012	TCP_RECEIVE_ERROR	TCP发送未完成
6013	TCP_OPEN_ERROR	TCP打开错误
6014	TCP_CONNECT_ERROR	TCP连接错误
6015	TCP_CLOSE_ERROR	TCP关闭错误
6016	TCP_SERVER_ERROR	TCP服务器错误
6017	WRONG_PARAMETER	参数错误
6018	ERROR_UNKNOWN	未知错误
6019	TCP_NO_CONNECTION	无TCP连接
6020	LOCTL_ERROR	内部错误（本机不支持）
6050	FIRST_MF	保留
6099	LAST_ERROR	保留

5.1.6 TCP_Server

创建 TCP 服务端的通信服务

- **指令格式**

指令	名称	FB/FC	LD 表现	ST表现	库
TCP_Server	创建TCP服务端	FB		TCP_Server(xEnable:= , xDone=> , xBusy=> , xError=> , ipAddr:= , uiPort:= , eError=> , hServer=>) ;	CAA Net Base Services

- **相关变量**

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
xEnable	功能块使能	BOOL	TRUE-FALSE	-	若为TRUE，则启动功能块运行
ipAddr	服务器IP地址	IP_ADDR	-	-	服务器IP地址，具体参考CAA Net Base Services库的结构体IP_ADDR
uiPort	本地端口号	UINT	0-4294967297	-	服务器端口号

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
xDone	完成	BOOL	TRUE-FALSE	-	客户端通信完成
xBusy	功能块运行中	BOOL	TRUE-FALSE	-	功能块运行中标志
xError	错误标志	BOOL	TRUE-FALSE	-	错误状态
eError	错误	ERROR	-	-	具体错误请参考CAA Net Base Services库的枚举体ERROR
hServer	服务器句柄	CAA. HANDLE	-	-	服务器句柄

	布尔	位串				整数								实数		时刻、持续时间 日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
xEnable	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
ipAddr	IP_ADDR																			
uiPort	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-
xDone	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
xBusy	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
xError	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
eError	ERROR																			

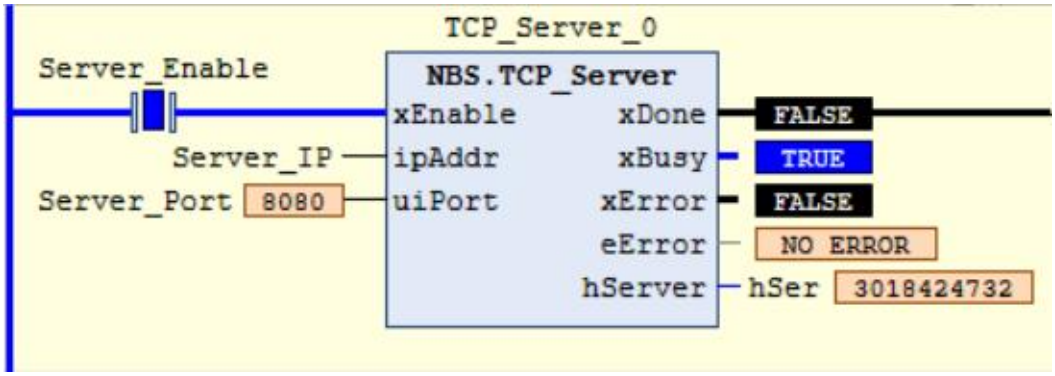
hServer	CAA. HANDLE

- **程序示例**

ST： 创建 TCP 服务器端通信服务。当 TCP_Server 指令 xEnable 为 TRUE 时，将创建本地服务器端与远端客户端 TCP 通信的有效句柄（hServer 大于 0），xBusy 为 TRUE。

```
1  Server_IP.sAddr '192.168.1.' :='192.168.1.10';
2  Server_Port 8080 :=8080;
3  TCP_Server_0(
4      xEnable TRUE := Server_Enable TRUE,
5      xDone=> ,
6      xBusy TRUE =>Server_xBusy TRUE ,
7      xError=> ,
8      ipAddr:= Server_IP,
9      uiPort 8080 :=Server_Port 8080 ,
10     eError=> ,
11     hServer 3017878424 => hSer 3017878424 );
12
```

LD:创建 TCP 服务器端通信服务。当 TCP_Server 指令 xEnable 为 TRUE 时，将创建本地服务器端与远端客户端 TCP 通信的有效句柄（hServer 大于 0），xBusy 为 TRUE。



- **注意事项**

➢ 如果通信过程中出现错误，xError将置TRUE，eError将被赋相关错误值（枚举型ERROR里的各种值），hServer将无效（等于0）。详细请看以下错误代码。

错误码	定义	描述
0	NO_ERROR	无错误
6000	FIRST_ERROR	保留
6001	TIME_OUT	保留

6002	INVALID_ADDR	客户端所连接服务器的IP地址无效
6003	INVALID_HANDLE	连接句柄无效
6004	INVALID_DATAPOINTER	数据指针无效
6005	INVALID_DATASIZE	数据大小无效
6006	UDP_RECEIVE_ERROR	UDP接收错误
6007	UDP_SEND_ERROR	UDP发送错误
6008	UDP_SEND_NOT_COMPLETE	UDP发送未完成
6009	UDP_OPEN_ERROR	UDP打开错误
6010	UDP_CLOSE_ERROR	UDP关闭错误
6011	TCP_SEND_ERROR	TCP发送错误
6012	TCP_RECEIVE_ERROR	TCP发送未完成
6013	TCP_OPEN_ERROR	TCP打开错误
6014	TCP_CONNECT_ERROR	TCP连接错误
6015	TCP_CLOSE_ERROR	TCP关闭错误
6016	TCP_SERVER_ERROR	TCP服务器错误
6017	WRONG_PARAMETER	参数错误
6018	ERROR_UNKNOWN	未知错误
6019	TCP_NO_CONNECTION	无TCP连接
6020	LOCTL_ERROR	内部错误（本机不支持）
6050	FIRST_MF	保留
6099	LAST_ERROR	保留

5.2 自由 UDP 通信指令

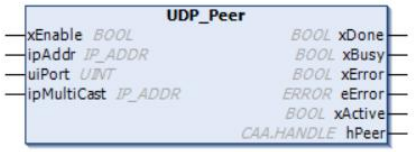
5.2.1 指令列表

指令类别	名称	FB/FC	功能
自由 UDP 通信指令	UDP_Peer	FB	创建 UDP 通信连接
	UDP_Receive	FB	UDP 通信数据接收
	UDP_Send	FB	UDP 通信数据发送

5.2.2 UDP_Peer

创建 UDP 通信连接

- **指令格式**

指令	名称	FB/FC	LD 表现	ST表现	库
UDP_Peer	创建UDP通信连接	FB		UDP_Peer(xEnable:= , xDone=> , xBusy=> , xError=> , ipAddr:= , uiPort:= , ipMultiCast:= , eError=> , xActive=> , hPeer=>);	CAA Net Base Services

- **相关变量**

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
xEnable	功能块使能	BOOL	TRUE-FALSE	-	若为TRUE，则启动功能块运行
ipAddr		IP_ADDR	-	-	

uiPort	通信端口号	UINT	0-4294967297	-	通信端口号
ipMultiCast		IP_ADDR	-	-	

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
xDone	完成	BOOL	TRUE-FALSE	-	客户端通信完成
xBusy	功能块运行中	BOOL	TRUE-FALSE	-	功能块运行中标志
xError	错误标志	BOOL	TRUE-FALSE	-	错误状态
eError	错误	ERROR	-	-	具体错误请参考CAA Net Base Services库的枚举体ERROR
hPeer	通信句柄	CAA. HANDLE	-	-	UDP通信句柄

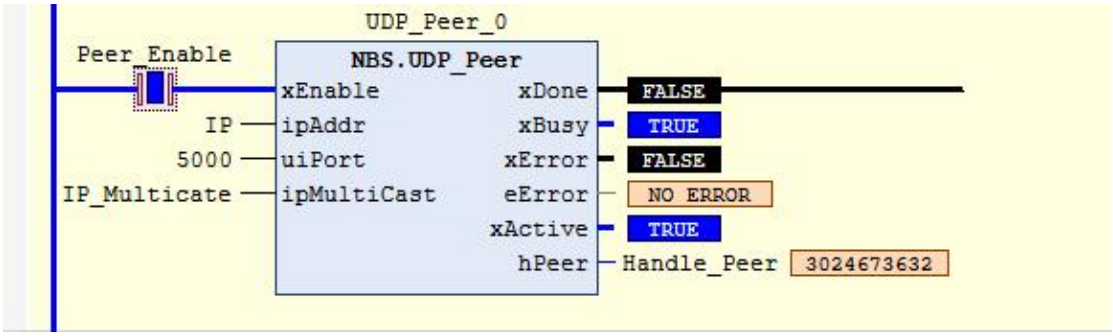
	布尔	位串				整数								实数		时刻、持续时间 日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
xEnable	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
ipAddr	IP_ADDR																			
uiPort	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-
ipMultiCast	IP_ADDR																			
xDone	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
xBusy	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
xError	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
eError	ERROR																			
hpeer	CAA. HANDLE																			

- **程序示例**

ST：创建 UDP 通信连接。当 UDP_Peer 指令 xEnable 为 TRUE 时，将创建 UDP 通信的有效句柄（hpeer 大于 0），xBusy 及 xActive 为 TRUE。

```
1  IP.sAddr[192.168.1.]:='192.168.1.10';
2  IP_Multicate.sAddr[255.255.25]:='255.255.255.255';
3  IP_Target.sAddr[192.168.1.]:='192.168.1.90';
4  UDP_Peer_0(
5      xEnable[TRUE]:=Peer_Enable[TRUE],
6      xDone=> ,
7      xBusy[TRUE]=>Peer_xBusy[TRUE] ,
8      xError=> ,
9      ipAddr:=IP ,
10     uiPort[5000]:= 5000,
11     ipMultiCast:=IP_Multicate ,
12     eError=> ,
13     xActive[TRUE]=>Peer_xActive[TRUE] ,
14     hPeer[3024673632]=>Handle_Peer[3024673632] );
```

LD：创建 UDP 通信连接。当 UDP_Peer 指令 xEnable 为 TRUE 时，将创建 UDP 通信的有效句柄（hpeer 大于 0），xBusy 及 xActive 为 TRUE。



- **注意事项**

➢ 如果通信过程中出现错误，xError将置TRUE，eError将被赋相关错误值（枚举型ERROR里的各种值），详细请看以下错误代码。

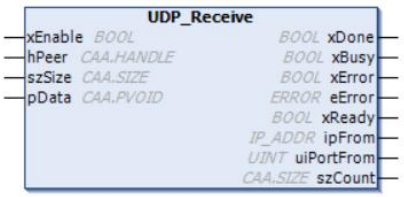
错误码	定义	描述
0	NO_ERROR	无错误
6000	FIRST_ERROR	保留
6001	TIME_OUT	保留
6002	INVALID_ADDR	客户端所连接服务器的IP地址无效
6003	INVALID_HANDLE	连接句柄无效
6004	INVALID_DATAPOINTER	数据指针无效

6005	INVALID_DATASIZE	数据大小无效
6006	UDP_RECEIVE_ERROR	UDP接收错误
6007	UDP_SEND_ERROR	UDP发送错误
6008	UDP_SEND_NOT_COMPLETE	UDP发送未完成
6009	UDP_OPEN_ERROR	UDP打开错误
6010	UDP_CLOSE_ERROR	UDP关闭错误
6011	TCP_SEND_ERROR	TCP发送错误
6012	TCP_RECEIVE_ERROR	TCP发送未完成
6013	TCP_OPEN_ERROR	TCP打开错误
6014	TCP_CONNECT_ERROR	TCP连接错误
6015	TCP_CLOSE_ERROR	TCP关闭错误
6016	TCP_SERVER_ERROR	TCP服务器错误
6017	WRONG_PARAMETER	参数错误
6018	ERROR_UNKNOWN	未知错误
6019	TCP_NO_CONNECTION	无TCP连接
6020	LOCTL_ERROR	内部错误（本机不支持）
6050	FIRST_MF	保留
6099	LAST_ERROR	保留

5.2.3 UDP_Receive

UDP 通信数据接收

- **指令格式**

指令	名称	FB/FC	LD 表现	ST表现	库
UDP_Receive	UDP通信数据接收	FB		UDP_Receive(xEnable:= , xDone=> , xBusy=> , xError=> , hPeer:= , szSize:= , pData:= , eError=> , xReady=> , ipFrom=> ,	CAA Net Base Services

				uiPortFrom=> , szCount=>);	

- **相关变量**

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
xEnable	功能块使能	BOOL	TRUE-FALSE	-	若为TRUE，则启动功能块运行
hPeer	通信句柄	CAA. HANDLE	-	-	UDP通信句柄
szSize	数据大小	CAA. SIZE	-	-	接收数据区域大小，byte
pData	发送缓存	CAA. PVOID	-	-	指向属性数据的指针，从目标设备接收数据

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
xDone	完成	BOOL	TRUE-FALSE	-	客户端通信完成
xBusy	功能块运行中	BOOL	TRUE-FALSE	-	功能块运行中标志
xError	错误标志	BOOL	TRUE-FALSE	-	错误状态
eError	错误	ERROR	-	-	具体错误请参考CAA Net Base Services库的枚举体ERROR
xReady	成功建立连接标志位	BOOL	TRUE-FALSE	-	从缓冲区内读取数据，如果数据不为空，置位一个扫描周期的标志位
ipFrom	当前数据包源IP	IP_ADDR	-	-	当前数据包源IP
uiPortFrom	当前数据包源端口号	UINT	0-4294967297	-	当前数据包源端口号
szCount	数据大小	CAA. SIZ	-	-	接收数据区域的实际

		E			大小

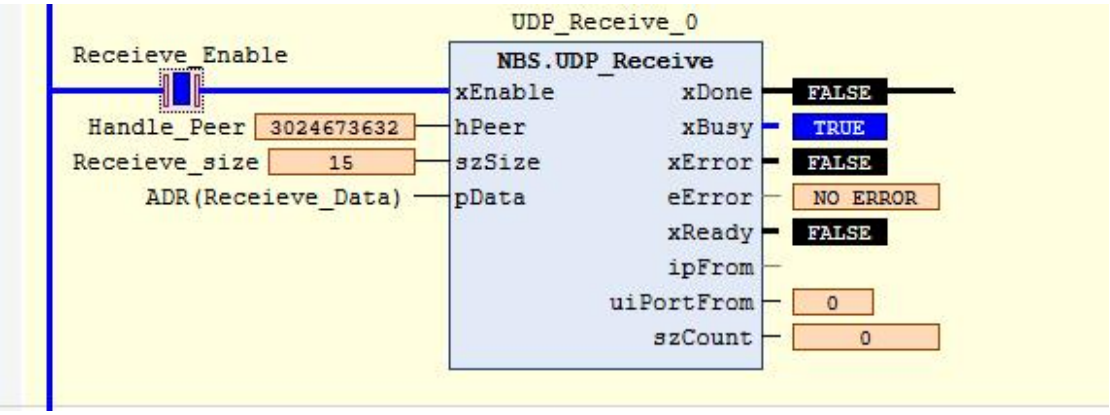
	布尔	位串				整数								实数		时刻、持续时间 日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
xEnable	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
hPeer	CAA. HANDLE																			
szSize	CAA. SIZE																			
pData	CAA. PVOID																			
xDone	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
xBusy	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
xError	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
eError	ERROR																			
xReady	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
ipFrom	IP_ADDR																			
uiPortFrom	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–
szCount	CAA. SIZE																			

- **程序示例**

ST： UDP 通信数据接收指令。当 UDP_Receive 指令 xEnable 为 TRUE 时，将从 UDP 通信缓冲区读取数据，xBusy 为 TRUE。若读取成功，xDone 置位一个扫描周期；读取的数据会被放置到地址为 pData 的变量里；同时 xReady 置位一个扫描周期；接收数据区域的实际大小值将会赋给 szCount，在一个扫描周期后 szCount 的值会被清零。

```
UDP_Receive_0(
    xEnable TRUE := Receieve_Enable TRUE,
    xDone FALSE => Receieve_xDone FALSE,
    xBusy TRUE => Receieve_xBusy TRUE,
    xError=> ,
    hPeer 3024673632 := Handle_Peer 3024673632,
    szSize 15 := 15,
    pData 3025087476 :=ADR(Receieve_Data) ,
    eError=> ,
    xReady FALSE =>Receieve_xReady FALSE ,
    ipFrom=> ,
    uiPortFrom=> ,
    szCount=> );
```

LD：UDP 通信数据接收指令。当 UDP_Receive 指令 xEnable 为 TRUE 时，将从 UDP 通信缓冲区读取数据，xBusy 为 TRUE。若读取成功，xDone 置位一个扫描周期；读取的数据会被放置到地址为 pData 的变量里；同时 xReady 置位一个扫描周期；接收数据区域的实际大小值将会赋给 szCount，在一个扫描周期后 szCount 的值会被清零。



- **注意事项**

➢ 如果通信过程中出现错误，xError将置TRUE，eError将被赋相关错误值（枚举型ERROR里的各种值），详细请看以下错误代码。

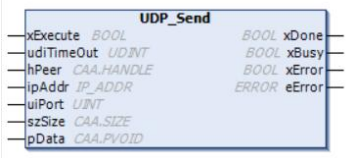
错误码	定义	描述
0	NO_ERROR	无错误
6000	FIRST_ERROR	保留
6001	TIME_OUT	保留
6002	INVALID_ADDR	客户端所连接服务器的IP地址无效
6003	INVALID_HANDLE	连接句柄无效

6004	INVALID_DATAPOINTER	数据指针无效
6005	INVALID_DATASIZE	数据大小无效
6006	UDP_RECEIVE_ERROR	UDP接收错误
6007	UDP_SEND_ERROR	UDP发送错误
6008	UDP_SEND_NOT_COMPLETE	UDP发送未完成
6009	UDP_OPEN_ERROR	UDP打开错误
6010	UDP_CLOSE_ERROR	UDP关闭错误
6011	TCP_SEND_ERROR	TCP发送错误
6012	TCP_RECEIVE_ERROR	TCP发送未完成
6013	TCP_OPEN_ERROR	TCP打开错误
6014	TCP_CONNECT_ERROR	TCP连接错误
6015	TCP_CLOSE_ERROR	TCP关闭错误
6016	TCP_SERVER_ERROR	TCP服务器错误
6017	WRONG_PARAMETER	参数错误
6018	ERROR_UNKNOWN	未知错误
6019	TCP_NO_CONNECTION	无TCP连接
6020	LOCTL_ERROR	内部错误（本机不支持）
6050	FIRST_MF	保留
6099	LAST_ERROR	保留

5.2.4 UDP_Send

UDP 通信数据发送

- **指令格式**

指令	名称	FB/FC	LD 表现	ST表现	库
UDP_Send	UDP通信数据发送	FB		UDP_Send(xExecute:= , udiTimeOut:= , xDone=> , xBusy=> , xError=> , hPeer:= , ipAddr:= , uiPort:= ,	CAA Net Base Services

				szSize:= , pData:= , eError=>);	

- 相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
xExecute	功能块使能	BOOL	TRUE-FALSE	-	若为TRUE（上升沿信号），则启动功能块运行
udiTimeOut	超时时间	UDINT	0-4294967295		建立通信后执行数据发送的超时时间，单位：μs
hPeer	通信句柄	CAA. HANDLE	-	-	UDP通信句柄
ipAddr	目标IP	IP_ADDR	-	-	发送数据的目标IP
uiPort	通信端口号	UINT	0-4294967297	-	通信端口号
szSize	数据大小	CAA. SIZE	-	-	接收数据区域大小，byte
pData	发送缓存	CAA. PVOID	-	-	指向属性数据的指针，从目标设备接收数据

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
xDone	完成	BOOL	TRUE-FALSE	-	客户端通信完成
xBusy	功能块运行中	BOOL	TRUE-FALSE	-	功能块运行中标志
xError	错误标志	BOOL	TRUE-FALSE	-	错误状态
eError	错误	ERROR	-	-	具体错误请参考CAA Net Base Services库的枚举体ERROR
xReady	成功建立连接标志位	BOOL	TRUE-FALSE	-	从缓冲区内读取数据，如果数据不为空，置位一个扫描

					周期的标志位
szCount	数据大小	CAA. SIZE	–	–	接收数据区域的实际大小

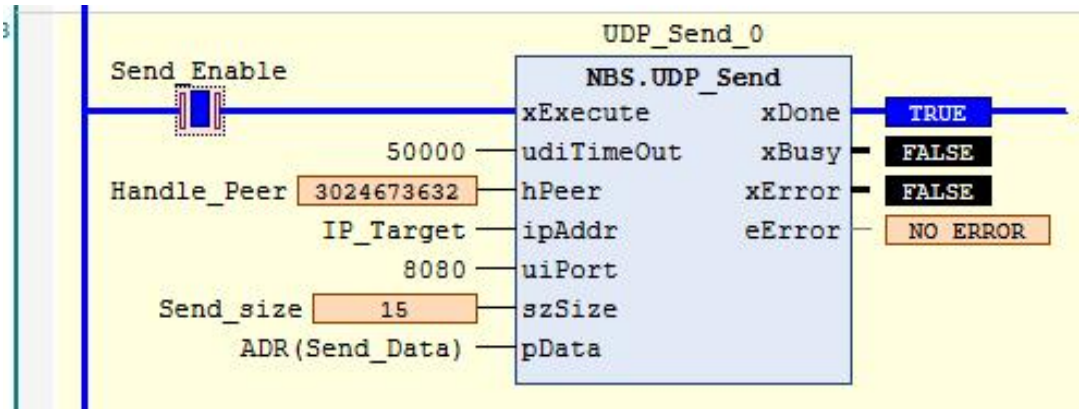
	布尔	位串				整数								实数		时刻、持续时间 日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
xEnable	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
udiTimeOut	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
hPeer	CAA. HANDLE																			
ipAddr	IP_ADDR																			
uiPort	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–
szSize	CAA. SIZE																			
pData	CAA. PVOID																			
xDone	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
xBusy	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
xError	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
eError	ERROR																			
xReady	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
szCount	CAA. SIZE																			

- **程序示例**

ST：UDP 通信数据发送指令。当 UDP_Send 指令 xExecute 为 TRUE 时，会将用户设置的发送缓冲区 pData 为首地址的长度为 szSize 的数据发送到目标设备。如果在超时时间内发送成功，xDone 置 TRUE。

```
30  UDP_Send_0(
31      xExecute TRUE := Send_Enable TRUE,
32      udiTimeOut 50000 :=50000 ,
33      xDone TRUE =>Send_xDone TRUE ,
34      xBusy FALSE => Send_xBusy FALSE,
35      xError=> ,
36      hPeer 3024673632 :=Handle_Peer 3024673632 ,
37      ipAddr:= IP_Target,
38      uiPort 8080 := 8080,
39      szSize 15 :=15 ,
40      pData 3025087491 := ADR(Send_Data),
41      eError=> );
42
```

LD：UDP 通信数据发送指令。当 UDP_Send 指令 xExecute 为 TRUE 时，会将用户设置的发送缓冲区 pData 为首地址的长度为 szSize 的数据发送到目标设备。如果在超时时间内发送成功，xDone 置 TRUE。



- **注意事项**

➢ 如果功能块出错，xError将置TRUE，eError将被赋相关错误值（枚举型ERROR里的各种值）。详细请看以下错误码。

错误码	定义	描述
0	NO_ERROR	无错误
6000	FIRST_ERROR	保留
6001	TIME_OUT	保留
6002	INVALID_ADDR	客户端所连接服务器的IP地址无效
6003	INVALID_HANDLE	连接句柄无效
6004	INVALID_DATAPOINTER	数据指针无效
6005	INVALID_DATASIZE	数据大小无效

6006	UDP_RECEIVE_ERROR	UDP接收错误
6007	UDP_SEND_ERROR	UDP发送错误
6008	UDP_SEND_NOT_COMPLETE	UDP发送未完成
6009	UDP_OPEN_ERROR	UDP打开错误
6010	UDP_CLOSE_ERROR	UDP关闭错误
6011	TCP_SEND_ERROR	TCP发送错误
6012	TCP_RECEIVE_ERROR	TCP发送未完成
6013	TCP_OPEN_ERROR	TCP打开错误
6014	TCP_CONNECT_ERROR	TCP连接错误
6015	TCP_CLOSE_ERROR	TCP关闭错误
6016	TCP_SERVER_ERROR	TCP服务器错误
6017	WRONG_PARAMETER	参数错误
6018	ERROR_UNKNOWN	未知错误
6019	TCP_NO_CONNECTION	无TCP连接
6020	LOCTL_ERROR	内部错误（本机不支持）
6050	FIRST_MF	保留
6099	LAST_ERROR	保留

5.3EtherCAT 通信指令

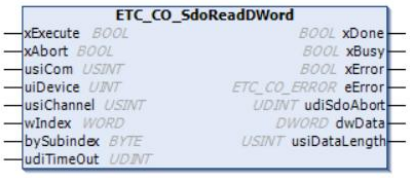
5.3.1 指令列表

指令类别	名称	FB/FC	功能
EtherCAT 通信指令	ETC_CO_SdoReadDWord	FB	EtherCAT 从站 SDO 读取
	ETC_CO_SdoRead4	FB	EtherCAT 从站 SDO 读取
	ETC_CO_SdoRead	FB	EtherCAT 从站 SDO 读取
	ETC_CO_SdoWrite_Dword	FB	EtherCAT 从站 SDO 写入
	ETC_CO_SdoWrite4	FB	EtherCAT 从站 SDO 写入
	ETC_CO_SdoWrite	FB	EtherCAT 从站 SDO 写入
	IoDrvEtherCAT_Diag	FB	EtherCAT 主站实例
	ETCSlave	FB	EtherCAT 从站实例

5.3.2 ETC_CO_SdoReadDWord

EtherCAT 从站 SD0 数据读取，且读取对象字典数值长度不大于 4 个字节

- **指令格式**

指令	名称	FB/FC	LD 表现	ST表现	库
ETC_CO_SdoReadDWord	读取 EtherCAT从站对象字典 SDO 数据	FB		ETC_CO_SdoReadDWord(xExecute:=, xAbort:=, usiCom:=, uiDevice:=, usiChannel:=, wIndex:=,	IODrvEtherCAT

				bySubindex:= , udiTimeOut:= , xDone=> , xBusy=> , xError=> , eError=> , udiSdoAbort=> , dwData=> , usiDataLength=>);	

- **相关变量**

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
xExecute	功能块使能	BOOL	TRUE-FALSE	-	若为TRUE（上升沿信号），则启动功能块运行
xAbort	功能块中止执行	BOOL	TRUE-FALSE	FALSE	若为TRUE，则当前功能块读取命令将终止
usiCom	EtherCAT主站个数	USINT	1-2	1	如果只使用一个EtherCAT主站，则usiCom为1。若使用多个主站，则第一个主站为1，第二个主站为2，以此类推。
uiDevice	从站站号	UINT	1-65535	0	从站站号如1001, 1002等
usiChannel	保留	USINT	1	1	保留
wIndex	对象字典主索引	WORD	1-65535	0	例如伺服的控制字对象字典16#6040
bySubindex	对象字典子索引	BYTE	0-255	0	例如伺服控制字对象字典子索引16#00
udiTimeOut	超时时间	UDINT	1-65535	0	读取从站参数的超时时间，单位：ms

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
xDone	读取完成	BOOL	TRUE-FALSE	-	读取参数已经成功，xDone为TRUE
xBusy	读取参数中	BOOL	TRUE-FALSE	-	读取参数中，xBusy为TRUE
xError	错误标志	BOOL	TRUE-FALSE	-	错误状态
eError	错误	ETC_CO_ERROR	-	-	具体错误请参考IODrvEtherCAT库的枚举体ETC_CO_ERROR
udiSdoAbort	中止	UDINT	0-4294967295	0	当设备出错时，可从此**输出变量**读取更多错误信息
dwData	读取的数据值	DWORD	0-4294967295	0	从从站读取到的SDO数据值
usiDataLength	读取到的数据长度	USINT	0-255	0	从从站读取到的SDO数据值长度

	布尔	位串				整数								实数		时刻、持续时间 日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
xExecute	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
xAbort	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
usiCom	_	_	_	_	_	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-
uiDevice	_	_	_	_	_	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-
usiChannel	_	_	_	_	_	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-
wIndex	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
bySubindex	_	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
udiTimeOut	_	_	_	_	_	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-

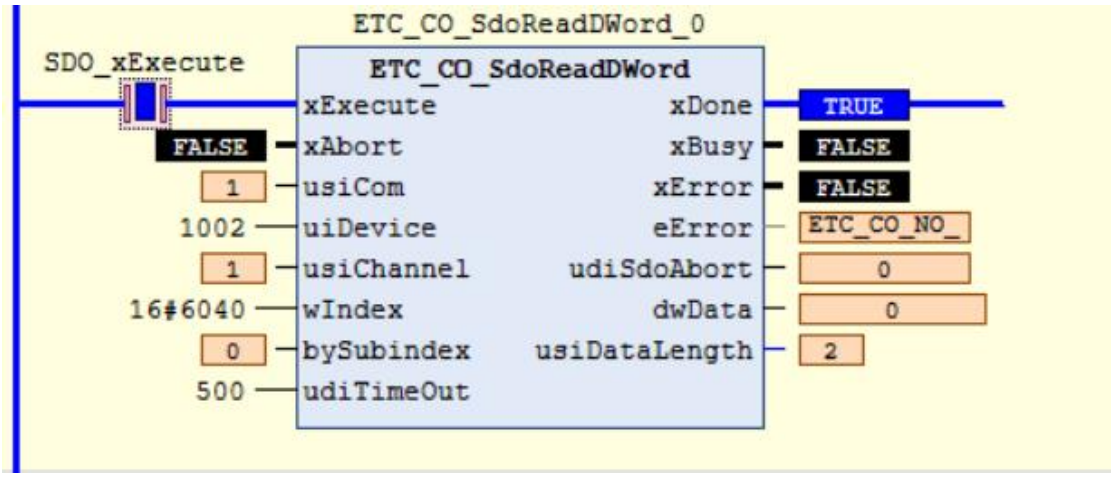
xDone	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
xBusy	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
xError	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
eError	ETC_CO_ERROR																			
udiSdoAbort	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–
dwData	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
usiDataLength	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–

- **程序示例**

ST：写伺服从站的控制字模式对象字典索引 16#6040，子索引 16#00。

```
ETC_CO_SdoReadDWord_0(
    xExecute:= SDO_xExecute,
    xAbort:= ,
    usiCom:= ,
    uiDevice:=1002 ,
    usiChannel:= ,
    wIndex:= 16#6040,
    bySubindex:= ,
    udiTimeOut:= 500,
    xDone=> ,
    xBusy=> ,
    xError=> ,
    eError=> ,
    udiSdoAbort=> ,
    dwData=>SDO_Data ,
    usiDataLength=> SDO_Length );
```

LD：写伺服从站的控制字模式对象字典索引 16#6040，子索引 16#00。



- **注意事项**

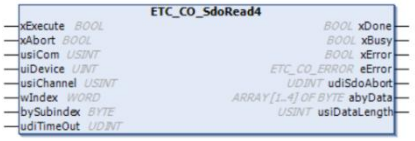
➢ 如果使用功能块中出现错误，xError将置TRUE，eError将被赋相关错误值（枚举型ETC_CO_ERROR里的各种值） 。详细请看以下错误代码。

名称	数据类型	描述
ETC_CO_NO_ERROR	WORD	-
ETC_CO_FIRST_ERROR	WORD	-
ETC_CO_OTHER_ERROR	WORD	此错误保留给所有SDO通道繁忙中！禁止更改
ETC_CO_DATA_OVERFLOW	WORD	-
ETC_CO_TIMEOUT	WORD	-
ETC_CO_FIRST_MF	WORD	-
ETC_CO_LAST_ERROR	WORD	-

5.3.3 ETC_CO_SdoRead4

EtherCAT 从站 SD0 数据读取，且读取对象字典数值长度不大于 4 个字节。与 ETC_CO_SdoReadDWord 功能块不同在于，ETC_CO_SdoRead4 是将读取到的数值存储在字节类型数组中。

- **指令格式**

指令	名称	FB/FC	LD 表现	ST表现	库
ETC_CO_SdoRead4	读取EtherCAT从站对象字典SDO数据	FB		ETC_CO_SdoRead4(xExecute:= , xAbort:= , usiCom:= , uiDevice:= , usiChannel:= , wIndex:= , bySubindex:= , udiTimeOut:= , xDone=> , xBusy=> , xError=> ,	IODrvEtherCAT

				eError=> , udiSdoAbort=> , abyData=> , usiDataLength=>);	

- **相关变量**

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
xExecute	功能块使能	BOOL	TRUE-FALSE	-	若为TRUE（上升沿信号），则启动功能块运行
xAbort	功能块中止执行	BOOL	TRUE-FALSE	FALSE	若为TRUE，则当前功能块读取命令将终止
usiCom	EtherCAT主站个数	USINT	1-2	1	如果只使用一个EtherCAT主站，则usiCom为1。若使用多个主站，则第一个主站为1，第二个主站为2，以此类推。
uiDevice	从站站号	UINT	1-65535	0	从站站号如1001, 1002等
usiChannel	保留	USINT	1	1	保留
wIndex	对象字典主索引	WORD	1-65535	0	例如伺服的控制字对象字典16#6040
bySubindex	对象字典子索引	BYTE	0-255	0	例如伺服控制字对象字典子索引16#00
udiTimeOut	超时时间	UDINT	1-65535	0	读取从站参数的超时时间，单位：ms

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
xDone	读取完成	BOOL	TRUE-FALSE	-	读取参数已经成功，

					xDone为TRUE
xBusy	读取参数中	BOOL	TRUE-FALSE	-	读取参数中, xBusy为TRUE
xError	错误标志	BOOL	TRUE-FALSE	-	错误状态
eError	错误	ETC_CO_ERROR	-	-	具体错误请参考IODrvEtherCAT库的枚举体ETC_CO_ERROR
udiSdoAbort	中止	UDINT	0-4294967295	0	当设备出错时，可从此**输出变量**读取更多错误信息
abyData	读取的数据值	ARRAY［1..4］OF BYTE	-	0	从从站读取到的SDO数据值，数值填充顺序从数组索引1到4
usiDataLength	读取到的数据长度	BYTE	0-255	0	从从站读取到的SDO数据值长度

	布尔	位串				整数								实数		时刻、持续时间 日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
xExecute	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
xAbort	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
usiCom	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-
uiDevice	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-
usiChannel	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-
wIndex	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
bySubindex	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
udiTimeOut	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-
xDone	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
xBusy	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

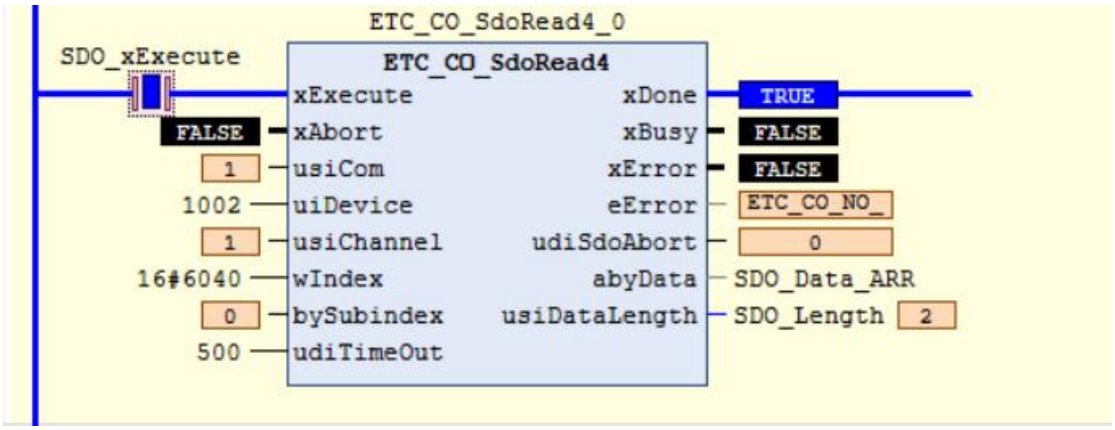
xError	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
eError	ETC_CO_ERROR																		
udiSdoAbort	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–
abyData	ARRAY［1..4］OF BYTE																		
usiDataLength	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–

- **程序示例**

ST：写伺服从站的控制字模式对象字典索引 16#6040，子索引 16#00。（注意：读取的数值存储在字节类型数组，数组大小为4个字节）

```
ETC_CO_SdoRead4_0(
    xExecute:= SDO_xExecute,
    xAbort:= ,
    usiCom:= ,
    uiDevice:=1002 ,
    usiChannel:= ,
    wIndex:= 16#6040,
    bySubindex:= ,
    udiTimeOut:=500 ,
    xDone=> ,
    xBusy=> ,
    xError=> ,
    eError=> ,
    udiSdoAbort=> ,
    abyData=> SDO_Data_ARR,
    usiDataLength=> SDO_Length );
```

LD：写伺服从站的控制字模式对象字典索引 16#6040，子索引 16#00。（注意：读取的数值存储在字节类型数组，数组大小为4个字节）



- **注意事项**

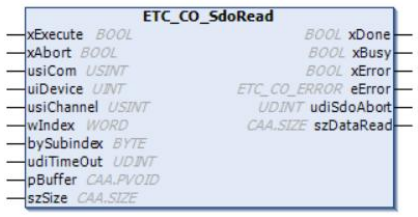
➢ 如果使用功能块中出现错误，xError将置TRUE，eError将被赋相关错误值（枚举型ETC_CO_ERROR里的各种值） 。详细请看以下错误代码。

名称	数据类型	描述
ETC_CO_NO_ERROR	WORD	-
ETC_CO_FIRST_ERROR	WORD	-
ETC_CO_OTHER_ERROR	WORD	此错误保留给所有SDO通道繁忙中！禁止更改
ETC_CO_DATA_OVERFLOW	WORD	-
ETC_CO_TIMEOUT	WORD	-
ETC_CO_FIRST_MF	WORD	-
ETC_CO_LAST_ERROR	WORD	-

5.3.4 ETC_CO_SdoRead

EtherCAT 从站 SD0 数据读取，与 ETC_CO_SdoReadDWord 或 ETC_CO_SdoRead4 功能块不同，ETC_CO_SdoRead 功能块读取的对象字典数值长度可大于 4 个字节。

- **指令格式**

指令	名称	FB/FC	LD 表现	ST表现	库
ETC_CO_SdoRead	读取EtherCAT从站对象字典SDO数据	FB		ETC_CO_SdoRead (xExecute:=, xAbort:=, usiCom:=, uiDevice:=, usiChannel:=, wIndex:=, bySubindex:=, udiTimeOut:=, pBuffer:=, szSize:=, xDone=>, xBusy=>,	IODrvEtherCAT

				xError=> , eError=> , udiSdoAbort=> , szDataRead=>) ;	

- **相关变量**

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
xExecute	功能块使能	BOOL	TRUE-FALSE	-	若为TRUE（上升沿信号），则启动功能块运行
xAbort	功能块中止执行	BOOL	TRUE-FALSE	FALSE	若为TRUE，则当前功能块读取命令将终止
usiCom	EtherCAT主站个数	USINT	1-2	1	如果只使用一个EtherCAT主站，则usiCom为1。若使用多个主站，则第一个主站为1，第二个主站为2，以此类推。
uiDevice	从站站号	UINT	1-65535	0	从站站号如1001, 1002等
usiChannel	保留	USINT	1	1	保留
wIndex	对象字典主索引	WORD	1-65535	0	例如伺服的控制字对象字典16#6040
bySubindex	对象字典子索引	BYTE	0-255	0	例如伺服控制字对象字典子索引16#00
udiTimeOut	超时时间	UDINT	1-65535	0	读取从站参数的超时时间，单位：ms
pBuffer	数据缓冲区的指针	CAA. PVOID	—	0	数据成功读取后的存储区域
szSize	数据缓冲区大小	CAA. SIZE	-	0	数据缓冲（pBuffer）的大小

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
xDone	读取完成	BOOL	TRUE-FALSE	-	读取参数已经成功，xDone为TRUE
xBusy	读取参数中	BOOL	TRUE-FALSE	-	读取参数中, xBusy为TRUE
xError	错误标志	BOOL	TRUE-FALSE	-	错误状态
eError	错误	ETC_CO_ERROR	-	-	具体错误请参考IODrvEtherCAT库的枚举体ETC_CO_ERROR
udiSdoAbort	中止	UDINT	0-4294967295	0	当设备出错时，可从此**输出变量**读取更多错误信息

	布尔	位串				整数								实数		时刻、持续时间 日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
xExecute	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
xAbort	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
usiCom	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-
uiDevice	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-
usiChannel	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-
wIndex	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
bySubindex	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
udiTimeOut	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-
pBuffer	CAA. PVOID																			

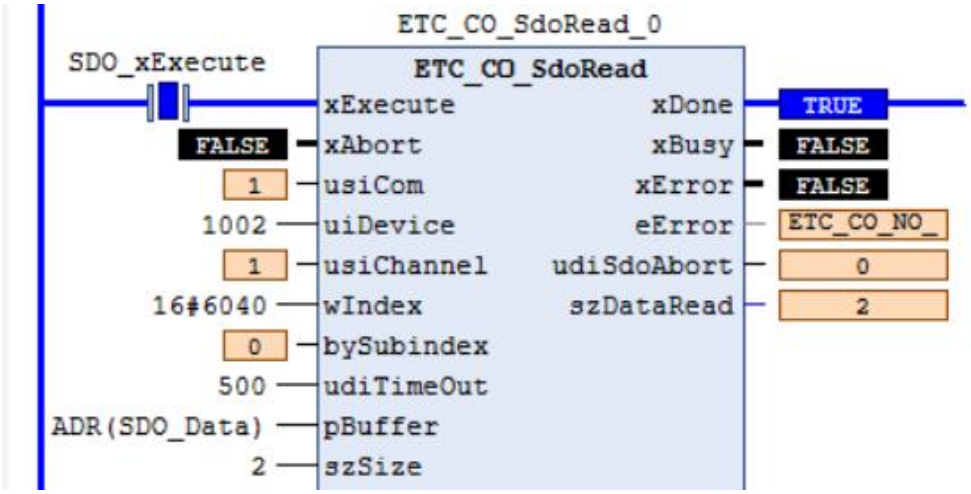
szSize	CAA. SIZE																			
xDone	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
xBusy	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
xError	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
eError	ETC_CO_ERROR																			
udiSdo Abort	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–

- **程序示例**

ST：写伺服从站的控制字模式对象字典索引 16#6040，子索引 16#00。

```
ETC_CO_SdoRead_0(
    xExecute TRUE := SDO_xExecute TRUE,
    xAbort:= ,
    usiCom:= ,
    uiDevice 1002 :=1002 ,
    usiChannel:= ,
    wIndex 24640 := 16#6040,
    bySubindex:= ,
    udiTimeOut 500 :=500 ,
    pBuffer 3019616116 := ADR(SDO_Data 0 ),
    szSize 2 :=2 ,
    xDone=> ,
    xBusy=> ,
    xError=> ,
    eError=> ,
    udiSdoAbort=> ,
    szDataRead 2 =>SDO_Length 2 );
```

LD：写伺服从站的控制字模式对象字典索引 16#6040，子索引 16#00。



- **注意事项**

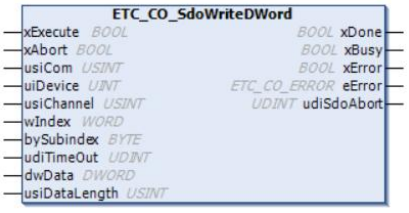
➢ 如果使用功能块中出现错误，xError将置TRUE，eError将被赋相关错误值（枚举型ETC_CO_ERROR里的各种值） 。详细请看以下错误代码。

名称	数据类型	描述
ETC_CO_NO_ERROR	WORD	-
ETC_CO_FIRST_ERROR	WORD	-
ETC_CO_OTHER_ERROR	WORD	此错误保留给所有SDO通道繁忙中！禁止更改
ETC_CO_DATA_OVERFLOW	WORD	-
ETC_CO_TIMEOUT	WORD	-
ETC_CO_FIRST_MF	WORD	-
ETC_CO_LAST_ERROR	WORD	-

5.3.5 ETC_CO_SdoWriteDword

EtherCAT 从站 SD0 数据写入，且写入对象字典数值长度不大于 4 个字节。

- **指令格式**

指令	名称	FB/FC	LD 表现	ST表现	库
ETC_CO_SdoWriteDWord	EtherCAT从站对象字典SDO数据写入	FB		ETC_CO_SdoWriteDWord(xExecute:= , xAbort:= , usiCom:= , uiDevice:= , usiChannel:= , wIndex:= , bySubindex:= , udiTimeOut:= , dwData:= , usiDataLength:= , xDone=> , xBusy=> , xError=> , eError=> , udiSdoAbort=>);	IODrvEtherCAT

- **相关变量**

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
xExecute	功能块使能	BOOL	TRUE-FALSE	-	若为TRUE（上升沿信号），则启动功能块运行
xAbort	功能块中止执行	BOOL	TRUE-FALSE	FALSE	若为TRUE，则当前功能块读取命令将终止
usiCom	EtherCAT主站个数	USINT	1-2	1	如果只使用一个EtherCAT主站，则usiCom为1。若使用多个主站，则第一个主站为1，第二个主站为2，以此类推。
uiDevice	从站站号	UINT	1-65535	0	从站站号如1001, 1002等
usiChannel	保留	USINT	1	1	保留
wIndex	对象字典主索引	WORD	1-65535	0	例如伺服的控制字对象字典16#6040
bySubindex	对象字典子索引	BYTE	0-255	0	例如伺服控制字对象字典子索引16#00
udiTimeOut	超时时间	UDINT	1-65535	0	读取从站参数的超时时间，单位：ms
dwData	写入的数据值	DWORD	0-4294967295	0	写入从站SDO的数据值
usiDataLength	读取到的数据长度	USINT	0-255	0	写入从站SDO数据值的长度

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
xDone	读取完成	BOOL	TRUE-FALSE	-	读取参数已经成功，xDone为TRUE
xBusy	读取参数中	BOOL	TRUE-FALSE	-	读取参数中，xBusy为TRUE
xError	错误标志	BOOL	TRUE-FALSE	-	错误状态

eError	错误	ETC_CO_ERROR	–	–	具体错误请参考IODrvEtherCAT库的枚举体ETC_CO_ERROR
udiSdoAbort	中止	UDINT	0-4294967295	0	当设备出错时，可从此**输出变量**读取更多错误信息

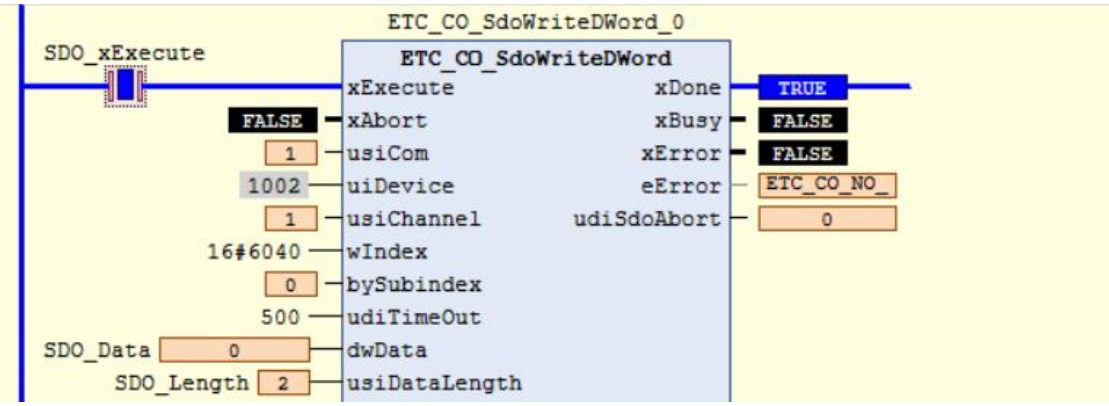
	布尔	位串				整数								实数		时刻、持续时间 日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
xExecute	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
xAbort	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
usiCom	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–
uiDevice	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–
usiChannel	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–
wIndex	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
bySubindex	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
udiTimeOut	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–
dwData	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
usiDataLength	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–
xDone	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
xBusy	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
xError	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
eError	ETC_CO_ERROR																			
udiSdoAbort	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–

- **程序示例**

ST：写伺服从站的控制字模式对象字典索引 16#6040，子索引 16#00。（注意：写入的数值存储在字节类型数组，数组大小为4个字节）

```
ETC_CO_SdoWriteDWord_0(
    xExecute TRUE :=SDO_xExecute TRUE ,
    xAbort:= ,
    usiCom:= ,
    uiDevice 1002 := 1002,
    usiChannel:= ,
    wIndex 24640 := 16#6040,
    bySubindex:= ,
    udiTimeOut 500 := 500,
    dwData 0 := SDO_Data 0,
    usiDataLength 2 := SDO_Length 2,
    xDone=> ,
    xBusy=> ,
    xError=> ,
    eError=> ,
    udiSdoAbort=> );
```

LD：写伺服从站的控制字模式对象字典索引 16#6040，子索引 16#00。（注意：写入的数值存储在字节类型数组，数组大小为4个字节）



- **注意事项**

➢ 如果使用功能块中出现错误，xError将置TRUE，eError将被赋相关错误值（枚举型ETC_CO_ERROR里的各种值） 。详细请看以下错误代码。

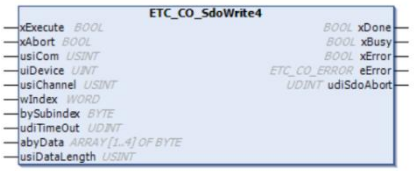
名称	数据类型	描述
ETC_CO_NO_ERROR	WORD	-
ETC_CO_FIRST_ERROR	WORD	-

ETC_CO_OTHER_ERROR	WORD	此错误保留给所有SDO通道繁忙中！禁止更改
ETC_CO_DATA_OVERFLOW	WORD	-
ETC_CO_TIMEOUT	WORD	-
ETC_CO_FIRST_MF	WORD	-
ETC_CO_LAST_ERROR	WORD	-

5.3.6 ETC_CO_SdoWrite4

EtherCAT 从站 SD0 数据写入，且写入对象字典数值长度不大于 4 个字节。与 ETC_CO_SdoWriteDWord 功能块不同在于，ETC_CO_SdoWrite4 是将数值以字节类型数组是写入。

- **指令格式**

指令	名称	FB/FC	LD 表现	ST表现	库
ETC_CO_SdoWrite4	EtherCAT从站对象字典SDO数据写入	FB		ETC_CO_SdoWrite4(xExecute:= , xAbort:= , usiCom:= , uiDevice:= , usiChannel:= , wIndex:= , bySubindex:= , udiTimeOut:= , abyData:= , usiDataLength:= , xDone=> , xBusy=> , xError=> , eError=> , udiSdoAbort=>);	IODrvEtherCAT

- **相关变量**

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
xExecute	功能块使能	BOOL	TRUE-FALSE	-	若为TRUE（上升沿信号），则启动功能块运行
xAbort	功能块中止执行	BOOL	TRUE-FALSE	FALSE	若为TRUE，则当前功能块读取命令将终止
usiCom	EtherCAT主站个数	USINT	1-2	1	如果只使用一个EtherCAT主站，则usiCom为1。若使用多个主站，则第一个主站为1，第二个主站为2，以此类推。
uiDevice	从站站号	UINT	1-65535	1	从站站号如1001, 1002等
usiChannel	保留	USINT	1	1	保留
wIndex	对象字典主索引	WORD	1-65535	1	例如伺服的控制字对象字典16#6040
bySubindex	对象字典子索引	BYTE	0-255	0	例如伺服控制字对象字典子索引16#00
udiTimeOut	超时时间	UDINT	1-65535	1	读取从站参数的超时时间，单位：ms
abyData	读取的数据值	ARRAY [1..4]OF BYTE	-	0	从从站读取到的SDO数据值，数值填充顺序从数组索引1到4
usiDataLength	读取到的数据长度	USINT	0-255	0	写入从站SDO数据值的长度

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
xDone	读取完成	BOOL	TRUE-FALSE	-	读取参数已经成功，xDone为TRUE
xBusy	读取参数中	BOOL	TRUE-FALSE	-	读取参数中，xBusy为TRUE

xError	错误标志	BOOL	TRUE-FALSE	-	错误状态
eError	错误	ETC_CO_ERROR	-	-	具体错误请参考IODrvEtherCAT库的枚举体ETC_CO_ERROR
udiSdoAbort	中止	UDINT	0-4294967295	0	当设备出错时，可从此**输出变量**读取更多错误信息

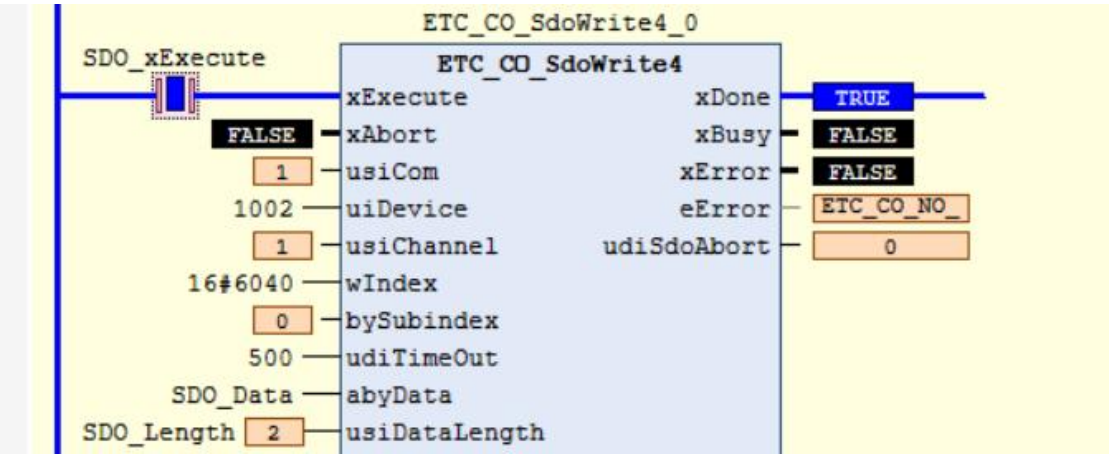
	布尔	位串				整数								实数		时刻、持续时间 日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
xExecute	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
xAbort	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
usiCom	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-
uiDevice	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-
usiChannel	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-
wIndex	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
bySubindex	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
udiTimeOut	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-
dwData	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
usiDataLength	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-
xDone	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
xBusy	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
xError	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
eError	ETC_CO_ERROR																			
udiSdoAbort	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-

- **程序示例**

ST：写伺服从站的控制字模式对象字典索引 16#6040，子索引 16#00。（注意：写入的数值存储在字节类型数组，数组大小为4个字节）

```
ETC_CO_SdoWrite4_0(
    xExecute TRUE := SDO_xExecute TRUE,
    xAbort:= ,
    usiCom:= ,
    uiDevice 1002 := 1002,
    usiChannel:= ,
    wIndex 24640 := 16#6040,
    bySubindex:= ,
    udiTimeOut 500 :=500 ,
    abyData:=SDO_Data ,
    usiDataLength 2 :=SDO_Length 2 ,
    xDone=> ,
    xBusy=> ,
    xError=> ,
    eError=> ,
    udiSdoAbort=> );
```

LD：写伺服从站的控制字模式对象字典索引16#6040，子索引 16#00。（注意：写入的数值存储在字节类型数组，数组大小为4个字节）



- **注意事项**

➢ 如果使用功能块中出现错误，xError将置TRUE，eError将被赋相关错误值（枚举型ETC_CO_ERROR里的各种值） 。详细请看以下错误代码。

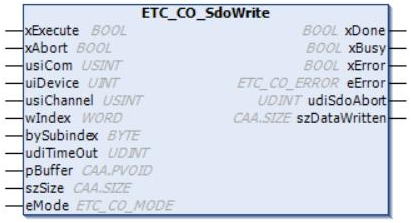
名称	数据类型	描述
ETC_CO_NO_ERROR	WORD	–
ETC_CO_FIRST_ERROR	WORD	–

ETC_CO_OTHER_ERROR	WORD	此错误保留给所有SDO通道繁忙中！禁止更改
ETC_CO_DATA_OVERFLOW	WORD	-
ETC_CO_TIMEOUT	WORD	-
ETC_CO_FIRST_MF	WORD	-
ETC_CO_LAST_ERROR	WORD	-

5.3.7 ETC_CO_SdoWrite

EtherCAT 从站 SD0 数据写入。与 ETC_CO_SdoWriteDWord 或 ETC_CO_SdoWrite4 功能块不同，ETC_CO_SdoWrite 功能块写入的对象字典数值长度可大于 4 个字节。

- **指令格式**

指令	名称	FB/FC	LD 表现	ST表现	库
ETC_CO_SdoWrite	EtherCAT从站对象字典SDO数据写入	FB		ETC_CO_SdoWrite(xExecute:=, xAbort:=, usiCom:=, uiDevice:=, usiChannel:=, wIndex:=, bySubindex:=, udiTimeOut:=, pBuffer:=, szSize:=, eMode:=, xDone=>, xBusy=>, xError=>, eError=>, udiSdoAbort=>, szDataWritten=>);	IODrvEtherCAT

- **相关变量**

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
xExecute	功能块使能	BOOL	TRUE-FALSE	-	若为TRUE（上升沿信号），则启动功能块运行
xAbort	功能块中止执行	BOOL	TRUE-FALSE	FALSE	若为TRUE，则当前功能块读取命令将终止
usiCom	EtherCAT主站个数	USINT	1-2	1	如果只使用一个EtherCAT主站，则usiCom为1。若使用多个主站，则第一个主站为1，第二个主站为2，以此类推。
uiDevice	从站站号	UINT	1-65535	1	从站站号如1001, 1002等
usiChannel	保留	USINT	1	1	保留
wIndex	对象字典主索引	WORD	1-65535	1	例如伺服的控制字对象字典16#6040
bySubindex	对象字典子索引	BYTE	0-255	0	例如伺服控制字对象字典子索引16#00
udiTimeOut	超时时间	UDINT	1-65535	1	读取从站参数的超时时间，单位：ms
pBuffer	数据缓冲区的指针	CAA. PVOID	—	—	写入的数据指针起始地址
szSize	数据缓冲区大小	CAA. SIZE	-	—	写入数据长度
eMode	写入模式选择	ETC_CO_MODE	-	—	写入模式选择

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
xDone	读取完成	BOOL	TRUE-FALSE	-	读取参数已经成功，xDone为TRUE

xBusy	读取参数中	BOOL	TRUE-FALSE	-	读取参数中, xBusy为TRUE
xError	错误标志	BOOL	TRUE-FALSE	-	错误状态
eError	错误	ETC_CO_ERROR	-	-	具体错误请参考IODrvEtherCAT库的枚举体ETC_CO_ERROR
udiSdoAbort	中止	UDINT	0-4294967295	0	当设备出错时，可从此**输出变量**读取更多错误信息
szDataWritten	完成写入数据长度	CAA. SIZE	-	-	完成写入数据长度

	布尔	位串				整数								实数		时刻、持续时间 日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
xExecute	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
xAbort	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
usiCom	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-
uiDevice	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-
usiChannel	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-
wIndex	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
bySubindex	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
udiTimeOut	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-
pBuffer	CAA. PVOID																			
szSize	CAA. SIZE																			
eMode	ETC_CO_MODE																			
xDone	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
xBusy	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

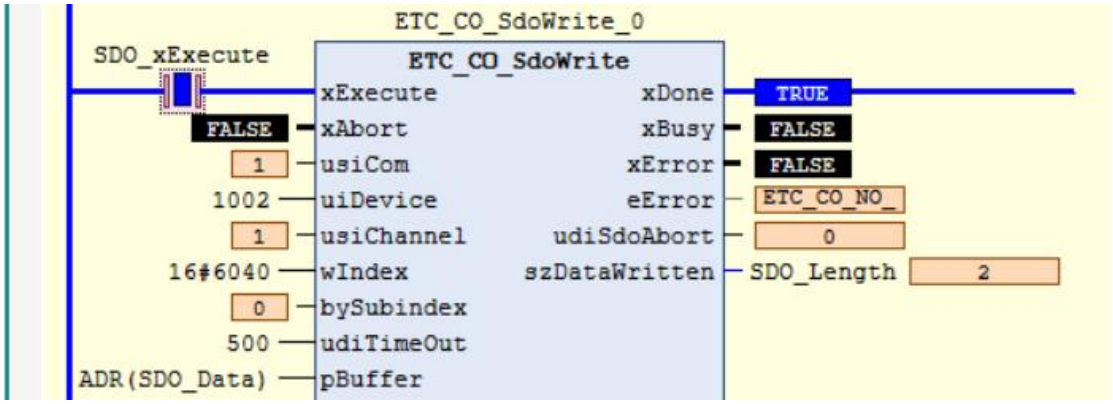
xError	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
eError	ETC_CO_ERROR																			
udiSdo Abort	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–
szDat Written	CAA. SIZE																			

- **程序示例**

ST：写伺服从站的控制字模式对象字典索引 16#6040，子索引 16#00。

```
ETC_CO_SdoWrite_0(
    xExecute TRUE := SDO_xExecute TRUE,
    xAbort:= ,
    usiCom:= ,
    uiDevice 1002 := 1002,
    usiChannel:= ,
    wIndex 24640 :=16#6040 ,
    bySubindex:= ,
    udiTimeOut 500 := 500,
    pBuffer 3019616116 := ADR(SDO_Data 16#00000000 ),
    szSize 2 := 2,
    eMode:= ,
    xDone=> ,
    xBusy=> ,
    xError=> ,
    eError=> ,
    udiSdoAbort=> ,
    szDataWritten 2 => SDO_Length 2 );
```

LD：写伺服从站的控制字模式对象字典索引 16#6040，子索引 16#00。



- **注意事项**

➢ 如果使用功能块中出现错误，xError将置TRUE，eError将被赋相关错误值（枚举型ETC_CO_ERROR里的各种值） 。详细请看以下错误代码。

名称	数据类型	描述
ETC_CO_NO_ERROR	WORD	-
ETC_CO_FIRST_ERROR	WORD	-
ETC_CO_OTHER_ERROR	WORD	此错误保留给所有SDO通道繁忙中！禁止更改
ETC_CO_DATA_OVERFLOW	WORD	-
ETC_CO_TIMEOUT	WORD	-
ETC_CO_FIRST_MF	WORD	-
ETC_CO_LAST_ERROR	WORD	-

5.3.8 IoDrvEtherCAT_Diag

EtherCAT 主站实例，实现 EtherCat 主站重启。

- **指令格式**

指令	名称	FB/FC	LD 表现	ST表现	库
IoDrvEthercat	EtherCAT主站实例	FB		IoDrvEtherCAT_Diag(xRestart:= , xStopBus:= , xConfigFinished=> , xDistributedClockInSync=> , xError=> , xSyncInWindow=>);	IODrvEtherCAT

- **相关变量**

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
xRestart	EtherCAT主站重启	BOOL	TRUE-FALSE	FALSE	若输入为TRUE（上升沿信号）则EtherCAT

					主站重启
xStopBus	EtherCAT 主站停止	BOOL	TRUE-FALSE	FALSE	停止EtherCAT主站

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
xConfigFinished	EtherCAT 主站配置完成	BOOL	TRUE-FALSE	FALSE	若EtherCAT 配置完成通讯正在运行，此值输出为TRUE
xDistributedClockInSync	DC同步信号	BOOL	TRUE-FALSE	FALSE	若同步模式为DC模式完成后，此时输出为TRUE
xError	错误标志	BOOL	TRUE-FALSE	-	错误状态
xSyncInWindow	同步窗口	BOOL	TRUE-FALSE	FALSE	若同步性在同步窗口内，此时输出为TRUE

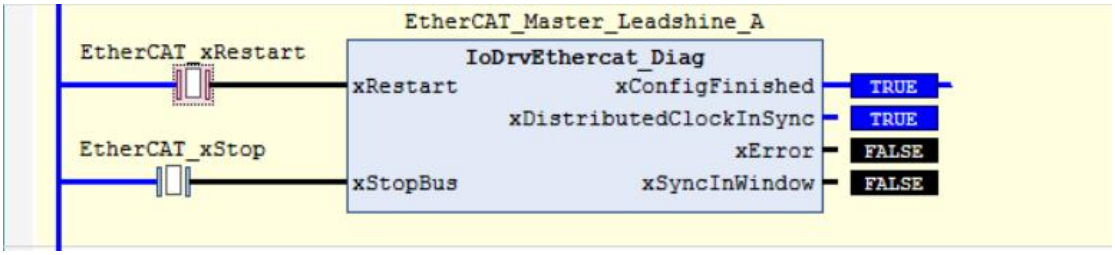
	布尔	位串				整数								实数		时刻、持续时间 日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
xRestart	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
xStopBus	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
xConfigFinished	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
xDistributedClockInSync	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
xError	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
xSyncnWindow	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

- **程序示例**

ST：若设备只有一个主站，实例化名称为EtherCAT_Master_Leadshine_A，若EtherCAT配置完成，通讯正在运行且同步模式为DC模式完成后，则xConfigFinished及xDistributedClockInSync此值输出为TRUE。

```
EtherCAT_Master_Leadshine_A(
    xRestart:= EtherCAT_xRestart,
    xStopBus:=EtherCAT_xStop ,
    xConfigFinished=> EtherCAT_xConfigFinished,
    xDistributedClockInSync=> EtherCAT_xDistributedClockInSync,
    xError=> ,
    xSyncInWindow=> );
```

LD：若设备只有一个主站，实例化名称为EtherCAT_Master_Leadshine_A，若EtherCAT配置完成，通讯正在运行且同步模式为DC模式完成后，则xConfigFinished及xDistributedClockInSync此值输出为TRUE。



- **注意事项**

➢ 实例化的名称需要与主站总线的名称相同，这样才能重启主站总线。

5.3.9 ETCSlave

获取从站状态

- **指令格式**

指令	名称	FB/FC	LD 表现	ST表现	库
IoDrvEthercat	获取从站状态	FB		ETCSlave(xSetOperational:= , wState=>);	IODrvEtherCAT

- **相关变量**

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
xSetOperational	尝试将从站设置成ETC_SLAVE_OPERATIONAL状态	BOOL	TRUE-FALSE	FALSE	若输入为TRUE（上升沿信号）则尝试将从站的状态机设置成ETC_SLAVE_OPERATIONAL状态

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
wState	从站状态	ETC_SLAVE_STATE	-	-	读取从站状态

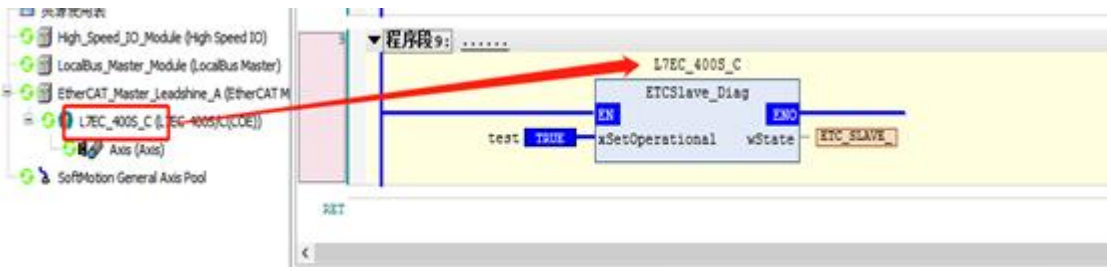
	布尔	位串				整数								实数		时刻、持续时间 日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
xSetOperational	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
wState	ETC_SLAVE_STATE																			

- **程序示例**

ST： 以 L7EC_400S为例添加相关伺服从站的实例名称，定义ETCSlave_state为ETC_SLAVE_STATE枚举体变量。（详细状态请看注意事项）

```
1  L7EC_400S(xSetOperational:= , wState 0 =>ETCSlave_state 0 );
2
```

LD： 以 L7EC_400S为例添加相关伺服从站的实例名称，定义ETCSlave_state为ETC_SLAVE_STATE枚举体变量。



- **注意事项**

➢ wState为枚举体ETC_SLAVE_STATE变量，读取从站状态，以下表格为ETC_SLAVE_STATE枚举体变量的信息。

名称	数据类型	描述
ETC_SLAVE_BOOT	WORD	取值为0
ETC_SLAVE_Init	WORD	取值为1
ETC_SLAVE_PREOPERATIONAL	WORD	取值为2
ETC_SLAVE_SAVEOPERATIONAL	WORD	取值为4
ETC_SLAVE_OPERATIONAL	WORD	取值为8

5.4Ethernet-IP 通信指令

5.4.1 指令列表

指令类别	名称	FB/FC	功能
Ethernet-IP 通信指令	IoDrvEtherNetIP_Diag	FB	Ethernet/IP 主站实例
	RemoteAdapter	FB	Ethernet/IP 从站实例
	Generic_Service	FB	CIP 通用显示服务
	Get_Attributes_All	FB	获取对象实例全部属性
	Get_Attribute_Single	FB	获取对象实例单个属性
	Set_Attributes_All	FB	设置对象实例所有属性
	Set_Attribute_Single	FB	设置对象实例单个属性

5.4.2 IoDrvEtherNetIP_Diag

Ethernet/IP 主站实例

- **指令格式**

指令	名称	FB/FC	LD 表现	ST表现	库
IoDrvEtherNetIP	Ethernet/IP主站实例	FB		IoDrvEtherNetIP(xReset:= , eState=> , eError=>);	IoDrvEtherNetIP

- **相关变量**

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
xReset	复位	BOOL	TRUE-FALSE	FALSE	若输入为TRUE（上升沿信号）则复位

					主站

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
eState	状态	ScannerState	-	-	EtherNet/IP 做扫描器时的状态
eError	错误	ERROR	-	-	具体错误请参考IoDrvEtherNetIP库的枚举体ERROR

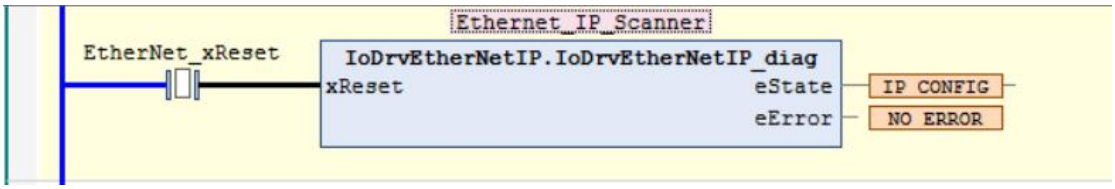
	布尔	位串				整数								实数		时刻、持续时间 日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
xReset	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
eState	ScannerState																			
eError	ERROR																			

- **程序示例**

ST：实例化名称为Ethernet_IP_Scanner， eState为当前做扫描器时的状态。

```
Ethernet_IP_Scanner(xReset:= EtherNet_xReset, eState=> EtherNet_state, eError=> );
```

LD：实例化名称为Ethernet_IP_Scanner， eState为当前做扫描器时的状态。



- **注意事项**

➢ 如果使用功能块中出现错误，eError将被赋相关错误值（枚举型ERROR里的各种值） 。详细请看以下错误代码。

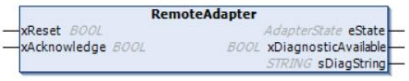
错误码	定义	描述
0	NO_ERROR	-
16#1	INVALID_COMMAND	-

16#2	OUT_OF_MEMORY	-
16#3	INVALID_DATA	-
16#64	INVALID_SESSION_HANDLE	-
16#65	INVALID_LENGTH	-
16#69	UNSUPPORTED_PROTOCOL_VERSION	-
16#AA	NBS_ERROR	-
	NBS_RCV_ERROR	-
	NBS_SND_ERROR	-
	ENCAPSULATION_ERROR	-
	TCPIP_CONFIG_ERROR	-
	UDP_CONFIG_ERROR	-
	UDP_RECV_ERROR	-
	UDP_SEND_ERROR	-
	UDP_CLOSE_ERROR	-
	NULL_POINTER	-
	DEVICE_STATE_ERROR	-
	RECONFIGURATION_FAILED	-
	PERFORMANCE_MONITOR_DISABLED	-
	INVALID_MEASURING_POINT	-
	IP_CONFIG_ERROR	-

5.4.3 RemoteAdapter

Ethernet/IP 从站实例

- **指令格式**

指令	名称	FB/FC	LD 表现	ST表现	库
Ethernet/IP 从站实例	Ethernet/IP从站实例	FB		RemoteAdapter (xReset:= , xAcknowledge:= , eState=> , xDiagnostic	IoDrvEtherNetIP

				Available=> , sDiagString=>) ;	

- **相关变量**

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
xReset	复位	BOOL	TRUE-FALSE	FALSE	若输入为TRUE（上升沿信号）则复位从站
xAcknowledge	确认诊断信息	BOOL	TRUE-FALSE	FALSE	若输入为TRUE（上升沿信号）则确认诊断信息

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
eState	状态	AdapterState	-	-	EtherNet/IP 做适配器时的状态
xDiagnosticAvailable	诊断可用	BOOL	TRUE-FALSE	FALSE	
sDiagString	诊断信息	STRING	-	-	诊断信息字符串，显示适配器的诊断信息

	布尔	位串				整数								实数		时刻、持续时间 日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
xReset	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
xAcknowledge	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
eState	AdapterState																			
xDiagnosticAvailabl	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

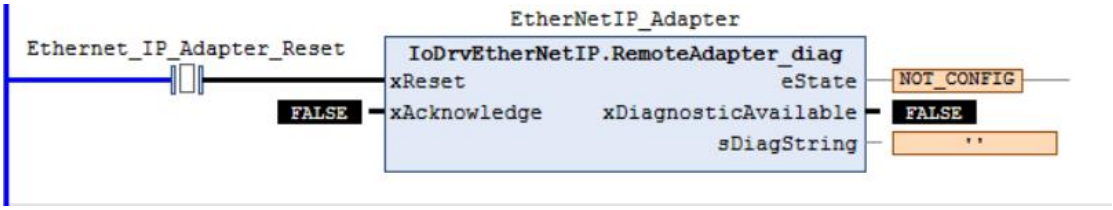
e																			
sDiagString	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	√

- **程序示例**

ST： 实例化名称为EtherNetIP_Adapter， eState为当前EtherNet/IP 做适配器时的状态。

```
EtherNetIP_Adapter(
    xReset:= Ethernet_IP_Adapter_Reset,
    xAcknowledge:= ,
    eState=>Ethernet_IP_Adapter_State ,
    xDiagnosticAvailable=> ,
    sDiagString=> );RETURN
```

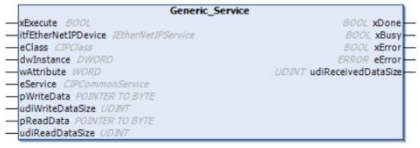
LD： 实例化名称为EtherNetIP_Adapter， eState为当前EtherNet/IP 做适配器时的状态。



5.4.4 Generic_Service

访问 EtherNetIP 适配器的 CIP 通用显示服务功能块

- **指令格式**

指令	名称	FB/FC	LD 表现	ST表现	库
Generic_Service	CIP通用服务	FB		Generic_Service(xExecute:= , xDone=> , xBusy=> , xError=> , itfEtherNetIPDevice:= , eClass:= , dwInstance:= , eError=> , wAttribute:= , eService:= ,	EtherNetIPService

				pWriteData:= , udiWriteDataSize:= , pReadData:= , udiReadDataSize:= , udiReceivedDataSize=>);	

- **相关变量**

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
xExecute	功能块使能	BOOL	TRUE-FALSE	-	若为TRUE（上升沿信号），则启动功能块运行
eClass	类	CIP Class	-	-	服务指向的对象类别，可以是 CIP 标准目标对象，或者第三方厂商自定义对象
itfEtherNetIPDevice	远程从站实例化名	IEtherNetIPService	-	-	远程从站实例化名
dwInstance	实例	DWORD	-	-	执行CIP时的实际实例数量，应大于等于1
wAttribute	属性	WORD	-	-	服务指向的属性
eService	公共服务	CIPCommonService	-	-	CIP通用公共服务
pWriteData	写入缓存	POINTER TO BYTE	-	-	指向属性数据的指针，将数据写入到目标适配器
udiWriteDataSize	写入数据长度	UDINT	1~65535	-	写入数据长度
pReadData	接收缓存	POINTER TO BYTE	-	-	指向属性数据的指针，从适配器接收的数据
udiReadDataSize	接收长度	UDINT	1~65535	-	从适配器接收的数据长度

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
xDone	完成	BOOL	TRUE-FALSE	-	客户端通信完成
xBusy	功能块运行中	BOOL	TRUE-FALSE	-	功能块运行中标志
xError	错误标志	BOOL	TRUE-FALSE	-	错误状态
eError	错误	ERROR	-	-	具体错误请参考 EtherNetIPService 库的枚举体 ERROR
udiReceivedDataSize	接收长度	UDINT	-	-	接收长度

	布尔	位串				整数								实数		时刻、持续时间 日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
xExecute	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
eClass	CIP Class																			
itfEtherNetIPDevice	IEtherNetIPService																			
dwInstance	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
wAttribute	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
eService	CIPCommonService																			
pWriteData	POINTER TO BYTE																			
udiWriteDataSize	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-
pReadData	POINTER TO BYTE																			
udiReadDataSize	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-

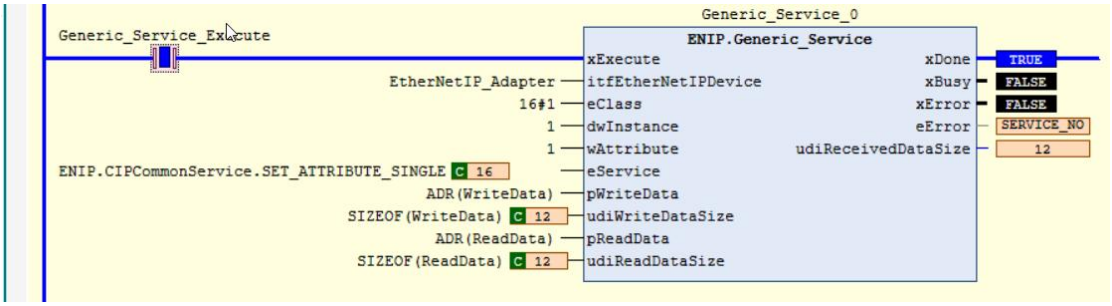
xDone	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
xBusy	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
xError	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
eError	ERROR																		
udiReceivedDataSize	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–

- **程序示例**

ST：可通过功能块Generic_Service，获取用户程序中的属性和服务数据。

```
Generic_Service_0(
    xExecute TRUE := Generic_Service_Execute TRUE,
    xDone TRUE => Generic_Service_Done TRUE,
    xBusy=> ,
    xError=> ,
    itfEtherNetIPDevice:=EtherNetIP_Adapter ,
    eClass IdentityOb :=16#1 ,
    dwInstance 1 := 1,
    eError=> ,
    wAttribute 1 :=1 ,
    eService SET_ATTRIB := ENIP.CIPCommonService.SET_ATTRIBUTE_SINGLE,
    pWriteData 16#B43AEE34 := ADR(WriteData),
    udiWriteDataSize 12 := SIZEOF(WriteData),
    pReadData 16#B43AEE40 :=ADR(ReadData) ,
    udiReadDataSize 12 :=SIZEOF(ReadData) ,
    udiReceivedDataSize=> );RETURN
```

LD：可通过功能块Generic_Service，获取用户程序中的属性和服务数据。



- **注意事项**

➢ itfEtherNetIPDevice为从站的实例化名称，eService为CIP 公用服务 ID 和名称（详见以下表格）。如果使用功能块中出现错误，eError将被赋相关错误值（枚举型ERROR里的各种值） 。详细请看以下错误代码。

CIP公用服务及属性

ID	属性
16#1	IdentityObject
16#2	MessageRouterObject
16#3	DeviceNetObject
16#4	AssemblyObject
16#5	ConnectionObject
16#6	ConnectionManagerObject
16#7	RegisterObject
16#8	DiscreteInputPointObject
16#9	DiscreteOutputPointObject
16#A	AnalogInputPointObject
16#B	AnalogOutputPointObject
16#E	PresenceSensingObject
16#F	ParameterObject
16#10	ParameterGroupObject
16#12	GroupObject
16#1D	DiscreteInputGroupObject
16#1E	DiscreteOutputGroupObject
16#1F	DiscreteGroupObject
16#20	AnalogInputGroupObject
16#21	AnalogOutputGroupObject
16#22	AnalogGroupObject
16#23	PositionSensorObject
16#24	PositionControllerSupervisorObject
16#25	PositionControllerObject
16#26	BlockSequencerObject
16#27	CommandBlockObject
16#28	MotorDataObject
16#29	ControlSupervisorObject
16#2A	ACDCDriveObject
16#2B	AcknowledgeHandlerObject
16#2C	OverloadObject

16#2D	SoftstartObject
16#2E	SelectionObject
16#30	S_DeviceSupervisorObject
16#31	S_AnalogSensorObject
16#32	S_AnalogActuatorObject
16#33	S_SingleStageControllerObject
16#34	S_GasCalibrationObject
16#35	TripPointObject
16#37	FileObject
16#38	S_PartialPressureObject
16#39	SafetySupervisorObject
16#3A	SafetyValidatorObject
16#3B	SafetyDiscreteOutputPointObject
16#3C	SafetyDiscreteOutputGroupObject
16#3D	SafetyDiscreteInputPointObject
16#3E	SafetyDiscreteInputGroupObject
16#3F	SafetyDualChannelOutputObject
16#40	S_SensorCalibrationObject
16#41	EventLogObject
16#42	MotionDeviceAxisObject
16#43	TimeSyncObject
16#44	ModbusObject
16#45	OriginatorConnectionListObject
16#46	ModbusSerialLinkObject
16#47	DeviceLevelRingObject
16#48	QoSObject
16#49	SafetyAnalogInputPointObject
16#4A	SafetyAnalogInputGroupObject
16#4B	SafetyDualChannelAnalogInputObject
16#4C	SERCOSIIILinkObject
16#4D	TargetConnectionListObject
16#4E	EnergyObject
16#4F	ElectricalEnergyObject

16#50	Non_ElectricalEnergyObject
16#51	BaseSwitchObject
16#52	SNMPObject
16#53	PowerManagementObject
16#F0	ControlNetObject
16#F1	ControlNetKeeperObject
16#F2	ControlNetSchedulingObject
16#F3	ConnectionConfigurationObject
16#F4	PortObject
16#F5	TCPIPInterfaceObject
16#F6	EthernetLinkObject
16#F7	CompoNetLink
16#F8	CompoNetRepeater

CIP及特定库错误

错误码	定义	描述
0	NO_ERROR	Service was successfully performed by the object specified.
16#1	CONNECTION_FAILURE	A connection related service failed along the connection path.
16#2	RESOURCE_UNAVAILABLE	Resources needed for the object to perform the requested service were unavailable
16#3	INVALID_PARAM_VALUE	See Status Code 0x20, which is the preferred value to use for this condition.
16#4	PATH_SEGMENT_ERROR	The path segment identifier or the segment syntax was not understood by the processing node. Path processing shall stop when a path segment error is encountered.
16#5	PATH_DESTINATION_UNKNOWN	The path is referencing an object class, instance or structure element that is not known or is not contained in the processing node. Path processing shall stop when a path destination unknown error is encountered.
16#6	PARTIAL_TRANSFER	Only part of the expected data was

		transferred.
16#7	CONNECTION_LOST	The messaging connection was lost.
16#8	SERVICE_NOT_SUPPORTED	The requested service was not implemented or was not defined for this Object Class/Instance.
16#9	INVALID_ATTRIBUTE_VALUE	Invalid attribute data detected.
16#A	ATTRIBUTE_LIST_ERROR	An attribute in the Get_Attribute_List or Set_Attribute_List response has a non-zero status.
16#B	ALREADY_IN_REQUEST_STATE	The object is already in the mode/state being requested by the service
16#C	OBJECT_STATE_ERROR	The object cannot perform the requested service in its current mode/state.
16#D	OBJECT_ALREADY_EXISTS	The requested instance of object to be created already exists.
16#E	ATTRIBUTE_NOT_SETTABLE	A request to modify a nonmodifiable attribute was received.
16#F	PRIVILEGE_VIOLATION	A permission/privilege check failed
16#10	DEVICE_STATE_ERROR	The device's current mode/state prohibits the execution of the requested service.
16#11	REPLY_DATA_TOO_LARGE	The data to be transmitted in the response buffer is larger than the allocated response buffer
16#12	FRAGMENTATION_OF_VALUE	The service specified an operation that is going to fragment a primitive data value, i. e. half a REAL data type.
16#13	NOT_ENOUGH_DATA	The service did not supply enough data to perform the specified operation.
16#14	ATTRIBUTE_NOT_SUPPORTED	The attribute specified in the request is not supported.
16#15	TOO_MUCH_DATA	The service supplied more data than was expected.
16#16	OBJECT_DOES_NOT_EXIST	The object specified does not exist in the device.
16#17	SERVICE_FRAGMENTATION_SEQUENCE_NOT_IN_PROGRESS	The fragmentation sequence for this service is not currently active for this data.
16#18	NO_STORED_ATTRIBUTE_DATA	The attribute data of this object was not saved prior to the requested service.

16#19	STORE_OPERATION_FAILURE	The attribute data of this object was not saved due to a failure during the attempt.
16#1A	ROUTING_FAILURE_REQUEST_PACKET_TOO_LARGE	The service request packet was too large for transmission on a CAA Net Base Services in the path to the destination. The routing device was forced to abort the service.
16#1B	ROUTING_FAILURE_RESPONSE_PACKET_TOO_LARGE	The service response packet was too large for transmission on a CAA Net Base Services in the path from the destination. The routing device was forced to abort the service.
16#1C	MISSING_ATTRIBUTE_LIST_ENTRY_DATA	The service did not supply an attribute in a list of attributes that was needed by the service to perform the requested behavior.
16#1D	INVALID_ATTRIBUTE_VALUE_LIST	The service is returning the list of attributes supplied with status information for those attributes that were invalid.
16#1E	EMBEDDED_SERVICE_ERROR	An embedded service resulted in an error.
16#1F	VENDOR_SPECIFIC_ERROR	A vendor specific error has been encountered. The Additional Code Field of the Error Response defines the particular error encountered. Use of this General Error Code should only be performed when none of the Error Codes presented in this table or within an Object Class definition accurately reflect the error.
16#20	INVALID_PARAMETER	A parameter associated with the request was invalid. This code is used when a parameter does not meet the requirements of this specification and/or the requirements defined in an Application Object Specification.
16#21	WRITE_ONEC_VALUE_OR_MEDIUM_ALREADY_WRITTEN	An attempt was made to write to a write-once medium (e. g. WORM drive, PROM) that has already been written, or to modify a value that cannot be changed once established.
16#22	INVALID_REPLY_RECEIVED	An invalid reply is received (e. g. reply

		service code does not match the request service code, or reply message is shorter than the minimum expected reply size). This status code can serve for other causes of invalid replies.
16#23	BUFFER_OVERFLOW	The message received is larger than the receiving buffer can handle. The entire message was discarded.
16#24	MESSAGE_FORMAT_ERROR	The format of the received message is not supported by the server.
16#25	KEY_FAILURE_IN_PATH	The Key Segment that was included as the first segment in the path does not match the destination module. The object specific status shall indicate which part of the key check failed.
16#26	PATH_SIZE_INVALID	The size of the path which was sent with the Service Request is either not large enough to allow the Request to be routed to an object or too much routing data was included.
16#27	UNEXPECTED_ATTRIBUTE_IN_LIST	An attempt was made to set an attribute that is not able to be set at this time.
16#28	INVALID_MEMBER_ID	The Member ID specified in the request does not exist in the specified Class/Instance/Attribute
16#29	MEMBER_NOT_SETTABLE	A request to modify a non-modifiable member was received
16#2A	GROUP_2_ONLY_SERVER_GENERAL_FAILURE	This error code may only be reported by DeviceNet Group 2 Only servers with 4K or less code space and only in place of Service not supported, Attribute not supported and Attribute not settable.
16#2B	UNKNOWN_MODBUS_ERROR	A CIP to Modbus translator received an unknown Modbus Exception Code.
16#2C	ATTRIBUTE_NOT_GETTABLE	A request to read a non-readable attribute was received
16#2D	INSTANCE_NOT_DELETABLE	The requested object instance cannot be deleted
16#2E	SERVICE_NOT_SUPPORTED_FOR_SPECIFIED_PATH	The object supports the service, but not for the designated application path (e. g. attribute). NOTE: Not to be used for any set service (use General

		Status Code 0x0E or 0x29 instead)
16#100	TIME_OUT	Request timed out.
	INTERFACE_MISSING	IEtherNetIPService is not implemented.
	REMOTE_CALL_FAILED	No physical connection.
	NULL_POINTER	Wrong input value NULL.
	INVALID_DATA_SIZE	Size of data unacceptable.
	WRONG_INTERFACE_VERSION	Version missmatch. Device implements not the same version of interface for the called method.
	NO_MEMORY	Not enough memory
	UNKNOWN_ERROR	An unknown error occured.
	ABORTED	Service was aborted

5.4.5 Get_Attributes_All

获取对象实例的属性和服务数据

- **指令格式**

指令	名称	FB/FC	LD 表现	ST表现	库
Get_Attributes_All	获取全部属性	FB		Get_Attributes_All(xExecute:= , xDone=> , xBusy=> , xError=> , itfEtherNetIPDevice:= , eClass:= , dwInstance:= , eError=> , pData:= , udiDataSize:= , udiReceivedDataSize=>);	EtherNetIP Services

- **相关变量**

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
xExecute	功能块使能	BOOL	TRUE-FALSE	-	若为TRUE（上升沿信号），则启动功能块运行
itfEtherNetIPDevice	远程从站实例化名	IEtherNetIPService	-	-	远程从站实例化名
eClass	类	CIP Class	-	-	服务指向的对象类别，可以是 CIP 标准目标对象，或者第三方厂商自定义对象
dwInstance	实例	DWORD	-	-	执行CIP时的实际实例数量，应大于等于1
pData	接收缓存	POINTER TO BYTE	-	-	指向属性数据的指针，从目标从站接收到数据。
udiDataSize	接收长度	UDINT	-	-	接收到数据长度

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
xDone	完成	BOOL	TRUE-FALSE	-	客户端通信完成
xBusy	功能块运行中	BOOL	TRUE-FALSE	-	功能块运行中标志
xError	错误标志	BOOL	TRUE-FALSE	-	错误状态
eError	错误	ERROR	-	-	具体错误请参考EtherNetIPService库的枚举体ERROR
udiReceivedDataSize	接收长度	UDINT	-	-	接收长度

	布尔	位串				整数								实数		时刻、持续时间 日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
xExecu	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

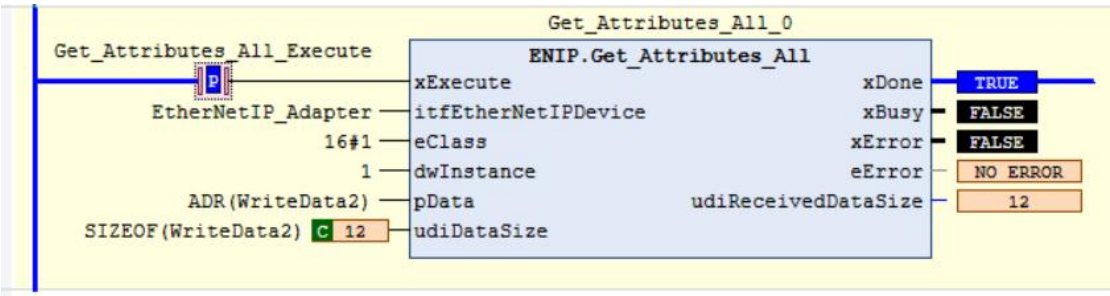
te																			
itfEtherNetIPDevice	IEtherNetIPService																		
eClass	CIP Class																		
dwInstance	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
pData	POINTER TO BYTE																		
udiDataSize	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–
xDone	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
xBusy	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
xError	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
eError	ERROR																		
udiReceivedDataSize	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–

- **程序示例**

ST：可通过功能块 Get_Attributes_All，获取对象实例的属性和服务数据。

```
Get_Attributes_All_0(
    xExecute TRUE :=Get_Attributes_All_Execute TRUE ,
    xDone=> ,
    xBusy=> ,
    xError=> ,
    itfEtherNetIPDevice:= EtherNetIP_Adapter,
    eClass IdentityOb :=16#1 ,
    dwInstance 1 := 1,
    eError=> ,
    pData 16#B43C2014 := ADR(WriteData2),
    udiDataSize 12 := SIZEOF(WriteData2),
    udiReceivedDataSize=> );
```

LD:可通过功能块Get_Attributes_All，获取对象实例的属性和服务数据。



- **注意事项**

- itfEtherNetIPDevice为从站的实例化名称，eClass服务指向的对象类别。如果使用功能块中出现错误，eError将被赋相关错误值（枚举型ERROR里的各种值） 。详细请看以下错误代码。

CIP 公用服务及属性

ID	属性
16#1	IdentityObject
16#2	MessageRouterObject
16#3	DeviceNetObject
16#4	AssemblyObject
16#5	ConnectionObject
16#6	ConnectionManagerObject
16#7	RegisterObject
16#8	DiscreteInputPointObject
16#9	DiscreteOutputPointObject
16#A	AnalogInputPointObject
16#B	AnalogOutputPointObject
16#E	PresenceSensingObject
16#F	ParameterObject
16#10	ParameterGroupObject
16#12	GroupObject
16#1D	DiscreteInputGroupObject
16#1E	DiscreteOutputGroupObject
16#1F	DiscreteGroupObject
16#20	AnalogInputGroupObject
16#21	AnalogOutputGroupObject
16#22	AnalogGroupObject
16#23	PositionSensorObject
16#24	PositionControllerSupervisorObject
16#25	PositionControllerObject
16#26	BlockSequencerObject
16#27	CommandBlockObject
16#28	MotorDataObject

16#29	ControlSupervisorObject
16#2A	ACDCDriveObject
16#2B	AcknowledgeHandlerObject
16#2C	OverloadObject
16#2D	SoftstartObject
16#2E	SelectionObject
16#30	S_DeviceSupervisorObject
16#31	S_AnalogSensorObject
16#32	S_AnalogActuatorObject
16#33	S_SingleStageControllerObject
16#34	S_GasCalibrationObject
16#35	TripPointObject
16#37	FileObject
16#38	S_PartialPressureObject
16#39	SafetySupervisorObject
16#3A	SafetyValidatorObject
16#3B	SafetyDiscreteOutputPointObject
16#3C	SafetyDiscreteOutputGroupObject
16#3D	SafetyDiscreteInputPointObject
16#3E	SafetyDiscreteInputGroupObject
16#3F	SafetyDualChannelOutputObject
16#40	S_SensorCalibrationObject
16#41	EventLogObject
16#42	MotionDeviceAxisObject
16#43	TimeSyncObject
16#44	ModbusObject
16#45	OriginatorConnectionListObject
16#46	ModbusSerialLinkObject
16#47	DeviceLevelRingObject
16#48	QoSObject
16#49	SafetyAnalogInputPointObject
16#4A	SafetyAnalogInputGroupObject
16#4B	SafetyDualChannelAnalogInputObject

16#4C	SERCOSIIILinkObject
16#4D	TargetConnectionListObject
16#4E	EnergyObject
16#4F	ElectricalEnergyObject
16#50	Non_ElectricalEnergyObject
16#51	BaseSwitchObject
16#52	SNMPObject
16#53	PowerManagementObject
16#F0	ControlNetObject
16#F1	ControlNetKeeperObject
16#F2	ControlNetSchedulingObject
16#F3	ConnectionConfigurationObject
16#F4	PortObject
16#F5	TCPIPInterfaceObject
16#F6	EthernetLinkObject
16#F7	CompoNetLink
16#F8	CompoNetRepeater

CIP及特定库错误

错误码	定义	描述
0	NO_ERROR	Service was successfully performed by the object specified.
16#1	CONNECTION_FAILURE	A connection related service failed along the connection path.
16#2	RESOURCE_UNAVAILABLE	Resources needed for the object to perform the requested service were unavailable
16#3	INVALID_PARAM_VALUE	See Status Code 0x20, which is the preferred value to use for this condition.
16#4	PATH_SEGMENT_ERROR	The path segment identifier or the segment syntax was not understood by the processing node. Path processing shall stop when a path segment error is encountered.
16#5	PATH_DESTINATION_UNKNOWN	The path is referencing an object class, instance or structure element that is not known or is not contained in the

		processing node. Path processing shall stop when a path destination unknown error is encountered.
16#6	PARTIAL_TRANSFER	Only part of the expected data was transferred.
16#7	CONNECTION_LOST	The messaging connection was lost.
16#8	SERVICE_NOT_SUPPORTED	The requested service was not implemented or was not defined for this Object Class/Instance.
16#9	INVALID_ATTRIBUTE_VALUE	Invalid attribute data detected.
16#A	ATTRIBUTE_LIST_ERROR	An attribute in the Get_Attribute_List or Set_Attribute_List response has a non-zero status.
16#B	ALREADY_IN_REQUEST_STATE	The object is already in the mode/state being requested by the service
16#C	OBJECT_STATE_ERROR	The object cannot perform the requested service in its current mode/state.
16#D	OBJECT_ALREADY_EXISTS	The requested instance of object to be created already exists.
16#E	ATTRIBUTE_NOT_SETTABLE	A request to modify a nonmodifiable attribute was received.
16#F	PRIVILEGE_VIOLATION	A permission/privilege check failed
16#10	DEVICE_STATE_ERROR	The device's current mode/state prohibits the execution of the requested service.
16#11	REPLY_DATA_TOO_LARGE	The data to be transmitted in the response buffer is larger than the allocated response buffer
16#12	FRAGMENTATION_OF_VALUE	The service specified an operation that is going to fragment a primitive data value, i. e. half a REAL data type.
16#13	NOT_ENOUGH_DATA	The service did not supply enough data to perform the specified operation.
16#14	ATTRIBUTE_NOT_SUPPORTED	The attribute specified in the request is not supported.
16#15	TOO_MUCH_DATA	The service supplied more data than was expected.
16#16	OBJECT_DOES_NOT_EXIST	The object specified does not exist in the device.

16#17	SERVICE_FRAGMENTATION_SEQUENCE_NOT_IN_PROGRESS	The fragmentation sequence for this service is not currently active for this data.
16#18	NO_STORED_ATTRIBUTE_DATA	The attribute data of this object was not saved prior to the requested service.
16#19	STORE_OPERATION_FAILURE	The attribute data of this object was not saved due to a failure during the attempt.
16#1A	ROUTING_FAILURE_REQUEST_PACKET_TOO_LARGE	The service request packet was too large for transmission on a CAA Net Base Services in the path to the destination. The routing device was forced to abort the service.
16#1B	ROUTING_FAILURE_RESPONSE_PACKET_TOO_LARGE	The service response packet was too large for transmission on a CAA Net Base Services in the path from the destination. The routing device was forced to abort the service.
16#1C	MISSING_ATTRIBUTE_LIST_ENTRY_DATA	The service did not supply an attribute in a list of attributes that was needed by the service to perform the requested behavior.
16#1D	INVALID_ATTRIBUTE_VALUE_LIST	The service is returning the list of attributes supplied with status information for those attributes that were invalid.
16#1E	EMBEDDED_SERVICE_ERROR	An embedded service resulted in an error.
16#1F	VENDOR_SPECIFIC_ERROR	A vendor specific error has been encountered. The Additional Code Field of the Error Response defines the particular error encountered. Use of this General Error Code should only be performed when none of the Error Codes presented in this table or within an Object Class definition accurately reflect the error.
16#20	INVALID_PARAMETER	A parameter associated with the request was invalid. This code is used when a parameter does not meet the requirements of this specification and/or the requirements defined in an Application Object Specification.

16#21	WRITE_ONEC_VALUE_OR_MEDIUM_ALREADY_WRITTEN	An attempt was made to write to a write-once medium (e. g. WORM drive, PROM) that has already been written, or to modify a value that cannot be changed once established.
16#22	INVALID_REPLY_RECEIVED	An invalid reply is received (e. g. reply service code does not match the request service code, or reply message is shorter than the minimum expected reply size). This status code can serve for other causes of invalid replies.
16#23	BUFFER_OVERFLOW	The message received is larger than the receiving buffer can handle. The entire message was discarded.
16#24	MESSAGE_FORMAT_ERROR	The format of the received message is not supported by the server.
16#25	KEY_FAILURE_IN_PATH	The Key Segment that was included as the first segment in the path does not match the destination module. The object specific status shall indicate which part of the key check failed.
16#26	PATH_SIZE_INVALID	The size of the path which was sent with the Service Request is either not large enough to allow the Request to be routed to an object or too much routing data was included.
16#27	UNEXPECTED_ATTRIBUTE_IN_LIST	An attempt was made to set an attribute that is not able to be set at this time.
16#28	INVALID_MEMBER_ID	The Member ID specified in the request does not exist in the specified Class/Instance/Attribute
16#29	MEMBER_NOT_SETTABLE	A request to modify a non-modifiable member was received
16#2A	GROUP_2_ONLY_SERVER_GENERAL_FAILURE	This error code may only be reported by DeviceNet Group 2 Only servers with 4K or less code space and only in place of Service not supported, Attribute not supported and Attribute not settable.
16#2B	UNKNOWN_MODBUS_ERROR	A CIP to Modbus translator received an unknown Modbus Exception Code.
16#2C	ATTRIBUTE_NOT_GETTABLE	A request to read a non-readable attribute was received

16#2D	INSTANCE_NOT_DELETABLE	The requested object instance cannot be deleted
16#2E	SERVICE_NOT_SUPPORTED_FOR_SPECIFIED_PATH	The object supports the service, but not for the designated application path (e. g. attribute). NOTE: Not to be used for any set service (use General Status Code 0x0E or 0x29 instead)
16#100	TIME_OUT	Request timed out.
	INTERFACE_MISSING	IEtherNetIPService is not implemented.
	REMOTE_CALL_FAILED	No physical connection.
	NULL_POINTER	Wrong input value NULL.
	INVALID_DATA_SIZE	Size of data unacceptable.
	WRONG_INTERFACE_VERSION	Version missmatch. Device implements not the same version of interface for the called method.
	NO_MEMORY	Not enough memory
	UNKNOWN_ERROR	An unknown error occured.
	ABORTED	Service was aborted

5.4.6 Get_Attribute_Single

获取对象实例的属性和服务数据。

- **指令格式**

指令	名称	FB/FC	LD 表现	ST表现	库
Get_Attributes_Single	获取单个属性	FB		Get_Attribute_Single(xExecute:=, xDone=>, xBusy=>, xError=>, itfEtherNetIPDevice:=, eClass:=, dwInstance:=, eError=>, wAttribute:=, pData:=,	EtherNetIP Services

				udiDataSize:= , udiReceive dDataSize=>);	

- 相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
xExecute	功能块使能	BOOL	TRUE-FALSE	-	若为TRUE（上升沿信号），则启动功能块运行
itfEtherNetIP Device	远程从站实例化名	IEtherNetIPS ervice	-	-	远程从站实例化名
eClass	类	CIP Class	-	-	服务指向的对象类别，可以是 CIP 标准目标对象，或者第三方厂商自定义对象
dwInstance	实例	DWORD	-	-	执行CIP时的实际实例数量，应大于等于1
wAttribute	属性	WORD	-	-	服务指向的属性
pData	接收缓存	POINTER TO BYTE	-	-	指向属性数据的指针，从目标从站接收到数据
udiDataSize	接收长度	UDINT	-	-	接收到数据长度

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
xDone	完成	BOOL	TRUE-FALSE	-	客户端通信完成
xBusy	功能块运行中	BOOL	TRUE-FALSE	-	功能块运行中标志
xError	错误标志	BOOL	TRUE-FALSE	-	错误状态
eError	错误	ERROR	-	-	具体错误请参考EtherNetIPService库的枚举体ERROR

udiReceivedDataSize	接收长度	UDINT	–	–	接收长度

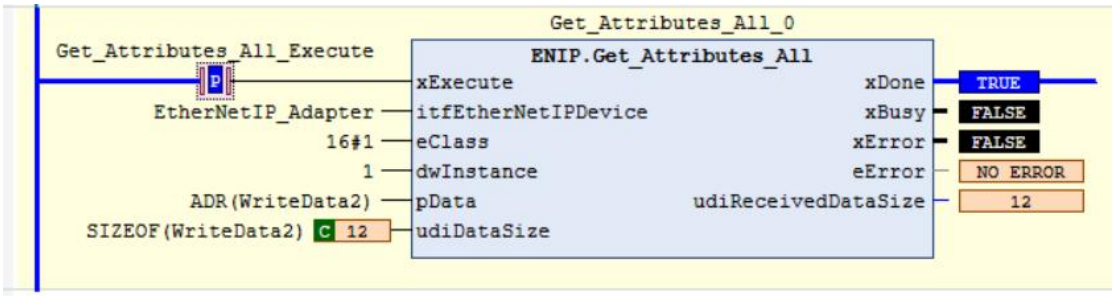
	布尔	位串				整数								实数		时刻、持续时间 日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
xExecute	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
itfEtherNetIPDevice	IEtherNetIPService																			
eClass	CIP Class																			
dwInstance	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
wAttribute	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
pData	POINTER TO BYTE																			
udiDataSize	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–
xDone	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
xBusy	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
xError	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
eError	ERROR																			
udiReceivedDataSize	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–

- **程序示例**

ST： 可通过功能块 Get_Attributes_Single，获取对象实例的单个属性。

```
Get_Attribute_Single_0(
    xExecute TRUE := Get_Attribute_Single_Execute TRUE,
    xDone=> ,
    xBusy=> ,
    xError=> ,
    itfEtherNetIPDevice:=EtherNetIP_Adapter ,
    eClass IdentityOb :=16#1 ,
    dwInstance 1 :=1 ,
    eError=> ,
    wAttribute 16 :=ENIP.CIPCommonService.SET_ATTRIBUTE_SINGLE ,
    pData 16#B43ABE4C := ADR(WriteData3),
    udiDataSize 12 :=SIZEOF(WriteData3) ,
    udiReceivedDataSize=> );
```

LD：可通过功能块 Get_Attributes_Single，获取对象实例的单个属性。



- **注意事项**

➢ itfEtherNetIPDevice为从站的实例化名称，eClass服务指向的对象类别。如果使用功能块中出现错误，eError将被赋相关错误值（枚举型ERROR里的各种值） 。详细请看以下错误代码。

CIP公用服务及属性

ID	属性
16#1	IdentityObject
16#2	MessageRouterObject
16#3	DeviceNetObject
16#4	AssemblyObject
16#5	ConnectionObject
16#6	ConnectionManagerObject
16#7	RegisterObject
16#8	DiscreteInputPointObject

16#9	DiscreteOutputPointObject
16#A	AnalogInputPointObject
16#B	AnalogOutputPointObject
16#E	PresenceSensingObject
16#F	ParameterObject
16#10	ParameterGroupObject
16#12	GroupObject
16#1D	DiscreteInputGroupObject
16#1E	DiscreteOutputGroupObject
16#1F	DiscreteGroupObject
16#20	AnalogInputGroupObject
16#21	AnalogOutputGroupObject
16#22	AnalogGroupObject
16#23	PositionSensorObject
16#24	PositionControllerSupervisorObject
16#25	PositionControllerObject
16#26	BlockSequencerObject
16#27	CommandBlockObject
16#28	MotorDataObject
16#29	ControlSupervisorObject
16#2A	ACDCDriveObject
16#2B	AcknowledgeHandlerObject
16#2C	OverloadObject
16#2D	SoftstartObject
16#2E	SelectionObject
16#30	S_DeviceSupervisorObject
16#31	S_AnalogSensorObject
16#32	S_AnalogActuatorObject
16#33	S_SingleStageControllerObject
16#34	S_GasCalibrationObject
16#35	TripPointObject
16#37	FileObject
16#38	S_PartialPressureObject

16#39	SafetySupervisorObject
16#3A	SafetyValidatorObject
16#3B	SafetyDiscreteOutputPointObject
16#3C	SafetyDiscreteOutputGroupObject
16#3D	SafetyDiscreteInputPointObject
16#3E	SafetyDiscreteInputGroupObject
16#3F	SafetyDualChannelOutputObject
16#40	S_SensorCalibrationObject
16#41	EventLogObject
16#42	MotionDeviceAxisObject
16#43	TimeSyncObject
16#44	ModbusObject
16#45	OriginatorConnectionListObject
16#46	ModbusSerialLinkObject
16#47	DeviceLevelRingObject
16#48	QoSObject
16#49	SafetyAnalogInputPointObject
16#4A	SafetyAnalogInputGroupObject
16#4B	SafetyDualChannelAnalogInputObject
16#4C	SERCOSIIILinkObject
16#4D	TargetConnectionListObject
16#4E	EnergyObject
16#4F	ElectricalEnergyObject
16#50	Non_ElectricalEnergyObject
16#51	BaseSwitchObject
16#52	SNMPObject
16#53	PowerManagementObject
16#F0	ControlNetObject
16#F1	ControlNetKeeperObject
16#F2	ControlNetSchedulingObject
16#F3	ConnectionConfigurationObject
16#F4	PortObject
16#F5	TCPIPInterfaceObject

16#F6	EthernetLinkObject
16#F7	CompoNetLink
16#F8	CompoNetRepeater

CIP及特定库错误

错误码	定义	描述
0	NO_ERROR	Service was successfully performed by the object specified.
16#1	CONNECTION_FAILURE	A connection related service failed along the connection path.
16#2	RESOURCE_UNAVAILABLE	Resources needed for the object to perform the requested service were unavailable
16#3	INVALID_PARAM_VALUE	See Status Code 0x20, which is the preferred value to use for this condition.
16#4	PATH_SEGMENT_ERROR	The path segment identifier or the segment syntax was not understood by the processing node. Path processing shall stop when a path segment error is encountered.
16#5	PATH_DESTINATION_UNKNOWN	The path is referencing an object class, instance or structure element that is not known or is not contained in the processing node. Path processing shall stop when a path destination unknown error is encountered.
16#6	PARTIAL_TRANSFER	Only part of the expected data was transferred.
16#7	CONNECTION_LOST	The messaging connection was lost.
16#8	SERVICE_NOT_SUPPORTED	The requested service was not implemented or was not defined for this Object Class/Instance.
16#9	INVALID_ATTRIBUTE_VALUE	Invalid attribute data detected.
16#A	ATTRIBUTE_LIST_ERROR	An attribute in the Get_Attribute_List or Set_Attribute_List response has a non-zero status.
16#B	ALREADY_IN_REQUEST_STATE	The object is already in the mode/state being requested by the service
16#C	OBJECT_STATE_ERROR	The object cannot perform the requested service in its current mode/state.

16#D	OBJECT_ALREADY_EXISTS	The requested instance of object to be created already exists.
16#E	ATTRIBUTE_NOT_SETTABLE	A request to modify a nonmodifiable attribute was received.
16#F	PRIVILEGE_VIOLATION	A permission/privilege check failed
16#10	DEVICE_STATE_ERROR	The device's current mode/state prohibits the execution of the requested service.
16#11	REPLY_DATA_TOO_LARGE	The data to be transmitted in the response buffer is larger than the allocated response buffer
16#12	FRAGMENTATION_OF_VALUE	The service specified an operation that is going to fragment a primitive data value, i. e. half a REAL data type.
16#13	NOT_ENOUGH_DATA	The service did not supply enough data to perform the specified operation.
16#14	ATTRIBUTE_NOT_SUPPORTED	The attribute specified in the request is not supported.
16#15	TOO_MUCH_DATA	The service supplied more data than was expected.
16#16	OBJECT_DOES_NOT_EXIST	The object specified does not exist in the device.
16#17	SERVICE_FRAGMENTATION_SEQUENCE_NOT_IN_PROGRESS	The fragmentation sequence for this service is not currently active for this data.
16#18	NO_STORED_ATTRIBUTE_DATA	The attribute data of this object was not saved prior to the requested service.
16#19	STORE_OPERATION_FAILURE	The attribute data of this object was not saved due to a failure during the attempt.
16#1A	ROUTING_FAILURE_REQUEST_PACKET_TOO_LARGE	The service request packet was too large for transmission on a CAA Net Base Services in the path to the destination. The routing device was forced to abort the service.
16#1B	ROUTING_FAILURE_RESPONSE_PACKET_TOO_LARGE	The service response packet was too large for transmission on a CAA Net Base Services in the path from the destination. The routing device was forced to abort the service.

16#1C	MISSING_ATTRIBUTE_LIST_ENTRY_DATA	The service did not supply an attribute in a list of attributes that was needed by the service to perform the requested behavior.
16#1D	INVALID_ATTRIBUTE_VALUE_LIST	The service is returning the list of attributes supplied with status information for those attributes that were invalid.
16#1E	EMBEDDED_SERVICE_ERROR	An embedded service resulted in an error.
16#1F	VENDOR_SPECIFIC_ERROR	A vendor specific error has been encountered. The Additional Code Field of the Error Response defines the particular error encountered. Use of this General Error Code should only be performed when none of the Error Codes presented in this table or within an Object Class definition accurately reflect the error.
16#20	INVALID_PARAMETER	A parameter associated with the request was invalid. This code is used when a parameter does not meet the requirements of this specification and/or the requirements defined in an Application Object Specification.
16#21	WRITE_ONEC_VALUE_OR_MEDIUM_ALREADY_WRITTEN	An attempt was made to write to a write-once medium (e. g. WORM drive, PROM) that has already been written, or to modify a value that cannot be changed once established.
16#22	INVALID_REPLY_RECEIVED	An invalid reply is received (e. g. reply service code does not match the request service code, or reply message is shorter than the minimum expected reply size). This status code can serve for other causes of invalid replies.
16#23	BUFFER_OVERFLOW	The message received is larger than the receiving buffer can handle. The entire message was discarded.
16#24	MESSAGE_FORMAT_ERROR	The format of the received message is not supported by the server.
16#25	KEY_FAILURE_IN_PATH	The Key Segment that was included as the first segment in the path does not

		match the destination module. The object specific status shall indicate which part of the key check failed.
16#26	PATH_SIZE_INVALID	The size of the path which was sent with the Service Request is either not large enough to allow the Request to be routed to an object or too much routing data was included.
16#27	UNEXPECTED_ATTRIBUTE_IN_LIST	An attempt was made to set an attribute that is not able to be set at this time.
16#28	INVALID_MEMBER_ID	The Member ID specified in the request does not exist in the specified Class/Instance/Attribute
16#29	MEMBER_NOT_SETTABLE	A request to modify a non-modifiable member was received
16#2A	GROUP_2_ONLY_SERVER_GENERAL_FAILURE	This error code may only be reported by DeviceNet Group 2 Only servers with 4K or less code space and only in place of Service not supported, Attribute not supported and Attribute not settable.
16#2B	UNKNOWN_MODBUS_ERROR	A CIP to Modbus translator received an unknown Modbus Exception Code.
16#2C	ATTRIBUTE_NOT_GETTABLE	A request to read a non-readable attribute was received
16#2D	INSTANCE_NOT_DELETABLE	The requested object instance cannot be deleted
16#2E	SERVICE_NOT_SUPPORTED_FOR_SPECIFIED_PATH	The object supports the service, but not for the designated application path (e. g. attribute). NOTE: Not to be used for any set service (use General Status Code 0x0E or 0x29 instead)
16#100	TIME_OUT	Request timed out.
	INTERFACE_MISSING	IEtherNetIPService is not implemented.
	REMOTE_CALL_FAILED	No physical connection.
	NULL_POINTER	Wrong input value NULL.
	INVALID_DATA_SIZE	Size of data unacceptable.
	WRONG_INTERFACE_VERSION	Version missmatch. Device implements not the same version of interface for the called method.
	NO_MEMORY	Not enough memory

	UNKNOWN_ERROR	An unknown error occured.
	ABORTED	Service was aborted

5.4.7 Set_Attributes_All

设置对象实例的属性和服务数据

- **指令格式**

指令	名称	FB/FC	LD 表现	ST表现	库
Set_Attributes_All	设置对象实例所有属性	FB		Set_Attributes_All(xExecute:= , xDone=> , xBusy=> , xError=> , itfEtherNetIPDevice:= , eClass:= , dwInstance:= , eError=> , pData:= , udiDataSize:=);	EtherNetIP Services

- **相关变量**

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
xExecute	功能块使能	BOOL	TRUE-FALSE	-	若为TRUE（上升沿信号），则启动功能块运行
itfEtherNetIPDevice	远程从站实例化名	IEtherNetIPService	-	-	远程从站实例化名
eClass	类	CIP Class	-	-	服务指向的对象类别，可以是 CIP 标准目标对象，或者第三方厂商自定义对象
dwInstance	实例	DWORD	-	-	执行CIP时的实际

					实例数量，应大于等于1
pData	接收缓存	POINTER TO BYTE	-	-	指向属性数据的指针，发送该数据到目标从站
udiDataSize	接收长度	UDINT	-	-	发送数据长度

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
xDone	完成	BOOL	TRUE-FALSE	-	客户端通信完成
xBusy	功能块运行中	BOOL	TRUE-FALSE	-	功能块运行中标志
xError	错误标志	BOOL	TRUE-FALSE	-	错误状态
eError	错误	ERROR	-	-	具体错误请参考 EtherNetIPService 库的枚举体 ERROR

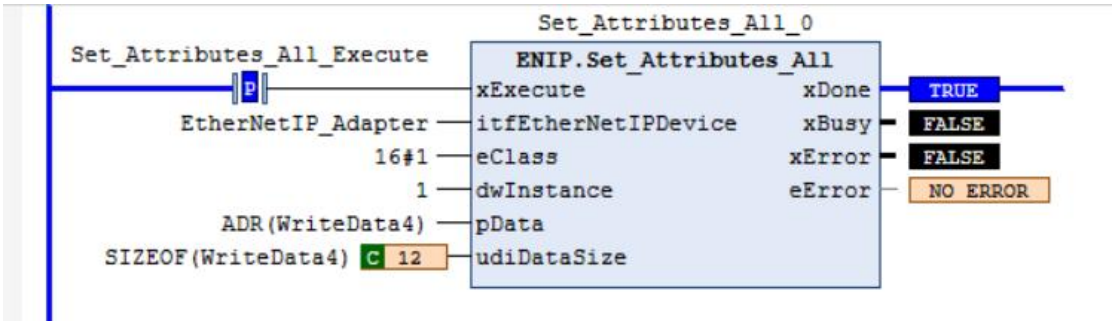
	布尔	位串				整数								实数		时刻、持续时间 日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
xExecute	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
itfEtherNetIPDevice	IEtherNetIPService																			
eClass	CIP Class																			
dwInstance	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
pData	POINTER TO BYTE																			
udiDataSize	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-
xDone	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
xBusy	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
xError	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
eError	ERROR																			

- **程序示例**

ST： 可通过功能块Set_Attributes_All，设置对象实例的全部属性和服务数据。

```
Set_Attributes_All_0(
    xExecute TRUE :=Set_Attributes_All_Execute TRUE ,
    xDone=> ,
    xBusy=> ,
    xError=> ,
    itfEtherNetIPDevice:=EtherNetIP_Adapter ,
    eClass IdentityOb :=16#1 ,
    dwInstance 1 := 1,
    eError=> ,
    pData 16#B43F0008 := ADR(WriteData4),
    udiDataSize 12 := SIZEOF(WriteData4));
```

LD： 可通过功能块Set_Attributes_All，设置对象实例的全部属性和服务数据。



- **注意事项**

➢ itfEtherNetIPDevice为从站的实例化名称，eClass服务指向的对象类别。如果使用功能块中出现错误，eError将被赋相关错误值（枚举型ERROR里的各种值） 。详细请看以下错误代码。

CIP 公用服务及属性

ID	属性
16#1	IdentityObject
16#2	MessageRouterObject
16#3	DeviceNetObject
16#4	AssemblyObject
16#5	ConnectionObject
16#6	ConnectionManagerObject
16#7	RegisterObject

16#8	DiscreteInputPointObject
16#9	DiscreteOutputPointObject
16#A	AnalogInputPointObject
16#B	AnalogOutputPointObject
16#E	PresenceSensingObject
16#F	ParameterObject
16#10	ParameterGroupObject
16#12	GroupObject
16#1D	DiscreteInputGroupObject
16#1E	DiscreteOutputGroupObject
16#1F	DiscreteGroupObject
16#20	AnalogInputGroupObject
16#21	AnalogOutputGroupObject
16#22	AnalogGroupObject
16#23	PositionSensorObject
16#24	PositionControllerSupervisorObject
16#25	PositionControllerObject
16#26	BlockSequencerObject
16#27	CommandBlockObject
16#28	MotorDataObject
16#29	ControlSupervisorObject
16#2A	ACDCDriveObject
16#2B	AcknowledgeHandlerObject
16#2C	OverloadObject
16#2D	SoftstartObject
16#2E	SelectionObject
16#30	S_DeviceSupervisorObject
16#31	S_AnalogSensorObject
16#32	S_AnalogActuatorObject
16#33	S_SingleStageControllerObject
16#34	S_GasCalibrationObject
16#35	TripPointObject
16#37	FileObject

16#38	S_PartialPressureObject
16#39	SafetySupervisorObject
16#3A	SafetyValidatorObject
16#3B	SafetyDiscreteOutputPointObject
16#3C	SafetyDiscreteOutputGroupObject
16#3D	SafetyDiscreteInputPointObject
16#3E	SafetyDiscreteInputGroupObject
16#3F	SafetyDualChannelOutputObject
16#40	S_SensorCalibrationObject
16#41	EventLogObject
16#42	MotionDeviceAxisObject
16#43	TimeSyncObject
16#44	ModbusObject
16#45	OriginatorConnectionListObject
16#46	ModbusSerialLinkObject
16#47	DeviceLevelRingObject
16#48	QoSObject
16#49	SafetyAnalogInputPointObject
16#4A	SafetyAnalogInputGroupObject
16#4B	SafetyDualChannelAnalogInputObject
16#4C	SERCOSIIILinkObject
16#4D	TargetConnectionListObject
16#4E	EnergyObject
16#4F	ElectricalEnergyObject
16#50	Non_ElectricalEnergyObject
16#51	BaseSwitchObject
16#52	SNMPObject
16#53	PowerManagementObject
16#F0	ControlNetObject
16#F1	ControlNetKeeperObject
16#F2	ControlNetSchedulingObject
16#F3	ConnectionConfigurationObject
16#F4	PortObject

16#F5	TCPIPInterfaceObject
16#F6	EthernetLinkObject
16#F7	CompoNetLink
16#F8	CompoNetRepeater

CIP 及特定库错误

错误码	定义	描述
0	NO_ERROR	Service was successfully performed by the object specified.
16#1	CONNECTION_FAILURE	A connection related service failed along the connection path.
16#2	RESOURCE_UNAVAILABLE	Resources needed for the object to perform the requested service were unavailable
16#3	INVALID_PARAM_VALUE	See Status Code 0x20, which is the preferred value to use for this condition.
16#4	PATH_SEGMENT_ERROR	The path segment identifier or the segment syntax was not understood by the processing node. Path processing shall stop when a path segment error is encountered.
16#5	PATH_DESTINATION_UNKNOWN	The path is referencing an object class, instance or structure element that is not known or is not contained in the processing node. Path processing shall stop when a path destination unknown error is encountered.
16#6	PARTIAL_TRANSFER	Only part of the expected data was transferred.
16#7	CONNECTION_LOST	The messaging connection was lost.
16#8	SERVICE_NOT_SUPPORTED	The requested service was not implemented or was not defined for this Object Class/Instance.
16#9	INVALID_ATTRIBUTE_VALUE	Invalid attribute data detected.
16#A	ATTRIBUTE_LIST_ERROR	An attribute in the Get_Attribute_List or Set_Attribute_List response has a non-zero status.
16#B	ALREADY_IN_REQUEST_STATE	The object is already in the mode/state being requested by the service

16#C	OBJECT_STATE_ERROR	The object cannot perform the requested service in its current mode/state.
16#D	OBJECT_ALREADY_EXISTS	The requested instance of object to be created already exists.
16#E	ATTRIBUTE_NOT_SETTABLE	A request to modify a nonmodifiable attribute was received.
16#F	PRIVILEGE_VIOLATION	A permission/privilege check failed
16#10	DEVICE_STATE_ERROR	The device's current mode/state prohibits the execution of the requested service.
16#11	REPLY_DATA_TOO_LARGE	The data to be transmitted in the response buffer is larger than the allocated response buffer
16#12	FRAGMENTATION_OF_VALUE	The service specified an operation that is going to fragment a primitive data value, i. e. half a REAL data type.
16#13	NOT_ENOUGH_DATA	The service did not supply enough data to perform the specified operation.
16#14	ATTRIBUTE_NOT_SUPPORTED	The attribute specified in the request is not supported.
16#15	TOO_MUCH_DATA	The service supplied more data than was expected.
16#16	OBJECT_DOES_NOT_EXIST	The object specified does not exist in the device.
16#17	SERVICE_FRAGMENTATION_SEQUENCE_NOT_IN_PROGRESS	The fragmentation sequence for this service is not currently active for this data.
16#18	NO_STORED_ATTRIBUTE_DATA	The attribute data of this object was not saved prior to the requested service.
16#19	STORE_OPERATION_FAILURE	The attribute data of this object was not saved due to a failure during the attempt.
16#1A	ROUTING_FAILURE_REQUEST_PACKET_TOO_LARGE	The service request packet was too large for transmission on a CAA Net Base Services in the path to the destination. The routing device was forced to abort the service.
16#1B	ROUTING_FAILURE_RESPONSE_PACKET_TOO_LARGE	The service response packet was too large for transmission on a CAA Net Base Services in the path from the destination. The routing device was

		forced to abort the service.
16#1C	MISSING_ATTRIBUTE_LIST_ENTRY_DATA	The service did not supply an attribute in a list of attributes that was needed by the service to perform the requested behavior.
16#1D	INVALID_ATTRIBUTE_VALUE_LIST	The service is returning the list of attributes supplied with status information for those attributes that were invalid.
16#1E	EMBEDDED_SERVICE_ERROR	An embedded service resulted in an error.
16#1F	VENDOR_SPECIFIC_ERROR	A vendor specific error has been encountered. The Additional Code Field of the Error Response defines the particular error encountered. Use of this General Error Code should only be performed when none of the Error Codes presented in this table or within an Object Class definition accurately reflect the error.
16#20	INVALID_PARAMETER	A parameter associated with the request was invalid. This code is used when a parameter does not meet the requirements of this specification and/or the requirements defined in an Application Object Specification.
16#21	WRITE_ONEC_VALUE_OR_MEDIUM_ALREADY_WRITTEN	An attempt was made to write to a write-once medium (e. g. WORM drive, PROM) that has already been written, or to modify a value that cannot be changed once established.
16#22	INVALID_REPLY_RECEIVED	An invalid reply is received (e. g. reply service code does not match the request service code, or reply message is shorter than the minimum expected reply size). This status code can serve for other causes of invalid replies.
16#23	BUFFER_OVERFLOW	The message received is larger than the receiving buffer can handle. The entire message was discarded.
16#24	MESSAGE_FORMAT_ERROR	The format of the received message is not supported by the server.
16#25	KEY_FAILURE_IN_PATH	The Key Segment that was included as

		the first segment in the path does not match the destination module. The object specific status shall indicate which part of the key check failed.
16#26	PATH_SIZE_INVALID	The size of the path which was sent with the Service Request is either not large enough to allow the Request to be routed to an object or too much routing data was included.
16#27	UNEXPECTED_ATTRIBUTE_IN_LIST	An attempt was made to set an attribute that is not able to be set at this time.
16#28	INVALID_MEMBER_ID	The Member ID specified in the request does not exist in the specified Class/Instance/Attribute
16#29	MEMBER_NOT_SETTABLE	A request to modify a non-modifiable member was received
16#2A	GROUP_2_ONLY_SERVER_GENERAL_FAILURE	This error code may only be reported by DeviceNet Group 2 Only servers with 4K or less code space and only in place of Service not supported, Attribute not supported and Attribute not settable.
16#2B	UNKNOWN_MODBUS_ERROR	A CIP to Modbus translator received an unknown Modbus Exception Code.
16#2C	ATTRIBUTE_NOT_GETTABLE	A request to read a non-readable attribute was received
16#2D	INSTANCE_NOT_DELETABLE	The requested object instance cannot be deleted
16#2E	SERVICE_NOT_SUPPORTED_FOR_SPECIFIED_PATH	The object supports the service, but not for the designated application path (e. g. attribute). NOTE: Not to be used for any set service (use General Status Code 0x0E or 0x29 instead)
16#100	TIME_OUT	Request timed out.
	INTERFACE_MISSING	IEtherNetIPService is not implemented.
	REMOTE_CALL_FAILED	No physical connection.
	NULL_POINTER	Wrong input value NULL.
	INVALID_DATA_SIZE	Size of data unacceptable.
	WRONG_INTERFACE_VERSION	Version missmatch. Device implements not the same version of interface for the called method.
	NO_MEMORY	Not enough memory

	UNKNOWN_ERROR	An unknown error occured.
	ABORTED	Service was aborted

5.4.8 Set_Attribute_Single

设置对象实例的属性和服务数据

- **指令格式**

指令	名称	FB/FC	LD 表现	ST表现	库
Set_Attributes_Single	设置单个属性	FB		Set_Attribute_Single(xExecute:= , xDone=> , xBusy=> , xError=> , itfEtherNetIPDevice:= , eClass:= , dwInstance:= , eError=> , wAttribute:= , pData:= , udiDataSize:=);	EtherNetIP Services

- **相关变量**

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
xExecute	功能块使能	BOOL	TRUE-FALSE	-	若为TRUE（上升沿信号），则启动功能块运行
itfEtherNetIPDevice	远程从站实例化名	IEtherNetIPService	-	-	远程从站实例化名
eClass	类	CIP Class	-	-	服务指向的对象类别，可以是 CIP 标准目标对象，或者第三方厂商

					自定义对象
dwInstance	实例	DWORD	-	-	执行CIP时的实际实例数量，应大于等于1
wAttribute	属性	WORD	-	-	服务指向的属性
pData	接收缓存	POINTER TO BYTE	-	-	指向属性数据的指针，发送该数据到目标从站
udiDataSize	接收长度	UDINT	-	-	发送数据长度

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
xDone	完成	BOOL	TRUE-FALSE	-	客户端通信完成
xBusy	功能块运行中	BOOL	TRUE-FALSE	-	功能块运行中标志
xError	错误标志	BOOL	TRUE-FALSE	-	错误状态
eError	错误	ERROR	-	-	具体错误请参考EtherNetIPService库的枚举体ERROR

	布尔	位串				整数								实数		时刻、持续时间 日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
xExecute	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
itfEtherNetIPDevice	IEtherNetIPService																			
eClass	CIP Class																			
dwInstance	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
wAttribute	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
pData	POINTER TO BYTE																			
udiDataSize	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-

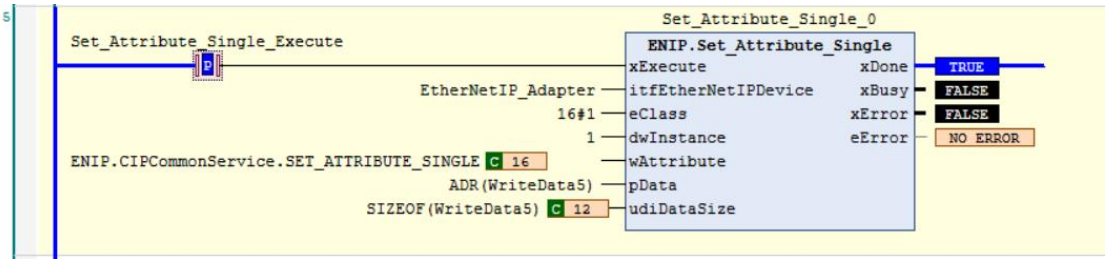
xDone	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
xBusy	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
xError	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
eError	ERROR																			

- **程序示例**

ST：可通过功能块Set_Attributes_Single，设置对象实例的单个属性和服务数据。

```
Set_Attribute_Single_0(
    xExecute TRUE :=Set_Attribute_Single_Execute TRUE ,
    xDone=> ,
    xBusy=> ,
    xError=> ,
    itfEtherNetIPDevice:= EtherNetIP_Adapter,
    eClass IdentityOb := 16#1,
    dwInstance 1 := 1,
    eError=> ,
    wAttribute 16 :=ENIP.CIPCommonService.SET_ATTRIBUTE_SINGLE ,
    pData 16#B43EFFC8 :=ADR(WriteData5) ,
    udiDataSize 12 := SIZEOF(WriteData5));
```

LD：可通过功能块Set_Attributes_Single，设置对象实例的单个属性和服务数据。



- **注意事项**

➢ itfEtherNetIPDevice为从站的实例化名称，eClass服务指向的对象类别。如果使用功能块中出现错误，eError将被赋相关错误值（枚举型ERROR里的各种值） 。详细请看以下错误代码。

CIP 公用服务及属性

ID	属性
16#1	IdentityObject
16#2	MessageRouterObject
16#3	DeviceNetObject

16#4	AssemblyObject
16#5	ConnectionObject
16#6	ConnectionManagerObject
16#7	RegisterObject
16#8	DiscreteInputPointObject
16#9	DiscreteOutputPointObject
16#A	AnalogInputPointObject
16#B	AnalogOutputPointObject
16#E	PresenceSensingObject
16#F	ParameterObject
16#10	ParameterGroupObject
16#12	GroupObject
16#1D	DiscreteInputGroupObject
16#1E	DiscreteOutputGroupObject
16#1F	DiscreteGroupObject
16#20	AnalogInputGroupObject
16#21	AnalogOutputGroupObject
16#22	AnalogGroupObject
16#23	PositionSensorObject
16#24	PositionControllerSupervisorObject
16#25	PositionControllerObject
16#26	BlockSequencerObject
16#27	CommandBlockObject
16#28	MotorDataObject
16#29	ControlSupervisorObject
16#2A	ACDCDriveObject
16#2B	AcknowledgeHandlerObject
16#2C	OverloadObject
16#2D	SoftstartObject
16#2E	SelectionObject
16#30	S_DeviceSupervisorObject
16#31	S_AnalogSensorObject
16#32	S_AnalogActuatorObject

16#33	S_SingleStageControllerObject
16#34	S_GasCalibrationObject
16#35	TripPointObject
16#37	FileObject
16#38	S_PartialPressureObject
16#39	SafetySupervisorObject
16#3A	SafetyValidatorObject
16#3B	SafetyDiscreteOutputPointObject
16#3C	SafetyDiscreteOutputGroupObject
16#3D	SafetyDiscreteInputPointObject
16#3E	SafetyDiscreteInputGroupObject
16#3F	SafetyDualChannelOutputObject
16#40	S_SensorCalibrationObject
16#41	EventLogObject
16#42	MotionDeviceAxisObject
16#43	TimeSyncObject
16#44	ModbusObject
16#45	OriginatorConnectionListObject
16#46	ModbusSerialLinkObject
16#47	DeviceLevelRingObject
16#48	QoSObject
16#49	SafetyAnalogInputPointObject
16#4A	SafetyAnalogInputGroupObject
16#4B	SafetyDualChannelAnalogInputObject
16#4C	SERCOSIIILinkObject
16#4D	TargetConnectionListObject
16#4E	EnergyObject
16#4F	ElectricalEnergyObject
16#50	Non_ElectricalEnergyObject
16#51	BaseSwitchObject
16#52	SNMPObject
16#53	PowerManagementObject
16#F0	ControlNetObject

16#F1	ControlNetKeeperObject
16#F2	ControlNetSchedulingObject
16#F3	ConnectionConfigurationObject
16#F4	PortObject
16#F5	TCPIPInterfaceObject
16#F6	EthernetLinkObject
16#F7	CompoNetLink
16#F8	CompoNetRepeater

CIP 及特定库错误

错误码	定义	描述
0	NO_ERROR	Service was successfully performed by the object specified.
16#1	CONNECTION_FAILURE	A connection related service failed along the connection path.
16#2	RESOURCE_UNAVAILABLE	Resources needed for the object to perform the requested service were unavailable
16#3	INVALID_PARAM_VALUE	See Status Code 0x20, which is the preferred value to use for this condition.
16#4	PATH_SEGMENT_ERROR	The path segment identifier or the segment syntax was not understood by the processing node. Path processing shall stop when a path segment error is encountered.
16#5	PATH_DESTINATION_UNKNOWN	The path is referencing an object class, instance or structure element that is not known or is not contained in the processing node. Path processing shall stop when a path destination unknown error is encountered.
16#6	PARTIAL_TRANSFER	Only part of the expected data was transferred.
16#7	CONNECTION_LOST	The messaging connection was lost.
16#8	SERVICE_NOT_SUPPORTED	The requested service was not implemented or was not defined for this Object Class/Instance.
16#9	INVALID_ATTRIBUTE_VALUE	Invalid attribute data detected.

16#A	ATTRIBUTE_LIST_ERROR	An attribute in the Get_Attribute_List or Set_Attribute_List response has a non-zero status.
16#B	ALREADY_IN_REQUEST_STATE	The object is already in the mode/state being requested by the service
16#C	OBJECT_STATE_ERROR	The object cannot perform the requested service in its current mode/state.
16#D	OBJECT_ALREADY_EXISTS	The requested instance of object to be created already exists.
16#E	ATTRIBUTE_NOT_SETTABLE	A request to modify a nonmodifiable attribute was received.
16#F	PRIVILEGE_VIOLATION	A permission/privilege check failed
16#10	DEVICE_STATE_ERROR	The device's current mode/state prohibits the execution of the requested service.
16#11	REPLY_DATA_TOO_LARGE	The data to be transmitted in the response buffer is larger than the allocated response buffer
16#12	FRAGMENTATION_OF_VALUE	The service specified an operation that is going to fragment a primitive data value, i. e. half a REAL data type.
16#13	NOT_ENOUGH_DATA	The service did not supply enough data to perform the specified operation.
16#14	ATTRIBUTE_NOT_SUPPORTED	The attribute specified in the request is not supported.
16#15	TOO_MUCH_DATA	The service supplied more data than was expected.
16#16	OBJECT_DOES_NOT_EXIST	The object specified does not exist in the device.
16#17	SERVICE_FRAGMENTATION_SEQUENCE_NOT_IN_PROGRESS	The fragmentation sequence for this service is not currently active for this data.
16#18	NO_STORED_ATTRIBUTE_DATA	The attribute data of this object was not saved prior to the requested service.
16#19	STORE_OPERATION_FAILURE	The attribute data of this object was not saved due to a failure during the attempt.
16#1A	ROUTING_FAILURE_REQUEST_PACKET_TOO_LARGE	The service request packet was too large for transmission on a CAA Net Base Services in the path to the destination.

		The routing device was forced to abort the service.
16#1B	ROUTING_FAILURE_RESPONSE_PACKET_TOO_LARGE	The service response packet was too large for transmission on a CAA Net Base Services in the path from the destination. The routing device was forced to abort the service.
16#1C	MISSING_ATTRIBUTE_LIST_ENTRY_DATA	The service did not supply an attribute in a list of attributes that was needed by the service to perform the requested behavior.
16#1D	INVALID_ATTRIBUTE_VALUE_LIST	The service is returning the list of attributes supplied with status information for those attributes that were invalid.
16#1E	EMBEDDED_SERVICE_ERROR	An embedded service resulted in an error.
16#1F	VENDOR_SPECIFIC_ERROR	A vendor specific error has been encountered. The Additional Code Field of the Error Response defines the particular error encountered. Use of this General Error Code should only be performed when none of the Error Codes presented in this table or within an Object Class definition accurately reflect the error.
16#20	INVALID_PARAMETER	A parameter associated with the request was invalid. This code is used when a parameter does not meet the requirements of this specification and/or the requirements defined in an Application Object Specification.
16#21	WRITE_ONEC_VALUE_OR_MEDIUM_ALREADY_WRITTEN	An attempt was made to write to a write-once medium (e. g. WORM drive, PROM) that has already been written, or to modify a value that cannot be changed once established.
16#22	INVALID_REPLY_RECEIVED	An invalid reply is received (e. g. reply service code does not match the request service code, or reply message is shorter than the minimum expected reply size). This status code can serve for other causes of invalid replies.

16#23	BUFFER_OVERFLOW	The message received is larger than the receiving buffer can handle. The entire message was discarded.
16#24	MESSAGE_FORMAT_ERROR	The format of the received message is not supported by the server.
16#25	KEY_FAILURE_IN_PATH	The Key Segment that was included as the first segment in the path does not match the destination module. The object specific status shall indicate which part of the key check failed.
16#26	PATH_SIZE_INVALID	The size of the path which was sent with the Service Request is either not large enough to allow the Request to be routed to an object or too much routing data was included.
16#27	UNEXPECTED_ATTRIBUTE_IN_LIST	An attempt was made to set an attribute that is not able to be set at this time.
16#28	INVALID_MEMBER_ID	The Member ID specified in the request does not exist in the specified Class/Instance/Attribute
16#29	MEMBER_NOT_SETTABLE	A request to modify a non-modifiable member was received
16#2A	GROUP_2_ONLY_SERVER_GENERAL_FAILURE	This error code may only be reported by DeviceNet Group 2 Only servers with 4K or less code space and only in place of Service not supported, Attribute not supported and Attribute not settable.
16#2B	UNKNOWN_MODBUS_ERROR	A CIP to Modbus translator received an unknown Modbus Exception Code.
16#2C	ATTRIBUTE_NOT_GETTABLE	A request to read a non-readable attribute was received
16#2D	INSTANCE_NOT_DELETABLE	The requested object instance cannot be deleted
16#2E	SERVICE_NOT_SUPPORTED_FOR_SPECIFIED_PATH	The object supports the service, but not for the designated application path (e. g. attribute). NOTE: Not to be used for any set service (use General Status Code 0x0E or 0x29 instead)
16#100	TIME_OUT	Request timed out.
	INTERFACE_MISSING	IEtherNetIPService is not implemented.
	REMOTE_CALL_FAILED	No physical connection.

	NULL_POINTER	Wrong input value NULL.
	INVALID_DATA_SIZE	Size of data unacceptable.
	WRONG_INTERFACE_VERSION	Version missmatch. Device implements not the same version of interface for the called method.
	NO_MEMORY	Not enough memory
	UNKNOWN_ERROR	An unknown error occured.
	ABORTED	Service was aborted

6 高速IO指令

6.1脉冲轴控制指令

6.1.1 指令列表

指令类别	名称	FB/FC	功能
脉冲轴控制	LS_Home_P	FB	脉冲轴回原
	LS_MotionControl_P	FB	脉冲轴绑定
	LS_ReadAxisPara_P	FB	获取脉冲轴的脉冲当量值
	LS_ResetAxis_P	FB	脉冲轴复位

6.1.2 LS_Home_P

脉冲轴回原点。

- **指令格式**

指令	名称	FB/FC	LD表现	ST表现	库
LS_Home_P	脉冲轴回原	FB		LS_Home_P(Axis:=, xExecute=, xDone=>, xBusy:=>, xCommandAborted:=> , xError=>, nErrorID=>);	MC_HSIO

- **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	初始值	描述
Axis	高速轴	AXIS_REF_VIRTUAL_SM3	-	-	参与回零轴

输入变量

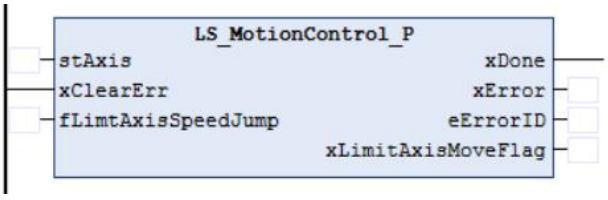
输入变量	名称	数据类型	有效范围	初始值	描述
xExecute	高电平有效	BOOL	TRUE-FALSE	FALSE	上升沿有效，状态从 FALSE 切换到 TRUE，执行功能块

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
xDone	完成标志	BOOL	TRUE-FALSE	FALSE	TRUE 表示功能块执行完成。
xBusy	执行标志	BOOL	TRUE-FALSE	FALSE	功能块尚未执行完，新的输出值正在计算中。
xCommandAborted	中止状态	BOOL	TRUE-FALSE	FALSE	当前功能块已不执行，被另一功能块所替代。
xError	错误标志	BOOL	TRUE-FALSE	FALSE	FALSE——没有错误， TRUE——执行错误。
diErrorID	错误 ID	SMC_ERROR	-	-	错误码，补充错误码请查找“MC_HSIO.LS_BasicModule.LS_ERROR”。

- **功能说明**

  - 本功能块需配合LS_MotionControl_P功能块一起使用，只需要在程序中调用该功能块即可，各输入输出引脚无需映射到变量。同时无论程序中使用多少路高速脉冲输出，只需调用一次。如图：



  - 在程序将需要回原点的轴与脉冲轴通道号进行关联，MC500系列最大支持6路脉冲轴，即PulAxis_0～PulAxis_5，如下“MC_HSIO.LS_MotionControl_P.stAxis.PulAxis_0 := ADR(LS_Axis_0);”，其中LS_Axis_0根据工程中虚拟驱动器的命名。

  - 将xEnable从FALSE切换为TRUE时，采用上升沿信号触发功能块执行。xEnable的下

降沿并不会停止或影响正在执行的功能块。

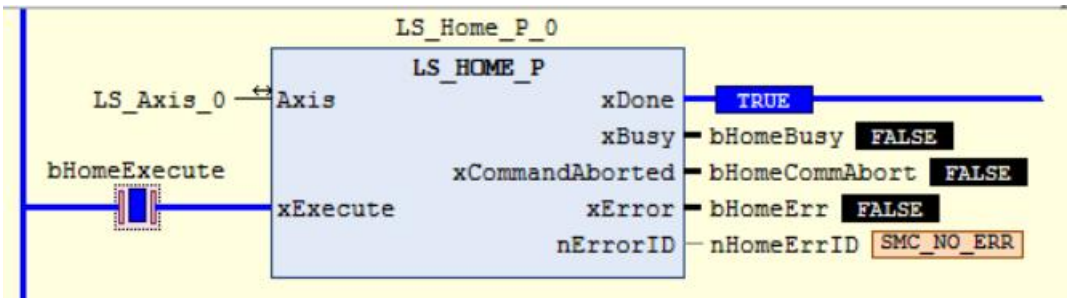
- 当指定轴正在执行回原点动作时，复位xEnable不会将轴停止，需使用MC_Stop功能块。

- **程序实例**

ST：

```
LS_Home_P_0(Axis:= LS_Axis_0,
        xExecute TRUE := bHomeExecute TRUE,
        xDone FALSE => bHomeDone FALSE,
        xBusy TRUE => bHomeBusy TRUE,
        xCommandAborted FALSE => bHomeCommAbort FALSE,
        xError FALSE => bHomeErr FALSE,
        nErrorID SMC_NO_ERR => nHomeErrID SMC_NO_ERR );
```

LD：



6.1.3 LS_MotionControl_P

脉冲轴绑定。

- **指令格式**

指令	名称	FB/FC	LD表现	ST表现	库
LS_Motion Control_P	脉冲轴绑定	FB		LS_MitionControl_P(stAxis:=, xClearErr:=, fLimtAxisSpeedJump:=, xDone=>, xError=>, nErrorID=>, xLimitAxisMoveFlag=>);	MC_HSIO

- **相关变量**

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
stAxis	高速轴	DUT_Pulse_Axis	-	-	6个脉冲规划轴结构体，需要在程序中绑定具体脉冲轴。
xClearErr	高电平有效	BOOL	TRUE-FALSE	FALSE	上升沿有效，状态从FALSE切换到TRUE，执行功能块。注意：使用时必须先调用MC_Reset清除轴报错状态，再调用ClearErr清除模块内报错。（该引脚仅用于清除模块内的错误标记和错误码）
fLimtAxisSpeedJump	限制轴实际输出时单周期速度跳变	LREAL	1～500000	500000	当速度跳变大于该值时，模块报错并限制脉冲输出。

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
xDone	完成标志	BOOL	TRUE-FALSE	FALSE	TRUE 表示可操作状态，FALSE 表示初始化等其他状态
xError	错误标志	BOOL	TRUE-FALSE	FALSE	FALSE——没有错误， TRUE——执行错误。
eErrorID	错误 ID	LS_ERROR	-	-	错误码。
xLimitAxisMoveFlag	限制轴运动标志	BOOL	TRUE-FALSE	FALSE	限制轴运动标志。

- **功能说明**

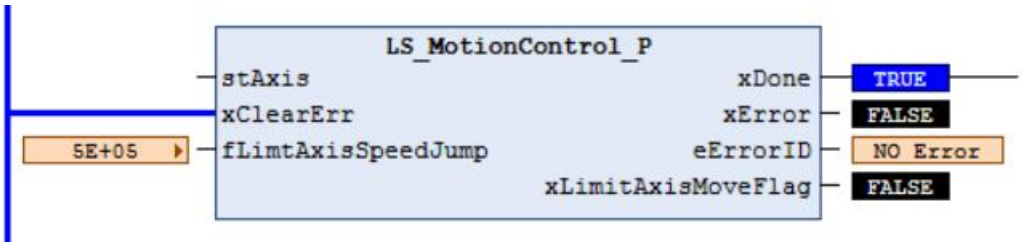
  - 用于脉冲轴绑定，只需调用一次即可。
  - 调用该功能块后，同时还需要在程序将脉冲轴与实际输出通道号进行关联，MC500系列最大支持6路脉冲轴，即PulAxis_0～PulAxis_5，如下“MC_HSIO.LS_MotionControl_P.stAxis.PulAxis_0 := ADR(LS_Axis_0);”，其中LS_Axis_0根据工程中虚拟驱动器的命名。

- **程序实例**

ST：

```
MC_HSIO.LS_MotionControl_P(stAxis:= ,
    xClearErr:= ,
    fLimtAxisSpeedJump:= ,
    xDone=> ,
    xError=> ,
    eErrorID=> ,
    xLimitAxisMoveFlag=> );
```

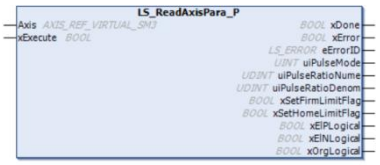
LD：



6.1.4 LS_ReadAxisPara_P

获取脉冲轴的脉冲当量值。

- **指令格式**

指令	名称	FB/FC	LD表现	ST表现	库
LS_ReadAxisPara_P	获取脉冲当量值	FB		LS_ReadAxisPara_P(Axis:=, xExecute:=, xDone=>, xError=>, eErrorID=>, uiPulseMode=>, uiPulseRatioNume=>, uiPulseRatioDenom=>, xSetFirmLimitFlag=>, xSetHomeLimitFlag=>, xEIPLogical=>,	MC_HSIO

				xEINLogical=>, xOrgLogical=>);	

- 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	初始值	描述
Axis	高速轴	AXIS_REF_VIRTUAL_SM3	-	-	获取脉冲当量值的轴。

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
xExecute	高电平有效	BOOL	TRUE-FALSE	FALSE	上升沿有效，状态从 FALSE 切换到 TRUE，执行功能块。

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
xDone	完成标志	BOOL	TRUE-FALSE	FALSE	功能块执行完成。
xError	错误标志	BOOL	TRUE-FALSE	FALSE	FALSE——没有错误， TRUE——执行错误。
eErrorID	错误 ID	LS_ERROR	-	-	错误码。
uiPulseMode	脉冲轴输出模式	UINT	0-6	0	轴的脉冲输出模式。0-3：脉冲方向模式；4-5：双脉冲模式；6：AB 相模式。
uiPulseRatioNume	轴传动比分子	UDINT	遵循数据类型	0	轴传动比分子。
uiPulseRatioDenom	轴传动比分母	UDINT	遵循数据类型	0	轴传动比分母。

xSetFirmLimitFlag	硬件限位是否有效	BOOL	TRUE-FALSE	FALSE	系统设置的硬件限位是否有效，FALSE 表示不处理硬件限位，TRUE 表示处理硬件限位。
xSetHomeLimitFlag	回原限位是否有效	BOOL	TRUE-FALSE	FALSE	回原时限位是否有效；注意：如果 SetFirmLimitFlag 置为 TRUE 时，对回原运动也有效。
xEIPLogical	正限位逻辑电平	BOOL	TRUE-FALSE	FALSE	正限位信号逻辑电平，FALSE：常开；TRUE：常闭。
xEINLogical	负限位逻辑电平	BOOL	TRUE-FALSE	FALSE	负限位信号逻辑电平，FALSE：常开；TRUE：常闭。
xOrgLogical	原点信号逻辑电平	BOOL	TRUE-FALSE	FALSE	原点信号逻辑电平，FALSE：常开；TRUE：常闭。

- **功能说明**

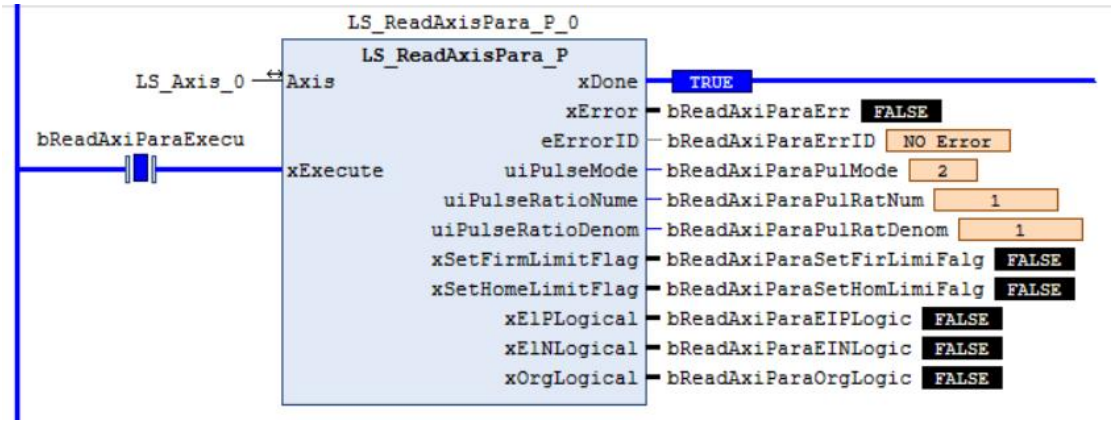
  - 将xEnable从FALSE设为TRUE时，采取上升沿信号触发功能块。

- **程序实例**

ST：

```
LS_ReadAxisPara_P_0(Axis:= LS_Axis_0,
                    xExecute TRUE := bReadAxiParaExecu TRUE,
                    xDone TRUE => bReadAxiParaDone TRUE,
                    xError FALSE => bReadAxiParaErr FALSE,
                    eErrorID NO_Error => bReadAxiParaErrID NO_Error,
                    uiPulseMode 2 => uiReadAxiParaPulMode 2,
                    uiPulseRatioNume 1 => udReadAxiParaPulRatNum 1,
                    uiPulseRatioDenom 1 => udReadAxiParaPulRatDenom 1,
                    xSetFirmLimitFlag FALSE => bReadAxiParaSetFirLimiFalg FALSE,
                    xSetHomeLimitFlag FALSE => bReadAxiParaSetHomLimiFalg FALSE,
                    xElPLogical FALSE => bReadAxiParaEIPLogic FALSE,
                    xElNLogical FALSE => bReadAxiParaEINLogic FALSE,
                    xOrgLogical FALSE => bReadAxiParaOrgLogic FALSE);
```

LD：



6.1.5 LS_ResetAxis_P

脉冲轴复位。主要实现脉冲输出轴错误复位操作。

- **指令格式**

指令	名称	FB/FC	LD表现	ST表现	库
LS_ResetAxis_P	脉冲轴复位	FB		LS_ResetAxis_P(Axis:=, xExecute:=, xDone=>, xBusy=>, xError=>, eErrorID=>);	MC_HSIO

- **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	初始值	描述
Axis	高速轴	AXIS_REF_VIRTUAL_SM3	-	-	指定使用的脉冲轴。

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
xExecute	高电平有效	BOOL	TRUE-FALSE	FALSE	上升沿有效，状态从 FALSE 切换到 TRUE，执行功能块。

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
xDone	完成标志	BOOL	TRUE-FALSE	FALSE	功能块执行完成。
xBusy	执行标志	BOOL	TRUE-FALSE	FALSE	功能块尚未执行完，新的输出值正在计算中。
xError	错误标志	BOOL	TRUE-FALSE	FALSE	FALSE——没有错误， TRUE——执行错误。
eErrorID	错误 ID	SMC_ERROR	-	-	错误码。

- **功能说明**

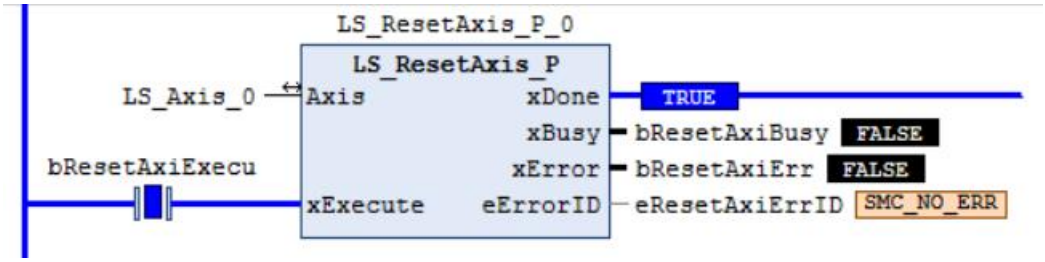
  - 将xEnable从FALSE设为TRUE时，采取上升沿信号触发功能块。

- **程序实例**

ST：

```
LS_ResetAxis_P_0(Axis:= LS_Axis_0,
                 xExecute TRUE := bResetAxiExecu TRUE,
                 xDone TRUE => bResetAxiDone TRUE,
                 xBusy FALSE => bResetAxiBusy FALSE,
                 xError FALSE => bResetAxiErr FALSE ,
                 eErrorID SMC_NO_ERR => eResetAxiErrID SMC_NO_ERR );
```

LD：



6.2 高速计数器指令

6.2.1 指令列表

指令类别	名称	FB/FC	功能
高速计数器	LS_Counter	FB	高速计数器指令
	LS_PresetValue	FB	预置计数值指令

6.2.2 LS_Counter

高速计数器指令。获取高速计数模块的设置参数以及计数值，需要先在 Device 下的 High Speed IO Module 中配置对应的硬件口功能及参数，否则该功能无效。

- **指令格式**

指令	名称	FB/FC	LD表现	ST表现	库
LS_Counter	高速计数器	FB		LS_Counter(xEnable:=, eChannel:=, xValid=>, xError=>, eErrorID=>, diValue=>);	MC_HSIO

- **相关变量**

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
xEnable	有效	BOOL	TRUE-FALSE	FALSE	必须设置为 TRUE，以激活功能块的处理。
eChannel	通道号	LS_ENCODER_CHAN	0-8	0	计数器0-5，编码器6-8。

输出变量

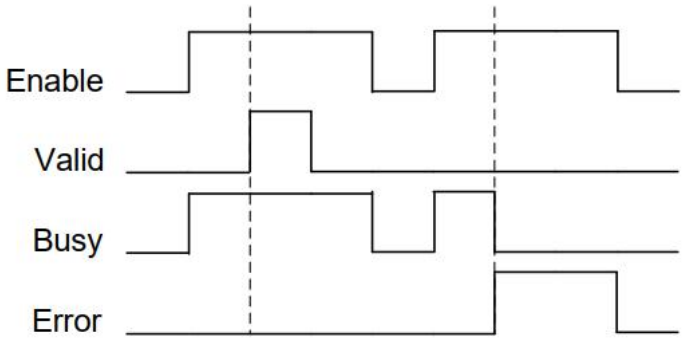
输出变量	名称	数据类型	有效范围	初始值	描述
xValid	执行完成	BOOL	TRUE-FALSE	FALSE	轴准备好则为 TRUE。

xError	错误	BOOL	TRUE-FALSE	FALSE	发生异常时为 TRUE
eErrorID	错误代码	LS_ERROR	-	0	正常时数值为 0，发生异常时，输出报错代码。
diValue	当前计数值	DINT	遵循数据类型	0	所选择通道的当前计数值。

- **功能说明**

  - 主要实现计数器使能、计数、测频、计数方向输出功能。

- **时序图：**



LS_Counter 的时序图

6.2.3 LS_PresetValue

预置计数值指令。需要先在 Device 下的 High Speed IO Module 中配置对应的硬件口功能及参数，否则该功能无效。

- **指令格式**

指令	名称	FB/FC	LD表现	ST表现	库
LS_PresetValue	预置计数值	FB		LS_PresetValue(xExecute:=, eChannel:=, eTriggerEdge:=, diPresetValue:=, xDone=>, xBusy=>, xError=>, eErrorID=>);	MC_HSIO

- **相关变量**

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
xExecute	高电平有效	BOOL	TRUE-FALSE	FALSE	上升沿有效，状态从 FALSE 切换到 TRUE，读取 IP 地址。
eChannel	通道号	LS_ENCODER_CHAN	0-8	0	计数器 0-5，编码器 6-8。
eTriggerEdge	厂商参数	LS_PRESET_EDGE_TYPE	-	-	目前暂不支持。
diPresetValue	计数器预设值	DINT	遵循数据类型	0	计数器预设值。

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
xDone	执行完成	BOOL	TRUE-FALSE	FALSE	功能块执行完成。
xBusy	功能块执行中	BOOL	TRUE-FALSE	FALSE	功能块执行中。
xError	错误	BOOL	TRUE-FALSE	FALSE	发生异常时为 TRUE。
eErrorID	错误代码	LS_ERROR	–	0	正常时数值为 0，发生异常时，输出报错代码。参考库 LS_BasicModule 的“LS_ERROR”。

- **功能说明**

  - 该功能块属于上升沿单次触发有效。

6.3LC1000本地高速计数器

6.3.1 指令列表

指令类别	名称	FB/FC	功能
LC1000本地高速计数器	LS_Counter	FB	高速计数器指令
	LS_PresetValue	FB	预置计数值指令
	LS_Compare	FB	单点比较输出
	LS_CompareStep	FB	等距比较输出
	LS_CompareFIFO	FB	队列比较输出
	LS_PWM	FB	PWM 输出
	LS_EnableInterrupt	FB	中断使能

6.3.2 LS_Counter

高速计数器使能。

- **指令格式**

指令	名称	FB/FC	LD表现	ST表现	库
LS_Counter	高速计数器	FB		LS_Counter(Axis:= , xEnable:= , xValid=> , xBusy=> , xError=> , eErrorID=> , fValue=> , fFrequency=> , eDir=>);	LS_HSIO

- **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	初始值	描述
Axis	编码器轴	AXIS_REF_MAPPING_LS	-	-	AXIS_REF_MAPPING_LS 类型编码器轴实例

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
xEnable	使能	BOOL	TRUE FALSE	FALSE	电平触发， TRUE 使能计数器。

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
xValid	执行完成	BOOL	TRUE FALSE	FALSE	计数器使能的状态
xBusy	执行标志	BOOL	TRUE FALSE	FALSE	TRUE 执行计数器使能命令
xError	错误	BOOL	TRUE FALSE	FALSE	发生异常时为 TRUE
eErrorID	错误代码	LS_ERROR	-	0	发生异常时，输出报错代码。
fValue	计数值	LREAL	遵循数据类型	0.0	当前计数值，根据缩放比例换算后的结果，单位 unit
fFrequency	计数频率	LREAL	遵循数据类型	0.0	当前计数频率，单位 unit/s
eDir	方向	LS_DIR	[-1, 1]	0	当前计数方向，-1:负向 1:正向

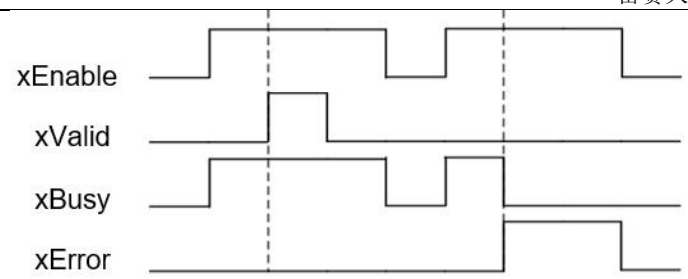
- **功能说明**

  - 主要实现本地计数器计数开始/停止，检测到 xEnable 置 TRUE 后，计数器响应外部脉冲信号，并开始计数，检测到 xEnable 置 FALSE 后,停止计数。 计数器位置在计数器模式的范围内变化，位置单位为 Unit。

  - 计数方向更改，需通过软件界面修改，不同计数模式的计数方向定义如下。更改方向后，需重新使能计数器功能块

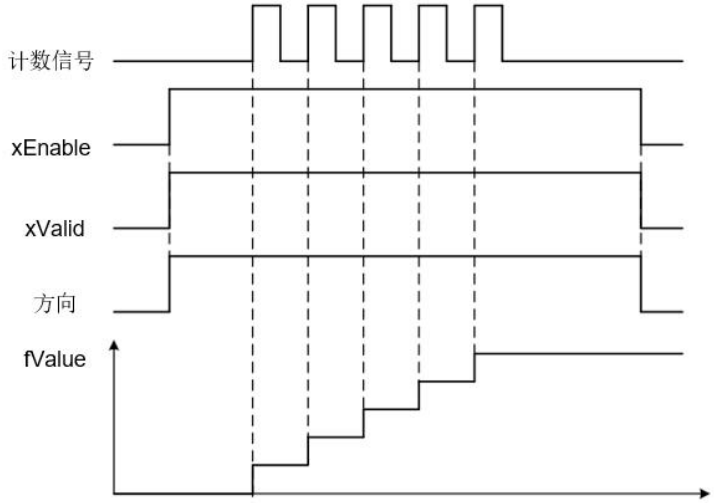
方向	AB相	脉冲+方向	CW/CCW	单相计数
正向	A 相超前 B 相增计数 B 相超前 A 相减计数	方向信号低电平减计数 方向信号高电平增计数	A 相增计数 B 相减计数	增计数
负向	A 相超前 B 相减计数 B 相超前 A 相增计数	方向信号低电平增计数 方向信号高电平减计数	A 相减计数 B 相增计数	减计数

- **时序图**



- **示例**

以脉冲 + 方式 正向计数为例，计数器增计数如下图所示。



6.3.3 LS_PresetValue

高速计数器预置值。

- **指令格式**

指令	名称	FB/FC	LD表现	ST表现	库
LS_ PresetValue	预置值	FB		LS_PresetValue(Axis:= , xExecute:= , xAbort:= , eTriggerType:= , fPresetValue:= , xDone=> , xBusy=> , xCommandAborted=> , xError=> , eErrorID=>);	LS_HSIO

- **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	初始值	描述
Axis	编码器轴	AXIS_REF_MAPPING_LS	-	-	AXIS_REF_MAPPING_LS 类型编码器轴实例

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
xExecute	执行	BOOL	TRUE FALSE	FALSE	边沿触发，执行预置功能的开关。 1）选择指令上升沿触发时，该管脚给入上升沿信号实现预置 2）选择外部DI触发时，该管脚给入电平信号
xAbort	中止	BOOL	TRUE FALSE	FALSE	电平触发 TRUE：中止功能块
eTriggerType	触发方式	LS_PRESET_EDGE_TYPE	[0,3]	0	0：指令上升沿触发 1：外部 DI 上升沿触发 2：外部 DI 下降沿触发 3：外部 DI 上升沿或下降沿触发
fPresetValue	预置值	LREAL	遵循数据类型	0.0	计数器的预置值，单位： unit，转换为 Pulse 单位需满足 -2147483648~2147483647 范围内

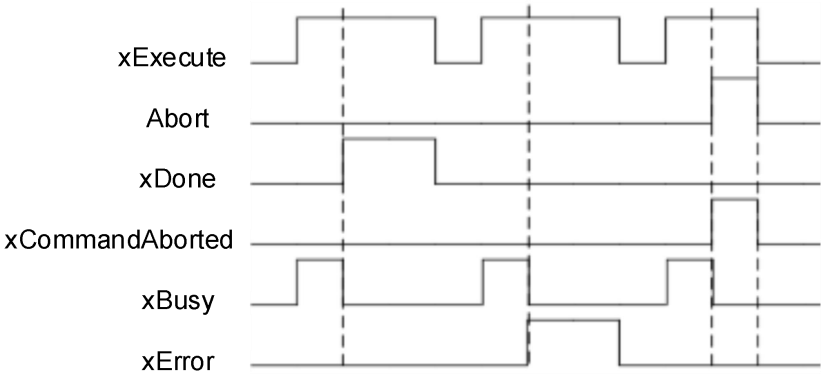
输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
xDone	执行完成	BOOL	TRUE FALSE	FALSE	TRUE 预置执行成功
xBusy	执行标志	BOOL	TRUE FALSE	FALSE	TRUE 执行预置命令中
xCommandAborted	中止标志	BOOL	TRUE FALSE	FALSE	TRUE功能块执行被中断
xError	错误	BOOL	TRUE FALSE	FALSE	发生异常时为 TRUE
eErrorID	错误代码	LS_ERROR	–	0	发生异常时，输出报错代码。

- **功能说明**

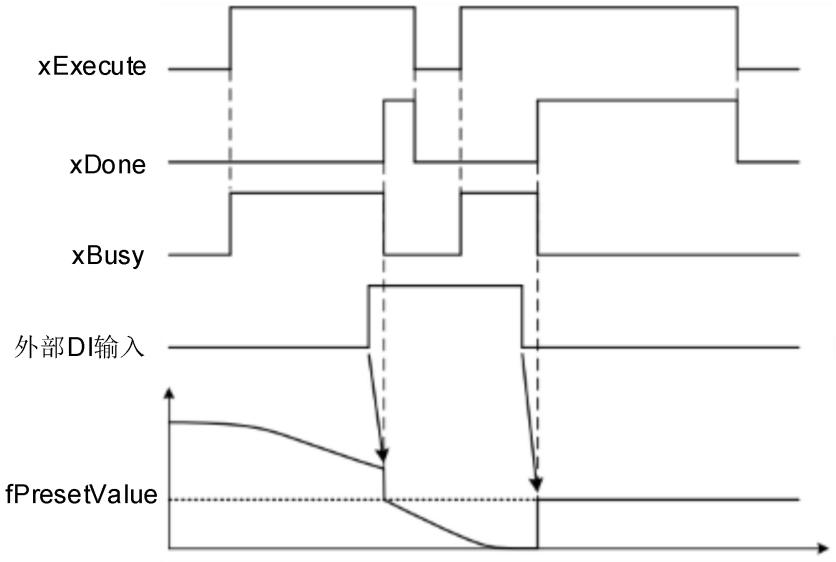
  - 本功能块主要实现计数器的预置功能，上升沿触发功能块执行，预置完成后输出 xDone 信号。
  - 预置触发方式 eTriggerType 总共 4 种，可选择指令上升沿触发、外部 DI 输入触发（上升沿，下降沿，任意沿）。
  - 预置条件选择指令上升沿触发时，xExecute为指令触发管脚，上升沿触发。
  - 预置条件选择外部DI输入触发时，需要在计数器参数配置界面选择预置输入口IN0~IN7，外部DI触发时，xExecute为电平使能信号。

- **时序图**



- **示例**

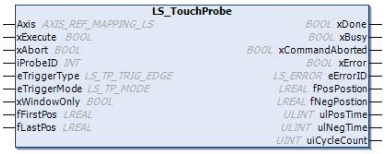
eTriggerType = 3（外部 DI 任意沿）时，指令时序如下



6.3.4 LS_TouchProbe

高速计数器探针。

- **指令格式**

指令	名称	FB/FC	LD表现	ST表现	库
LS_TouchProbe	探针	FB		LS_TouchProbe(Axis:= , xExecute:= , xAbort:= , iProbeID:= , eTriggerType:= , eTriggerMode:= , xWindowOnly:= , fFirstPos:= , fLastPos:= , xDone=> , xBusy=> , xCommandAborted=> , xError=> , eErrorID=> , fPosPostion=> , fNegPostion=> , ulPosTime=> , ulNegTime=> , uiCycleCount=>);	LS_HSIO

- **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	初始值	描述
Axis	编码器轴	AXIS_REF_MAPPING_LS	-	-	AXIS_REF_MAPPING_LS 类型编码器轴实例

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
xExecute	执行	BOOL	TRUE FALSE	FALSE	边沿触发，启动探针功能

xAbort	中止	BOOL	TRUE FALSE	FALSE	电平触发 TRUE：中止功能块执行
iProbeID	探针 ID	INT	[0,1]	0	探针 ID 号
eTriggerType	触发类型	LS_TP_TRIG_EDGE	[0,2]	0	触发类型： 0：上升沿 1：下降沿 2：双边沿
eTriggerMode	触发模式	LS_TP_MODE	[0,1]	0	触发模式： 0：单次锁存 1：连续锁存
xWindowOnly	窗口有效	BOOL	TRUE FALSE	FALSE	True时锁存窗生效，只接受窗内位置
fFirstPos	起始位置	LREAL	遵循数据范围	0.0	指定开始接收触发的位置，unit 单位
fLastPos	结束位置	LREAL	遵循数据范围	0.0	指定停止接受触发的位置，unit 单位

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
xDone	执行完成	BOOL	TRUE FALSE	FALSE	TRUE 探针功能执行成功
xBusy	执行标志	BOOL	TRUE FALSE	FALSE	TRUE 执行探针功能中
xCommandAborted	中止标志	BOOL	TRUE FALSE	FALSE	TRUE功能块执行被中断
xError	错误	BOOL	TRUE FALSE	FALSE	发生异常时为 TRUE
eErrorID	错误代码	LS_ERROR	-	0	发生异常时，输出报错代码。
fPosPostion	上升沿锁存位置	LREAL	遵循数据范围	0.0	上升沿锁存位置，unit 单位
fNegPosition	下降沿锁存位置	LREAL	遵循数据范围	0.0	下降沿锁存位置，unit 单位
ulPosTime	上升沿锁	ULINT	[0,2^63)	0	上升沿锁存时间（单位 ns）

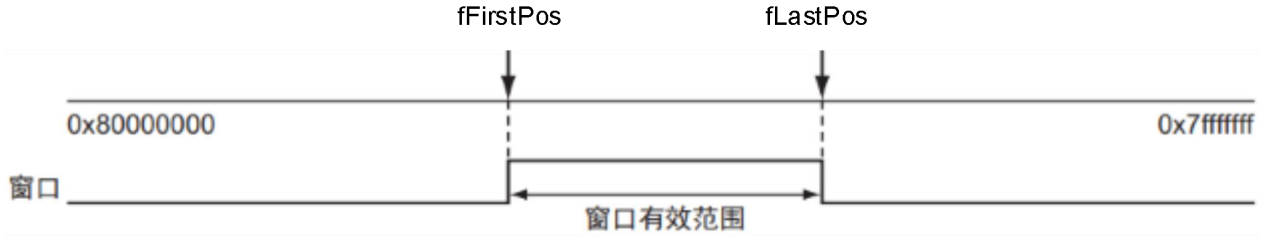
	存时间				
ulNegTime	下降沿锁存时间	ULINT	[0,2^63)	0	下降沿锁存时间（单位 ns）
uiCycleCount	锁存次数	UINT	[0,2^16)	0	连续锁存模式下，锁存完成一次累加一次

- **功能说明**

  - 本功能块主要实现外部 DI发生时刻，锁存计数器计数值的功能，同时支持锁存本地高速 IO 模块内部时间戳功能，上升沿触发功能块执行，锁存完成后输出 xDone 信号。
  - 在同一 LS_TouchProbe指令的实例中再次锁存计数器位置前，锁存位置的输出值将保持不变
  - 每个计数器轴支持两路探针，软件参数配置界面，选择探针输入口DI0~DI7。每路探针支持DI的上升沿、下降沿和双边沿锁存。
  - 每路探针支持单次锁存、连续锁存两种锁存方式。
  - 支持窗口有效设定，xWindowOnly指定为 TRUE（有效）时，仅检测计数器位置为 fFirstPos（开始位置）和 fLastPos（结束位置）范围内的触发。
  - 计数器可在参数配置界面，选择直线模式和旋转模式。

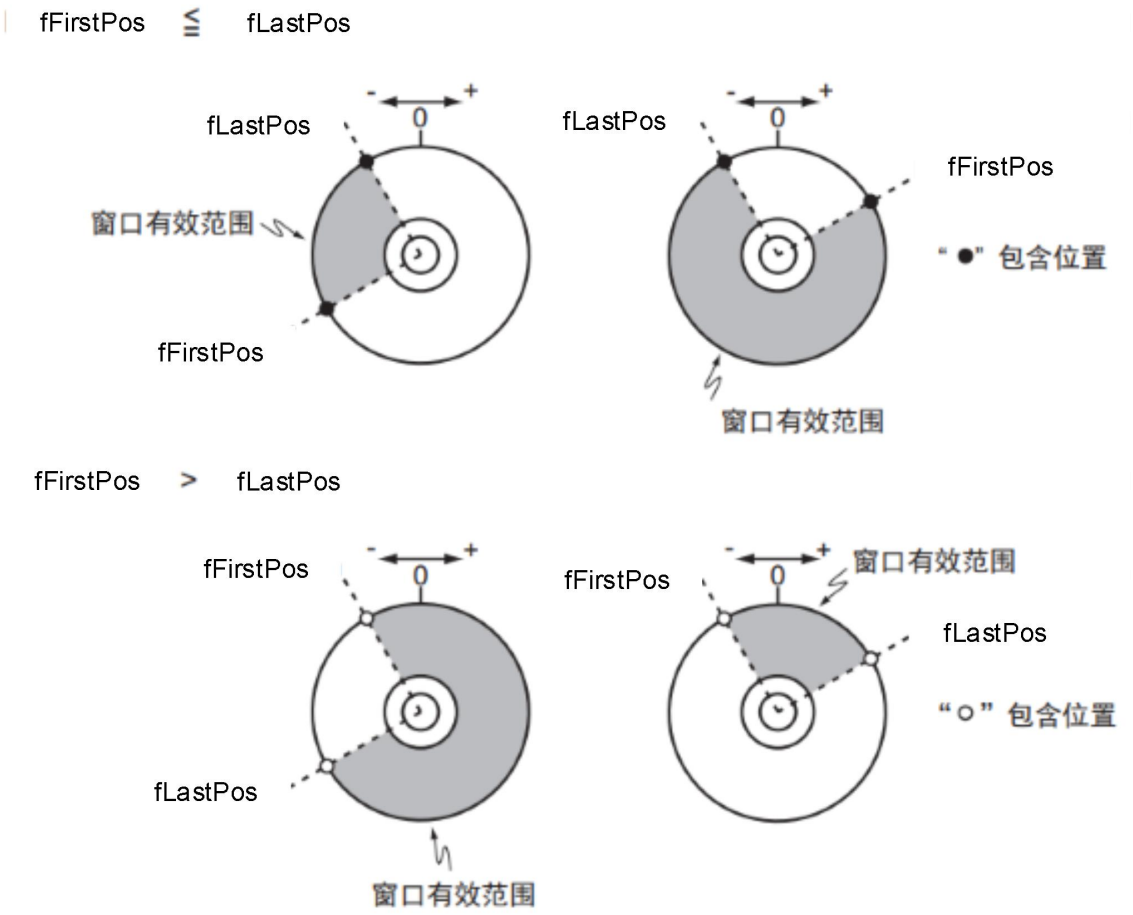
直线模式

  - 窗口的有效范围按以下公式表述。 fFirstPos （开始位置）≦ 窗口范围 ≦ fLastPos （结束位置）
  - 指定为 fFirstPos （开始位置）＞ fLastPos （结束位置）时，将发生异常。
  - 指定为超出 ［直线模式］的位置范围时，也会发生异常。

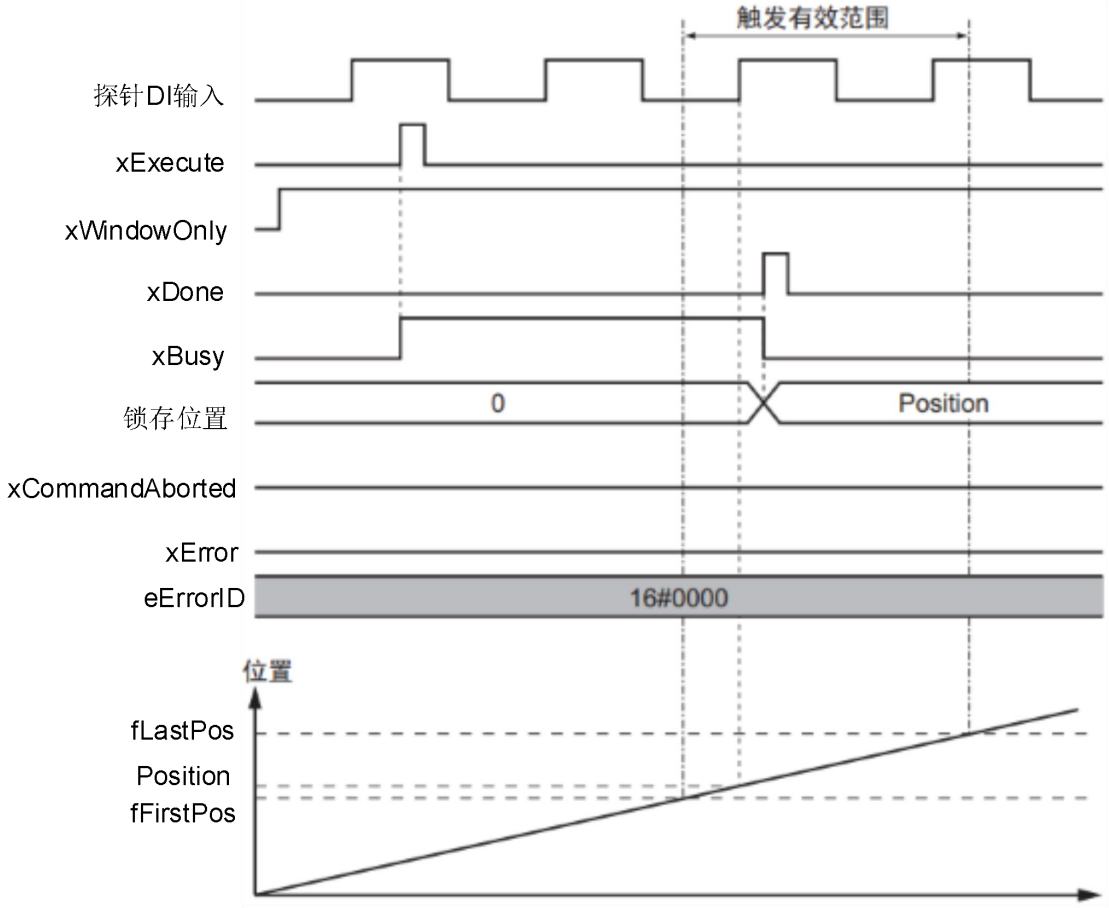


旋转模式

- 可指定为 fFirstPos （开始位置）≦ fLastPos（结束位置）及 fFirstPos（开始位置）＞ fLastPos（结束位置）。
- fFirstPos（开始位置）＞ fLastPos（结束位置）时，将跨循环计数器的上下限位置。
- 若指定的位置超出循环计数器上下限范围，将发生异常。

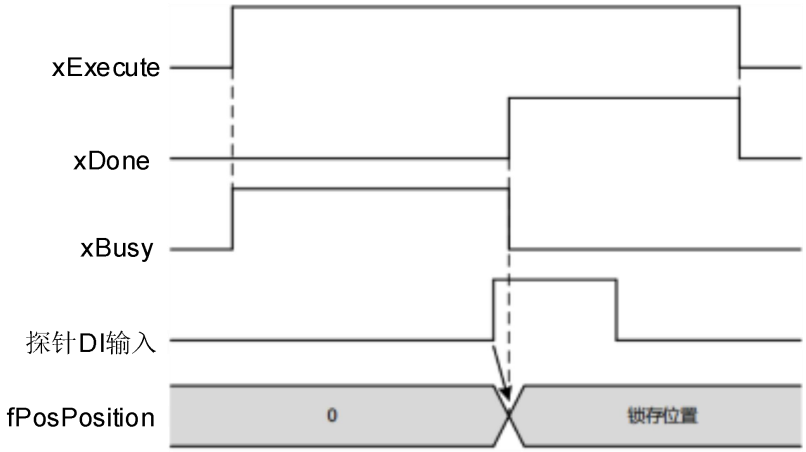


- **时序图**

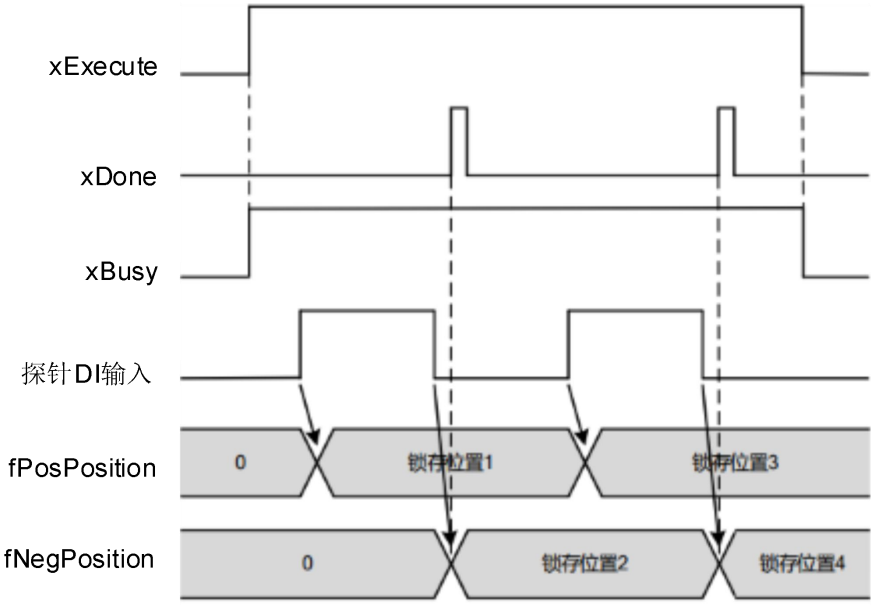


- **示例**

外部 DI 上升沿触发（eTriggerType =0），单次触发模式（eTriggerMode =0），指令时序图如下图所示.



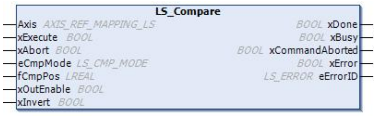
外部 DI 双边沿触发（eTriggerType =2），单次触发模式（eTriggerMode =0），先上升沿，后下降沿， 指令时序图如下图所示。



6.3.5 LS_Compare

高速一维单点比较。

- **指令格式**

指令	名称	FB/FC	LD表现	ST表现	库
LS_Compare	高速一维单点比较	FB		LS_Compare(Axis:= , xExecute:= , xAbort:= , fCmpPos:= , xOutEnable:= , eOutType:= , udiOutPara:= , xDone=> , xBusy=> , xCommandAborted=> , xError=> , eErrorID=>);	LS_HSIO

- **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	初始值	描述

Axis	编码器轴	AXIS_REF_MAPPING_LS	-	-	AXIS_REF_MAPPING_LS 类型编码器轴实例

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
xExecute	执行	BOOL	TRUE FALSE	FALSE	边沿触发，启动功能块
xAbort	中止	BOOL	TRUE FALSE	FALSE	电平触发 TRUE：中止功能块执行
fCmpPos	比较位置	LREAL	遵循数据范围	0.0	比较值，单位：unit
xOutEnable	使能输出	BOOL	TRUE FALSE	FALSE	使能硬件端口输出功能。使用该功能时，需在参数配置界面配置对应的输出口。 未使能时，功能块正常执行，仅硬件端口无输出
eOutType	输出类型	LS_CMP_OUTTPYE	[0,3]	0	0：时间方式-输出FALSE 1：时间方式-输出TRUE 2：脉冲方式-输出FALSE 3：脉冲方式-输出TRUE
udiOutPara	输出保持参数	UDINT	遵循数据类型	0	① 时间方式：打开输出口保持时间，us单位，[1,20000000] ② 脉冲方式：距离值，pulse单位，脉冲值范围[1~2147483647]

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
xDone	执行完成	BOOL	TRUE FALSE	FALSE	TRUE 比较功能执行成功
xBusy	执行标志	BOOL	TRUE FALSE	FALSE	TRUE 执行比较功能中
xCommandAborted	中止标志	BOOL	TRUE FALSE	FALSE	TRUE功能块执行被中断
xError	错误	BOOL	TRUE FALSE	FALSE	发生异常时为 TRUE
eErrorID	错误代码	LS_ERRO	-	0	发生异常时，输出报错代码。

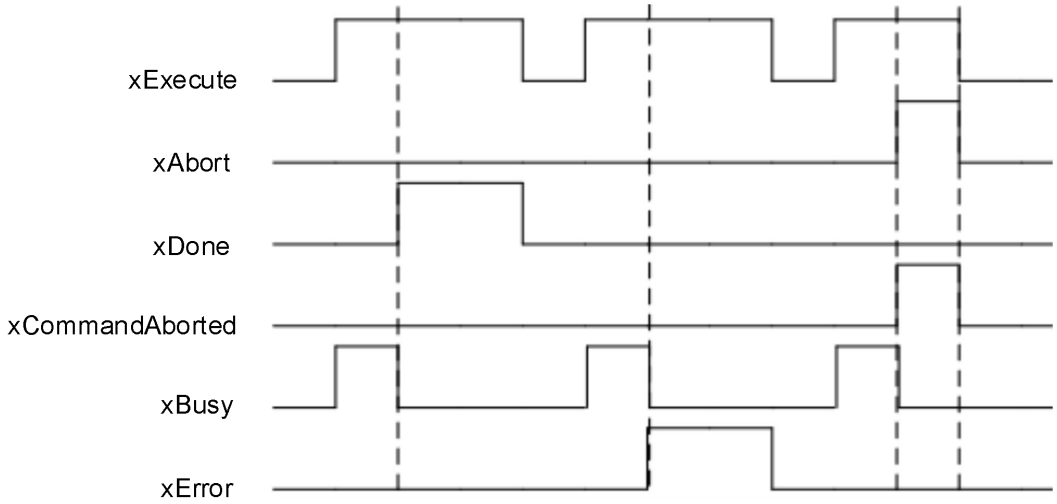
		R			

- **功能说明**

  - 本功能块实现计数器当前计数值与设置比较值相等时，触发一个高速输出口输出，并保持一段时间或一段距离。比较一致输出功能属于单次触发，当 xDone=TRUE 后，比较一致功能完成，如果需要继续使用比较功能，再次触发 xExecute。

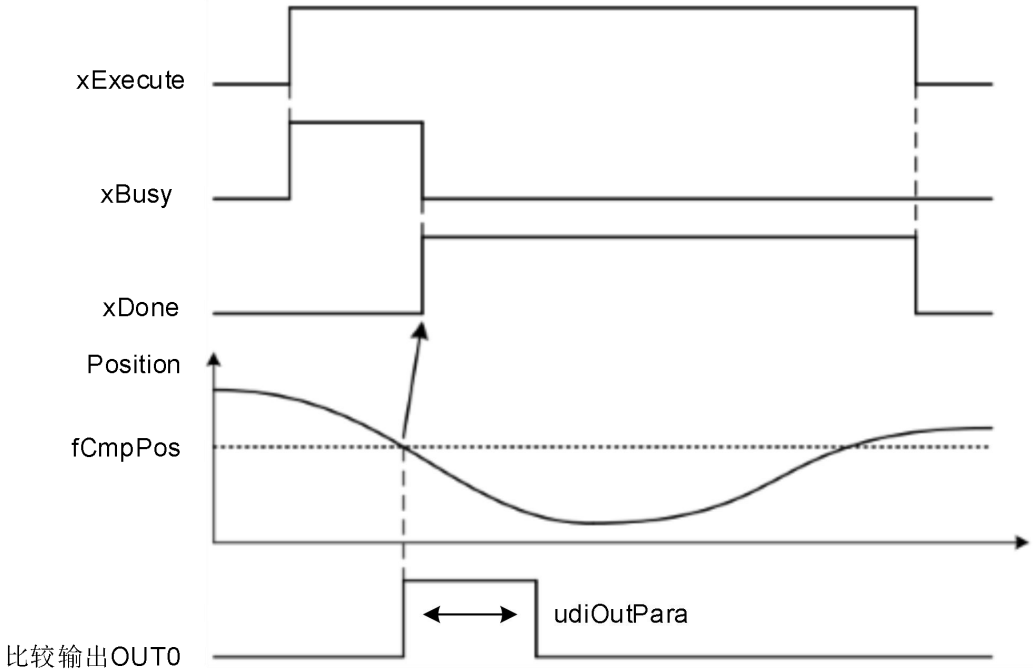
  - 比较功能块使用前，必须在对应计数器参数界面上配置输出端口，输出端口可配置 OUT0~OUT7。

- **时序图**



- **示例**

计数器界面上配置比较输出端口 OUT0。功能块使能比较输出端口（xOutEnable = TRUE），比较位置（fCmpPos = 1000.0，假设齿轮比为1：1，1unit=1pulse），输出类型 脉冲方式-输出TRUE（eOutType = 3），输出保持参数（udiOutPara = 100），当计数值到达比较位置1000时，OUT0输出口导通，再累计100个脉冲后，输出口关闭。



- **注意事项**

➢ 启动高速比较输出功能块时，输出OUT点的初始状态会自动翻转为第一个比较输出点的相反状态。

6.3.6 LS_CompareStep

高速计数器等距比较输出。

- **指令格式**

指令	名称	FB/FC	LD表现	ST表现	库
LS_CompareStep	等距比较	FB		LS_CompareStep(Axis:= , xExecute:= , xAbort:= , fStartPos:= , fStep:= , uiCmpNum:= , xOutEnable:= , eOutType:= , udiOutPara:= , xDone=> , xBusy=> , xCommandAborted=> ,	LS_HSIO

				xError=> , eErrorID=> , uiPoints=> , fCurrentPos=>);	

- **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	初始值	描述
Axis	编码器轴	AXIS_REF_MAPPING_LS	-	-	AXIS_REF_MAPPING_LS 类型编码器轴实例

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
xExecute	执行	BOOL	TRUE FALSE	FALSE	边沿触发，启动功能块
xAbort	中止	BOOL	TRUE FALSE	FALSE	电平触发 TRUE：中止功能块执行
fStartPos	开始位置	LREAL	遵循数据范围	0.0	起始比较值，unit 单位
fStep	步长	LREAL	遵循数据范围	0.0	比较等距步长，unit 单位，正值位置递增，负值位置递减
uiCmpNum	比较点个数	UINT	[1,65535]	0	比较点个数
xOutEnable	使能输出	BOOL	TRUE FALSE	FALSE	使能硬件端口输出功能。使用该功能时，需在参数配置界面配置对应的输出口。 未使能时，功能块正常执行，仅硬件端口无输出
eOutType	输出类型	LS_CMP_OUTTPYE	[0,3]	0	0：时间方式-输出FALSE 1：时间方式-输出TRUE 2：脉冲方式-输出FALSE 3：脉冲方式-输出TRUE

udiOutPara	输出保持参数	UDINT	遵循数据类型	0	③ 时间方式：打开输出口保持时间，us单位，[1,20000000] ④ 脉冲方式：距离值，pulse单位，脉冲值范围[1~2147483647]

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
xDone	执行完成	BOOL	TRUE FALSE	FALSE	TRUE 比较功能执行成功
xBusy	执行标志	BOOL	TRUE FALSE	FALSE	TRUE 执行比较功能中
xCommand Aborted	中止标志	BOOL	TRUE FALSE	FALSE	TRUE功能块执行被中断
xError	错误	BOOL	TRUE FALSE	FALSE	发生异常时为 TRUE
eErrorID	错误代码	LS_ERROR	-	0	发生异常时，输出报错代码。
uiPoints	已比较点数	UINT	[0,65535]	0	已比较点数
fCurrentPos	当前比较位置	LREAL	遵循数据范围	0.0	当前比较位置，unit 单位

- **功能说明**

  - 本功能块实现计数器当前计数值与设置的比较值相等时，触发一个硬件高速输出口，并保持一段时间或一段距离。比较值根据步长进行递增或递减，假设比较初始值为5，比较步长为2，比较点数为5，则比较值为5,7,9,11,13。

  - 比较功能块使用前，必须在对应计数器界面上配置输出端口，输出端口可配置 OUT0~OUT7。

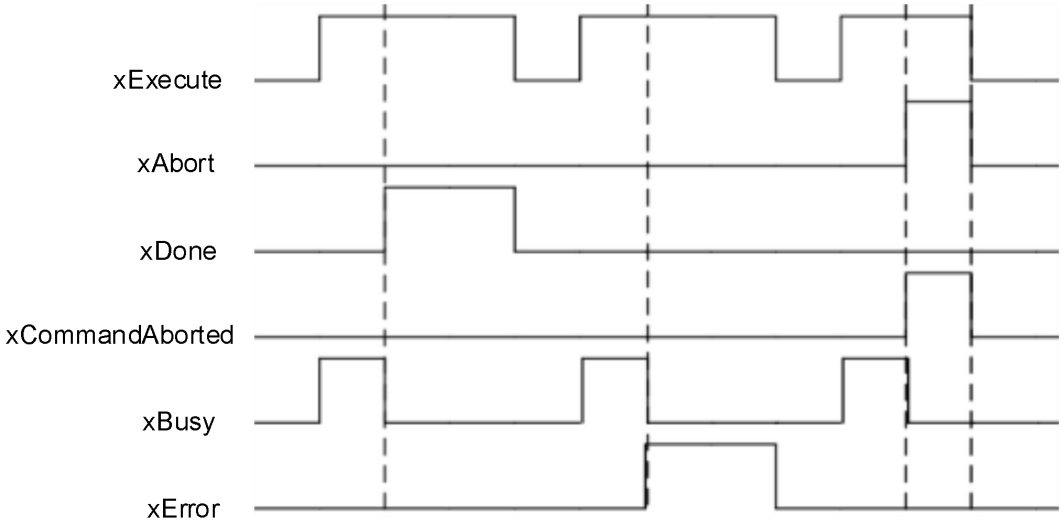
  - 比较输出保持参数支持两种方式。时间方式：当计数值等于设置的比较值时，打开比较输出 OUT 点，保持 udiOutPara *1us 时间后，关闭输出 OUT 点。脉冲方式：当计数值等于设置的比较值时，打开比较输出 OUT 点，保持 udiOutPara *1pulse计数个数后，关闭输出 OUT 点。

  - xOutEnable为使能硬件端口输出功能。FALSE时，功能块正常执行，硬件 OUT 点无

输出。TRUE时，功能块正常执行，且硬件 OUT 点正常输出。

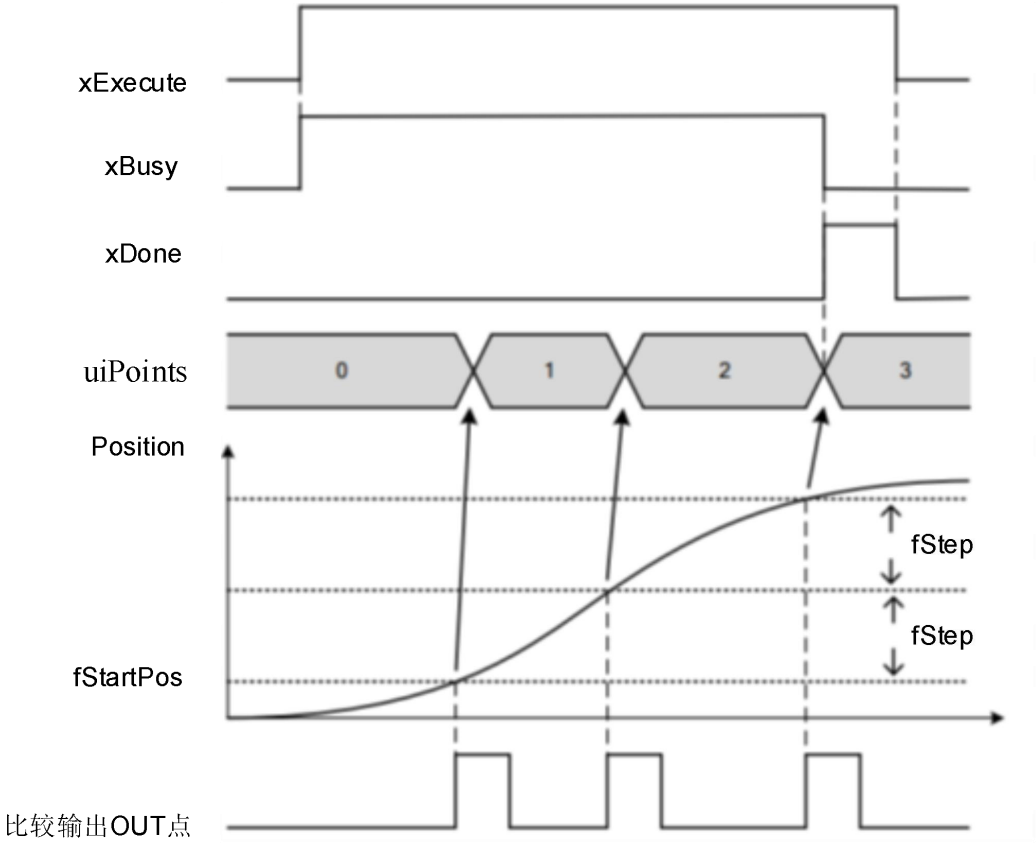
- 相邻 2 点触发间隔的时间小于比较输出保持时间情况下，输出口将连续保持输出状态，本次输出时间直接覆盖上一次输出时间。
- 比较过程中，再次触发 xExecute 上升沿，功能块会从第一个点重新执行比较输出功能。

- **时序图**



- **示例**

fStep 为正值时，比较位置递增，指令时序图如下图所示。fStep 为负值时，与此类似，比较位置递减。



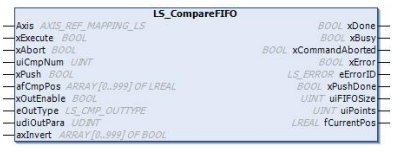
- **注意事项**

➢ 启动高速比较输出功能块时，输出OUT点的初始状态会自动翻转为第一个比较输出点的相反状态。

6.3.7 LS_CompareFIFO

高速计数器队列比较输出。

- **指令格式**

指令	名称	FB/FC	LD表现	ST表现	库
LS_CompareFIFO	队列比较	FB		LS_CompareFIFO(Axis:= , xExecute:= , xAbort:= , uiCmpNum:= , xPush:= , afCmpPos:= , xOutEnable:= , eOutType:= ,	LS_HSIO

				udiOutPara:= , axInvert:= , xDone=> , xBusy=> , xCommandAborted=> , xError=> , eErrorID=> , xPushDone=> , uiFIFOSize=> , uiPoints=> , fCurrentPos=>);	

- **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	初始值	描述
Axis	编码器轴	AXIS_REF_MAPPING_LS	-	-	AXIS_REF_MAPPING_LS 类型编码器轴实例

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
xExecute	执行	BOOL	TRUE FALSE	FALSE	边沿触发，启动功能块
xAbort	中止	BOOL	TRUE FALSE	FALSE	电平触发 TRUE：中止功能块执行
uiCmpNum	比较点个数	UINT	[1,65535]	0	比较点个数
xPush	FIFO 缓存点压入	BOOL	TRUE FALSE	FALSE	压点进 FIFO，支持动态压入
afCmpPos	比较点位置	ARRAY[0..999] OF LREAL	遵循数据范围	0.0	比较点位置集合，unit 单位,最大 1000 点
xOutEnable	使能输出	BOOL	TRUE FALSE	FALSE	使能硬件端口输出功能。使用该功能时，需在参数配置界面配置对应的输出口。 未使能时，功能块正常执行，仅

					硬件端口无输出
eOutType	输出类型	LS_CMP_OUTTPYE	[0,3]	0	0：时间方式-输出FALSE 1：时间方式-输出TRUE 2：脉冲方式-输出FALSE 3：脉冲方式-输出TRUE 4：电平方式
udiOutPara	输出保持参数	UDINT	遵循数据类型	0	输出类型为时间或脉冲方式时，设置该参数。 ① 时间方式：打开输出口保持时间，us单位，[1,20000000] ② 脉冲方式：距离值，pulse单位，脉冲值范围[1~2147483647]
axInvert	电平参数	ARRAY[0..999] OF BOOL	TRUE FALSE	FALSE	输出类型为电平方式时，设置该参数。 每个比较点的输出状态 TRUE：输出有效 FALSE：输出无效

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
xDone	执行完成	BOOL	TRUE FALSE	FALSE	TRUE 比较功能执行成功
xBusy	执行标志	BOOL	TRUE FALSE	FALSE	TRUE 执行比较功能中
xCommandAborted	中止标志	BOOL	TRUE FALSE	FALSE	TRUE功能块执行被中断
xError	错误	BOOL	TRUE FALSE	FALSE	发生异常时为 TRUE
eErrorID	错误代码	LS_ERROR	-	0	发生异常时，输出报错代码。
xPushDone	压入比较点完成	BOOL	TRUE FALSE	FALSE	TRUE：FIFO 缓存压入比较点完成
uiFIFOSize	FIFO 内点数	UINT	[0,128]	0	FIFO 缓存内的点数
uiPoints	已比较点数	UINT	[0,65535]	0	已比较点数
fCurrentPos	当前比较	LREAL	遵循数据范围	0.0	当前比较位置，unit 单位

	位置				

- **功能说明**

  - 本功能块实现计数器当前计数值与数组设置的比较值相等时，触发一个硬件高速输出口，并保持一段时间或一段距离或维持输出状态。

  - 比较功能块使用前，必须在对应计数器界面上配置输出端口，输出端口可配置OUT0~OUT7。

  - 比较输出类型支持三种方式。时间方式：当计数值等于设置的比较值时，打开比较输出 OUT 点，保持 udiOutPara *1us 时间后，关闭输出 OUT 点。脉冲方式：当计数值等于设置的比较值时，打开比较输出 OUT 点，保持 udiOutPara *1pulse计数个数后，关闭输出 OUT 点。电平方式，当计数值等于设置的比较值时，按每个点设置的电平参数axInvert，打开该点比较输出 OUT 点，当axInvert为TRUE时，硬件OUT点有输出，当axInvert为FALSE时，硬件OUT点无输出。

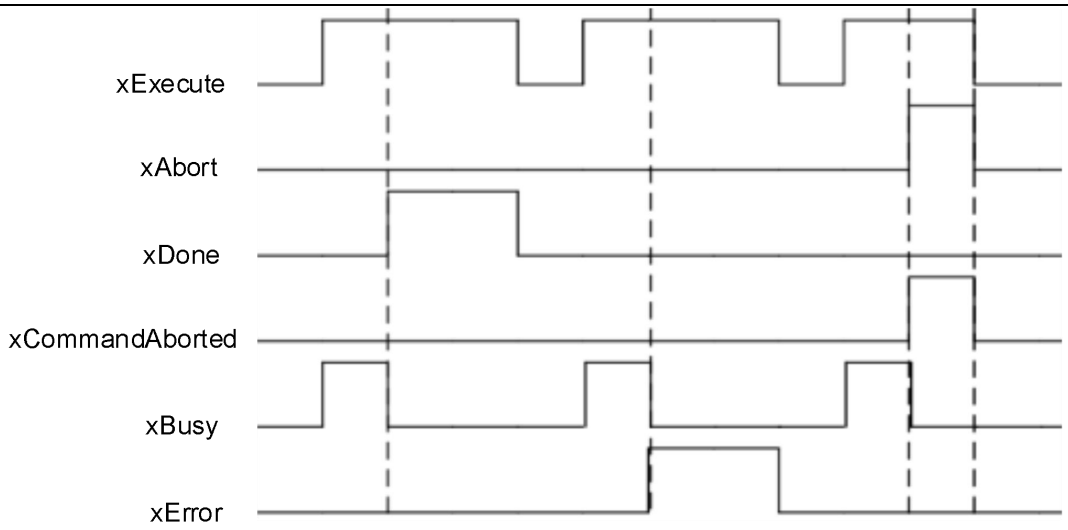
  - xOutEnable为使能硬件端口输出功能。FALSE时，功能块正常执行，硬件 OUT 点无输出。TRUE时，功能块正常执行，且硬件 OUT 点正常输出。

  - 相邻 2 点触发间隔的时间小于比较输出保持时间情况下，输出口将连续保持输出状态，本次输出时间直接覆盖上一次输出时间。

  - 多点比较过程中，再次触发 Execute 上升沿，功能块会从第一个点重新执行比较输出功能。

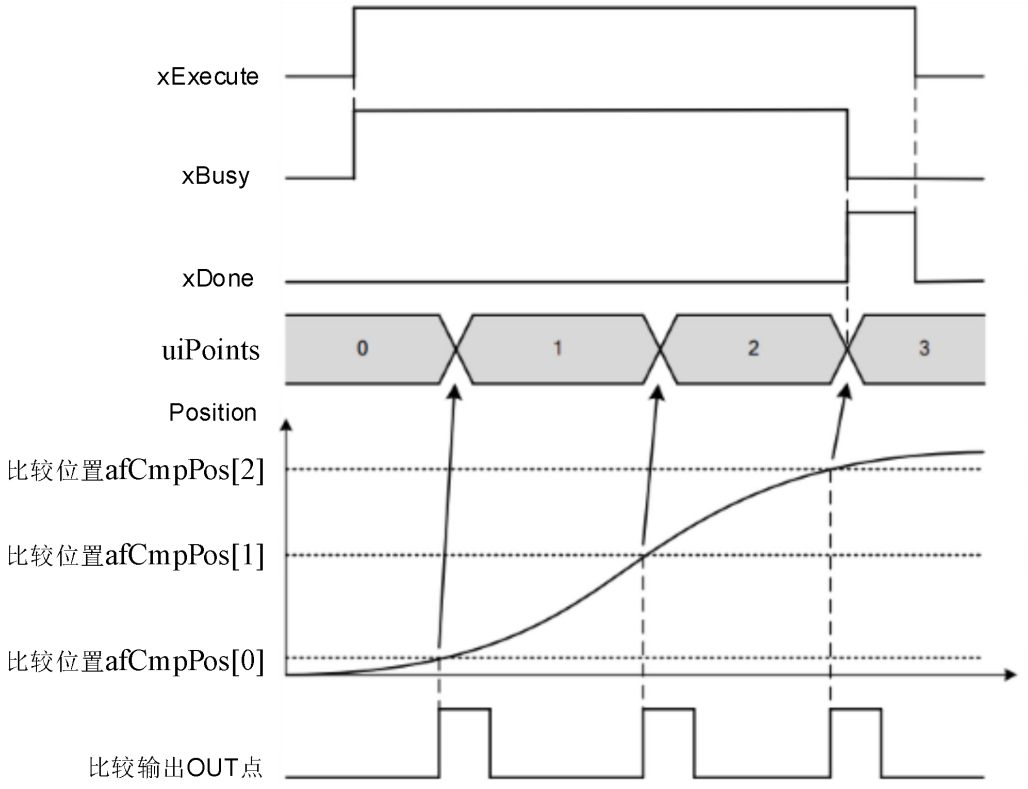
  - 队列比较模式，单次最大添加1000个比较点，低于1000点时，只需在afCmpPos中存入比较点，触发xExecute执行队列比较，无需使用xPush压入缓存数据。当超过1000点时，可在比较过程中，动态添加比较点，通过xPush将比较点数据afCmpPos压入缓存队列，当xPushDone为TRUE时，压入缓存完成。此时，用户可再次压入新的缓存比较点数据。

- **时序图**



- **示例**

队列比较 3 个位置（uiCmpNum =3），指令时序图如下图所示。



- **注意事项**

➢ 启动高速比较输出功能块时，输出OUT点的初始状态会自动翻转为第一个比较输出点的相反状态。

6.3.8 LS_PWM

脉冲宽度调制 PWM 输出。

- **指令格式**

指令	名称	FB/FC	LD表现	ST表现	库
LS_PWM	脉冲宽度调制	FB		LS_PWM(xEnable:= , byIndex:= , diFrequency:= , fRatio:= , xInVelocity=> , xBusy=> , xError=> , eErrorID=>);	LS_HSIO

- **相关变量**

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
xEnable	使能	BOOL	TRUE FALSE	FALSE	功能块使能，电平触发
byIndex	PWM 序号	BYTE	0 - 3	0	PWM 序号(0-3)
diFrequency	频率	DINT	[1,200000]	1	PWM 频率
fRatio	占空比	REAL	[0,1]	0	PWM 占空比

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
xInVelocity	频率值达到 设定值标志	BOOL	TRUE FALSE	FALSE	TRUE脉宽频率达到设定值
xBusy	执行标志	BOOL	TRUE FALSE	FALSE	TRUE 功能块执行中

xError	错误	BOOL	TRUE FALSE	FALSE	发生异常时为 TRUE
eErrorID	错误代码	LS_ERROR	-	0	发生异常时，输出报错代码。

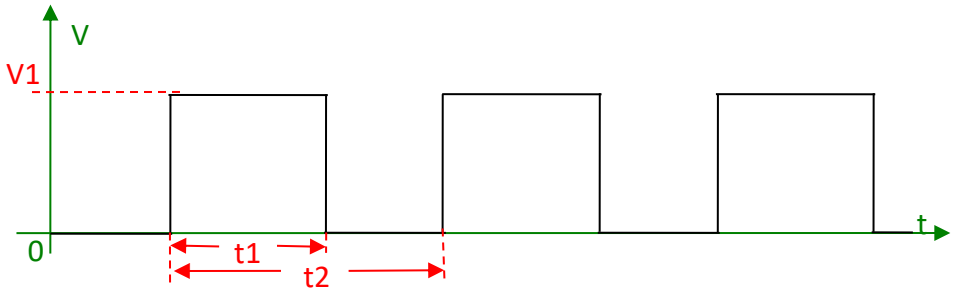
- **功能说明**

  - 本功能块实现高速输出OUT点输出脉宽调制PWM。

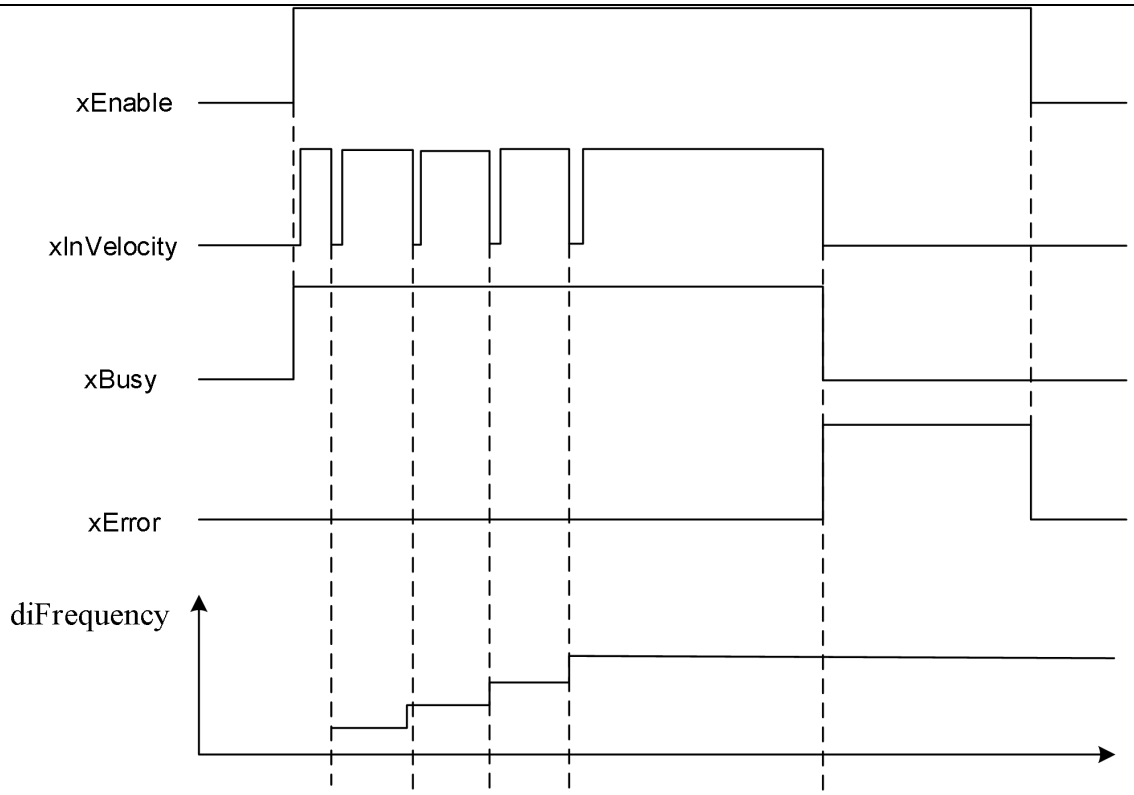
  - 支持4路PWM输出，使用前，必须在对应PWM界面上启用PWM功能，输出点与PWM序号（byIndex）对应关系如下表。

PWM 序号	输出点
PWM0	OUT4
PWM1	OUT5
PWM2	OUT6
PWM3	OUT7

  - 使能状态下（xEnable = TRUE），脉宽调制的占空比和频率可实时修改，输出频率范围：1HZ~200KHZ。占空比（t1/t2）范围：0~1，幅值为V1 = 24V，输出波形如下。



- **时序图**



6.3.9 LS_EnableInterrupt

中断使能。

- **指令格式**

指令	名称	FB/FC	LD表现	ST表现	库
LS_Enable Interrupt	中断使能	FB		LS_EnableInterrupt(xEnable:= , uiExternal:= , uiCompare:= , xValid=> , xBusy=> , xError=> , eErrorID=>);	LS_HSIO

- **相关变量**

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述

xEnable	执行	BOOL	[FALSE,TRUE]	FALSE	使能中断, TRUE：使能，FALSE：无效
uiExternal	打开外部输入中断	UINT	-	0	打开外部输入中断，比如 3，二进制为 2#11，则输入端子 IN0 和 IN1 位被打开
uiCMPNum	打开比较器中断	UINT	-	0	打开比较一致中断，比如 3，二进制为 2#11，则高速计数比较器 Counter0 和 Counter1 被打开

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
xValid	输出有效	BOOL	TRUE FALSE	FALSE	TRUE 功能块输出有效
xBusy	执行标志	BOOL	TRUE FALSE	FALSE	TRUE 功能块执行中
xError	错误	BOOL	TRUE FALSE	FALSE	发生异常时为 TRUE
eErrorID	错误代码	LS_ERROR	–	0	发生异常时，输出报错代码。

- **功能说明**

  - 本功能块用于打开外部输入中断、高速计数器比较一致中断（单点比较）；当发生中断时，执行关联的中断任务。

  - 中断任务需在软件任务界面配置为“外部的”，外部事件选择“IN[0~7]InteruptEvent”或“Counter[0~3]CmpEvent”。当选择外部输入中断“IN[0~7]InteruptEvent”时，还需高速IO通用设置界面指定输入点触发边沿，上升沿、下降沿或任意沿。

7 PMC600 专用指令

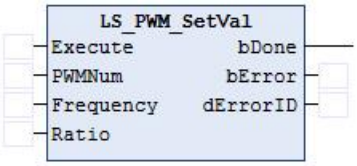
7.1PWM输出指令

7.1.1 指令列表

指令类别	名称	FB/FC	功能
PWM输出指令	LS_PWM_SetVal	FC	PWM输出指令
	LS_PWM_GetVal	FC	PWM参数读取指令

7.1.2 LS_PWM_SetVal

- 指令格式

指令	名称	FB/FC	LD 表现	ST 表现	库
LS_PWM_SetVal	PWM输出指令	FC		LS_PWM_SetVal_0(Execute:= , PWMNum:= , Frequency:= , Ratio:= , bDone=> , bError=> , dErrorID=>);	PMC_Controller库

- 相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Execute	启动	BOOL	FALSE-TRUE	FALSE	启动信号；True-使能功能块，True状态需要保持；
PWMNum	PWM端口号	DINT	0-3	0	PWM通道：0~3
Frequency	频率	DINT	1-500000	1	PWM的频率1~500000Hz
Ratio	占空比	REAL	0-1	0	PWM的占空比0%~100%

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
bDone	返回值	BOOL	FALSE-TRUE	FALSE	FALSE-PWM停止输出； True- PWM正在输出；
bError	错误	BOOL	FALSE-TRUE	FALSE	FALSE-没有错误；True-输入参数越界；
dErrorID	错误代码	DINT	遵照数据类型	0	错误代码

	布尔	位串				整数								实数		时刻、持续时间、日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Execute	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
PWMNum	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-
Frequency	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-
Ratio	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-
bDone	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
bError	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
dErrorID	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-

- **功能说明：**

  - 按设置的PWM端口号、频率及占空比输出PWM脉冲波形。
  - PMC610运动控制器提供了4路PWM（脉冲宽度调制）输出信号，且有光电隔离电路，频率和占空比可调，输出频率范围：1HZ~500KHZ，其输出端口及接线方式见硬件手册。其输出波形如图4. 1所示，PWM波形的周期为t2（频率即为1/t2），占空比为t1/t2，幅值为V1 = 5V，一般用于控制变频器等。

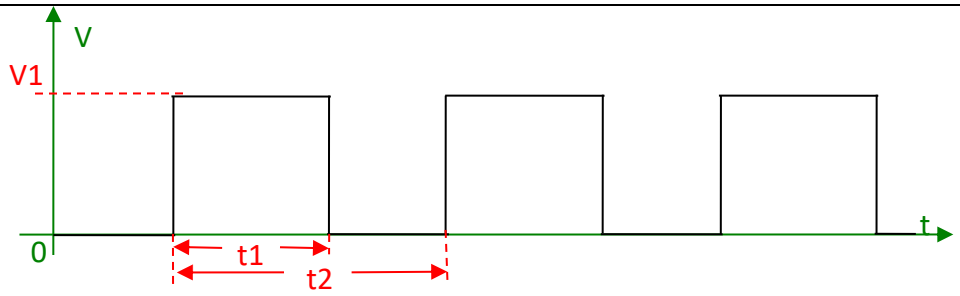


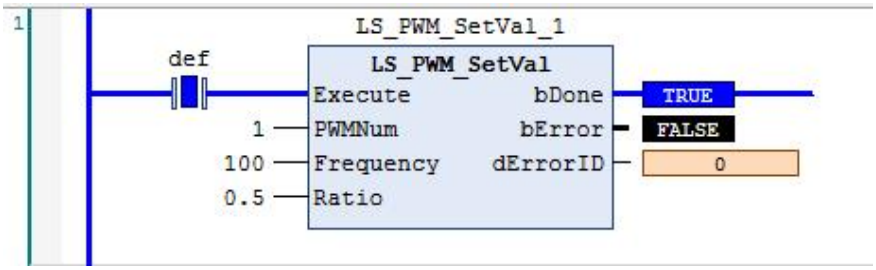
图5.1　PWM输出示意图

- **程序示例**

ST：设置输出PWM的端口号为0，输出频率为10HZ，占空比为30%的脉冲信号。当启动信号abc为True时，端口0输出相应频率，占空比的PWM脉冲波形。

```
LS_PWM_SetVal_0(
    Execute:= abc,
    PWMNum:=0 ,
    Frequency:=10 ,
    Ratio:=0.3 ,
    bDone=> ,
    bError=> ,
    dErrorID=> );RETURN
```

LD：设置输出PWM的端口号为1，输出频率为100HZ，占空比为50%的脉冲信号。当启动信号def为True时，端口1输出相应频率，占空比的PWM脉冲波形。



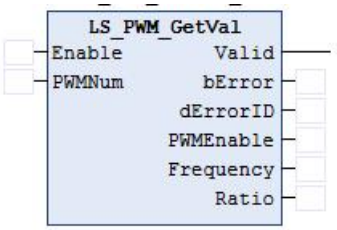
- **注意事项**

➢ 当使用对应的PWM端口输出脉冲波形时，需先在硬件端口配置中将对应端口的功能勾选上PWM输出。

➢ 对应PWM端口的频率，占空比可用LS_PWM_GetVal指令读取。

7.1.3 LS_PWM_GetVal

- 指令格式

指令	名称	FB/FC	LD 表现	ST 表现	库
LS_PWM_GetVal	PWM参数读取指令	FC		LS_PWM_GetVal_0(Enable:= , PWMNum:= , Valid=> , bError=> , dErrorID=> , PWMEnable=> , Frequency=> , Ratio=>);	PMC_Controller库

- 相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Enable	启动	BOOL	FALSE-TRUE	FALSE	启动信号；True-使能功能块，True状态需要保持；
PWMNum	PWM端口号	DINT	0-3	0	PWM通道：0~3

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
Valid	返回值	BOOL	FALSE-TRUE	FALSE	True-读取的PWM状态和参数有效；
bError	错误	BOOL	FALSE-TRUE	FALSE	FALSE-没有错误；
dErrorID	错误代码	DINT	遵照数据类型	0	错误代码
PWMEnable	使能状态	BOOL	FALSE-TRUE	FALSE	PWM端口的使能状态；FALSE-禁止PWM输出；True-使能PWM输出
Frequency	频率	DINT	1-500000	1	PWM的频率1~500000Hz
Ratio	占空比	REAL	0-1	0	PWM的占空比0%~100%

	布尔	位串				整数								实数		时刻、持续时间、日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Enable	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
PWMNum	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-
Valid	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
bError	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
dErrorID	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-
PWMEnable	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Frequency	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-
Ratio	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-

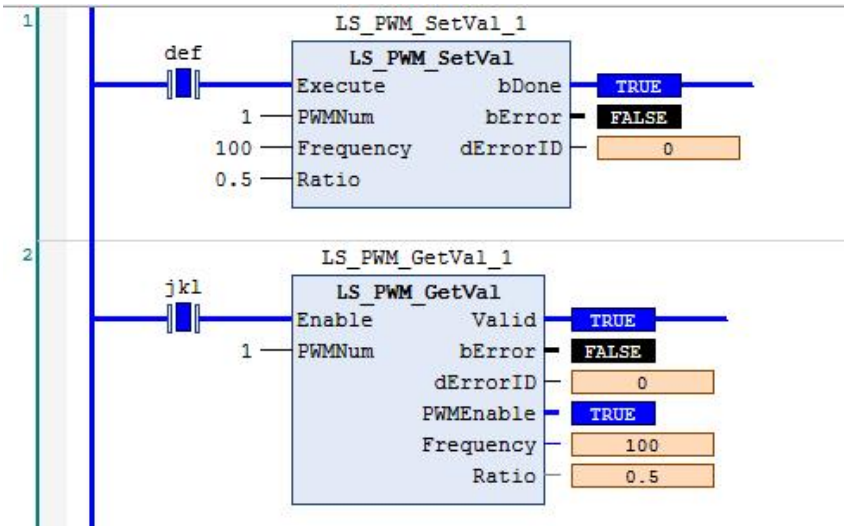
- **功能说明：**
  - 该功能块用作回读指定端口设置的PWM参数信息。
  - 当Enable变为TRUE时，能够获取对应PWM端口的使能状态、频率和占空比等参数。
- **程序示例**

ST：设置读取PWM的端口号为0。当启动信号ghi为True时，输出PWM端口0的使能状态、频率和占空比等参数。

```
LS_PWM_SetVal_0(
    Execute TRUE := abc TRUE,
    PWMNum 0 :=0 ,
    Frequency 10 :=10 ,
    Ratio 0.3 :=0.3 ,
    bDone=> ,
    bError=> ,
    dErrorID=> );

LS_PWM_GetVal_0(
    Enable TRUE :=ghi TRUE ,
    PWMNum 0 := 0,
    Valid TRUE =>V1 TRUE ,
    bError FALSE =>B1 FALSE ,
    dErrorID 0 =>D1 0 ,
    PWMEnable TRUE =>PW1 TRUE ,
    Frequency 10 =>F1 10 ,
    Ratio 0.3 =>R1 0.3 );RETURN
```

LD：设置读取PWM的端口号为1。当启动信号jkl为True时，输出PWM端口1的使能状态、频率和占空比等参数。



- **注意事项**

➢ Enable变为True时会读取对应PWM端口的使能状态,频率，占空比；Enable变为FALSE时会保持最后读取到的PWM端口使能状态,频率，占空比。

➢ 如果没有先执行LS_PWM_SetVal指令激活指定端口输出PWM波形，LS_PWM_GetVal读取该端口参数时PWMEnable为FALSE，即PWM端口没有脉冲波形输出。

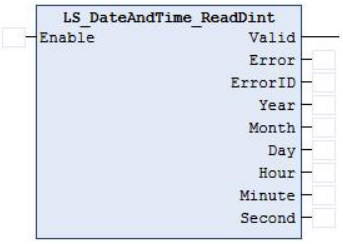
7.2 系统时间指令

7.2.1 指令列表

指令类别	名称	FB/FC	功能
系统时间指令	LS_DateAndTime_ReadDint	FB	读取系统时间指令
	LS_DateAndTime_ReadString	FB	读取系统时间指令
	LS_DateAndTime_SetDint	FB	设置系统时间指令
	LS_DateAndTime_SetString	FB	设置系统时间指令

7.2.2 LS_DateAndTime_ReadDint

- 指令格式

指令	名称	FB/FC	LD 表现	ST 表现	库
LS_DateAndTime_ReadDint	读取系统时间指令	FB		LS_DateAndTime_ReadDint_0(Enable:= , Valid=> , Error=> , ErrorID=> , Year=> , Month=> , Day=> , Hour=> , Minute=> , Second=>);	PMC_Controller库

- 相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Enable	启动	BOOL	FALSE-TRUE	FALSE	启动信号；True-使能功能块，True状态需要保持；

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
Valid	返回值	BOOL	FALSE-TRUE	FALSE	True-读取的系统时间参数有

					效；
Error	错误	BOOL	FALSE-TRUE	FALSE	错误状态；FALSE-没有错误；
ErrorID	错误代码	DINT	遵照数据类型	0	错误代码
Year	年	UINT	遵照数据类型	0	年
Month	月	UINT	1-12	0	月
Day	日	UINT	1-31	0	日
Hour	时	UINT	0-23	0	时
Minute	分	UINT	0-59	0	分
Second	秒	UINT	0-59	0	秒

	布尔	位串				整数								实数		时刻、持续时间、日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Enable	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Valid	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Error	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
ErrorID	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-
Year	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-
Month	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-
Day	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-
Hour	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-
Minute	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-
Second	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-

- **功能说明：**

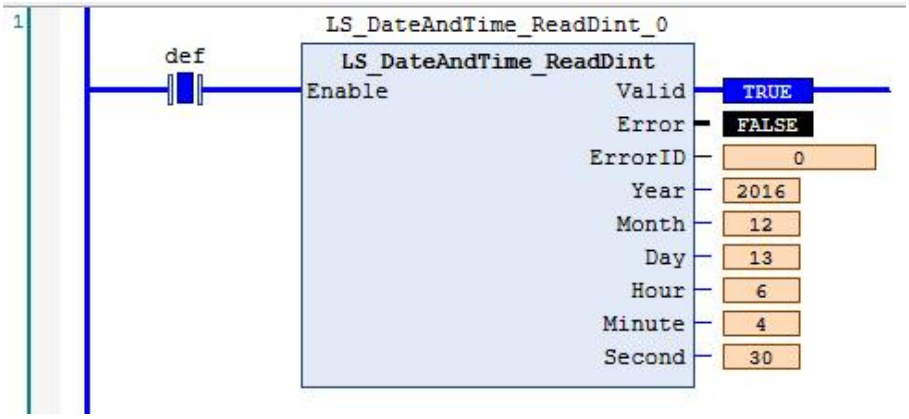
  - 读取控制器系统时间，将年月日时分秒拆分读取。

- **程序示例**

ST：当启动信号abc为True时，读取控制器系统时间。

```
LS_DateAndTime_ReadDint_0(
    Enable TRUE :=abc TRUE ,
    Valid TRUE =>V1 TRUE ,
    Error FALSE => E1 FALSE,
    ErrorID 0 => E2 0,
    Year 2016 => Y1 2016,
    Month 12 =>MON1 12 ,
    Day 13 => D1 13,
    Hour 6 => H1 6,
    Minute 7 => MIN1 7,
    Second 31 =>SE1 31 );RETURN
```

LD：当启动信号def为True时，读取控制器系统时间。



- **注意事项**

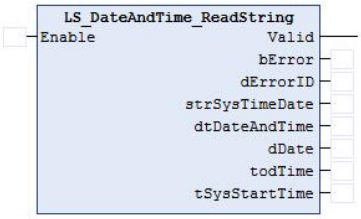
➢ 系统时间需在有电池供电情况下断电才不会丢失。使用该功能时只需直接调用该指令即可，注意指令中参数的数据类型。

➢ Enable为True时实时读取系统时间，需要保持True状态；Enable变为FALSE时将保持最后读取到的系统时间参数。

7.2.3 LS_DateAndTime_ReadString

- **指令格式**

指令	名称	FB/FC	LD 表现	ST 表现	库

LS_DateAndTime_ReadString	读取系统时间指令	FB		LS_DateAndTime_ReadString_0(Enable:=, Valid=> , bError=> , dErrorID=> , strSysTimeDate=> , dtDateAndTime=> , dDate=> , todTime=> , tSysStartTime=>);	PMC_Controller库

- **相关变量**

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Enable	启动	BOOL	FALSE-TRUE	FALSE	启动信号；True-使能功能块，True状态需要保持；

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
Valid	返回值	BOOL	FALSE-TRUE	FALSE	True-读取的系统时间参数有效；
bError	错误	BOOL	FALSE-TRUE	FALSE	错误状态；FALSE-没有错误；
dErrorID	错误代码	DINT	遵照数据类型	0	错误代码
strSysTimeDate	控制器日期时间	SYSTIME DATE	遵照数据类型	-	控制器日期时间
dtDateAndTime	日期和时间	DATE_AND_TIME	遵照数据类型	DT#1970-1-1-0:0:0	日期和时间
dDate	日期	DATE	遵照数据类型	D#1970-1-1	日期
todTime	时间	TOD	遵照数据类型	TOD#0:0:0	时间
tSysStartTime	开机时间	TIME	遵照数据类型	T#0ms	控制器开机时间

	布尔	位串				整数								实数		时刻、持续时间、日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Enable	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Valid	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
bError	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
dErrorID	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-
strSysTimeDate	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
dtDateAndTime	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
dDate	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-
todTime	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-
tSysStartTime	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-

- **功能说明：**

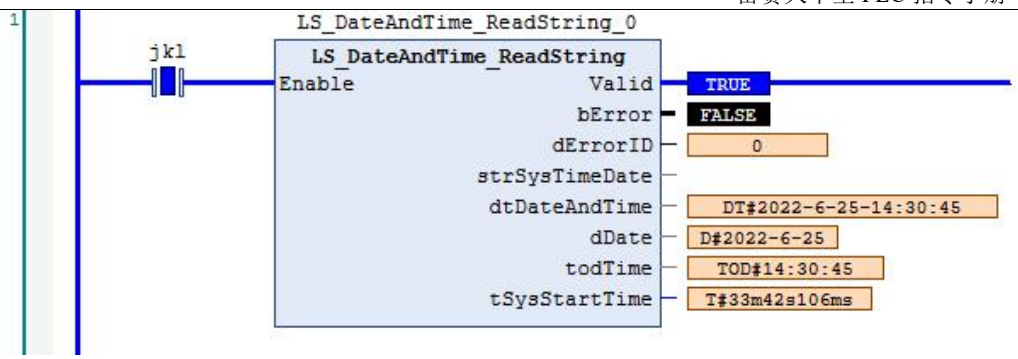
  - 读取控制器系统时间，将日期时间直接读取。

- **程序示例**

ST：当启动信号ghi为True时，读取控制器系统时间。

```
1  LS_DateAndTime_ReadString_0(
2      Enable TRUE := ghi TRUE,
3      Valid TRUE => V1 TRUE,
4      bError FALSE =>B1 FALSE ,
5      dErrorID 0 =>D1 0 ,
6      strSysTimeDate=>S1 ,
7      dtDateAndTime DT#2022-6-25-14:29:20 =>DT1 DT#2022-6-25-14:29:20 ,
8      dDate D#2022-6-25 =>DD1 D#2022-6-25 ,
9      todTime TOD#14:29:20 => TO1 TOD#14:29:20 ,
10     tSysStartTime T#32m17s224ms => TS1 T#32m17s224ms );RETURN
```

LD：当启动信号jkl为True时，读取控制器系统时间。



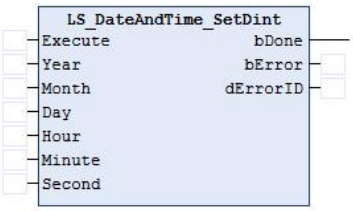
- **注意事项**

➢ 系统时间需在有电池供电情况下断电才不会丢失。使用该功能时只需直接调用该指令即可，注意指令中参数的数据类型。

➢ Enable为True时实时读取系统时间，需要保持True状态；Enable变为FALSE时将保持最后读取到的系统时间参数。

7.2.4 LS_DateAndTime_SetDint

- **指令格式**

指令	名称	FB/FC	LD 表现	ST 表现	库
LS_DateAndTime_SetDint	设置系统时间指令	FB		LS_DateAndTime_SetDint_0(Execute:= , Year:= , Month:= , Day:= , Hour:= , Minute:= , Second:= , bDone=> , bError=> , dErrorID=>);	PMC_Controller库

- **相关变量**

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述

Execute	启动	BOOL	FALSE-TRUE	FALSE	启动信号，TRUE时使能功能块，上升沿触发；
Year	年	UINT	遵照数据类型	0	年
Month	月	UINT	1-12	0	月
Day	日	UINT	1-31	0	日
Hour	时	UINT	0-23	0	时
Minute	分	UINT	0-59	0	分
Second	秒	UINT	0-59	0	秒

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
bDone	返回值	BOOL	FALSE-TRUE	FALSE	完成信号；FALSE-没有完成；
bError	错误	BOOL	FALSE-TRUE	FALSE	错误状态；FALSE-没有错误；
dErrorID	错误代码	DINT	遵照数据类型	0	错误代码

	布尔	位串				整数								实数		时刻、持续时间、日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Execute	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Year	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-
Month	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-
Day	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-
Hour	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-
Minute	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-
Second	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-
bDone	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
bError	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
dErrorID	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-

- **功能说明：**

- 按年月日时分秒字段设置系统时间。

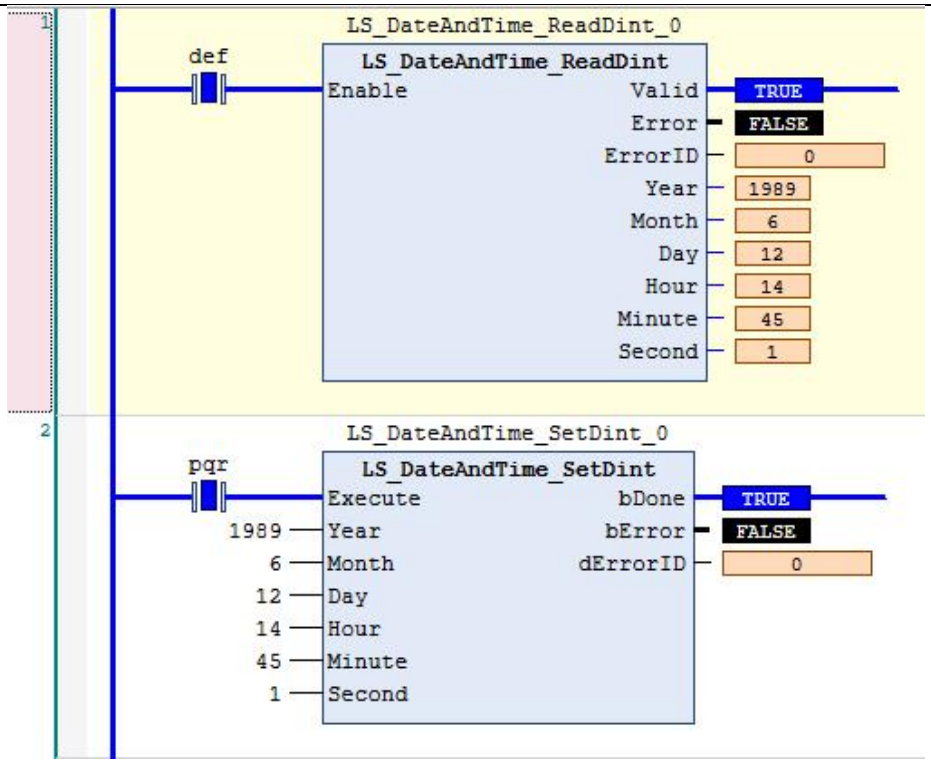
- **程序示例**

ST：当启动信号mno变为True时，将预设的年月日时分秒写进控制器系统。

```
LS_DateAndTime_ReadDint_0(
    Enable TRUE :=abc TRUE ,
    Valid TRUE =>V1 TRUE ,
    Error FALSE => E1 FALSE,
    ErrorID 0 => E2 0,
    Year 2036 => Y1 2036,
    Month 12 =>MON1 12 ,
    Day 14 => D1 14,
    Hour 5 => H1 5,
    Minute 45 => MIN1 45,
    Second 45 =>SE1 45 );

    LS_DateAndTime_SetDint_0(
    Execute TRUE :=mno TRUE ,
    Year 2036 :=2036 ,
    Month 12 := 12,
    Day 14 :=14 ,
    Hour 5 := 5,
    Minute 45 :=45 ,
    Second 45 :=45 ,
    bDone=> ,
    bError=> ,
    dErrorID=> );RETURN
```

LD：当启动信号pqr变为True时，将预设的年月日时分秒写进控制器系统。



- **注意事项**

➢ 系统时间需在有电池供电情况下断电才不会丢失。使用该功能时只需直接调用该指令即可，注意指令中参数的数据类型。

➢ Execute上升沿时写入时间参数，注意指令中参数的数据类型以及数据范围，否则写入参数失败，bError会变True报错，此时bDone为False。

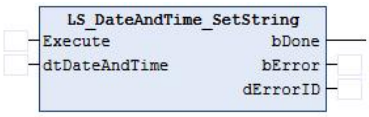
➢ PLC本身存在一定的日期限制，写入系统日期时请输入与当前世界时间相近的日期；例如当前是2022年，请输入日期范围在1970年-2036年以内，否则写入日期时报错，bError变为True，此时bDone为FALSE。

➢ 请注意在线仿真时运行该指令会改变电脑的系统时间，可能导致电脑系统崩溃。

7.2.5 LS_DateAndTime_SetString

- **指令格式**

指令	名称	FB/FC	LD 表现	ST 表现	库

LS_DateAndTime_SetString	设置系统时间指令	FB		LS_DateAndTime_SetString_0(Execute:= , dtDateAndTime:= , bDone=> , bError=> , dErrorID=>);	PMC_Controller库

- **相关变量**

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Execute	启动	BOOL	FALSE-TRUE	FALSE	启动信号；TRUE 时使能功能块，上升沿触发；
dtDateAndTime	日期和时间	DATE_AND_TIME	遵照数据类型	DT#1970-1-1-0:0:0	日期和时间

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
bDone	返回值	BOOL	FALSE-TRUE	FALSE	完成信号；FALSE-没有完成；
bError	错误	BOOL	FALSE-TRUE	FALSE	错误状态；FALSE-没有错误；
dErrorID	错误代码	DINT	遵照数据类型	0	错误代码

	布尔	位串				整数								实数		时刻、持续时间、日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Execute	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
dtDateAndTime	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
bDone	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
bError	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
dErrorID	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-

- **功能说明：**

  - 按预设的日期时间设置系统时间。

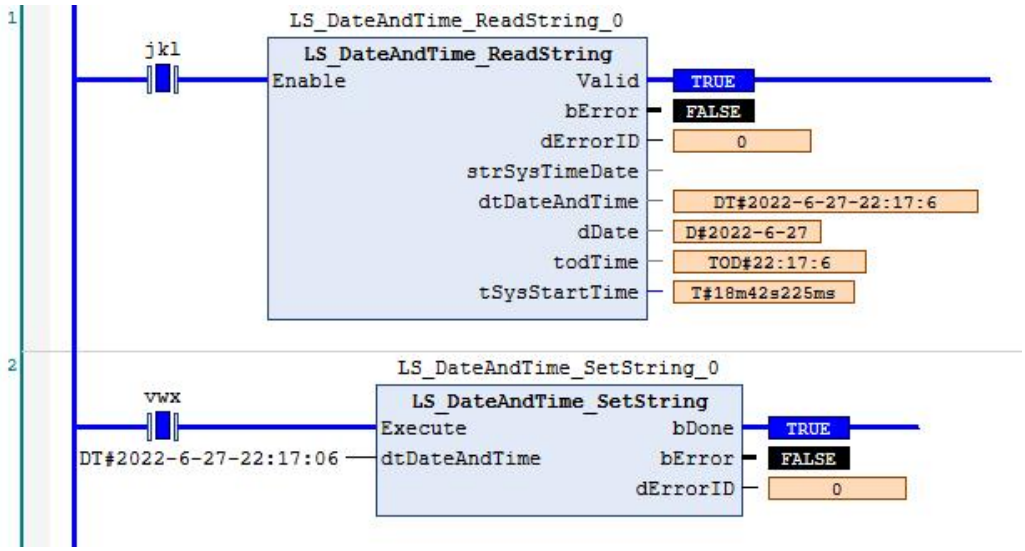
- **程序示例**

ST：当启动信号stu变为True时，将预设的日期时间写进控制器系统。

```
LS_DateAndTime_ReadString_0(
    Enable:= ghi,
    Valid=> V1,
    bError=>B1 ,
    dErrorID 0 =>D1 0 ,
    strSysTimeDate=>S1 ,
    dtDateAndTime DT#2022-6-27-10:30:6 =>DT1 DT#2022-6-27-10:30:6 ,
    dDate D#2022-6-27 =>DD1 D#2022-6-27 ,
    todTime TOD#10:30:6 => TO1 TOD#10:30:6 ,
    tSysStartTime T#17m11s969ms => TS1 T#17m11s969ms );

    LS_DateAndTime_SetString_0(
    Execute :=stu ,
    dtDateAndTime DT#2022-6-27-10:30:6 :=DT#2022-6-27-10:30:06 ,
    bDone=> ,
    bError=> ,
    dErrorID=> );RETURN
```

LD：当启动信号vwx变为True时，将预设的日期时间写进控制器系统。



- **注意事项**

- 系统时间需在有电池供电情况下断电才不会丢失。使用该功能时只需直接调用该指令即可，注意指令中参数的数据类型。

- Execute上升沿时写入时间参数，注意指令中参数的数据类型以及数据范围，否则写入参

数失败，bError会变True报错，此时bDone为False。

- PLC本身存在一定的日期限制，写入系统日期时请输入与当前世界时间相近的日期；例如当前是2022年，请输入日期范围在1970年-2036年以内，否则写入日期时报错，bError变为True，此时bDone为FALSE。
- 请注意在线仿真时运行该指令会改变电脑的系统时间，可能导致电脑系统崩溃。

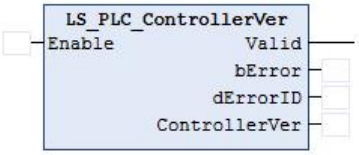
7.3固件版本指令

7.3.1 指令列表

指令类别	名称	FB/FC	功能
固件版本指令	LS_PLC_ControllerVer	FC	读取硬件版本指令
	LS_PLC_CPUID	FC	读取CPU ID号指令
	LS_PLC_HardwareVerInformation	FC	读取固件FPGA版本指令
	LS_PLC_SoftwareVerInformation	FC	读取固件ARM版本指令
	LS_PLC_VerInformation	FC	读取控制器的所有版本信息指令

7.3.2 LS_PLC_ControllerVer

- **指令格式**

指令	名称	FB/FC	LD 表现	ST 表现	库
LS_PLC_ControllerVer	读取硬件版本指令	FC		LS_PLC_ControllerVer(Enable:= , Valid=> , bError=> , dErrorID=> , ControllerVer=>);	PMC_Controller库

- **相关变量**

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Enable	启动	BOOL	FALSE-TRUE	FALSE	启动信号；True-使能功能块，True状态需要保持；

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
Valid	返回值	BOOL	FALSE-TRUE	FALSE	True-读取的硬件版本参数有效；
bError	错误	BOOL	FALSE-TRUE	FALSE	错误状态；FALSE-没有错误；
dErrorID	错误代	DINT	遵照数据类型	0	错误代码

	码				
ControllerVer	控制器硬件版本	DINT	遵照数据类型	0	控制器硬件版本

	布尔	位串				整数								实数		时刻、持续时间、日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Enable	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Valid	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
bError	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
dErrorID	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-
Controller Ver	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-

- **功能说明：**

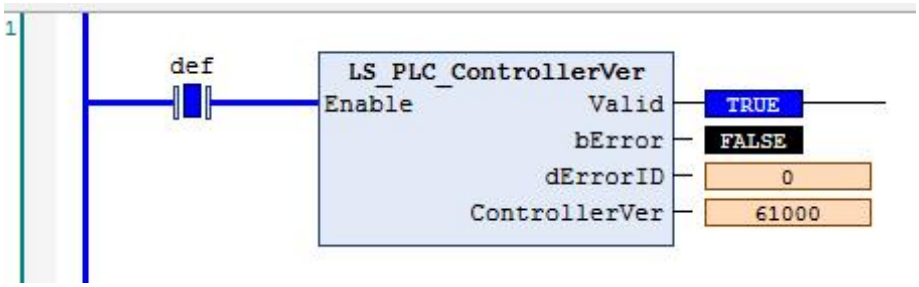
  - 该函数获取控制器的硬件版本信息。

- **程序示例**

ST：当启动信号abc变为True时，读取控制器的硬件版本信息。

```
1  LS_PLC_ControllerVer(
2      Enable FALSE :=abc TRUE ,
3      Valid FALSE =>V1 TRUE ,
4      bError FALSE => B1 FALSE ,
5      dErrorID 0 => D1 0 ,
6      ControllerVer 61000 =>CON1 61000 );RETURN
```

LD：当启动信号def变为True时，读取控制器的硬件版本信息。

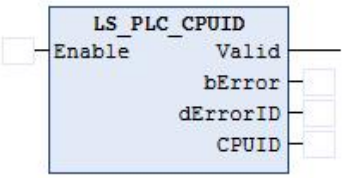


- **注意事项**

➢ Enable需要保持True状态，Enable变为False时将保持最后读取到的硬件版本信息。

7.3.3 LS_PLC_CPUID

- **指令格式**

指令	名称	FB/FC	LD 表现	ST 表现	库
LS_PLC_CPUID	读取CPU ID号指令	FC		LS_PLC_CPUID (Enable:= , Valid=> , bError=> , dErrorID=> , CPUID=>);	PMC_Controller库

- **相关变量**

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Enable	启动	BOOL	FALSE-TRUE	FALSE	启动信号；True-使能功能块，True状态需要保持；

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
Valid	返回值	BOOL	FALSE-TRUE	FALSE	True-读取的CPU ID号参数有效；
bError	错误	BOOL	FALSE-TRUE	FALSE	错误状态；FALSE-没有错误；
dErrorID	错误代码	DINT	遵照数据类型	0	错误代码
CPUID	CPU ID号	UDINT	遵照数据类型	0	控制器CPU ID号

	布尔	位串	整数	实数	时刻、持续时间、日期、字符串

	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Enable	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Valid	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
bError	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
dErrorID	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-
CPUID	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-

- **功能说明：**

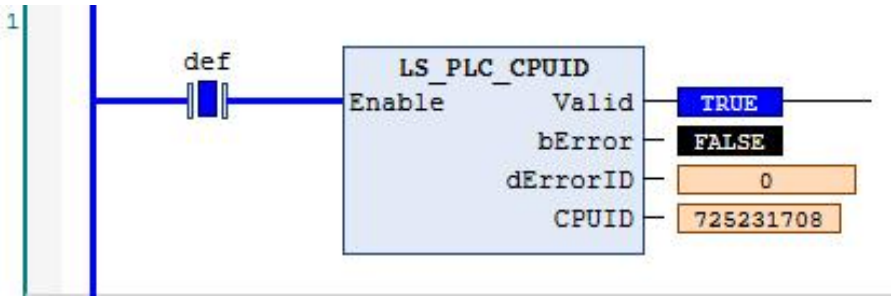
  - 该函数获取控制器的CPU ID号。

- **程序示例**

ST：当启动信号abc变为True时，读取控制器的CPU ID号。

```
1  LS_PLC_CPUID(
2       Enable TRUE := abc TRUE,
3       Valid TRUE =>vl TRUE ,
4       bError FALSE => bl FALSE,
5       dErrorID 0 =>dl 0 ,
6       CPUID 725231708 =>cpl 725231708 );RETURN
```

LD：当启动信号def变为True时，读取控制器的CPU ID号。



- **注意事项**

  - Enable需要保持True状态，Enable变为False时将保持最后读取到的CPU ID号。

7.3.4 LS_PLC_HardwareVerInformation

- **指令格式**

指令	名称	FB/FC	LD 表现	ST 表现	库
LS_PLC_HardwareVerInformation	读取固件FPGA版本指令	FC		LS_PLC_HardwareVerInformation(Enable:= , Valid=> , bError=> , dErrorID=> , HardWareVer=>);	PMC_Controller库

- **相关变量**

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Enable	启动	BOOL	FALSE-TRUE	FALSE	启动信号；True-使能功能块，True状态需要保持；

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
Valid	返回值	BOOL	FALSE-TRUE	FALSE	True-读取的固件FPGA版本参数有效；
bError	错误	BOOL	FALSE-TRUE	FALSE	错误状态；FALSE-没有错误；
dErrorID	错误代码	DINT	遵照数据类型	0	错误代码
HardWareVer	固件FPGA版本	DINT	遵照数据类型	0	控制器固件FPGA版本

	布尔	位串				整数								实数		时刻、持续时间、日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Enable	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Valid	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
bError	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

dErrorID	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-
HardWare Ver	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-

- **功能说明：**

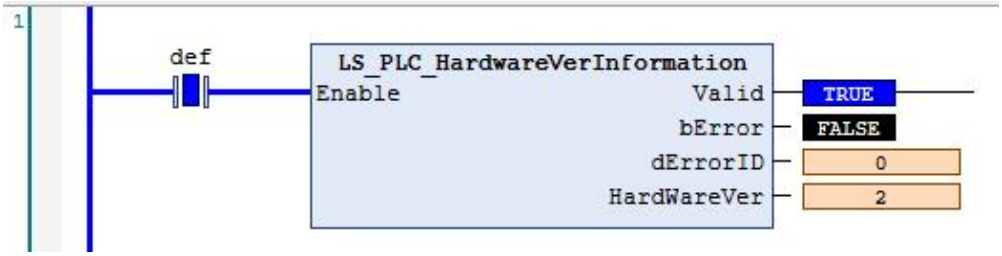
  - 该函数获取控制器的固件FPGA版本信息。

- **程序示例**

ST：当启动信号abc变为True时，读取控制器的固件FPGA版本信息。

```
LS_PLC_HardwareVerInformation(
    Enable TRUE :=abc TRUE ,
    Valid TRUE =>vl TRUE ,
    bError FALSE =>bl FALSE ,
    dErrorID 0 => dl 0 ,
    HardWareVer 2 =>hal 2 );RETURN
```

LD：当启动信号def变为True时，读取控制器的固件FPGA版本信息。



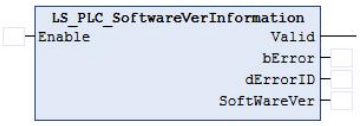
- **注意事项**

  - Enable需要保持True状态，Enable变为False时将保持最后读取到的固件FPGA版本信息。

7.3.5 LS_PLC_SoftwareVerInformation

- **指令格式**

指令	名称	FB/ FC	LD 表现	ST 表现	库

LS_PLC_SoftwareVerInformation	读取固件ARM版本指令	FC		LS_PLC_SoftwareVerInformation(Enable:= , Valid=> , bError=> , dErrorID=> , SoftWareVer=>);	PMC_Controller库

- **相关变量**

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Enable	启动	BOOL	FALSE-TRUE	FALSE	启动信号；True-使能功能块，True状态需要保持；

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
Valid	返回值	BOOL	FALSE-TRUE	FALSE	True-读取的固件ARM版本参数有效；
bError	错误	BOOL	FALSE-TRUE	FALSE	错误状态；FALSE-没有错误；
dErrorID	错误代码	DINT	遵照数据类型	0	错误代码
SoftWareVer	控制器固件ARM版本	DINT	遵照数据类型	0	控制器固件ARM版本

	布尔	位串				整数								实数		时刻、持续时间、日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Enable	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Valid	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
bError	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
dErrorID	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-

SoftWare Ver	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-

- **功能说明：**

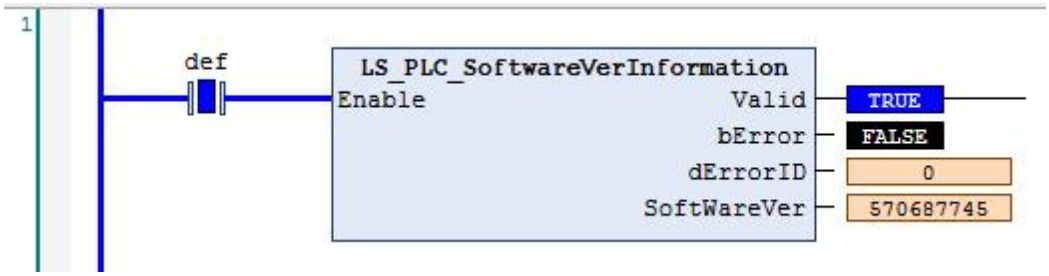
  - 该函数获取控制器的固件ARM版本信息。

- **程序示例**

ST：当启动信号abc变为True时，读取控制器的固件ARM版本信息。

```
LS_PLC_SoftwareVerInformation(
    Enable TRUE :=abc TRUE ,
    Valid TRUE =>vl TRUE ,
    bError FALSE =>bl FALSE ,
    dErrorID 0 => dl 0 ,
    SoftWareVer 570687745 => sol 570687745 );RETURN
```

LD：当启动信号def变为True时，读取控制器的固件ARM版本信息。



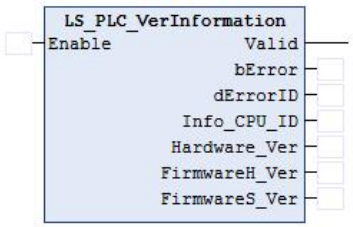
- **注意事项**

  - Enable需要保持True状态，Enable变为False时将保持最后读取到的固件ARM版本信息。

7.3.6 LS_PLC_VerInformation

- **指令格式**

指令	名称	FB/ FC	LD 表现	ST 表现	库

LS_PLC_VerInformation	读取控制器的所有版本信息指令	FC		LS_PLC_VerInformation(Enable:= , Valid=> , bError=> , dErrorID=> , Info_CPU_ID=> , Hardware_Ver=> , FirmwareH_Ver=> , FirmwareS_Ver=>);	PMC_Controller库

- **相关变量**

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Enable	启动	BOOL	FALSE-TRUE	FALSE	启动信号；True-使能功能块，True状态需要保持；

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
Valid	返回值	BOOL	FALSE-TRUE	FALSE	True-读取的版本信息有效；
bError	错误	BOOL	FALSE-TRUE	FALSE	错误状态；FALSE-没有错误；
dErrorID	错误代码	DINT	遵照数据类型	0	错误代码
Info_CPU_ID	CPU ID号	UDINT	遵照数据类型	0	控制器CPU ID号
Hardware_Ver	硬件版本	DINT	遵照数据类型	0	控制器硬件版本
FirmwareH_Ver	固件FPGA版本	DINT	遵照数据类型	0	控制器固件FPGA版本
FirmwareS_Ver	固件ARM版本	DINT	遵照数据类型	0	控制器固件ARM版本

	布尔	位串				整数								实数		时刻、持续时间、日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Enable	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Valid	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
bError	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
dErrorID	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-
Info_CPU_ID	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-
Hardware_Ver	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-
FirmwareH_Ver	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-
FirmwareS_Ver	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-

- **功能说明：**

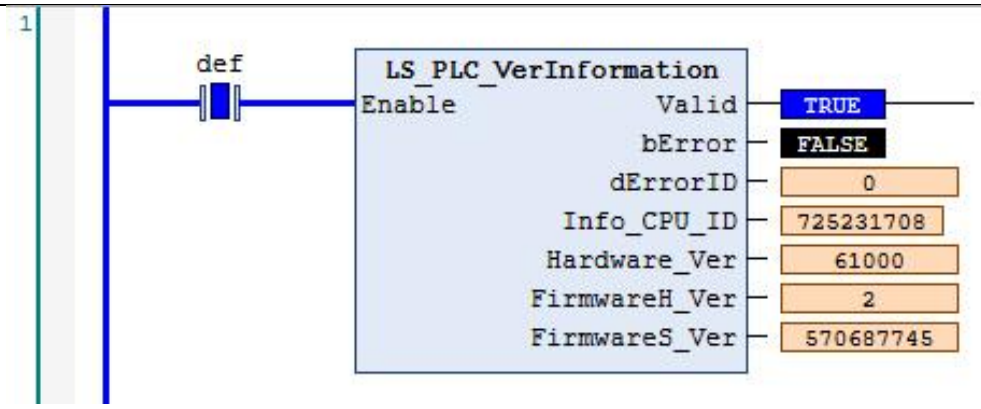
  - 该函数获取控制器的所有版本信息，包括ID号、硬件、固件等信息。

- **程序示例**

ST：当启动信号abc变为True时，读取控制器的所有版本信息。

```
LS_PLC_VerInformation(
    Enable TRUE :=abc TRUE ,
    Valid TRUE => vl TRUE,
    bError FALSE =>bl FALSE ,
    dErrorID 0 => dl 0,
    Info_CPU_ID 725231708 => cpl 725231708,
    Hardware_Ver 61000 =>hal 61000 ,
    FirmwareH_Ver 2 =>hl 2 ,
    FirmwareS_Ver 570687745 =>sl 570687745 );RETURN
```

LD：当启动信号def变为True时，读取控制器的所有版本信息。



- **注意事项**

➢ Enable需要保持True状态，Enable变为False时将保持最后读取到的控制器的所有版本信息。

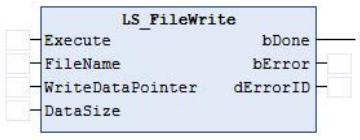
7.4 UsrData内文件操作指令

7.4.1 指令列表

指令类别	名称	FB/FC	功能
UsrData内文件操作指令	LS_FileWrite	FB	文件写入指令
	LS_FileWriteAppend	FB	文件追加写入指令
	LS_FileRead	FB	文件读取指令
	LS_FileCopy	FB	文件拷贝指令
	LS_FileRename	FB	文件重命名指令
	LS_FileDelete	FB	文件删除指令
	LS_FileDeleteAll	FB	文件批量删除指令
	LS_GetDirectoryFile	FB	获取目录内文件信息指令
	LS_GetFileInformation	FB	获取指定文件信息指令
	LS_UDisk_CopyFromUDisk	FB	从U盘拷贝文件指令
	LS_UDisk_CopyToUDisk	FB	拷贝文件至U盘指令
	LS_UDisk_GetDirectoyFile	FB	获取U盘根目录内文件信息指令

7.4.2 LS_FileWrite

- **指令格式**

指令	名称	FB/FC	LD 表现	ST 表现	库
LS_FileWrite	文件写入指令	FB		LS_FileWrite_0(Execute:=, FileName:=, WriteDataPointer:=, DataSize:=, bDone=>, bError=>,	PMC_FileManage库

				dErrorID=>);	

- **相关变量**

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Execute	启动	BOOL	FALSE-TRUE	FALSE	启动信号；TRUE时使能功能块，上升沿有效；
FileName	文件名	STRING（32）	遵照数据类型		新建立的文件名称
WriteDataPointer	写入文件内容的指针	POINTER TO BYTE	遵照数据类型	0	写入文件内的数据内容的地址
DataSize	数据大小	DWORD	遵照数据类型	0	新建文件大小，单位是字节

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
bDone	返回值	BOOL	FALSE-TRUE	FALSE	完成信号；True-写入文件完成；
bError	错误	BOOL	FALSE-TRUE	FALSE	错误状态；FALSE-没有错误；
dErrorID	错误代码	UDINT	遵照数据类型	0	错误代码

	布尔	位串				整数								实数		时刻、持续时间、日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Execute	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
FileName	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√
WriteDataPointer	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
DataSize	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

bDone	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
bError	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
dErrorID	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-

- **功能说明：**

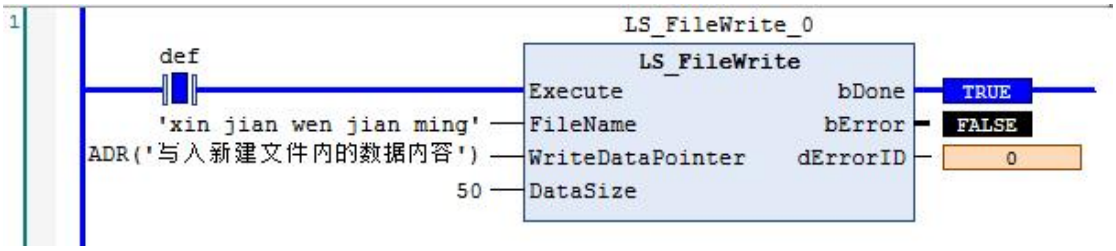
  - 在控制器“UsrData”文件夹内创建新的文件（文件名称由FileName指定，文件大小由DataSize指定），在文件内写入数据内容（数据内容是WriteDataPointer指向的地址内储存的数据）。若该文件名已存在，旧的文件内容将会被覆盖。

- **程序示例**

ST：当启动信号 abc 变为 True 时，新建文件（文件名称为“xin jian wen jian ming”）并在文件内写入数据内容(内容为“写入新建文件内的数据内容”)。

```
LS_FileWrite_0(
    Execute TRUE :=abc TRUE,
    FileName xin jian w :='xin jian wen jian ming',
    WriteDataPointer 16#B4466216 :=ADR('写入新建文件内的数据内容')
    DataSize 50 := 50,
    bDone TRUE =>bl TRUE,
    bError FALSE =>bel FALSE ,
    dErrorID 0 =>dl 0 );
    RETURN
```

LD：当启动信号 def 变为 True 时，新建文件（xin jian wen jian ming）并在文件内写入数据内容（写入新建文件内的数据内容）。



- **注意事项**

- DataSize是新建立文件的大小，需要比WriteDataPointer所指定的数据大小大，否则写入的数据可能丢失。

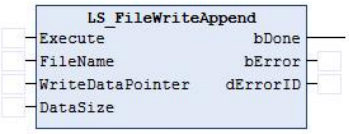
- 注意PLC不支持建立中文名字的文件，文件内容最好不包含中文，否则LS_FileRead指令

读取到的可能无法识别（上述例子中写入内容是中文，读取到的是乱码）。

- WriteDataPointer是写入文件内的数据内容的地址。
- 指令默认在控制器“UsrData”文件夹内创建新的文件，不支持在线仿真。

7.4.3 LS_FileWriteAppend

- **指令格式**

指令	名称	FB/FC	LD 表现	ST 表现	库
LS_FileWriteAppend	文件追加写入指令	FB		LS_FileWriteAppend_0(Execute:= , FileName:= , WriteDataPointer:= , DataSize:= , bDone=> , bError=> , dErrorID=>);	PMC_FileManage库

- **相关变量**

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Execute	启动	BOOL	FALSE-TRUE	FALSE	启动信号；TRUE时使能功能块，上升沿触发；
FileName	文件名	STRING（32）	遵照数据类型		待追加写入数据内容的文件名称
WriteDataPointer	写入数据内容的指针	POINTER TO BYTE	遵照数据类型	0	追加写入的数据内容的地址
DataSize	数据大小	DWORD	遵照数据类型	0	追加写入的数据大小，单位是字节

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
bDone	返回值	BOOL	FALSE-TRUE	FALSE	完成信号；True-文件追加写入完成；

bError	错误	BOOL	FALSE-TRUE	FALSE	错误状态；FALSE-没有错误；
dErrorID	错误代码	UDINT	遵照数据类型	0	错误代码

	布尔	位串				整数								实数		时刻、持续时间、日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Execute	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
FileName	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√
WriteDataPointer	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
DataSize	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
bDone	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
bError	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
dErrorID	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-

- **功能说明：**

  - 在控制器“UsrData”文件夹内，对FileName指定的文件以追加（扩大原文件大小）的方式写入WriteDataPointer指定的数据内容。

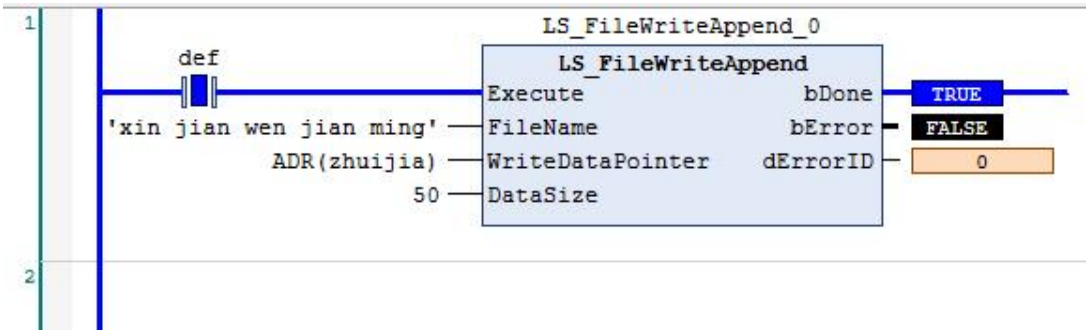
- **程序示例**

ST：当启动信号 abc 变为 True 时，在指定文件（xin jian wen jian ming）中追加写入数据内容（ARE YOU OK ?）。

```
1  LS_FileWriteAppend_0(
2      Execute TRUE := abc TRUE,
3      FileName 'xin jian w ▸ := 'xin jian wen jian ming',
4      WriteDataPointer 16#B44B9930 :=ADR(zhuijia 'ARE YOU OK ▸ ) ,
5      DataSize 50 :=50 ,
6      bDone TRUE => bl TRUE,
7      bError FALSE => bel FALSE,
8      dErrorID 0 =>dl 0 );
9
10 RETURN
```

LD：当启动信号 def 变为 True 时，在指定文件（xin jian wen jian ming）中追加写入数据

内容（HANDSOME BOY !）。



- **注意事项**

➢ DataSize是追加写入的数据大小，需要比WriteDataPointer所指定的数据大小大，否则追加写入的数据可能丢失。

➢ Execute上升沿触发，每次执行文件追加写入指令后，文件大小变为原文件大小加上DataSize的大小;追加写入的数据从增加的字节开始写入,可以用LS_FileRead指令读取(设置偏移值OffSetPosition，从新增加的字节处开始读取）。

➢ 注意PLC不支持建立中文名字的文件，文件内容最好不包含中文，否则LS_FileRead指令读取到的可能无法识别。

➢ WriteDataPointer是追加写入的数据内容的地址。

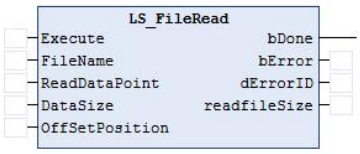
➢ 如果在控制器“UsrData”文件夹内没有FileName指定的文件名的文件，则会新建一个文件，然后写入数据内容。

➢ 该指令在控制器“UsrData”文件夹内进行操作，不支持在线仿真。

7.4.4 LS_FileRead

- **指令格式**

指令	名称	FB/FC	LD 表现	ST 表现	库

LS_FileRead	文件读取指令	FB		LS_FileRead_0(Execute:= , FileName:= , ReadDataPoint:= , DataSize:= , OffSetPosition:= , bDone=> , bError=> , dErrorID=> , readfileSize=>);	PMC_FileManage 库

- **相关变量**

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Execute	启动	BOOL	FALSE-TRUE	FALSE	启动信号；TRUE时使能功能块，上升沿触发；
FileName	文件名	STRING（32）	遵照数据类型		要读取数据内容的文件名称
ReadDataPoint	数据接收指针	POINTER TO BYTE	遵照数据类型		将读取到的数据内容保存到指定的地址
DataSize	数据大小	DWORD	遵照数据类型	0	最大能够读取的数据大小
OffSetPosition	位置偏移	DWORD	遵照数据类型	0	从指定字节之后开始读取数据

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
bDone	返回值	BOOL	FALSE-TRUE	FALSE	完成信号；True-读取文件完成；
bError	错误	BOOL	FALSE-TRUE	FALSE	错误状态；FALSE-没有错误；
dErrorID	错误代码	UDINT	遵照数据类型	0	错误代码
readfileSize	读取的文件大小	DWORD	遵照数据类型	0	实际读取到的数据大小

	布尔	位串	整数	实数	时刻、持续时间、日期、字符串

	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Execute	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
FileName	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√
ReadDataPoint	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
DataSize	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
OffSetPosition	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
bDone	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
bError	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
dErrorID	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-
readfileSize	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

- **功能说明：**

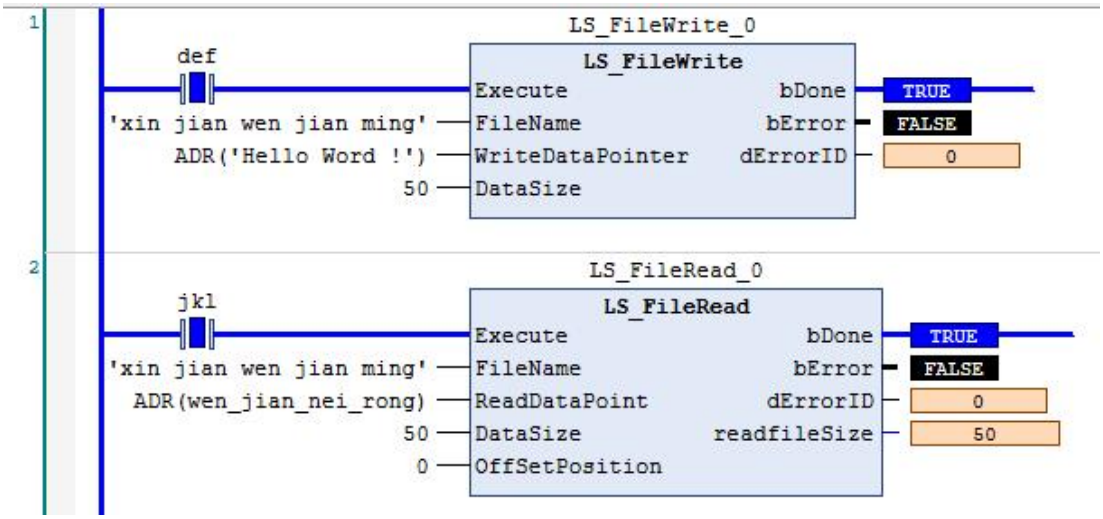
  - 在控制器“UsrData”文件夹内读取FileName所指定名称的文件中的数据内容，读取的数据内容存储在ReadDataPointer所指定的地址中。

- **程序示例**

  ST：当启动信号 ghi 变为 True 时，读取指定文件（xin jian wen jian ming）的数据内容保存到 ReadDataPoint 所指定的地址(ADR(wen_jian_nei_rong))。

```
1  LS_FileWrite_0(
2       Execute TRUE :=abc TRUE,
3       FileName 'xin jian w ▸ :='xin jian wen jian ming',
4       WriteDataPointer 16#B44665EC :=ADR('welcome to leadshine !') ,
5       DataSize 50 := 50,
6       bDone TRUE =>b1 TRUE,
7       bError FALSE =>be1 FALSE ,
8       dErrorID 0 =>d1 0 );
9
10
11 LS_FileRead_0(
12      Execute TRUE := ghi TRUE,
13      FileName 'xin jian w ▸ :='xin jian wen jian ming' ,
14      ReadDataPoint 16#B4401A60 :=ADR(wen_jian_nei_rong 'welcome to ▸ ) ,
15      DataSize 50 :=50 ,
16      OffSetPosition 0 :=0 ,
17      bDone TRUE => b2 TRUE,
18      bError FALSE => be2 FALSE,
19      dErrorID 0 =>d2 0 ,
20      readfileSize 50 =>r2 50 );
21      RETURN
```

LD：当启动信号jkl变为True时，读取指定文件的数据内容保存到ReadDataPoint所指定的地址。



- **注意事项**

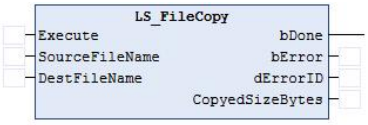
  - DataSize是最大能读取到的数据大小（要比被读取文件大，否则数据可能读取不完整），readfileSize是实际读取到的数据大小（取决于文件大小与DataSize）。
  - OffSetPosition是从指定的字节后开始读取数据内容。
  - 读取到的文件数据内容存储在ReadDataPointer所指定的地址中。

- 注意PLC不支持建立中文名字的文件，文件内容最好不包含中文，否则LS_FileRead指令读取到的可能无法识别。

- 该指令在控制器“UsrData”文件夹内进行操作，不支持在线仿真。

7.4.5 LS_FileCopy

- **指令格式**

指令	名称	FB/FC	LD 表现	ST 表现	库
LS_FileCopy	文件拷贝指令	FB		LS_FileCopy_0(Execute:= , SourceFileName:= , DestFileName:= , bDone=> , bError=> , dErrorID=> , CopyedSizeBytes=>);	PMC_FileManage库

- **相关变量**

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Execute	启动	BOOL	FALSE-TRUE	FALSE	启动信号；TRUE时使能功能块，上升沿触发；
SourceFileName	源文件名	STRING（32）	遵照数据类型		需要拷贝的源文件名称
DestFileName	目标文件名	STRING（32）	遵照数据类型		拷贝后生成的新文件名称

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
bDone	返回值	BOOL	FALSE-TRUE	FALSE	完成信号；True-拷贝文件

					完成；
bError	错误	BOOL	FALSE-TRUE	FALSE	错误状态；FALSE-没有错误；
dErrorID	错误代码	UDINT	遵照数据类型	0	错误代码
CopyedSizeBytes	拷贝的字节数	DWORD	遵照数据类型	0	拷贝后生成的新文件大小

	布尔	位串				整数								实数		时刻、持续时间、日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Execute	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
SourceFileName	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√
DestFileName	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√
bDone	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
bError	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
dErrorID	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-
CopyedSizeBytes	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

- **功能说明：**

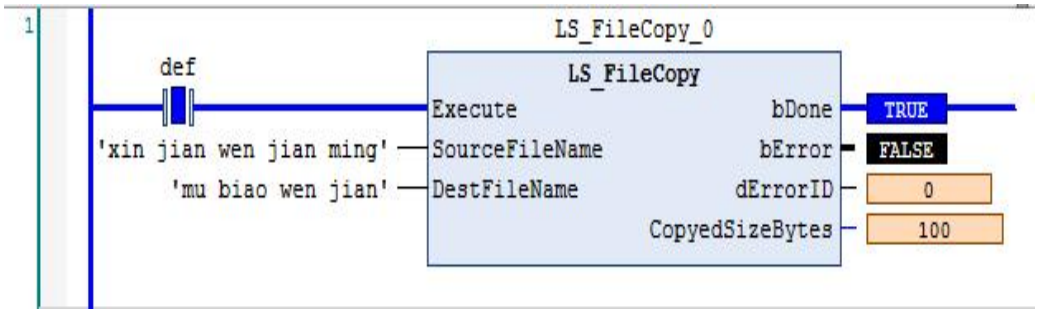
  - 在控制器“UsrData”文件夹内将SourceFileName指定的源文件拷贝到DestFileName所指定的目标文件中，拷贝的字节数CopyedSizeBytes是源文件大小（拷贝的是整个源文件，拷贝的字节数就是源文件大小）。

- **程序示例**

ST：当启动信号 abc 变为 True 时，将指定的源文件（xin jian wen jian ming）拷贝到指定的目标文件（mu biao wen jian）中。

```
LS_FileCopy_0(
    Execute TRUE :=abc TRUE ,
    SourceFileName 'xin jian w ▸ :='xin jian wen jian ming' ,
    DestFileName 'mu biao we ▸ :='mu biao wen jian' ,
    bDone TRUE =>bl TRUE ,
    bError FALSE =>bel FALSE ,
    dErrorID 0 => dl 0 ,
    CopyedSizeBytes 100 =>col 100 );
RETURN
```

LD：当启动信号def变为True时，将指定的源文件拷贝到指定的目标文件中。

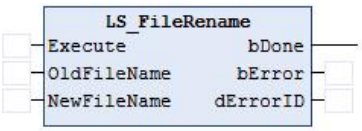


- **注意事项**
  - LS_FileCopy指令实际上是复制整个源文件，另保存为另一个名称的文件。
  - 如果源文件在控制器“UsrData”文件夹内不存在，运行LS_FileCopy指令会报错。
  - 如果目标文件名字在控制器“UsrData”文件夹内已经存在，则会覆盖旧文件内容。
  - 注意PLC不支持建立中文名字的文件。
  - 该指令在控制器“UsrData”文件夹内进行操作，不支持在线仿真。

7.4.6 LS_FileRename

- **指令格式**

指令	名称	FB/ FC	LD 表现	ST 表现	库

LS_FileRename	文件重命名指令	FB		LS_FileRename_0(Execute:= , OldFileName:= , NewFileName:= , bDone=> , bError=> , dErrorID=>);	PMC_FileManage 库

- **相关变量**

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Execute	启动	BOOL	FALSE-TRUE	FALSE	启动信号；TRUE时使能功能块，上升沿触发；
OldFileName	原文件名	STRING（32）	遵照数据类型		原文件名称
NewFileName	新文件名	STRING（32）	遵照数据类型		新文件名称

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
bDone	返回值	BOOL	FALSE-TRUE	FALSE	完成信号；True-文件重命名完成；
bError	错误	BOOL	FALSE-TRUE	FALSE	错误状态；FALSE-没有错误；
dErrorID	错误代码	UDINT	遵照数据类型	0	错误代码

	布尔	位串				整数								实数		时刻、持续时间、日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Execute	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
OldFileNa	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√

me																				
NewFileName	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√
bDone	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
bError	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
dErrorID	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-

- **功能说明：**

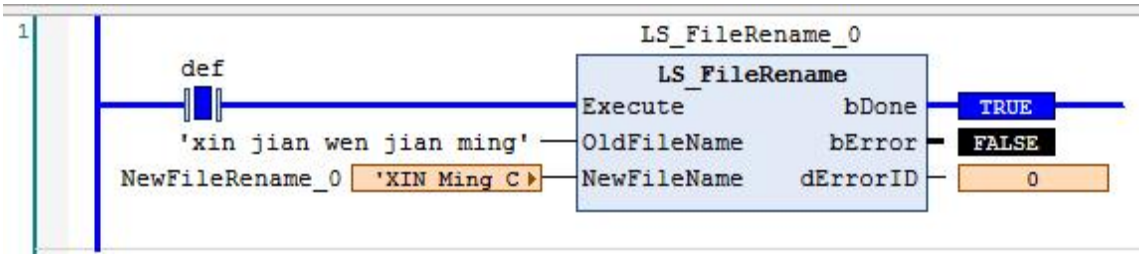
  - 在控制器“UsrData”文件夹内对OldFileName所指定的文件名称进行重命名操作，新文件名称由NewFileName指定。

- **程序示例**

ST：当启动信号 abc 变为 True 时，将指定的原文件名称（mu biao wen jian）改成 NewFileName 指定的新名称（xin ming cheng）。

```
1  LS_FileRename_0(
2       Execute TRUE :=abc TRUE ,
3       OldFileName 'mu biao we' := 'mu biao wen jian',
4       NewFileName 'xin ming c' :='xin ming cheng' ,
5       bDone TRUE =>bl TRUE ,
6       bError FALSE => bel FALSE,
7       dErrorID 0 =>dl 0 );
8  RETURN
```

LD：当启动信号def变为True时，将指定的原文件名称改成NewFileName指定的新名称。



- **注意事项**

  - 如果NewFileName指定的新名称在控制器“UsrData”文件夹内已存在，运行LS_FileRename指令后会覆盖旧文件内容。

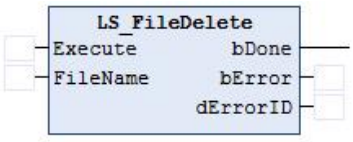
  - 如果OldFileName所指定的原文件在控制器“UsrData”文件夹内不存在，运行

LS_FileRename指令会报错。

- 注意PLC不支持建立中文名字的文件，PLC识别不了。
- 该指令在控制器“UsrData”文件夹内进行操作，不支持在线仿真。

7.4.7 LS_FileDelete

- **指令格式**

指令	名称	FB/FC	LD 表现	ST 表现	库
LS_FileDelete	文件删除指令	FB		LS_FileDelete_0(Execute:= , FileName:= , bDone=> , bError=> , dErrorID=>);	PMC_FileManage 库

- **相关变量**

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Execute	启动	BOOL	FALSE-TRUE	FALSE	启动信号；TRUE时使能功能块，上升沿触发；
FileName	文件名	STRING（32）	遵照数据类型		需要删除的文件名称

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
bDone	返回值	BOOL	FALSE-TRUE	FALSE	完成信号；True-文件删除完成；
bError	错误	BOOL	FALSE-TRUE	FALSE	错误状态；FALSE-没有错误；
dErrorID	错误代码	UDINT	遵照数据类型	0	错误代码

	布尔	位串				整数								实数		时刻、持续时间、日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Execute	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
FileName	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√
bDone	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
bError	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
dErrorID	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-

- **功能说明：**

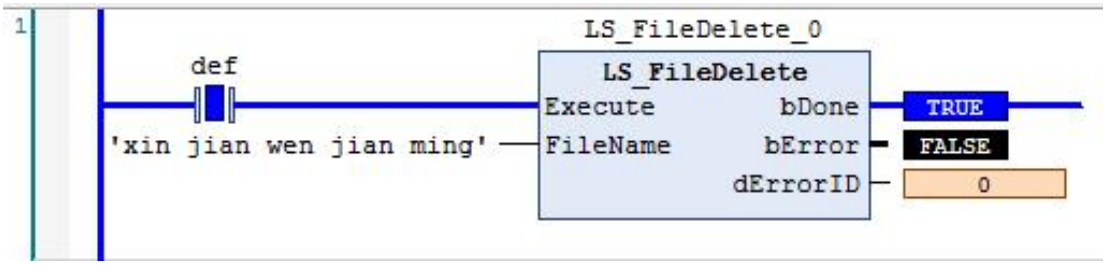
  - 在控制器“UsrData”文件夹内删除FileName所指定名称的文件。

- **程序示例**

ST：当启动信号 abc 变为 True 时，将指定名称的文件（XIN Ming Ccheng）删除。

```
1  LS_FileDelete_0(
2       Execute TRUE :=abc TRUE ,
3       FileName XIN Ming C :='XIN Ming Ccheng' ,
4       bDone TRUE =>b1 TRUE ,
5       bError FALSE => be1 FALSE,
6       dErrorID 0 =>d1 0 );
7  RETURN
```

LD：当启动信号def变为True时，将指定名称的文件删除。



- **注意事项**

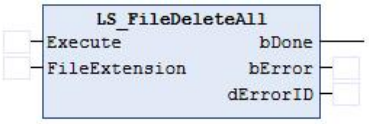
  - 如果FileName所指定名称的文件在控制器“UsrData”文件夹内不存在，运行

LS_FileDelete指令后会报错。

- 该指令在控制器“UsrData”文件夹内进行操作，不支持在线仿真。

7.4.8 LS_FileDeleteAll

- **指令格式**

指令	名称	FB/FC	LD 表现	ST 表现	库
LS_FileDeleteAll	文件批量删除指令	FB		LS_FileDeleteAll_0(Execute:= , FileExtension:= , bDone=> , bError=> , dErrorID=>);	PMC_FileManage库

- **相关变量**

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Execute	启动	BOOL	FALSE-TRUE	FALSE	启动信号；TRUE时使能功能块，上升沿触发；
FileExtension	文件名包含字符串	STRING（8）	遵照数据类型		文件名称包含的字符串；不输入字符表示删除所有文件；

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
bDone	返回值	BOOL	FALSE-TRUE	FALSE	完成信号；True-文件批量删除完成；
bError	错误	BOOL	FALSE-TRUE	FALSE	错误状态；FALSE-没有错误；
dErrorID	错误代码	UDINT	遵照数据类型	0	错误代码

	布尔	位串	整数	实数	时刻、持续时间、日期、字符串

	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Execute	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
FileExtension	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√
bDone	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
bError	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
dErrorID	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-

- **功能说明：**

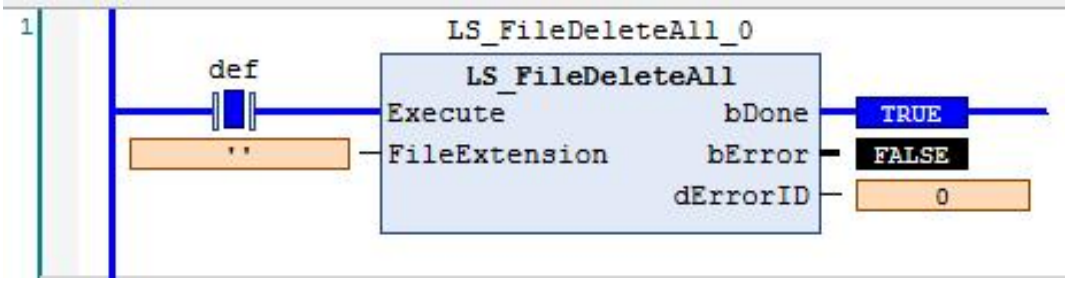
  - 在控制器“UsrData”文件夹内批量删除文件名称中包含FileExtension所指定字符串的多个文件；若FileExtension指定字符串为空，则删除控制器“UsrData”文件夹内的所有文件。

- **程序示例**

ST：当启动信号 abc 变为 True 时，将名称中包含 FileExtension 所指定字符串（例子中为“.txt”）的多个文件删除。

```
1  LS_FileDeleteAll_0(
2      Execute TRUE :=abc TRUE ,
3      FileExtension '.txt' :='.txt' ,
4      bDone TRUE =>bl TRUE ,
5      bError FALSE =>bel FALSE ,
6      dErrorID 0 =>del 0 );
7  RETURN
```

LD：将FileExtension指定字符串留空，当启动信号def变为True时，将控制器“UsrData”文件夹内的所有文件删除。

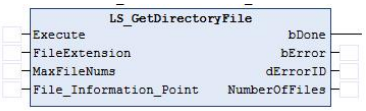


- **注意事项**

- 如果控制器“UsrData”文件夹内有多个文件的名称中包含有FileExtension所指定的字符串，运行LS_FileDeleteAll指令后，该多个文件被删除。
- 如果FileExtension不输入字符串，运行LS_FileDeleteAll指令后，控制器“UsrData”文件夹内所有文件被删除。
- 该指令在控制器“UsrData”文件夹内进行操作，不支持在线仿真。

7.4.9 LS_GetDirectoryFile

- **指令格式**

指令	名称	FB/FC	LD 表现	ST 表现	库
LS_GetDirectoryFile	获取目录内文件信息指令	FB		LS_GetDirectoryFile_0(Execute:= , FileExtension:= , MaxFileNums:= , File_Information_Point:= , bDone=> , bError=> , dErrorID=> , NumberOfFiles=>);	PMC_FileManage库

- **相关变量**

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Execute	启动	BOOL	FALSE-TRUE	FALSE	启动信号；TRUE时使能功能块，上升沿触发；
FileExtension	文件名包含字符串	STRING（8）	遵照数据类型		文件名称中包含的字符串；不输入字符表示获取目录内所有文件的信息
MaxFileNums	最大文	INT	遵照数据类型	20	最大能读取的文件

	件数				个数
File_Information_Point	文件信息存储的指针	POINTER TO File_Information	遵照数据类型		读取的文件信息存储的地址

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
bDone	返回值	BOOL	FALSE-TRUE	FALSE	完成信号；True-获取目录内文件信息完成；
bError	错误	BOOL	FALSE-TRUE	FALSE	错误状态；FALSE-没有错误；
dErrorID	错误代码	UDINT	遵照数据类型	0	错误代码
NumberOfFiles	读取文件个数	INT	遵照数据类型	0	实际读取到信息的文件个数

	布尔	位串				整数								实数		时刻、持续时间、日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Execute	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
FileExtension	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√
MaxFileNums	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-
File_Information_Point	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
bDone	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
bError	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
dErrorID	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-
NumberOfFiles	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-

- **功能说明：**

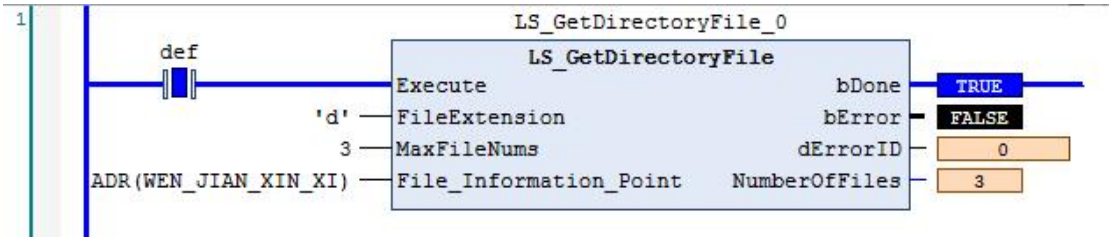
  - 在控制器“UsrData”文件夹内，获取该目录下文件名称中包含FileExtension所指定字符串的所有文件的信息；如果FileExtension没有指定字符串，则获取控制器“UsrData”文件夹内所有文件的信息；（文件信息包括文件名，文件创建时间，文件最近修改时间，文件大小）。

- **程序示例**

ST：当启动信号abc变为True时，获取控制器“UsrData”文件夹内，文件名称中包含“txt”的所有文件的信息。

```
1  LS_GetDirectoryFile_0(
2       Execute TRUE :=abc TRUE,
3       FileExtension 'txt' := 'txt',
4       MaxFileNums 4 :=4 ,
5       File_Information_Point 16#B44CD86C :=ADR(fileinfor) ,
6       bDone TRUE => B1 TRUE,
7       bError FALSE => BE1 FALSE,
8       dErrorID 0 => D1 0,
9       NumberOfFiles 4 =>NUM1 4 );
10 RETURN
```

LD：当启动信号def变为True时，获取控制器“UsrData”文件夹内，文件名称中包含“d”的所有文件的信息。



- **注意事项**

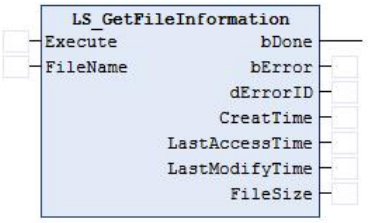
- 最大读取的文件个数由MaxFileNums指定，当控制器“UsrData”文件夹内存在较多文件符合FileExtension指定字符串要求时，只读取排序靠前指定个数的文件信息。
- NumberOfFiles是实际读取到的文件个数，取决于符合FileExtension指定字符串要求的文件数和最大读取的文件个数MaxFileNums。
- 读取到的文件信息分为四个字符串，每个字符串占31个字节；若文件名称过长，超过31字节会溢出，导致识别不出文件名称。
- 读取控制器“UsrData”文件夹内目录下所有文件时，由于读到的是多个文件的信息，定义的文件信息指针File_Information_Point必须是指向文件信息结构体数组的，而不是一个文件信息结构体。
- 如上面LD示例中，定义WEN_JIAN_XIN_XI ： array [0..20] OF File_Information；是

一个文件信息结构体数组。

- File_Information是PMC_FileManage库自带的，是一个文件信息结构体。
- 该指令在控制器“UsrData”文件夹内进行操作，不支持在线仿真。

7.4.10 LS_GetFileInformation

- **指令格式**

指令	名称	FB/FC	LD 表现	ST 表现	库
LS_GetFileInformation	获取指定文件信息指令	FB		LS_GetFileInformation_0(Execute:=, FileName:=, bDone=>, bError=>, dErrorID=>, CreatTime=>, LastAccessTime=>, LastModifyTime=>, FileSize=>);	PMC_FileManage库

- **相关变量**

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Execute	启动	BOOL	FALSE-TRUE	FALSE	启动信号；TRUE时使能功能块，上升沿触发；
FileName	文件名	STRING（32）	遵照数据类型		指定要读取信息的文件名称

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
bDone	返回值	BOOL	FALSE-TRUE	FALSE	完成信号；True-获取指定文件信息完成；
bError	错误	BOOL	FALSE-TRUE	FALSE	错误状态；FALSE-没有错误；
dErrorID	错误代码	UDINT	遵照数据类型	0	错误代码
CreatTime	创建时间	STRING（31）	遵照数据类型		指定名称文件的创建时间
LastAccessTime	最近访问时间	STRING（31）	遵照数据类型		指定名称文件的最近访问时间
LastModifyTime	最近修改时间	STRING（31）	遵照数据类型		指定名称文件的最近修改时间
FileSize	文件大小	UDINT	遵照数据类型	0	指定名称文件的大小（Byte）

	布尔	位串				整数								实数		时刻、持续时间、日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Execute	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
FileName	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√
bDone	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
bError	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
dErrorID	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-
CreatTime	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√
LastAccess Time	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√
LastModify Time	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√
FileSize	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-

- **功能说明：**

  - 在控制器“UsrData”文件夹内，获取Filename所指定名称的文件的信息，包括创建时间、最近访问、修改时间，文件大小等。

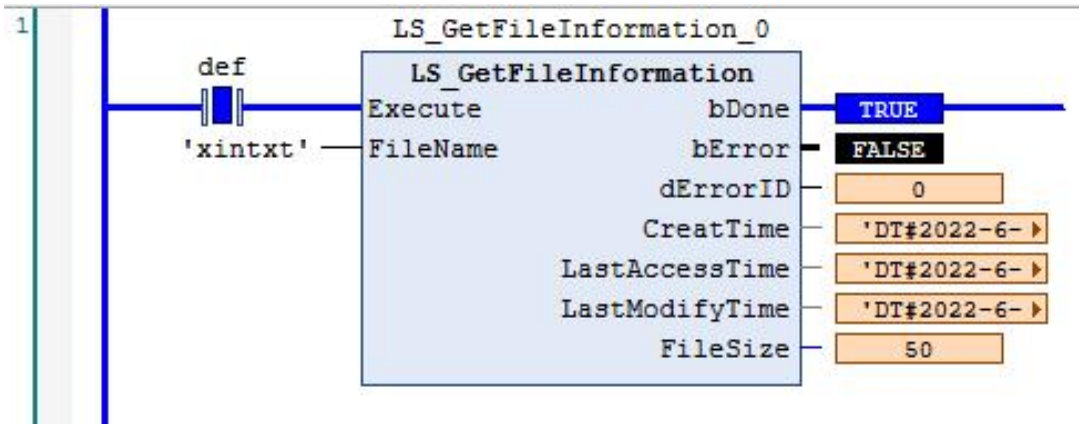
- **程序示例**

ST：当启动信号 abc 变为 True 时，获取控制器“UsrData”文件夹内，文件（xintxt）的信息。

```
LS_GetFileInformation_0(
    Execute TRUE :=abc TRUE ,
    FileName 'xintxt' :='xintxt' ,
    bDone TRUE =>bl TRUE ,
    bError FALSE => bel FALSE,
    dErrorID 0 => dl 0,
    CreatTime 'DT#2022-6- =>cl 'DT#2022-6- ,
    LastAccessTime 'DT#2022-6- =>lal 'DT#2022-6- ,
    LastModifyTime 'DT#2022-6- => la2 'DT#2022-6- ,
    FileSize 50 => fil 50 );

RETURN
```

LD：当启动信号 def 变为 True 时，获取控制器“UsrData”文件夹内，文件（xintxt）的信息。

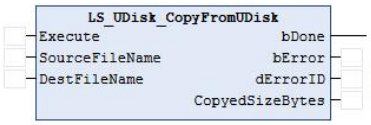


- **注意事项**

- Execute上升沿触发；变为Flase后会保持最后读取到的文件信息。
- 如果在控制器“UsrData”文件夹内不存在Filename指定名称的文件，运行LS_GetFileInformation指令会报错。
- 该指令在控制器“UsrData”文件夹内进行操作，不支持在线仿真。

7.4.11 LS_UDisk_CopyFromUDisk

- **指令格式**

指令	名称	FB/FC	LD 表现	ST 表现	库
LS_UDisk_CopyFromUDisk	从U盘拷贝文件指令	FB		LS_UDisk_CopyFromUDisk_0(Execute:= , SourceFileName:= , DestFileName:= , bDone=> , bError=> , dErrorID=> , CopyedSizeBytes=>);	PMC_FileManage库

- **相关变量**

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Execute	启动	BOOL	FALSE-TRUE	FALSE	启动信号；TRUE时使能功能块，上升沿触发；
SourceFileName	源文件名	STRING（32）	遵照数据类型		U盘中需要拷贝的源文件名称
DestFileName	目标文件名	STRING（32）	遵照数据类型		拷贝后在控制器“UsrData”文件夹内生成的新文件名称

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
bDone	返回值	BOOL	FALSE-TRUE	FALSE	完成信号；True-从U盘拷贝文件至控制器“UsrData”文件夹内完成；
bError	错误	BOOL	FALSE-TRUE	FALSE	错误状态；FALSE-没有错误；
dErrorID	错误代码	UDINT	遵照数据类型	0	错误代码
CopyedSizeByt	拷贝的	DWORD	遵照数据类型	0	拷贝后生成的新文件大

es	字节数				小

	布尔	位串				整数								实数		时刻、持续时间、日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Execute	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
SourceFileName	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√
DestFileName	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√
bDone	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
bError	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
dErrorID	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-
CopyedSizeBytes	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

- **功能说明：**

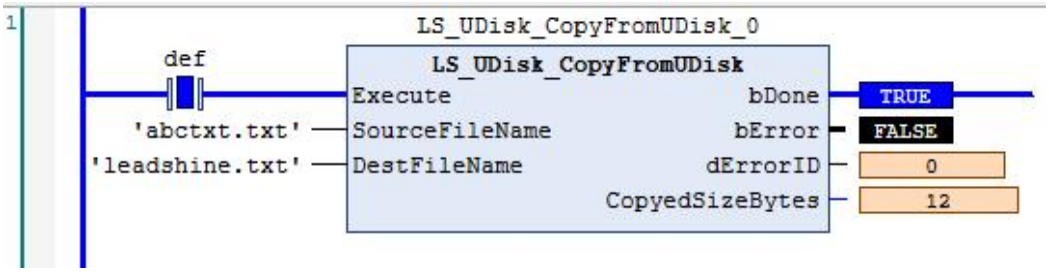
  - 从U盘根目录拷贝文件（文件名称由SourceFileName指定）至控制器“UsrData”文件夹内（拷贝生成的新文件名称由DestFileName指定）。

- **程序示例**

ST：当启动信号 abc 变为 True 时，从 U 盘根目录拷贝文件（abc.txt.txt）至控制器“UsrData”文件夹内（新生成文件 mubiaownejian.txt）。

```
LS_UDisk_CopyFromUDisk_0(
    Execute TRUE :=abc TRUE ,
    SourceFileName 'abc.txt.tx' := 'abc.txt.txt',
    DestFileName 'mubiaownej' :='mubiaownejian.txt' ,
    bDone TRUE => bl TRUE,
    bError FALSE => bel FALSE,
    dErrorID 0 =>del 0 ,
    CopyedSizeBytes 24 => col 24 );
RETURN
```

LD: 当启动信号 def 变为 True 时，从 U 盘根目录拷贝文件（abctxt.txt）至控制器“UsrData”文件夹内（新生成文件 leadshine.txt）。



- **注意事项**

➢ 如果SourceFileName指定的源文件在U盘根目录中不存在，运行LS_UDisk_CopyFromUDisk后将会报错；错误代码16。

➢ 如果DestFileName指定的目标文件在控制器“UsrData”文件夹内已存在，运行LS_UDisk_CopyFromUDisk后，旧文件的数据内容会被覆盖。

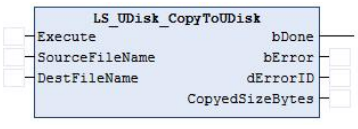
➢ 注意SourceFileName指定的源文件需要带格式后缀；如ST例子中，在U盘根目录中文件的名字为(abc.txt)，其格式为(.txt)。

➢ 注意PLC不支持建立中文名字的文件，文件中的内容最好不用中文，PLC识别不了。

➢ 该指令在控制器“UsrData”文件夹内进行操作，不支持在线仿真。

7.4.12 LS_UDisk_CopyToUDisk

- **指令格式**

指令	名称	FB/FC	LD 表现	ST 表现	库
LS_UDisk_CopyToUDisk	拷贝文件至U盘指令	FB		LS_UDisk_CopyToUDisk_0(Execute:= , SourceFileName:= , DestFileName:= , bDone=> ,	PMC_FileManage库

				bError=> , dErrorID=> , CopyedSizeBytes=>);	

- **相关变量**

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Execute	启动	BOOL	FALSE-TRUE	FALSE	启动信号；TRUE时使能功能块，上升沿触发；
SourceFileName	源文件名	STRING（32）	遵照数据类型		控制器“UsrData”文件夹内需要拷贝的源文件名称
DestFileName	目标文件名	STRING（32）	遵照数据类型		拷贝后在U盘生成的新文件名称

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
bDone	返回值	BOOL	FALSE-TRUE	FALSE	完成信号；True-拷贝文件至U盘完成；
bError	错误	BOOL	FALSE-TRUE	FALSE	错误状态；FALSE-没有错误；
dErrorID	错误代码	UDINT	遵照数据类型	0	错误代码
CopyedSizeBytes	拷贝的字节数	DWORD	遵照数据类型	0	拷贝后生成的新文件大小

	布尔	位串				整数								实数		时刻、持续时间、日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Execute	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
SourceFileName	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√
DestFileName	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√

bDone	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
bError	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
dErrorID	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-
CopyedSize Bytes	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

- **功能说明：**

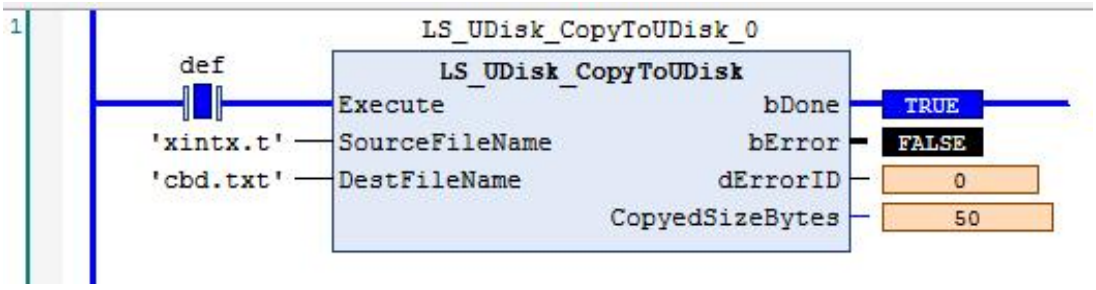
  - 从控制器“UsrData”文件夹内拷贝文件（文件名称由SourceFileName指定）至U盘根目录（拷贝生成的新文件名称由DestFileName指定）。

- **程序示例**

ST：当启动信号 abc 变为 True 时，从控制器“UsrData”文件夹内拷贝文件（mubiaowenjian.txt）至 U 盘根目录（新生成文件 newfile.txt）。

```
LS_UDisk_CopyToUDisk_0(
    Execute TRUE :=abc TRUE ,
    SourceFileName 'mubiaownej ▸ := 'mubiaownejian.txt',
    DestFileName 'newfile.tx ▸ := 'newfile.txt',
    bDone TRUE =>bdl TRUE ,
    bError FALSE =>bel FALSE ,
    dErrorID 0 =>del 0 ,
    CopyedSizeBytes 24 =>col 24 );
    RETURN
```

LD：当启动信号 def 变为 True 时，从控制器“UsrData”文件夹内拷贝文件（xintx.t）至 U 盘根目录（新生成文件 cbd.txt）。



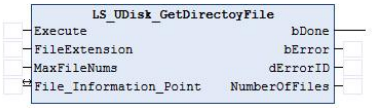
- **注意事项**

  - 如果SourceFileName指定的源文件在控制器“UsrData”文件夹内不存在，运行LS_UDisk_CopyToUDisk后将会报错；错误代码16。

- 如果DestFileName指定的目标文件在U盘根目录内已存在，运行LS_UDisk_CopyToUDisk后，旧文件的数据内容会被覆盖。
- 注意DestFileName指定的目标文件如果带格式后缀，在U盘根目录中会自动生成对应格式的文件。
- 注意PLC不支持建立中文名字的文件，文件内容最好不带中文，否则复制到U盘中会是乱码。
- 该指令在控制器“UsrData”文件夹内进行操作，不支持在线仿真。

7.4.13 LS_UDisk_GetDirectoyFile

- **指令格式**

指令	名称	FB/FC	LD 表现	ST 表现	库
LS_UDisk_GetDirectoyFile	获取U盘根目录内文件信息指令	FB		LS_UDisk_GetDirectoyFile_0(Execute:= , FileExtension:= , MaxFileNums:= , File_Information_Point:= , bDone=> , bError=> , dErrorID=> , NumberOfFiles=>);	PMC_FileManage库

- **相关变量**

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Execute	启动	BOOL	FALSE-TRUE	FALSE	启动信号；TRUE时使能功能块，上升沿触发；
FileExtension	文件名	STRING（8）	遵照数据类型		文件名称中包含的

	包含字符串				字符串；不输入字符表示获取目录内所有文件的信息
MaxFileNums	最大文件数	INT	遵照数据类型	20	最大能读取的文件个数
File_Information_Point	文件信息存储的指针	POINTER TO File_Information	遵照数据类型		读取的文件信息存储的地址

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
bDone	返回值	BOOL	FALSE-TRUE	FALSE	完成信号；True-获取U盘根目录下的文件信息完成；
bError	错误	BOOL	FALSE-TRUE	FALSE	错误状态；FALSE-没有错误；
dErrorID	错误代码	UDINT	遵照数据类型	0	错误代码
NumberOfFiles	读取文件个数	INT	遵照数据类型	0	实际读取到信息的文件个数

	布尔	位串				整数								实数		时刻、持续时间、日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Execute	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
FileExtension	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√
MaxFileNums	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-
File_Information_Point	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
bDone	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
bError	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
dErrorID	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-
NumberOfFiles	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-

- **功能说明：**

  - 在U盘根目录下，获取该目录下文件名称中包含FileExtension所指定字符串的所有文

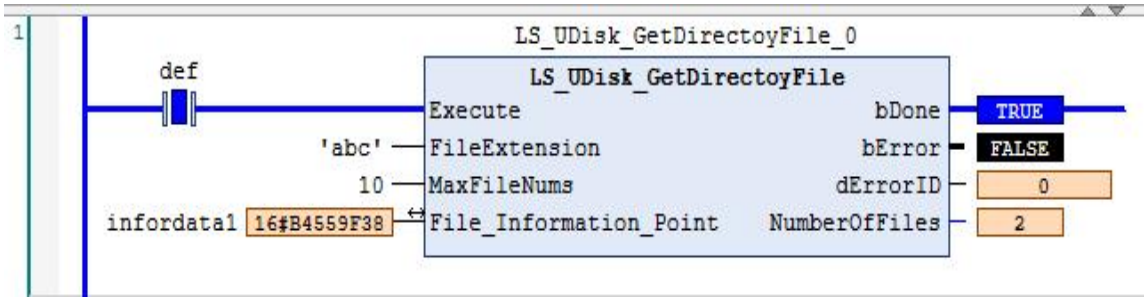
件的信息；如果FileExtension没有指定字符串，则获取U盘根目录下所有文件的信息；（文件信息包括文件名，文件创建时间，文件最近修改时间，文件大小）。

- **程序示例**

ST：将FileExtension留空，当启动信号abc变为True时，获取U盘根目录下，所有文件的信息。

```
LS_UDisk_GetDirectoyFile_0(
    Execute TRUE := abc TRUE,
    FileExtension:= ,
    MaxFileNums 10 :=10 ,
    File_Information_Point 16#B4558440 :=infordata2 16#B4558440 ,
    bDone TRUE => bd1 TRUE,
    bError FALSE => be1 FALSE,
    dErrorID 0 => de1 0,
    NumberOfFiles 7 =>num1 7 );
    RETURN
```

LD：当启动信号def变为True时，获取U盘根目录下，文件名称中包含“abc”的所有文件的信息。



- **注意事项**

- 最大读取的文件个数由MaxFileNums指定，当U盘根目录下存在较多文件符合FileExtension指定字符串要求时，只读取排序靠前指定个数的文件信息。
- NumberOfFiles是实际读取到的文件个数，取决于符合FileExtension指定字符串要求的文件数和最大读取的文件个数MaxFileNums。
- 读取到的文件信息分为四个字符串，每个字符串占31个字节；若文件名称过长，超过31字节会溢出，导致识别不出文件名称。

- 读取U盘根目录下所有文件时，由于读到的是多个文件的信息，定义的文件信息指针File_Information_Point必须是指向文件信息结构体数组的，而不是一个文件信息结构体。

  如上面LD示例中，定义

  infordata1: POINTER TO File_Information := ADR(fileinfor1) ;

- fileinfor1 : ARRAY [0..20] OF File_Information ;
- File_Information是PMC_FileManage库自带的，是一个文件信息结构体。
- 该指令在控制器“UsrData”文件夹内进行操作，不支持在线仿真。

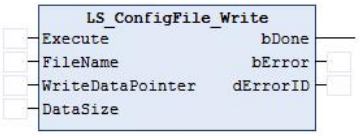
7.5 UsrConfig内文件操作指令

7.5.1 指令列表

指令类别	名称	FB/FC	功能
UsrConfig内文件操作指令	LS_ConfigFile_Write	FB	文件写入指令
	LS_ConfigFile_WriteAppend	FB	文件追加写入指令
	LS_ConfigFile_Read	FB	文件读取指令
	LS_ConfigFile_Del	FB	文件删除指令
	LS_ConfigFile_GetDirectoryFile	FB	获取目录内所有文件信息指令
	LS_ConfigFile_CopyFromUDisk	FB	从U盘拷贝文件指令
	LS_ConfigFile_CopyToUDisk	FB	拷贝文件至U盘指令

7.5.2 LS_ConfigFile_Write

- **指令格式**

指令	名称	FB/FC	LD 表现	ST 表现	库
LS_ConfigFile_Write	文件写入指令	FB		LS_ConfigFile_Write_0(Execute:= , FileName:= , WriteDataPointer:= , DataSize:= , bDone=> , bError=> , dErrorID=>);	PMC_FileManage库

- **相关变量**

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Execute	启动	BOOL	FALSE-TRUE	FALSE	启动信号；TRUE时使能功能块，上升沿有效；

FileName	文件名	STRING（32）	遵照数据类型		新建立文件名称
WriteDataPointer	数据内容的指针	POINTER TO BYTE	遵照数据类型	0	写入文件数据内容的地址
DataSize	数据大小	DWORD	遵照数据类型	0	新建文件大小，单位是字节

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
bDone	返回值	BOOL	FALSE-TRUE	FALSE	完成信号；True-写入文件完成；
bError	错误	BOOL	FALSE-TRUE	FALSE	错误状态；FALSE-没有错误；
dErrorID	错误代码	UDINT	遵照数据类型	0	错误代码

	布尔	位串				整数								实数		时刻、持续时间、日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Execute	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
FileName	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√
WriteDataPointer	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
DataSize	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
bDone	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
bError	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
dErrorID	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-

- **功能说明：**

  - 在控制器“UsrConfig”文件夹内创建新的文件（文件名称由FileName指定，文件大小由DataSize指定），在文件内写入数据内容（数据内容是WriteDataPointer指向的地址内存储的数据）。若该文件名已存在，旧的文件内容将会被覆盖。

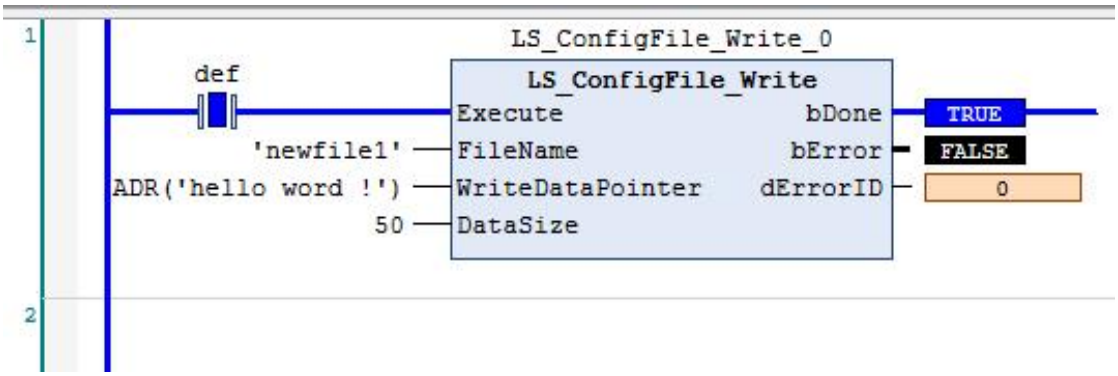
- **程序示例**

ST：当启动信号 abc 变为 True 时，新建文件（文件名称为“newfile0”）并在文件内写

入数据内容(内容为“welcome to leadshine !”)。

```
1  LS_ConfigFile_Write_0(
2       Execute TRUE :=abc TRUE ,
3       FileName 'newfile0' :='newfile0' ,
4       WriteDataPointer 16#B44B7408 := ADR('welcome to leadshine !'),
5       DataSize 50 := 50,
6       bDone TRUE =>bd1 TRUE ,
7       bError FALSE =>be1 FALSE ,
8       dErrorID 0 => de1 0 );
9  RETURN
```

LD：当启动信号 def 变为 True 时，新建文件（newfile1）并在文件内写入数据内容。



- **注意事项**

➢ DataSize是新建立文件的大小，需要比WriteDataPointer指定的数据大小大，否则写入的数据可能丢失。

➢ 注意PLC不支持建立中文名字的文件，文件内容最好不包含中文，否则LS_ConfigFile_Read指令读取到的可能无法识别。

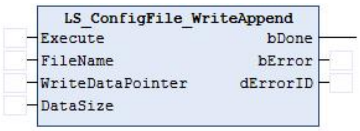
➢ WriteDataPointer是写入文件内的数据内容的地址。

➢ 指令默认在控制器“UsrConfig”文件夹内创建新的文件，不支持在线仿真。

7.5.3 LS_ConfigFile_WriteAppend

- **指令格式**

指令	名称	FB/FC	LD 表现	ST 表现	库

LS_ConfigFile_WriteAppend	文件追加写入指令	FB		LS_ConfigFile_WriteAppend_0(Execute:= , FileName:= , WriteDataPointer:= , DataSize:= , bDone=> , bError=> , dErrorID=>);	PMC_FileManage库

- **相关变量**

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Execute	启动	BOOL	FALSE-TRUE	FALSE	启动信号；TRUE时使能功能块，上升沿触发；
FileName	文件名	STRING（32）	遵照数据类型		待追加写入数据内容的文件名称
WriteDataPointer	追加写入数据内容的指针	POINTER TO BYTE	遵照数据类型	0	追加写入的数据内容的地址
DataSize	数据大小	DWORD	遵照数据类型	0	追加写入的数据大小，单位是字节

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
bDone	返回值	BOOL	FALSE-TRUE	FALSE	完成信号；True-文件追加写入完成；
bError	错误	BOOL	FALSE-TRUE	FALSE	错误状态；FALSE-没有错误；
dErrorID	错误代码	UDINT	遵照数据类型	0	错误代码

	布尔	位串	整数	实数	时刻、持续时间、日期、字符串

	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Execute	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
FileName	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√
WriteDataPointer	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
DataSize	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
bDone	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
bError	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
dErrorID	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-

- **功能说明：**

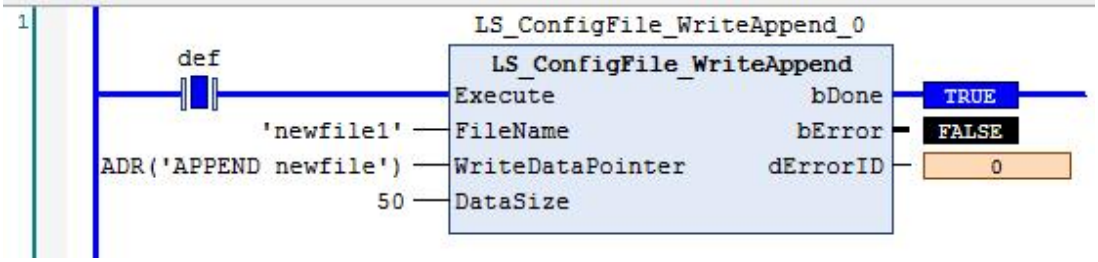
  - 在控制器“UsrConfig”文件夹内，对FileName指定的文件以追加（扩大原文件大小）的方式写入WriteDataPointer指定的数据内容。

- **程序示例**

ST：当启动信号 abc 变为 True 时，在指定文件（newfile0）中追加写入数据内容（APPEND newfile）。

```
LS_ConfigFile_WriteAppend_0(
    Execute TRUE := abc TRUE,
    FileName 'newfile0' := 'newfile0',
    WriteDataPointer 16#B44B6E76 := ADR('APPEND newfile'),
    DataSize 50 := 50,
    bDone TRUE => BD1 TRUE,
    bError FALSE => BE1 FALSE,
    dErrorID 0 =>DE1 0 );
RETURN
```

LD：当启动信号 def 变为 True 时，在指定文件（newfile1）中追加写入数据内容（APPEND newfile）。

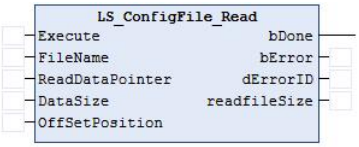


- **注意事项**

- DataSize是追加写入的数据大小，需要比WriteDataPointer指定的数据大小大，否则追加写入的数据可能丢失。
- Execute上升沿触发，每次执行文件追加写入指令后，文件大小变为原文件大小加上DataSize的大小；追加写入的数据从增加的字节开始写入，可以用LS_ConfigFile_Read指令读取（设置偏移值OffSetPosition，从新增加的字节处开始读取）。
- 注意PLC不支持建立中文名字的文件，文件内容最好不包含中文，否则LS_ConfigFile_Read指令读取到的可能无法识别。
- WriteDataPointer是追加写入的数据内容的地址。
- 如果在控制器“UsrConfig”文件夹内没有FileName指定的文件名的文件，则会新建一个文件，然后写入数据内容。
- 该指令在控制器“UsrConfig”文件夹内进行操作，不支持在线仿真。

7.5.4 LS_ConfigFile_Read

- **指令格式**

指令	名称	FB/FC	LD 表现	ST 表现	库
LS_ConfigFile_Read	文件读取指令	FB		LS_ConfigFile_Read_0(Execute:= , FileName:= , ReadDataPointer:= , DataSize:= ,	PMC_FileManage库

				OffSetPosition:= , bDone=> , bError=> , dErrorID=> , readfileSize=>);	

- **相关变量**

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Execute	启动	BOOL	FALSE-TRUE	FALSE	启动信号；TRUE时使能功能块，上升沿触发；
FileName	文件名	STRING（32）	遵照数据类型		要读取数据内容的文件名称
ReadDataPoint	数据接收指针	POINTER TO BYTE	遵照数据类型		将读取到的数据内容保存到指针指向的地址
DataSize	数据大小	DWORD	遵照数据类型	0	最大能够读取的数据大小
OffSetPosition	位置偏移	DWORD	遵照数据类型	0	从指定字节之后开始读取数据

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
bDone	返回值	BOOL	FALSE-TRUE	FALSE	完成信号；True-读取文件完成；
bError	错误	BOOL	FALSE-TRUE	FALSE	错误状态；FALSE-没有错误；
dErrorID	错误代码	UDINT	遵照数据类型	0	错误代码
readfileSize	读取的文件大小	DWORD	遵照数据类型	0	实际读取到的数据大小

	布尔	位串				整数								实数		时刻、持续时间、日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Execute	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
FileName	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√

ReadDataPoint	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
DataSize	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
OffSetPosition	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
bDone	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
bError	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
dErrorID	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-
readfileSize	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

- **功能说明：**

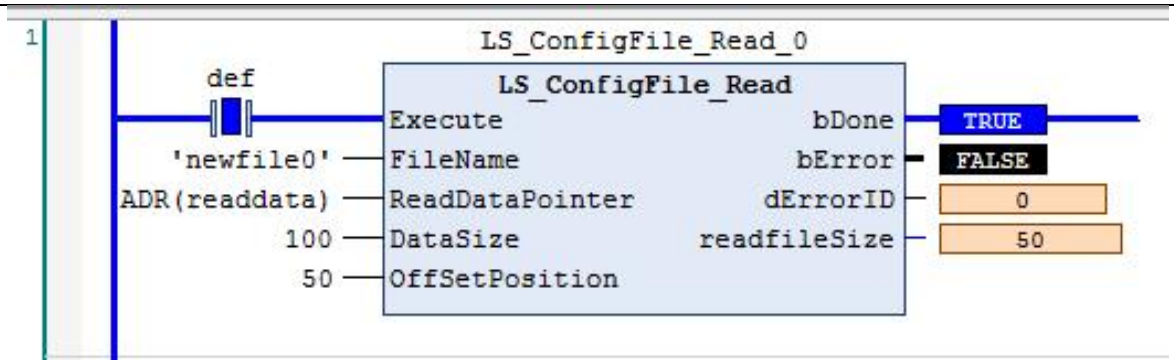
  - 在控制器“UsrConfig”文件夹内读取FileName所指定文件名称的文件中的数据内容，读取的数据内容存储在ReadDataPointer所指定的地址中。

- **程序示例**

ST：当启动信号 abc 变为 True 时，读取指定文件（newfile0）的数据内容保存到 ReadDataPoint 所指定的地址(data)。

```
1  LS_ConfigFile_Read_0(
2      Execute TRUE :=abc TRUE ,
3      FileName 'newfile0' := 'newfile0',
4      ReadDataPointer 16#B44B0F60 :=ADR(data 'welcome to ▶ ) ,
5      DataSize 100 := 100,
6      OffSetPosition 0 := 0,
7      bDone TRUE =>bdl TRUE ,
8      bError FALSE =>bel FALSE ,
9      dErrorID 0 =>del 0 ,
10     readfileSize 100 =>rel 100 );
11 RETURN
```

LD：当启动信号 def 变为 True 时，读取指定文件（newfile0）的数据内容保存到 ReadDataPoint 所指定的地址（readdata）；偏移值 OffSetPosition 设置为 50，即从文件（newfile0）的第 51 个字节开始读取数据。



- **注意事项**

➢ DataSize是最大能读取到的数据大小（要比被读取文件大，否则数据可能读取不完整），readfileSize是实际读取到的数据大小（取决于文件大小与DataSize）。

➢ OffSetPosition是从指定的字节后开始读取文件数据内容，该值设为0时即完整读取整个文件内容。

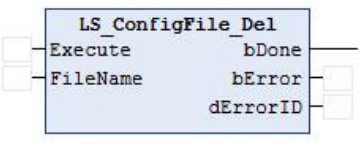
➢ 读取到的文件数据内容存储在ReadDataPointer所指定的地址中。

➢ 注意PLC不支持建立中文名字的文件，文件内容最好不包含中文，否则LS_FileRead指令读取到的可能无法识别。

➢ 该指令在控制器“UsrConfig”文件夹内进行操作，不支持在线仿真。

7.5.5 LS_ConfigFile_Del

- **指令格式**

指令	名称	FB/ FC	LD 表现	ST 表现	库
LS_ConfigFile_Del	文件删除指令	FB		LS_ConfigFile_Del_0(Execute:= , FileName:= , bDone=> , bError=> , dErrorID=>) ;	PMC_FileManage库

- **相关变量**

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Execute	启动	BOOL	FALSE-TRUE	FALSE	启动信号；TRUE时使能功能块，上升沿触发；
FileName	文件名	STRING（32）	遵照数据类型		需要删除的文件名称

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
bDone	返回值	BOOL	FALSE-TRUE	FALSE	完成信号；True-文件删除完成；
bError	错误	BOOL	FALSE-TRUE	FALSE	错误状态；FALSE-没有错误；
dErrorID	错误代码	UDINT	遵照数据类型	0	错误代码

	布尔	位串				整数								实数		时刻、持续时间、日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Execute	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
FileName	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√
bDone	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
bError	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
dErrorID	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-

- **功能说明：**

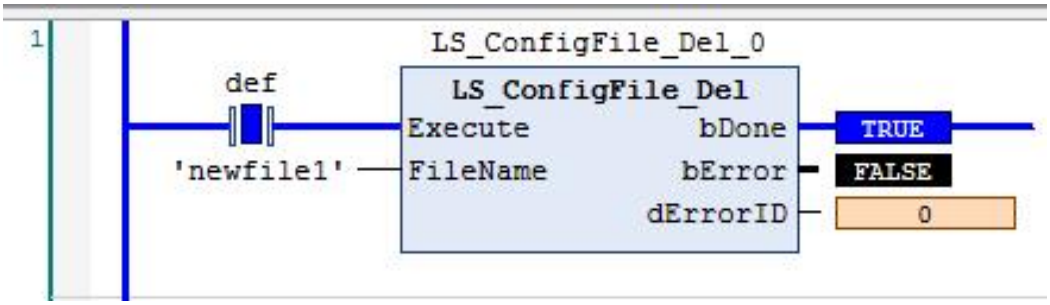
  - 在控制器“UsrConfig”文件夹内删除FileName所指定名称的文件。

- **程序示例**

ST：当启动信号 abc 变为 True 时，将指定名称的文件（newfile0）删除。

```
1  LS_ConfigFile_Del_0(
2      Execute TRUE :=abc TRUE ,
3      FileName 'newfile0' := 'newfile0',
4      bDone TRUE => bdl TRUE,
5      bError FALSE =>bel FALSE ,
6      dErrorID 0 =>del 0 );
7  RETURN
```

LD：当启动信号 def 变为 True 时，将指定名称的文件（newfile1）删除。



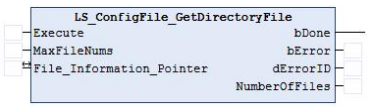
- **注意事项**

➢ 如果FileName所指定名称的文件在控制器“UsrConfig”文件夹内不存在，运行LS_FileDelete指令后会报错。

➢ 该指令在控制器“UsrConfig”文件夹内进行操作，不支持在线仿真。

7.5.6 LS_ConfigFile_GetDirectoryFile

- **指令格式**

指令	名称	FB/FC	LD 表现	ST 表现	库
LS_ConfigFile_GetDirectoryFile	获取目录内所有文件信息指令	FB		LS_ConfigFile_GetDirectoryFile_0(Execute:= , MaxFileNums := , File_Information_Pointer:= , bDone=> , bError=> ,	PMC_FileManage库

				dErrorID=> , NumberOfFiles=>);	

- **相关变量**

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Execute	启动	BOOL	FALSE-TRUE	FALSE	启动信号；TRUE 时使能功能块，上升沿触发；
MaxFileNums	最大文件数	INT	遵照数据类型	20	最大能读取的文件个数
File_Information_Point	文件信息存储的指针	POINTER TO File_Information	遵照数据类型		读取的文件信息存储的地址

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
bDone	返回值	BOOL	FALSE-TRUE	FALSE	完成信号；True-获取目录内所有文件信息完成；
bError	错误	BOOL	FALSE-TRUE	FALSE	错误状态；FALSE-没有错误；
dErrorID	错误代码	UDINT	遵照数据类型	0	错误代码
NumberOfFiles	读取文件个数	INT	遵照数据类型	0	实际读取到信息的文件个数

	布尔	位串				整数								实数		时刻、持续时间、日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Execute	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
MaxFileNums	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-
File_Information_Pointer	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

bDone	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
bError	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
dErrorID	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-
NumberOfFiles	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-

- **功能说明：**

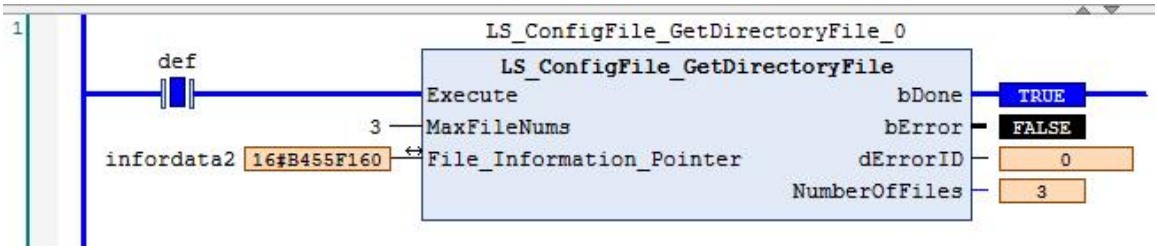
  - 在控制器“UsrConfig”文件夹内，获取该目录下的所有文件的信息（文件信息包括文件名，文件创建时间，文件最近修改时间，文件大小）。

- **程序示例**

ST：当启动信号abc变为True时，获取控制器“UsrConfig”文件夹内，所有文件的信息。

```
1  LS_ConfigFile_GetDirectoryFile_0(
2       Execute TRUE := abc TRUE,
3       MaxFileNums 3 := 3,
4       File_Information_Pointer 16#B455E6E0 :=infordata1 16#B455E6E0 ,
5       bDone TRUE =>bd1 TRUE ,
6       bError FALSE =>be1 FALSE ,
7       dErrorID 0 =>de1 0 ,
8       NumberOfFiles 3 =>num1 3 );
9  RETURN
```

LD：当启动信号def变为True时，获取控制器“UsrConfig”文件夹内，所有文件的信息。



- **注意事项**

➢ 上述ST例子中定义变量：

infordata1: POINTER TO File_Information := ADR(fileinfor1) ;

➢ fileinfor1 : ARRAY [0..20] OF File_Information ;

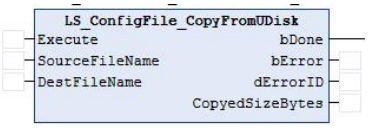
读取控制器“UsrConfig”文件夹内目录下所有文件时，由于读到的是多个文件的信息，定义的文件信息指针File_Information_Pointer必须是指向文件信息结构体数组的，而不是

一个文件信息结构体。

- File_Information是PMC_FileManage库自带的，是一个文件信息结构体
- 最大读取的文件个数由MaxFileNums指定。
- NumberOfFiles是实际读取到的文件个数，取决于“UsrConfig”文件夹内存在的文件数和最大读取的文件个数MaxFileNums。
- 读取到的文件信息分为四个字符串，每个字符串占31个字节；若文件名称过长，超过31字节会溢出，导致识别不出文件名称。
- 该指令在控制器“UsrConfig”文件夹内进行操作，不支持在线仿真。

7.5.7 LS_ConfigFile_CopyFromUDisk

- **指令格式**

指令	名称	FB/FC	LD 表现	ST 表现	库
LS_ConfigFile_CopyFromUDisk	从U盘拷贝文件指令	FB		LS_ConfigFile_CopyFromUDisk_0(Execute:= , SourceFileName:= , DestFileName := , bDone=> , bError=> , dErrorID=> , CopyedSizeBytes=>);	PMC_FileManage库

- **相关变量**

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Execute	启动	BOOL	FALSE-TRUE	FALSE	启动信号；TRUE时使能功能块，上升沿触发；
SourceFileName	源文件	STRING	遵照数据类		U盘中需要拷贝的源文件

	名	（32）	型		名称
DestFileName	目标文件名	STRING（32）	遵照数据类型		拷贝后在控制器“UsrConfig”文件夹内生成的新文件名称

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
bDone	返回值	BOOL	FALSE-TRUE	FALSE	完成信号；True-从U盘拷贝文件至控制器“UsrConfig”文件夹内完成；
bError	错误	BOOL	FALSE-TRUE	FALSE	错误状态；FALSE-没有错误；
dErrorID	错误代码	UDINT	遵照数据类型	0	错误代码
CopyedSizeBytes	拷贝的字节数	DWORD	遵照数据类型	0	拷贝后生成的新文件大小

	布尔	位串				整数								实数		时刻、持续时间、日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Execute	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
SourceFileName	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√
DestFileName	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√
bDone	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
bError	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
dErrorID	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-
CopyedSizeBytes	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

- **功能说明：**

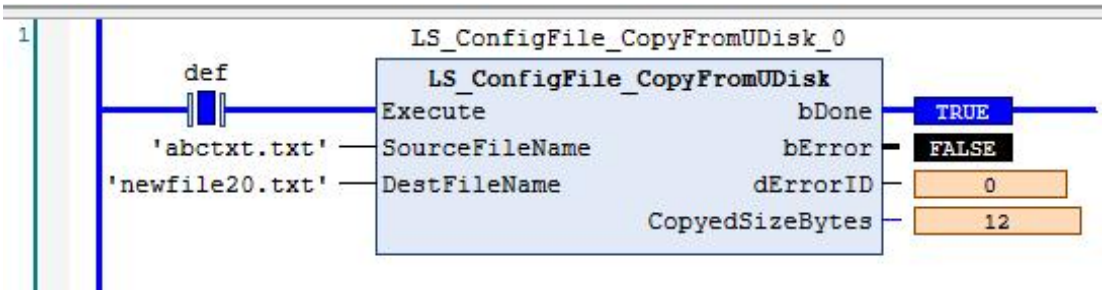
  - 从U盘根目录拷贝文件（文件名称由SourceFileName指定）至控制器“UsrConfig”文件夹内（拷贝生成的新文件名称由DestFileName指定）。

- **程序示例**

ST：当启动信号 abc 变为 True 时，从 U 盘根目录拷贝文件（newfile.txt）至控制器“UsrConfig”文件夹内（新生成文件 newfile10）。

```
1  LS_ConfigFile_CopyFromUDisk_0(
2      Execute TRUE :=abc TRUE ,
3      SourceFileName 'newfile.tx ▸ :='newfile.txt' ,
4      DestFileName 'newfile10' := 'newfile10',
5      bDone TRUE =>bdl TRUE ,
6      bError FALSE =>bel FALSE ,
7      dErrorID 0 =>del 0 ,
8      CopyedSizeBytes 24 =>col 24 );
9  RETURN
```

LD：当启动信号 def 变为 True 时，从 U 盘根目录拷贝文件（abctxt.txt）至控制器“UsrConfig”文件夹内（新生成文件 newfile20.txt）。



- **注意事项**

- 如果SourceFileName指定的源文件在U盘根目录中不存在，运行LS_ConfigFile_CopyFromUDisk指令后将会报错；错误代码16。
- 如果DestFileName指定的目标文件在控制器“UsrConfig”文件夹内已存在，运行LS_ConfigFile_CopyFromUDisk指令后，旧文件的数据内容会被覆盖。
- 注意SourceFileName指定的源文件需要带格式后缀；如ST例子中，在U盘中文件的名字为(newfile)，其格式为(.txt)。
- 注意PLC不支持建立中文名字的文件，文件中的内容最好不用中文，PLC识别不了。
- 该指令在控制器“UsrConfig”文件夹内进行操作，不支持在线仿真。

7.5.8 LS_ConfigFile_CopyToUDisk

- **指令格式**

指令	名称	FB/FC	LD 表现	ST 表现	库
LS_ConfigFile_CopyToUDisk	拷贝文件至U盘指令	FB		LS_ConfigFile_CopyToUDisk_0(Execute:= , SourceFileName:= , DestFileName := , bDone=> , bError=> , dErrorID=> , CopyedSizeBytes=>);	PMC_File Manage库

- **相关变量**

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Execute	启动	BOOL	FALSE-TRUE	FALSE	启动信号；TRUE时使能功能块，上升沿触发；
SourceFileName	源文件名	STRING（32）	遵照数据类型		控制器“UsrConfig”文件夹内需要拷贝的源文件名称
DestFileName	目标文件名	STRING（32）	遵照数据类型		拷贝后在U盘生成的新文件名称

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
bDone	返回值	BOOL	FALSE-TRUE	FALSE	完成信号；True-拷贝文件至U盘完成；
bError	错误	BOOL	FALSE-TRUE	FALSE	错误状态；FALSE-没有错误；
dErrorID	错误代码	UDINT	遵照数据类型	0	错误代码
CopyedSizeBytes	拷贝的字节数	DWORD	遵照数据类型	0	拷贝后生成的新文件大小

	布尔	位串				整数								实数		时刻、持续时间、日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Execute	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
SourceFileName	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√
DestFileName	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√
bDone	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
bError	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
dErrorID	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-
CopyedSizeBytes	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

- **功能说明：**

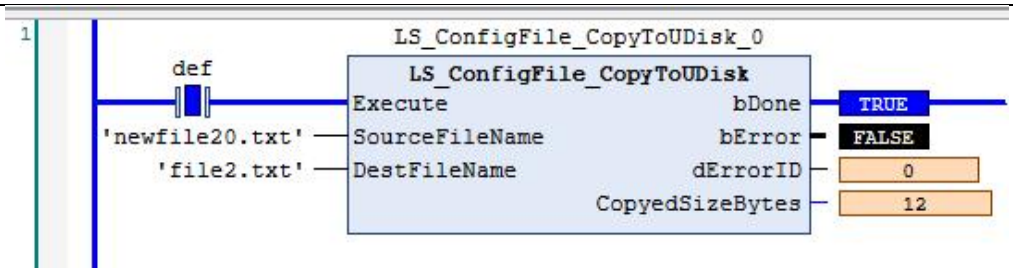
  - 从控制器“UsrConfig”文件夹内拷贝文件（文件名称由SourceFileName指定）至U盘根目录（拷贝生成的新文件名称由DestFileName指定）。

- **程序示例**

ST：当启动信号 abc 变为 True 时，从控制器“UsrConfig”文件夹内拷贝文件（newlife0）至 U 盘根目录（新生成文件 file1.txt）。

```
LS_ConfigFile_CopyToUDisk_0(
    Execute TRUE := abc TRUE,
    SourceFileName 'newlife0' := 'newlife0',
    DestFileName 'file1.txt' :='file1.txt' ,
    bDone TRUE => bd1 TRUE,
    bError FALSE =>be1 FALSE ,
    dErrorID 0 =>de1 0 ,
    CopyedSizeBytes 50 =>co1 50 );
RETURN
```

LD：当启动信号 def 变为 True 时，从控制器“UsrConfig”文件夹内拷贝文件（newfile20.txt）至 U 盘根目录（新生成文件 file2.txt）。



- **注意事项**

➢ 如果SourceFileName指定的源文件在控制器“UsrConfig”文件夹内不存在，运行LS_ConfigFile_CopyToUDisk指令后将会报错；错误代码16。

➢ 如果DestFileName指定的目标文件在U盘根目录内已存在，运行LS_ConfigFile_CopyToUDisk指令后，旧文件的数据内容会被覆盖。

➢ 注意DestFileName指定的目标文件如果带格式后缀，在U盘中会自动生成对应格式的文件。

➢ 注意PLC不支持建立中文名字的文件，文件内容最好不带中文，否则复制到U盘中会是乱码。

➢ 该指令在控制器“UsrConfig”文件夹内进行操作，不支持在线仿真。

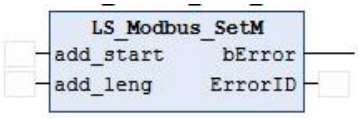
7.6HMI库变量批量操作

7.6.1 指令列表

指令类别	名称	FB/FC	功能
HMI库变量批量操作	LS_Modbus_SetM	FB	BOOL变量批量置位指令
	LS_Modbus_ResetM	FB	BOOL变量批量复位指令
	LS_Modbus_SetByteToM	FB	BYTE变量批量置位指令
	LS_Modbus_SetDWS	FB	INT变量批量置值指令
	LS_Modbus_SetDWU	FB	UINT变量批量置值指令
	LS_Modbus_SetDDS	FB	DINT变量批量置值指令
	LS_Modbus_SetDDU	FB	UDINT变量批量置值指令
	LS_Modbus_SetDR	FB	REAL变量批量置值指令
	LS_Modbus_SetDS	FB	STRING变量批量置值指令

7.6.2 LS_Modbus_SetM

- 指令格式

指令	名称	FB/FC	LD 表现	ST 表现	库
LS_Modbus_SetM	BOOL变量批量置位指令	FB		LS_Modbus_SetM_0(add_start:= , add_leng:= , bError=> , ErrorID=>);	PMC_Communication库

- 相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
add_start	变量起始地址	INT	0-3999	0	变量起始地址
add_leng	变量个数	INT	1-4000	0	变量个数

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
bError	错误	BOOL	FALSE-TRUE	FALSE	错误状态；FALSE-没有错误；

dErrorID	错误代码	INT	遵照数据类型	0	错误代码

	布尔	位串				整数								实数		时刻、持续时间、日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
add_start	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-
add_leng	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-
bError	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
dErrorID	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-

- **功能说明：**

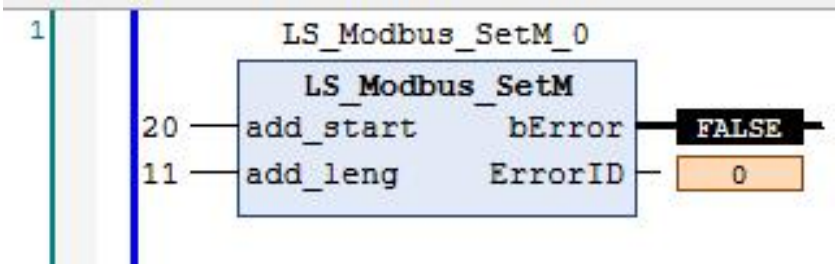
  - 批量置位BOOL型变量M0~M3999的值，对应Modbus地址0x_1~0x_4000。

- **程序示例**

ST：将 M5 至 M14 的值设置为 true；（从 M5 开始，长度为 10）程序如下：

```
1  LS_Modbus_SetM_0
2  (add_start  5  :=5 ,
3   add_leng  10  :=10 ,
4   bError FALSE =>bel FALSE ,
5   ErrorID  0  => erl  0  );
6  RETURN
```

LD：将 M20 至 M30 的值设置为 true。（从 M20 开始，长度为 11）程序如下：

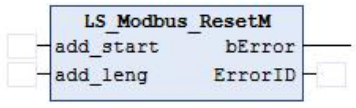


- **注意事项**

- HMI库定义的BOOL型变量M在控制器内部的地址（%M）与Modbus地址的对应关系，可以在“左侧目录树-库管理器-PMC_Communication库-HMI-GVL_Modbus_Register_0x4x输入输出”处查询到。
- LS_Modbus_SetM指令是对定义的BOOL型变量M进行批量置位处理；可以同时对多个连续的BOOL型变量M置TRUE。
- 如上述LD例子中将M[20]至M[30]这11个BOOL型变量M置TRUE。

7.6.3 LS_Modbus_ResetM

- **指令格式**

指令	名称	FB/FC	LD 表现	ST 表现	库
LS_Modbus_ResetM	BOOL变量批量复位指令	FB		LS_Modbus_ResetM_0(add_start:= , add_leng:= , bError=> , ErrorID=>);	PMC_Communication库

- **相关变量**

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
add_start	变量起始地址	INT	0-3999	0	变量起始地址
add_leng	变量个数	INT	1-4000	0	变量个数

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
bError	错误	BOOL	FALSE-TRUE	FALSE	错误状态；FALSE-没有错误；
dErrorID	错误代码	INT	遵照数据类型	0	错误代码

	布	位串	整数	实数	时刻、持续时间、

	尔															日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
add_start	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-
add_leng	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-
bError	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
dErrorID	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-

- **功能说明：**

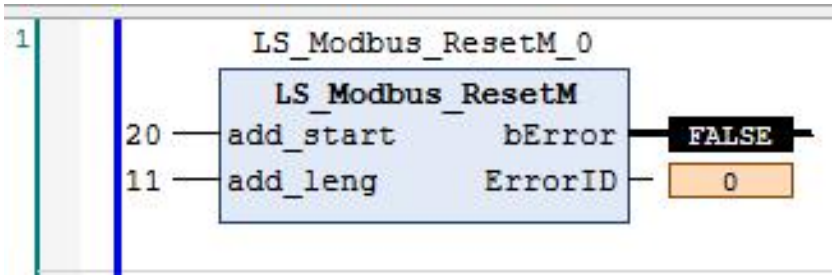
  - 批量复位BOOL型变量M0~M3999的值，对应Modbus地址0x_1~0x_4000。

- **程序示例**

ST：将M1000至M1049的值设置为FALSE；（从M1000开始，长度为50）程序如下：

```
1  LS_Modbus_ResetM_0(
2  add_start 1000 := 1000,
3  add_leng 50 := 50,
4  bError FALSE => bel FALSE,
5  ErrorID 0 =>erl 0 );
6  RETURN
```

LD：将M20至M30的值设置为FALSE。（从M20开始，长度为11）程序如下：



- **注意事项**

  - HMI库定义的BOOL型变量M在控制器内部的地址（%M）与Modbus地址的对应关系，可以在“左侧目录树-库管理器-PMC_Communication库-HMI-GVL_Modbus_ Register_0x4x

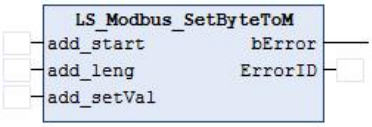
输入输出”处查询到。

- LS_Modbus_ResetM指令是对定义的BOOL型变量M进行批量复位处理；可以同时对多个连续的BOOL型变量M置FALSE。

  如上述LD例子中将M[20]至M[30]这11个BOOL型变量M置FALSE。

7.6.4 LS_Modbus_SetByteToM

- **指令格式**

指令	名称	FB/FC	LD 表现	ST 表现	库
LS_Modbus_SetByteToM	BYTE变量批量置位指令	FB		LS_Modbus_SetByteToM_0(add_start:=, add_leng:=, add_setVal:=, bError=>, ErrorID=>); 	PMC_Communication库

- **相关变量**

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
add_start	变量起始地址	INT	0-499	0	变量起始地址
add_leng	变量个数	INT	1-500	0	变量个数
add_setVal	设置的值	BYTE	遵照数据类型	0	设置的值

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
bError	错误	BOOL	FALSE-TRUE	FALSE	错误状态；FALSE-没有错误；
dErrorID	错误代码	INT	遵照数据类型	0	错误代码

	布尔	位串				整数								实数		时刻、持续时间、日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
add_start	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-
add_leng	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-
add_setVal	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
bError	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
dErrorID	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-

- **功能说明：**

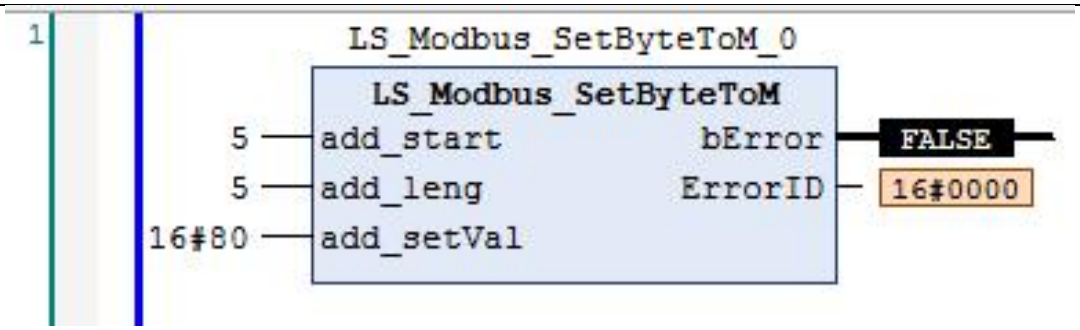
  - 按位设置字节型变量ByteToM[0]~ ByteToM[499]的值，对应Modbus地址0x_4001~0x_8000。实际上是通过给定义的字节型变量ByteToM赋值，达到指定位置TRUE。

- **程序示例**

ST：将 ByteToM[0]至 ByteToM[4]的每位设置为 TRUE；（从 ByteToM[0]开始，长度为 5）程序如下：

```
1  LS_Modbus_SetByteToM_0(
2      add_start 16#0000 :=0 ,
3      add_leng 16#0005 := 5,
4      add_setVal 16#FF := 16#FF,
5      bError FALSE =>bel FALSE ,
6      ErrorID 16#0000 => erl 16#0000 );
7      RETURN
```

LD：将 ByteToM[5]至 ByteToM[9]每个字节的最高位设置为 TRUE；（从 ByteToM[5]开始，长度为 5）程序如下：



- **注意事项**

➢ HMI库定义的BYTE型变量ByteToM在控制器内部的地址（%M）与Modbus地址的对应关系，可以在

“左侧目录树-库管理器-PMC_Communication库-HMI-GVL_Modbus_Register_0x4x输入输出”处查询到。

➢ add_setVal是对定义的BYTE型变量ByteToM的设置值；

比如上述ST例子中将ByteToM[0]这个BYTE的每位都设置为1，即2#11111111，也就是给ByteToM[0]赋值16#FF；

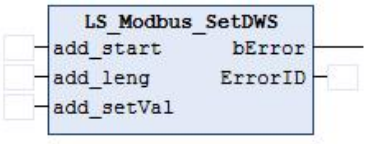
比如上述LD例子中将ByteToM[5]这个BYTE的最高位设置为1，即2#10000000，也就是给ByteToM[5]赋值16#80。

➢ LS_Modbus_SetByteToM指令是对定义的BYTE型变量ByteToM进行批量置位处理；可以同时对多个连续的BYTE型变量ByteToM设置同一个值。

如上述LD例子中将ByteToM[5]至ByteToM[9]这5个BYTE型变量ByteToM的值都设置成16#80。

7.6.5 LS_Modbus_SetDWS

- **指令格式**

指令	名称	FB/FC	LD 表现	ST 表现	库
LS_Modbus_SetDWS	INT变量批量置值指	FB		LS_Modbus_SetDWS_0(add_start:= , add_leng:= ,	PMC_Communication库

	令			add_setVal:= , bError=> , ErrorID=>);	

- **相关变量**

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
add_start	变量起始地址	INT	0-4999	0	变量起始地址
add_leng	变量个数	INT	1-5000	0	变量个数
add_setVal	设置的值	INT	遵照数据类型	0	设置的值

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
bError	错误	BOOL	FALSE-TRUE	FALSE	错误状态；FALSE-没有错误；
dErrorID	错误代码	INT	遵照数据类型	0	错误代码

	布尔	位串				整数								实数		时刻、持续时间、日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
add_start	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-
add_leng	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-
add_setVal	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-
bError	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
dErrorID	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-

- **功能说明：**

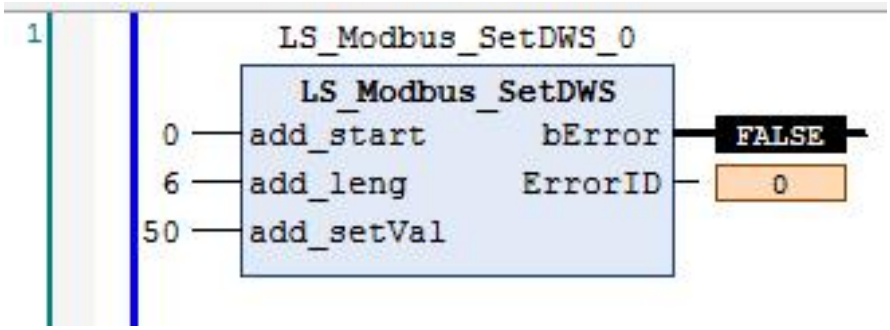
  - 设置INT型变量DWS [0]~ DWS[4999]的值，对应Modbus地址4x_1~4x_5000。

- **程序示例**

ST：将 DWS[6]至 DWS[15]的值设置为 89；（从 DWS[6]开始，长度为 10）程序如下：

```
1  LS_Modbus_SetDWS_0(
2      add_start  6   := 6,
3      add_leng   10  := 10,
4      add_setVal 89  := 89,
5      bError FALSE =>bel FALSE ,
6      ErrorID  0   => errl  0  );
7      RETURN
```

LD：将 DWS[0]至 DWS[5] 的值设置为 50；（从 DWS[0]开始，长度为 6）程序如下：



- **注意事项**

➢ HMI库定义的INT型变量DWS在控制器内部的地址（%M）与Modbus地址的对应关系，可以在

“左侧目录树-库管理器-PMC_Communication库-HMI-GVL_Modbus_Register_0x4x输入输出”处查询到。

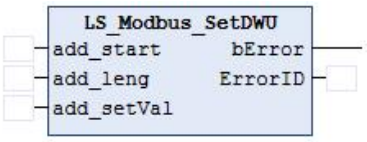
➢ add_setVal是对定义的INT型变量DWS的设置值；

比如上述ST例子中将DWS[6]至DWS[15]共10个INT型变量的值都设置为89。

➢ 该指令实际上是批量给HMI库定义的INT型变量DWS赋值。

7.6.6 LS_Modbus_SetDWU

- **指令格式**

指令	名称	FB/FC	LD 表现	ST 表现	库
LS_Modbus_SetDWU	UINT 变量批量置值指令	FB		LS_Modbus_SetDWU_0(add_start:= , add_leng:= , add_setVal:= , bError=> , ErrorID=>);	PMC_Communication库

- **相关变量**

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
add_start	变量起始地址	INT	0-4999	0	变量起始地址
add_leng	变量个数	INT	1-5000	0	变量个数
add_setVal	设置的值	UINT	遵照数据类型	0	设置的值

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
bError	错误	BOOL	FALSE-TRUE	FALSE	错误状态；FALSE-没有错误；
dErrorID	错误代码	INT	遵照数据类型	0	错误代码

	布尔	位串				整数								实数		时刻、持续时间、日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
add_start	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-
add_leng	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-
add_setVal	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-
bError	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
dErrorID	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-

- **功能说明：**

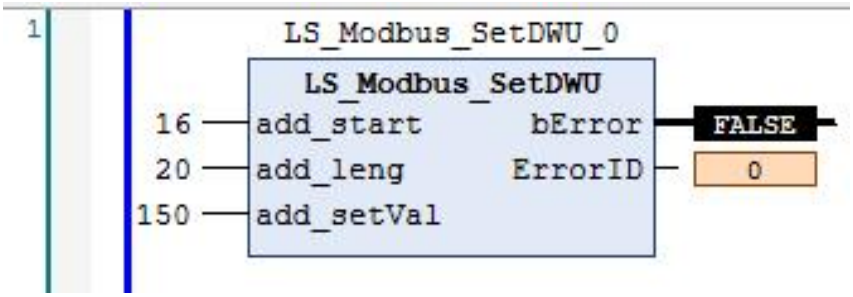
  - 设置UINT型变量DWU [0]~ DWU[4999]的值，对应Modbus地址4x _5001~4x_10000。

- **程序示例**

ST：将 DWU[0]至 DWU[5]的值设置为 200；（从 DWU[0]开始，长度为 6）程序如下：

```
1  LS_Modbus_SetDWU_0(
2       add_start  0   := 0,
3       add_leng   6   := 6,
4       add_setVal 200 := 200,
5       bError FALSE =>BE1 FALSE ,
6       ErrorID  0  => ERR1  0 );
7  RETURN
```

LD：将 DWU[16]至 DWU[35] 的值设置为 150；（从 DWU[16]开始，长度为 20）程序如下：



- **注意事项**

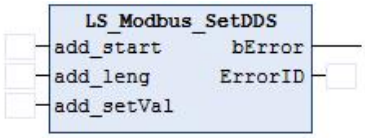
- HMI库定义的UINT型变量DWU在控制器内部的地址（%M）与Modbus地址的对应关系，可以在
  “左侧目录树-库管理器-PMC_Communication库-HMI-GVL_Modbus_Register_0x4x输入输出”处查询到。

- add_setVal是对定义的UINT型变量DWU的设置值；
  比如上述ST例子中将DWU[0]至DWU[5]共6个UINT型变量的值都设置为200。

- 该指令实际上是批量给HMI库定义的UINT型变量DWU赋值。

7.6.7 LS_Modbus_SetDDS

- 指令格式

指令	名称	FB/FC	LD 表现	ST 表现	库
LS_Modbus_SetDDS	DINT 变量批量置值指令	FB		LS_Modbus_SetDDS_0(add_start:= , add_leng:= , add_setVal:= , bError=> , ErrorID=>);	PMC_Communication库

- 相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
add_start	变量起始地址	INT	0-4999	0	变量起始地址
add_leng	变量个数	INT	1-5000	0	变量个数
add_setVal	设置的值	DINT	遵照数据类型	0	设置的值

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
bError	错误	BOOL	FALSE-TRUE	FALSE	错误状态；FALSE-没有错误；
dErrorID	错误代码	INT	遵照数据类型	0	错误代码

	布尔	位串				整数								实数		时刻、持续时间、日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
add_start	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-
add_leng	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-
add_setVal	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-

bError	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
dErrorID	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-

- **功能说明：**

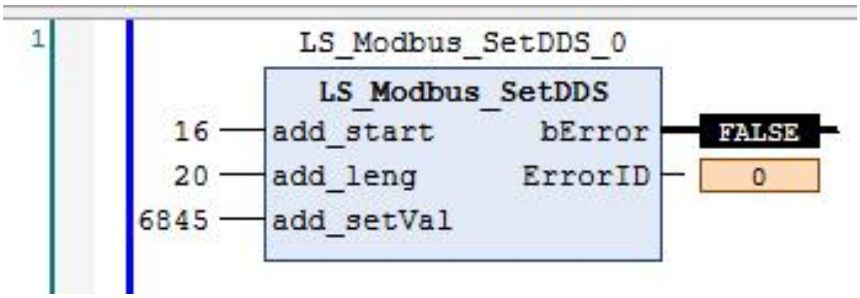
  - 设置DINT型变量DDS [0]~ DDS[4999]的值，对应Modbus地址4x _10001~4x_19999。

- **程序示例**

ST：将 DDS[0]至 DDS[5]的值设置为-3214；（从 DDS[0]开始，长度为 6）程序如下：

```
1  LS_Modbus_SetDDS_0(
2        add_start  0  := 0,
3        add_leng  6  := 6,
4        add_setVal  -3214  := -3214,
5        bError  FALSE  => bel  FALSE ,
6        ErrorID  0  => errl  0  );
7  RETURN
```

LD：将 DDS[16]至 DDS[35] 的值设置为 6845；（从 DDS[16]开始，长度为 20）程序如下：



- **注意事项**

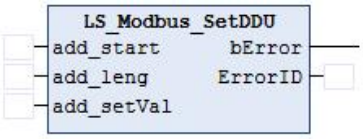
- HMI库定义的DINT型变量DDS在控制器内部的地址（%M）与Modbus地址的对应关系，可以在“左侧目录树-库管理器-PMC_Communication库-HMI-GVL_Modbus_Register_0x4x输入输出”处查询到。

- add_setVal是对定义的DINT型变量DDS的设置值；比如上述ST例子中将DDS[0]至DDS[5]共6个DINT型变量的值都设置为-3214。

- 该指令实际上是批量给HMI库定义的DINT型变量DDS赋值。

7.6.8 LS_Modbus_SetDDU

- **指令格式**

指令	名称	FB/FC	LD 表现	ST 表现	库
LS_Modbus_SetDDU	UDINT 变量批量置值指令	FB		LS_Modbus_SetDDU_0(add_start:= , add_leng:= , add_setVal:= , bError=> , ErrorID=>);	PMC_Communication库

- **相关变量**

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
add_start	变量起始地址	INT	0-4999	0	变量起始地址
add_leng	变量个数	INT	1-5000	0	变量个数
add_setVal	设置的值	UDINT	遵照数据类型	0	设置的值

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
bError	错误	BOOL	FALSE-TRUE	FALSE	错误状态；FALSE-没有错误；
dErrorID	错误代码	INT	遵照数据类型	0	错误代码

	布尔	位串				整数								实数		时刻、持续时间、日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
add_start	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-
add_leng	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-
add_setVal	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-

bError	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
dErrorID	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-

- **功能说明：**

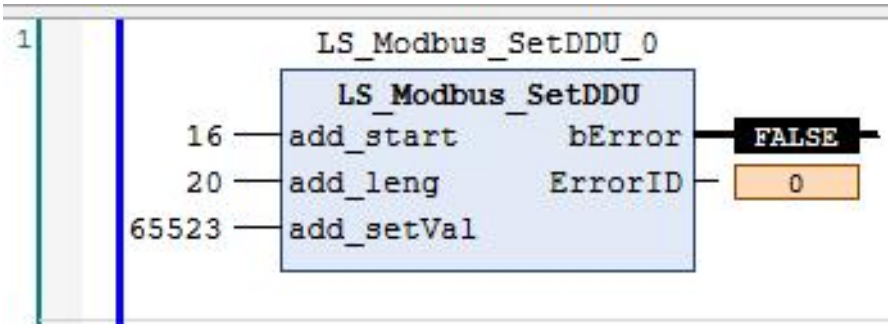
  - 设置UDINT型变量DDU[0]~ DDU[4999]的值，对应Modbus地址4x _20001~4x_29999。

- **程序示例**

ST：将 DDU[0]至 DDU[5]的值设置为 3214；（从 DDU[0]开始，长度为 6）程序如下：

```
1  LS_Modbus_SetDDU_0(
2       add_start  0  := 0,
3       add_leng  6  := 6,
4       add_setVal  3214  := 3214,
5       bError FALSE => BE1 FALSE,
6       ErrorID  0  => ERR1  0  );
7  RETURN
```

LD：将 DDU[16]至 DDU[35] 的值设置为 65523；（从 DDU[16]开始，长度为 20）程序如下：



- **注意事项**

- HMI库定义的UDINT型变量DDU在控制器内部的地址（%M）与Modbus地址的对应关系，可以在

  “左侧目录树-库管理器-PMC_Communication库-HMI-GVL_Modbus_Register_0x4x输入输出”处查询到。

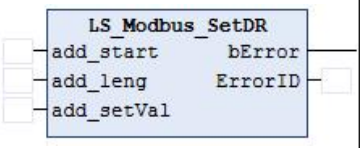
- add_setVal是对定义的UDINT型变量DDU的设置值；

比如上述ST例子中将DDU[0]至DDU[5]共6个UDINT型变量的值都设置为3214。

➢ 该指令实际上是批量给HMI库定义的UDINT型变量DDU赋值。

7.6.9 LS_Modbus_SetDR

- **指令格式**

指令	名称	FB/FC	LD 表现	ST 表现	库
LS_Modbus_SetDR	REAL变量批量置值指令	FB		LS_Modbus_SetDR_0(add_start:= , add_leng:= , add_setVal:= , bError=> , ErrorID=>);	PMC_Communication库

- **相关变量**

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
add_start	变量起始地址	INT	0-2499	0	变量起始地址
add_leng	变量个数	INT	1-2500	0	变量个数
add_setVal	设置的值	REAL	遵照数据类型	0	设置的值

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
bError	错误	BOOL	FALSE-TRUE	FALSE	错误状态；FALSE-没有错误；
dErrorID	错误代码	INT	遵照数据类型	0	错误代码

	布尔	位串				整数								实数		时刻、持续时间、日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
add_start	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-
add_len	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-

g																				
add_setVal	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-
bError	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
dErrorID	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-

- **功能说明：**

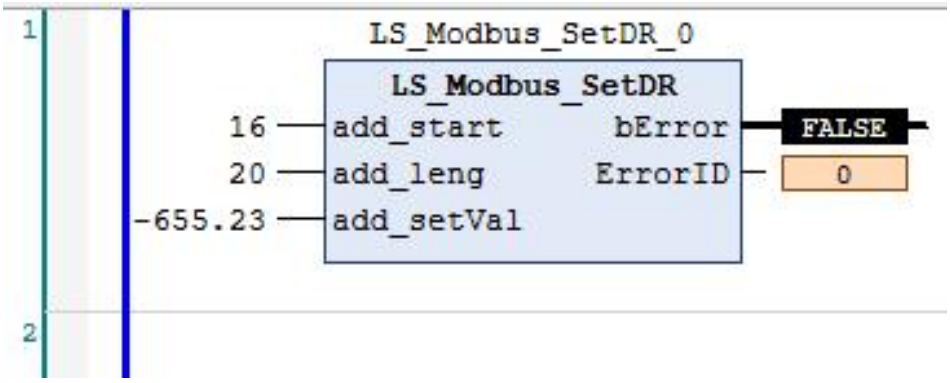
  - 设置REAL型变量DR[0]~ DR[2499]的值，对应Modbus地址4x _30001~4x_34999。

- **程序示例**

ST：将 DR[0]至 DR[5]的值设置为 3214.5；（从 DR[0]开始，长度为 6）程序如下：

```
1  LS_Modbus_SetDR_0(
2        add_start  0  := 0,
3        add_leng  6  := 6,
4        add_setVal  3.21E+03  := 3214.5,
5        bError FALSE => BE1 FALSE,
6        ErrorID  0  => ERR1  0 );
7  RETURN
```

LD：将 DR[16]至 DR[35] 的值设置为-655.23；（从 DR[16]开始，长度为 20）程序如下：



- **注意事项**

  - HMI库定义的REAL型变量DR在控制器内部的地址（%M）与Modbus地址的对应关系，可以在

    “左侧目录树-库管理器-PMC_Communication库-HMI-GVL_Modbus_Register_0x4x输入

输出”处查询到。

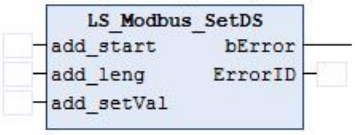
- add_setVal是对定义的REAL型变量DR的设置值；

  比如上述ST例子中将DR[0]至DR[5]共6个REAL型变量的值都设置为3214.5。

- 该指令实际上是批量给HMI库定义的REAL型变量DR赋值。

7.6.10 LS_Modbus_SetDS

- **指令格式**

指令	名称	FB/FC	LD 表现	ST 表现	库
LS_Modbus_SetDS	STRING变量批量置值指令	FB		LS_Modbus_SetDS_0(add_start:= , add_leng:= , add_setVal:= , bError=> , ErrorID=>);	PMC_Communication库

- **相关变量**

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
add_start	变量起始地址	INT	0-149	0	变量起始地址
add_leng	变量个数	INT	1-150	0	变量个数
add_setVal	设置的值	STRING	遵照数据类型	0	设置的值

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
bError	错误	BOOL	FALSE-TRUE	FALSE	错误状态；FALSE-没有错误；
dErrorID	错误代码	INT	遵照数据类型	0	错误代码

	布尔	位串	整数	实数	时刻、持续时间、日期、字符串

	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
add_start	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-
add_leng	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-
add_setVal	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√
bError	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
dErrorID	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-

- **功能说明：**

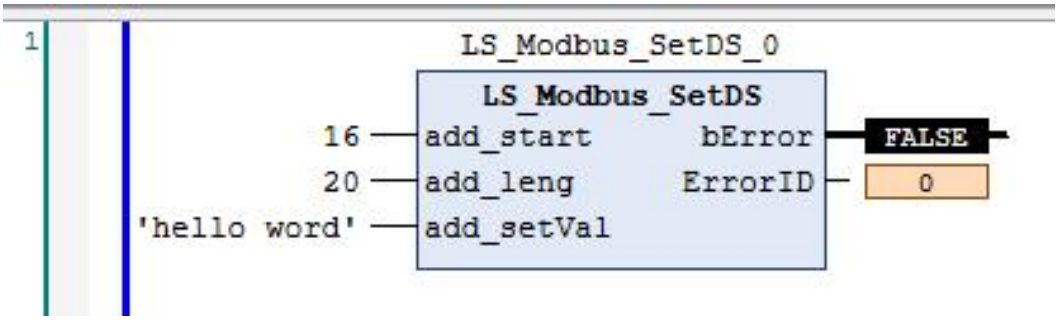
  - 设置STRING型变量DS[0]~ DS[149]的值，对应Modbus地址4x _35001~4x_37385。

- **程序示例**

ST：将 DS[0]至 DS[5]的值设置为’leadshine’；（从 DS[0]开始，长度为 6）程序如下：

```
1  LS_Modbus_SetDS_0(
2       add_start  0  := 0,
3       add_leng  6  := 6,
4       add_setVal  'leadshine'  := 'leadshine',
5       bError FALSE => BE1 FALSE,
6       ErrorID  0  => ERR1  0  );
7  RETURN
```

LD：将 DS[16]至 DS[35] 的值设置为’hello word’；（从 DS[16]开始，长度为 20）程序如下：



- **注意事项**

➢ HMI库定义的STRING型变量DS在控制器内部的地址（%M）与Modbus地址的对应关系，可以在

“左侧目录树-库管理器-PMC_Communication库-HMI-GVL_Modbus_Register_0x4x输入输出”处查询到。

➢ add_setVal是对定义的STRING型变量DS的设置值；

比如上述ST例子中将DS[0]至DS[5]共6个STRING型变量的值都设置为’leadshine’。

➢ 该指令实际上是批量给HMI库定义的STRING型变量DS赋值。

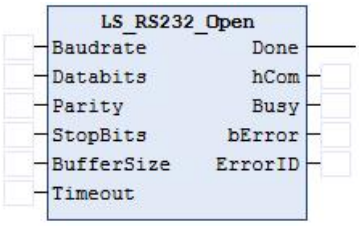
7.7RS232无协议通信

7.7.1 指令列表

指令类别	名称	FB/FC	功能
RS232 无协议通信	LS_RS232_Open	FB	打开指定的 232 端口指令
	LS_RS232_Write	FB	从 232 端口进行无协议数据发送指令
	LS_RS232_Read	FB	从 232 端口读取无协议的接收数据指令
	LS_RS232_Close	FB	关闭指定的 232 端口指令

7.7.2 LS_RS232_Open

- **指令格式**

指令	名称	FB/FC	LD 表现	ST 表现	库
LS_RS232_Open	打开指定的232端口指令	FB		LS_RS232_Open_0(Baudrate:= , Databits:= , Parity:= , StopBits:= , BufferSize:= , Timeout:= , Done=> , hCom=> , Busy=> , bError=> , ErrorID=>);	PMC_Communication库

- **相关变量**

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Baudrate	波特率	DINT	遵照描述	115200	波特率:115200,38400,19200,9600,4800,2400,1200
Databits	数据位	DINT	7-8	8	数据位：8,7
Parity	奇偶校验	STRING	遵照描述	‘e’	校验:'e'或者'E','o'或者'O','n'或者'N'（e为偶检验；o为奇校验；n为无校验）

StopBits	停止位	DINT	1-2	1	1:1stop bit; 2:2stop bits;
BufferSize	缓冲区大小	UINT	遵照数据类型	1024	保持默认值1024，不需要修改
Timeout	串口打开超时时间	UDINT	遵照数据类型	0	串口打开超时时间

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
Done	返回值	BOOL	FALSE-TRUE	FALSE	完成信号；TRUE-232端口已打开
hCom	连接句柄	PMC_Communication.CAA.HANDLE	遵照数据类型	0	连接句柄，打开的232端口号
Busy	正在执行	BOOL	FALSE-TRUE	FALSE	正在执行
bError	错误	BOOL	FALSE-TRUE	FALSE	错误状态；FALSE-没有错误；
dErrorID	错误代码	UDINT	遵照数据类型	0	错误代码

	布尔	位串				整数								实数		时刻、持续时间、日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Baudrate	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-
Databits	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-
Parity	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√
StopBits	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-
BufferSize	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-
Timeout	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-
Done	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
hCom	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

Busy	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
bError	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
dErrorID	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-

- **功能说明：**

  - 打开连接句柄hCom指定的232端口；该模块Done为True以后，不要反复调用。

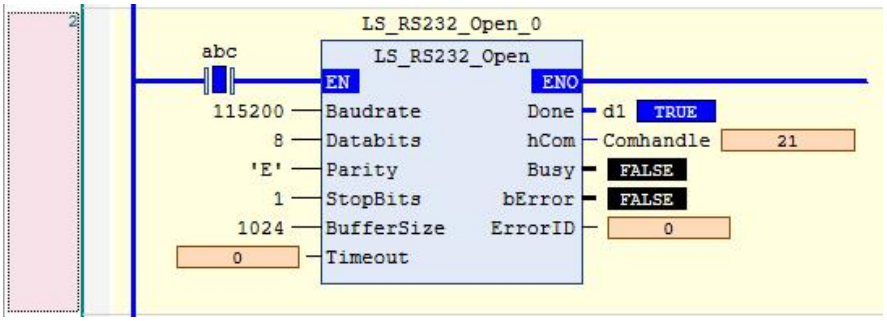
- **程序示例**

ST：将状态位 iState 的值置为 1，执行打开 232 串口指令，打开连接句柄 hCom 指定的 232 端口 20；程序如下：

```
1: //open   //将状态位iState置1，打开串口

   LS_RS232_Open_0(Baudrate:= 115200, Databits:= 8, Parity:= 'E', StopBits:= 1, BufferSize:= 1024,
      Timeout:= , Done=> , hCom=>Comhandle ,Busy=>, bError=> , ErrorID=> );
   IF LS_RS232_Open_0.Done THEN
      iState:=2;     //串口打开后，将状态位iState置2，进入写操作
   END_IF
```

LD：当 abc 变为 TRUE 时，执行打开 232 串口指令，打开连接句柄 hCom 指定的 232 端口 21；程序如下：



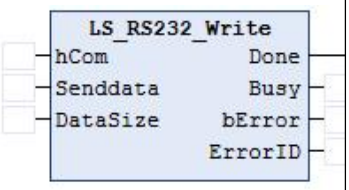
- **注意事项**

- 调用LS_RS232_Open指令配置RS232接口参数(包括通讯波特率、数据位、停止位、校验位等)并打开RS232接口。

- LS_RS232_Open指令运行后，Done变为TRUE，用于无协议通信的232端口hCom已经打开，此时可以对打开的232端口hCom进行重复的读写数据操作，不需要再反复调用LS_RS232_Open指令。

7.7.3 LS_RS232_Write

- **指令格式**

指令	名称	FB/FC	LD 表现	ST 表现	库
LS_RS232_Write	从232端口进行无协议数据发送指令	FB		LS_RS232_Write_0(hCom:= , Senddata:= , DataSize:= , Done=> , Busy=> , bError=> , ErrorID=>);	PMC_Communication库

- **相关变量**

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
hCom	连接句柄	PMC_Communication.CAA.HANDLE	遵照数据类型	0	连接句柄，打开的232端口号
Senddata	需要发送的数据指针	POINTER TO BYTE	遵照数据类型	-	需要发送的数据指针
DataSize	发送数据大小	UINT	遵照数据类型	0	发送数据大小

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
Done	返回值	BOOL	FALSE-TRUE	FALSE	完成信号；TRUE-数据发送完成
Busy	正在执行	BOOL	FALSE-TRUE	FALSE	正在执行
bError	错误	BOOL	FALSE-TRUE	FALSE	错误状态；FALSE-没有错误；
dErrorID	错误代码	UDINT	遵照数据类型	0	错误代码

	布尔	位串	整数	实数	时刻、持续时间、日期、字符串

	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
hCom	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Senddata	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
DataSize	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-
Done	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Busy	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
bError	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
dErrorID	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-

- **功能说明：**

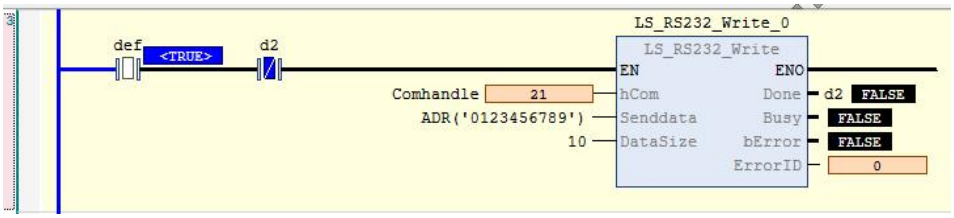
  - 从连接句柄hCom指定的232端口进行无协议数据发送。发送的数据为Senddata指定的内容，发送的数据大小在DataSize中指定。数据发送完成Done置为TRUE。

- **程序示例**

ST：232 端口打开完成后，状态位 iState 置 2，执行写数据操作；发送 10 个字节的数据（0123456789）到指定 232 端口 hCom(20)；程序如下：

```
2: //write
   LS_RS232_Write_0(hCom 20 :=Comhandle 20 , Senddata 16#B44AFD56 := ADR('0123456789'), DataSize 10 := 10,
      Done=> , Busy=> , bError=> , ErrorID=> );
   IF LS_RS232_Write_0.Done TRUE THEN
      iState 2 :=3;        //写操作完成后，将状态位iState置3，进入读操作
   END_IF
```

LD：232 端口打开完成后，将 def 置为 TRUE，执行写数据操作；发送 10 个字节的数据（0123456789）到指定 232 端口 hCom(21)；程序如下：



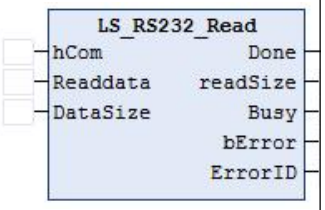
- **注意事项**

  - LS_RS232_Write指令仅在指定的232端口hCom已打开时可以执行，仅可用于无协议模式。

- 调用LS_RS232_Write指令可实现RS232端口对外发送数据。
- DataSize是发送数据的大小，要比Senddata指向的数据大，否则发送数据不完整；DataSize为0时，不发送数据。

7.7.4 LS_RS232_Read

- **指令格式**

指令	名称	FB/FC	LD 表现	ST 表现	库
LS_RS232_Read	从232端口读取无协议的接收数据指令	FB		LS_RS232_Read_0(hCom:= , Readdata:= , DataSize:= , Done=> , readSize=> , Busy=> , bError=> , ErrorID=>);	PMC_Communication库

- **相关变量**

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
hCom	连接句柄	PMC_Communication.CAA.HANDLE	遵照数据类型	0	连接句柄，打开的232端口号
Readdata	读取数据指针	POINTER TO BYTE	遵照数据类型	-	将读取到的数据保存到指针指向的地址
DataSize	最大读取数据大小	UINT	遵照数据类型	0	最大能读取数据大小，单位字节

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
Done	返回值	BOOL	FALSE-TRUE	FALSE	完成信号；TRUE-读取数据完成

readSize	实际读取数据大小	UINT	遵照数据类型	0	实际读取到的数据大小，单位字节
Busy	正在执行	BOOL	FALSE-TRUE	FALSE	正在执行
bError	错误	BOOL	FALSE-TRUE	FALSE	错误状态；FALSE-没有错误；
dErrorID	错误代码	UDINT	遵照数据类型	0	错误代码

	布尔	位串				整数								实数		时刻、持续时间、日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
hCom	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Readdata	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
DataSize	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-
Done	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
readSize	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-
Busy	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
bError	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
dErrorID	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-

- **功能说明：**

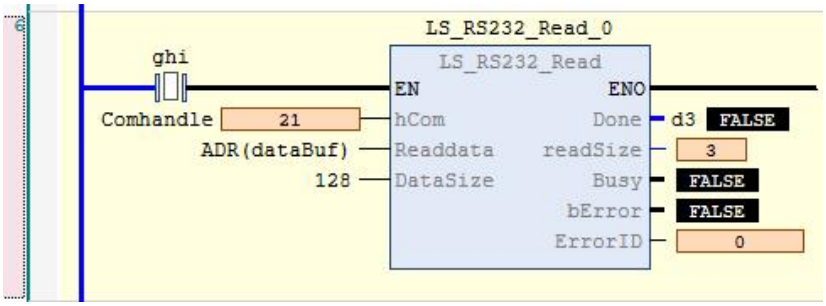
  - 从连接句柄hCom指定的232端口接收无协议的通信数据，接收的数据首先保存在232端口的接收缓存区中。本指令将按接收数据大小DataSize，把接收缓存的数据传输到读取数据Readdata指向的内存中。

- **程序示例**

ST：写数据操作完成后，状态位 iState 置 3，执行读数据操作；读取到 PC 端串口调试工具发来的 readSize（4）个字节的数据（errt）；程序如下：

```
3: //read
   LS_RS232_Read_0(hCom 20 :=Comhandle 20 , Readdata 16#B44490F8 :=ADR(dataBuf) , DataSize 128 := 128,
       Done=> , readSize 4 => readByte 4 , Busy=> , bError=> , ErrorID=> );
   IF readByte 4 >0 THEN
       Size 4 :=readByte 4 ;
       FOR i 128 :=UINT_TO_INT(Size 4 ) TO 127 DO//如果当次读取字节数小于上一次，则清掉多余的数据
           dataBuf[i 128 ] ??? :=0;
       END_FOR
       iState 3 :=2;    //返回写数据操作
   END_IF
```

LD：写数据操作完成后；将 ghi 置为 TRUE，执行读数据操作；读取到 PC 端串口调试工具发来的 readSize（3）个字节的数据（abc）；程序如下：

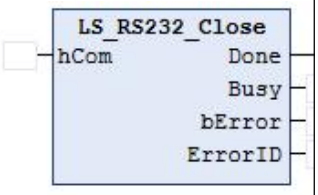


- **注意事项**

- LS_RS232_Read指令仅在指定的232端口hCom已打开时可以执行，仅可用于无协议模式。
- DataSize是最大能读取数据的大小，要比PC端串口调试工具发过来的数据大，否则Readdata读取数据不完整；DataSize为0时，则不会将接收缓存区的数据传送到Readdata指向的内存中。
- 调用LS_RS232_Read指令可实现读取RS232接收到的数据。
- Readdata读取数据指针参数必须是指向字节类型的数组，否则如果只是单个BYTE变量的话，当读取的数据不止一个BYTE的时候会造成数据丢失；而且如果本次读到的数据长度小于上一次，则保存数据的数组的前面的数据会被覆盖掉，后面的还会保存上一次的数据，这时候需要看情况清掉对应的数据了。比如上述LD例子中：
  每次执行读取数据指令，都会将之前读取到的数据（dataBuf）清空。
- 每次运行LS_RS232_Read指令，最大只能读取64个字节的数据，超过64个字节需要分多次运行LS_RS232_Read指令读取。

7.7.5 LS_RS232_Close

- **指令格式**

指令	名称	FB/FC	LD 表现	ST 表现	库
LS_RS232_Close	关闭指定的232端口指令	FB		LS_RS232_Close_0(hCom:= , Done=> , Busy=> , bError=> , ErrorID=>);	PMC_Communication库

- **相关变量**

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
hCom	连接句柄	PMC_Communication.CAA.HANDLE	遵照数据类型	0	连接句柄，要关闭的232端口号

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
Done	返回值	BOOL	FALSE-TRUE	FALSE	完成信号；TRUE-232端口已关闭
Busy	正在执行	BOOL	FALSE-TRUE	FALSE	正在执行
bError	错误	BOOL	FALSE-TRUE	FALSE	错误状态；FALSE-没有错误；
dErrorID	错误代码	UDINT	遵照数据类型	0	错误代码

	布尔	位串				整数								实数		时刻、持续时间、日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
hCom	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Done	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Busy	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

bError	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
dErrorID	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-

- **功能说明：**

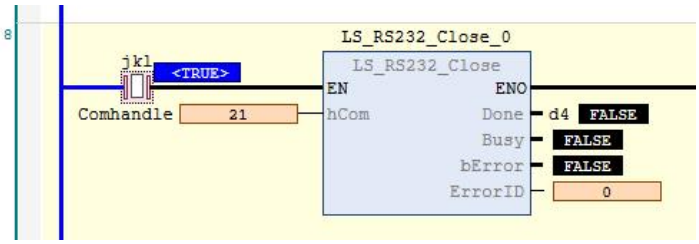
  - 关闭连接句柄hCom指定的232端口。

- **程序示例**

ST：将 iState 的值强制为 4，执行关闭 232 端口指令，关闭连接句柄 hCom 指定的 232 端口 20；程序如下：

```
4: //close
   LS_RS232_Close_0(hCom:= 20 Comhandle:= 20 ,Done=> ,Busy=> ,bError=> ,ErrorID=> );   //将状态位iState置4，关闭串口
   IF LS_RS232_Close_0.Done TRUE THEN
      iState:= 4 :=5;
   END_IF
```

LD：将 jkl 置为 TRUE，执行关闭 232 端口指令，关闭连接句柄 hCom 指定的 232 端口 21；程序如下：



- **注意事项**

- 执行LS_RS232_Close指令关闭指定的232端口后，需要重启PLC（热复位或冷复位）才能再次运行LS_RS232_Open指令打开指定232端口。

- 在使用完RS232接口后调用LS_RS232_Close指令关闭RS232接口。

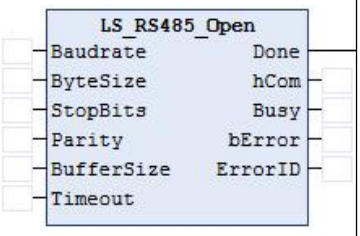
7.8RS485无协议通信

7.8.1 指令列表

指令类别	名称	FB/FC	功能
RS485 无协议通信	LS_RS485_Open	FB	打开指定的RS485 端口指令
	LS_RS485_Write	FB	从RS485 端口进行无协议数据发送指令
	LS_RS485_Read	FB	从RS485 端口进行无协议的接收数据指令
	LS_RS485_Close	FB	关闭指定的RS485 端口指令

7.8.2 LS_RS485_Open

- **指令格式**

指令	名称	FB/FC	LD 表现	ST 表现	库
LS_RS485_Open	打开指定的RS485端口指令	FB		LS_RS485_Open_0(Baudrate:= , ByteSize:= , StopBits:= , Parity:= , BufferSize:= , Timeout:= , Done=> , hCom=> , Busy=> , bError=> , ErrorID=>);	PMC_Communication库

- **相关变量**

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Baudrate	波特率	DINT	遵照描述	9600	波特率:115200,38400,19200,9600,4800,2400,1200
ByteSize	数据位	DINT	7-8	8	数据位：8,7

StopBits	停止位	DINT	1-2	1	1:1stop bit; 2:2stop bits;
Parity	奇偶校验	STRING	遵照描述	‘e’	校验:'e'或者'E','o'或者'O','n'或者'N'（e为偶检验；o为奇校验；n为无校验）
BufferSize	缓冲区大小	UINT	遵照数据类型	1024	保持默认值1024，不需要修改
Timeout	串口打开超时时间	UDINT	遵照数据类型	0	串口打开超时时间

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
Done	返回值	BOOL	FALSE-TRUE	FALSE	完成信号；TRUE-RS485端口已打开
hCom	连接句柄	PMC_Communication.CAA.HANDLE	遵照数据类型	0	连接句柄，打开的RS485端口号
Busy	正在执行	BOOL	FALSE-TRUE	FALSE	正在执行
bError	错误	BOOL	FALSE-TRUE	FALSE	错误状态；FALSE-没有错误；
dErrorID	错误代码	UDINT	遵照数据类型	0	错误代码

	布尔	位串				整数								实数		时刻、持续时间、日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Baudrate	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-
ByteSize	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-
StopBits	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-
Parity	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√
BufferSize	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-
Timeout	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-

Done	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
hCom	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Busy	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
bError	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
dErrorID	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-

- **功能说明：**

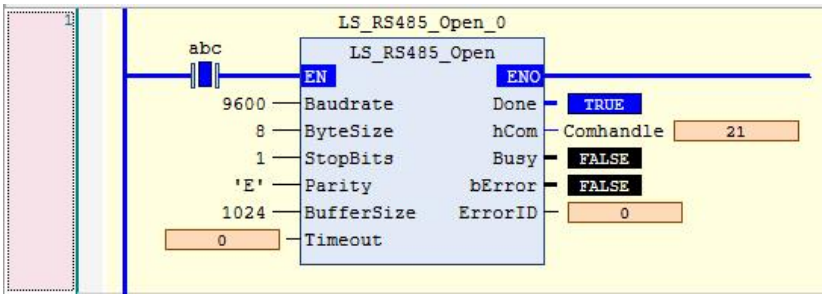
  - 打开连接句柄hCom指定的RS485端口；该模块Done为True以后，不要反复调用。

- **程序示例**

ST：将状态位 iState 的值置为 1，执行打开 RS485 端口指令，打开连接句柄 hCom 指定的 RS485 端口 20；程序如下：

```
1: //open              //将状态位iState置2，打开RS485端口
   LS_RS485_Open_0(Baudrate:= 115200, ByteSize:= 8, StopBits:= 1, Parity:= 'E', BufferSize:= ,
      Timeout:= , Done=> , hCom=>Comhandle:= 20 , bError=> , ErrorID=> );
   IF LS_RS485_Open_0.Done = TRUE THEN
      iState:=2;     //端口打开后，状态位iState置2，进入写操作
   END_IF
```

LD：当 abc 变为 TRUE 时，执行打开 RS485 端口指令，打开连接句柄 hCom 指定的 RS485 端口 21；程序如下：



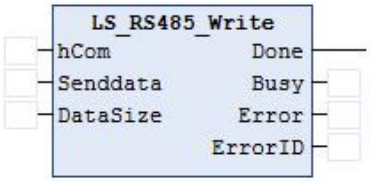
- **注意事项**

- 调用LS_RS485_Open指令配置RS485接口参数(包括通讯波特率、数据位、停止位、校验位等)并打开RS485接口。

- LS_ RS485_Open指令运行后，Done变为TRUE，用于无协议通信的RS485端口hCom已经打开，此时可以对打开的RS485端口hCom进行重复的读写数据操作，不需要再反复调用LS_RS485_Open指令。

7.8.3 LS_RS485_Write

- 指令格式

指令	名称	FB/FC	LD 表现	ST 表现	库
LS_RS485_Write	从RS485端口进行无协议数据发送指令	FB		LS_RS485_Write_0(hCom:= , Senddata:= , DataSize:= , Done=> , Busy=> , Error=> , ErrorID=>);	PMC_Communication库

- 相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
hCom	连接句柄	PMC_Communication.CAA.HANDLE	遵照数据类型	0	连接句柄，打开的485端口号
Senddata	发送的数据指针	POINTER TO BYTE	遵照数据类型	-	需要发送的数据指针
DataSize	发送数据大小	UINT	遵照数据类型	0	发送数据大小

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
Done	返回值	BOOL	FALSE-TRUE	FALSE	完成信号；TRUE-数据发送完成
Busy	正在执行	BOOL	FALSE-TRUE	FALSE	正在执行
Error	错误	BOOL	FALSE-TRUE	FALSE	错误状态；FALSE-没有错误；

ErrorID	错误代码	UDINT	遵照数据类型	0	错误代码

	布尔	位串				整数								实数		时刻、持续时间、日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
hCom	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Senddata	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
DataSize	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-
Done	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Busy	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Error	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
ErrorID	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-

- **功能说明：**

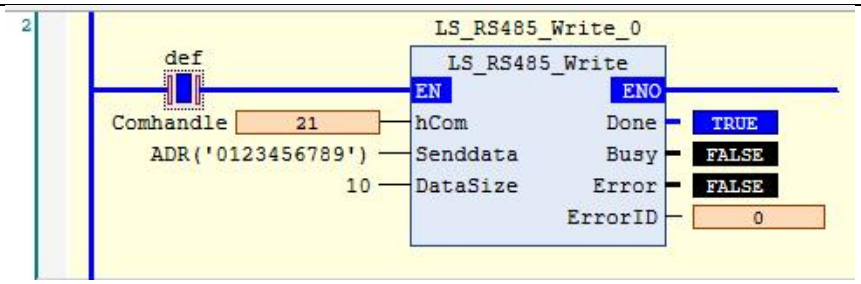
  - 从连接句柄hCom指定的RS485端口进行无协议数据发送。发送的数据为Senddata指定的内容，发送的数据大小在DataSize中指定。数据发送完成Done置为TRUE。

- **程序示例**

ST：RS485 端口打开完成后，状态位 iState 置 2，执行写数据操作；发送 10 个字节的数据（0123456789）到指定 485 端口 hCom(20)；程序如下：

```
2: //write
LS_RS485_Write_0(hCom 20 :=Comhandle 20 , Senddata 16#B445BB5E := ADR('0123456789'), DataSize 10 := 10,
    Done=> , Busy=> , Error=> , ErrorID=> );
IF LS_RS485_Write_0.Done TRUE THEN
    iState 2 :=3;        //写数据完成后，状态位iState置3，进入读操作
END_IF
```

LD：RS485 端口打开完成后，将 def 置为 TRUE，执行写数据操作；发送 10 个字节的数据（0123456789）到指定 485 端口 hCom(21)；程序如下：



- **注意事项**

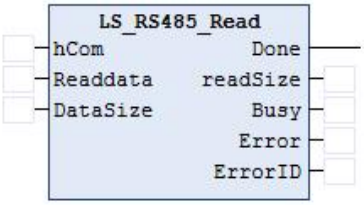
➢ LS_RS485_Write指令仅在指定的RS485端口hCom已打开时可以执行，仅可用于无协议模式。

➢ 调用LS_ RS485_Write指令可实现RS485端口对外发送数据。

➢ DataSize是发送数据的大小，要比Senddata指向的数据大，否则发送数据不完整；DataSize为0时，不发送数据。

7.8.4 LS_RS485_Read

- **指令格式**

指令	名称	FB/FC	LD 表现	ST 表现	库
LS_RS485_Read	从RS485端口进行无协议的接收数据指令	FB		LS_RS485_Read_0 (hCom:= , Readdata:= , DataSize:= , Done=> , readSize=> , Busy=> , Error=> , ErrorID=>) ;	PMC_Communication库

- **相关变量**

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
hCom	连接句柄	PMC_Communication.CAA.HANDLE	遵照数据类型	0	连接句柄，打开的RS485端口号

Readdata	读取数据指针	POINTER TO BYTE	遵照数据类型	-	将读取到的数据保存到指针指向的地址
DataSize	最大读取数据大小	UINT	遵照数据类型	0	最大能读取数据大小，单位字节

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
Done	返回值	BOOL	FALSE-TRUE	FALSE	完成信号；TRUE-读取数据完成
readSize	实际读取数据大小	UINT	遵照数据类型	0	实际读取到的数据大小，单位字节
Busy	正在执行	BOOL	FALSE-TRUE	FALSE	正在执行
Error	错误	BOOL	FALSE-TRUE	FALSE	错误状态；FALSE-没有错误；
ErrorID	错误代码	UDINT	遵照数据类型	0	错误代码

	布尔	位串				整数								实数		时刻、持续时间、日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
hCom	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Readdata	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
DataSize	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-
Done	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
readSize	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-
Busy	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Error	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
ErrorID	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-

- **功能说明：**

  - 从连接句柄hCom指定的RS485端口接收无协议的通信数据，接收的数据首先保存在RS485端口的接收缓存区中。本指令将按接收数据大小DataSize，把接收缓存的数据传输到读取数据Readdata指向的内存中。DataSize为0时，不读取数据。

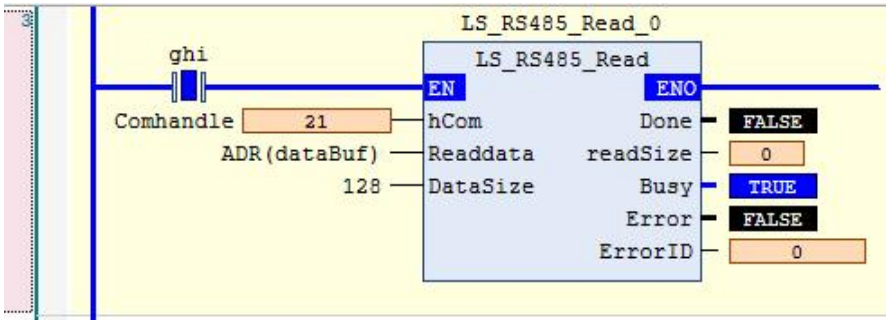
- **程序示例**

ST：写数据操作完成后，状态位 iState 置 3，执行读数据操作；读取到 PC 端串口调试工具发来的 readSize（4）个字节的数据（errt）；程序如下：

```
3: //read

    LS_RS485_Read_0(hCom 20 :=Comhandle 20 , Readdata 16#B445FC14 :=ADR(dataBuf) , DataSize 128 := 128,
        Done=> , readSize 4 => readByte 4 , Busy=> , Error=> , ErrorID=> );
    IF readByte 4 >0 THEN
    FOR i 4 :=0 TO readByte 4 -1 DO    //把读取到的数据复制给dataBuf2
        dataBuf2[buf2Num 4 ] 0 := dataBuf[i 4 ] 0 ;     //dataBuf2记录连续读取到的所有数据
            buf2Num 4 := buf2Num 4 + 1;
    END_FOR
        Size 4 :=readByte 4 ;
        iState 3 :=2;        //读数据完成后，状态位iState置2，再次进入写操作
    END_IF
```

LD：写数据操作完成后；将 ghi 置为 TRUE，执行读数据操作；读取到 PC 端串口调试工具发来的 readSize（3）个字节的数据（abc）；程序如下：



- **注意事项**

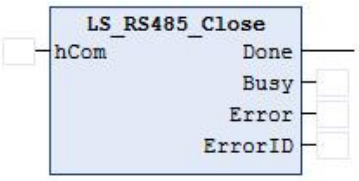
- LS_ RS485_Read指令仅在指定的RS485端口hCom已打开时可以执行，仅可用于无协议模式。
- 调用LS_ RS485_Read指令可实现读取RS485端口接收到的数据。
- DataSize是最大能读取数据的大小，要比串口调试工具发过来的数据大，否则Readdata读取数据不完整；DataSize为0时，则不会将接收缓存区的数据传送到Readdata指向的内存中。
- Readdata读取数据指针参数必须是指向字节类型的数组，否则如果只是单个BYTE变量的话，当读取的数据不止一个BYTE的时候会造成数据丢失；而且如果本次读到的数据长度小于上一次，则保存数据的数组的前面的数据会被覆盖掉，后面的还会保存上一次的数据，这时候需要看情况清掉对应的数据或者只将本次读取的数据复制给另一变量。比如上述ST例子中：

每次执行读取数据指令，都会将本次读取到的数组数据（dataBuf）复制给数组dataBuf2。

- 不要同时进行读写数据指令，否则会读取到乱码。
- 每次运行LS_RS485_Read指令，最大只能读取8个字节的数据，超过8个字节需要分多次运行LS_RS485_Read指令读取。

7.8.5 LS_RS485_Close

- **指令格式**

指令	名称	FB/FC	LD 表现	ST 表现	库
LS_RS485_Close	关闭指定的RS485端口指令	FB		LS_RS485_Close_0(hCom:= , Done=> , Busy=> , Error=> , ErrorID=>);	PMC_Communication库

- **相关变量**

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
hCom	连接句柄	PMC_Communication.CAA.HANDLE	遵照数据类型	0	连接句柄，要关闭的RS485端口号

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
Done	返回值	BOOL	FALSE-TRUE	FALSE	完成信号；TRUE- RS485端口已关闭
Busy	正在执行	BOOL	FALSE-TRUE	FALSE	正在执行
Error	错误	BOOL	FALSE-TRUE	FALSE	错误状态；FALSE-没有错误；
ErrorID	错误代码	UDINT	遵照数据类型	0	错误代码

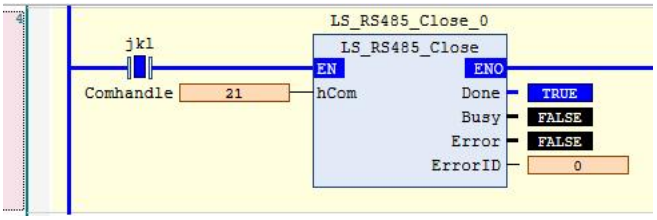
	布尔	位串				整数								实数		时刻、持续时间、日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
hCom	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Done	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Busy	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Error	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
ErrorID	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-

- **功能说明：**
  - 关闭连接句柄hCom指定的RS485端口。
- **程序示例**

ST：将 iState 的值强制为 4，执行关闭 RS485 端口指令，关闭连接句柄 hCom 指定的 RS485 端口 20；程序如下：

```
4:  //close                //将状态位iState置4，关闭RS485端口
    LS_RS485_Close_0(hCom 0 := Comhandle 20 ,Done=> ,Busy=> ,Error=> ,ErrorID=> );
    IF LS_RS485_Close_0.Done FALSE THEN
        iState 4 :=5;
    END_IF
```

LD：将 jkl 置为 TRUE，执行关闭 RS485 端口指令，关闭连接句柄 hCom 指定的 RS485 端口 21；程序如下：



- **注意事项**

- 执行LS_ RS485_Close指令关闭指定的RS485端口后，需要重启PLC（热复位或冷复位）才能再次运行LS_ RS485_Open指令打开指定RS485端口。
- 在使用完RS485接口后调用LS_RS485_Close指令关闭RS485接口。

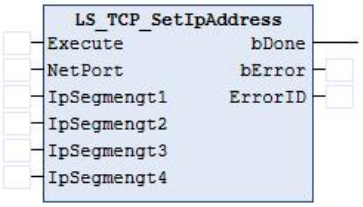
7.9 以太网通信

7.9.1 指令列表

指令类别	名称	FB/FC	功能
以太网通信	LS_TCP_SetIpAddress	FB	设置网口IP指令
	LS_TCP_GetIpAddress	FB	读取网口IP指令
	LS_TCP_SetServer	FB	将控制器设为TCP服务器指令
	LS_TCP_SetClient	FB	将控制器设为TCP客户端指令
	LS_TCP_WriteData	FB	TCP发送数据指令
	LS_TCP_ReadData	FB	TCP接收数据指令
	LS_UDP_SetConnect	FB	启动UDP/IP服务指令
	LS_UDP_WriteData	FB	UDP发送数据指令
	LS_UDP_ReadData	FB	UDP接收数据指令

7.9.2 LS_TCP_SetIpAddress

- **指令格式**

指令	名称	FB/FC	LD 表现	ST 表现	库
LS_TCP_SetIpAddress	设置网口IP指令	FB		LS_TCP_SetIpAddress_0(Execute:= , NetPort:= , IpSegmengt1:= , IpSegmengt2:= , IpSegmengt3:= , IpSegmengt4:= , bDone=> , bError=> , ErrorID=>);	PMC_Communication库

- **相关变量**

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Execute	启动	BOOL	FALSE-TRUE	FALSE	TRUE时使能功能块，上升沿

					触发；
NetPort	网口编号	BYTE	0-1	0	网口编号，从0号开始
IpSegmengt1	Ip地址第1段	BYTE	0-255	0	Ip地址第1段
IpSegmengt2	Ip地址第2段	BYTE	0-255	0	Ip地址第2段
IpSegmengt3	Ip地址第3段	BYTE	0-255	0	Ip地址第3段
IpSegmengt4	Ip地址第4段	BYTE	0-255	0	Ip地址第4段

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
Done	返回值	BOOL	FALSE-TRUE	FALSE	完成信号；IP地址设置完成
bError	错误	BOOL	FALSE-TRUE	FALSE	错误状态；FALSE-没有错误；
ErrorID	错误代码	DINT	遵照数据类型	0	错误代码

	布尔	位串				整数								实数		时刻、持续时间、日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Execute	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
NetPort	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
IpSegmengt1	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
IpSegmengt2	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
IpSegmengt3	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
IpSegmengt4	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Done	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
bError	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
ErrorID	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-

- **功能说明：**

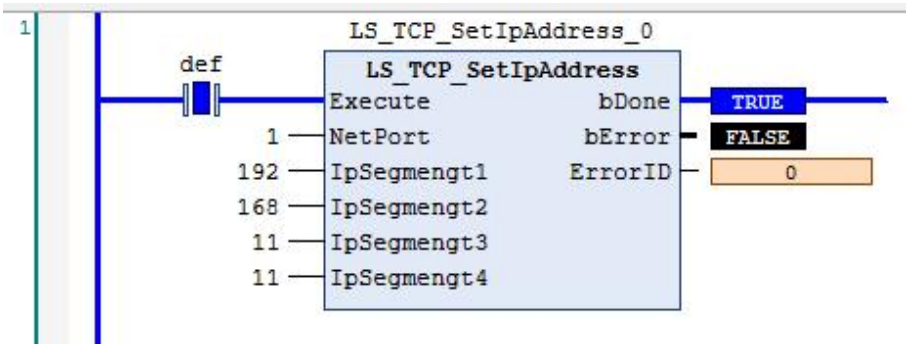
  - 配置控制器内置网口的IP地址，硬件网口编号由NetPort指定，从0开始编号，0对应eth1网口，1对应eth2和3网口，（注意： 2为EtherCAT口不可设置），IP地址由IpSegmengt参数指定，该功能块Execute上升沿生效。

- **程序示例**

ST：设置 NetPort 为 1，当 abc 变为 True 时，将预设的 IP 地址（192.168.20.10）写进 eth2 和 3 网口；程序如下：

```
1  LS_TCP_SetIpAddress_0(
2      Execute TRUE := abc TRUE,
3      NetPort 1 := 1,
4      IpSegmengt1 192 := 192,
5      IpSegmengt2 168 := 168,
6      IpSegmengt3 20 := 20,
7      IpSegmengt4 10 := 10,
8      bDone TRUE => b1 TRUE,
9      bError FALSE => be1 FALSE,
10     ErrorID 0 => er1 0 );
11 RETURN
```

LD：设置 NetPort 为 1，当 def 变为 True 时，将预设的 IP 地址（192.168.11.11）写进 eth2 和 3 网口；程序如下：



- **注意事项**

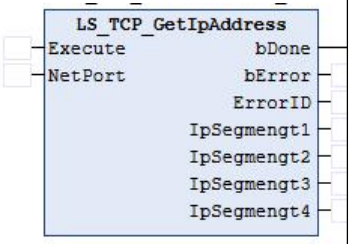
- LS_TCP_SetIpAddress设置IP指令最好在使用PLC前单独一个程序改IP，在PLC运行功能程序中不要改，而且设置完成后立即生效，该参数会保存。

- 使用LS_TCP_SetIpAddress指令更改eth1网口IP地址后，需要同步更改PC端的IP地址在同一网段，才能重新连接上PLC（假设控制器通过eth1网口与PC连接）。

- 控制器默认eth1网口IP地址为192.168.1.3，用户可直接使用该IP地址，而无需调用LS_TCP_SetIpAddress指令设置IP。
- eth2和eth3网口共用一个IP地址；eth1网口跟eth2（eth3）网口不能在同一网段（即IpSegmengt3不能设置一样）；否则控制器会报错。
- 可直接在上位机软件中，左侧目录树“Device-系统设置”中设置对应网口的IP地址。

7.9.3 LS_TCP_GetIpAddress

- **指令格式**

指令	名称	FB/FC	LD 表现	ST 表现	库
LS_TCP_GetIpAddress	读取网口IP指令	FB		LS_TCP_GetIpAddress_0(Execute:= , NetPort:= , bDone=> , bError=> , ErrorID=> , IpSegmengt1=> , IpSegmengt2=> , IpSegmengt3=> , IpSegmengt4=>);	PMC_Communication库

- **相关变量**

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Execute	启动	BOOL	FALSE-TRUE	FALSE	TRUE时使能功能块，上升沿触发；
NetPort	网口编号	BYTE	0-1	0	网口编号，从0号开始

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
Done	返回值	BOOL	FALSE-TRUE	FALSE	完成信号；获取IP地址完成
bError	错误	BOOL	FALSE-TRUE	FALSE	错误状态；FALSE-没有错误；

ErrorID	错误代码	DINT	遵照数据类型	0	错误代码
IpSegmengt1	Ip地址第1段	BYTE	0-255	0	Ip地址第1段
IpSegmengt2	Ip地址第2段	BYTE	0-255	0	Ip地址第2段
IpSegmengt3	Ip地址第3段	BYTE	0-255	0	Ip地址第3段
IpSegmengt4	Ip地址第4段	BYTE	0-255	0	Ip地址第4段

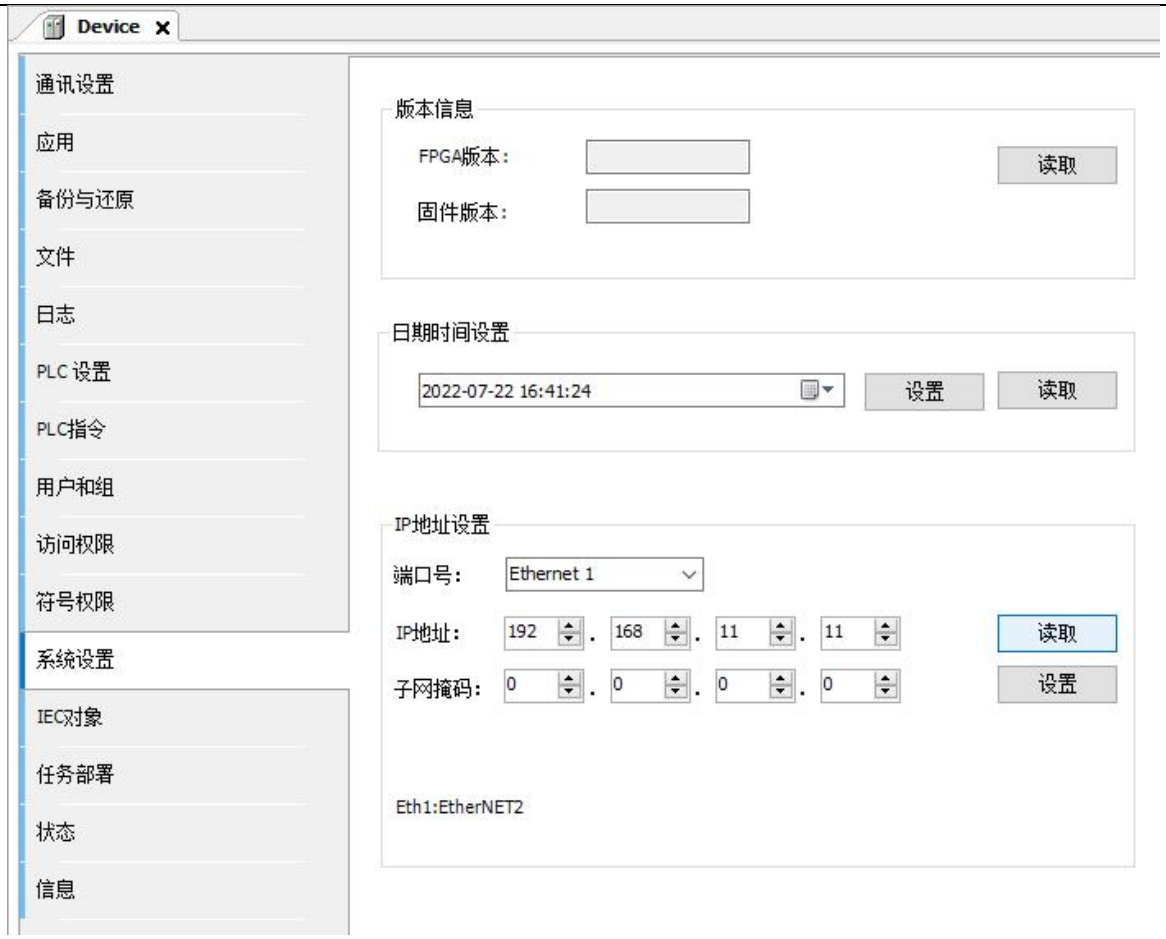
	布尔	位串				整数								实数		时刻、持续时间、日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Execute	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
NetPort	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Done	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
bError	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
ErrorID	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-
IpSegmengt1	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
IpSegmengt2	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
IpSegmengt3	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
IpSegmengt4	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

- **功能说明：**

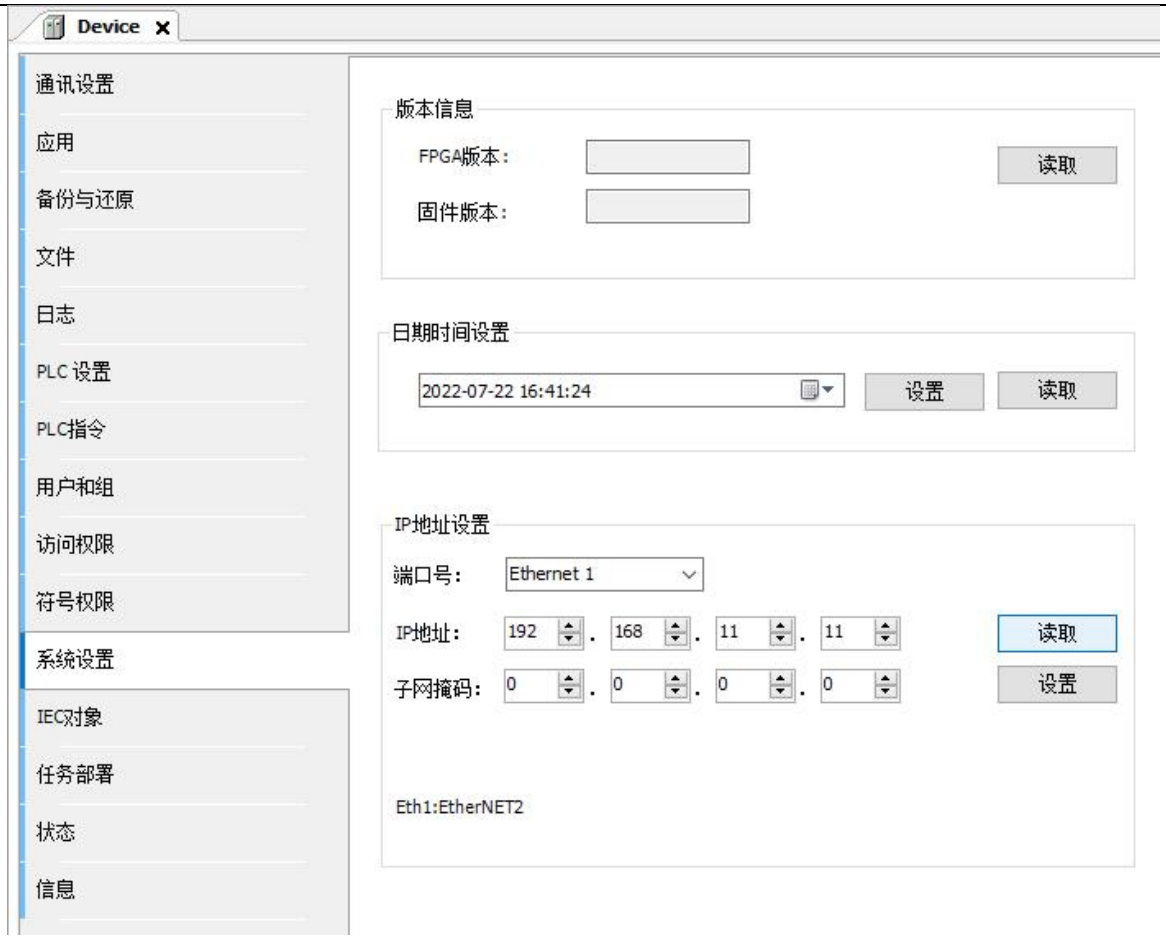
  - 获取NetPort所指定编号的网口IP地址，该功能块Execute上升沿触发。

- **程序示例**

ST：设置 NetPort 为 1，当 abc 变为 True 时，获取 eth2 和 3 网口 IP 地址；程序如下：



LD：设置 NetPort 为 1，当 def 变为 True 时，获取 eth2 和 3 网口 IP 地址；程序如下：



- **注意事项**

➢ 运行LS_TCP_GetIpAddress可以获取指定编号的网口IP地址；编号0代表eth1网口；编号1代表eth2和3网口。

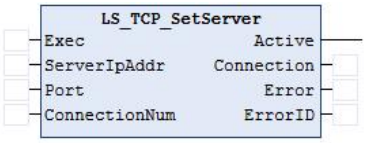
➢ eth2和eth3两个网口共用一个IP地址。

➢ 可直接在上位机软件中，左侧目录树“Device-系统设置”中读取对应网口的IP地址。

7.9.4 LS_TCP_SetServer

- **指令格式**

指令	名称	FB/FC	LD 表现	ST 表现	库

LS_TCP_SetServer	将控制器设为TCP服务器指令	FB		LS_TCP_SetServer_0(Exec:= , ServerIpAddr:= , Port:= , ConnectionNum:= , Active=> , Connection=> , Error=> , ErrorID=>);	PMC_Communication库

- **相关变量**

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Exec	启动	BOOL	FALSE-TRUE	FALSE	TRUE时使能功能块；TRUE状态需要保持；
ServerIpAddr	服务器的IP地址	STRING(32)	遵照数据类型	192.168.1.3	设置服务器的IP地址，即本控制器的IP地址；
Port	端口号	UINT	遵照数据类型	5000	设置服务器的端口号；
ConnectionNum	可连接客户端个数	USINT	遵照数据类型	1	设置可连接客户端个数，最多不能超过50个；

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
Active	服务器连接激活	ARRAY [1..50] OF BOOL	遵照数据类型	[50(FALSE)]	服务器与客户端连接激活
Connection	服务器连接句柄	ARRAY [1..50] OF DWORD	遵照数据类型	[50(1)]	服务器连接句柄
Error	错误	BOOL	FALSE-TRUE	FALSE	错误状态；FALSE-没有错误；
ErrorID	错误代	UDINT	遵照数据类型	0	错误代码

	码				

	布尔	位串				整数								实数		时刻、持续时间、日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Exec	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
ServerIpAddr	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√
Port	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-
Connection Num	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Active	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Connection	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Error	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
ErrorID	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-

- **功能说明：**

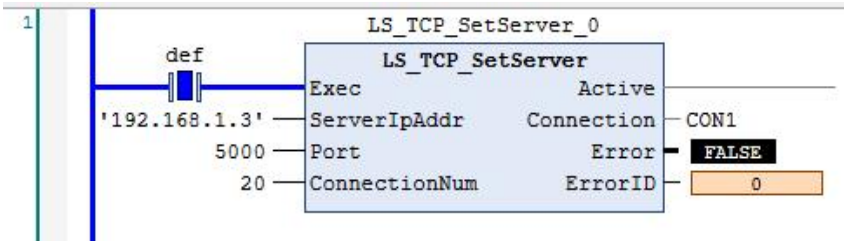
  - 将本控制器设置为TCP服务器，从对应IP地址（由ServerIpAddr指定）的网口对外通信，可与一个客户端点对点通信（与多个客户端一对多通信）；服务器端口号由Port指定。

- **程序示例**

ST：当 abc 变为 True 时，将本控制器设置为 TCP 服务器；从指定网口（192.168.1.3）对外通信，服务器端口号 5000；程序如下：

```
LS_TCP_SetServer_0(
    Exec TRUE := abc TRUE,
    ServerIpAddr '192.168.1. ' := '192.168.1.3',
    Port 5000 := 5000,
    ConnectionNum 10 := 10,
    Active=> act1,
    Connection=> con1,
    Error FALSE => err1 FALSE,
    ErrorID 0 => err2 0 );
RETURN
```

LD：当 def 变为 True 时，将本控制器设置为 TCP 服务器；从指定网口（192.168.1.3）对外通信，服务器端口号 5000；程序如下：



- **注意事项**

➢ 运行LS_TCP_SetServer指令后，可将控制器配置为TCP服务器，从ServerIpAddr指定的网口处连接客户端；由Port指定服务器的端口号。

➢ 在PC端网络调试工具处配置好TCP客户端模式，设置好服务器IP地址（ServerIpAddr）与端口号（Port），再点击连接上PLC服务器。

➢ 由ConnectionNum指定可连接客户端个数，最多不能超过50个客户端。

➢ 当Exec变为True，设置控制器工作在服务器模式并初始化连接；当客户端连接控制器后，对应的Active[]变为True，表示服务器与客户端连接成功；此时可以根据对应的服务器连接句柄Connection[]编程收发数据指令。

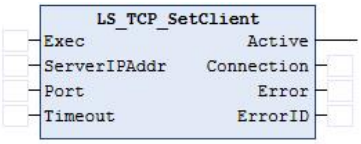
➢ Exec需要保持True状态；当Exec变为FALSE时，服务器关闭，通信停止；此时若想重启TCP通信，需要重启PLC（热复位或冷复位），再运行LS_TCP_SetServer指令配置控制器成服务器模式。

➢ 当ConnectionNum指定的客户端数大于1时，按照客户端与服务器连接顺序排序；第一个连接上服务器器的，激活结果为Active[1]，连接句柄为Connection[1]；第二个连接上服务器的，激活结果为Active[2]，连接句柄为Connection[2]；以此类推。

7.9.5 LS_TCP_SetClient

- **指令格式**

指令	名称	FB/	LD 表现	ST 表现	库

		FC			
LS_TCP_SetClient	将控制器设为TCP客户端指令	FB		LS_TCP_SetClient_0(Exec:= , ServerIPAddr:= , Port:= , Timeout:= , Active=> , Connection=> , Error=> , ErrorID=>);	PMC_Communication库

- **相关变量**

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Exec	启动	BOOL	FALSE-TRUE	FALSE	TRUE时使能功能块；TRUE状态需要保持；
ServerIpAddr	服务器的IP地址	STRING(32)	遵照数据类型	192.168.1.50	设置需要连接的服务器IP地址；即PC端的IP地址；
Port	端口号	UINT	遵照数据类型	5000	设置需要连接的服务器端口号；即PC端设置的端口号
Timeout	超时时间	UDINT	遵照数据类型	8000	连接错误超时时间，单位us

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
Active	连接激活	BOOL	FALSE-TRUE	FALSE	与服务器连接激活
Connection	连接句柄	DWORD	遵照数据类型	0	连接句柄
Error	错误	BOOL	FALSE-TRUE	FALSE	错误状态；FALSE-没有错误；
ErrorID	错误代码	UDINT	遵照数据类型	0	错误代码

	布尔	位串				整数								实数		时刻、持续时间、日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Exec	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
ServerIpAddr	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√
Port	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-
Timeout	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-
Active	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Connection	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Error	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
ErrorID	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-

- **功能说明：**

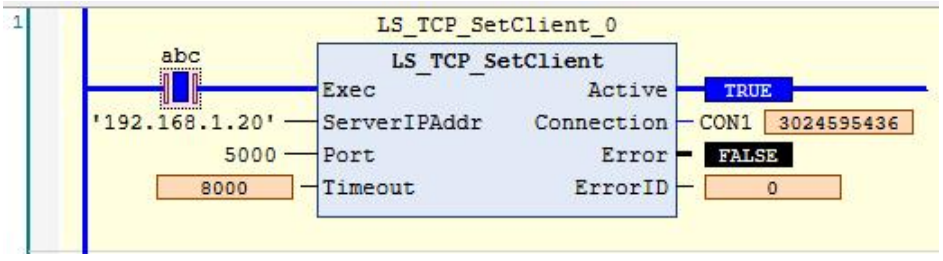
  - 将本控制器设置为TCP通信的客户端，并连接开启TCP通信服务的服务器（服务器IP地址为ServerIpAddr，端口号为Port）。

- **程序示例**

ST：当 abc 变为 True 时，将本控制器设置为 TCP 客户端；连接 IP 地址为（192.168.1.20），端口号为（5000）的服务器（服务器已开启 TCP 通信服务）；程序如下：

```
LS_TCP_SetClient_0(Exec TRUE := abc TRUE, ServerIPAddr '192.168.1. ':= '192.168.1.20',
                   Port 5000 := 5000, Timeout:= , Active TRUE => act1 TRUE,
                   Connection 3024590800 => CON 3024590800,
                   Error FALSE => er1 FALSE, ErrorID 0 => er2 0);
```

LD：当 abc 变为 True 时，将本控制器设置为 TCP 客户端；连接 IP 地址为（192.168.1.20），端口号为（5000）的服务器（服务器已开启 TCP 通信服务）；程序如下：

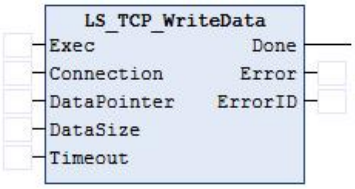


- **注意事项**

- 需要先在PC端的网络调试助手处将PC配置为TCP服务器，设置服务器IP地址（与ServerIpAddr一致），端口号（与Port一致）并打开服务器；再运行LS_TCP_ SetClient指令配置控制器为TCP客户端模式，连接服务器。
- 在PC端设定服务器并启动后，当Exec变为True，配置控制器工作在客户端模式并初始化连接；可以通过监控Active是否为True，来判断客户端与服务器是否连接成功；连接成功后可以根据生成的连接句柄Connection编程收发数据指令。
- 在设定控制器Client模式，PC为Server模式时，需要将PC的Windows防火墙关掉才能建立连接进行通讯。
- 由ServerIpAddr指定服务器的IP地址，需与控制器在同一网段。
- Exec需要保持True状态；当Exec变为FALSE时，连接关闭，通信停止；此时若想重启TCP通信，再次运行LS_TCP_SetClient指令即可。
- 服务器不小心断开连接，需要重启服务器后再次运行LS_TCP_SetClient指令即可重新建立通信。

7.9.6 LS_TCP_WriteData

- **指令格式**

指令	名称	FB/FC	LD 表现	ST 表现	库
LS_TCP_WriteData	TCP发送数据指令	FB		LS_TCP_WriteData_0(Exec:= , Connection:= , DataPointer:= , DataSize:= , Timeout:= , Done=> , Error=> , ErrorID=>) ;	PMC_Communication库

- **相关变量**

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Exec	启动	BOOL	FALSE-TRUE	FALSE	TRUE时使能功能块；上升沿触发；
Connection	连接句柄	DWORD	遵照数据类型	0	连接句柄
DataPointer	发送的数据指针	POINTER TO BYTE	遵照数据类型	-	设置需要发送的数据指针
DataSize	发送数据大小	UDINT	遵照数据类型	0	发送数据大小
Timeout	超时时间	UDINT	遵照数据类型	8000	发送超时时间，单位us

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
Done	返回值	BOOL	FALSE-TRUE	FALSE	完成信号；True-发送数据完成
Error	错误	BOOL	FALSE-TRUE	FALSE	错误状态；FALSE-没有错误；
ErrorID	错误代码	UDINT	遵照数据类型	0	错误代码

	布尔	位串				整数								实数		时刻、持续时间、日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Exec	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Connection	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
DataPointer	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
DataSize	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-
Timeout	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-
Done	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

Error	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
ErrorID	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-

- **功能说明：**

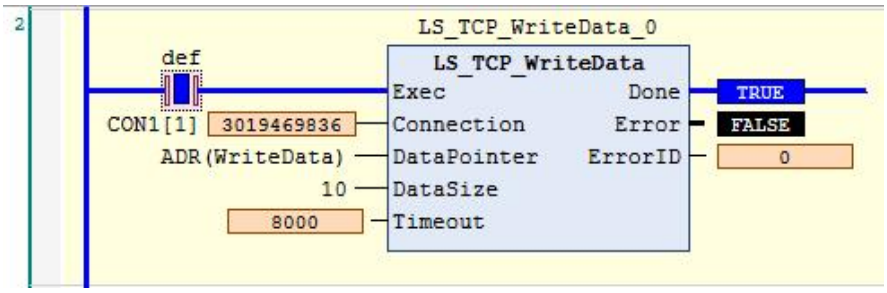
  - 由Connection指定的连接句柄将DataPointer指定的数据发送出去，发送数据的大小由DataSize指定，DataSzie的值为0时，不发送数据；

- **程序示例**

ST：当 writeda 变为 True 时，通过 Connection 指定的连接句柄（ConnectHandle[1]），发送 DataSize（10）个字节的数据（0123456789）到 PC 端；程序如下：

```
LS_TCP_WriteData_0(Exec:= writeda, Connection:= ConnectHandle[1],
                   DataPointer:= ADR('0123456789'), DataSize:= 10,
                   Timeout:= , Done=> dl, Error=> err5, ErrorID=> err6);
```

LD：当 def 变为 True 时，通过 Connection 指定的连接句柄（CON1[1]），发送 DataSize（10）个字节的数据（0123456789）到 PC 端；程序如下：



- **注意事项**

- 需要先将控制器配置为服务器或客户端模式，并成功建立TCP/IP通信后（Active变为True），才能运行发送数据LS_TCP_WriteData指令，向外发送数据(参数连接句柄Connection由设置控制器工作模式指令生成)。

- Exec上升沿触发；当Exec变为True，通过连接句柄Connection发送数据；数据发送完成后，Done变为True。

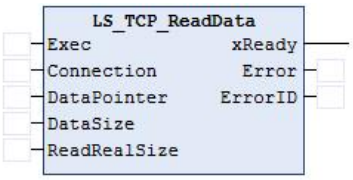
- DataPointer是待发送数据的地址。

- DataSize是发送数据的大小，要比DataPointer指向的数据大，否则发送数据不完整；

DataSize为0时，不发送数据。

7.9.7 LS_TCP_ReadData

- 指令格式

指令	名称	FB/FC	LD 表现	ST 表现	库
LS_TCP_ReadData	TCP接收数据指令	FB		LS_TCP_ReadData_0(Exec:= , Connection:= , DataPointer:= , DataSize:= , ReadRealSize:= , xReady=> , Error=> , ErrorID=>);	PMC_Communication库

- 相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Exec	启动	BOOL	FALSE-TRUE	FALSE	TRUE时使能功能块；TRUE状态需要保持；
Connection	连接句柄	DWORD	遵照数据类型	0	连接句柄
DataPointer	读取数据的指针	POINTER TO BYTE	遵照数据类型		将读取到的数据保存到指针指定的地址
DataSize	最大读取数据大小	UDINT	遵照数据类型	0	最大读取数据大小，单位字节
ReadRealSize	实际读取数据大小	POINTER TO UDINT	遵照数据类型		实际读取的数据大小

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
xReady	返回值	BOOL	FALSE-TRUE	FALSE	True-表示接收到数据；FALSE-未接收到数据；
Error	错误	BOOL	FALSE-TRUE	FALSE	错误状态；FALSE-没有

					错误；
ErrorID	错误代码	UDINT	遵照数据类型	0	错误代码

	布尔	位串				整数								实数		时刻、持续时间、日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Exec	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Connection	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
DataPointer	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
DataSize	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-
ReadRealSize	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
xReady	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Error	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
ErrorID	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-

- **功能说明：**

  - 将Connection指定的通信连接句柄中接收的数据保存在DataPointer中，保存数据的大小在DataSize中指定，保存数据后，将实际保存的数据大小代入ReadRealSize中。指令正常结束时（xReady的值变为TRUE），完成向DataPointer保存数据。DataSzie的值为0时，不将通信连接句柄中接收的数据保存到DataPointer中。

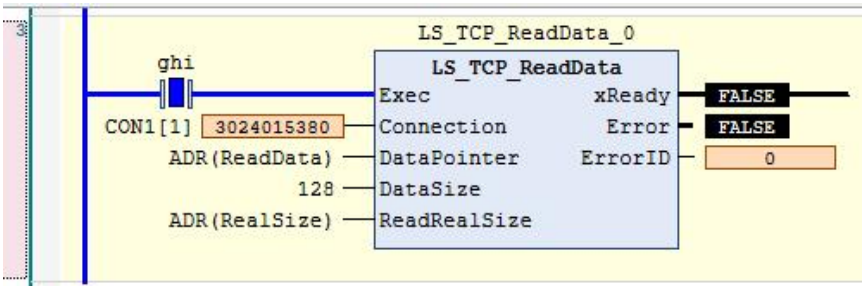
- **程序示例**

ST：当 reata 变为 True 时，读取到网络调试工具发来的 ReadRealSize（6）个字节的数据（abc123）；程序如下：

```
12  LS_TCP_ReadData_0(Exec TRUE := reata TRUE, Connection 3024778336 := ConnectHandle[1] 3024778336,
13                    DataPointer 3024790504 := ADR(redata), DataSize 128 := 128,
14                    ReadRealSize 16#B44AA46C := ADR(ReadSize_Real 0 ), xReady FALSE => xrl FALSE,
15                    Error FALSE => err3 FALSE, ErrorID 0 => err4 0 );
```

LD：当 ghi 变为 True 时，读取到网络调试工具发来的 ReadRealSize（6）个字节的数据

（abc123）；程序如下：



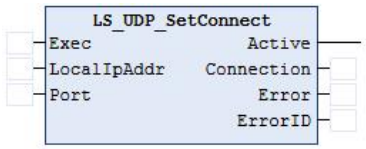
- **注意事项**

- 需要先将控制器配置为服务器或客户端模式，并成功建立TCP/IP通信后（Active变为True），才能运行读取数据LS_TCP_ReadData指令，读取通信连接句柄中收到的数据(参数连接句柄由设置控制器工作模式指令生成)。
- Exec需要保持True状态，当接收到网络调试助手发来的数据时，xReady变为True；下一个扫描周期扫描到该指令，xReady变为FALSE。
- DataSize是最大能读取数据的大小，要比网络调试工具发过来的数据大，否则读取数据不完整；DataSize为0时，则不会将通信连接句柄中接收的数据保存到DataPointer中。
- DataPointer读取数据指针参数必须是指向字节类型的数组，否则如果只是单个BYTE变量的话，当读取的数据不止一个BYTE的时候会造成数据丢失；而且如果本次读到的数据长度小于上一次，则保存数据的数组的前面的数据会被覆盖掉，后面的还会保存上一次的数据，这时候需要看情况清掉对应的数据了。
- ReadRealSize是实际读取到的数据大小，单位字节。

7.9.8 LS_UDP_SetConnect

- **指令格式**

指令	名称	FB/FC	LD 表现	ST 表现	库

LS_UDP_SetConnect	启动UDP/IP服务指令	FB		LS_UDP_SetConnect_0(Exec:= , LocalIpAddr:= , Port:= , Active=> , Connection=> , Error=> , ErrorID=>);	PMC_Communication库

- **相关变量**

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Exec	启动	BOOL	FALSE-TRUE	FALSE	TRUE时使能功能块；TRUE状态需要保持；
LocalIpAddr	本机IP地址	STRING(32)	遵照数据类型	192.168.1.3	本机IP地址
Port	本机端口号	UINT	遵照数据类型	5000	本机端口号

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
Active	UDP连接激活	BOOL	FALSE-TRUE	FALSE	UDP连接激活
Connection	连接句柄	DWORD	遵照数据类型	0	连接句柄
Error	错误	BOOL	FALSE-TRUE	FALSE	错误状态；FALSE-没有错误；
ErrorID	错误代码	UDINT	遵照数据类型	0	错误代码

	布尔	位串				整数								实数		时刻、持续时间、日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING

Exec	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
LocalIpAddr	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√
Port	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-
Active	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Connection	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Error	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
ErrorID	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-

- **功能说明：**

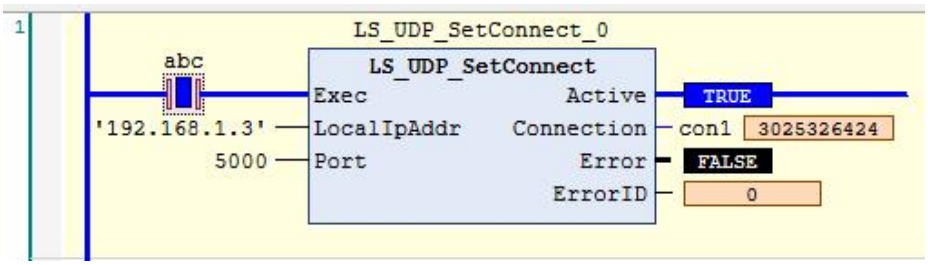
  - 将LocalIpAddr中指定的IP地址和Port中指定的端口号启用UDP/IP服务；Active的值为TRUE后，UDP连接激活；生成的UDP连接句柄保存在Connection中，用作发送和接收数据。

- **程序示例**

ST：当 abc 变为 True 时，将本控制器 IP 地址（192.168.1.3），端口号（5000）启用 UDP/IP 服务；程序如下：

```
LS_UDP_SetConnect_0(
    Exec TRUE := abc TRUE, LocalIpAddr '192.168.1.' := '192.168.1.3', Port 5000 := 5000,
    Active TRUE => ACT1 TRUE, Connection 3025335520 => con1 3025335520, Error FALSE => ER1 FALSE,
    ErrorID 0 => ER11 0 );
```

LD：当 abc 变为 True 时，将本控制器 IP 地址（192.168.1.3），端口号（5000）启用 UDP/IP 服务；程序如下：



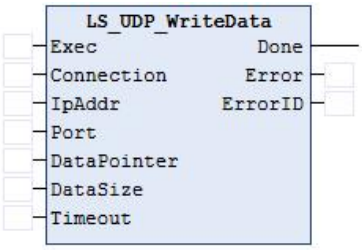
- **注意事项**

  - 运行LS_UDP_SetConnect后，Active立即变为True；表示对应IP地址，端口已开启UDP/IP服务；此时需要在PC端的网络调试助手处也开启UDP/IP服务，才能正常收发数据。

- 控制器IP地址与PC端IP地址需在同一网段，端口号可以不同。
- Exec需要保持True状态；当Exec变为FALSE时，关闭UDP/IP服务；此时若想重启UDP/IP通信，再次运行LS_UDP_SetConnect指令即可。
- 在控制器和PC进行UDP通讯时，需要将PC的Windows防火墙关掉才能正常通讯。
- 开启UDP/IP服务后，可通过生成的UDP连接句柄Connection进行收发数据指令的编程。

7.9.9 LS_UDP_WriteData

- **指令格式**

指令	名称	FB/FC	LD 表现	ST 表现	库
LS_UDP_WriteData	UDP发送数据指令	FB		LS_UDP_WriteData_0(Exec:= , Connection:= , IpAddr:= , Port:= , DataPointer:= , DataSize:= , Timeout:= , Done=> , Error=> , ErrorID=>) ;	PMC_Communication库

- **相关变量**

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Exec	启动	BOOL	FALSE-TRUE	FALSE	TRUE时使能功能块；上升沿触发；
Connection	连接句柄	DWORD	遵照数据类型	0	当前发送数据连接句柄
IpAddr	IP地址	STRING(32)	遵照数据类型	192.168.1.5	设置接收数据端的IP地址
Port	端口号	UINT	遵照数据类型	5000	设置接收数据端IP地址对

					应的端口号
DataPointer	发送的数据指针	POINTER TO BYTE	遵照数据类型	-	需要发送的数据指针
DataSize	发送数据大小	UDINT	遵照数据类型	0	发送数据大小
Timeout	超时时间	UDINT	遵照数据类型	10000	发送超时时间，单位us

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
Done	返回值	BOOL	FALSE-TRUE	FALSE	完成信号；True-发送数据完成
Error	错误	BOOL	FALSE-TRUE	FALSE	错误状态；FALSE-没有错误；
ErrorID	错误代码	UDINT	遵照数据类型	0	错误代码

	布尔	位串				整数								实数		时刻、持续时间、日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Exec	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Connection	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
IpAddr	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√
Port	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-
DataPointer	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
DataSize	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-
Timeout	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-
Done	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Error	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
ErrorID	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-

- **功能说明：**

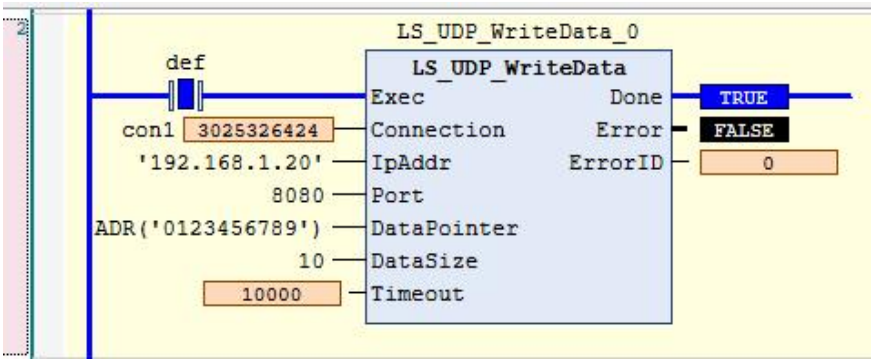
- 由启动UDP/IP服务生成的UDP连接句柄Connection将DataPointer指定的数据发送出去，发送的数据大小在DataSize中指定，指令正常结束时（Done的值为TRUE）；DataSzie的值为0时，不发送数据。

- **程序示例**

ST：当 def 变为 True 时，通过 UDP 连接句柄 Connection 发送数据（0123456789）到目标 IP（192.168.1.20），端口号（5000）的 UDP 接收端；程序如下：

```
LS_UDP_WriteData_0(
    Exec TRUE := def TRUE, Connection 3025335520 := con1 3025335520, IpAddr '192.168.1. ▸ := '192.168.1.20',
    Port 5000 := 5000, DataPointer 3024750774 := ADR('0123456789'),
    DataSize 10 := 10, Timeout:= , Done TRUE => d2 TRUE,
    Error FALSE => er2 FALSE, ErrorID 0 => er22 0 );
```

LD：当 def 变为 True 时，通过 UDP 连接句柄 Connection 发送数据（0123456789）到目标 IP（192.168.1.20），端口号（8080）的 UDP 接收端；程序如下：



- **注意事项**

- 需要在PC端网络调试助手处先开启UDP/IP服务，运行LS_UDP_WriteData指令后，在网络助手处可以读取到控制器发出的数据。

- Exec上升沿触发；当Exec变为True，通过UDP连接句柄Connection发送数据；数据发送完成后，Done变为True。

- IpAddr指定接收数据端（即PC端网络调试助手）的IP地址；Port指定接收数据端（即PC端网络调试助手）的端口号。

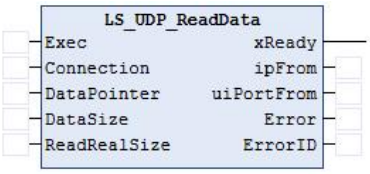
- DataPointer是发送数据的地址。

- DataSize是发送数据的大小，要比DataPointer指向的数据大，否则发送数据不完整；

DataSize为0时，不发送数据。

7.9.10 LS_UDP_ReadData

- **指令格式**

指令	名称	FB/FC	LD 表现	ST 表现	库
LS_UDP_ReadData	UDP接收数据指令	FB		LS_UDP_ReadData_0(Exec:= , Connection:= , DataPointer:= , DataSize:= , ReadRealSize:= , xReady=> , ipFrom=> , uiPortFrom=> , Error=> , ErrorID=>);	PMC_Communication库

- **相关变量**

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Exec	启动	BOOL	FALSE-TRUE	FALSE	TRUE时使能功能块；TRUE状态需要保持；
Connection	接收端连接句柄	DWORD	遵照数据类型	0	接收端连接句柄
DataPointer	读取数据的指针	POINTER TO BYTE	遵照数据类型		将读取到的数据保存到指针指定的地址
DataSize	最大读取数据大小	UDINT	遵照数据类型	0	最大读取数据大小，单位字节
ReadRealSize	实际读取	POINTER TO	遵照数据类型		实际读取的数据大小

	数据大小	UDINT			

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
xReady	返回值	BOOL	FALSE-TRUE	FALSE	True-表示接收到数据；FALSE-未接收到数据；
ipFrom	IP地址	STRING(32)	遵照数据类型		发送方的IP地址
uiPortFrom	端口号	UINT	遵照数据类型		发送方的端口号
Error	错误	BOOL	FALSE-TRUE	FALSE	错误状态；FALSE-没有错误；
ErrorID	错误代码	UDINT	遵照数据类型	0	错误代码

	布尔	位串				整数								实数		时刻、持续时间、日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Exec	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Connection	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
DataPointer	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
DataSize	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-
ReadRealSize	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
xReady	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
ipFrom	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√
uiPortFrom	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-
Error	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
ErrorID	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-

- **功能说明：**

  - 将UDP连接句柄Connection中接收的数据保存在DataPointer中，保存数据的大小在

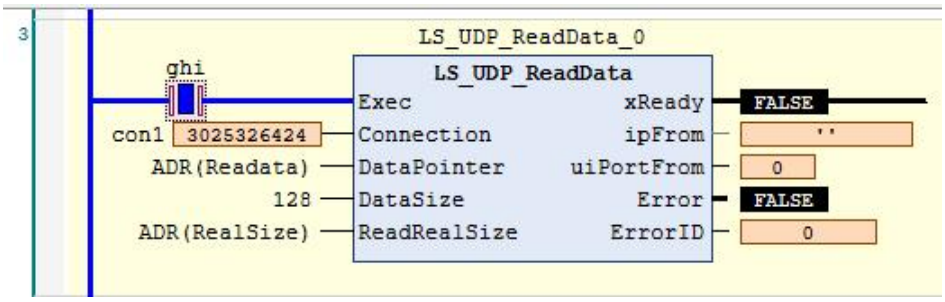
DataSize中指定，保存数据后，将实际保存的数据大小代入ReadRealSize中。指令正常结束时（xReady的值变为TRUE），完成向DataPointer保存数据。DataSzie的值为0时，不读取接收数据。

- **程序示例**

ST：当 ghi 变为 True 时，将 Connection 中接收的数据（errt）保存在 DataPointer 指定的地址（Readata）中；程序如下：

```
LS_UDP_ReadData_0(
    Exec TRUE := ghi TRUE, Connection 3025335520 := con1 3025335520, DataPointer 3025323224 := ADR(Readata),
    DataSize 128 := 128, ReadRealSize 16#B452BE34 := ADR(RealSize 0 ), xReady FALSE => xr3 FALSE,
    ipFrom '' => ip3 '', uiPortFrom 0 => pot3 0 , Error FALSE => er3 FALSE,
    ErrorID 0 => er33 0 );
```

LD：当 ghi 变为 True 时，将 Connection 中接收的数据（abc123）保存在 DataPointer 指定的地址（Readata）中；程序如下：



- **注意事项**

- 需要在PC端网络调试助手处先开启UDP/IP服务，运行LS_UDP_ReadData指令后，可以将UDP连接句柄Connection中接收到的数据保存在DataPointer中。
- Exec需要保持True状态，当接收到网络调试助手发来的数据时，xReady变为True；下一个扫描周期扫描到该指令，xReady变为FALSE。
- DataSize是最大能读取数据的大小，要比网络调试工具发过来的数据大，否则读取数据不完整；DataSize为0时，则不会将UDP连接句柄中接收的数据保存到DataPointer中。
- DataPointer读取数据指针参数必须是指向字节类型的数组，否则如果只是单个BYTE变量的话，当读取的数据不止一个BYTE的时候会造成数据丢失；而且如果本次读到的数据长度小于上一次，则保存数据的数组的前面的数据会被覆盖掉，后面的还会保存上一次的

数据，这时候需要看情况清掉对应的数据了。

- ReadRealSize是实际读取到的数据大小，单位字节。

7.10 雷赛插补指令

7.10.1 指令列表

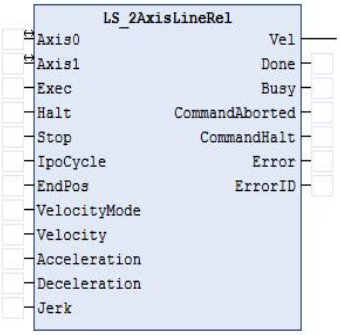
指令类别	名称	FB/FC	功能
雷赛插补指令	LS_2AxisLineAbs	FB	两轴绝对直线插补
	LS_2AxisLineRel	FB	两轴相对直线插补
	LS_3AxisLineAbs	FB	三轴绝对直线插补
	LS_3AxisLineRel	FB	三轴相对直线插补
	LS_4AxisLineAbs	FB	四轴绝对直线插补
	LS_4AxisLineRel	FB	四轴相对直线插补
	LS_5AxisLineAbs	FB	五轴绝对直线插补
	LS_5AxisLineRel	FB	五轴相对直线插补
	LS_6AxisLineAbs	FB	六轴绝对直线插补
	LS_6AxisLineRel	FB	六轴相对直线插补
	LS_2AxisLineAbs_Ratio	FB	两轴可调速绝对直线插补
	LS_2AxisLineRel_Ratio	FB	两轴可调速相对直线插补
	LS_LineFollowAbs	FB	绝对跟随
	LS_LineFollowRel	FB	相对跟随
	LS_2AxisCircleAbs	FB	两轴绝对圆弧插补
	LS_2AxisCircleRel	FB	两轴相对圆弧插补
	LS_3AxisCircleAbs	FB	三轴绝对圆弧插补
	LS_3AxisCircleRel	FB	三轴相对圆弧插补
	LS_2AxisEllipsesAbs	FB	两轴绝对椭圆插补
	LS_2AxisEllipsesRel	FB	两轴相对椭圆插补
	LS_2AxisCircleAbs_Helical	FB	绝对螺旋插补
	LS_2AxisCircleRel_Helical	FB	相对螺旋插补
	LS_3AxisGCode	FB	三轴 G 代码插补
	LS_3AxisGCode_File	FB	三轴 G 代码插补-文件

	LS_4AxisGCode	FB	四轴 G 代码插补
	LS_4AxisGCode_File	FB	四轴 G 代码插补-文件
	LS_4AxisGCodeAxisP	FB	四轴 G 代码插补-P
	LS_4AxisGCodeAxisP_File	FB	四轴 G 代码插补-文件-P
	LS_6AxisGCodeAxisUVW_file	FB	六轴 G 代码插补-文件
	LS_6Axis_ZeroOffset	FB	6 轴零点偏移
	LS_3AxisMoveSequence	FB	三轴连续插补运动
	LS_4AxisMoveSequence	FB	四轴连续插补运动
	LS_6AxisMoveSequence	FB	六轴连续插补运动

7.10.2 LS_2AxisLineAbs

指定的两个轴做直线插补运动，位置坐标为绝对坐标。

- **指令格式**

指令	名称	FB/FC	LD表现	ST表现	库
LS_2AxisLineRel	两轴相对直线插补	FB		LS_2AxisLineRel(Axis0:=, Axis1:=, Exec:=, Halt:=, Stop:=, IpoCycle:=, EndPos:=, VelocityMode:=, Velocity:=, Acceleration:=, Deceleration:=, Jerk:=, Vel=>, Done=>, Busy=>, CommandAborted=> , CommandHalt=>, Error=>, ErrorID=>);	PMC_IpoLib

- **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	初始值	描述
Axis0	轴0	AXIS_REF_VIRTUAL_SM3	-	-	插补0轴
Axis1	轴1	AXIS_REF_VIRTUAL_SM3	-	-	插补1轴

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Exec	启动	BOOL	TRUE-FALSE	FALSE	启动命令，上升沿有效。
Halt	暂停	BOOL	TRUE-FALSE	FALSE	暂停命令，当该输入量为 True 时，当前插补运动暂停；当状态再次切换到 False 时，插补模块继续之前的未完成的插补任务。
Stop	停止	BOOL	TRUE-FALSE	FALSE	停止命令，该命令为 True 时，插补模块停止。
IpoCycle	插补周期	DWORD	-	2000	插补周期，单位：us，必须和所在任务周期一致；
Endpos	目标位置	ARRAY[0..1]OFLREAL	负数，0 正数	0	轴的目标位置，单位：Pulse
VelocityMode	速度模式	SMC_INT_VELMODE	0-3	SIGMOID	速度模式：梯形-0（TRAPEZOID）；S 形-1（SIGMOID）；四次方-3（QUADRATIC）
Velocity	速度	LREAL	0，正数	0	插补合成速度，Pulse/s

Acceleration	加速度	LREAL	0，正数	0	插补合成加速度，Pulse/s2
Deceleration	减速度	LREAL	0，正数	0	插补合成减速度，Pulse/s2
Jerk	加加速度	LREAL	负数，0 正数	90000000	加加速度，速度模式为 3 时需要设置此参数，该参数值不能为 0。

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
Vel	合速度	LREAL	0，正数	0	当前插补运动合成速度。
Done	完成	BOOL	TRUE-FALSE	FALSE	True 表示插补模块已完成插补任务。
Busy	运动中	BOOL	TRUE-FALSE	FALSE	True 表示插补运动正在进行之中。
Command Aborted	运动中断	BOOL	TRUE-FALSE	FALSE	当 Stop 有效时，该变量为 True，表示插补模块非正常中止。
Command Halt	运动暂停	BOOL	TRUE-FALSE	FALSE	当 Halt 为 True 时，该变量为 True，表示插补模块处于暂停状态。
Error	错误	BOOL	TRUE-FALSE	FALSE	错误状态；False-没有错误。
ErrorID	错误代码	DINT	0，正数	0	错误代码

	布尔	位串				整数								实数		时刻、持续时间 日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Axis0	AXIS_REF_VIRTUAL_SM3																			
Axis1	AXIS_REF_VIRTUAL_SM3																			
Exec	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Halt	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Stop	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
IpoCycle	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

Endpos	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
VelocityMode	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Velocity	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–
Acceleration	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–
Deceleration	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–
Jerk	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–
Vel	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–
Done	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Busy	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
CommandAborted	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
CommandHalt	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Error	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
ErrorID	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–

- **功能说明：**

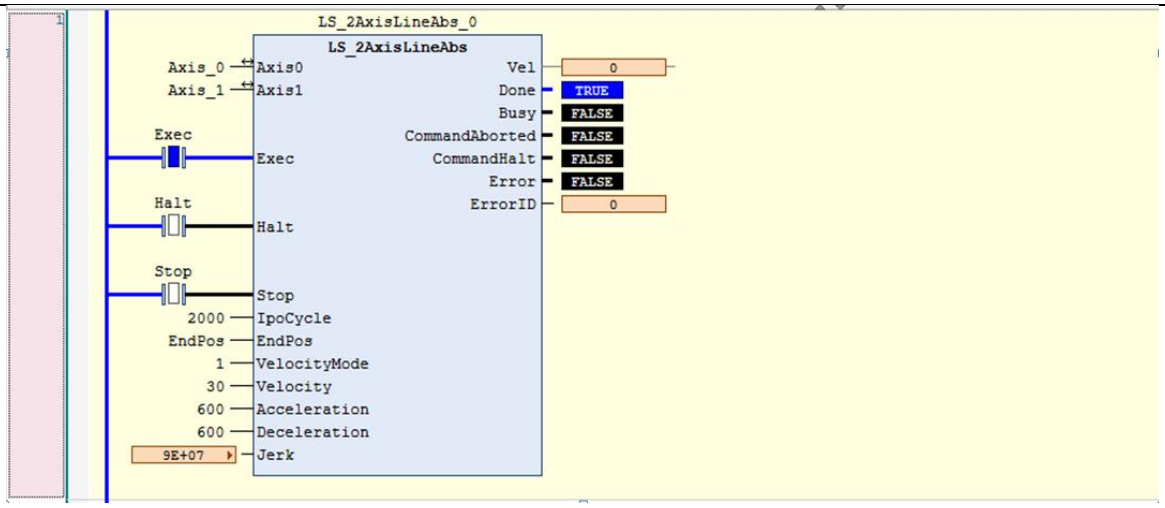
  - 指定的两个轴做直线插补运动，位置坐标为绝对坐标。

- **程序示例**

ST：当 Exec 变为 True 时，执行两轴绝对直线插补指令；程序如下：

```
LS_2AxisLineAbs_0(
    Axis0:= Axis_0,
    Axis1:= Axis_1,
    Exec TRUE := Exec TRUE,
    Halt FALSE := Halt FALSE,
    Stop FALSE := Stop FALSE,
    IpoCycle 2000 := 2000,
    EndPos:= EndPos,
    VelocityMode SIGMOID := 1,
    Velocity 30 := 30,
    Acceleration 600 := 600,
    Deceleration 600 := 600,
    Jerk 9E+07 := jerk 9E+07 ,
    Vel 0 => vel 0 ,
    Done TRUE => done TRUE ,
    Busy FALSE => busy FALSE ,
    CommandAborted FALSE => Abort FALSE ,
    CommandHalt FALSE => ComHalt FALSE ,
    Error FALSE => err FALSE ,
    ErrorID 0 => errid 0 );
```

LD：当 Exec 变为 True 时，执行两轴绝对直线插补指令；程序如下：



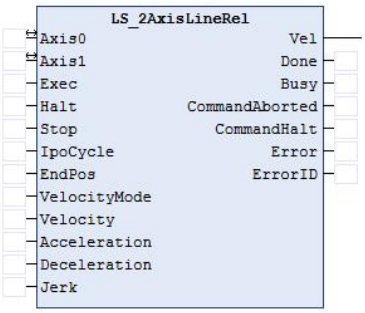
- **注意事项：**

- 这个指令由PMC_Ipolib库实现。
- IpoCycle参数需要与运动指令所在的任务周期时间保持一致，否则指令执行时可能会出现Error报错。
- 参数IpoCycle、Jerk、Velocity、Acceleration、Deceleration不可以设置为0，否则指令执行时可能会出现Error报错。
- 在指令运行期间，主从轴都不能再被其它运动指令所调用。
- Endpos是两插补轴的目标位置（绝对坐标）；Endpos[0]是插补0轴目标位置，Endpos[1]是插补1轴目标位置。
- Velocity、Acceleration、Deceleration、Jerk大小不能超过“轴设置-SoftMotion驱动：通用”里面的速度，加速度，减速度，加加速度的动态限制，否则指令执行时可能会出现Error报错。
- 插补轴需要先把轴上使能（运行MC_POWER指令），才能运行插补指令。
- 本指令根据两个插补轴需要移动的距离来按比例分配插补轴速度，使得两插补轴的合速度为Velocity（30Pulse/s），（需要移动的距离=目标位置-当前位置）。

7.10.3 LS_2AxisLineRel

指定的两个轴做直线插补运动，位置坐标为相对坐标。

- **指令格式**

指令	名称	FB/FC	LD表现	ST表现	库
LS_2AxisLineRel	两轴相对直线插补	FB		LS_2AxisLineRel(Axis0:=, Axis1:=, Exec:=, Halt:=, Stop:=, IpoCycle:=, EndPos:=, VelocityMode:=, Velocity:=, Acceleration:=, Deceleration:=, Jerk:=, Vel=>, Done=>, Busy=>, CommandAborted=>, CommandHalt=>, Error=>, ErrorID=>);	PMC_IpoLib

- **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	初始值	描述
Axis0	轴0	AXIS_REF_VIRTUAL_SM3	-	-	插补0轴
Axis1	轴1	AXIS_REF_VIRTUAL_SM3	-	-	插补1轴

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Exec	启动	BOOL	TRUE-FALSE	FALSE	启动命令，上升沿有效。
Halt	暂停	BOOL	TRUE-FALSE	FALSE	暂停命令，当该输入量为 True 时，当前插补运动暂停；当状态再次切换到 False 时，插补模块继续之前的未完成的插补任务。
Stop	停止	BOOL	TRUE-FALSE	FALSE	停止命令，该命令为 True 时，插补模块停止。
IpoCycle	插补周期	DWORD	-	2000	插补周期，单位：us，必须和所在任务周期一致；
Endpos	目标位置	ARRAY[0..1]OFLREAL	负数，0 正数	0	轴的目标位置，单位：Pulse
VelocityMode	速度模式	SMC_INT_VELMODE	0-3	SIGMOID	速度模式：梯形-0（TRAPEZOID）；S 形-1（SIGMOID）；四次方-3（QUADRATIC）
Velocity	速度	LREAL	0，正数	0	插补合成速度，Pulse/s
Acceleration	加速度	LREAL	0，正数	0	插补合成加速度，Pulse/s2
Deceleration	减速度	LREAL	0，正数	0	插补合成减速度，Pulse/s2
Jerk	加加速度	LREAL	负数，0 正数	90000000	加加速度，速度模式为 3 时需要设置此参数，该参数值不能为 0。

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
Vel	合速度	LREAL	0，正数	0	当前插补运动合成速度。
Done	完成	BOOL	TRUE-FALSE	FALSE	True 表示插补模块已完成插补任务。
Busy	运动中	BOOL	TRUE-FALSE	FALSE	True 表示插补运动正在进行之中。
CommandAborted	运动中断	BOOL	TRUE-FALSE	FALSE	当 Stop 有效时，该变量为 True，表示插补模块非正常中止。
CommandHalt	运动暂停	BOOL	TRUE-FALSE	FALSE	当 Halt 为 True 时，该变量为 True，表示插补模块处于暂停状态。
Error	错误	BOOL	TRUE-FALSE	FALSE	错误状态；False-没有错误。
ErrorID	错误代码	DINT	0，正数	0	错误代码

	布尔	位串				整数								实数		时刻、持续时间 日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Axis0	AXIS_REF_VIRTUAL_SM3																			
Axis1	AXIS_REF_VIRTUAL_SM3																			
Exec	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Halt	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Stop	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
IpoCycle	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Endpos	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
VelocityMode	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Velocity	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–
Acceleration	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–
Deceleration	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–
Jerk	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–
Vel	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–
Done	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

Busy	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
CommandAborted	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
CommandHalt	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Error	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
ErrorID	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–

- **功能说明：**

  - 指定的两个轴做直线插补运动，位置坐标为相对坐标。

- **程序示例**

ST：当 Exec 变为 True 时，执行两轴相对直线插补指令；程序如下：

```
LS_2AxisLineRel_0(
    Axis0:= Axis_0,
    Axis1:= Axis_1,
    Exec TRUE := Exec TRUE,
    Halt FALSE := Halt FALSE,
    Stop FALSE := Stop FALSE,
    IpoCycle 2000 := 2000,
    EndPos:= EndPos,
    VelocityMode SIGMOID := 1,
    Velocity 30 := 30,
    Acceleration 600 := 600,
    Deceleration 600 := 600,
    Jerk 9E+07 := jerk 9E+07 ,
    Vel 0 => vel 0 ,
    Done TRUE => done TRUE ,
    Busy FALSE => busy FALSE ,
    CommandAborted FALSE => Abort FALSE ,
    CommandHalt FALSE => ComHalt FALSE ,
    Error FALSE => err FALSE ,
    ErrorID 0 => errid 0 );RETURN
```

LD：当Exec变为True时，执行两轴相对直线插补指令；程序如下：



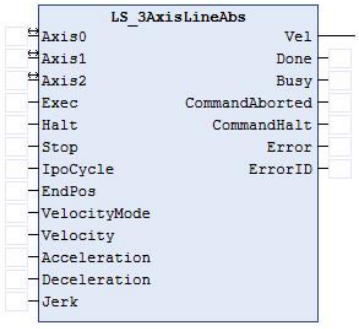
- **注意事项**

- 这个指令由“PMC_Ipolib”库实现。
- IpoCycle参数需要与运动指令所在的任务周期时间保持一致，否则指令执行时可能会出现Error报错。
- 参数IpoCycle、Jerk、Velocity、Acceleration、Deceleration不可以设置为0。
- 在指令运行期间，主从轴都不能再被其它运动指令所调用。
- Endpos是两插补轴的目标位置，是相对坐标；Endpos[0]是插补0轴需要移动的距离，Endpos[1]是插补1轴需要移动的距离。
- Velocity、Acceleration、Deceleration、Jerk大小不能超过“轴设置-SoftMotion驱动：通用”里面的速度，加速度，减速度，加加速度的动态限制，否则指令执行时可能会出现Error报错。
- 插补轴需要先把轴上使能（运行MC_POWER指令），才能运行插补指令。
- 本指令根据两个插补轴需要移动的距离来按比例分配插补轴速度，使得两插补轴的合速度为Velocity（30Pulse/s）。
- 相对坐标简单理解：相对于起始位置的坐标，即将起始位置作为坐标原点建立的坐标系里的坐标。

7.10.4 LS_3AxisLineAbs

指定的三个轴做直线插补运动，位置坐标为绝对坐标。

- **指令格式**

指令	名称	FB/FC	LD表现	ST表现	库
LS_3AxisLineAbs	三轴绝对直线插补	FB		LS_3AxisLineAbs(Axis0:=, Axis1:=, Axis2:=, Exec:=, Halt:=, Stop:=,	PMC_IpoLib

				IpoCycle:=, EndPos:=, VelocityMode:=, Velocity:=, Acceleration:=, Deceleration:=, Jerk:=, Vel=>, Done=>, Busy=>, CommandAborted=>, CommandHalt=>, Error=>, ErrorID=>);	

- 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	初始值	描述
Axis0	轴0	AXIS_REF_VIRTUAL_SM3	-	-	插补0轴
Axis1	轴1	AXIS_REF_VIRTUAL_SM3	-	-	插补1轴
Axis2	轴2	AXIS_REF_VIRTUAL_SM3	-	-	插补2轴

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Exec	启动	BOOL	TRUE-FALSE	FALSE	启动命令，上升沿有效。
Halt	暂停	BOOL	TRUE-FALSE	FALSE	暂停命令，当该输入量为 True 时，当前插补运动暂停；当状态再次切换到 False 时，插补模块继续之前的未完成

					的插补任务。
Stop	停止	BOOL	TRUE-FALSE	FALSE	停止命令，该命令为 True 时，插补模块停止。
IpoCycle	插补周期	DWORD	-	2000	插补周期，单位：us，必须和所在任务周期一致；
Endpos	目标位置	ARRAY[0..1]OFLREAL	负数，0 正数	0	轴的目标位置，单位：Pulse
VelocityMode	速度模式	SMC_INT_VELMODE	0-3	SIGMOID	速度模式：梯形-0（TRAPEZOID）；S 形-1（SIGMOID）；四次方-3（QUADRATIC）
Velocity	速度	LREAL	0，正数	0	插补合成速度，Pulse/s
Acceleration	加速度	LREAL	0，正数	0	插补合成加速度，Pulse/s2
Deceleration	减速度	LREAL	0，正数	0	插补合成减速度，Pulse/s2
Jerk	加加速度	LREAL	负数，0 正数	90000000	加加速度，速度模式为 3 时需要设置此参数，该参数值不能为 0。

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
Vel	合速度	LREAL	0，正数	0	当前插补运动合成速度。
Done	完成	BOOL	TRUE-FALSE	FALSE	True 表示插补模块已完成插补任务。
Busy	运动中	BOOL	TRUE-FALSE	FALSE	True 表示插补运动正在进行之中。
CommandAborted	运动中断	BOOL	TRUE-FALSE	FALSE	当 Stop 有效时，该变量为 True，表示插补模块非正常

					中止。
CommandHalt	运动暂停	BOOL	TRUE-FALSE	FALSE	当 Halt 为 True 时，该变量为 True，表示插补模块处于暂停状态。
Error	错误	BOOL	TRUE-FALSE	FALSE	错误状态；False-没有错误。
ErrorID	错误代码	DINT	0，正数	0	错误代码

	布尔	位串				整数								实数		时刻、持续时间 日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Axis0	AXIS_REF_VIRTUAL_SM3																			
Axis1	AXIS_REF_VIRTUAL_SM3																			
Axis2	AXIS_REF_VIRTUAL_SM3																			
Exec	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Halt	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Stop	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
IpoCycle	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Endpos	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
VelocityMode	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Velocity	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–
Acceleration	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–
Deceleration	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–
Jerk	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–
Vel	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–
Done	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Busy	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
CommandAborted	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
CommandHalt	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Error	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
ErrorID	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–

- **功能说明：**

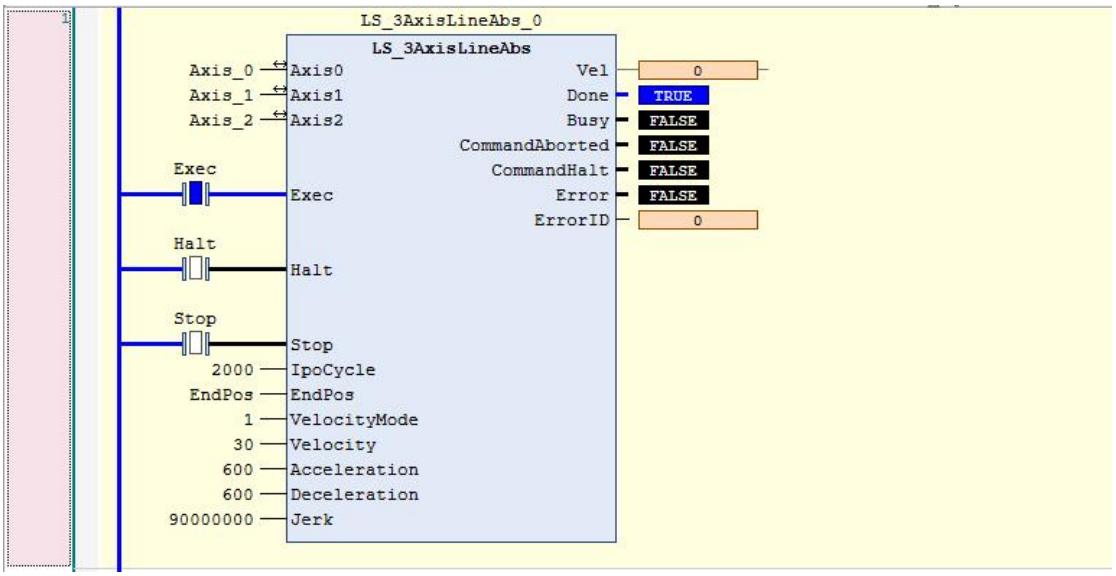
  - 指定的三个轴做直线插补运动，位置坐标为绝对坐标。

- **程序示例**

ST：当 Exec 变为 True 时，执行三轴相对直线插补指令；程序如下：

```
LS_3AxisLineAbs_0(
    Axis0:= Axis_0,
    Axis1:= Axis_1,
    Axis2:= Axis_2,
    Exec TRUE := Exec TRUE,
    Halt FALSE := Halt FALSE,
    Stop FALSE := Stop FALSE,
    IpoCycle 2000 := 2000,
    EndPos:= EndPos,
    VelocityMode SIGMOID := 1,
    Velocity 30 := 30,
    Acceleration 600 := 600,
    Deceleration 600 := 600,
    Jerk 9E+07 := jerk 9E+07,
    Vel 0 => vel 0,
    Done TRUE => done TRUE,
    Busy FALSE => busy FALSE,
    CommandAborted FALSE => Abort FALSE,
    CommandHalt FALSE => ComHalt FALSE,
    Error FALSE => err FALSE,
    ErrorID 0 => errid 0 );
```

LD：当 Exec 变为 True 时，执行三轴相对直线插补指令；程序如下：



- **注意事项**

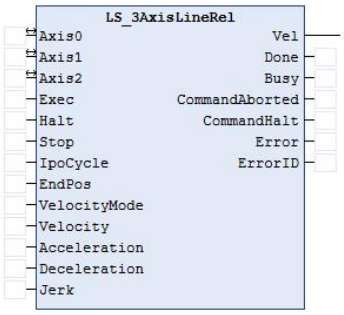
- 这个指令由“PMC_Ipolib”库实现。
- IpoCycle参数需要与运动指令所在的任务周期时间保持一致，否则指令执行时可能会出现Error报错。

- 参数IpoCycle、Jerk、Velocity、Acceleration、Deceleration不可以设置为0。
- 在指令运行期间，三个轴都不能再被其它运动指令所调用。
- Endpos是三个插补轴的目标位置（绝对坐标）；Endpos[0]是插补0轴目标位置，Endpos[1]是插补1轴目标位置，Endpos[2]是插补2轴目标位置。
- Velocity、Acceleration、Deceleration、Jerk大小不能超过“轴设置-SoftMotion驱动：通用”里面的速度，加速度，减速度，加加速度的动态限制，否则指令执行时可能会出现Error报错。
- 本指令根据三个插补轴需要移动的距离来按比例分配插补轴速度，使得三个插补轴的合速度为Velocity（30Pulse/s），（需要移动的距离=目标位置-当前位置）。

7.10.5 LS_3AxisLineRel

指定的三个轴做直线插补运动，位置坐标为相对坐标。

- **指令格式**

指令	名称	FB/FC	LD表现	ST表现	库
LS_3AxisLineRel	三轴相对直线插补	FB		LS_3AxisLineRel(Axis0:=, Axis1:=, Axis2:=, Exec:=, Halt:=, Stop:=, IpoCycle:=, EndPos:=, VelocityMode:=, Velocity:=, Acceleration:=, Deceleration:=, Jerk:=, Vel=>, Done=>, Busy=>, CommandAborted=>	PMC_IpoLib

				, CommandHalt=>, Error=>, ErrorID=>);	

- 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	初始值	描述
Axis0	轴0	AXIS_REF_VIRTUAL_SM3	-	-	插补0轴
Axis1	轴1	AXIS_REF_VIRTUAL_SM3	-	-	插补1轴
Axis2	轴2	AXIS_REF_VIRTUAL_SM3	-	-	插补2轴

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Exec	启动	BOOL	TRUE-FALSE	FALSE	启动命令，上升沿有效。
Halt	暂停	BOOL	TRUE-FALSE	FALSE	暂停命令，当该输入量为 True 时，当前插补运动暂停；当状态再次切换到 False 时，插补模块继续之前的未完成的插补任务。
Stop	停止	BOOL	TRUE-FALSE	FALSE	停止命令，该命令为 True 时，插补模块停止。
IpoCycle	插补周期	DWORD	-	2000	插补周期，单位：us，必须和所在任务周期一致；
Endpos	目标位置	ARRAY[0..1]OFLREAL	负数，0 正数	0	轴的目标位置，单位：Pulse

VelocityMode	速度模式	SMC_INT_VELMODE	0-3	SIGMOID	速度模式：梯形-0（TRAPEZOID）；S 形-1（SIGMOID）；四次方-3（QUADRATIC）
Velocity	速度	LREAL	0，正数	0	插补合成速度，Pulse/s
Acceleration	加速度	LREAL	0，正数	0	插补合成加速度，Pulse/s2
Deceleration	减速度	LREAL	0，正数	0	插补合成减速度，Pulse/s2
Jerk	加加速度	LREAL	负数，0 正数	90000000	加加速度，速度模式为 3 时需要设置此参数，该参数值不能为 0。

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
Vel	合速度	LREAL	0，正数	0	当前插补运动合成速度。
Done	完成	BOOL	TRUE-FALSE	FALSE	True 表示插补模块已完成插补任务。
Busy	运动中	BOOL	TRUE-FALSE	FALSE	True 表示插补运动正在进行之中。
CommandAborted	运动中断	BOOL	TRUE-FALSE	FALSE	当 Stop 有效时，该变量为 True，表示插补模块非正常中止。
CommandHalt	运动暂停	BOOL	TRUE-FALSE	FALSE	当 Halt 为 True 时，该变量为 True，表示插补模块处于暂停状态。
Error	错误	BOOL	TRUE-FALSE	FALSE	错误状态；False-没有错误。
ErrorID	错误代码	DINT	0，正数	0	错误代码

	布尔	位串	整数	实数	时刻、持续时间 日期、字符串

	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Axis0	AXIS_REF_VIRTUAL_SM3																			
Axis1	AXIS_REF_VIRTUAL_SM3																			
Axis2	AXIS_REF_VIRTUAL_SM3																			
Exec	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Halt	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Stop	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
IpoCycle	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Endpos	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
VelocityMode	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Velocity	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–
Acceleration	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–
Deceleration	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–
Jerk	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–
Vel	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–
Done	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Busy	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
CommandAborted	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
CommandHalt	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Error	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
ErrorID	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–

- **功能说明：**

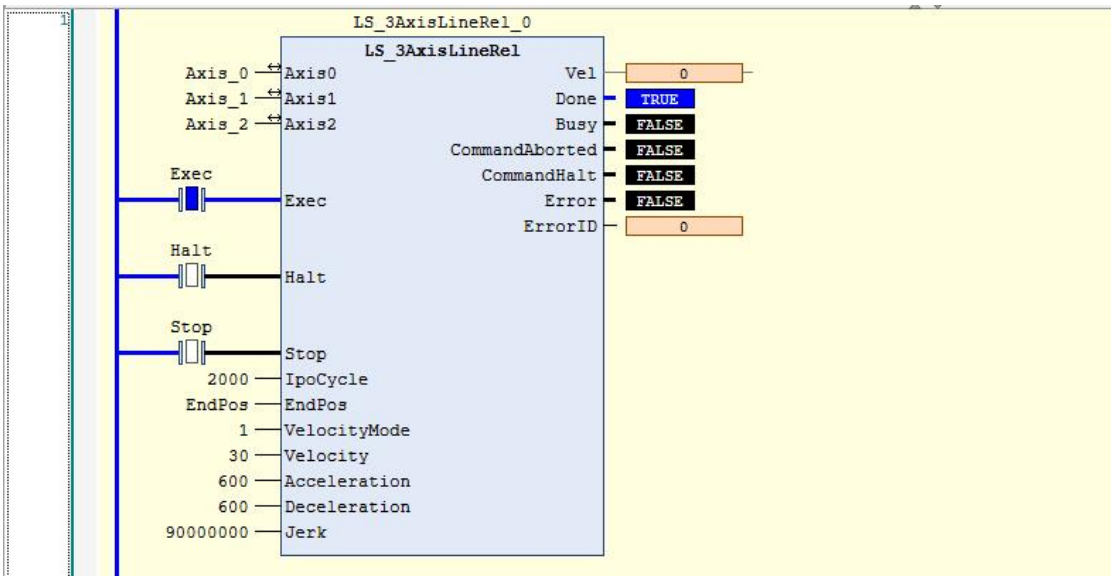
  - 指定的三个轴做直线插补运动，位置坐标为相对坐标。

- **程序示例**

ST：当 Exec 变为 True 时，执行三轴相对直线插补指令；程序如下：

```
LS_3AxisLineRel_0(
    Axis0:= Axis_0,
    Axis1:= Axis_1,
    Axis2:= Axis_2,
    Exec TRUE := Exec TRUE,
    Halt FALSE := Halt FALSE,
    Stop FALSE := Stop FALSE,
    IpoCycle 2000 := 2000,
    EndPos:= EndPos,
    VelocityMode SIGMOID := 1,
    Velocity 30 := 30,
    Acceleration 600 := 600,
    Deceleration 600 := 600,
    Jerk 9E+07 := jerk 9E+07 ,
    Vel 0 => vel 0 ,
    Done TRUE => done TRUE ,
    Busy FALSE => busy FALSE ,
    CommandAborted FALSE => Abort FALSE ,
    CommandHalt FALSE => ComHalt FALSE ,
    Error FALSE => err FALSE ,
    ErrorID 0 => errid 0 );
```

LD：当Exec变为True时，执行三轴相对直线插补指令；程序如下：



- **注意事项**

➢ 这个指令由“PMC_Ipolib”库实现。

➢ IpoCycle参数需要与运动指令所在的任务周期时间保持一致，否则指令执行时可能会出现Error报错。

➢ 参数IpoCycle、Jerk、Velocity、Acceleration、Deceleration不可以设置为0。

➢ 在指令运行期间，三个轴都不能再被其它运动指令所调用。

➢ Endpos是三个插补轴需要移动的距离；Endpos[0]是插补0轴需要移动的距离，Endpos[1]

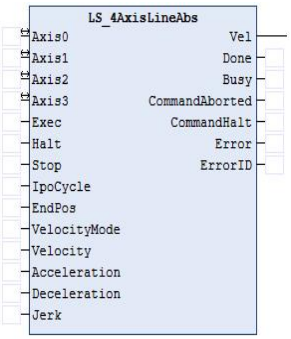
是插补1轴需要移动的距离，Endpos[2]是插补2轴需要移动的距离。

- Velocity、Acceleration、Deceleration、Jerk大小不能超过“轴设置-SoftMotion驱动：通用”里面的速度，加速度，减速度，加加速度的动态限制，否则指令执行时可能会出现Error报错。
- 本指令根据三个插补轴需要移动的距离来按比例分配插补轴速度，使得三个插补轴的合速度为Velocity（30Pulse/s）。
- 相对坐标简单理解：相对于起始位置的坐标，即将起始位置作为坐标原点建立的坐标系里的坐标。

7.10.6 LS_4AxisLineAbs

指定的四个轴做直线插补运动，位置坐标为绝对坐标。（前三轴做空间直线插补运动，后一轴做跟随运动）

- **指令格式**

指令	名称	FB/FC	LD表现	ST表现	库
LS_4AxisLineAbs	四轴绝对直线插补	FB		LS_4AxisLineAbs(Axis0:=, Axis1:=, Axis2:=, Axis3:=, Exec:=, Halt:=, Stop:=, IpoCycle:=, EndPos:=, VelocityMode:=, Velocity:=, Acceleration:=, Deceleration:=, Jerk:=, Vel=>, Done=>, Busy=>, CommandAborted=>, CommandHalt=>,	PMC_IpoLib

				Error=>, ErrorID=>);	

- **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	初始值	描述
Axis0	轴0	AXIS_REF_VIRTUAL_SM3	-	-	插补0轴
Axis1	轴1	AXIS_REF_VIRTUAL_SM3	-	-	插补1轴
Axis2	轴2	AXIS_REF_VIRTUAL_SM3	-	-	插补2轴
Axis3	轴3	AXIS_REF_VIRTUAL_SM3	-	-	跟随 3 轴

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Exec	启动	BOOL	TRUE-FALSE	FALSE	启动命令，上升沿有效。
Halt	暂停	BOOL	TRUE-FALSE	FALSE	暂停命令，当该输入量为 True 时，当前插补运动暂停；当状态再次切换到 False 时，插补模块继续之前的未完成的插补任务。
Stop	停止	BOOL	TRUE-FALSE	FALSE	停止命令，该命令为 True 时，插补模块停止。
IpoCycle	插补周期	DWORD	-	2000	插补周期，单位：us，必须和所在任务周期一致；

Endpos	目标位置	ARRAY[0..1]OFLREAL	负数，0 正数	0	轴的目标位置，单位：Pulse
Velocity Mode	速度模式	SMC_INT_VELMODE	0-3	SIGMOID	速度模式：梯形-0（TRAPEZOID）；S 形-1（SIGMOID）；四次方-3（QUADRATIC）
Velocity	速度	LREAL	0，正数	0	插补合成速度，Pulse/s
Acceleration	加速度	LREAL	0，正数	0	插补合成加速度，Pulse/s2
Deceleration	减速度	LREAL	0，正数	0	插补合成减速度，Pulse/s2
Jerk	加加速度	LREAL	负数，0 正数	90000000	加加速度，速度模式为 3 时需要设置此参数，该参数值不能为 0。

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
Vel	合速度	LREAL	0，正数	0	当前插补运动合成速度。
Done	完成	BOOL	TRUE-FALSE	FALSE	True 表示插补模块已完成插补任务。
Busy	运动中	BOOL	TRUE-FALSE	FALSE	True 表示插补运动正在进行之中。
CommandAborted	运动中断	BOOL	TRUE-FALSE	FALSE	当 Stop 有效时，该变量为 True，表示插补模块非正常中止。
CommandHalt	运动暂停	BOOL	TRUE-FALSE	FALSE	当 Halt 为 True 时，该变量为 True，表示插补模块处于暂停状态。
Error	错误	BOOL	TRUE-FALSE	FALSE	错误状态；False-没有错误。
ErrorID	错误代码	DINT	0，正数	0	错误代码

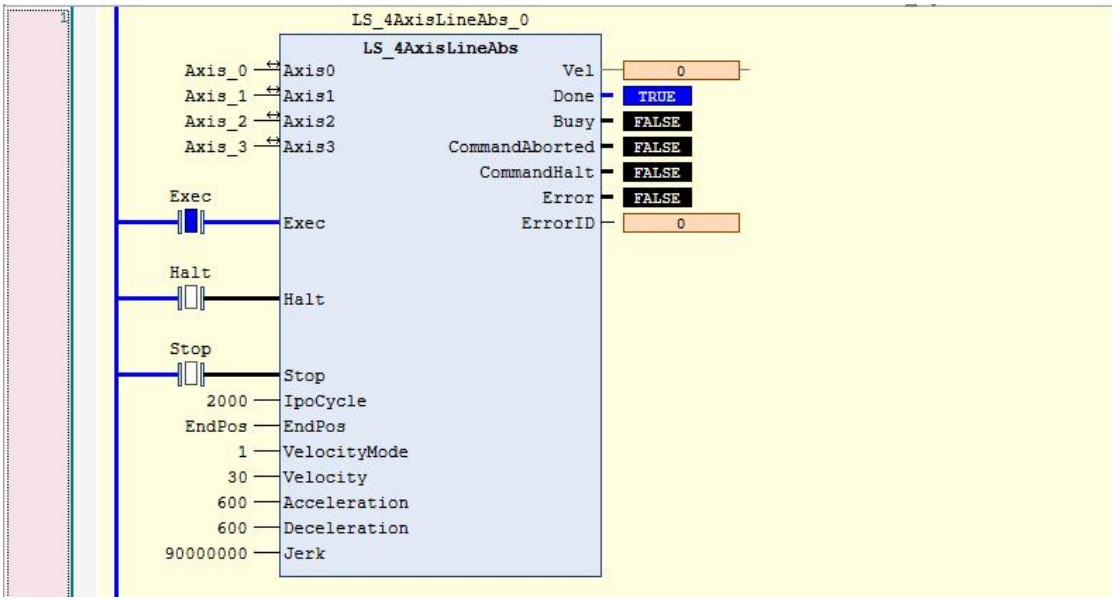
	布尔	位串				整数								实数		时刻、持续时间 日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Axis0	AXIS_REF_VIRTUAL_SM3																			
Axis1	AXIS_REF_VIRTUAL_SM3																			
Axis2	AXIS_REF_VIRTUAL_SM3																			
Axis3	AXIS_REF_VIRTUAL_SM3																			
Exec	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Halt	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Stop	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
IpoCycle	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Endpos	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
VelocityMode	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Velocity	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–
Acceleration	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–
Deceleration	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–
Jerk	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–
Vel	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–
Done	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Busy	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
CommandAborted	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
CommandHalt	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Error	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
ErrorID	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–

- **功能说明：**
  - 指定的四个轴做直线插补运动，位置坐标为绝对坐标（前三轴做空间直线插补运动，后一轴做跟随运动）
- **程序示例**

ST：当 Exec 变为 True 时，执行四轴绝对直线插补指令；程序如下：

```
LS_4AxisLineAbs_0(
    Axis0:= Axis_0,
    Axis1:= Axis_1,
    Axis2:= Axis_2,
    Axis3:= Axis_3,
    Exec TRUE := Exec TRUE,
    Halt FALSE := Halt FALSE,
    Stop FALSE := Stop FALSE,
    IpoCycle 2000 := 2000,
    EndPos:= EndPos,
    VelocityMode SIGMOID := 1,
    Velocity 30 := 30,
    Acceleration 600 := 600,
    Deceleration 600 := 600,
    Jerk 9E+07 := jerk 9E+07,
    Vel 0 => vel 0,
    Done TRUE => done TRUE,
    Busy FALSE => busy FALSE,
    CommandAborted FALSE => Abort FALSE,
    CommandHalt FALSE => ComHalt FALSE,
    Error FALSE => err FALSE,
    ErrorID 0 => errid 0);
```

LD：当Exec变为True时，执行四轴绝对直线插补指令；程序如下：



- **注意事项**

➢ 这个指令由“PMC_Ipolib”库实现。

➢ IpoCycle参数需要与运动指令所在的任务周期时间保持一致，否则指令执行时可能会出现Error报错。

➢ 参数IpoCycle、Jerk、Velocity、Acceleration、Deceleration不可以设置为0。

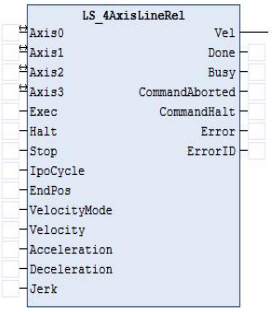
➢ 在指令运行期间，四个轴都不能再被其它运动指令所调用。

- Endpos是四个插补轴的目标位置（绝对坐标）；Endpos[0]是插补0轴目标位置，Endpos[1]是插补1轴目标位置，Endpos[2]是插补2轴目标位置，Endpos[3]是插补3轴目标位置。
- Velocity、Acceleration、Deceleration、Jerk大小不能超过“轴设置-SoftMotion驱动：通用”里面的速度，加速度，减速度，加加速度的动态限制，否则指令执行时可能会出现Error报错。
- 四轴插补直线运动中前三个轴做空间直线插补运动，第四个轴做跟随运动，和参与插补运动的轴同时启停。
- 根据四轴需要移动的距离来按比例分配各轴速度，Vel是前三个插补轴的当前合速度，（需要移动的距离=目标位置-当前位置）。

7.10.7 LS_4AxisLineRel

指定的四个轴做直线插补运动，位置坐标为绝对坐标。（前三轴做空间直线插补运动，后一轴做跟随运动）

- **指令格式**

指令	名称	FB/FC	LD表现	ST表现	库
LS_4AxisLineRel	四轴相对直线插补	FB		LS_4AxisLineRel(Axis0:=, Axis1:=, Axis2:=, Axis3:=, Exec:=, Halt:=, Stop:=, IpoCycle:=, EndPos:=, VelocityMode:=, Velocity:=, Acceleration:=, Deceleration:=, Jerk:=, Vel=>, Done=>,	PMC_IpoLib

				Busy=>, CommandAborted=>, CommandHalt=>, Error=>, ErrorID=>);	

- 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	初始值	描述
Axis0	轴0	AXIS_REF_VIRTUAL_SM3	-	-	插补0轴
Axis1	轴1	AXIS_REF_VIRTUAL_SM3	-	-	插补1轴
Axis2	轴2	AXIS_REF_VIRTUAL_SM3	-	-	插补2轴
Axis3	轴3	AXIS_REF_VIRTUAL_SM3	-	-	跟随 3 轴

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Exec	启动	BOOL	TRUE-FALSE	FALSE	启动命令，上升沿有效。
Halt	暂停	BOOL	TRUE-FALSE	FALSE	暂停命令，当该输入量为 True 时，当前插补运动暂停；当状态再次切换到 False 时，插补模块继续之前的未完成的插补任务。
Stop	停止	BOOL	TRUE-FALSE	FALSE	停止命令，该命令为 True 时，插补模块停止。
IpoCycle	插补周期	DWORD	-	2000	插补周期，单位：us，必须和所在任务周期一致；

Endpos	目标位置	ARRAY[0..1]OFLREAL	负数，0，正数	0	轴的目标位置，单位：Pulse
Velocity Mode	速度模式	SMC_INT_VELMODE	0-3	SIGMOID	速度模式：梯形-0（TRAPEZOID）；S 形-1（SIGMOID）；四次方-3（QUADRATIC）
Velocity	速度	LREAL	0，正数	0	插补合成速度，Pulse/s
Acceleration	加速度	LREAL	0，正数	0	插补合成加速度，Pulse/s2
Deceleration	减速度	LREAL	0，正数	0	插补合成减速度，Pulse/s2
Jerk	加加速度	LREAL	负数，0，正数	90000000	加加速度，速度模式为 3 时需要设置此参数，该参数值不能为 0。

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
Vel	合速度	LREAL	0，正数	0	当前插补运动合成速度。
Done	完成	BOOL	TRUE-FALSE	FALSE	True 表示插补模块已完成插补任务。
Busy	运动中	BOOL	TRUE-FALSE	FALSE	True 表示插补运动正在进行之中。
CommandAborted	运动中断	BOOL	TRUE-FALSE	FALSE	当 Stop 有效时，该变量为 True，表示插补模块非正常中止。
CommandHalt	运动暂停	BOOL	TRUE-FALSE	FALSE	当 Halt 为 True 时，该变量为 True，表示插补模块处于暂停状态。
Error	错误	BOOL	TRUE-FALSE	FALSE	错误状态；False-没有错误。
ErrorID	错误代码	DINT	0，正数	0	错误代码

	布尔	位串				整数								实数		时刻、持续时间 日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Axis0	AXIS_REF_VIRTUAL_SM3																			
Axis1	AXIS_REF_VIRTUAL_SM3																			
Axis2	AXIS_REF_VIRTUAL_SM3																			
Axis3	AXIS_REF_VIRTUAL_SM3																			
Exec	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Halt	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Stop	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
IpoCycle	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Endpos	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
VelocityMode	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Velocity	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–
Acceleration	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–
Deceleration	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–
Jerk	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–
Vel	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–
Done	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Busy	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
CommandAborted	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
CommandHalt	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Error	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
ErrorID	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–

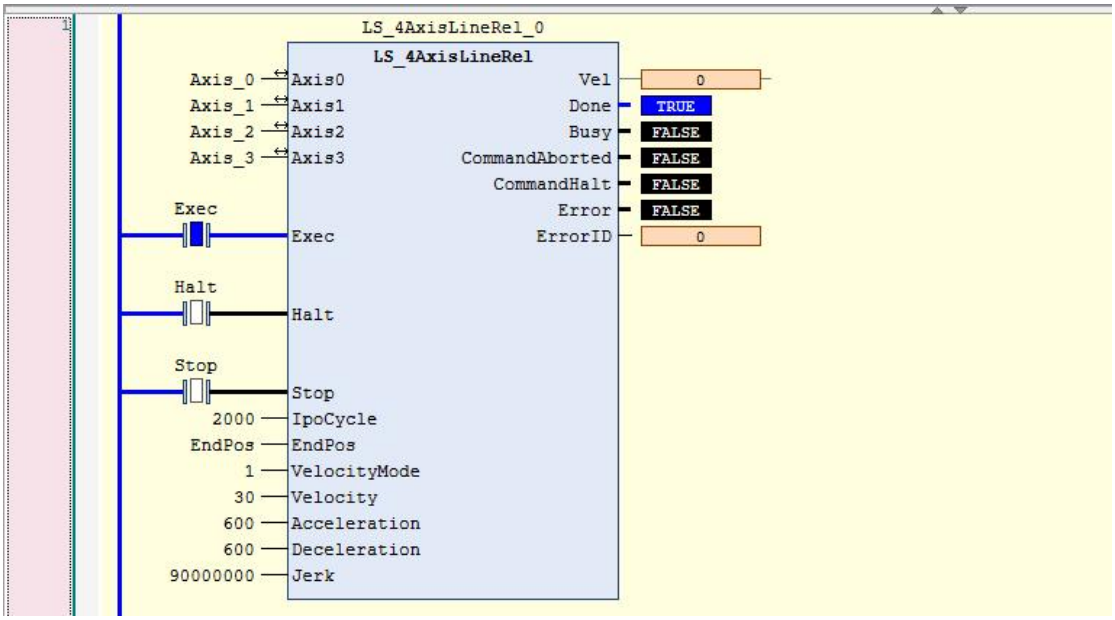
- **功能说明：**
  - 指定的四个轴做直线插补运动，位置坐标为相对坐标（前三轴做空间直线插补运动，后一轴做跟随运动）

- **程序示例**

ST：当 Exec 变为 True 时，执行四轴绝对直线插补指令；程序如下：

```
LS_4AxisLineRel_0(
    Axis0:= Axis_0,
    Axis1:= Axis_1,
    Axis2:= Axis_2,
    Axis3:= Axis_3,
    Exec TRUE := Exec TRUE,
    Halt FALSE := Halt FALSE,
    Stop FALSE := Stop FALSE,
    IpoCycle 2000 := 2000,
    EndPos:= EndPos,
    VelocityMode SIGMOID := 1,
    Velocity 30 := 30,
    Acceleration 600 := 600,
    Deceleration 600 := 600,
    Jerk 9E+07 := jerk 9E+07,
    Vel 0 => vel 0,
    Done TRUE => done TRUE,
    Busy FALSE => busy FALSE,
    CommandAborted FALSE => Abort FALSE,
    CommandHalt FALSE => ComHalt FALSE,
    Error FALSE => err FALSE,
    ErrorID 0 => errid 0 );
```

LD：当 Exec 变为 True 时，执行四轴相对直线插补指令；程序如下：



- **注意事项**

➢ 这个指令由“PMC_Ipolib”库实现。

➢ IpoCycle参数需要与运动指令所在的任务周期时间保持一致，否则指令执行时可能会出现Error报错。

➢ 参数IpoCycle、Jerk、Velocity、Acceleration、Deceleration不可以设置为0。

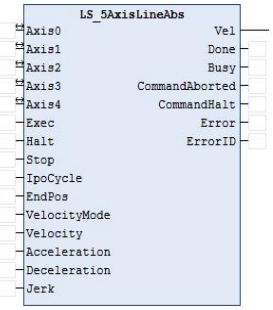
➢ 在指令运行期间，四个轴都不能再被其它运动指令所调用。

- Endpos是四个插补轴需要移动的距离；Endpos[0]是插补0轴需要移动的距离，Endpos[1]是插补1轴需要移动的距离，Endpos[2]是插补2轴需要移动的距离，Endpos[3]是插补3轴需要移动的距离。
- Velocity、Acceleration、Deceleration、Jerk大小不能超过“轴设置-SoftMotion驱动：通用”里面的速度，加速度，减速度，加加速度的动态限制，否则指令执行时可能会出现Error报错。
- 四轴插补直线运动中前三个轴做空间直线插补运动，第四个轴做跟随运动，和参与插补运动的轴同时启停。
- 根据四轴需要移动的距离来按比例分配各轴速度，Vel是前三个插补轴的当前合速度。
- 相对坐标简单理解：相对于起始位置的坐标，即将起始位置作为坐标原点建立的坐标系里的坐标。

7.10.8 LS_5AxisLineAbs

指定的五个轴做直线插补运动，位置坐标为绝对坐标。（前三轴做空间直线插补运动，后两轴做跟随运动）

- **指令格式**

指令	名称	FB/FC	LD表现	ST表现	库
LS_5AxisLineAbs	五轴绝对直线插补	FB		LS_5AxisLineAbs(Axis0:=, Axis1:=, Axis2:=, Axis3:=, Axis4:=, Exec:=, Halt:=, Stop:=, IpoCycle:=, EndPos:=, VelocityMode:=, Velocity:=,	PMC_IpoLib

				Acceleration:=, Deceleration:=, Jerk:=, Vel=>, Done=>, Busy=>, 、 CommandAborted=>, CommandHalt=>, Error=>, ErrorID=>);	

- **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	初始值	描述
Axis0	轴0	AXIS_REF_VIRTUAL_SM3	-	-	插补0轴
Axis1	轴1	AXIS_REF_VIRTUAL_SM3	-	-	插补1轴
Axis2	轴2	AXIS_REF_VIRTUAL_SM3	-	-	插补2轴
Axis3	轴3	AXIS_REF_VIRTUAL_SM3	-	-	跟随 3 轴
Axis4	轴4	AXIS_REF_VIRTUAL_SM3	-	-	跟随 4 轴

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Exec	启动	BOOL	TRUE-FALSE	FALSE	启动命令，上升沿有效。
Halt	暂停	BOOL	TRUE-FALSE	FALSE	暂停命令，当该输入量为 True 时，当前插补运动暂停；当状态再次切换到

					False 时，插补模块继续之前的未完成的插补任务。
Stop	停止	BOOL	TRUE-FALSE	FALSE	停止命令，该命令为 True 时，插补模块停止。
IpoCycle	插补周期	DWORD	-	2000	插补周期，单位：us，必须和所在任务周期一致；
Endpos	目标位置	ARRAY[0..1]OFLREAL	负数，0 正数	0	轴的目标位置，单位：Pulse
Velocity Mode	速度模式	SMC_INT_VELMODE	0-3	SIGMOID	速度模式：梯形-0（TRAPEZOID）；S 形-1（SIGMOID）；四次方-3（QUADRATIC）
Velocity	速度	LREAL	0，正数	0	插补合成速度，Pulse/s
Acceleration	加速度	LREAL	0，正数	0	插补合成加速度，Pulse/s2
Deceleration	减速度	LREAL	0，正数	0	插补合成减速度，Pulse/s2
Jerk	加加速度	LREAL	负数，0 正数	90000000	加加速度，速度模式为 3 时需要设置此参数，该参数值不能为 0。

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
Vel	合速度	LREAL	0，正数	0	当前插补运动合成速度。
Done	完成	BOOL	TRUE-FALSE	FALSE	True 表示插补模块已完成插补任务。
Busy	运动中	BOOL	TRUE-FALSE	FALSE	True 表示插补运动正在进行之中。
CommandAborted	运动中断	BOOL	TRUE-FALSE	FALSE	当 Stop 有效时，该变量为 True，表示插补模块非正常

					中止。
CommandHalt	运动暂停	BOOL	TRUE-FALSE	FALSE	当 Halt 为 True 时，该变量为 True，表示插补模块处于暂停状态。
Error	错误	BOOL	TRUE-FALSE	FALSE	错误状态；False-没有错误。
ErrorID	错误代码	DINT	0，正数	0	错误代码

	布尔	位串				整数								实数		时刻、持续时间 日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Axis0	AXIS_REF_VIRTUAL_SM3																			
Axis1	AXIS_REF_VIRTUAL_SM3																			
Axis2	AXIS_REF_VIRTUAL_SM3																			
Axis3	AXIS_REF_VIRTUAL_SM3																			
Axis4	AXIS_REF_VIRTUAL_SM3																			
Exec	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Halt	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Stop	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
IpoCycle	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Endpos	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
VelocityMode	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Velocity	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–
Acceleration	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–
Deceleration	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–
Jerk	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–
Vel	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–
Done	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Busy	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
CommandAborted	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
CommandHalt	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Error	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
ErrorID	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–

- **功能说明：**

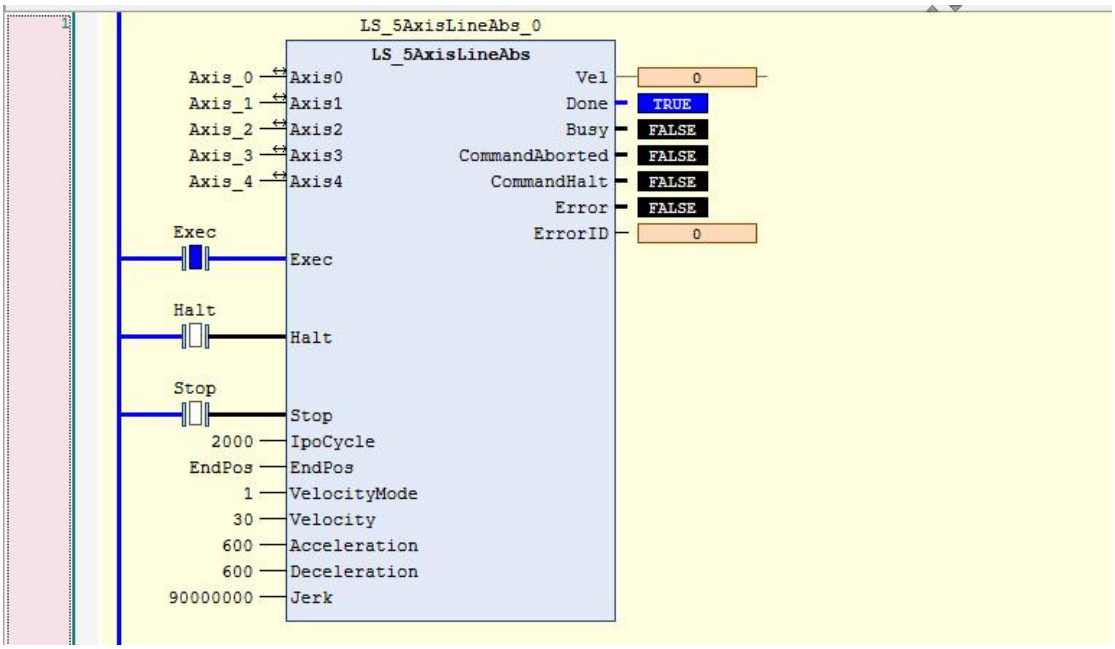
  - 指定的五个轴做直线插补运动，位置坐标为绝对坐标。（前三轴做空间直线插补运动，后两轴做跟随运动）

- **程序示例**

ST：当 Exec 变为 True 时，执行五轴绝对直线插补指令；程序如下：

```
LS_5AxisLineAbs_0(
    Axis0:= Axis_0,
    Axis1:= Axis_1,
    Axis2:= Axis_2,
    Axis3:= Axis_3,
    Axis4:= Axis_4,
    Exec TRUE := Exec TRUE,
    Halt FALSE := Halt FALSE,
    Stop FALSE := Stop FALSE,
    IpoCycle 2000 := 2000,
    EndPos:= EndPos,
    VelocityMode SIGMOID := 1,
    Velocity 30 := 30,
    Acceleration 600 := 600,
    Deceleration 600 := 600,
    Jerk 9E+07 := jerk 9E+07 ,
    Vel 0 => vel 0 ,
    Done TRUE => done TRUE ,
    Busy FALSE => busy FALSE ,
    CommandAborted FALSE => Abort FALSE ,
    CommandHalt FALSE => ComHalt FALSE ,
    Error FALSE => err FALSE ,
    ErrorID 0 => errid 0 );
```

LD：当 Exec 变为 True 时，执行五轴绝对直线插补指令；程序如下：



- **注意事项**

➢ 这个指令由“PMC_Ipolib”库实现。

➢ IpoCycle参数需要与运动指令所在的任务周期时间保持一致，否则指令执行时可能会出现Error报错。

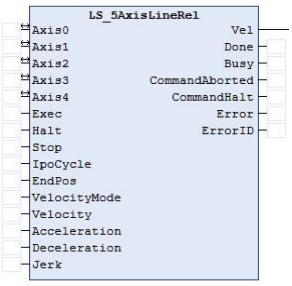
➢ 参数IpoCycle、Jerk、Velocity、Acceleration、Deceleration不可以设置为0。

- 在指令运行期间，五个轴都不能再被其它运动指令所调用。
- Endpos是五个插补轴的目标位置（绝对坐标）；Endpos[0]是插补0轴目标位置，Endpos[1]是插补1轴目标位置，Endpos[2]是插补2轴目标位置，Endpos[3]是插补3轴目标位置，Endpos[4]是插补4轴目标位置。
- Velocity、Acceleration、Deceleration、Jerk大小不能超过“轴设置-SoftMotion驱动：通用”里面的速度，加速度，减速度，加加速度的动态限制，否则指令执行时可能会出现Error报错。
- 五轴插补直线运动中前三个轴做空间直线插补运动，后两轴做跟随运动，和参与插补运动的轴同时启停。
- 根据五个轴需要移动的距离来按比例分配各轴速度，Vel是前三个插补轴的当前合速度，（需要移动的距离=目标位置-当前位置）。

7.10.9 LS_5AxisLineRel

指定的五个轴做直线插补运动，位置坐标为相对坐标。（前三轴做空间直线插补运动，后两轴做跟随运动）

- **指令格式**

指令	名称	FB/FC	LD表现	ST表现	库
LS_5AxisLineRel	五轴相对直线插补	FB		LS_5AxisLineRel(Axis0:=, Axis1:=, Axis2:=, Axis3:=, Axis4:=, Exec:=, Halt:=, Stop:=, IpoCycle:=, EndPos:=, VelocityMode:=, Velocity:=,	PMC_IpoLib

				Acceleration:=, Deceleration:=, Jerk:=, Vel=>, Done=>, Busy=>, CommandAborted=>, CommandHalt=>, Error=>, ErrorID=>);	

- **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	初始值	描述
Axis0	轴0	AXIS_REF_VIRTUAL_SM3	-	-	插补0轴
Axis1	轴1	AXIS_REF_VIRTUAL_SM3	-	-	插补1轴
Axis2	轴2	AXIS_REF_VIRTUAL_SM3	-	-	插补2轴
Axis3	轴3	AXIS_REF_VIRTUAL_SM3	-	-	跟随 3 轴
Axis4	轴4	AXIS_REF_VIRTUAL_SM3	-	-	跟随 4 轴

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Exec	启动	BOOL	TRUE-FALSE	FALSE	启动命令，上升沿有效。
Halt	暂停	BOOL	TRUE-FALSE	FALSE	暂停命令，当该输入量为 True 时，当前插补运动暂停；当状态再次切换到 False 时，插补模块继续之前的未完成的插补任务。
Stop	停止	BOOL	TRUE-FALSE	FALSE	停止命令，该命令为 True 时，插补模块停止。

IpoCycle	插补周期	DWORD	-	2000	插补周期，单位：us，必须和所在任务周期一致；
Endpos	目标位置	ARRAY[0..1]OFLREAL	负数，0，正数	0	轴的目标位置，单位：Pulse
Velocity Mode	速度模式	SMC_INT_VELMODE	0-3	SIGMOID	速度模式：梯形-0（TRAPEZOID）；S 形-1（SIGMOID）；四次方-3（QUADRATIC）
Velocity	速度	LREAL	0，正数	0	插补合成速度，Pulse/s
Acceleration	加速度	LREAL	0，正数	0	插补合成加速度，Pulse/s2
Deceleration	减速度	LREAL	0，正数	0	插补合成减速度，Pulse/s2
Jerk	加加速度	LREAL	负数，0，正数	90000000	加加速度，速度模式为 3 时需要设置此参数，该参数值不能为 0。

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
Vel	合速度	LREAL	0，正数	0	当前插补运动合成速度。
Done	完成	BOOL	TRUE-FALSE	FALSE	True 表示插补模块已完成插补任务。
Busy	运动中	BOOL	TRUE-FALSE	FALSE	True 表示插补运动正在进行之中。
Command Aborted	运动中断	BOOL	TRUE-FALSE	FALSE	当 Stop 有效时，该变量为 True，表示插补模块非正常中止。
Command Halt	运动暂停	BOOL	TRUE-FALSE	FALSE	当 Halt 为 True 时，该变量为 True，表示插补模块处于暂停状态。
Error	错误	BOOL	TRUE-FALSE	FALSE	错误状态；False-没有错误。

ErrorID	错误代码	DINT	0，正数	0	错误代码

	布尔	位串				整数								实数		时刻、持续时间 日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Axis0	AXIS_REF_VIRTUAL_SM3																			
Axis1	AXIS_REF_VIRTUAL_SM3																			
Axis2	AXIS_REF_VIRTUAL_SM3																			
Axis3	AXIS_REF_VIRTUAL_SM3																			
Axis4	AXIS_REF_VIRTUAL_SM3																			
Exec	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Halt	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Stop	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
IpoCycle	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Endpos	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
VelocityMode	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Velocity	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–
Acceleration	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–
Deceleration	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–
Jerk	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–
Vel	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–
Done	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Busy	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
CommandAborted	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
CommandHalt	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Error	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
ErrorID	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–

- **功能说明：**

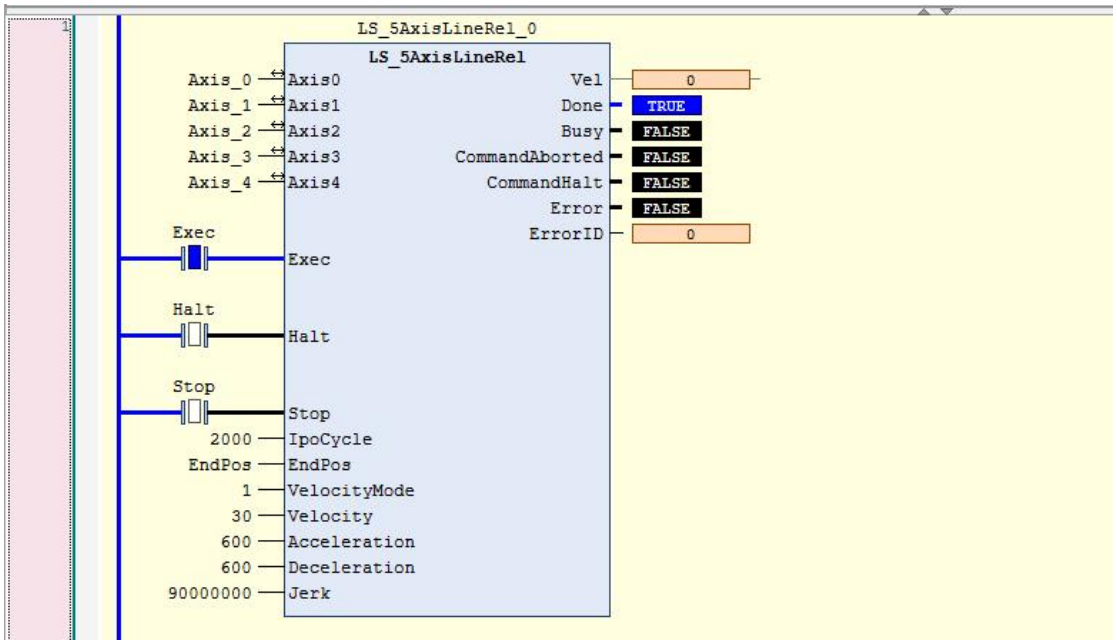
  - 指定的五个轴做直线插补运动，位置坐标为相对坐标。（前三轴做空间直线插补运动，后两轴做跟随运动）

- **程序示例**

ST：当 Exec 变为 True 时，执行五轴相对直线插补指令；程序如下：

```
LS_5AxisLineRel_0(
    Axis0:= Axis_0,
    Axis1:= Axis_1,
    Axis2:= Axis_2,
    Axis3:= Axis_3,
    Axis4:= Axis_4,
    Exec TRUE := Exec TRUE,
    Halt FALSE := Halt FALSE,
    Stop FALSE := Stop FALSE,
    IpoCycle 2000 := 2000,
    EndPos:= EndPos,
    VelocityMode SIGMOID := 1,
    Velocity 30 := 30,
    Acceleration 600 := 600,
    Deceleration 600 := 600,
    Jerk 9E+07 := jerk 9E+07 ,
    Vel 0 => vel 0 ,
    Done TRUE => done TRUE ,
    Busy FALSE => busy FALSE ,
    CommandAborted FALSE => Abort FALSE ,
    CommandHalt FALSE => ComHalt FALSE ,
    Error FALSE => err FALSE ,
    ErrorID 0 => errid 0 );
```

LD：当 Exec 变为 True 时，执行五轴相对直线插补指令；程序如下：



- **注意事项**

➢ 这个指令由“PMC_Ipolib”库实现。

➢ IpoCycle参数需要与运动指令所在的任务周期时间保持一致，否则指令执行时可能会出现Error报错。

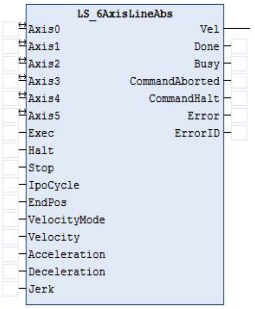
➢ 参数IpoCycle、Jerk、Velocity、Acceleration、Deceleration不可以设置为0。

- 在指令运行期间，五个轴都不能再被其它运动指令所调用。
- Endpos是五个插补轴需要移动的距离；Endpos[0]是插补0轴需要移动的距离，Endpos[1]是插补1轴需要移动的距离，Endpos[2]是插补2轴需要移动的距离，Endpos[3]是插补3轴需要移动的距离，Endpos[4]是插补4轴需要移动的距离。
- Velocity、Acceleration、Deceleration、Jerk大小不能超过“轴设置-SoftMotion驱动：通用”里面的速度，加速度，减速度，加加速度的动态限制，否则指令执行时可能会出现Error报错。
- 五轴插补直线运动中前三个轴做空间直线插补运动，后两轴做跟随运动，和参与插补运动的轴同时启停。
- 根据五个轴需要移动的距离来按比例分配各轴速度，Vel是前三个插补轴的当前合速度。
- 相对坐标简单理解：相对于起始位置的坐标，即将起始位置作为坐标原点建立的坐标系里的坐标。

7.10.10 LS_6AxisLineAbs

指定的六个轴做直线插补运动，位置坐标为绝对坐标。（前三轴做空间直线插补运动，后三轴做跟随运动）

- **指令格式**

指令	名称	FB/FC	LD表现	ST表现	库
LS_6AxisLineAbs	六轴绝对直线插补	FB		LS_6AxisLineAbs(Axis0:=, Axis1:=, Axis2:=, Axis3:=, Axis4:=, Axis5:=, Exec:=, Halt:=, Stop:=, IpoCycle:=,	PMC_IpoLib

				EndPos:=, VelocityMode:=, Velocity:=, Acceleration:=, Deceleration:=, Jerk:=, Vel=>, Done=>, Busy=>, CommandAborted=>, CommandHalt=>, Error=>, ErrorID=>);	

- **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	初始值	描述
Axis0	轴0	AXIS_REF_VIRTUAL_SM3	-	-	插补0轴
Axis1	轴1	AXIS_REF_VIRTUAL_SM3	-	-	插补1轴
Axis2	轴2	AXIS_REF_VIRTUAL_SM3	-	-	插补2轴
Axis3	轴3	AXIS_REF_VIRTUAL_SM3	-	-	跟随 3 轴
Axis4	轴4	AXIS_REF_VIRTUAL_SM3	-	-	跟随 4 轴
Axis5	轴5	AXIS_REF_VIRTUAL_SM3	-	-	跟随 5 轴

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Exec	启动	BOOL	TRUE-FALSE	FALSE	启动命令，上升沿有效。
Halt	暂停	BOOL	TRUE-FALSE	FALSE	暂停命令，当该输入量为 True 时，当前插补运动暂停；

					当状态再次切换到 False 时，插补模块继续之前的未完成的插补任务。
Stop	停止	BOOL	TRUE-FALSE	FALSE	停止命令，该命令为 True 时，插补模块停止。
IpoCycle	插补周期	DWORD	-	2000	插补周期，单位：us，必须和所在任务周期一致；
Endpos	目标位置	ARRAY[0..1]OFLREAL	负数，0 正数	0	轴的目标位置，单位：Pulse
VelocityMode	速度模式	SMC_INT_VELMODE	0-3	SIGMOID	速度模式：梯形-0（TRAPEZOID）；S 形-1（SIGMOID）；四次方-3（QUADRATIC）
Velocity	速度	LREAL	0，正数	0	插补合成速度，Pulse/s
Acceleration	加速度	LREAL	0，正数	0	插补合成加速度，Pulse/s2
Deceleration	减速度	LREAL	0，正数	0	插补合成减速度，Pulse/s2
Jerk	加加速度	LREAL	负数，0 正数	90000000	加加速度，速度模式为 3 时需要设置此参数，该参数值不能为 0。

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
Vel	合速度	LREAL	0，正数	0	当前插补运动合成速度。
Done	完成	BOOL	TRUE-FALSE	FALSE	True 表示插补模块已完成插补任务。
Busy	运动中	BOOL	TRUE-FALSE	FALSE	True 表示插补运动正在进行之中。
CommandAb	运动中断	BOOL	TRUE-FALSE	FALSE	当 Stop 有效时，该变量为

orted					True，表示插补模块非正常中止。
CommandHalt	运动暂停	BOOL	TRUE-FALSE	FALSE	当 Halt 为 True 时，该变量为 True，表示插补模块处于暂停状态。
Error	错误	BOOL	TRUE-FALSE	FALSE	错误状态；False-没有错误。
ErrorID	错误代码	DINT	0，正数	0	错误代码

	布尔	位串				整数								实数		时刻、持续时间 日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Axis0	AXIS_REF_VIRTUAL_SM3																			
Axis1	AXIS_REF_VIRTUAL_SM3																			
Axis2	AXIS_REF_VIRTUAL_SM3																			
Axis3	AXIS_REF_VIRTUAL_SM3																			
Axis4	AXIS_REF_VIRTUAL_SM3																			
Axis5	AXIS_REF_VIRTUAL_SM3																			
Exec	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Halt	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Stop	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
IpoCycle	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Endpos	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
VelocityMode	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Velocity	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–
Acceleration	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–
Deceleration	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–
Jerk	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–
Vel	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–
Done	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Busy	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
CommandAborted	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
CommandHalt	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Error	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
ErrorID	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–

- **功能说明：**

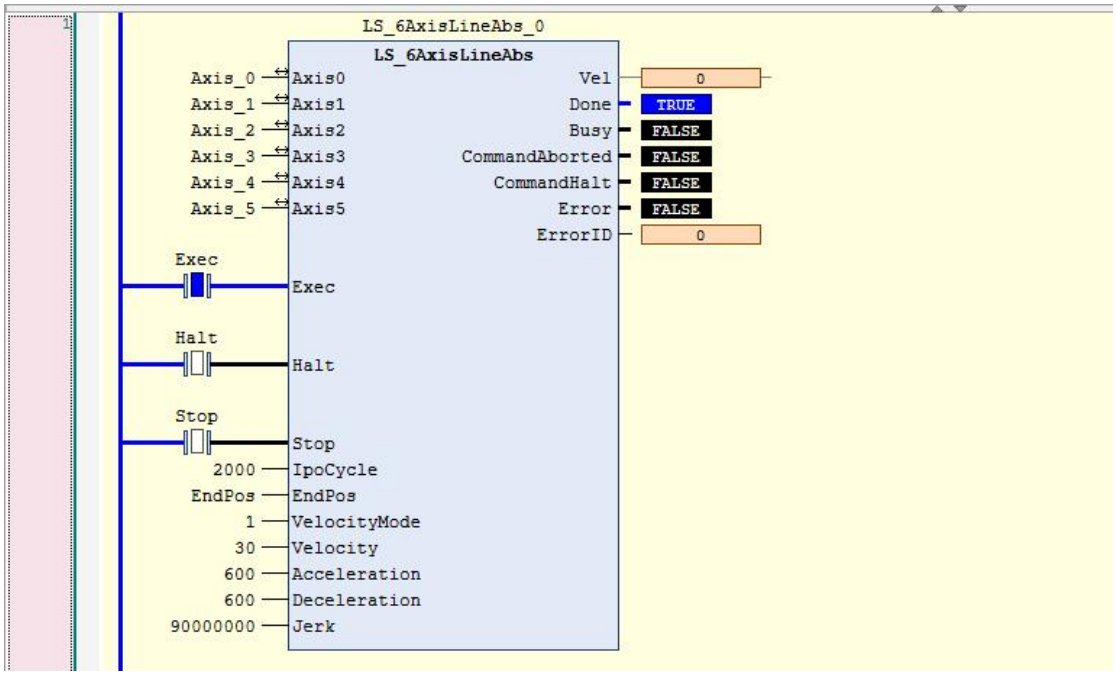
  - 指定的六个轴做直线插补运动，位置坐标为绝对坐标。（前三轴做空间直线插补运动，后三轴做跟随运动）

- **程序示例**

ST：当 Exec 变为 True 时，执行六轴绝对直线插补指令；程序如下：

```
LS_6AxisLineAbs_0(
    Axis0:= Axis_0,
    Axis1:= Axis_1,
    Axis2:= Axis_2,
    Axis3:= Axis_3,
    Axis4:= Axis_4,
    Axis5:= Axis_5,
    Exec TRUE := Exec TRUE,
    Halt FALSE := Halt FALSE,
    Stop FALSE := Stop FALSE,
    IpoCycle 2000 := 2000,
    EndPos:= EndPos,
    VelocityMode SIGMOID := 1,
    Velocity 30 := 30,
    Acceleration 600 := 600,
    Deceleration 600 := 600,
    Jerk 9E+07 := jerk 9E+07,
    Vel 0 => vel 0,
    Done TRUE => done TRUE,
    Busy FALSE => busy FALSE,
    CommandAborted FALSE => Abort FALSE,
    CommandHalt FALSE => ComHalt FALSE,
    Error FALSE => err FALSE,
    ErrorID 0 => errid 0);
```

LD：当 Exec 变为 True 时，执行六轴绝对直线插补指令；程序如下：



- **注意事项**

➢ 这个指令由“PMC_Ipolib”库实现。

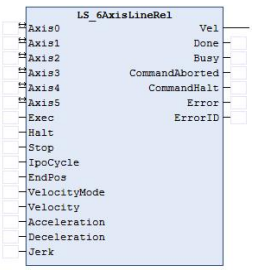
➢ IpoCycle参数需要与运动指令所在的任务周期时间保持一致，否则指令执行时可能会出现Error报错。

- 参数IpoCycle、Jerk、Velocity、Acceleration、Deceleration不可以设置为0。
- 在指令运行期间，六个轴都不能再被其它运动指令所调用。
- Endpos是六个插补轴的目标位置（绝对坐标）；Endpos[0]是插补0轴目标位置，Endpos[1]是插补1轴目标位置，Endpos[2]是插补2轴目标位置，Endpos[3]是插补3轴目标位置，Endpos[4]是插补4轴目标位置，Endpos[5]是插补5轴目标位置。
- Velocity、Acceleration、Deceleration、Jerk大小不能超过“轴设置-SoftMotion驱动：通用”里面的速度，加速度，减速度，加加速度的动态限制，否则指令执行时可能会出现Error报错。
- 六轴插补直线运动中前三个轴做空间直线插补运动，后三轴做跟随运动，和参与插补运动的轴同时启停。
- 根据六个轴需要移动的距离来按比例分配各轴速度，Vel是前三个插补轴的当前合速度，（需要移动的距离=目标位置-当前位置）。

7.10.11 LS_6AxisLineRel

指定的六个轴做直线插补运动，位置坐标为相对坐标。（前三轴做空间直线插补运动，后三轴做跟随运动）

- **指令格式**

指令	名称	FB/FC	LD表现	ST表现	库
LS_6AxisLineRel	六轴相对直线插补	FB		LS_6AxisLineRel(Axis0:=, Axis1:=, Axis2:=, Axis3:=, Axis4:=, Axis5:=, Exec:=, Halt:=, Stop:=, IpoCycle:=,	PMC_IpoLib

				EndPos:=, VelocityMode:=, Velocity:=, Acceleration:=, Deceleration:=, Jerk:=, Vel=>, Done=>, Busy=>, CommandAborted=>, CommandHalt=>, Error=>, ErrorID=>);	

- **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	初始值	描述
Axis0	轴0	AXIS_REF_VIRTUAL_SM3	-	-	插补0轴
Axis1	轴1	AXIS_REF_VIRTUAL_SM3	-	-	插补1轴
Axis2	轴2	AXIS_REF_VIRTUAL_SM3	-	-	插补2轴
Axis3	轴3	AXIS_REF_VIRTUAL_SM3	-	-	跟随 3 轴
Axis4	轴4	AXIS_REF_VIRTUAL_SM3	-	-	跟随 4 轴
Axis5	轴5	AXIS_REF_VIRTUAL_SM3	-	-	跟随 5 轴

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Exec	启动	BOOL	TRUE-FALSE	FALSE	启动命令，上升沿有效。
Halt	暂停	BOOL	TRUE-FALSE	FALSE	暂停命令，当该输入量为 True 时，当前插补运动暂停；当状

					态再次切换到 False 时，插补模块继续之前的未完成的插补任务。
Stop	停止	BOOL	TRUE-FALSE	FALSE	停止命令，该命令为 True 时，插补模块停止。
IpoCycle	插补周期	DWORD	-	2000	插补周期，单位：us，必须和所在任务周期一致；
Endpos	目标位置	ARRAY[0..1]OFLREAL	负数，0 正数	0	轴的目标位置，单位：Pulse
Velocity Mode	速度模式	SMC_INT_VELMODE	0-3	SIGMOID	速度模式：梯形-0（TRAPEZOID）；S 形-1（SIGMOID）；四次方-3（QUADRATIC）
Velocity	速度	LREAL	0，正数	0	插补合成速度，Pulse/s
Acceleration	加速度	LREAL	0，正数	0	插补合成加速度，Pulse/s2
Deceleration	减速度	LREAL	0，正数	0	插补合成减速度，Pulse/s2
Jerk	加加速度	LREAL	负数，0 正数	90000000	加加速度，速度模式为 3 时需要设置此参数，该参数值不能为 0。

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
Vel	合速度	LREAL	0，正数	0	当前插补运动合成速度。
Done	完成	BOOL	TRUE-FALSE	FALSE	True 表示插补模块已完成插补任务。
Busy	运动中	BOOL	TRUE-FALSE	FALSE	True 表示插补运动正在进行之中。
CommandAborted	运动中断	BOOL	TRUE-FALSE	FALSE	当 Stop 有效时，该变量为 True，表示插补模块非正常

					中止。
CommandHalt	运动暂停	BOOL	TRUE-FALSE	FALSE	当 Halt 为 True 时，该变量为 True，表示插补模块处于暂停状态。
Error	错误	BOOL	TRUE-FALSE	FALSE	错误状态；False-没有错误。
ErrorID	错误代码	DINT	0，正数	0	错误代码

	布尔	位串				整数								实数		时刻、持续时间 日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Axis0	AXIS_REF_VIRTUAL_SM3																			
Axis1	AXIS_REF_VIRTUAL_SM3																			
Axis2	AXIS_REF_VIRTUAL_SM3																			
Axis3	AXIS_REF_VIRTUAL_SM3																			
Axis4	AXIS_REF_VIRTUAL_SM3																			
Axis5	AXIS_REF_VIRTUAL_SM3																			
Exec	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Halt	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Stop	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
IpoCycle	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Endpos	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Velocity Mode	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Velocity	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–
Acceleration	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–
Deceleration	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–
Jerk	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–
Vel	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–
Done	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Busy	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
CommandAborted	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

CommandHalt	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Error	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
ErrorID	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–

- **功能说明：**

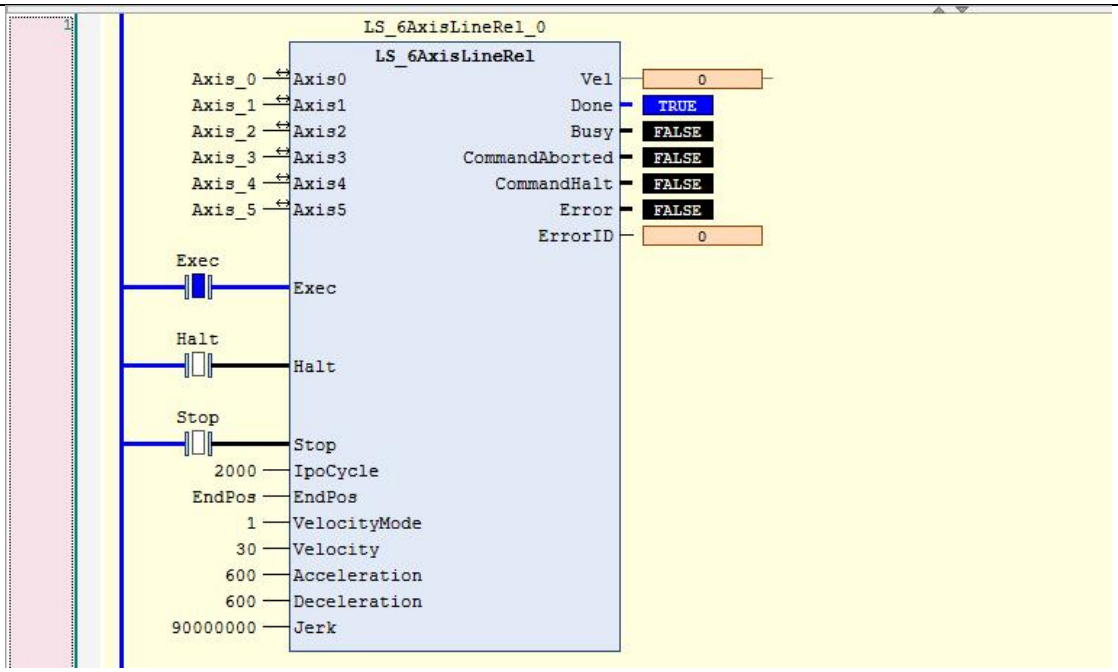
  - 指定的六个轴做直线插补运动，位置坐标为相对坐标。（前三轴做空间直线插补运动，后三轴做跟随运动）

- **程序示例**

ST：当 Exec 变为 True 时，执行六轴相对直线插补指令；程序如下：

```
LS_6AxisLineRel_0(
    Axis0:= Axis_0,
    Axis1:= Axis_1,
    Axis2:= Axis_2,
    Axis3:= Axis_3,
    Axis4:= Axis_4,
    Axis5:= Axis_5,
    Exec TRUE := Exec TRUE,
    Halt FALSE := Halt FALSE,
    Stop FALSE := Stop FALSE,
    IpoCycle 2000 := 2000,
    EndPos:= EndPos,
    VelocityMode SIGMOID := 1,
    Velocity 30 := 30,
    Acceleration 600 := 600,
    Deceleration 600 := 600,
    Jerk 9E+07 := jerk 9E+07,
    Vel 0 => vel 0,
    Done TRUE => done TRUE,
    Busy FALSE => busy FALSE,
    CommandAborted FALSE => Abort FALSE,
    CommandHalt FALSE => ComHalt FALSE,
    Error FALSE => err FALSE,
    ErrorID 0 => errid 0);
```

LD：当 Exec 变为 True 时，执行六轴相对直线插补指令；程序如下：



- **注意事项**

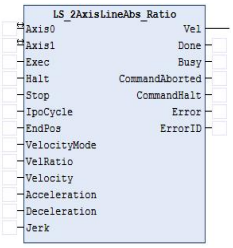
- 这个指令由“PMC_Ipolib”库实现。
- IpoCycle参数需要与运动指令所在的任务周期时间保持一致，否则指令执行时可能会出现Error报错。
- 参数IpoCycle、Jerk、Velocity、Acceleration、Deceleration不可以设置为0。
- 在指令运行期间，六个轴都不能再被其它运动指令所调用。
- Endpos是六个插补轴需要移动的距离；Endpos[0]是插补0轴需要移动的距离，Endpos[1]是插补1轴需要移动的距离，Endpos[2]是插补2轴需要移动的距离，Endpos[3]是插补3轴需要移动的距离，Endpos[4]是插补4轴需要移动的距离，Endpos[5]是插补5轴需要移动的距离。
- Velocity、Acceleration、Deceleration、Jerk大小不能超过“轴设置-SoftMotion驱动：通用”里面的速度，加速度，减速度，加加速度的动态限制，否则指令执行时可能会出现Error报错。
- 六轴插补直线运动中前三个轴做空间直线插补运动，后三轴做跟随运动，和参与插补运动的轴同时启停。

- 根据六个轴需要移动的距离来按比例分配各轴速度，Vel是前三个插补轴的当前合速度。
- 相对坐标简单理解：相对于起始位置的坐标，即将起始位置作为坐标原点建立的坐标系里的坐标。

7.10.12 LS_2AxisLineAbs_Ratio

指定的两个轴做可调速直线插补运动，位置坐标为绝对坐标。

- **指令格式**

指令	名称	FB/FC	LD表现	ST表现	库
LS_2AxisLineAbs_Ratio	两轴可调速绝对直线插补	FB		LS_2AxisLineAbs_Ratio(Axis0:=, Axis1:=, Exec:=, Halt:=, Stop:=, IpoCycle:=, EndPos:=, VelocityMode:=, VelRatio:=, Velocity:=, Acceleration:=, Deceleration:=, Jerk:=, Vel=>, Done=>, Busy=>, CommandAborted=>, CommandHalt=>, Error=>, ErrorID=>);	PMC_IpoLib

- **相关变量**

输入输出变量

输入输	名称	数据类型	有效	初始	描述

出变量			范围	值	
Axis0	轴0	AXIS_REF_VIRTUAL_SM3	-	-	插补0轴
Axis1	轴1	AXIS_REF_VIRTUAL_SM3	-	-	插补1轴

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Exec	启动	BOOL	TRUE-FALSE	FALSE	启动命令，上升沿有效；
Halt	暂停	BOOL	TRUE-FALSE	FALSE	暂停命令，当该输入量为 True 时，当前插补运动暂停；当状态再次切换到 False 时，插补模块继续之前的未完成的插补任务。
Stop	停止	BOOL	TRUE-FALSE	FALSE	停止命令，该命令为 True 时，插补模块停止。
IpoCycle	插补周期	DWORD	-	2000	插补周期，单位：us，必须和所在任务周期一致；
Endpos	目标位置	ARRAY[0..1]OFLREAL	负数，0 正数	0	轴的目标位置，单位：Pulse
Velocity Mode	速度模式	SMC_INT_VELMODE	0-3	SIGMOID	速度模式：梯形-0（TRAPEZOID）；S 形-1（SIGMOID）；四次方-3（QUADRATIC）
VelRatio	速度倍率	LREAL	0.01-2	1	速度倍率，最小值 0.01，最大值 2；
Velocity	速度	LREAL	0，正数	0	插补合成速度，Pulse/s
Acceleration	加速度	LREAL	0，正数	0	插补合成加速度，Pulse/s2

Deceleration	减速度	LREAL	0，正数	0	插补合成减速度，Pulse/s2
Jerk	加加速度	LREAL	负数，0 正数	90000000	加加速度，速度模式为 3 时需要设置此参数，该参数值不能为 0。

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
Vel	合速度	LREAL	0，正数	0	当前插补运动合成速度。
Done	完成	BOOL	TRUE-FALSE	FALSE	True 表示插补模块已完成插补任务。
Busy	运动中	BOOL	TRUE-FALSE	FALSE	True 表示插补运动正在进行之中。
CommandAborted	运动中断	BOOL	TRUE-FALSE	FALSE	当 Stop 有效时，该变量为 True，表示插补模块非正常中止。
CommandHalt	运动暂停	BOOL	TRUE-FALSE	FALSE	当 Halt 为 True 时，该变量为 True，表示插补模块处于暂停状态。
Error	错误	BOOL	TRUE-FALSE	FALSE	错误状态；False-没有错误。
ErrorID	错误代码	DINT	0，正数	0	错误代码

	布尔	位串				整数								实数		时刻、持续时间 日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Axis0	AXIS_REF_VIRTUAL_SM3																			
Axis1	AXIS_REF_VIRTUAL_SM3																			
Exec	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Halt	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Stop	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
IpoCycle	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

Endpos	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
VelocityMode	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
VelRatio	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–
Velocity	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–
Acceleration	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–
Deceleration	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–
Jerk	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–
Vel	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–
Done	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Busy	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
CommandAborted	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
CommandHalt	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Error	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
ErrorID	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–

- **功能说明：**

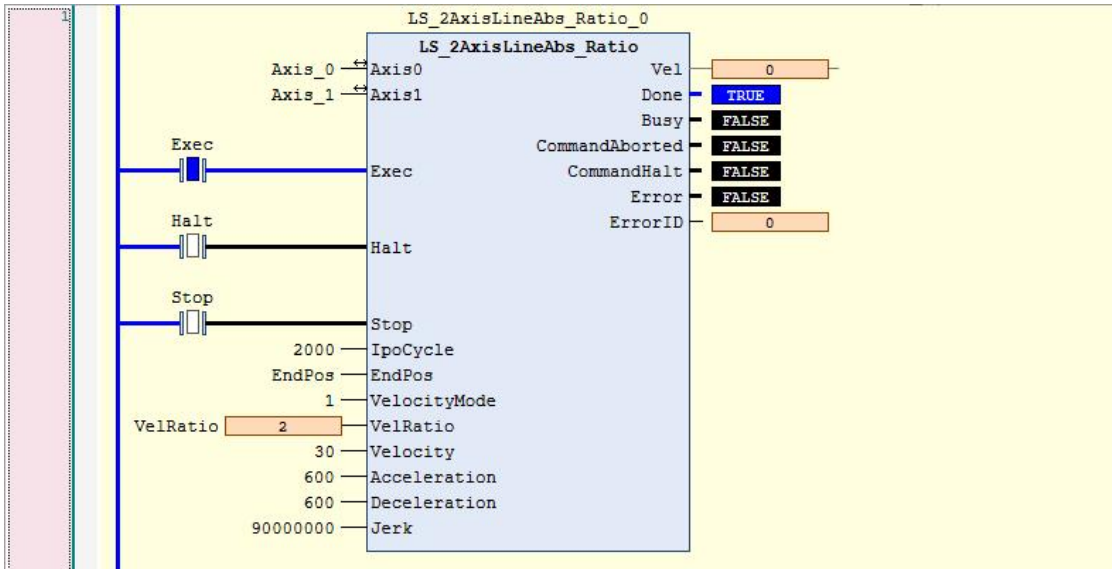
  - 指定的两个轴做可调速直线插补运动，位置坐标为绝对坐标；Vel（合速度）=Velocity（速度）*VelRatio（速度倍率），在插补过程中通过改变速度倍率（0.01-2）来改变插补运动的合速度。

- **程序示例**

ST：当Exec变为True时，执行两轴可调速绝对直线插补指令；程序如下：

```
26  LS_2AxisLineAbs_Ratio_0(
27      Axis0:= Axis_0,
28      Axis1:= Axis_1,
29      Exec TRUE := Exec TRUE,
30      Halt FALSE := Halt FALSE,
31      Stop FALSE := Stop FALSE,  IpoCycle 2000 := 2000,
32      EndPos:= EndPos,
33      VelocityMode SIGMOID := 1,
34      VelRatio 2 := VelRatio 2,
35      Velocity 30 := 30,
36      Acceleration 600 := 600,
37      Deceleration 600 := 600,
38      Jerk 9E+07 := jerk 9E+07 ,
39      Vel 0 => vel 0 ,
40      Done TRUE => done TRUE ,
41      Busy FALSE => busy FALSE ,
42      CommandAborted FALSE => Abort FALSE ,
43      CommandHalt FALSE => ComHalt FALSE ,
44      Error FALSE => err FALSE ,
45      ErrorID 0 => errid 0 );
46
```

LD：当Exec变为True时，执行两轴可调速绝对直线插补指令；程序如下：



- **注意事项**

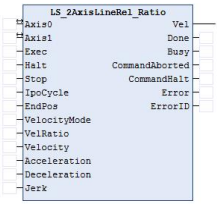
- 这个指令由“PMC_Ipolib”库实现。
- IpoCycle参数需要与运动指令所在的任务周期时间保持一致，否则指令执行时可能会出现Error报错。
- 参数IpoCycle、Jerk、Velocity、Acceleration、Deceleration不可以设置为0。
- 在指令运行期间，两个轴都不能再被其它运动指令所调用。
- Endpos是两个插补轴的目标位置（绝对坐标）；Endpos[0]是插补0轴目标位置，Endpos[1]是插补1轴目标位置。
- Velocity、Acceleration、Deceleration、Jerk大小不能超过“轴设置-SoftMotion驱动：通用”里面的速度，加速度，减速度，加加速度的动态限制，否则指令执行时可能会出现Error报错。
- 与LS_2AxisLineAbs相比，增加了可调速功能，即在插补过程中可通过改变VelRatio的值来在线改变插补速度;VelRatio最小值为0.01，倍率小于0.01的按0.01算；最大值为2，倍率大于2的按2算。
- 本指令根据两个插补轴需要移动的距离来分配插补轴速度，使得两插补轴的合速度为

Velocity（30Pulse/s），（需要移动的距离=目标位置-当前位置）。

7.10.13 LS_2AxisLineRel_Ratio

指定的两个轴做可调速直线插补运动，位置坐标为绝对坐标。

- **指令格式**

指令	名称	FB/FC	LD表现	ST表现	库
LS_2AxisLineRel_Ratio	两轴可调速相对直线插补	FB		LS_2AxisLineRel_Ratio (Axis0:=, Axis1:=, Exec:=, Halt:=, Stop:=, IpoCycle:=, EndPos:=, VelocityMode:=, VelRatio:=, Velocity:=, Acceleration:=, Deceleration:=, Jerk:=, Vel=>, Done=>, Busy=>, CommandAborted=>, CommandHalt=>, Error=>, ErrorID=>);	PMC_IpoLib

- **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	初始值	描述
Axis0	轴0	AXIS_REF_VIRTUAL_SM3	-	-	插补0轴

Axis1	轴1	AXIS_REF_VIRTUAL_SM3	-	-	插补1轴

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Exec	启动	BOOL	TRUE-FALSE	FALSE	启动命令，上升沿有效；
Halt	暂停	BOOL	TRUE-FALSE	FALSE	暂停命令，当该输入量为 True 时，当前插补运动暂停；当状态再次切换到 False 时，插补模块继续之前的未完成的插补任务。
Stop	停止	BOOL	TRUE-FALSE	FALSE	停止命令，该命令为 True 时，插补模块停止。
IpoCycle	插补周期	DWORD	-	2000	插补周期，单位：us，必须和所在任务周期一致；
Endpos	目标位置	ARRAY[0..1]OFLREAL	负数，0 正数	0	轴的目标位置，单位：Pulse
Velocity Mode	速度模式	SMC_INT_VELMODE	0-3	SIGMOID	速度模式：梯形-0（TRAPEZOID）；S 形-1（SIGMOID）；四次方-3（QUADRATIC）
VelRatio	速度倍率	LREAL	0.01-2	1	速度倍率，最小值 0.01，最大值 2；
Velocity	速度	LREAL	0，正数	0	插补合成速度，Pulse/s
Acceleration	加速度	LREAL	0，正数	0	插补合成加速度，Pulse/s2
Deceleration	减速度	LREAL	0，正数	0	插补合成减速度，Pulse/s2

Jerk	加加速度	LREAL	负数，0 正数	90000000	加加速度，速度模式为 3 时需要设置此参数，该参数值不能为 0。

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
Vel	合速度	LREAL	0，正数	0	当前插补运动合成速度。
Done	完成	BOOL	TRUE-FALSE	FALSE	True 表示插补模块已完成插补任务。
Busy	运动中	BOOL	TRUE-FALSE	FALSE	True 表示插补运动正在进行之中。
Command Aborted	运动中断	BOOL	TRUE-FALSE	FALSE	当 Stop 有效时，该变量为 True，表示插补模块非正常中止。
Command Halt	运动暂停	BOOL	TRUE-FALSE	FALSE	当 Halt 为 True 时，该变量为 True，表示插补模块处于暂停状态。
Error	错误	BOOL	TRUE-FALSE	FALSE	错误状态；False-没有错误。
ErrorID	错误代码	DINT	0，正数	0	错误代码

	布尔	位串				整数								实数		时刻、持续时间 日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Axis0	AXIS_REF_VIRTUAL_SM3																			
Axis1	AXIS_REF_VIRTUAL_SM3																			
Exec	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Halt	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Stop	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
IpoCycle	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Endpos	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
VelocityMode	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_
VelRatio	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–

Velocity	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–
Acceleration	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–
Deceleration	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–
Jerk	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–
Vel	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–
Done	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Busy	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
CommandAborted	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
CommandHalt	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Error	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
ErrorID	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–

- **功能说明：**

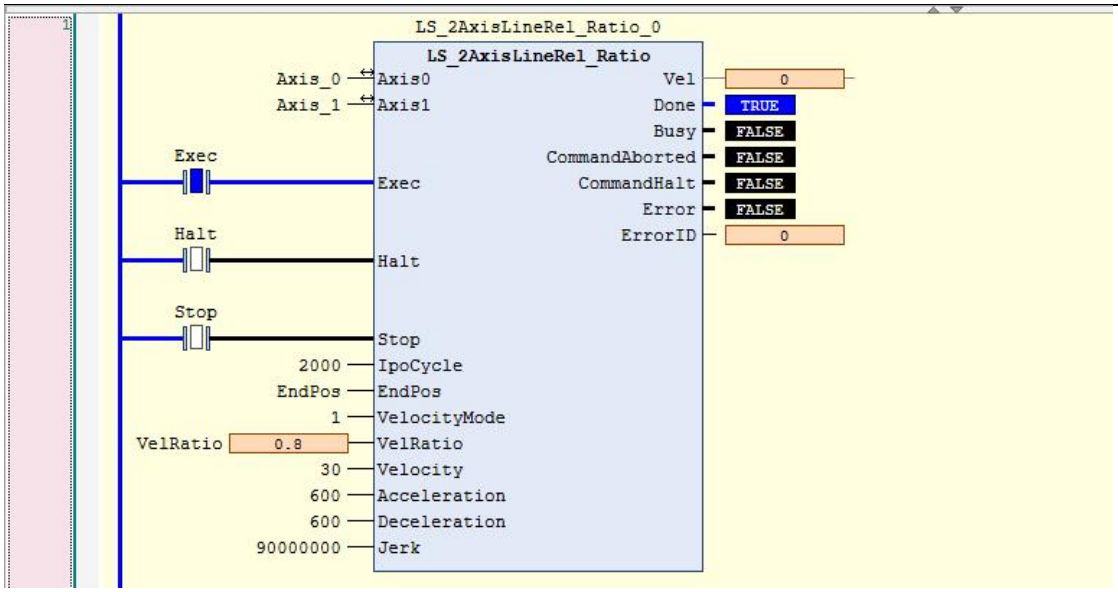
  - 指定的两个轴做可调速直线插补运动，位置坐标为相对坐标；Vel（合速度）=Velocity（速度）*VelRatio（速度倍率），在插补过程中通过改变速度倍率（0.01-2）来改变插补运动的合速度。

- **程序示例**

ST：当 Exec 变为 True 时，执行两轴可调速相对直线插补指令；程序如下：

```
LS_2AxisLineRel_Ratio_0(
    Axis0:= Axis_0,
    Axis1:= Axis_1,
    Exec TRUE := Exec TRUE,
    Halt FALSE := Halt FALSE,
    Stop FALSE := Stop FALSE,  IpoCycle 2000 := 2000,
    EndPos:= EndPos,
    VelocityMode SIGMOID := 1,
    VelRatio 1.52 := VelRatio 1.52 ,
    Velocity 30 := 30,
    Acceleration 600 := 600,
    Deceleration 600 := 600,
    Jerk 9E+07 := jerk 9E+07 ,
    Vel 0 => vel 0 ,
    Done TRUE => done TRUE ,
    Busy FALSE => busy FALSE ,
    CommandAborted FALSE => Abort FALSE ,
    CommandHalt FALSE => ComHalt FALSE ,
    Error FALSE => err FALSE ,
    ErrorID 0 => errid 0 );
```

LD：当 Exec 变为 True 时，执行两轴可调速相对直线插补指令；程序如下：



- **注意事项**

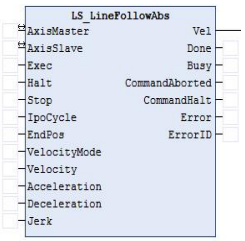
- 这个指令由“PMC_Ipolib”库实现。
- IpoCycle参数需要与运动指令所在的任务周期时间保持一致，否则指令执行时可能会出现Error报错。
- 参数IpoCycle、Jerk、Velocity、Acceleration、Deceleration不可以设置为0。
- 在指令运行期间，两个轴都不能再被其它运动指令所调用。
- Endpos是两个插补轴需要移动的距离；Endpos[0]是插补0轴需要移动的距离，Endpos[1]是插补1轴需要移动的距离。
- Velocity、Acceleration、Deceleration、Jerk大小不能超过“轴设置-SoftMotion驱动：通用”里面的速度，加速度，减速度，加加速度的动态限制，否则指令执行时可能会出现Error报错。
- 与LS_2AxisLineRel相比，增加了可调速功能，即在插补过程中可通过改变VelRatio的值来在线改变插补速度;VelRatio最小值为0.01，倍率小于0.01的按0.01算；最大值为2，倍率大于2的按2算。
- 本指令根据两个插补轴需要移动的距离来分配插补轴速度，使得两插补轴的合速度为Velocity（30Pulse/s）。

➢ 相对坐标简单理解：相对于起始位置的坐标，即将起始位置作为坐标原点建立的坐标系里的坐标。

7.10.14 LS_LineFollowAbs

主轴做点位运动，从轴跟随主轴做绝对位置模式的跟随运动。

- **指令格式**

指令	名称	FB/FC	LD表现	ST表现	库
LS_LineFollowAbs	绝对跟随	FB		LS_LineFollowAbs(AxisMaster:=, AxisSlave:=, Exec:=, Halt:=, Stop:=, IpoCycle:=, EndPos:=, VelocityMode:=, Velocity:=, Acceleration:=, Deceleration:=, Jerk:=, Vel=>, Done=>, Busy=>, CommandAborted=>, CommandHalt=>, Error=>, ErrorID=>);	PMC_IpoLib

- **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	初始值	描述
AxisMaster	主轴	AXIS_REF_VIRTUAL_SM3	-	-	主轴

AxisSlave	从轴	AXIS_REF_VIRTUAL_SM3	-	-	跟随轴

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Exec	启动	BOOL	TRUE-FALSE	FALSE	启动命令，上升沿有效；
Halt	暂停	BOOL	TRUE-FALSE	FALSE	暂停命令，当该输入量为 True 时，当前跟随运动暂停；当状态再次切换到 False 时，跟随模块继续之前的未完成的跟随运动。
Stop	停止	BOOL	TRUE-FALSE	FALSE	停止命令，该命令为 True 时，跟随模块停止。
IpoCycle	周期	DWORD	-	2000	周期，单位：us，必须和所在任务周期一致；
Endpos	目标位置	ARRAY[0..1]OFLREAL	负数，0 正数	0	轴的目标位置，单位：Pulse
VelocityMode	速度模式	SMC_INT_VELMODE	0-3	SIGMOID	速度模式：梯形-0（TRAPEZOID）；S 形-1（SIGMOID）；四次方-3（QUADRATIC）
Velocity	速度	LREAL	0，正数	10	主从轴最大速度，Pulse/s
Acceleration	加速度	LREAL	0，正数	10	主从轴平均加速度，Pulse/s^2
Deceleration	减速度	LREAL	0，正数	10	主从轴平均减速度，Pulse/s^2

Jerk	加加速度	LREAL	负数，0 正数	90000000	加加速度，速度模式为 3 时需要设置此参数，该参数值不能为 0。

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
Vel	速度值	LREAL	0，正数	0	主轴当前速度
Done	完成	BOOL	TRUE-FALSE	FALSE	True 表示运动完成
Busy	运动中	BOOL	TRUE-FALSE	FALSE	True 表示运动正在进行之中。
CommandAborted	运动中断	BOOL	TRUE-FALSE	FALSE	当 Stop 有效时，该变量为 True，表示跟随模块非正常中止。
CommandHalt	运动暂停	BOOL	TRUE-FALSE	FALSE	当 Halt 为 True 时，该变量为 True，表示跟随模块处于暂停状态。
Error	错误	BOOL	TRUE-FALSE	FALSE	错误状态；False-没有错误；
ErrorID	错误代码	DINT	0，正数	0	错误代码

	布尔	位串				整数								实数		时刻、持续时间 日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
AxisMaster	AXIS_REF_VIRTUAL_SM3																			
AxisSlave	AXIS_REF_VIRTUAL_SM3																			
Exec	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Halt	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Stop	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
IpoCycle	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Endpos	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
VelocityMode	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_
Velocity	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–
Acceleration	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–

Deceleration	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–
Jerk	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–
Vel	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–
Done	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Busy	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
CommandAborted	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
CommandHalt	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Error	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
ErrorID	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–

- **功能说明：**

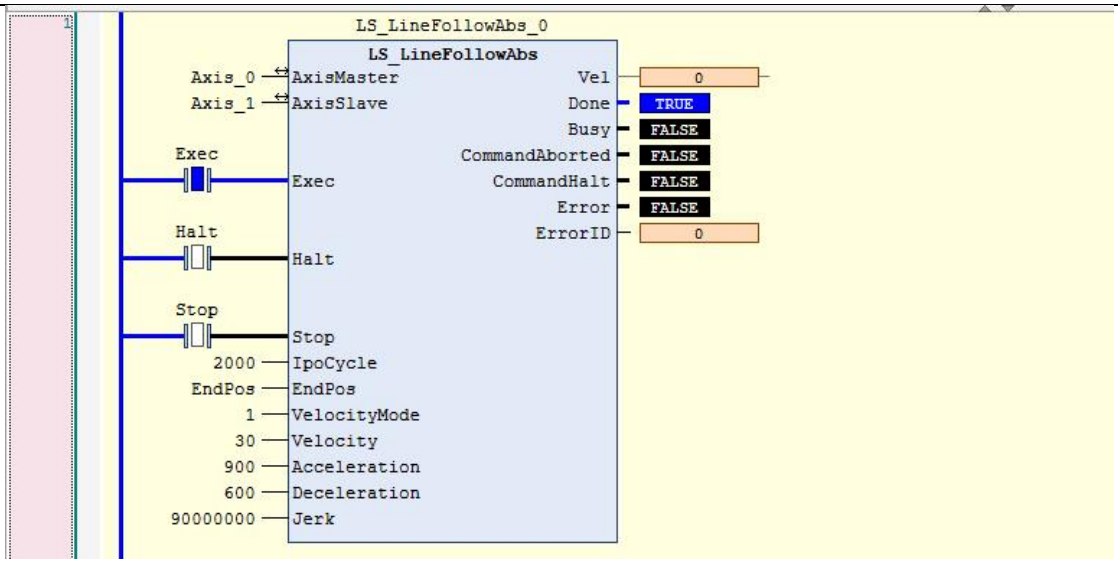
  - 主轴做点位运动，从轴跟随主轴做绝对位置模式的跟随运动；从轴和主轴同时启停，速度曲线类似，根据从轴与主轴的移动距离来分配主从轴的速度。

- **程序示例**

ST：当 Exec 变为 True 时，执行两轴绝对坐标跟随运动；程序如下：

```
LS_LineFollowAbs_0(
    AxisMaster:= Axis_0,
    AxisSlave:= Axis_1,
    Exec TRUE := Exec TRUE,
    Halt FALSE := Halt FALSE,
    Stop FALSE := Stop FALSE,
    IpoCycle 2000 := 2000,
    EndPos:= EndPos,
    VelocityMode SIGMOID := 1,
    Velocity 30 := 30,
    Acceleration 600 := 600,
    Deceleration 600 := 600,
    Jerk 9E+07 := jerk 9E+07 ,
    Vel 0 => vel 0 ,
    Done TRUE => done TRUE ,
    Busy FALSE => busy FALSE ,
    CommandAborted FALSE => Abort FALSE ,
    CommandHalt FALSE => ComHalt FALSE ,
    Error FALSE => err FALSE ,
    ErrorID 0 => errid 0 );
```

LD：当 Exec 变为 True 时，执行两轴绝对坐标跟随运动；程序如下：



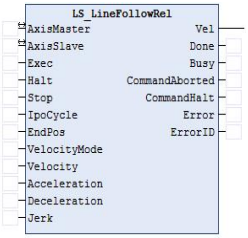
- **注意事项**

- 这个指令由“PMC_Ipolib”库实现。
- IpoCycle参数需要与运动指令所在的任务周期时间保持一致，否则指令执行时可能会出现Error报错。
- 参数IpoCycle、Jerk、Velocity、Acceleration、Deceleration不可以设置为0，否则指令执行时可能会出现Error报错。
- 在指令运行期间，主从轴都不能再被其它运动指令所调用。
- Endpos是主从轴的目标位置（绝对坐标）；Endpos[0]是主轴目标位置，Endpos[1]是从轴目标位置。
- Velocity、Acceleration、Deceleration、Jerk大小不能超过“轴设置-SoftMotion驱动：通用”里面的速度，加速度，减速度，加加速度的动态限制，否则指令执行时可能会出现Error报错。
- 跟随运动指令仅在指令启动到结束这段时间有效，对主从轴有跟随控制效果，到达指令位置后，主从轴不再同步，都恢复standstill状态。
- 本指令根据主从轴需要移动的距离来分配主从轴速度，移动距离大的轴其速度为Velocity（30Pulse/s），另一轴的速度按移动距离比例分配，（移动距离=目标位置-当前位置）。

7.10.15 LS_LineFollowRel

主轴做点位运动，从轴跟随主轴做绝对位置模式的跟随运动。

- **指令格式**

指令	名称	FB/FC	LD表现	ST表现	库
LS_LineFollowRel	相对跟随	FB		LS_LineFollowRel(AxisMaster:=, AxisSlave:=, Exec:=, Halt:=, Stop:=, IpoCycle:=, EndPos:=, VelocityMode:=, Velocity:=, Acceleration:=, Deceleration:=, Jerk:=, Vel=>, Done=>, Busy=>, CommandAborted=>, CommandHalt=>, Error=>, ErrorID=>);	PMC_IpoLib

- **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	初始值	描述
AxisMaster	主轴	AXIS_REF_VIRTUAL_SM3	-	-	主轴
AxisSlave	从轴	AXIS_REF_VIRTUAL_SM3	-	-	跟随轴

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Exec	启动	BOOL	TRUE-FALSE	FALSE	启动命令，上升沿有效；
Halt	暂停	BOOL	TRUE-FALSE	FALSE	暂停命令，当该输入量为 True 时，当前跟随运动暂停；当状态再次切换到 False 时，跟随模块继续之前的未完成的跟随运动。
Stop	停止	BOOL	TRUE-FALSE	FALSE	停止命令，该命令为 True 时，跟随模块停止。
IpoCycle	周期	DWORD	-	2000	周期，单位：us，必须和所在任务周期一致；
Endpos	目标位置	ARRAY[0..1]OFLREAL	负数，0 正数	0	轴需要移动的距离，单位：Pulse
VelocityMode	速度模式	SMC_INT_VELMODE	0-3	SIGMOID	速度模式：梯形-0（TRAPEZOID）；S 形-1（SIGMOID）；四次方-3（QUADRATIC）
Velocity	速度	LREAL	0，正数	0	主从轴最大速度，Pulse/s
Acceleration	加速度	LREAL	0，正数	0	主从轴平均加速度，Pulse/s^2
Deceleration	减速度	LREAL	0，正数	0	主从轴平均减速度，Pulse/s^2
Jerk	加加速度	LREAL	负数，0 正数	90000000	加加速度，速度模式为 3 时需要设置此参数，该参数值不能为 0。

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
Vel	合速度	LREAL	0，正数	0	主轴当前速度
Done	完成	BOOL	TRUE-FALSE	FALSE	True 表示运动完成
Busy	运动中	BOOL	TRUE-FALSE	FALSE	True 表示运动正在进行之中。
Command Aborted	运动中断	BOOL	TRUE-FALSE	FALSE	当 Stop 有效时，该变量为 True，表示跟随模块非正常中止。
Command Halt	运动暂停	BOOL	TRUE-FALSE	FALSE	当 Halt 为 True 时，该变量为 True，表示跟随模块处于暂停状态。
Error	错误	BOOL	TRUE-FALSE	FALSE	错误状态；False-没有错误；
ErrorID	错误代码	DINT	0，正数	0	错误代码

	布尔	位串				整数								实数		时刻、持续时间 日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
AxisMaster	AXIS_REF_VIRTUAL_SM3																			
AxisSlave	AXIS_REF_VIRTUAL_SM3																			
Exec	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Halt	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Stop	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
IpoCycle	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Endpos	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
VelocityMode	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_
Velocity	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–
Acceleration	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–
Deceleration	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–
Jerk	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

Vel	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–
Done	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Busy	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
CommandAborted	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
CommandHalt	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Error	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
ErrorID	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–

- **功能说明：**

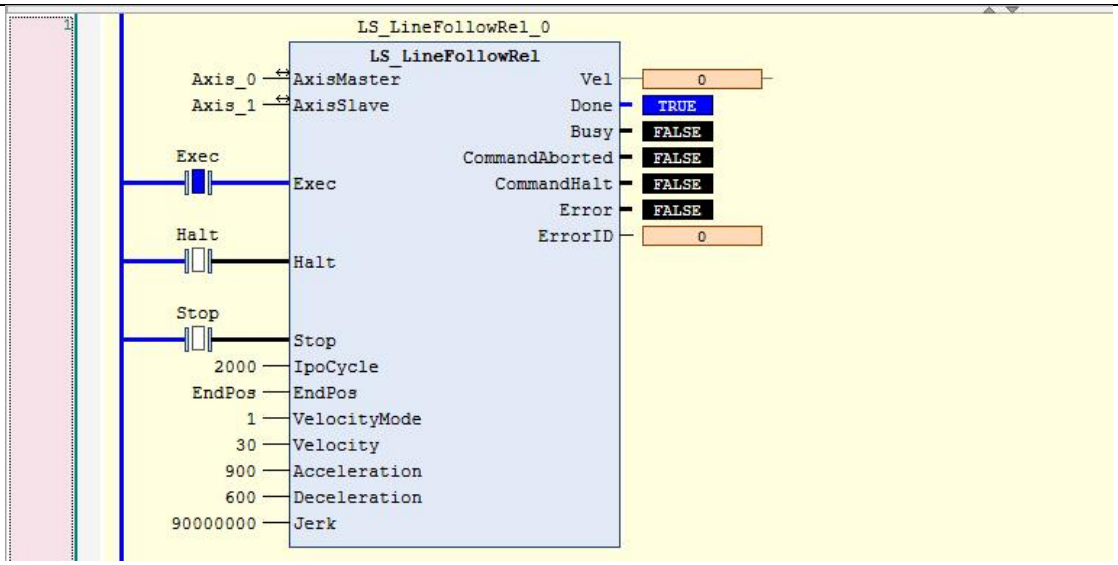
  - 主轴做点位运动，从轴跟随主轴做相对位置模式的跟随运动；从轴和主轴同时启停，速度曲线类似，根据从轴与主轴的移动距离来分配主从轴的速度。

- **程序示例**

ST：当 Exec 变为 True 时，执行两轴相对坐标跟随运动；程序如下：

```
LS_LineFollowRel_0(
    AxisMaster:= Axis_0,
    AxisSlave:= Axis_1,
    Exec TRUE := Exec TRUE,
    Halt FALSE := Halt FALSE,
    Stop FALSE := Stop FALSE,
    IpoCycle 2000 := 2000,
    EndPos:= EndPos,
    VelocityMode SIGMOID := 1,
    Velocity 30 := 30,
    Acceleration 600 := 600,
    Deceleration 600 := 600,
    Jerk 9E+07 := jerk 9E+07 ,
    Vel 0 => vel 0 ,
    Done TRUE => done TRUE ,
    Busy FALSE => busy FALSE ,
    CommandAborted FALSE => Abort FALSE ,
    CommandHalt FALSE => ComHalt FALSE ,
    Error FALSE => err FALSE ,
    ErrorID 0 => errid 0 );
```

LD：当 Exec 变为 True 时，执行两轴相对坐标跟随运动；程序如下：



- **注意事项**

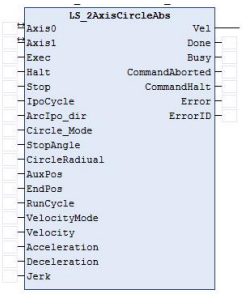
- 这个指令由“PMC_Ipolib”库实现。
- IpoCycle参数需要与运动指令所在的任务周期时间保持一致，否则指令执行时可能会出现Error报错。
- 参数IpoCycle、Jerk、Velocity、Acceleration、Deceleration不可以设置为0。
- 在指令运行期间，主从轴都不能再被其它运动指令所调用。
- Endpos是主从轴需要移动的距离；Endpos[0]是主轴需要移动的距离，Endpos[1]是从轴需要移动的距离。
- Velocity、Acceleration、Deceleration、Jerk大小不能超过“轴设置-SoftMotion驱动：通用”里面的速度，加速度，减速度，加加速度的动态限制，否则指令执行时可能会出现Error报错。
- 跟随运动指令仅在指令启动到结束这段时间有效，对主从轴有跟随控制效果，到达指令位置后，主从轴不再同步，都恢复standstill状态。
- 本指令根据主从轴需要移动的距离来分配主从轴速度，移动距离大的轴其速度为Velocity（30Pulse/s），另一轴的速度按移动距离比例分配。
- 相对坐标简单理解：相对于起始位置的坐标，即将起始位置作为坐标原点建立的坐标系

里的坐标。

7.10.16 LS_2AxisCircleAbs

指定的两个轴做圆弧插补运动，位置坐标为绝对坐标。

- **指令格式**

指令	名称	FB/FC	LD表现	ST表现	库
LS_2AxisCircleAbs	两轴绝对圆弧插补	FB		LS_2AxisCircleAbs(Axis0:=, Axis1:=, Exec:=, Halt:=, Stop:=, IpoCycle:=, ArcIpo_dir:=, Circle_Mode:=, StopAngle:=, CircleRadiual:=, AuxPos:=, EndPos:=, RunCycle:=, VelocityMode:=, Velocity:=, Acceleration:=, Deceleration:=, Jerk:=, Vel=>, Done=>, Busy=>, CommandAborted=>, CommandHalt=>, Error=>, ErrorID=>);	PMC_IpoLib

- **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	初始值	描述
Axis0	轴0	AXIS_REF_VIRTUAL_SM3	-	-	插补0轴
Axis1	轴1	AXIS_REF_VIRTUAL_SM3	-	-	插补1轴

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Exec	启动	BOOL	FALSE,TRUE	FALSE	启动命令，上升沿有效；
Halt	暂停	BOOL	TRUE-FALSE	FALSE	圆弧插补暂停命令，当该输入量为 True 时，当前插补运动暂停；当状态再次切换到 False 时，插补模块继续之前的未完成的插补任务。
Stop	停止	BOOL	TRUE-FALSE	FALSE	停止命令，该命令为 True 时，插补模块停止。
IpoCycle	插补周期	DWORD	-	2000	插补周期，单位：us，必须和所在任务周期一致；
ArcIpo_dir	插补方向	INT	0，1	0	圆弧插补的方向，0 表示逆时针方向，1 表示顺时针方向。圆弧模式 1、2、3 需要设置该值；
Circle_Mode	插补模式	INT	0-3	1	圆弧模式：0 表示三点圆弧模式；1 表示圆心终点模式；2 表示终点半径模式；3 表示目标角度模式；

StopAngle	停止角度	LREAL	0-360.0	0	停止的目标角度，相对于起点的角度，单位为度；插补模式 3 需要设置该值；
CircleRadiual	圆弧半径	LREAL	负数，0，正数	0	圆弧半径，单位：Pulse；插补模式 2 需要设置该值；半径为负表示圆弧角小于 180 度，半径为正表示圆弧角大于 180 度；
AuxPos	辅助位置	ARRAY[0..1]OFLREAL	负数，0，正数	0	辅助点，单位：Pulse
EndPos	终点位置	ARRAY[0..1]OFLREAL	负数，0，正数	0	终点位置，单位：Pulse
RunCycle	循环次数	UINT	0，正数	0	圆弧运动的循环次数（即圆弧旋转的圈数）
VelocityMode	速度模式	SMC_INT_VELMODE	0-3	SIGMOID	速度模式：梯形-0（TRAPEZOID）；S 形-1（SIGMOID）；四次方-3（QUADRATIC）
Velocity	速度	LREAL	0，正数	0	插补合成速度，Pulse/s
Acceleration	加速度	LREAL	0，正数	0	插补合成加速度，$Pulse/s^2$
Deceleration	减速度	LREAL	0，正数	0	插补合成减速度，$Pulse/s^2$
Jerk	加加速度	LREAL	负数，0，正数	90000000	加加速度，速度模式为 3 时需要设置此参数，该参数值不能为 0。

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
Vel	合速度	LREAL	0，正数	0	当前插补运动合成速度
Done	完成	BOOL	TRUE-FALSE	FALSE	True 表示插补模块已完成插补任务
Busy	运动中	BOOL	TRUE-FALSE	FALSE	True 表示插补运动正在进行之中。
Command Aborted	运动中断	BOOL	TRUE-FALSE	FALSE	当 Stop 有效时，该变量为 True，表示插补模块非正常中止。
Command Halt	运动暂停	BOOL	TRUE-FALSE	FALSE	当 Halt 为 True 时，该变量为 True，表示插补模块处于暂停状态。
Error	错误	BOOL	TRUE-FALSE	FALSE	错误状态；False-没有错误；
ErrorID	错误代码	DINT	0，正数	0	错误代码

	布尔	位串				整数								实数		时刻、持续时间 日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Axis0	AXIS_REF_VIRTUAL_SM3																			
Axis1	AXIS_REF_VIRTUAL_SM3																			
Exec	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Halt	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Stop	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
IpoCycle	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
ArcIpo_dir	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–
Circle_Mode	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–
StopAngle	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–
CircleRadiual	_	_	_	_	_	_	_	_	_	_	_	_	_	_	√	_	_	_	_	_
AuxPos	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
EndPos	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

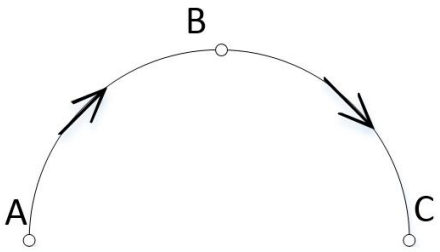
RunCycle	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–
VelocityMode	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Velocity	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–
Acceleration	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–
Deceleration	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–
Jerk	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–
Vel	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–
Done	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Busy	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
CommandAborted	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
CommandHalt	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Error	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
ErrorID	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–

- **功能说明：**

  - 指定的两个轴做圆弧插补运动，位置坐标为绝对坐标。

  - “Circle_Mode”参数用来设置圆弧插补的模式，支持四种模式，分别如下：

  1) 模式0（Circle_Mode=0）：三点圆弧模式，如下图；

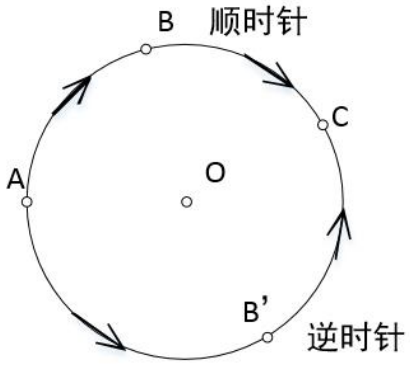


图三点圆弧插补

在使用三点圆弧模式时，需要在圆弧轨迹上确定 3 个不重合的点，如上图中的 A、B、C 三点。在指令中，A 代表圆弧插补轴 0 和 1 的当前位置，不需要设置；B 代表圆弧经过的点，需要设置为参数 AuxPos；C 代表圆弧的终点位置，需要设置为参数 Endpos。

注意：三个点不能在一条直线上！

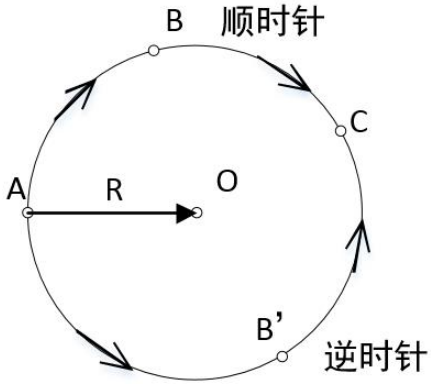
2) 模式1（Circle_Mode=1）：圆心与终点圆弧模式，如下图；



图圆心终点圆弧插补

在使用圆心终点圆弧模式时，需要设置圆心 O、终点 C 和运动方向等参数。在指令中 O 代表圆心坐标，需要设置为参数 AuxPos；C 代表圆弧的终点坐标，需要设置为参数 Endpos；运动方向需要设置参数 ArcIpo_dir。（注意：设置圆心坐标时要保证圆心到初始点 A 与终点 C 的距离相等）

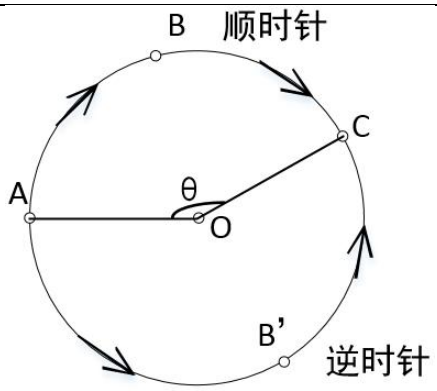
3) 模式2（Circle_Mode=2）：终点半径圆弧模式，如下图；



终点半径圆弧插补

在使用终点半径圆弧模式时，需要设置终点 C，半径 R 和运动方向等参数。在指令中 C 代表圆弧的终点位置，需要设置为参数 Endpos；半径 R 需要设置为参数 CircleRadiual；运动方向需要设置为参数 ArcIpo_dir。（注意：半径取值最小为“起始点-终点”距离的一半）注意：终点半径圆弧插补模式确定的圆有两个，需要通过半径设置正负来确定唯一的运动轨迹！

4) 模式3（Circle_Mode=3）：目标角度圆弧模式，如下图。



目标角度圆弧插补

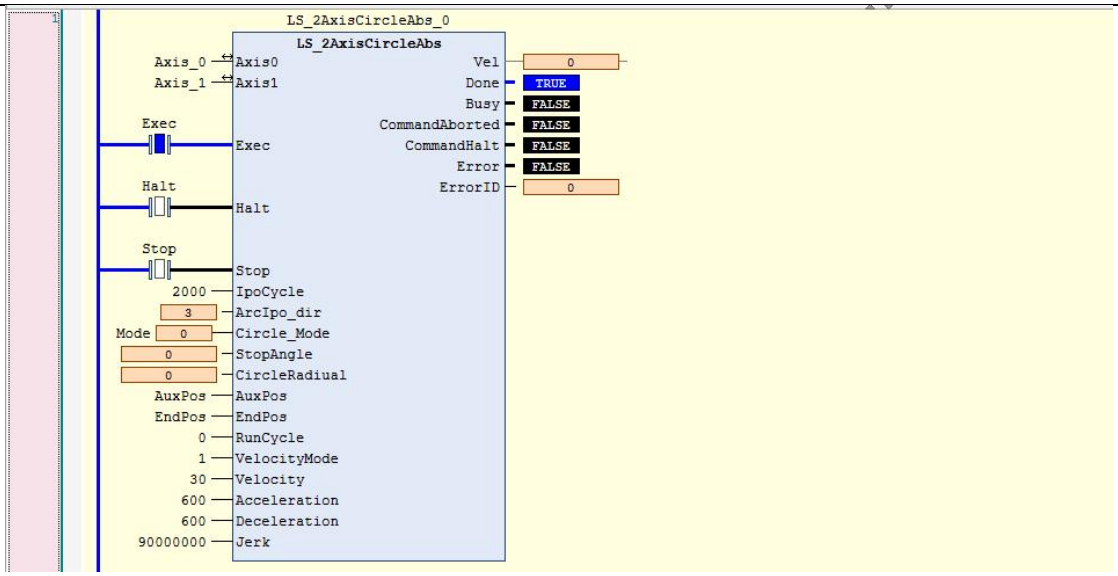
在使用目标角度圆弧模式时，需要设置圆心 O，角度 θ 和运动方向等参数。在指令中 O 代表圆心位置，需要设置为参数 Auxpos；目标角度 θ 需要设置为参数 StopAngle；运动方向需要设置为参数 ArcIpo_dir。

- **程序示例**

ST：当 Exec 变为 True 时，执行两轴绝对圆弧插补指令（插补模式 2）；程序如下：

```
LS_2AxisCircleAbs_0(
    Axis0:= Axis_0,
    Axis1:= Axis_1,
    Exec TRUE := Exec TRUE,
    Halt FALSE := Halt FALSE,
    Stop FALSE := Stop FALSE,
    IpoCycle 2000 := 2000,
    ArcIpo_dir 1 := dir 1,
    Circle_Mode 2 := Mode 2,
    StopAngle:= ,
    CircleRadiual 250 := Radiual 250,
    AuxPos:= ,
    EndPos:= EndPos,
    RunCycle 0 := 0,
    VelocityMode SIGMOID := 1,
    Velocity 30 := 30,
    Acceleration 600 := 600,
    Deceleration 600 := 600,
    Jerk 9E+07 := jerk 9E+07,
    Vel 0 => vel 0,
    Done TRUE => done TRUE,
    Busy FALSE => busy FALSE,
    CommandAborted FALSE => Abort FALSE,
    CommandHalt FALSE => ComHalt FALSE,
    Error FALSE => err FALSE,
    ErrorID 0 => errid 0 );RETURN
```

LD：当 Exec 变为 True 时，执行两轴绝对圆弧插补指令（插补模式 0）；程序如下：



- **注意事项**

- 这个指令由“PMC_Ipolib”库实现。
- IpoCycle参数需要与运动指令所在的任务周期时间保持一致，否则指令执行时可能会出现Error报错。
- 参数IpoCycle、Jerk、Velocity、Acceleration、Deceleration不可以设置为0，否则指令执行时可能会出现Error报错。
- 在指令运行期间，运动轴都不能再被其它运动指令所调用。
- Endpos是两插补轴的终点位置（绝对坐标）；Endpos[0]是插补0轴终点位置，Endpos[1]是插补1轴终点位置。
- Velocity、Acceleration、Deceleration、Jerk大小不能超过“轴设置-SoftMotion驱动：通用”里面的速度，加速度，减速度，加加速度的动态限制，否则指令执行时可能会出现Error报错。
- 插补轴需要先把轴上使能（运行MC_POWER指令），才能运行插补指令。
- 本指令始终保证两插补轴的合速度Vel等于设置的插补速度Velocity（30Pulse/s）。
- 各个插补模式中没有用到的参数就不要填。
- 注意插补模式2中，半径取值最小为“起始点-终点”距离的一半；否则运行指令可能报错，

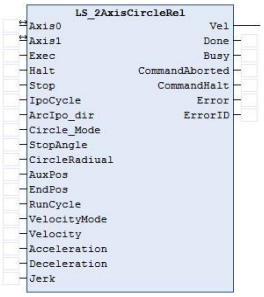
错误代码为268435443。

- RunCycle是圆弧插补运动旋转的圈数，设置为0表示直接从起始点运动到终点位置；设置为1,2，3..表示从起始点绕圆心旋转1,2.3..圈回到起始点位置，再从起始点运动到终点位置。

7.10.17 LS_2AxisCircleRel

指定的两个轴做圆弧插补运动，位置坐标为相对坐标。

- **指令格式**

指令	名称	FB/FC	LD表现	ST表现	库
LS_2AxisCircleRel	两轴相对圆弧插补	FB		LS_2AxisCircleRel(Axis0:=, Axis1:=, Exec:=, Halt:=, Stop:=, IpoCycle:=, ArcIpo_dir:=, Circle_Mode:=, StopAngle:=, CircleRadiual:=, AuxPos:=, EndPos:=, RunCycle:=, VelocityMode:=, Velocity:=, Acceleration:=, Deceleration:=, Jerk:=, Vel=>, Done=>, Busy=>, CommandAborted=>, CommandHalt=>, Error=>, ErrorID=>);	PMC_IpoLib

- 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	初始值	描述
Axis0	轴0	AXIS_REF_VIRTUAL_SM3	-	-	插补0轴
Axis1	轴1	AXIS_REF_VIRTUAL_SM3	-	-	插补1轴

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Exec	启动	BOOL	FALSE,TRUE	FALSE	启动命令，上升沿有效；
Halt	暂停	BOOL	TRUE-FALSE	FALSE	圆弧插补暂停命令，当该输入量为 True 时，当前插补运动暂停；当状态再次切换到 False 时，插补模块继续之前的未完成的插补任务。
Stop	停止	BOOL	TRUE-FALSE	FALSE	停止命令，该命令为 True 时，插补模块停止。
IpoCycle	插补周期	DWORD	-	2000	插补周期，单位：us，必须和所在任务周期一致；
ArcIpo_dir	插补方向	INT	0，1	0	圆弧插补的方向，0 表示逆时针方向，1 表示顺时针方向。圆弧模式 1、2、3 需要设置该值；
Circle_Mode	插补模式	INT	0-3	1	圆弧模式：0 表示三点圆弧模式；1 表示圆心终点模式；2 表示终点半径模式；3 表示目

					标角度模式；
StopAngle	停止角度	LREAL	0-360.0	0	停止的目标角度，相对于起点的角度，单位为度；插补模式 3 需要设置该值；
CircleRadiual	圆弧半径	LREAL	负数，0 正数	0	圆弧半径，单位：Pulse；插补模式 2 需要设置该值；半径为负表示圆弧角小于 180 度，半径为正表示圆弧角大于 180 度；
AuxPos	辅助位置	ARRAY[0..1]OFLREAL	负数，0 正数	0	辅助点，单位：Pulse
EndPos	终点位置	ARRAY[0..1]OFLREAL	负数，0 正数	0	终点位置，单位：Pulse
RunCycle	循环次数	UINT	0，正数	0	圆弧运动的循环次数（即圆弧旋转的圈数）
VelocityMode	速度模式	SMC_INT_VELMODE	0-3	SIGMOID	速度模式：梯形-0（TRAPEZOID）；S 形-1（SIGMOID）；四次方-3（QUADRATIC）
Velocity	速度	LREAL	0，正数	0	插补合成速度，Pulse/s
Acceleration	加速度	LREAL	0，正数	0	插补合成加速度，Pulse/s^2
Deceleration	减速度	LREAL	0，正数	0	插补合成减速度，Pulse/s^2
Jerk	加加速度	LREAL	负数，0 正数	90000000	加加速度，速度模式为 3 时需要设置此参数，该参数值不能为 0。

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
Vel	合速度	LREAL	0，正数	0	当前插补运动合成速度
Done	完成	BOOL	TRUE-FALSE	FALSE	True 表示插补模块已完成插补任务
Busy	运动中	BOOL	TRUE-FALSE	FALSE	True 表示插补运动正在进行之中。
CommandAborted	运动中断	BOOL	TRUE-FALSE	FALSE	当 Stop 有效时，该变量为 True，表示插补模块非正常中止。
CommandHalt	运动暂停	BOOL	TRUE-FALSE	FALSE	当 Halt 为 True 时，该变量为 True，表示插补模块处于暂停状态。
Error	错误	BOOL	TRUE-FALSE	FALSE	错误状态；False-没有错误；
ErrorID	错误代码	DINT	0，正数	0	错误代码

	布尔	位串				整数								实数		时刻、持续时间 日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Axis0	AXIS_REF_VIRTUAL_SM3																			
Axis1	AXIS_REF_VIRTUAL_SM3																			
Exec	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Halt	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Stop	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
IpoCycle	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
ArcIpo_dir	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Circle_Mode	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
StopAngle	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
CircleRadiual	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
AuxPos	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
EndPos	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
RunCycle	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–
VelocityMode	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_
Velocity	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–
Acceleration	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–
Deceleration	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–
Jerk	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–
Vel	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–
Done	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

Busy	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
CommandAborted	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
CommandHalt	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Error	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
ErrorID	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–

- **功能说明：**

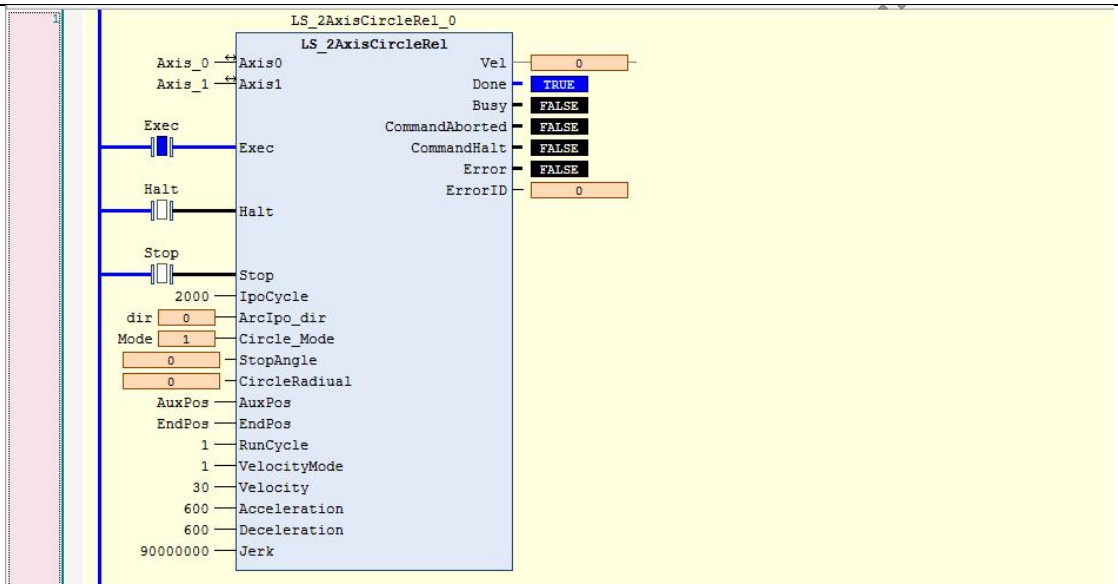
  - 指定的两个轴做圆弧插补运动，位置坐标为相对坐标。除相对坐标外，其他参数设置与绝对坐标两轴圆弧插补指令相同。

- **程序示例**

ST：当Exec变为True时，执行两轴相对圆弧插补指令（插补模式3）；程序如下：

```
LS_2AxisCircleRel_0(
    Axis0:= Axis_0,
    Axis1:= Axis_1,
    Exec:= Exec,
    Halt:= Halt,
    Stop:= Stop,
    IpoCycle:= 2000,
    ArcIpo_dir:= dir,
    Circle_Mode:= Mode,
    StopAngle:= Angle,
    CircleRadiual:= ,
    AuxPos:= AuxPos,
    EndPos:= ,
    RunCycle:= 1,
    VelocityMode:= 1,
    Velocity:= 30,
    Acceleration:= 600,
    Deceleration:= 600,
    Jerk:= jerk,
    Vel=> vel,
    Done=> done,
    Busy=> busy,
    CommandAborted=> Abort,
    CommandHalt=> ComHalt,
    Error=> err,
    ErrorID=> errid);
```

LD：当Exec变为True时，执行两轴相对圆弧插补指令（插补模式3）；程序如下：



- **注意事项**

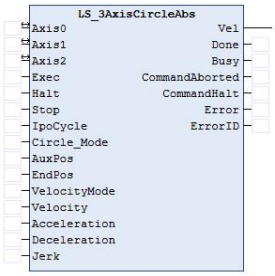
- 这个指令由“PMC_Ipolib”库实现。
- IpoCycle参数需要与运动指令所在的任务周期时间保持一致，否则指令执行时可能会出现Error报错。
- 参数IpoCycle、Jerk、Velocity、Acceleration、Deceleration不可以设置为0，否则指令执行时可能会出现Error报错。
- 在指令运行期间，运动轴都不能再被其它运动指令所调用。
- AuxPos是圆弧插补运动的圆心坐标。AuxPos，Endpos都是相对坐标。
- Velocity、Acceleration、Deceleration、Jerk大小不能超过“轴设置-SoftMotion驱动：通用”里面的速度，加速度，减速度，加加速度的动态限制，否则指令执行时可能会出现Error报错。
- 插补轴需要先把轴上使能（运行MC_POWER指令），才能运行插补指令。
- 本指令始终保证两插补轴的合速度Vel等于设置的插补速度Velocity（30Pulse/s）。
- 各个插补模式中没有用到的参数就不要填。
- 注意插补模式1中，设置圆心坐标时要保证圆心到起始点与终点的距离相等。

- RunCycle是圆弧插补运动旋转的圈数，设置为0表示圆弧运动无限循环；设置为1表示从起始点运动到终点位置；设置为2,3,4..表示从起始点绕圆心旋转1,2.3..圈回到起始点位置，再从起始点运动到终点位置。
- 相对坐标简单理解：相对于起始位置的坐标，即将起始位置作为坐标原点建立的坐标系里的坐标。

7.10.18 LS_3AxisCircleAbs

指定的三个轴做空间圆弧插补运动，位置坐标为绝对坐标。

- **指令格式**

指令	名称	FB/FC	LD表现	ST表现	库
LS_3AxisCircleAbs	三轴绝对圆弧插补	FB		LS_3AxisCircleAbs(Axis0:=, Axis1:=, Axis2:=, Exec:=, Halt:=, Stop:=, IpoCycle:=, Circle_Mode:=, AuxPos:=, EndPos:=, VelocityMode:=, Velocity:=, Acceleration:=, Deceleration:=, Jerk:=, Vel=>, Done=>, Busy=>, CommandAborted=>, CommandHalt=>, Error=>, ErrorID=>);	PMC_IpoLib

- **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	初始值	描述
Axis0	轴0	AXIS_REF_VIRTUAL_SM3	-	-	插补0轴
Axis1	轴1	AXIS_REF_VIRTUAL_SM3	-	-	插补1轴
Axis2	轴2	AXIS_REF_VIRTUAL_SM3	-	-	插补2轴

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Exec	启动	BOOL	FALSE,TRUE	FALSE	启动命令，上升沿有效；
Halt	暂停	BOOL	FALSE,TRUE	FALSE	圆弧插补暂停命令，当该输入量为 True 时，当前插补运动暂停；当状态再次切换到 False 时，插补模块继续之前的未完成的插补任务。
Stop	停止	BOOL	FALSE,TRUE	FALSE	停止命令，该命令为 True 时，插补模块停止。
IpoCycle	插补周期	DWORD	-	2000	插补周期，单位：us，必须和所在任务周期一致；
Circle_Mode	插补模式	INT	0	0	圆弧模式：0 表示三点圆弧模式；只支持三点圆弧模式；
AuxPos	辅助位置	ARRAY[0..2]OFLREAL	负数，0 正数	0	辅助点，单位：Pulse
EndPos	终点位置	ARRAY[0..2]OFLREAL	负数，0 正数	0	终点位置，单位：Pulse

VelocityMode	速度模式	SMC_INT_VELMODE	0-3	SIGMOID	速度模式：梯形-0（TRAPEZOID）；S 形-1（SIGMOID）；四次方-3（QUADRATIC）
Velocity	速度	LREAL	0，正数	0	插补合成速度，Pulse/s
Acceleration	加速度	LREAL	0，正数	0	插补合成加速度，Pulse/s^2
Deceleration	减速度	LREAL	0，正数	0	插补合成减速度，Pulse/s^2
Jerk	加加速度	LREAL	负数，0 正数	90000000	加加速度，速度模式为 3 时需要设置此参数，该参数值不能为 0。

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
Vel	合速度	LREAL	0，正数	0	当前插补运动合成速度
Done	完成	BOOL	FALSE,TRUE	FALSE	True 表示插补模块已完成插补任务
Busy	运动中	BOOL	FALSE,TRUE	FALSE	True 表示插补运动正在进行之中。
Command Aborted	运动中断	BOOL	FALSE,TRUE	FALSE	当 Stop 有效时，该变量为 True，表示插补模块非正常中止。
Command Halt	运动暂停	BOOL	FALSE,TRUE	FALSE	当 Halt 为 True 时，该变量为 True，表示插补模块处于暂停状态。
Error	错误	BOOL	FALSE,TRUE	FALSE	错误状态；False-没有错误；
ErrorID	错误代码	DINT	0，正数	0	错误代码

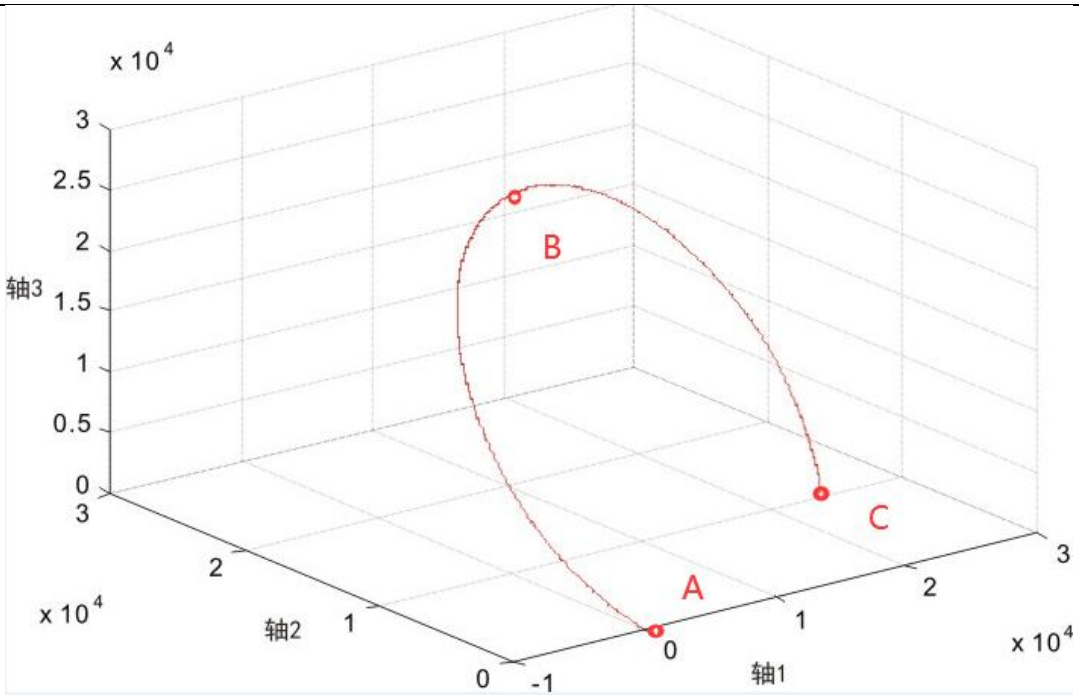
	布尔	位串	整数	实数	时刻、持续时间 日期、字符串

	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Axis0	AXIS_REF_VIRTUAL_SM3																			
Axis1	AXIS_REF_VIRTUAL_SM3																			
Axis2	AXIS_REF_VIRTUAL_SM3																			
Exec	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Halt	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Stop	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
IpoCycle	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Circle_Mode	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–
AuxPos	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
EndPos	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
VelocityMode	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Velocity	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–
Acceleration	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–
Deceleration	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–
Jerk	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–
Vel	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–
Done	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Busy	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
CommandAborted	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
CommandHalt	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Error	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
ErrorID	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–

- **功能说明：**

  - 指定的三个轴做空间圆弧插补运动，位置坐标为绝对坐标。

  - “Circle_Mode”参数用来设置三轴空间圆弧插补的模式，仅支持三点圆弧插补模式：模式 0（Circle_Mode=0）：三点圆弧模式，如下图；



图三点圆弧插补

- 在使用三点圆弧模式时，需要在圆弧轨迹上确定3个不重合的点，如上图中的A、B、C三点。在指令中，A代表圆弧插补轴0，1，2的当前位置，不需要设置；B代表圆弧经过的点，需要设置为参数AuxPos；C代表圆弧的终点位置，需要设置为参数Endpos。

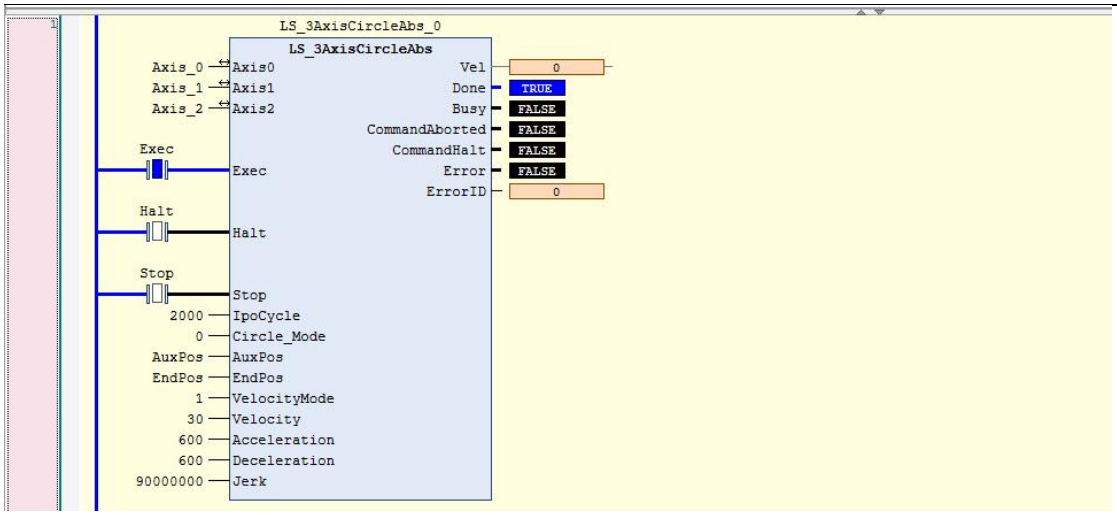
注意：三个点不能在一条直线上！

- **程序示例**

ST：当 Exec 变为 True 时，执行三轴绝对圆弧插补指令；程序如下：

```
LS_3AxisCircleAbs_0(
    Axis0:= Axis_0,
    Axis1:= Axis_1,
    Axis2:= Axis_2,
    Exec TRUE := Exec TRUE,
    Halt FALSE := Halt FALSE,
    Stop FALSE := Stop FALSE,
    IpoCycle 2000 := 2000,
    Circle_Mode 0 := 0,  //只支持插补模式0，三点圆弧模式
    AuxPos:= AuxPos,
    EndPos:= EndPos,
    VelocityMode SIGMOID := 1,
    Velocity 30 := 30,
    Acceleration 600 := 600,
    Deceleration 600 := 600,
    Jerk 9E+07 := jerk 9E+07 ,
    Vel 0 => vel 0 ,
    Done TRUE => done TRUE ,
    Busy FALSE => busy FALSE ,
    CommandAborted FALSE => Abort FALSE ,
    CommandHalt FALSE => ComHalt FALSE ,
    Error FALSE => err FALSE ,
    ErrorID 0 => errid 0 );
```

LD：当 Exec 变为 True 时，执行三轴绝对圆弧插补指令；程序如下：



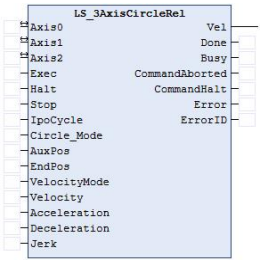
- **注意事项**

- 这个指令由“PMC_Ipolib”库实现。
- IpoCycle参数需要与运动指令所在的任务周期时间保持一致，否则指令执行时可能会出现Error报错。
- 参数IpoCycle、Jerk、Velocity、Acceleration、Deceleration不可以设置为0，否则指令执行时可能会出现Error报错。
- 在指令运行期间，运动轴都不能再被其它运动指令所调用。
- Endpos是三个插补轴的终点位置（绝对坐标）；Endpos[0]是插补0轴终点位置，Endpos[1]是插补1轴终点位置，Endpos[2]是插补2轴终点位置。
- Velocity、Acceleration、Deceleration、Jerk大小不能超过“轴设置-SoftMotion驱动：通用”里面的速度，加速度，减速度，加加速度的动态限制，否则指令执行时可能会出现Error报错。
- 插补轴需要先把轴上使能（运行MC_POWER指令），才能运行插补指令。
- 本指令始终保证三个插补轴的合速度Vel等于设置的插补速度Velocity（30Pulse/s）。

7.10.19 LS_3AxisCircleRel

指定的三个轴做空间圆弧插补运动，位置坐标为相对坐标。

- **指令格式**

指令	名称	FB/FC	LD表现	ST表现	库
LS_3AxisCircleRel	三轴相对对圆弧插补	FB		LS_3AxisCircleRel(Axis0:=, Axis1:=, Axis2:=, Exec:=, Halt:=, Stop:=, IpoCycle:=, Circle_Mode:=, AuxPos:=, EndPos:=, VelocityMode:=, Velocity:=, Acceleration:=, Deceleration:=, Jerk:=, Vel=>, Done=>, Busy=>, CommandAborted=>, CommandHalt=>, Error=>, ErrorID=>);	PMC_IpoLib

- **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	初始值	描述
Axis0	轴0	AXIS_REF_VIRTUAL_SM3	-	-	插补0轴
Axis1	轴1	AXIS_REF_VIRTUAL_SM3	-	-	插补1轴

Axis2	轴2	AXIS_REF_VIRTUAL_SM3	-	-	插补2轴

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Exec	启动	BOOL	FALSE,TRUE	FALSE	启动命令，上升沿有效；
Halt	暂停	BOOL	FALSE,TRUE	FALSE	圆弧插补暂停命令，当该输入量为 True 时，当前插补运动暂停；当状态再次切换到 False 时，插补模块继续之前的未完成的插补任务。
Stop	停止	BOOL	FALSE,TRUE	FALSE	停止命令，该命令为 True 时，插补模块停止。
IpoCycle	插补周期	DWORD	-	2000	插补周期，单位：us，必须和所在任务周期一致；
Circle_Mode	插补模式	INT	0	0	圆弧模式：0 表示三点圆弧模式；只支持三点圆弧模式；
AuxPos	辅助位置	ARRAY[0..2]OFLREAL	负数，0 正数	0	辅助点，单位：Pulse
EndPos	终点位置	ARRAY[0..2]OFLREAL	负数，0 正数	0	终点位置，单位：Pulse
VelocityMode	速度模式	SMC_INT_VELMODE	0-3	SIGMOID	速度模式：梯形-0（TRAPEZOID）；S 形-1（SIGMOID）；四次方-3（QUADRATIC）
Velocity	速度	LREAL	0，正数	0	插补合成速度，Pulse/s

Acceleration	加速度	LREAL	0，正数	0	插补合成加速度，Pulse/s^2
Deceleration	减速度	LREAL	0，正数	0	插补合成减速度，Pulse/s^2
Jerk	加加速度	LREAL	负数，0 正数	90000000	加加速度，速度模式为 3 时需要设置此参数，该参数值不能为 0。

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
Vel	合速度	LREAL	0，正数	0	当前插补运动合成速度
Done	完成	BOOL	FALSE,TRUE	FALSE	True 表示插补模块已完成插补任务
Busy	运动中	BOOL	FALSE,TRUE	FALSE	True 表示插补运动正在进行之中。
Command Aborted	运动中断	BOOL	FALSE,TRUE	FALSE	当 Stop 有效时，该变量为 True，表示插补模块非正常中止。
Command Halt	运动暂停	BOOL	FALSE,TRUE	FALSE	当 Halt 为 True 时，该变量为 True，表示插补模块处于暂停状态。
Error	错误	BOOL	FALSE,TRUE	FALSE	错误状态；False-没有错误；
ErrorID	错误代码	DINT	0，正数	0	错误代码

	布尔	位串				整数								实数		时刻、持续时间 日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Axis0	AXIS_REF_VIRTUAL_SM3																			
Axis1	AXIS_REF_VIRTUAL_SM3																			
Axis2	AXIS_REF_VIRTUAL_SM3																			
Exec	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Halt	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Stop	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
IpoCycle	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Circle_Mode	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–
AuxPos	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
EndPos	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
VelocityMode	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Velocity	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–
Acceleration	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–
Deceleration	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–
Jerk	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–
Vel	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–
Done	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Busy	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
CommandAborted	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
CommandHalt	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Error	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
ErrorID	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–

- **功能说明：**

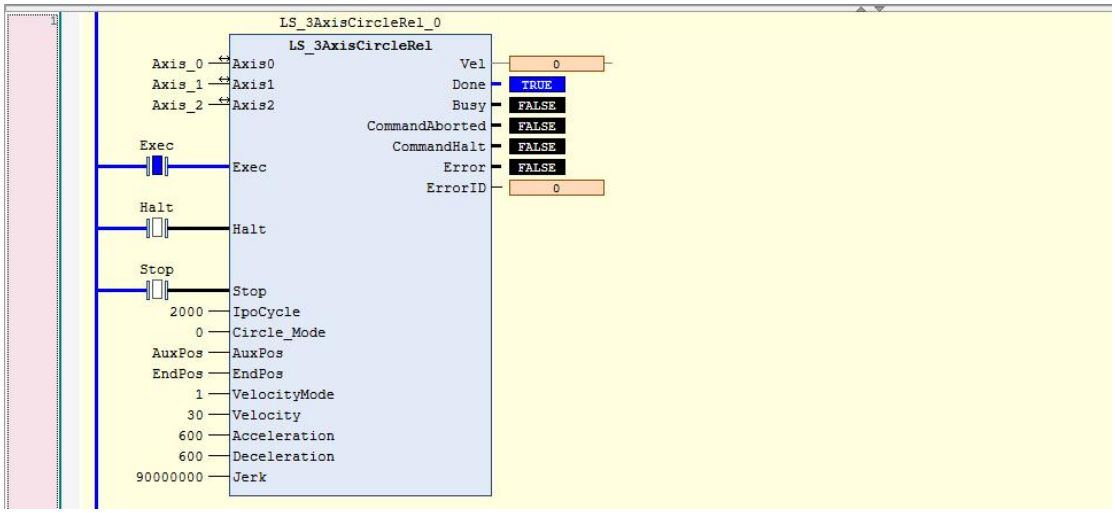
  - 指定的三个轴做空间圆弧插补运动，位置坐标为相对坐标。除相对坐标外，其他参数设置与绝对坐标三轴空间圆弧插补指令相同。

- **程序示例**

ST：当 Exec 变为 True 时，执行三轴相对圆弧插补指令；程序如下：

```
MC_Power_0( [11 lines]
MC_Power_1( [11 lines]
MC_Power_2( [11 lines]
LS_3AxisCircleRel_0(
    Axis0:= Axis_0,
    Axis1:= Axis_1,
    Axis2:= Axis_2,
    Exec TRUE := Exec TRUE,
    Halt FALSE := Halt FALSE,
    Stop FALSE := Stop FALSE,
    IpoCycle 2000 := 2000,
    Circle_Mode 0 := 0,  //只支持插补模式0，三点圆弧模式
    AuxPos:= AuxPos,
    EndPos:= EndPos,
    VelocityMode SIGMOID := 1,
    Velocity 30 := 30,
    Acceleration 600 := 600,
    Deceleration 600 := 600,
    Jerk 9E+07 := jerk 9E+07 ,
    Vel 0 => vel 0 ,
    Done TRUE => done TRUE ,
    Busy FALSE => busy FALSE ,
    CommandAborted FALSE => Abort FALSE ,
    CommandHalt FALSE => ComHalt FALSE ,
    Error FALSE => err FALSE ,
    ErrorID 0 => errid 0 );
```

LD：当 Exec 变为 True 时，执行三轴相对圆弧插补指令；程序如下：



- **注意事项**

➢ 这个指令由“PMC_Ipolib”库实现。

➢ IpoCycle参数需要与运动指令所在的任务周期时间保持一致，否则指令执行时可能会出现Error报错。

➢ 参数IpoCycle、Jerk、Velocity、Acceleration、Deceleration不可以设置为0，否则指令执行时可能会出现Error报错。

➢ 在指令运行期间，运动轴都不能再被其它运动指令所调用。

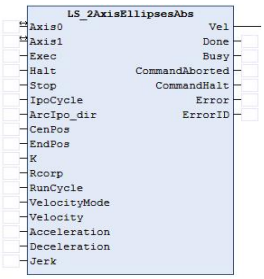
➢ AuxPos是空间圆弧插补运动的圆心坐标。AuxPos，Endpos都是相对坐标。

- Velocity、Acceleration、Deceleration、Jerk大小不能超过“轴设置-SoftMotion驱动：通用”里面的速度，加速度，减速度，加加速度的动态限制，否则指令执行时可能会出现Error报错。
- 插补轴需要先把轴上使能（运行MC_POWER指令），才能运行插补指令。
- 本指令始终保证三个插补轴的合速度Vel等于设置的插补速度Velocity（30Pulse/s）。
- 相对坐标简单理解：相对于起始位置的坐标，即将起始位置作为坐标原点建立的坐标系里的坐标。

7.10.20 LS_2AxisEllipsesAbs

指定的两个轴做椭圆插补运动，位置坐标为绝对坐标。

- **指令格式**

指令	名称	FB/FC	LD表现	ST表现	库
LS_2AxisEllipsesAbs	两轴绝对椭圆插补	FB		LS_2AxisEllipsesAbs (Axis0:=, Axis1:=, Exec:=, Halt:=, Stop:=, IpoCycle:=, ArcIpo_dir:=, CenPos:=, EndPos:=, K:=, Rcorp:=, RunCycle:=, VelocityMode:=, Velocity:=, Acceleration:=, Deceleration:=, Jerk:=, Vel=>, Done=>,	PMC_IpoLib

				Busy=>, CommandAborted=>, CommandHalt=>, Error=>, ErrorID=>);	

- **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	初始值	描述
Axis0	轴0	AXIS_REF_VIRTUAL_SM3	-	-	插补0轴
Axis1	轴1	AXIS_REF_VIRTUAL_SM3	-	-	插补1轴

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Exec	启动	BOOL	FALSE,TRUE	FALSE	启动命令，上升沿有效；
Halt	暂停	BOOL	FALSE,TRUE	FALSE	椭圆插补暂停命令，当该输入量为 True 时，当前插补运动暂停；当状态再次切换到 False 时，插补模块继续之前的未完成的插补任务。
Stop	停止	BOOL	FALSE,TRUE	FALSE	停止命令，该命令为 True 时，插补模块停止。
IpoCycle	插补周期	DWORD	-	2000	插补周期，单位：us，必须和所在任务周期一致；
ArcIpo_dir	插补方向	INT	0，1	0	椭圆插补的方向，0 表示逆时针方向，1 表示顺时针方向。

CenPos	中心位置	ARRAY[0..1]OFLREAL	负数，0 正数	0	椭圆的中心位置，单位：Pulse
EndPos	终点位置	ARRAY[0..1]OFLREAL	负数，0 正数	0	终点位置，单位：Pulse
K	主轴角度	LREAl	0-360.0	0	主轴的角度，主轴相对于X轴正向的角度；长轴与X轴正向的夹角，用来确定主轴的方向。
Rcorp	短轴长轴比	LREAL	0，1	0	短轴与长轴的比，取值范围为[0，1]
RunCycle	循环次数	UINT	0，正数	0	椭圆运动的循环次数（即椭圆的圈数）
VelocityMode	速度模式	SMC_INT_VELMODE	0-3	SIGMOID	速度模式：梯形-0（TRAPEZOID）；S 形-1（SIGMOID）；四次方-3（QUADRATIC）
Velocity	速度	LREAL	0，正数	0	插补合成速度，Pulse/s
Acceleration	加速度	LREAL	0，正数	0	插补合成加速度，Pulse/s^2
Deceleration	减速度	LREAL	0，正数	0	插补合成减速度，Pulse/s^2
Jerk	加加速度	LREAL	负数，0，正数	90000000	加加速度，速度模式为 3 时需要设置此参数，该参数值不能为 0。

输出变量

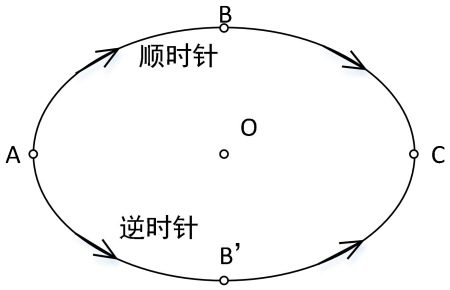
输出变量	名称	数据类型	有效范围	初始值	描述
Vel	合速度	LREAL	0，正数	0	当前插补运动合成速度
Done	完成	BOOL	FALSE,TRUE	FALSE	True 表示插补模

					块已完成插补任务
Busy	运动中	BOOL	FALSE,TRUE	FALSE	True 表示插补运动正在进行之中。
CommandAborted	运动中断	BOOL	FALSE,TRUE	FALSE	当 Stop 有效时，该变量为 True，表示插补模块非正常中止。
CommandHalt	运动暂停	BOOL	FALSE,TRUE	FALSE	当 Halt 为 True 时，该变量为 True，表示插补模块处于暂停状态。
Error	错误	BOOL	FALSE,TRUE	FALSE	错误状态；False-没有错误；
ErrorID	错误代码	DINT	0，正数	0	错误代码

	布尔	位串				整数								实数		时刻、持续时间 日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Axis0	AXIS_REF_VIRTUAL_SM3																			
Axis1	AXIS_REF_VIRTUAL_SM3																			
Exec	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Halt	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Stop	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
IpoCycle	-	-	-	√	-	-	-	-	-	-	-	-	-	–	-	-	-	-	–	-
ArcIpo_dir	-	-	-	-	-	-	-	-	-	-	√	-	-	–	-	-	-	-	–	–
CenPos	-	-	-	-	-	-	-	-	-	-	-	-	-	–	-	-	-	-	–	–
EndPos	-	-	-	-	-	-	-	-	-	-	-	-	-	–	-	-	-	-	–	–
K	-	-	-	-	-	-	-	-	-	-	-	-	-	–	√	-	-	-	–	–
Rcorp	-	-	-	-	-	-	-	-	-	-	-	-	-	–	√	-	-	-	–	–
RunCycle	-	-	-	-	-	-	√	-	-	-	-	-	-	–	-	-	-	-	–	–
Velocity Mode	-	-	-	-	-	-	-	-	-	-	-	-	-	–	-	-	-	-	–	–

Velocity	-	-	-	-	-	-	-	-	-	-	-	-	-	–	√	-	-	-	–	–
Acceleration	-	-	-	-	-	-	-	-	-	-	-	-	-	–	√	-	-	-	–	–
Deceleration	-	-	-	-	-	-	-	-	-	-	-	-	-	–	√	-	-	-	–	–
Jerk	-	-	-	-	-	-	-	-	-	-	-	-	-	–	√	-	-	-	–	–
Vel	-	-	-	-	-	-	-	-	-	-	-	-	-	–	√	-	-	-	–	–
Done	√	-	-	-	-	-	-	-	-	-	-	-	-	–	-	-	-	-	–	–
Busy	√	-	-	-	-	-	-	-	-	-	-	-	-	–	-	-	-	-	–	–
CommandAborted	√	-	-	-	-	-	-	-	-	-	-	-	-	–	-	-	-	-	–	–
CommandHalt	√	-	-	-	-	-	-	-	-	-	-	-	-	–	-	-	-	-	–	–
Error	√	-	-	-	-	-	-	-	-	-	-	-	-	–	-	-	-	-	–	–
ErrorID	-	-	-	-	-	-	-	-	-	-	-	√	-	–	-	-	-	-	–	–

- **功能说明：**
  - 指定的两个轴做椭圆插补运动，位置坐标为绝对坐标。
  - 椭圆插补的轨迹，如下图所示：



椭圆插补

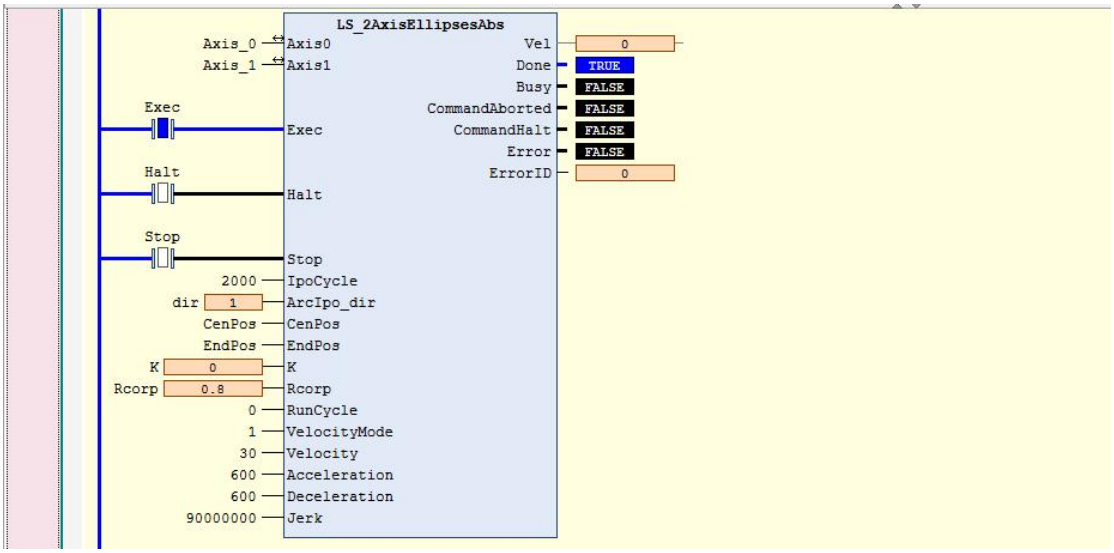
  - 在使用椭圆插补运动时，需要设置圆心O、终点C和运动方向等参数。在指令中，A代表圆弧插补轴0和1的当前位置，不需要设置；O代表圆心位置，需要设置为参数CenPos；C代表圆弧的终点位置，需要设置为参数Endpos；运动方向需要设置为参数ArcIpo_dir；椭圆轨迹执行圈数，则需要设置为参数RunCycle。
  - K是椭圆主轴相对于X轴正向的角度，用来确定主轴的方向；Rcorp是椭圆短轴与长轴的比值（当Rcorp=1时，是一个圆）。

- **程序示例**

ST：当Exec变为True时，执行两轴绝对椭圆插补指令；程序如下：

```
LS_2AxisEllipsesAbs_0(
    Axis0:= Axis_0,
    Axis1:= Axis_1,
    Exec TRUE := Exec TRUE,
    Halt FALSE := Halt FALSE,
    Stop FALSE := Stop FALSE,
    IpoCycle 2000 := 2000,
    ArcIpo_dir 0 := dir 0,
    CenPos:= CenPos,
    EndPos:= EndPos,
    K 20 := K 20,
    Rcorp 0.8 := Rcorp 0.8,
    RunCycle 0 := 0,
    VelocityMode SIGMOID := 1,
    Velocity 30 := 30,
    Acceleration 600 := 600,
    Deceleration 600 := 600,
    Jerk 9E+07 := jerk 9E+07,
    Vel 0 => vel 0,
    Done TRUE => done TRUE,
    Busy FALSE => busy FALSE,
    CommandAborted FALSE => Abort FALSE,
    CommandHalt FALSE => ComHalt FALSE,
    Error FALSE => err FALSE,
    ErrorID 0 => errid 0);
```

LD：当Exec变为True时，执行两轴绝对椭圆插补指令；程序如下：



- **注意事项：**

➢ 这个指令由“PMC_Ipolib”库实现。

➢ IpoCycle参数需要与运动指令所在的任务周期时间保持一致，否则指令执行时可能会出现Error报错。

➢ 参数IpoCycle、Jerk、Velocity、Acceleration、Deceleration不可以设置为0，否则指令执行

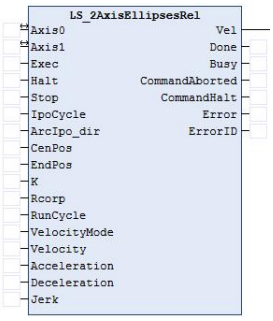
时可能会出现Error报错。

- 在指令运行期间，运动轴都不能再被其它运动指令所调用。
- Endpos是两插补轴的终点位置（绝对坐标）；Endpos[0]是插补0轴终点位置，Endpos[1]是插补1轴终点位置。
- Velocity、Acceleration、Deceleration、Jerk大小不能超过“轴设置-SoftMotion驱动：通用”里面的速度，加速度，减速度，加加速度的动态限制，否则指令执行时可能会出现Error报错。
- 插补轴需要先把轴上使能（运行MC_POWER指令），才能运行插补指令。
- 本指令始终保证两插补轴的合速度Vel等于设置的插补速度Velocity（30Pulse/s）。
- RunCycle是椭圆插补运动旋转的圈数，设置为0表示直接从起始点运动到终点位置；设置为1,2，3..表示从起始点绕圆心旋转1,2.3..圈回到起始点位置，再从起始点运动到终点位置。

7.10.21 LS_2AxisEllipsesRel

指定的两个轴做椭圆插补运动，位置坐标为相对坐标。

- **指令格式**

指令	名称	FB/FC	LD 表现	ST 表现	库
LS_2AxisEllipsesRel	两轴相对椭圆插补	FB		LS_2AxisEllipsesRel(Axis0:=, Axis1:=, Exec:=, Halt:=, Stop:=, IpoCycle:=, ArcIpo_dir:=, CenPos:=, EndPos:=, K:=, Rcorp:=, RunCycle:=,	PMC_IpoLib

				VelocityMode:=, Velocity:=, Acceleration:=, Deceleration:=, Jerk:=, Vel=>, Done=>, Busy=>, CommandAborted=>, CommandHalt=>, Error=>, ErrorID=>);	

- **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	初始值	描述
Axis0	轴0	AXIS_REF_VIRTUAL_SM3	-	-	插补0轴
Axis1	轴1	AXIS_REF_VIRTUAL_SM3	-	-	插补1轴

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Exec	启动	BOOL	FALSE,TRUE	FALSE	启动命令，上升沿有效；
Halt	暂停	BOOL	FALSE,TRUE	FALSE	椭圆插补暂停命令，当该输入量为 True 时，当前插补运动暂停；当状态再次切换到 False 时，插补模块继续之前的未完成的插补任务。
Stop	停止	BOOL	FALSE,TRUE	FALSE	停止命令，该命令为 True 时，插补模块停止。

IpoCycle	插补周期	DWORD	-	2000	插补周期，单位：us，必须和所在任务周期一致；
ArcIpo_dir	插补方向	INT	0，1	0	椭圆插补的方向，0 表示逆时针方向，1 表示顺时针方向。
CenPos	中心位置	ARRAY[0..1]OFLREAL	负数，0，正数	0	椭圆的中心位置，增量坐标，单位：Pulse
EndPos	终点位置	ARRAY[0..1]OFLREAL	负数，0，正数	0	终点位置，增量坐标，单位：Pulse
K	主轴角度	LREAl	0-360.0	0	主轴的角度，主轴相对于 X 轴正向的角度；长轴与 X 轴正向的夹角，用来确定主轴的方向。
Rcorp	短轴长轴比	LREAL	0，1	0	短轴与长轴的比，取值范围为[0，1]
RunCycle	循环次数	UINT	0，正数	0	椭圆运动的循环次数（即椭圆的圈数）；0 表示无限循环；
VelocityMode	速度模式	SMC_INT_VELMODE	0-3	SIGMOID	速度模式：梯形-0（TRAPEZOID）；S 形-1（SIGMOID）；四次方-3（QUADRATIC）
Velocity	速度	LREAL	0，正数	0	插补合成速度，Pulse/s
Acceleration	加速度	LREAL	0，正数	0	插补合成加速度，$Pulse/s^2$
Deceleration	减速度	LREAL	0，正数	0	插补合成减速度，$Pulse/s^2$

Jerk	加加速度	LREAL	负数，0，正数	90000000	加加速度，速度模式为 3 时需要设置此参数，该参数值不能为 0。

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
Vel	合速度	LREAL	0，正数	0	当前插补运动合成速度
Done	完成	BOOL	FALSE,TRUE	FALSE	True 表示插补模块已完成插补任务
Busy	运动中	BOOL	FALSE,TRUE	FALSE	True 表示插补运动正在进行之中。
CommandAborted	运动中断	BOOL	FALSE,TRUE	FALSE	当 Stop 有效时，该变量为 True，表示插补模块非正常中止。
CommandHalt	运动暂停	BOOL	FALSE,TRUE	FALSE	当 Halt 为 True 时，该变量为 True，表示插补模块处于暂停状态。
Error	错误	BOOL	FALSE,TRUE	FALSE	错误状态；False-没有错误；
ErrorID	错误代码	DINT	0，正数	0	错误代码

	布尔	位串				整数								实数		时刻、持续时间 日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Axis0	AXIS_REF_VIRTUAL_SM3																			
Axis1	AXIS_REF_VIRTUAL_SM3																			
Exec	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Halt	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

Stop	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
IpoCycle	-	-	-	√	-	-	-	-	-	-	-	-	-	–	-	-	-	-	–	-
ArcIpo_dir	-	-	-	-	-	-	-	-	-	-	√	-	-	–	-	-	-	-	–	–
CenPos	-	-	-	-	-	-	-	-	-	-	-	-	-	–	-	-	-	-	–	–
EndPos	-	-	-	-	-	-	-	-	-	-	-	-	-	–	-	-	-	-	–	–
K	-	-	-	-	-	-	-	-	-	-	-	-	-	–	√	-	-	-	–	–
Rcorp	-	-	-	-	-	-	-	-	-	-	-	-	-	–	√	-	-	-	–	–
RunCycle	-	-	-	-	-	-	√	-	-	-	-	-	-	–	-	-	-	-	–	–
VelocityMode	-	-	-	-	-	-	-	-	-	-	-	-	-	–	-	-	-	-	–	–
Velocity	-	-	-	-	-	-	-	-	-	-	-	-	-	–	√	-	-	-	–	–
Acceleration	-	-	-	-	-	-	-	-	-	-	-	-	-	–	√	-	-	-	–	–
Deceleration	-	-	-	-	-	-	-	-	-	-	-	-	-	–	√	-	-	-	–	–
Jerk	-	-	-	-	-	-	-	-	-	-	-	-	-	–	√	-	-	-	–	–
Vel	-	-	-	-	-	-	-	-	-	-	-	-	-	–	√	-	-	-	–	–
Done	√	-	-	-	-	-	-	-	-	-	-	-	-	–	-	-	-	-	–	–
Busy	√	-	-	-	-	-	-	-	-	-	-	-	-	–	-	-	-	-	–	–
CommandAborted	√	-	-	-	-	-	-	-	-	-	-	-	-	–	-	-	-	-	–	–
CommandHalt	√	-	-	-	-	-	-	-	-	-	-	-	-	–	-	-	-	-	–	–
Error	√	-	-	-	-	-	-	-	-	-	-	-	-	–	-	-	-	-	–	–
ErrorID	-	-	-	-	-	-	-	-	-	-	-	√	-	–	-	-	-	-	–	–

- **功能说明：**

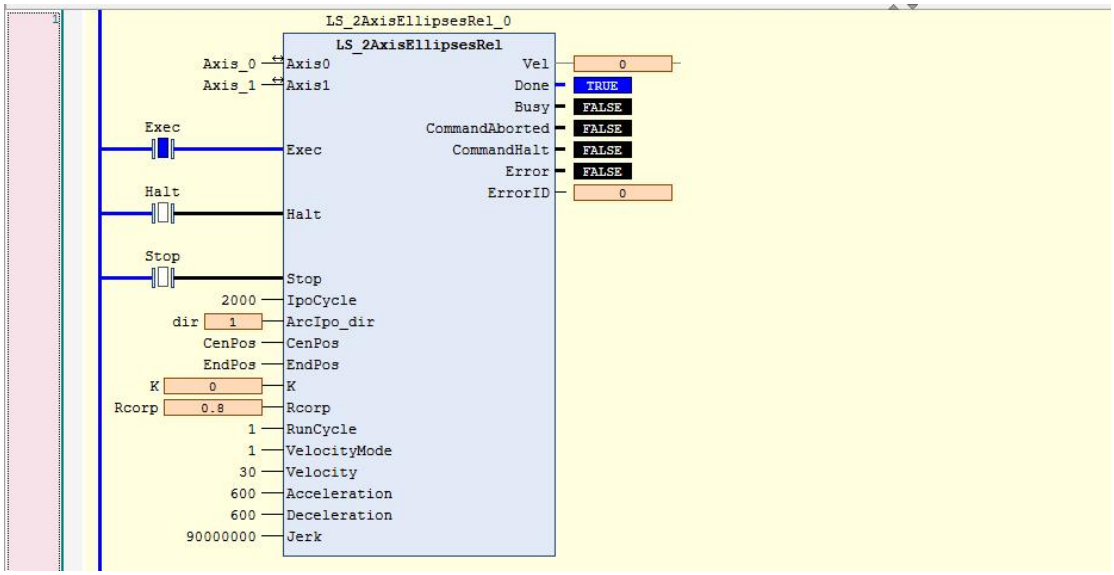
  - 指定的两个轴做椭圆插补运动，位置坐标为相对坐标。除相对坐标外，其他参数设置与绝对坐标两轴椭圆插补指令相同。

- **程序示例**

  ST：当 Exec 变为 True 时，执行两轴相对椭圆插补指令；程序如下：

```
LS_2AxisEllipsesRel_0(
    Axis0:= Axis_0,
    Axis1:= Axis_1,
    Exec:= Exec,
    Halt:= Halt,
    Stop:= Stop,
    IpoCycle:= 2000,
    ArcIpo_dir:= dir,
    CenPos:= CenPos,
    EndPos:= EndPos,
    K:= K,
    Rcorp:= Rcorp,
    RunCycle:= 1,
    VelocityMode:= 1,
    Velocity:= 30,
    Acceleration:= 600,
    Deceleration:= 600,
    Jerk:= jerk,
    Vel=> vel,
    Done=> done,
    Busy=> busy,
    CommandAborted=> Abort,
    CommandHalt=> ComHalt,
    Error=> err,
    ErrorID=> errid);
```

LD：当 Exec 变为 True 时，执行两轴相对椭圆插补指令；程序如下



- **注意事项：**

➢ 这个指令由“PMC_Ipolib”库实现。

➢ IpoCycle参数需要与运动指令所在的任务周期时间保持一致，否则指令执行时可能会出现Error报错。

➢ 参数IpoCycle、Jerk、Velocity、Acceleration、Deceleration不可以设置为0，否则指令执行时可能会出现Error报错。

➢ 在指令运行期间，运动轴都不能再被其它运动指令所调用。

➢ CenPos是椭圆插补运动的圆心坐标。CenPos，Endpos都是相对坐标。

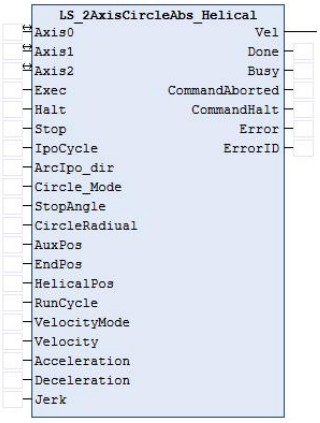
- Velocity、Acceleration、Deceleration、Jerk大小不能超过“轴设置-SoftMotion驱动：通用”里面的速度，加速度，减速度，加加速度的动态限制，否则指令执行时可能会出现Error报错。
- 插补轴需要先把轴上使能（运行MC_POWER指令），才能运行插补指令。
- 本指令始终保证两插补轴的合速度Vel等于设置的插补速度Velocity（30Pulse/s）。
- RunCycle是椭圆插补运动旋转的圈数，设置为0表示圆弧运动无限循环；设置为1表示从起始点运动到终点位置；设置为2,3,4..表示从起始点绕圆心旋转1,2.3..圈回到起始点位置，再从起始点运动到终点位置。
- 相对坐标简单理解：相对于起始位置的坐标，即将起始位置作为坐标原点建立的坐标系里的坐标。

7.10.22 LS_2AxisCircleAbs_Helical

指定的三个轴做空间螺旋线插补运动，位置坐标为绝对坐标。

- **指令格式**

指令	名称	FB/FC	LD表现	ST表现	库

LS_2Axis CircleAbs_ Helical	绝对三轴圆弧螺旋线插补	FB		LS_2AxisCircleAbs_Helical(Axis0:=, Axis1:=, Axis2:=, Exec:=, Halt:=, Stop:=, IpoCycle:=, ArcIpo_dir:=, Circle_Mode:=, StopAngle:=, CircleRadiual:=, AuxPos:=, EndPos:=, HelicalPos:=, RunCycle:=, VelocityMode:=, Velocity:=, Acceleration:=, Deceleration:=, Jerk:=, Vel=>, Done=>, Busy=>, CommandAborted=>, CommandHalt=>, Error=>, ErrorID=>);	PMC_IpoLib

- **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	初始值	描述
Axis0	轴0	AXIS_REF_VIRTUAL_SM3	-	-	插补0轴
Axis1	轴1	AXIS_REF_VIRTUAL_SM3	-	-	插补1轴
Axis2	轴2	AXIS_REF_VIRTUAL_SM3	-	-	插补2轴

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Exec	启动	BOOL	FALSE,TRUE	FALSE	启动命令，上升沿有效；

Halt	暂停	BOOL	FALSE,TRUE	FALSE	插补暂停命令，当该输入量为True时，当前插补运动暂停；当状态再次切换到False时，插补模块继续之前的未完成的插补任务。
Stop	停止	BOOL	FALSE,TRUE	FALSE	停止命令，该命令为True时，插补模块停止。
IpoCycle	插补周期	DWORD	-	2000	插补周期，单位：us，必须和所在任务周期一致；
ArcIpo_dir	插补方向	INT	0，1	0	圆弧插补的方向，0表示逆时针方向，1表示顺时针方向。圆弧插补模式1、2、3需要设置该值；
Circle_Mode	圆弧插补模式	INT	0-3	1	圆弧模式：0表示三点圆弧模式；1表示圆心与终点圆弧模式；2表示终点半径模式；3表示目标角度模式；
StopAngle	停止角度	LREAL	0-360.0	0	停止的目标角度，相对于起点的角度，单位为度；圆弧模式3需要设置该值。
CircleRadiual	圆弧半径	LREAL	负数，0 正数	0	圆弧半径，圆弧模式2需要设置该值。半径为负表示小圆弧；半径为正表示大圆弧；
AuxPos	辅助位置	ARRAY[0..1]OFLREAL	负数，0 正数	0	辅助点，单位：Pulse
EndPos	终点位置	ARRAY[0..1]OFLREAL	负数，0 正数	0	终点位置，单位：Pulse
HelicalPos	螺旋轴位置	LREAl	负数，0 正数	0	螺旋轴终点位置，单位：Pulse
RunCycle	循环次数	UINT	0，正数	0	圆弧运动的循环次数（即圆弧旋转的圈数）
VelocityMode	速度模式	SMC_INT_VELMODE	0-3	SIGMOID	速度模式：梯形-0（TRAPEZOID）；S形-1（SIGMOID）；四次方-3（QUADRATIC）

Velocity	速度	LREAL	0，正数	0	插补合成速度，Pulse/s
Acceleration	加速度	LREAL	0，正数	0	插补合成加速度，Pulse/s^2
Deceleration	减速度	LREAL	0，正数	0	插补合成减速度，Pulse/s^2
Jerk	加加速度	LREAL	负数，0 正数	90000000	加加速度，速度模式为3时需要设置此参数，该参数值不能为0。

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
Vel	合速度	LREAL	0，正数	0	当前插补运动合成速度
Done	完成	BOOL	FALSE,TRUE	FALSE	True表示插补模块已完成插补任务
Busy	运动中	BOOL	FALSE,TRUE	FALSE	True表示插补运动正在进行之中。
CommandAborted	运动中断	BOOL	FALSE,TRUE	FALSE	当Stop有效时，该变量为True，表示插补模块非正常中止。
CommandHalt	运动暂停	BOOL	FALSE,TRUE	FALSE	当Halt为True时，该变量为True，表示插补模块处于暂停状态。
Error	错误	BOOL	FALSE,TRUE	FALSE	错误状态；False-没有错误；
ErrorID	错误代码	DINT	0，正数	0	错误代码

	布尔	位串				整数								实数		时刻、持续时间、日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Axis0	AXIS_REF_VIRTUAL_SM3																			
Axis1	AXIS_REF_VIRTUAL_SM3																			
Axis2	AXIS_REF_VIRTUAL_SM3																			
Exec	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

Halt	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Stop	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
IpoCycle	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
ArcIpo _dir	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-
Circle _Mode	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-
StopAngle	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-
Circle Radiual	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-
AuxPos	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
EndPos	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
HelicalPos	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-
RunCycle	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-
Velocity Mode	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Velocity	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-
Acce leration	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-
Dece leration	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-
Jerk	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-
Vel	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-
Done	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Busy	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
CommandAborted	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

CommandHalt	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Error	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
ErrorID	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-

- **功能说明：**

  - 指定的三个轴做空间螺旋线插补运动，位置坐标为绝对坐标。该指令可理解为轴0与轴1做圆弧插补，轴2在垂直方向做跟随运动。

  - 螺旋插补的轨迹，如下图所示：

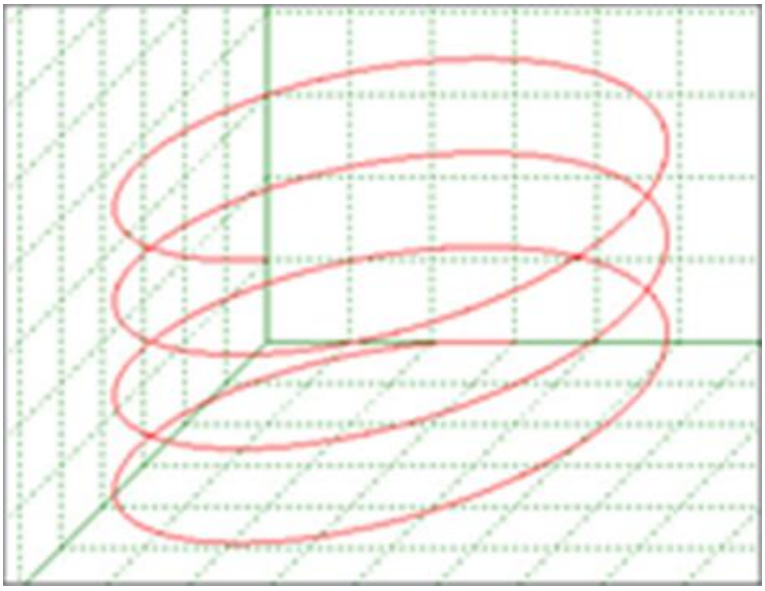


图1.6螺旋插补轨迹补

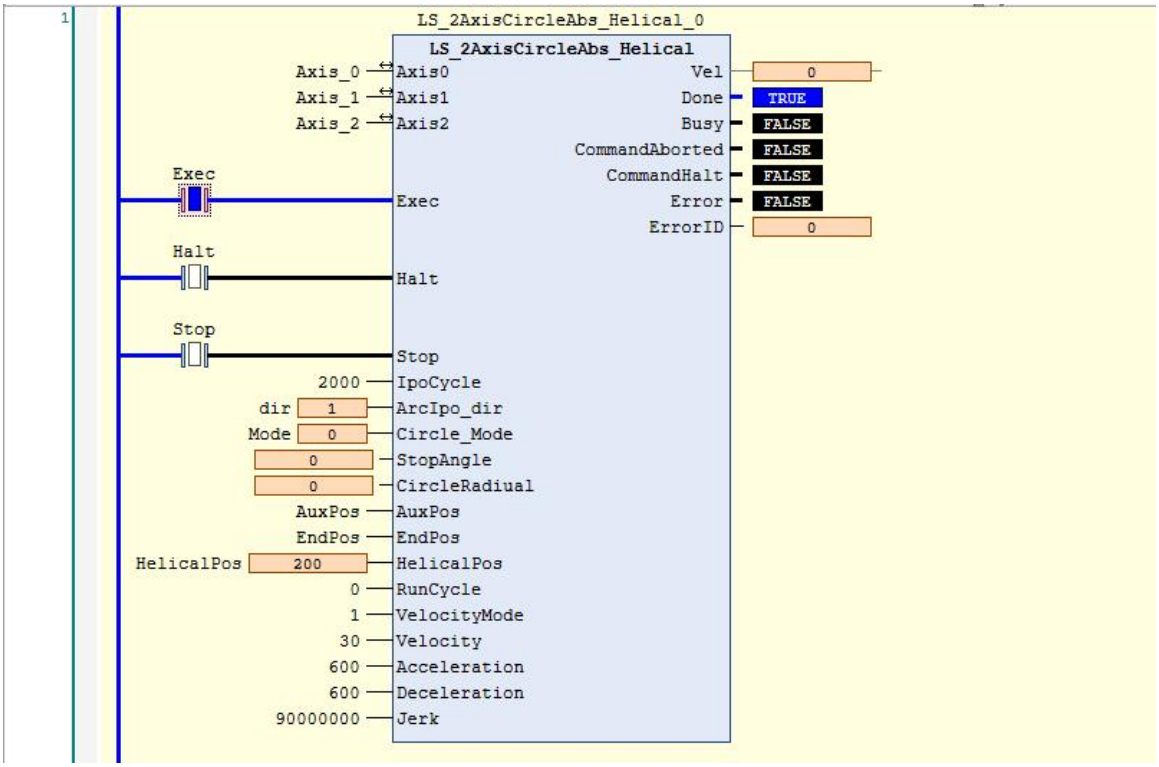
  - 在螺旋插补运动之中，轴0和轴1用于实现圆弧运动，“Circle_Mode”参数用来设置轴0、轴1的圆弧插补的模式，定义与两轴圆弧插补指令相同。

- **程序示例**

ST：当 Exec 变为 True 时，执行绝对三轴圆弧螺旋线插补指令；程序如下：

```
LS_2AxisCircleAbs_Helical_0(
    Axis0:= Axis_0,
    Axis1:= Axis_1,
    Axis2:= Axis_2,
    Exec TRUE := Exec TRUE,
    Halt FALSE := Halt FALSE,
    Stop FALSE := Stop FALSE,
    IpoCycle 2000 := 2000,
    ArcIpo_dir 0 := dir 0,
    Circle_Mode 1 := Mode 1,
    StopAngle:= ,
    CircleRadiual:= ,
    AuxPos:= AuxPos,
    EndPos:= EndPos,
    HelicalPos 500 := HelicalPos 500,
    RunCycle 0 := 0,
    VelocityMode SIGMOID := 1,
    Velocity 30 := 30,
    Acceleration 600 := 600,
    Deceleration 600 := 600,
    Jerk 9E+07 := jerk 9E+07,
    Vel 0 => vel 0,
    Done TRUE => done TRUE,
    Busy FALSE => busy FALSE,
    CommandAborted FALSE => Abort FALSE,
    CommandHalt FALSE => ComHalt FALSE,
    Error FALSE => err FALSE,
    ErrorID 0 => errid 0 );
```

LD：当 Exec 变为 True 时，执行绝对三轴圆弧螺旋线插补指令；程序如下：



- **注意事项**

➢ 这个指令由“PMC_Ipolib”库实现。

➢ IpoCycle参数需要与运动指令所在的任务周期时间保持一致，否则指令执行时可能会出现

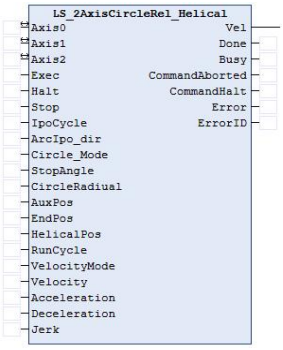
Error报错。

- 参数IpoCycle、Jerk、Velocity、Acceleration、Deceleration不可以设置为0，否则指令执行时可能会出现Error报错。
- 在指令运行期间，运动轴都不能再被其它运动指令所调用。
- Endpos是两个圆弧插补轴的终点位置（绝对坐标）；Endpos[0]是插补0轴终点位置，Endpos[1]是插补1轴终点位置；HelicalPos是螺旋轴的终点位置。
- Velocity、Acceleration、Deceleration、Jerk大小不能超过“轴设置-SoftMotion驱动：通用”里面的速度，加速度，减速度，加加速度的动态限制，否则指令执行时可能会出现Error报错。
- 插补轴需要先把轴上使能（运行MC_POWER指令），才能运行插补指令。
- 本指令始终保证两个插补轴的合速度Vel等于设置的插补速度Velocity（30Pulse/s）。

7.10.23 LS_2AxisCircleRel_Helical

指定的三个轴做空间螺旋线插补运动，位置坐标为相对坐标。

- **指令格式**

指令	名称	FB/FC	LD表现	ST表现	库
LS_2AxisCircleRel_Helical	相对三轴圆弧螺旋线插补	FB		LS_2AxisCircleRel_Helical(Axis0:=, Axis1:=, Axis2:=, Exec:=, Halt:=, Stop:=, IpoCycle:=, ArcIpo_dir:=, Circle_Mode:=, StopAngle:=, CircleRadiual:=,	PMC_IpoLib

				AuxPos:=, EndPos:=, HelicalPos:=, RunCycle:=, VelocityMode:=, Velocity:=, Acceleration:=, Deceleration:=, Jerk:=, Vel=>, Done=>, Busy=>,CommandAborted=>, CommandHalt=>, Error=>, ErrorID=>);	

- **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	初始值	描述
Axis0	轴0	AXIS_REF_VIRTUAL_SM3	-	-	插补0轴
Axis1	轴1	AXIS_REF_VIRTUAL_SM3	-	-	插补1轴
Axis2	轴2	AXIS_REF_VIRTUAL_SM3	-	-	插补2轴

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Exec	启动	BOOL	FALSE,TRUE	FALSE	启动命令，上升沿有效；
Halt	暂停	BOOL	FALSE,TRUE	FALSE	插补暂停命令，当该输入量为 True 时，当前插补运动暂停；当状态再次切换到 False 时，插补模块继续之前的未完成的插补任务。
Stop	停止	BOOL	FALSE,TRUE	FALSE	停止命令，该命令为 True 时，插补

					模块停止。
IpoCycle	插补周期	DWORD	-	2000	插补周期，单位：us，必须和所在任务周期一致；
ArcIpo_dir	插补方向	INT	0，1	0	圆弧插补的方向，0表示逆时针方向，1表示顺时针方向。圆弧插补模式1、2、3需要设置该值；
Circle_Mode	圆弧插补模式	INT	0-3	1	圆弧模式：0表示三点圆弧模式；1表示圆心与终点圆弧模式；2表示终点半径模式；3表示目标角度模式；
StopAngle	停止角度	LREAL	0-360.0	0	停止的目标角度，相对于起点的角度，单位为度；圆弧模式3需要设置该值。
CircleRadiual	圆弧半径	LREAL	负数，0 正数	0	圆弧半径，圆弧模式2需要设置该值。半径为负表示小圆弧；半径为正表示大圆弧；
AuxPos	辅助位置	ARRAY[0..1]OFLREAL	负数，0 正数	0	辅助点，增量坐标，单位：Pulse
EndPos	终点位置	ARRAY[0..1]OFLREAL	负数，0 正数	0	终点位置，增量坐标，单位：Pulse
HelicalPos	螺旋轴位置	LREAl	负数，0 正数	0	螺旋轴终点位置，增量坐标，单位：Pulse
RunCycle	循环次数	UINT	0，正数	0	圆弧运动的循环次数（即圆弧旋转的圈数）
VelocityMode	速度模式	SMC_INT_VELMODE	0-3	SIGMOID	速度模式：梯形-0（TRAPEZOID）；S 形-1

					（SIGMOID）；四次方-3（QUADRATIC）
Velocity	速度	LREAL	0，正数	0	插补合成速度，Pulse/s
Acceleration	加速度	LREAL	0，正数	0	插补合成加速度，Pulse/s^2
Deceleration	减速度	LREAL	0，正数	0	插补合成减速度，Pulse/s^2
Jerk	加加速度	LREAL	负数，0 正数	90000000	加加速度，速度模式为 3 时需要设置此参数，该参数值不能为 0。

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
Vel	合速度	LREAL	0，正数	0	当前插补运动合成速度
Done	完成	BOOL	FALSE,TRUE	FALSE	True 表示插补模块已完成插补任务
Busy	运动中	BOOL	FALSE,TRUE	FALSE	True 表示插补运动正在进行之中。
CommandAborted	运动中断	BOOL	FALSE,TRUE	FALSE	当 Stop 有效时，该变量为 True，表示插补模块非正常中止。
CommandHalt	运动暂停	BOOL	FALSE,TRUE	FALSE	当 Halt 为 True 时，该变量为 True，表示插补模块处于暂停状态。
Error	错误	BOOL	FALSE,TRUE	FALSE	错误状态；False-没有错误；
ErrorID	错误代码	DINT	0，正数	0	错误代码

	布尔	位串	整数	实数	时刻、持续时间、日期、字符串

	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Axis0	AXIS_REF_VIRTUAL_SM3																			
Axis1	AXIS_REF_VIRTUAL_SM3																			
Axis2	AXIS_REF_VIRTUAL_SM3																			
Exec	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Halt	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Stop	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
IpoCycle	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
ArcIpo_dir	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-
Circle_Mode	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-
Stop Angle	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-
Circle Radiual	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-
AuxPos	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
EndPos	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Helical Pos	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-
RunCycle	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-
VelocityMode	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Velocity	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-
Acceleration	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-
Deceleration	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-
Jerk	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-
Vel	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-
Done	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Busy	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
CommandAborted	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
CommandHalt	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

Error	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
ErrorID	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-

- **功能说明：**

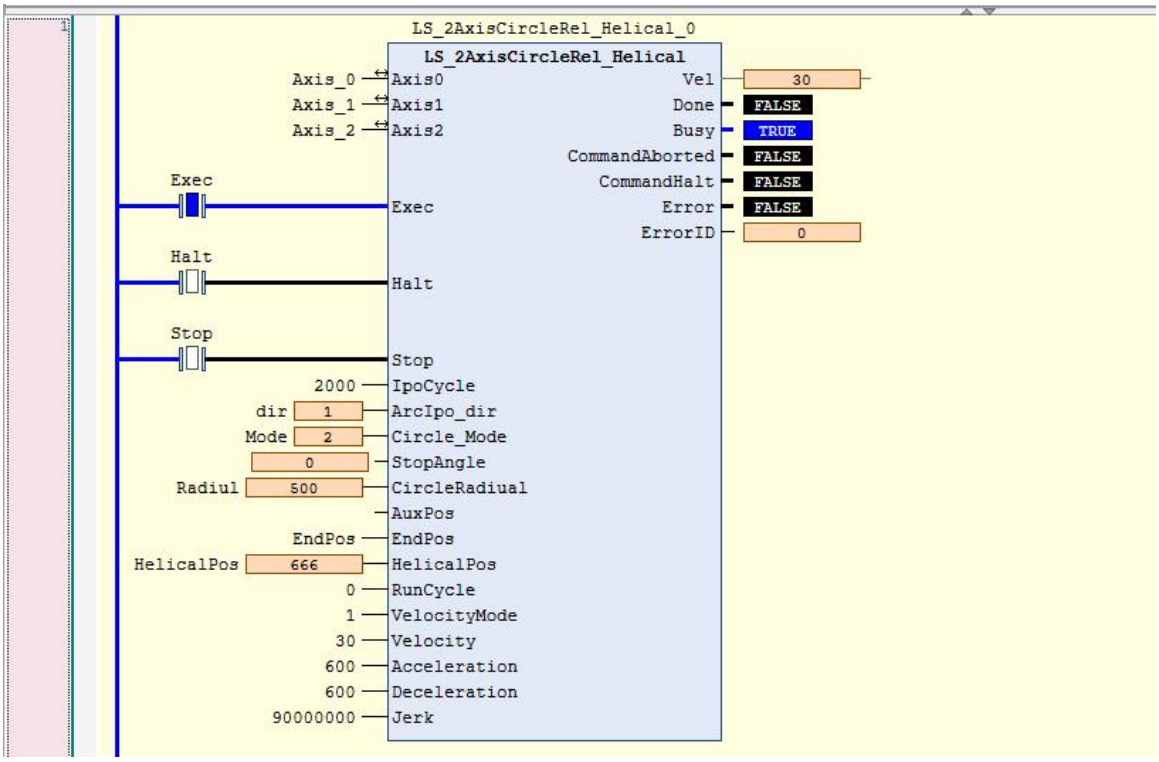
  - 指定的三个轴做空间螺旋线插补运动，位置坐标为相对坐标。
  - 除相对坐标外，其他参数设置与LS_2AxisCircleAbs_Helical指令相同。

- **程序示例**

ST：当Exec变为True时，执行相对三轴圆弧螺旋线插补指令；程序如下：

```
LS_2AxisCircleAbs_Helical_0(
    Axis0:= Axis_0,
    Axis1:= Axis_1,
    Axis2:= Axis_2,
    Exec TRUE := Exec TRUE,
    Halt FALSE := Halt FALSE,
    Stop FALSE := Stop FALSE,
    IpoCycle 2000 := 2000,
    ArcIpo_dir 0 := dir 0,
    Circle_Mode 3 := Mode 3,
    StopAngle 120 := Angle 120,
    CircleRadiual:= ,
    AuxPos:= AuxPos,
    EndPos:= ,
    HelicalPos 250 := HelicalPos 250,
    RunCycle 0 := 0,
    VelocityMode SIGMOID := 1,
    Velocity 30 := 30,
    Acceleration 600 := 600,
    Deceleration 600 := 600,
    Jerk 9E+07 := jerk 9E+07,
    Vel 0 => vel 0,
    Done TRUE => done TRUE,
    Busy FALSE => busy FALSE,
    CommandAborted FALSE => Abort FALSE,
    CommandHalt FALSE => ComHalt FALSE,
    Error FALSE => err FALSE,
    ErrorID 0 => errid 0);
```

LD：当Exec变为True时，执行相对三轴圆弧螺旋线插补指令；程序如下：



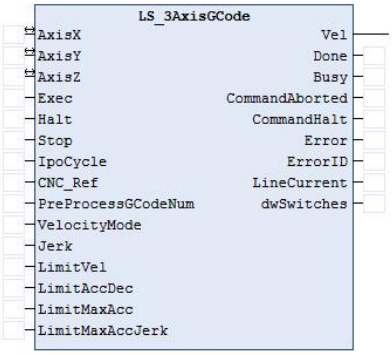
- **注意事项**

- 这个指令由“PMC_Ipolib”库实现。
- IpoCycle参数需要与运动指令所在的任务周期时间保持一致，否则指令执行时可能会出现Error报错。
- 参数IpoCycle、Jerk、Velocity、Acceleration、Deceleration不可以设置为0，否则指令执行时可能会出现Error报错。
- 在指令运行期间，运动轴都不能再被其它运动指令所调用。
- Velocity、Acceleration、Deceleration、Jerk大小不能超过“轴设置-SoftMotion驱动：通用”里面的速度，加速度，减速度，加加速度的动态限制，否则指令执行时可能会出现Error报错。
- 插补轴需要先把轴上使能（运行MC_POWER指令），才能运行插补指令。
- 本指令始终保证两个圆弧插补轴的合速度Vel等于设置的插补速度Velocity（30Pulse/s）。
- 相对坐标简单理解：相对于起始位置的坐标，即将起始位置作为坐标原点建立的坐标系里的坐标。

7.10.24 LS_3AxisGCode

本指令用于 G 代码三轴连续插补运动，支持坐标轴 X、Y、Z。（G 代码形式）

- **指令格式**

指令	名称	FB/FC	LD 表现	ST 表现	库
LS_3AxisGCode	三轴G代码连续插补	FB		LS_3AxisGCode(AxisX:=, AxisY:=, AxisZ:=, Exec:=, Halt:=, Stop:=, IpoCycle:=, CNC_Ref:=, PreProcessGCodeNum:=, VelocityMode:=, Jerk:=, LimitVel:=, LimitAccDec:=,LimitMaxAcc:=, LimitMaxAccJerk:=, Vel=>, Done=>, Busy=>, CommandAborted=>, CommandHalt=>, Error=>, ErrorID=>, LineCurrent=>, dwSwitches=>);	PMC_IpoLib

- **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	初始值	描述
AxisX	G代码X轴	AXIS_REF_VIRTUAL_SM3	-	-	对应G代码X轴

AxisY	G代码Y轴	AXIS_REF_VIRTUAL_SM3	-	-	对应G代码Y轴
AxisZ	G代码Z轴	AXIS_REF_VIRTUAL_SM3	-	-	对应G代码Z轴

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Exec	启动	BOOL	TRUE-FALSE	FALSE	启动命令，上升沿有效；
Halt	暂停	BOOL	TRUE-FALSE	FALSE	插补暂停命令，当该输入量为 True 时，当前插补运动暂停；当状态再次切换到 False 时，插补模块继续之前的未完成的插补任务。
Stop	停止	BOOL	TRUE-FALSE	FALSE	停止命令，该命令为 True 时，插补模块停止。
IpoCycle	插补周期	DWORD	-	2000	插补周期，单位：us，必须和所在任务周期一致；
CNC_Ref	G 文件指针	POINTERTO SMC_CNC_REF	-	-	G 代码文件指针
PreProcessGCodeNum	每周期段数	INT	正数	1	每周期读取 G 代码段数
VelocityMode	速度模式	SMC_INT_VELMODE	0-3	SIGMOID	速度模式：梯形-0（TRAPEZOID）；S 形-1（SIGMOID）；四次方-3（QUADRATIC）
Jerk	加加速度	LREAL	正数	90000000	加加速度，速度模式为 3 时需要设置此参数，该参数值不能为 0。
LimitVel	最大速度	LREAL	正数	100	各轴允许的最大速度
LimitAccDec	最大加速度	LREAL	正数	1000	各轴允许的最大加减速度
LimitMaxAcc	最大圆弧加速度	LREAL	正数	9000000	过渡圆弧允许的最大加速度
LimitMaxAccJerk	最大圆弧加加速度	LREAL	正数	90000000	过渡圆弧允许的最大加加速度

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
Vel	合速度	LREAL	0，正数	0	当前插补运动合成速度
Done	完成	BOOL	TRUE-FALSE	FALSE	True 表示插补模块已完成插补任务
Busy	运动中	BOOL	TRUE-FALSE	FALSE	True 表示插补运动正在进行之中。
CommandAborted	运动中断	BOOL	TRUE-FALSE	FALSE	当 Stop 有效时，该变量为 True，表示插补模块非正常中止。
CommandHalt	运动暂停	BOOL	TRUE-FALSE	FALSE	当 Halt 为 True 时，该变量为 True，表示插补模块处于暂停状态。
Error	错误	BOOL	TRUE-FALSE	FALSE	错误状态；False-没有错误；
ErrorID	错误代码	DINT	0，正数	0	错误代码
LineCurrent	当前执行的行号	DINT	-	0	当前执行的行号
dwSwitches	当前H代码对应的开关状态	DWORD	-	0	当前H代码对应的开关状态

	布尔	位串				整数								实数		时刻、持续时间、日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
AxisX	AXIS_REF_VIRTUAL_SM3																			
AxisY	AXIS_REF_VIRTUAL_SM3																			
AxisZ	AXIS_REF_VIRTUAL_SM3																			
Exec	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Halt	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Stop	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
IpoCycle	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
CNC_Ref	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

PreProcessGCodeNum	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-
Velocity Mode	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Jerk	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-
LimitVel	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-
Limit AccDec	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-
Limi tMaxAcc	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-
Limit MaxAccJerk	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-
Vel	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-
Done	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Busy	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Command Aborted	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Command Halt	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Error	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
ErrorID	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-
LineCurrent	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-
dwSwitches	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

- **功能说明：**

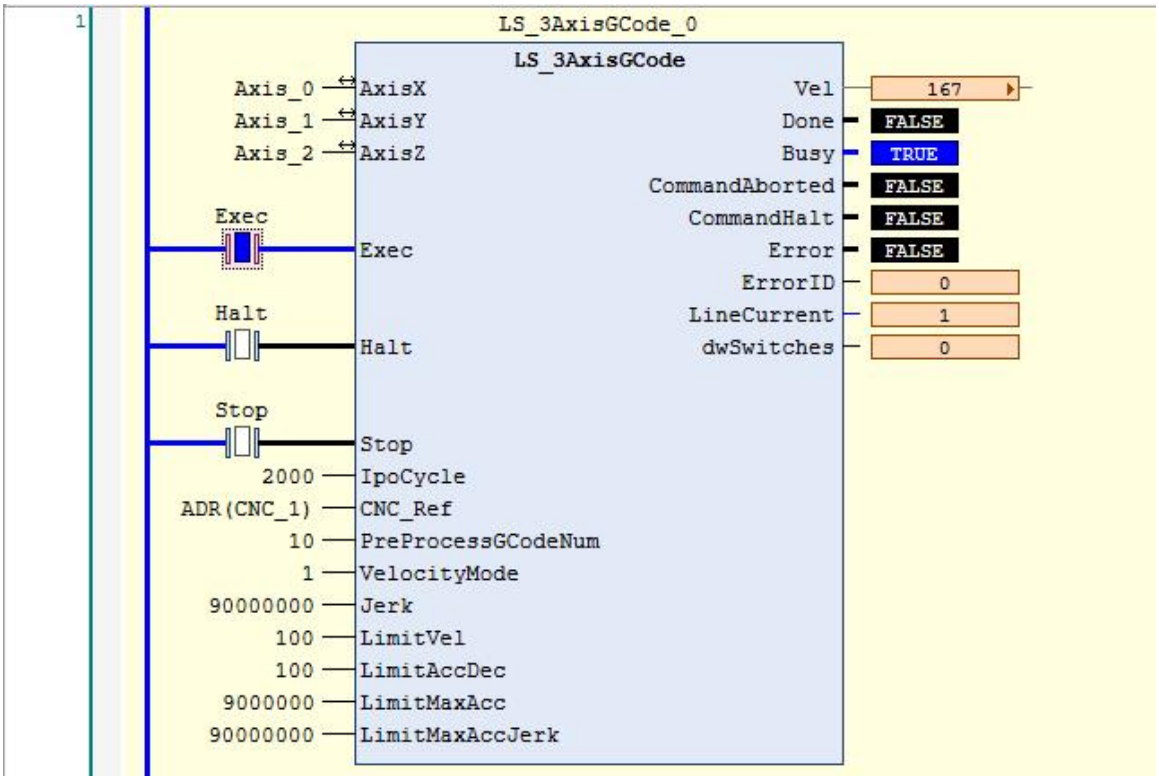
  - 本指令用于G代码三轴连续插补运动，支持坐标轴X、Y、Z。（G代码形式）

- **程序示例**

ST：当Exec变为True时，执行G代码三轴连续插补指令（G代码形式）；程序如下：

```
LS_3AxisGCode_0(
    AxisX:= Axis_0,
    AxisY:= Axis_1,
    AxisZ:= Axis_2,
    Exec TRUE := Exec TRUE,
    Halt FALSE := Halt FALSE,
    Stop FALSE := Stop FALSE,
    IpoCycle 2000 := 2000,
    CNC_Ref 0000026603A57 := ADR(CNC_1),
    PreProcessGCodeNum 10 := 10,
    VelocityMode SIGMOID := 1,
    Jerk 9E+07 := 90000000,
    LimitVel 200 := 200,
    LimitAccDec 200 := 200,
    LimitMaxAcc 9E+06 := 9000000,
    LimitMaxAccJerk 9E+07 := 90000000,
    Vel 0 => Vel 0,
    Done TRUE => Done TRUE,
    Busy FALSE => Busy FALSE,
    CommandAborted FALSE => CommandAborted FALSE,
    CommandHalt FALSE => CommandHalt FALSE,
    Error FALSE => err FALSE,
    ErrorID 0 => errid 0,
    LineCurrent -1 => LineCurrent -1,
    dwSwitches 0 => dwSwitches 0 );
```

LD：当Exec变为True时，执行G代码三轴连续插补指令（G代码形式）；程序如下：



- **注意事项**

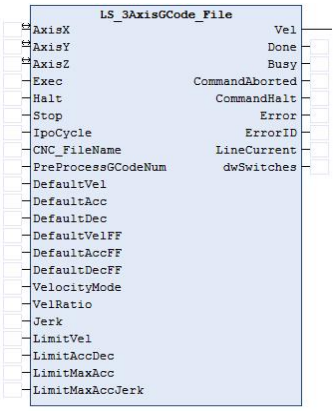
➢ 这个指令由“PMC_Ipolib”库实现。

- IpoCycle参数需要与运动指令所在的任务周期时间保持一致，否则指令执行时可能会出现Error报错。
- 参数IpoCycle、Jerk、PreProcessGCodeNum、LimitVel、LimitAccDec、LimitMaxAcc、LimitMaxAccJerk不可以设置为0。
- 在指令运行期间，运动轴都不能再被其它运动指令所调用。
- 插补轴需要先把轴上使能（运行MC_POWER指令），才能运行插补指令。

7.10.25 LS_3AxisGCode_File

本指令用于 G 代码三轴连续插补运动，支持坐标轴 X、Y、Z。（G 代码文件形式）

- **指令格式**

指令	名称	FB/FC	LD表现	ST表现	库
LS_3AxisGCode_File	三轴 G 代码连续插补-文件	FB		LS_3AxisGCode_File(AxisX:=, AxisY:=, AxisZ:=, Exec:=, Halt:=, Stop:=, IpoCycle:=, CNC_FileName:=, PreProcessGCodeNum:=, DefaultVel:=, DefaultAcc:=, DefaultDec:=, DefaultVelFF:=, DefaultAccFF:=, DefaultDecFF:=, VelocityMode:=, VelRatio:=, Jerk:=, LimitVel:=, LimitAccDec:=,	PMC_IpoLib

				LimitMaxAcc:=, LimitMaxAccJerk:=, Vel=>, Done=>, Busy=>, CommandAborted=>, CommandHalt=>, Error=>, ErrorID=>, LineCurrent=>, dwSwitches=>);	

- **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	初始值	描述
AxisX	G代码X轴	AXIS_REF_VIRTUAL_SM3	-	-	对应G代码X轴
AxisY	G代码Y轴	AXIS_REF_VIRTUAL_SM3	-	-	对应G代码Y轴
AxisZ	G代码Z轴	AXIS_REF_VIRTUAL_SM3	-	-	对应G代码Z轴

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Exec	启动	BOOL	TRUE-FALSE	FALSE	启动命令，上升沿有效；
Halt	暂停	BOOL	TRUE-FALSE	FALSE	插补暂停命令，当该输入量为 True 时，当前插补运动暂停；当状态再次切换到 False 时，插补模块继续之前的未完成的插补任务。
Stop	停止	BOOL	TRUE-FALSE	FALSE	停止命令，该命令为 True 时，插补模块停止。
IpoCycle	插补周期	DWORD	-	2000	插补周期，单位：us，必须和所在任务周期一致；
CNC_FileName	G 代码文件名	STRING	-	-	G 代码文件名称

PreProcessGCodeNum	每周期段数	INT	正数	1	每周期读取 G 代码段数
DefaultVel	默认速度	LREAL	正数	10	在CNC文件中，没有指定 F 的情况下，默认的 F 值
DefaultAcc	默认加速度	LREAL	正数	10	在CNC文件中，没有指定加速度的情况下，默认的加速度值
DefaultDec	默认减速度	LREAL	正数	10	在CNC文件中，没有指定减速度的情况下，默认的减速度值
DefaultVelFF	默认G01速度	LREAL	正数	20	在CNC文件中，没有指定G01的F情况下，默认的 F 值
DefaultAccFF	默认G01加速度	LREAL	正数	20	在CNC文件中，没有指定G01的加速度的情况下，默认的加速度值
DefaultDecFF	默G0减速度	LREAL	正数	20	在CNC文件中，没有指定G01的减速度的情况下，默认的减速度值
VelocityMode	速度模式	SMC_INT_VELMODE	0-3	SIGMOID	速度模式：梯形-0（TRAPEZOID）；S 形-1（SIGMOID）；四次方-3（QUADRATIC）
VelRatio	速度倍率	LREAL	0.01-2	1	速度倍率，最小值0.01，最大值2；
Jerk	加加速度	LREAL	正数	90000000	加加速度，速度模式为 3 时需要设置此参数，该参数值不能为 0。
LimitVel	最大速度	LREAL	正数	100	各轴允许的最大速度
Limit AccDec	最大加速度	LREAL	正数	1000	各轴允许的最大加减速度
Limit MaxAcc	最大圆弧加速度	LREAL	正数	9000000	过渡圆弧允许的最大加速度
Limit MaxAccJerk	最大圆弧加加速度	LREAL	正数	90000000	过渡圆弧允许的最大加加速度

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
Vel	合速度	LREAL	0，正数	0	当前插补运动合成速度
Done	完成	BOOL	TRUE-FALSE	FALSE	True 表示插补模块已完成插补任务
Busy	运动中	BOOL	TRUE-FALSE	FALSE	True 表示插补运动正在进行之中。
Command Aborted	运动中断	BOOL	TRUE-FALSE	FALSE	当 Stop 有效时，该变量为 True，表示插补模块非正常中止。
CommandHalt	运动暂停	BOOL	TRUE-FALSE	FALSE	当 Halt 为 True 时，该变量为 True，表示插补模块处于暂停状态。
Error	错误	BOOL	TRUE-FALSE	FALSE	错误状态；False-没有错误；
ErrorID	错误代码	DINT	0，正数	0	错误代码
Line Current	当前执行的行号	DINT	-	0	当前执行的行号
Dw Switches	当前H代码对应的开关状态	DWORD	-	0	当前H代码对应的开关状态

	布尔	位串				整数								实数		时刻、持续时间、日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
AxisX	AXIS_REF_VIRTUAL_SM3																			
AxisY	AXIS_REF_VIRTUAL_SM3																			
AxisZ	AXIS_REF_VIRTUAL_SM3																			
Exec	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Halt	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Stop	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
IpoCycle	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
CNC_FileNam	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√

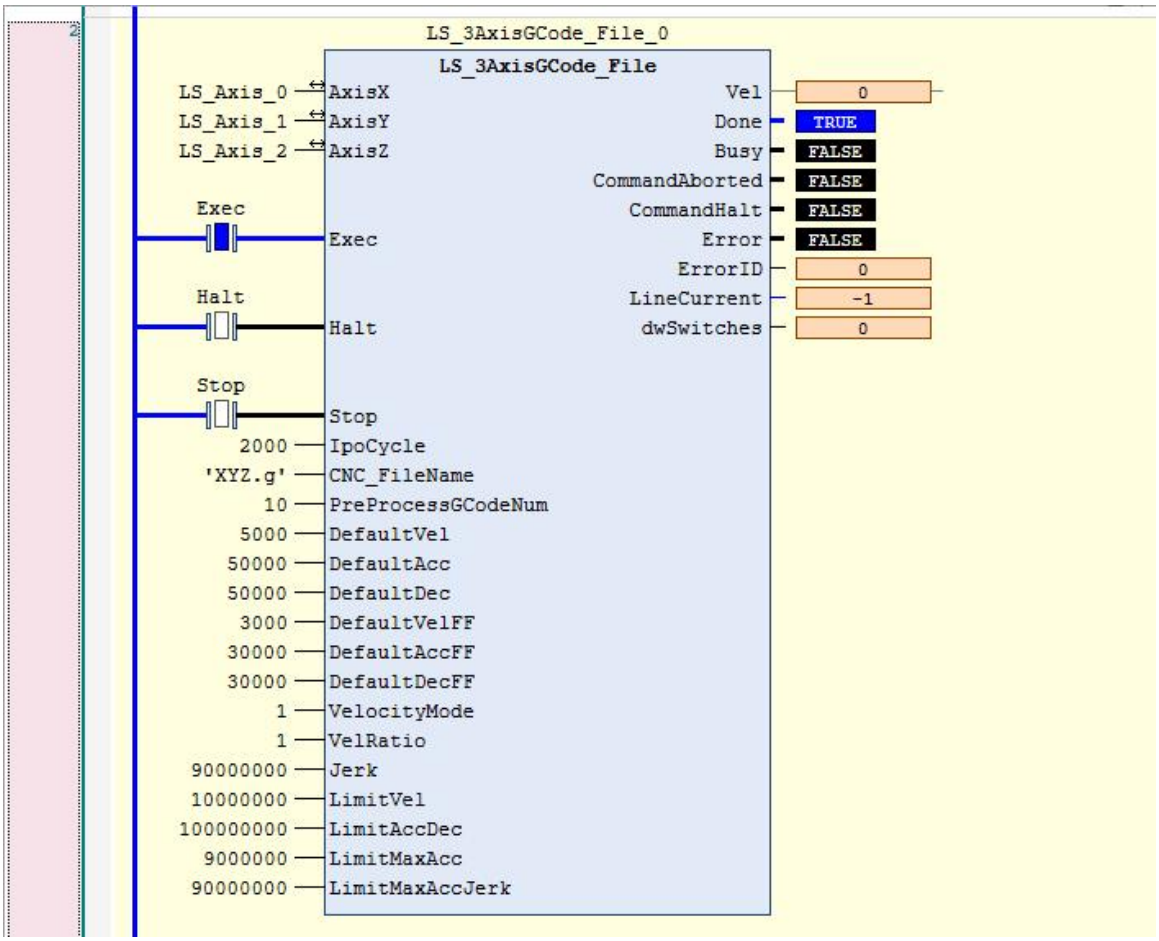
e																				
PreProcessGCodeNum	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-
DefaultVel	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-
DefaultAcc	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-
DefaultDec	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-
Default VelFF	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-
Default AccFF	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-
Default DecFF	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-
Velocity Mode	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
VelRatio	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-
Jerk	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-
LimitVel	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-
Limit AccDec	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-
Limit MaxAcc	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-
Limit MaxAccJerk	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-
Vel	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-
Done	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Busy	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Command Aborted	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Command Halt	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Error	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
ErrorID	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-
Line Current	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-
Dw Switches	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

- **功能说明：**

  - 本指令用于G代码三轴连续插补运动，支持坐标轴X、Y、Z。（G代码文件形式）

- **程序示例**

ST：当Exec变为True时，执行G代码三轴连续插补指令（G代码文件形式）；程序如下：



LD：当Exec变为True时，执行G代码三轴连续插补指令（G代码文件形式）；程序如下：

```
LS_3AxisGCode_File_0(
    AxisX:= LS_Axis_0,
    AxisY:= LS_Axis_1,
    AxisZ:= LS_Axis_2,
    Exec TRUE := Exec TRUE,
    Halt FALSE := Halt FALSE,
    Stop FALSE := Stop FALSE,
    IpoCycle 2000 := 2000,
    CNC_FileName 'ABC.g' := 'ABC.g',
    PreProcessGCodeNum 10 := 10,
    DefaultVel 5E+05 := 500000,
    DefaultAcc 5E+06 := 5000000,
    DefaultDec 5E+06 := 5000000,
    DefaultVelFF 2E+05 := 200000,
    DefaultAccFF 2E+06 := 2000000,
    DefaultDecFF 2E+06 := 2000000,
    VelocityMode SIGMOID := 1,
    VelRatio 1 := 1,
    Jerk 9E+07 := 90000000,
    LimitVel 5E+04 := 50000,
    LimitAccDec 5E+05 := 500000,
    LimitMaxAcc 9E+06 := 9000000,
    LimitMaxAccJerk 9E+07 := 90000000,
    Vel 853 => Vel 848,
    Done FALSE => Done FALSE,
    Busy TRUE => Busy TRUE,
    CommandAborted FALSE => CommandAborted FALSE,
    CommandHalt FALSE => CommandHalt FALSE,
    Error FALSE => err FALSE,
    ErrorID 0 => errid 0,
    LineCurrent 2 => LineCurrent 2,
    dwSwitches 0 => dwSwitches 0);
```

- **注意事项**

➢ 这个指令由“PMC_Ipolib”库实现。

➢ IpoCycle参数需要与运动指令所在的任务周期时间保持一致，否则指令执行时可能会出现Error报错。

➢ 参数IpoCycle、Jerk、PreProcessGCodeNum、LimitVel、LimitAccDec、LimitMaxAcc、LimitMaxAccJerk不可以设置为0。

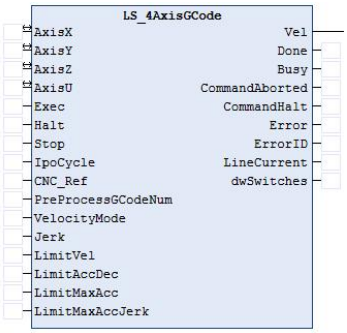
➢ 在指令运行期间，运动轴都不能再被其它运动指令所调用。

➢ 插补轴需要先把轴上使能（运行MC_POWER指令），才能运行插补指令。

7.10.26 LS_4AxisGCode

本指令用于 G 代码四轴连续插补运动，支持坐标轴 X、Y、Z、U。（G 代码形式）

- **指令格式**

指令	名称	FB/FC	LD表现	ST表现	库
LS_4AxisGCode	四轴 G 代码连续插补	FB		LS_4AxisGCode(AxisX:=, AxisY:=, AxisZ:=, AxisU:=, Exec:=, Halt:=, Stop:=, IpoCycle:=, CNC_Ref:=, PreProcessGCodeNum:=, VelocityMode:=, Jerk:=, LimitVel:=, LimitAccDec:=, LimitMaxAcc:=, LimitMaxAccJerk:=, Vel=>, Done=>, Busy=>, CommandAborted=>, CommandHalt=>, Error=>, ErrorID=>, LineCurrent=>,dwSwitches=>);	PMC_IpoLib

- **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	初始值	描述
AxisX	G代码X轴	AXIS_REF_VIRTUAL_SM3	-	-	对应G代码X轴
AxisY	G代码Y轴	AXIS_REF_VIRTUAL_SM3	-	-	对应G代码Y轴
AxisZ	G代码Z轴	AXIS_REF_VIRTUAL_SM3	-	-	对应G代码Z轴
AxisU	G代码U轴	AXIS_REF_VIRTUAL_SM3	-	-	对应G代码U

					轴

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Exec	启动	BOOL	TRUE-FALSE	FALSE	启动命令，上升沿有效；
Halt	暂停	BOOL	TRUE-FALSE	FALSE	插补暂停命令，当该输入量为 True 时，当前插补运动暂停；当状态再次切换到 False 时，插补模块继续之前的未完成的插补任务。
Stop	停止	BOOL	TRUE-FALSE	FALSE	停止命令，该命令为 True 时，插补模块停止。
IpoCycle	插补周期	DWORD	-	2000	插补周期，单位：us，必须和所在任务周期一致；
CNC_Ref	G 文件指针	POINTERTOSMC_CNC_REF	-	-	G 代码文件指针
PreProcessGCodeNum	每周期段数	INT	正数	1	每周期读取 G 代码段数
Velocity Mode	速度模式	SMC_INT_VELMODE	0-3	SIGMOID	速度模式：梯形-0（TRAPEZOID）；S 形-1（SIGMOID）；四次方-3（QUADRATIC）
Jerk	加加速度	LREAL	正数	90000000	加加速度，速度模式为 3 时需要设置此参数，该参数值不能为 0。
LimitVel	最大速度	LREAL	正数	100	各轴允许的最大速度
Limit AccDec	最大加速度	LREAL	正数	1000	各轴允许的最大加减速度
Limit MaxAcc	最大圆弧加速度	LREAL	正数	9000000	过渡圆弧允许的最大加速度
LimitMaxAccJerk	最大圆弧加加速度	LREAL	正数	90000000	过渡圆弧允许的最大加加速度

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
Vel	合速度	LREAL	0，正数	0	当前插补运动合成速度
Done	完成	BOOL	TRUE-FALSE	FALSE	True 表示插补模块已完成插补任务
Busy	运动中	BOOL	TRUE-FALSE	FALSE	True 表示插补运动正在进行之中。
CommandAborted	运动中断	BOOL	TRUE-FALSE	FALSE	当 Stop 有效时，该变量为 True，表示插补模块非正常中止。
ComandHalt	运动暂停	BOOL	TRUE-FALSE	FALSE	当 Halt 为 True 时，该变量为 True，表示插补模块处于暂停状态。
Error	错误	BOOL	TRUE-FALSE	FALSE	错误状态；False-没有错误；
Error ID	错误代码	DINT	0，正数	0	错误代码
LineCurrent	当前执行的行号	DINT	-	0	当前执行的行号
dwSwitches	当前H代码对应的开关状态	DWORD	-	0	当前H代码对应的开关状态

	布尔	位串				整数								实数		时刻、持续时间、日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
AxisX	AXIS_REF_VIRTUAL_SM3																			
AxisY	AXIS_REF_VIRTUAL_SM3																			
AxisZ	AXIS_REF_VIRTUAL_SM3																			
AxisU	AXIS_REF_VIRTUAL_SM3																			
Exec	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Halt	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Stop	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

IpoCycle	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
CNC_Ref	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
PreProcessGCodeNum	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-
VelocityMode	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Jerk	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-
LimitVel	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-
LimitAccDec	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-
LimitMaxAcc	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-
LimitMaxAccJerk	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-
Vel	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-
Done	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Busy	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
CommandAborted	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
CommandHalt	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Error	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
ErrorID	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-
Line Current	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-
Dw Switches	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

- **功能说明：**

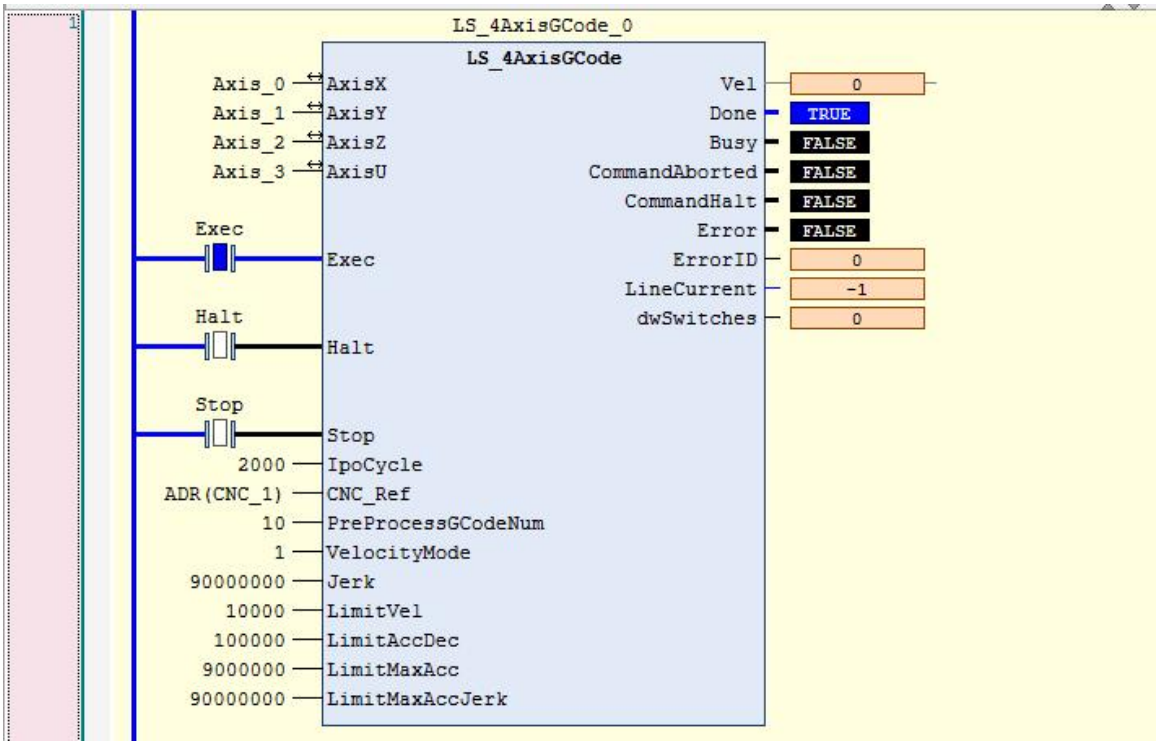
  - 本指令用于G代码四轴连续插补运动，支持坐标轴X、Y、Z、U。（G代码形式）

- **程序示例**

ST：当Exec变为True时，执行G代码四轴连续插补指令（G代码形式）；程序如下：

```
LS_4AxisGCode_0(
    AxisX:= Axis_0,
    AxisY:= Axis_1,
    AxisZ:= Axis_2,
    AxisU:= Axis_3,
    Exec TRUE := Exec TRUE,
    Halt FALSE := Halt FALSE,
    Stop FALSE := Stop FALSE,
    IpoCycle 2000 := 2000,
    CNC_Ref 0000026603A581ce ADR(CNC_2),
    PreProcessGCodeNum 10 := 10,
    VelocityMode SIGMOID := 1,
    Jerk 9E+07 := 90000000,
    LimitVel 200 := 200,
    LimitAccDec 200 := 200,
    LimitMaxAcc 9E+06 := 9000000,
    LimitMaxAccJerk 9E+07 := 90000000,
    Vel 1E+03 => Vel 1E+03,
    Done FALSE => Done FALSE,
    Busy TRUE => Busy TRUE,
    CommandAborted FALSE => CommandAborted FALSE,
    CommandHalt FALSE => CommandHalt FALSE,
    Error FALSE => err FALSE,
    ErrorID 0 => errid 0,
    LineCurrent 2 => LineCurrent 2,
    dwSwitches 0 => dwSwitches 0);
```

LD：当Exec变为True时，执行G代码四轴连续插补指令（G代码形式）；程序如下：



- **注意事项**

➢ 这个指令由“PMC_Ipolib”库实现。

➢ IpoCycle参数需要与运动指令所在的任务周期时间保持一致，否则指令执行时可能会出现

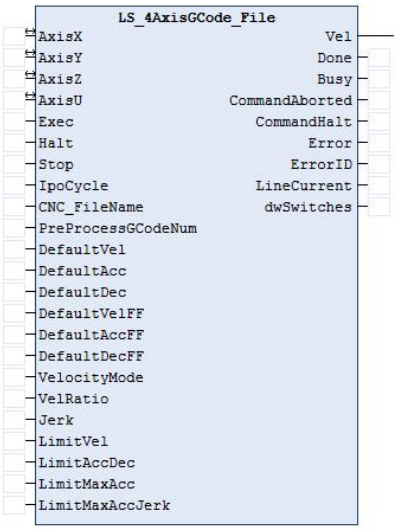
Error报错。

- 参数IpoCycle、Jerk、PreProcessGCodeNum、LimitVel、LimitAccDec、LimitMaxAcc、LimitMaxAccJerk不可以设置为0。
- 在指令运行期间，运动轴都不能再被其它运动指令所调用。
- 插补轴需要先把轴上使能（运行MC_POWER指令），才能运行插补指令。

7.10.27 LS_4AxisGCode_File

本指令用于 G 代码四轴连续插补运动，支持坐标轴 X、Y、Z、U。（G 代码文件形式）

- **指令格式**

指令	名称	FB/FC	LD表现	ST表现	库
LS_4AxisGCode_File	四轴 G 代码连续插补-文件	FB		LS_4AxisGCode_File(AxisX:=, AxisY:=, AxisZ:=, AxisU:=, Exec:=, Halt:=, Stop:=, IpoCycle:=, CNC_FileName:=, PreProcessCodeNum:=, DefaultVel:=, DefaultAcc:=, DefaultDec:=,DefaultVelFF:=, DefaultAccFF:=, DefaultDecFF:=, VelocityMode:=, VelRatio:=, Jerk:=, LimitVel:=, LimitAccDec:=, LimitMaxAcc:=,	PMC_IpoLib

				LimitMaxAccJerk:=, Vel=>, Done=>, Busy=>, CommandAborted=>, CommandHalt=>, Error=>, ErrorID=>, LineCurrent=>, dwSwitches=>);	

- **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	初始值	描述
AxisX	G代码X轴	AXIS_REF_VIRTUAL_SM3	-	-	对应G代码X轴
AxisY	G代码Y轴	AXIS_REF_VIRTUAL_SM3	-	-	对应G代码Y轴
AxisZ	G代码Z轴	AXIS_REF_VIRTUAL_SM3	-	-	对应G代码Z轴
AxisU	G代码U轴	AXIS_REF_VIRTUAL_SM3	-	-	对应G代码U轴

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Exec	启动	BOOL	TRUE-FALSE	FALSE	启动命令，上升沿有效；
Halt	暂停	BOOL	TRUE-FALSE	FALSE	插补暂停命令，当该输入量为 True 时，当前插补运动暂停；当状态再次切换到 False 时，插补模块继续之前的未完成的插补任务。
Stop	停止	BOOL	TRUE-FALSE	FALSE	停止命令，该命令为 True 时，插补模块停止。
IpoCycle	插补周期	DWORD	-	2000	插补周期，单位：us，必须和所在任务周期

					一致；
CNC_FileName	G 代码文件名	STRING	-	-	G 代码文件名称
PreProcessGCodeNum	每周期段数	INT	正数	1	每周期读取 G 代码段数
DefaultVel	默认速度	LREAL	正数	10	在CNC文件中，没有指定 F 的情况下，默认的 F 值
DefaultAcc	默认加速度	LREAL	正数	10	在CNC文件中，没有指定加速度的情况下，默认的加速度值
DefaultDec	默认减速度	LREAL	正数	10	在CNC文件中，没有指定减速度的情况下，默认的减速度值
DefaultVelFF	默认G01速度	LREAL	正数	20	在CNC文件中，没有指定G01的F情况下，默认的 F 值
DefaultAccFF	默认G01加速度	LREAL	正数	20	在CNC文件中，没有指定G01的加速度的情况下，默认的加速度值
DefaultDecFF	默认G01减速度	LREAL	正数	20	在CNC文件中，没有指定G01的减速度的情况下，默认的减速度值
VelocityMode	速度模式	SMC_INT_VELMODE	0-3	SIGMOID	速度模式：梯形-0（TRAPEZOID）；S形-1（SIGMOID）；四次方-3（QUADRATIC）
VelRatio	速度倍率	LREAL	0.01-2	1	速度倍率，最小值0.01，最大值2；
Jerk	加加速度	LREAL	正数	90000000	加加速度，速度模式为3 时需要设置此参数，该参数值不能为 0。
LimitVel	最大速度	LREAL	正数	100	各轴允许的最大速度
Limit AccDec	最大加速度	LREAL	正数	1000	各轴允许的最大加减速度
Limit MaxAcc	最大圆弧加速	LREAL	正数	9000000	过渡圆弧允许的最大加速度

	度				
LimitMaxAccJerk	最大圆弧加加速度	LREAL	正数	90000000	过渡圆弧允许的最大加加速度

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
Vel	合速度	LREAL	0，正数	0	当前插补运动合成速度
Done	完成	BOOL	TRUE-FALSE	FALSE	True 表示插补模块已完成插补任务
Busy	运动中	BOOL	TRUE-FALSE	FALSE	True 表示插补运动正在进行之中。
Commad Aborted	运动中断	BOOL	TRUE-FALSE	FALSE	当 Stop 有效时，该变量为 True，表示插补模块非正常中止。
CommandHalt	运动暂停	BOOL	TRUE-FALSE	FALSE	当 Halt 为 True 时，该变量为 True，表示插补模块处于暂停状态。
Error	错误	BOOL	TRUE-FALSE	FALSE	错误状态；False-没有错误；
ErrorID	错误代码	DINT	0，正数	0	错误代码
Line Current	当前执行的行号	DINT	-	0	当前执行的行号
Dw Switches	当前H代码对应的开关状态	DWORD	-	0	当前H代码对应的开关状态

	布尔	位串				整数								实数		时刻、持续时间、日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
AxisX	AXIS_REF_VIRTUAL_SM3																			
AxisY	AXIS_REF_VIRTUAL_SM3																			
AxisZ	AXIS_REF_VIRTUAL_SM3																			

AxisU	AXIS_REF_VIRTUAL_SM3																			
Exec	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Halt	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Stop	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
IpoCycle	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
CNC_FileName	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√
PreProcessGCodeNum	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-
DefaultVel	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-
DefaultAcc	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-
DefaultDec	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-
DefaultVelFF	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-
DefaultAccFF	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-
DefaultDecFF	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-
VelocityMode	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
VelRatio	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-
Jerk	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-
LimitVel	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-
LimitAccDec	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-
LimitMaxAcc	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-
LimitMaxAccJerk	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-
Vel	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-
Done	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Busy	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
CommandAborted	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
CommandHalt	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Error	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
ErrorID	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-
LineCurrent	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-
dwSwitches	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

- **功能说明：**

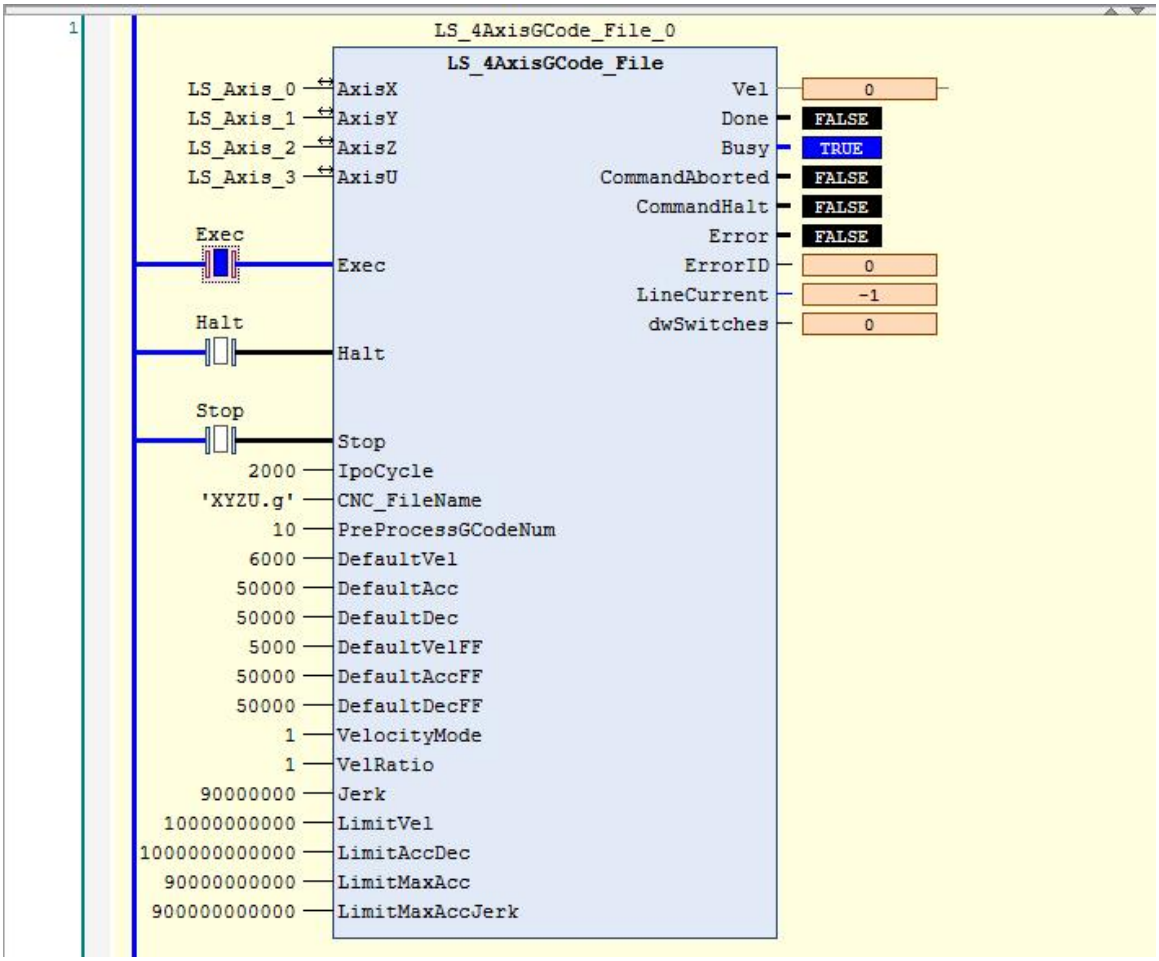
  - 本指令用于G代码四轴连续插补运动，支持坐标轴X、Y、Z、U。（G代码文件形式）

- **程序示例**

ST：当 Exec 变为 True 时，执行 G 代码四轴连续插补指令（G 代码文件形式）；程序如下：

```
LS_4AxisGCode_File_0(
    AxisX:= LS_Axis_0,
    AxisY:= LS_Axis_1,
    AxisZ:= LS_Axis_2,
    AxisU:= LS_Axis_3,
    Exec TRUE := Exec TRUE,
    Halt FALSE := Halt FALSE,
    Stop FALSE := Stop FALSE,
    IpoCycle 2000 := 2000,
    CNC_FileName 'XYZ.g' := 'XYZ.g',
    PreProcessGCodeNum 10 := 10,
    DefaultVel 5E+03 := 5000,
    DefaultAcc 5E+04 := 50000,
    DefaultDec 5E+04 := 50000,
    DefaultVelFF 2E+03 := 2000,
    DefaultAccFF 2E+04 := 20000,
    DefaultDecFF 2E+04 := 20000,
    VelocityMode SIGMOID := 1,
    VelRatio 1 := 1,
    Jerk 9E+10 := 90000000000,
    LimitVel 5E+09 := 5000000000,
    LimitAccDec 5E+10 := 50000000000,
    LimitMaxAcc 9E+11 := 900000000000,
    LimitMaxAccJerk 9E+12 := 9000000000000,
    Vel 0 => Vel 0,
    Done FALSE => Done FALSE,
    Busy TRUE => Busy TRUE,
    CommandAborted FALSE => CommandAborted FALSE,
    CommandHalt FALSE => CommandHalt FALSE,
    Error FALSE => err FALSE,
    ErrorID 0 => errid 0,
    LineCurrent -1 => LineCurrent -1,
    dwSwitches 0 => dwSwitches 0);
```

LD：当 Exec 变为 True 时，执行 G 代码四轴连续插补指令（G 代码文件形式）；程序如下：



- **注意事项**

➢ 这个指令由“PMC_Ipolib”库实现。

➢ IpoCycle参数需要与运动指令所在的任务周期时间保持一致，否则指令执行时可能会出现Error报错。

➢ 参数IpoCycle、Jerk、PreProcessGCodeNum、LimitVel、LimitAccDec、LimitMaxAcc、LimitMaxAccJerk不可以设置为0。

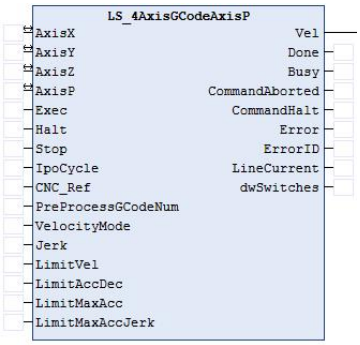
➢ 在指令运行期间，运动轴都不能再被其它运动指令所调用。

➢ 插补轴需要先把轴上使能（运行MC_POWER指令），才能运行插补指令。

7.10.28 LS_4AxisGCodeAxisP

本指令用于 G 代码四轴连续插补运动，支持坐标轴 X、Y、Z、P。（G 代码形式）

- **指令格式**

指令	名称	FB/FC	LD表现	ST表现	库
LS_4AxisGCodeAxisP	四轴 G 代码连续插补-P	FB		LS_4AxisGCodeAxisP (AxisX:=, AxisY:=, AxisZ:=, AxisP:=, Exec:=, Halt:=, Stop:=, IpoCycle:=, CNC_Ref:=, PreProcessGCodeNum:=, VelocityMode:=, Jerk:=, LimitVel:=,LimitAccDec:=, LimitMaxAcc:=, LimitMaxAccJerk:=, Vel=>, Done=>, Busy=>, CommandAborted=>, CommandHalt=>, Error=>, ErrorID=>, LineCurrent=>, dwSwitches=>);	PMC_IpoLib

- **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	初始值	描述
AxisX	G代码X轴	AXIS_REF_VIRTUAL_SM3	-	-	对应G代码X轴
AxisY	G代码Y轴	AXIS_REF_VIRTUAL_SM3	-	-	对应G代码Y轴

AxisZ	G代码Z轴	AXIS_REF_VIRTUAL_SM3	-	-	对应G代码Z轴
AxisP	G代码P轴	AXIS_REF_VIRTUAL_SM3	-	-	对应G代码P轴

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Exec	启动	BOOL	TRUE-FALSE	FALSE	启动命令，上升沿有效；
Halt	暂停	BOOL	TRUE-FALSE	FALSE	插补暂停命令，当该输入量为 True 时，当前插补运动暂停；当状态再次切换到 False 时，插补模块继续之前的未完成的插补任务。
Stop	停止	BOOL	TRUE-FALSE	FALSE	停止命令，该命令为 True 时，插补模块停止。
IpoCycle	插补周期	DWORD	-	2000	插补周期，单位：us，必须和所在任务周期一致；
CNC_Ref	G 文件指针	POINTERTOSMC_CNC_REF	-	-	G 代码文件指针
PreProcessGCodeNum	每周期段数	INT	正数	1	每周期读取 G 代码段数
VelocityMode	速度模式	SMC_INT_VELMODE	0-3	SIGMOID	速度模式：梯形-0（TRAPEZOID）；S 形-1（SIGMOID）；四次方-3（QUADRATIC）
Jerk	加加速度	LREAL	正数	90000000	加加速度，速度模式为 3 时需要设置此参数，该参数值不能为 0。

LimitVel	最大速度	LREAL	正数	100	各轴允许的最大速度
Limit AccDec	最大加速度	LREAL	正数	1000	各轴允许的最大加减速度
Limit MaxAcc	最大圆弧加速度	LREAL	正数	9000000	过渡圆弧允许的最大加速度
LimitMaxAccJerk	最大圆弧加加速度	LREAL	正数	90000000	过渡圆弧允许的最大加加速度

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
Vel	合速度	LREAL	0，正数	0	当前插补运动合成速度
Done	完成	BOOL	TRUE-FALSE	FALSE	True 表示插补模块已完成插补任务
Busy	运动中	BOOL	TRUE-FALSE	FALSE	True 表示插补运动正在进行之中。
CommandAborted	运动中断	BOOL	TRUE-FALSE	FALSE	当 Stop 有效时，该变量为 True，表示插补模块非正常中止。
Command Halt	运动暂停	BOOL	TRUE-FALSE	FALSE	当 Halt 为 True 时，该变量为 True，表示插补模块处于暂停状态。
Error	错误	BOOL	TRUE-FALSE	FALSE	错误状态；False-没有错误；
ErrorID	错误代码	DINT	0，正数	0	错误代码
Line Current	当前执行的行号	DINT	-	0	当前执行的行号
dw Switches	当前H代码对应的开关状态	DWORD	-	0	当前H代码对应的开关状态

	布	位串	整数	实数	时刻、持续时间、

	尔															日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
AxisX	AXIS_REF_VIRTUAL_SM3																			
AxisY	AXIS_REF_VIRTUAL_SM3																			
AxisZ	AXIS_REF_VIRTUAL_SM3																			
AxisP	AXIS_REF_VIRTUAL_SM3																			
Exec	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Halt	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Stop	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Ipo Cycle	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
CNC_Ref	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
PreProcessGCodeNum	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-
Velociy Mode	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Jerk	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-
LimitVel	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-
Limit AccDec	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-
LimitMaxAcc	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-
LimitMaxAccJerk	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-
Vel	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-
Done	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Busy	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Command Aborted	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Command Halt	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Error	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
ErrorID	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-
Line Current	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-
Dw	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

Switches																			

- **功能说明：**

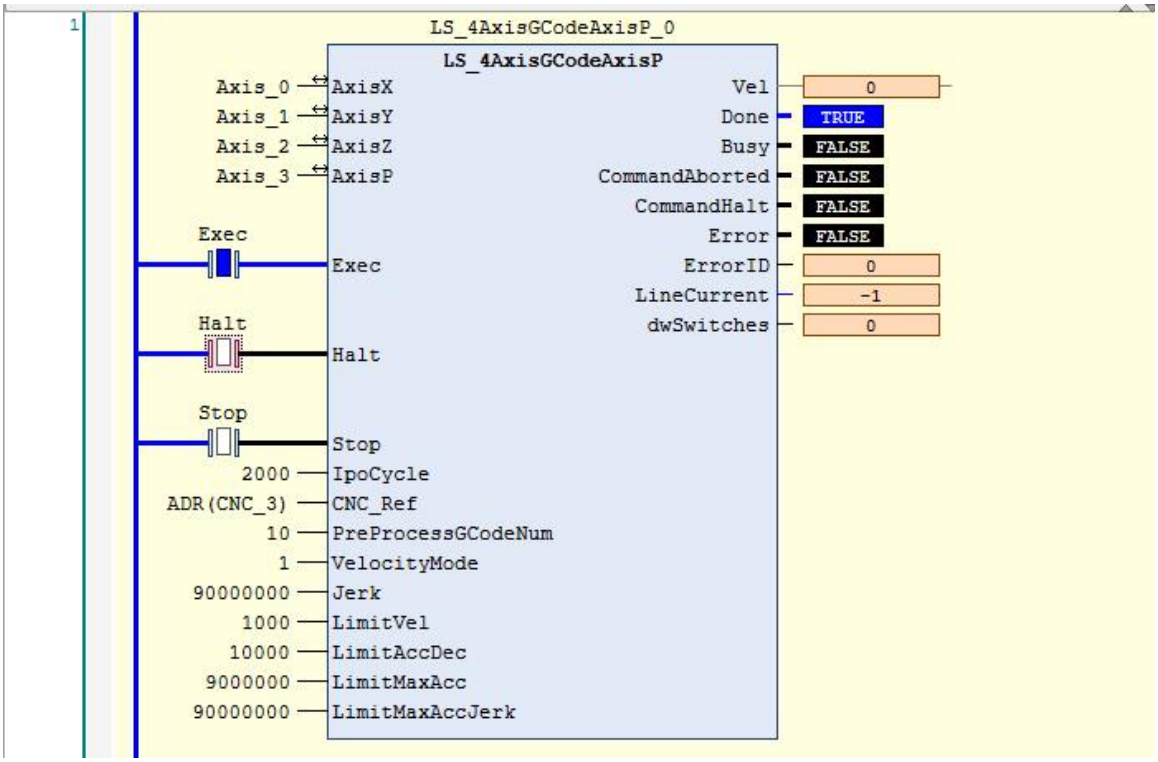
  - 本指令用于G代码四轴连续插补运动，支持坐标轴X、Y、Z、P。（G代码形式）

- **程序示例**

ST：当Exec变为True时，执行G代码四轴连续插补指令（G代码形式）；程序如下：

```
LS_4AxisGCodeAxisP_0(
    AxisX:= Axis_0,
    AxisY:= Axis_1,
    AxisZ:= Axis_2,
    AxisP:= Axis_3,
    Exec TRUE := Exec TRUE,
    Halt FALSE := Halt FALSE,
    Stop FALSE := Stop FALSE,
    IpoCycle 2000 := 2000,
    CNC_Ref 0000026603A58888 := ADR(CNC_3),
    PreProcessGCodeNum 10 := 10,
    VelocityMode SIGMOID := 1,
    Jerk 9E+07 := 90000000,
    LimitVel 200 := 200,
    LimitAccDec 200 := 200,
    LimitMaxAcc 9E+06 := 9000000,
    LimitMaxAccJerk 9E+07 := 90000000,
    Vel 158 => Vel 158,
    Done FALSE => Done FALSE,
    Busy TRUE => Busy TRUE,
    CommandAborted FALSE => CommandAborted FALSE,
    CommandHalt FALSE => CommandHalt FALSE,
    Error FALSE => err FALSE,
    ErrorID 0 => errid 0,
    LineCurrent 1 => LineCurrent 1,
    dwSwitches 0 => dwSwitches 0);
```

LD：当Exec变为True时，执行G代码四轴连续插补指令（G代码形式）；程序如下：



- **注意事项**

➢ 这个指令由“PMC_Ipolib”库实现。

➢ IpoCycle参数需要与运动指令所在的任务周期时间保持一致，否则指令执行时可能会出现Error报错。

➢ 参数IpoCycle、Jerk、PreProcessGCodeNum、LimitVel、LimitAccDec、LimitMaxAcc、LimitMaxAccJerk不可以设置为0。

➢ 在指令运行期间，运动轴都不能再被其它运动指令所调用。

➢ 插补轴需要先把轴上使能（运行MC_POWER指令），才能运行插补指令。

7.10.29 LS_4AxisGCodeAxisP_File

本指令用于 G 代码四轴连续插补运动，支持坐标轴 X、Y、Z、P。（G 代码文件形式）

- **指令格式**

指令	名称	FB/FC	LD表现	ST表现	库

LS_4AxisGCodeAxisP_File	四轴 G 代码连续插补-文件-P	FB		LS_4AxisGCodeAxisP_File(AxisX:=, AxisY:=, AxisZ:=, AxisP:=, Exec:=, Halt:=, Stop:=, IpoCycle:=, CNC_File Name:=, PreProcessGCodeNum:=, DefaultVel:=, DefaultAcc:=, DefaultDec:=, DefaultVelFF:=, DefaultAccFF:=, DefaultDecFF:=, VelocityMode:=, VelRatio:=, Jerk:=, LimitVel:=, Limit AccDec:=, LimitMaxAcc:=, LimitMaxAccJerk:=, Vel=>, Done=>, Busy=>, CommandAborted=>, CommandHalt=>, Error=>, ErrorID=>, LineCurrent=>, DwSwitches=>);	PMC_IpoLib

- **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	初始值	描述
AxisX	G代码X轴	AXIS_REF_VIRTUAL_SM3	-	-	对应G代码X轴
AxisY	G代码Y轴	AXIS_REF_VIRTUAL_SM3	-	-	对应G代码Y轴
AxisZ	G代码Z轴	AXIS_REF_VIRTUAL_SM3	-	-	对应G代码Z轴
AxisP	G代码P轴	AXIS_REF_VIRTUAL_SM3	-	-	对应G代码P轴

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Exec	启动	BOOL	TRUE-FALSE	FALSE	启动命令，上升沿有效；
Halt	暂停	BOOL	TRUE-FALSE	FALSE	插补暂停命令，当该输入量为 True 时，当前插补运动暂停；当状态再次切换到 False 时，插补模块继续之前的未完成的插补任务。
Stop	停止	BOOL	TRUE-FALSE	FALSE	停止命令，该命令为 True 时，插补模块停止。
IpoCycle	插补周期	DWORD	-	2000	插补周期，单位：us，必须和所在任务周期一致；
CNC_FileName	G 代码文件名	STRING	-	-	G 代码文件名称
PreProcessGCode Num	每周期段数	INT	正数	1	每周期读取 G 代码段数
DefaultVel	默认速度	LREAL	正数	10	在CNC文件中，没有指定 F 的情况下，默认的 F 值
DefaultAcc	默认加速度	LREAL	正数	10	在CNC文件中，没有指定加速度的情况下，默认的加速度值
DefaultDec	默认	LREAL	正数	10	在CNC文件中，没

	减速度				有指定减速度的情况下，默认的减速度值
DefaultVelFF	默认G01速度	LREAL	正数	20	在CNC文件中，没有指定G01的F情况下，默认的 F 值
DefaultAccFF	默认G01加速度	LREAL	正数	20	在CNC文件中，没有指定G01的加速度的情况下，默认的加速度值
DefaultDecFF	默认G01减速度	LREAL	正数	20	在CNC文件中，没有指定G01的减速度的情况下，默认的减速度值
VelocityMode	速度模式	SMC_INT_VELMODE	0-3	SIGMOID	速度模式：梯形-0（TRAPEZOID）；S 形-1（SIGMOID）；四次方-3（QUADRATIC）
VelRatio	速度倍率	LREAL	0.01-2	1	速度倍率，最小值0.01，最大值2；
Jerk	加加速度	LREAL	正数	90000000	加加速度，速度模式为 3 时需要设置此参数，该参数值不能为 0。
LimitVel	最大速度	LREAL	正数	100	各轴允许的最大速度
LimitAccDec	最大加速度	LREAL	正数	1000	各轴允许的最大加减速度
LimitMaxAcc	最大圆弧加速度	LREAL	正数	9000000	过渡圆弧允许的最大加速度
LimitMaxAccJerk	最大圆弧加加速度	LREAL	正数	90000000	过渡圆弧允许的最大加加速度

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述

Vel	合速度	LREAL	0，正数	0	当前插补运动合成速度
Done	完成	BOOL	TRUE-FALSE	FALSE	True 表示插补模块已完成插补任务
Busy	运动中	BOOL	TRUE-FALSE	FALSE	True 表示插补运动正在进行之中。
Command Aborted	运动中断	BOOL	TRUE-FALSE	FALSE	当 Stop 有效时，该变量为 True，表示插补模块非正常中止。
Command Halt	运动暂停	BOOL	TRUE-FALSE	FALSE	当 Halt 为 True 时，该变量为 True，表示插补模块处于暂停状态。
Error	错误	BOOL	TRUE-FALSE	FALSE	错误状态；False-没有错误；
ErrorID	错误代码	DINT	0，正数	0	错误代码
Line Current	当前执行的行号	DINT	-	0	当前执行的行号
dw Switches	当前H代码对应的开关状态	DWORD	-	0	当前H代码对应的开关状态

	布尔	位串				整数								实数		时刻、持续时间、日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
AxisX	AXIS_REF_VIRTUAL_SM3																			
AxisY	AXIS_REF_VIRTUAL_SM3																			
AxisZ	AXIS_REF_VIRTUAL_SM3																			
AxisP	AXIS_REF_VIRTUAL_SM3																			
Exec	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Halt	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Stop	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Ipo Cycle	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
CNC_FileNam	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√

e																				
PreProcessGCodeNum	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-
DefaultVel	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-
DefaultAcc	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-
DefaultDec	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-
Default VelFF	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-
Default AccFF	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-
Default DecFF	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-
Velocity Mode	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Vel Ratio	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-
Jerk	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-
LimitVel	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-
Limit AccDec	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-
Limit MaxAcc	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-
Limit MaxAccJerk	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-
Vel	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-
Done	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Busy	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Command Aborted	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Command Halt	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Error	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
ErrorID	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-
Line Current	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-
Dw Switches	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

- **功能说明：**

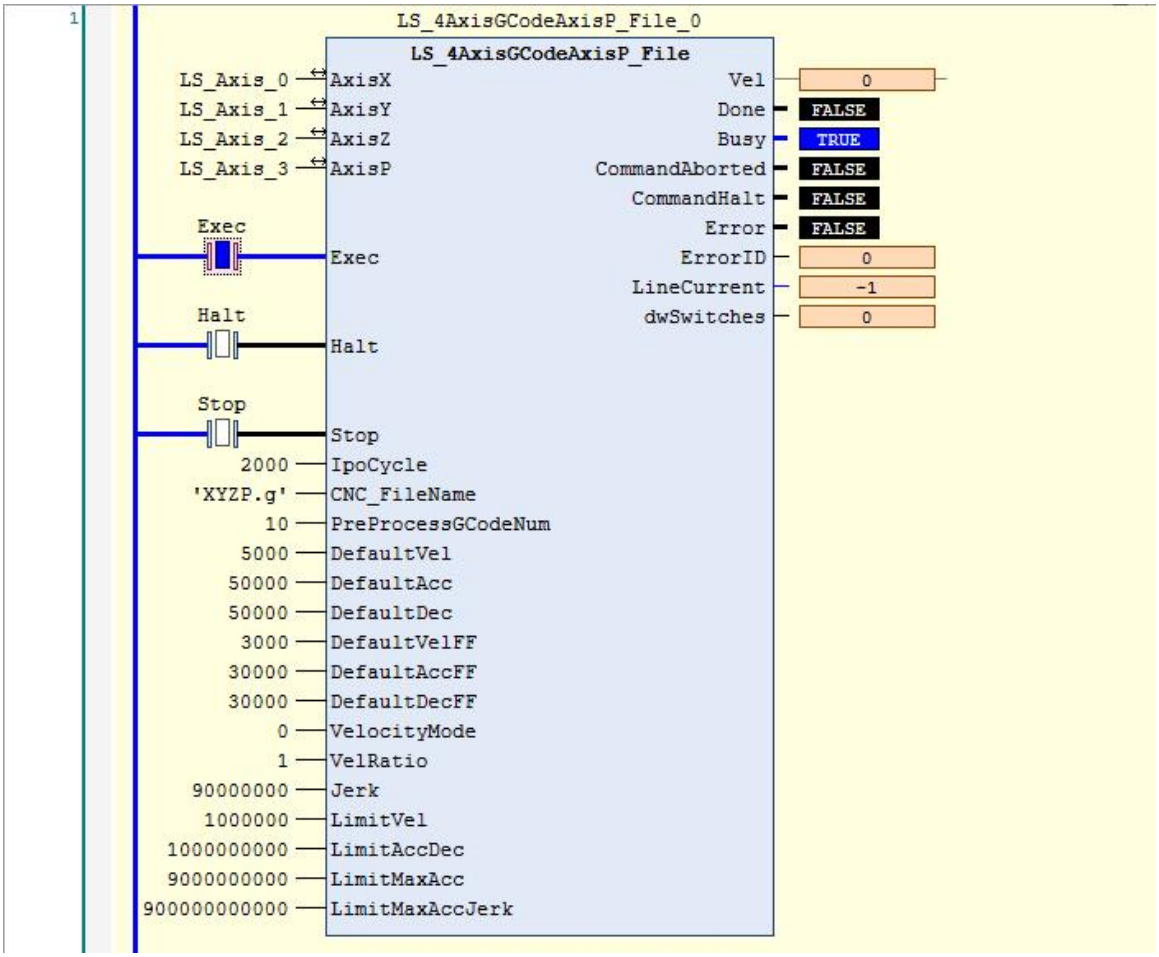
➢ 本指令用于G代码四轴连续插补运动，支持坐标轴X、Y、Z、P。（G代码文件形式）

- **程序示例**

ST：当Exec变为True时，执行G代码四轴连续插补指令（G代码文件形式）；程序如下：

```
LS_4AxisGCodeAxisP_File_0(
    AxisX:= LS_Axis_0,
    AxisY:= LS_Axis_1,
    AxisZ:= LS_Axis_2,
    AxisP:= LS_Axis_3,
    Exec TRUE := Exec TRUE,
    Halt FALSE := Halt FALSE,
    Stop FALSE := Stop FALSE,
    IpoCycle 2000 := 2000,
    CNC_FileName 'abc.g' := 'abc.g',
    PreProcessGCodeNum 10 := 10,
    DefaultVel 5E+03 := 5000,
    DefaultAcc 5E+04 := 50000,
    DefaultDec 5E+04 := 50000,
    DefaultVelFF 2E+03 := 2000,
    DefaultAccFF 2E+04 := 20000,
    DefaultDecFF 2E+04 := 20000,
    VelocityMode TRAPEZOID := 0,
    VelRatio 1 := 1,
    Jerk 9E+07 := 90000000,
    LimitVel 5E+03 := 5000,
    LimitAccDec 5E+04 := 50000,
    LimitMaxAcc 9E+06 := 9000000,
    LimitMaxAccJerk 9E+07 := 90000000,
    Vel 0 => Vel 0,
    Done FALSE => Done FALSE,
    Busy TRUE => Busy TRUE,
    CommandAborted FALSE => CommandAborted FALSE,
    CommandHalt FALSE => CommandHalt FALSE,
    Error FALSE => err FALSE,
    ErrorID 0 => errid 0,
    LineCurrent -1 => LineCurrent -1,
    dwSwitches 0 => dwSwitches 0);
```

LD：当Exec变为True时，执行G代码四轴连续插补指令（G代码文件形式）；程序如下：



- **注意事项**

➢ 这个指令由“PMC_Ipolib”库实现。

➢ IpoCycle参数需要与运动指令所在的任务周期时间保持一致，否则指令执行时可能会出现Error报错。

➢ 参数IpoCycle、Jerk、PreProcessGCodeNum、LimitVel、LimitAccDec、LimitMaxAcc、LimitMaxAccJerk不可以设置为0。

➢ 在指令运行期间，运动轴都不能再被其它运动指令所调用。

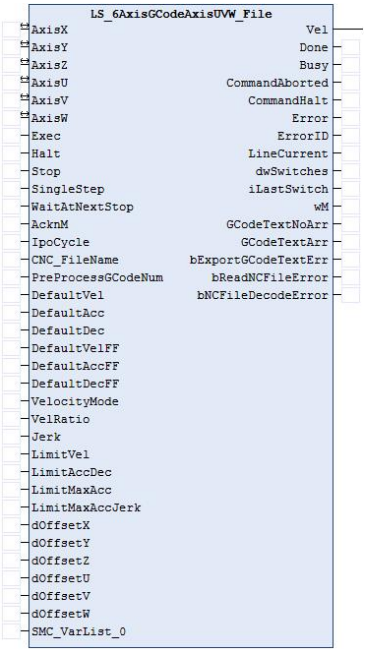
➢ 插补轴需要先把轴上使能（运行MC_POWER指令），才能运行插补指令。

7.10.30 LS_6AxisGCodeAxisUVW_File

本指令用于 G 代码六轴连续插补运动，支持坐标轴 X、Y、Z、U、V、W。（G 代码

文件形式）

- **指令格式**

指令	名称	FB/FC	LD表现	ST表现	库
LS_6AxisGCodeAxisUVW_File	六轴G代码连续插补-文件	FB		LS_6AxisGCodeAxisUVW_File(AxisX:=, AxisY:=, AxisZ:=, AxisU:=, AxisV:=, AxisW:=, Exec:=, Halt:=, Stop:=, SingleStep:=, WaitAtNextStop:=, AcknM:=, IpoCycle:=, CNC_File Name:=, PreProcessGCodeNum: =, DefaultVel:=, DefaultAcc:=, DefaultDec:=, DefaultVelFF:=, DefaultAccFF:=, DefaultDecFF:=, Velocity Mode:=, VelRatio:=, Jerk:=, LimitVel:=, LimitAcc Dec:=, LimitMax Acc:=, LimitMaxAccJerk:=, dOffsetX:=, dOffsetY:=,	PMC_IpoLib

				dOffsetZ:=, dOffsetU:=, dOffsetV:=, dOffsetW:=, SMC_Var List_0:=, Vel=>, Done=>, Busy=>, CommandAborted=>, CommandHalt=>, Error=>, ErrorID=>, LineCurrent=>, dwSwitches=>, iLastSwitch=>, wM=>, GCodeTextNoArr=>, GcodeText Arr=>, bExportGCodeTextErr =>, bReadNCFileError=>, bNCFileDecodeError= >);	

- **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	初始值	描述
AxisX	G代码X轴	AXIS_REF_VIRTUAL_SM3	-	-	对应G代码X轴
AxisY	G代码Y轴	AXIS_REF_VIRTUAL_SM3	-	-	对应G代码Y轴
AxisZ	G代码Z轴	AXIS_REF_VIRTUAL_SM3	-	-	对应G代码Z轴
AxisU	G代码U轴	AXIS_REF_VIRTUAL_SM3	-	-	对应G代码U轴
AxisV	G代码V轴	AXIS_REF_VIRTUAL_SM3	-	-	对应G代码V轴
AxisW	G代码W轴	AXIS_REF_VIRTUAL_SM3	-	-	对应G代码W轴

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Exec	启动	BOOL	TRUE-FALSE	FALSE	启动命令，上升沿有效；
Halt	暂停	BOOL	TRUE-FALSE	FALSE	插补暂停命令，当该输入量为 True 时，当前插补运动暂停；当状态再次切换到 False 时，插补模块继续之前的未完成的插补任务。
Stop	停止	BOOL	TRUE-FALSE	FALSE	停止命令，该命令为 True 时，插补模块停止。
Single Step	单步标志	BOOL	TRUE-FALSE	False	单步标志
WaitAt NextStop	单步启动	BOOL	TRUE-FALSE	False	单步启动
AcknM	Mcode 处理标志	BOOL	TRUE-FALSE	False	Mcode处理标志
IpoCycle	插补周期	DWORD	-	2000	插补周期，单位：us，必须和所在任务周期一致；
CNC_FileName	G 代码文件名	STRING	-	-	G 代码文件名称
PreProcessGCodeNum	每周期段数	INT	正数	1	每周期读取 G 代码段数
DefaultVel	默认速度	LREAL	正数	10	在CNC文件中，没有指定 F 的情况下，默认的 F 值
DefaultAcc	默认加速度	LREAL	正数	10	在CNC文件中，没有指定加速度的情况下，默认的加速度值
DefaultDec	默认减	LREAL	正数	10	在CNC文件中，没有指定减速

	速度				度的情况下，默认的减速度值
Default VelFF	默认G01速度	LREAL	正数	20	在CNC文件中，没有指定G01的F情况下，默认的 F 值
Default AccFF	默认G01加速度	LREAL	正数	20	在CNC文件中，没有指定G01的加速度的情况下，默认的加速度值
Default DecFF	默认G01减速度	LREAL	正数	20	在CNC文件中，没有指定G01的减速度的情况下，默认的减速度值
Velocity Mode	速度模式	SMC_INT_VEL MODE	0-3	SIGMOID	速度模式：梯形-0（TRAPEZOID）；S 形-1（SIGMOID）；四次方-3（QUADRATIC）
VelRatio	速度倍率	LREAL	0.01-2	1	速度倍率，最小值0.01，最大值2；
Jerk	加加速度	LREAL	正数	90000000	加加速度，速度模式为 3 时需要设置此参数，该参数值不能为 0。
LimitVel	最大速度	LREAL	正数	100	各轴允许的最大速度
Limit AccDec	最大加速度	LREAL	正数	1000	各轴允许的最大加减速度
Limit MaxAcc	最大圆弧加速度	LREAL	正数	9000000	过渡圆弧允许的最大加速度
LimitMax AccJerk	最大圆弧加加	LREAL	正数	90000000	过渡圆弧允许的最大加加速

	速度				度
dOffsetX	X 轴偏移值	LREAL	负数，0，正数	0	X轴零点坐标偏移值
dOffsetY	Y 轴偏移值	LREAL	负数，0，正数	0	Y轴零点坐标偏移值
dOffsetZ	Z 轴偏移值	LREAL	负数，0，正数	0	Z轴零点坐标偏移值
dOffsetU	U 轴偏移值	LREAL	负数，0，正数	0	U轴零点坐标偏移值
dOffsetV	V 轴偏移值	LREAL	负数，0，正数	0	V轴零点坐标偏移值
dOffsetW	W 轴偏移值	LREAL	负数，0，正数	0	W轴零点坐标偏移值
SMC_VarList_0	系统变量	SMC_VarList	-	-	DIN66025系统变量

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
Vel	合速度	LREAL	0，正数	0	当前插补运动合成速度
Done	完成	BOOL	TRUE-FALSE	FALSE	True 表示插补模块已完成插补任务
Busy	运动中	BOOL	TRUE-FALSE	FALSE	True 表示插补运动正在进行之中。
Command Aborted	运动中断	BOOL	TRUE-FALSE	FALSE	当 Stop 有效时，该变量为 True，表示插补模块非正常中止。
Command Halt	运动暂停	BOOL	TRUE-FALSE	FALSE	当 Halt 为 True 时，该变量为 True，表示插补模块处于暂停状态。
Error	错误	BOOL	TRUE-FALSE	FALSE	错误状态；False-没有错误；

ErrorID	错误代码	DINT	0，正数	0	错误代码
Line Current	当前执行的行号	DINT	-	0	当前执行的行号
Dw Switches	当前H代码对应的开关状态	DWORD	-	0	当前H代码对应的开关状态
iLastSwitch	-	INT	-	0	-
wM	当前M指令值	WORD	-	0	当前M指令值
GcodeText NoArr	暂存译码行号	ARRAY[0..2]OFDINT	-	-	暂存译码行号
Gcode TextArr	暂存译码内容	ARRAY[0..2]OFSMC_GCODE_TEXT	-	-	暂存译码内容
bExportodeTextErr	G文本报错	BOOL	TRUE-FALSE	FALSE	输出G代码文本出错
bReadNCFileError	变量名称不存在	BOOL	TRUE-FALSE	FALSE	不能取代DIN66025格式G代码文件中的变量，SMC_VarList中变量名称不存在
bNCFileecodeError	G文件不存在	BOOL	TRUE-FALSE	FALSE	加载的G代码文件不存在

	布尔	位串				整数								实数		时刻、持续时间、日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
AxisX	AXIS_REF_VIRTUAL_SM3																			
AxisY	AXIS_REF_VIRTUAL_SM3																			
AxisZ	AXIS_REF_VIRTUAL_SM3																			
AxisU	AXIS_REF_VIRTUAL_SM3																			

AxisV	AXIS_REF_VIRTUAL_SM3																			
AxisW	AXIS_REF_VIRTUAL_SM3																			
Exec	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Halt	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Stop	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
SingleStep	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Wait AtNext Stop	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
AcknM	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Ipo Cycle	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
CNC_FileNam e	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√
PreProcessGCo deNum	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-
DefaultVel	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-
DefaultAcc	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-
DefaultDec	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-
Default VelFF	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-
Default AccFF	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-
Default DecFF	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-
Velocity Mode	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
VelRatio	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-
Jerk	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-
LimitVel	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-
Limit AccDec	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-
Limit MaxAcc	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-
LimitMaxAccJ erk	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-
dOffsetX	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-
dOffsetY	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-

dOffsetZ	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-
dOffsetU	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-
dOffsetV	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-
dOffsetW	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-
SMC_VarList_0	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Vel	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-
Done	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Busy	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Command Aborted	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Command Halt	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Error	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
ErrorID	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-
Line Current	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-
dwSwitches	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
iLast Switch	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-
wM	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
GCodeTextNoArr	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Gcode TextArr	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
bExportGCodeTextErr	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
bReadNCFileError	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
bNCFileDecodeError	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

- **功能说明：**

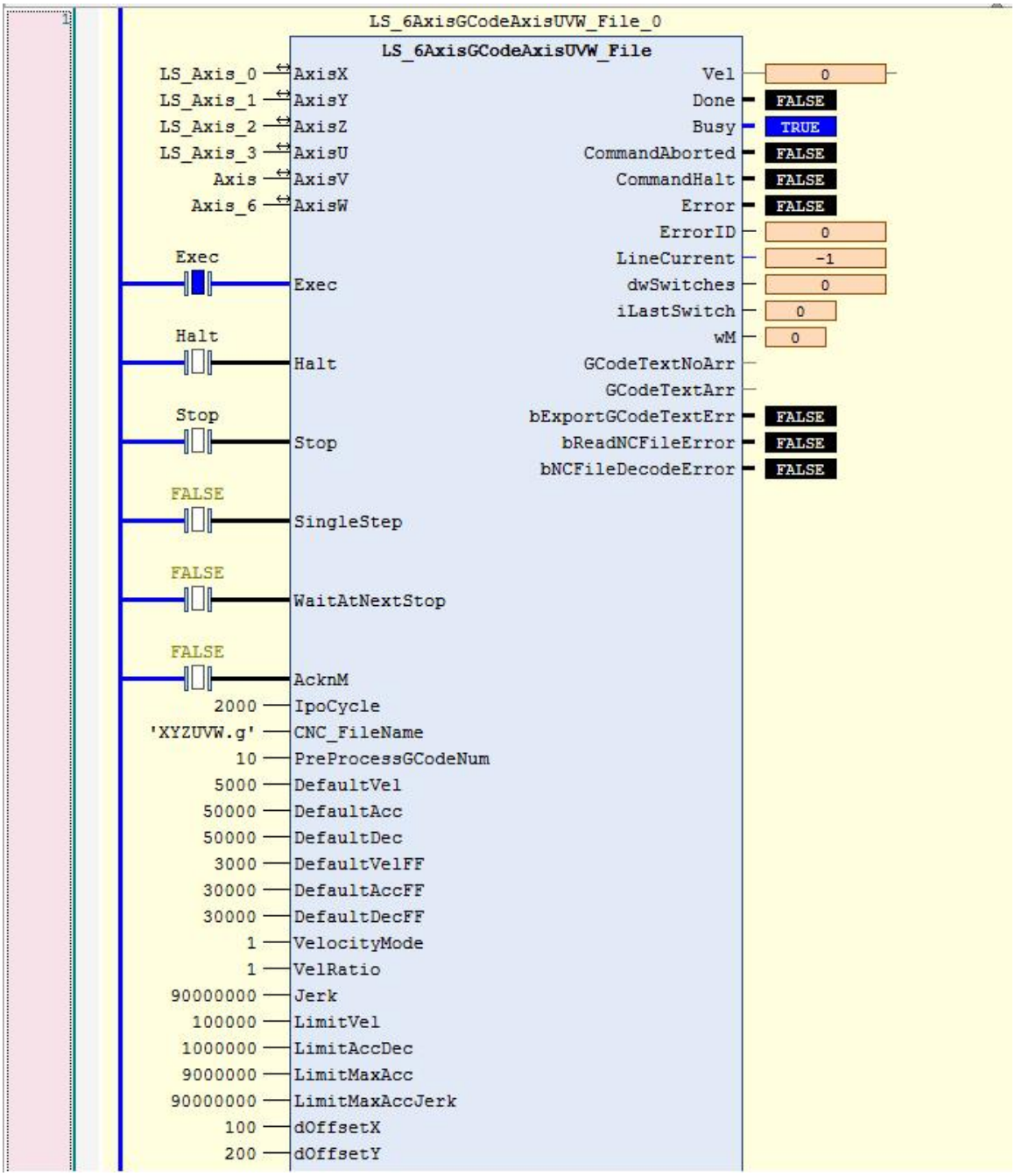
  - 本指令用于G代码六轴连续插补运动，支持坐标轴X、Y、Z、U、V、W。（G代码文件形式）

- **程序示例**

ST：当 Exec 变为 True 时，执行 G 代码六轴连续插补指令（G 代码文件形式）；程序如下：

```
LS_6AxisGCodeAxisUVW_File_0(
    AxisX:= LS_Axis_0,
    AxisY:= LS_Axis_1,
    AxisZ:= LS_Axis_2,
    AxisU:= LS_Axis_3,
    AxisV:= Axis,
    AxisW:= Axis_6,
    Exec TRUE := Exec TRUE,
    Halt FALSE := Halt FALSE,
    Stop FALSE := Stop FALSE,
    SingleStep FALSE := FALSE,
    WaitAtNextStop FALSE := FALSE,
    AcknM FALSE := FALSE,
    IpoCycle 2000 := 2000,
    CNC_FileName 'XYZUVW.g' := 'XYZUVW.g',
    PreProcessGCodeNum 10 := 10,
    DefaultVel 5E+04 := 50000,
    DefaultAcc 5E+05 := 500000,
    DefaultDec 5E+05 := 500000,
    DefaultVelFF 3E+04 := 30000,
    DefaultAccFF 3E+05 := 300000,
    DefaultDecFF 3E+05 := 300000,
    VelocityMode SIGMOID := 1,
    VelRatio 1 := 1,
    Jerk 9E+07 := 90000000,
    LimitVel 9E+06 := 9000000,
    LimitAccDec 9E+07 := 90000000,
    LimitMaxAcc 9E+06 := 9000000,
    LimitMaxAccJerk 9E+07 := 90000000,
    dOffsetX 10 := 10,
    dOffsetY 20 := 20,
    dOffsetZ 30 := 30,
    dOffsetU 40 := 40,
    dOffsetV 50 := 50,
    dOffsetW 60 := 60,
    SMC_VarList_0:= ,
    Vel=> ,
    Done=> ,
    Busy=> ,
```

LD：当 Exec 变为 True 时，执行 G 代码六轴连续插补指令（G 代码文件形式）；程序如下：



- **注意事项**

➢ 这个指令由“PMC_Ipolib”库实现。

➢ IpoCycle参数需要与运动指令所在的任务周期时间保持一致，否则指令执行时可能会出现Error报错。

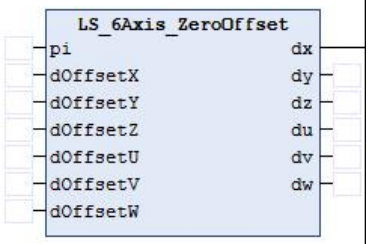
➢ 参数IpoCycle、Jerk、PreProcessGCodeNum、LimitVel、LimitAccDec、LimitMaxAcc、LimitMaxAccJerk不可以设置为0。

- 在指令运行期间，运动轴都不能再被其它运动指令所调用。
- 插补轴需要先把轴上使能（运行MC_POWER指令），才能运行插补指令。

7.10.31 LS_6Axis_ZeroOffset

设置零点坐标偏移

- **指令格式**

指令	名称	FB/FC	LD表现	ST表现	库
LS_6Axis_ZeroOffset	六轴零点偏移	FB		LS_6Axis_ZeroOffset(pi:=, dOffsetX:=, dOffsetY:=, dOffsetZ:=, dOffsetU:=, dOffsetV:=, dOffsetW:=, dx=>, dy=>, dz=>, du=>, dv=>, dw=>);	PMC_IpoLib

- **相关变量**

输入变量

输入输出变量	名称	数据类型	有效范围	初始值	描述
pi	插补输出位置	SMC_POSINFO	-	-	插补模块输出的位置
dOffsetX	X轴偏移值	LREAL	负数，0，正数	0	X轴零点坐标偏移值
dOffsetY	Y轴偏移值	LREAL	负数，0，正数	0	Y轴零点坐标偏移值
dOffsetZ	Z轴偏移	LREAL	负数，0，	0	Z轴零点坐标偏移值

	值		正数		
dOffsetU	U轴偏移值	LREAL	负数，0，正数	0	U轴零点坐标偏移值
dOffsetV	V轴偏移值	LREAL	负数，0，正数	0	V轴零点坐标偏移值
dOffsetW	W轴偏移值	LREAL	负数，0，正数	0	W轴零点坐标偏移值

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
dx	偏移后X位置	LREAL	负数，0，正数	0	偏移后X轴位置
dy	偏移后Y位置	LREAL	负数，0，正数	0	偏移后Y轴位置
dz	偏移后Z位置	LREAL	负数，0，正数	0	偏移后Z轴位置
du	偏移后U位置	LREAL	负数，0，正数	0	偏移后U轴位置
dv	偏移后V位置	LREAL	负数，0，正数	0	偏移后V轴位置
dw	偏移后W位置	LREAL	负数，0，正数	0	偏移后W轴位置

	布尔	位串				整数								实数		时刻、持续时间、日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
pi	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
dOffsetX	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-
dOffsetY	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-
dOffsetZ	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-
dOffsetU	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-

dOffsetV	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-
dOffsetW	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-
dx	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-
dy	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-
dz	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-
du	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-
dv	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-
dw	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-

- **功能说明：**

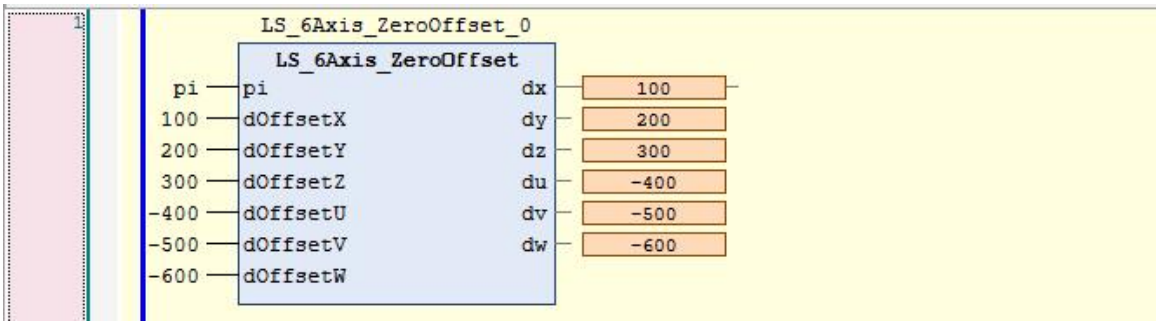
  - 本指令用于设置零点坐标偏移，相当于G54。

- **程序示例**

ST：当 Exec 变为 True 时，执行六轴零点偏移指令；程序如下：

```
LS_6Axis_ZeroOffset_0(
    pi:= pi,
    dOffsetX:= 10,
    dOffsetY:= 20,
    dOffsetZ:= 30,
    dOffsetU:= 40,
    dOffsetV:= 50,
    dOffsetW:= 60,
    dx => dx,
    dy => dy,
    dz => dz,
    du => du,
    dv => dv,
    dw => dw);
    RETURN
```

LD：当 Exec 变为 True 时，执行六轴零点偏移指令；程序如下：



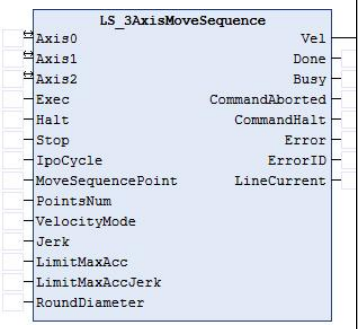
- **注意事项**

  - 这个指令由“PMC_Ipolib”库实现。

7.10.32 LS_3AxisMoveSequence(PMC600)

指定的三个轴做连续插补运动，位置坐标为绝对坐标。

- **指令格式**

指令	名称	FB/FC	LD表现	ST表现	库
LS_3AxisMoveSequence	三轴连续插补	FB		LS_3AxisMoveSequence(Axis0:=, Axis1:=, Axis2:=, Exec:=, Halt:=, Stop:=, IpoCycle:=, MoveSequencePoint:=, PointsNum:=, Velocity Mode:=, Jerk:=, Limit MaxAcc:=, LimitMax AccJerk:=, Round Diameter:=, Vel=>, Done=>, Busy=>, Command Aborted=>, Command Halt=>, Error=>, ErrorID=>, LineCurrent=>);	PMC_IpoLib

- **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	初始值	描述
Axis0	轴0	AXIS_REF_VIRTUAL_SM3	-	-	插补X轴
Axis1	轴1	AXIS_REF_VIRTUAL_SM3	-	-	插补Y轴
Axis2	轴2	AXIS_REF_VIRTUAL_SM3	-	-	插补Z轴

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Exec	启动	BOOL	TRUE-FALSE	FALSE	启动命令，上升沿有效；
Halt	暂停	BOOL	TRUE-FALSE	FALSE	暂停命令，当该输入量为 True 时，当前插补运动暂停；当状态再次切换到 False 时，插补模块继续之前的未完成的插补任务。
Stop	停止	BOOL	TRUE-FALSE	FALSE	停止命令，该命令为 True 时，插补模块停止。
IpoCycle	插补周期	DWORD	-	2000	插补周期，单位：us，必须和所在任务周期一致；
MoveSequencePoint	插补数据表	ARRAY[0..MoveSequence_MAX]OF MoveSequence	-	-	输入的数据点
PointsNum	点数	UINT	正数	0	输入的数据点数
VelocityMode	速度模式	SMC_INT_VELMODE	0-3	SIGMOID	速度模式：梯形-0（TRAPEZOID）；S 形-1（SIGMOID）；四次方-3

					（QUADRATIC）
Jerk	加加速度	LREAL	正数	90000000	加加速度，速度模式为 3 时需要设置此参数，该参数值不能为 0。
LimitMaxAcc	最大加速度	LREAL	正数	90000000	拐角过渡圆弧最大加速度限制
Limit MaxAccJerk	最大加加速度	LREAL	正数	90000000	拐角过渡圆弧最大加加速度限制
RoundDiameter	拐角半径	LREAL	正数	100	拐角过渡圆弧半径，单位：Pulse

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
Vel	合速度	LREAL	0，正数	0	当前插补运动合成速度
Done	完成	BOOL	TRUE-FALSE	FALSE	True表示插补模块已完成插补任务
Busy	运动中	BOOL	TRUE-FALSE	FALSE	True表示插补运动正在进行之中。
Command Aborted	运动中断	BOOL	TRUE-FALSE	FALSE	当Stop有效时，该变量为True，表示插补模块非正常中止。
Command Halt	运动暂停	BOOL	TRUE-FALSE	FALSE	当Halt为True时，该变量为True，表示插补模块处于暂停状态。
Error	错误	BOOL	TRUE-FALSE	FALSE	错误状态；False-没有错误；
ErrorID	错误代码	DINT	0，正数	0	错误代码
LineCurrent	当前执行的数据行号	UINT	0，正数	0	当前执行的数据行号

	布尔	位串				整数								实数		时刻、持续时间、日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Axis0	AXIS_REF_VIRTUAL_SM3																			
Axis1	AXIS_REF_VIRTUAL_SM3																			
Axis2	AXIS_REF_VIRTUAL_SM3																			
Exec	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Halt	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Stop	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Ipo Cycle	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
MoveSequencePoint	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Points Num	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-
VelocityMode	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Jerk	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-
Limit MaxAcc	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-
Limit MaxAccJerk	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-
RoundDiameter	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-
Vel	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-
Done	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Busy	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Command Aborted	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Command Halt	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Error	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
ErrorID	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-
LineCurrent	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-

- **功能说明：**

  - 指定的三个轴做连续插补运动，位置坐标为绝对坐标，一次最多可以输入200个插补

运动点。

- 在连续插补运动中，每个运动点位，都是以MoveSequence结构体的形式存在的，具体的参数设置说明，“MoveSequence”结构体部分参数如下表1.1所示：（完整结构体参数请查阅“PMC_Ipolib”库-数据类型-MoveSequence）

名称	类型	初始化	注释
LineNum	LREAL	0	数据行号
DataType	LREAL	0	数据类型1：直线运动；
X_EndPos	LREAL	0	X终点坐标，绝对坐标，脉冲数
Y_EndPos	LREAL	0	Y终点坐标，绝对坐标，脉冲数
Z_EndPos	LREAL	0	Z终点坐标，绝对坐标，脉冲数
U_EndPos	LREAL	0	U终点坐标，绝对坐标，脉冲数
Vel	LREAL	0	合成速度，Pulse/s
Acc	LREAL	0	合成加速度，Pulse/s2
Aux_Pos	ARRAY[0..1]OFLREAL	[0,0]	辅助点坐标

- **程序示例**

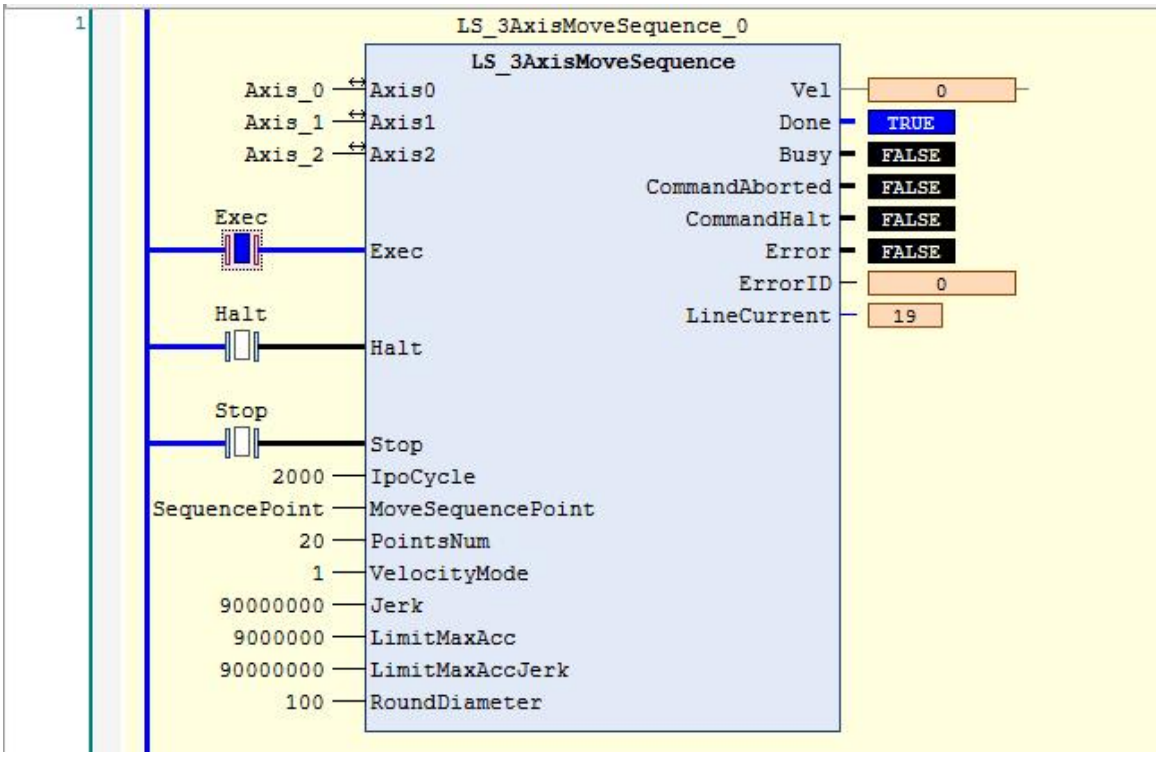
ST：当 Exec 变为 True 时，执行三轴连续插补指令；程序如下：

```
LS_3AxisMoveSequence_0(
    Axis0:= Axis_0,
    Axis1:= Axis_1,
    Axis2:= Axis_2,
    Exec TRUE := Exec TRUE,
    Halt FALSE := Halt FALSE,
    Stop FALSE := Stop FALSE,
    IpoCycle 2000 := 2000,
    MoveSequencePoint:= SequencePoint,
    PointsNum 100 := 100,
    VelocityMode SIGMOID := 1,
    Jerk 9E+07 := 90000000,
    LimitMaxAcc 9E+06 := 9000000,
    LimitMaxAccJerk 9E+07 := 90000000,
    RoundDiameter 100 := 100,
    Vel 1E+03 => Vel 1E+03,
    Done FALSE => Done FALSE,
    Busy TRUE => Busy TRUE,
    CommandAborted FALSE => CommandAborted FALSE,
    CommandHalt FALSE => CommandHalt FALSE,
    Error FALSE => err FALSE,
    ErrorID 0 => errid 0,
    LineCurrent 86 => LineCurrent 86 );

IF aaa TRUE THEN            //配置连续插补数据点参数
    FOR i 101 :=0 TO 100 DO
        SequencePoint[i 101 ].Acc 0 :=2000;        //设置插补运动加速度2000
        SequencePoint[i 101 ].DataType 0 :=1;      //设置插补运动模式：1表示直线插补
        SequencePoint[i 101 ].LineNum 0 :=i 101 ;      //执行的数据行号
        SequencePoint[i 101 ].Vel 0 :=1000;        //设置插补运动合速度

        SequencePoint[i 101 ].X_EndPos 0 :=1000+100*i 101 ;     //设置X终点坐标
        SequencePoint[i 101 ].Y_EndPos 0 :=2000+200*i 101 ;     //设置Y终点坐标
        SequencePoint[i 101 ].Z_EndPos 0 :=3000+300*i 101 ;     //设置Z终点坐标
    END_FOR
END_IF
```

LD：当 Exec 变为 True 时，执行三轴连续插补指令；程序如下：



- **注意事项**

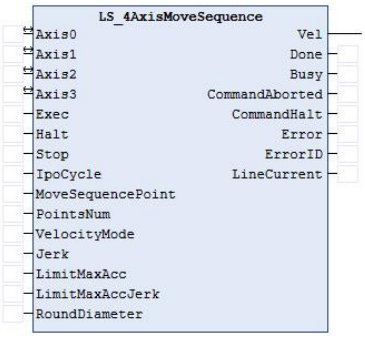
- 这个指令由“PMC_Ipolib”库实现。
- IpoCycle参数需要与运动指令所在的任务周期时间保持一致，否则指令执行时可能会出现Error报错。
- 参数IpoCycle、Jerk、Velocity、Acceleration、Deceleration不可以设置为0。
- 在指令运行期间，三个轴都不能再被其它运动指令所调用。
- Velocity、Acceleration、Deceleration、Jerk大小不能超过“轴设置-SoftMotion驱动：通用”里面的速度，加速度，减速度，加加速度的动态限制，否则指令执行时可能会出现Error报错。
- 当选择的插补类型为直线插补时，如果相邻两段插补运动的轨迹相切或拐角时，规划出来的合速度便会是连续的，如果拐角太大，则速度不连续；当选择的插补类型为圆弧插补时，相邻两段插补运动的轨迹在交接点处如果是相切的，则速度连续，如果不相切时，则速度不连续。
- 启动连续插补指令后，指令内部需要先进行压缓存的操作，插补点数越多，压缓存的时间会越长。
- 在执行连续插补指令前要先配置插补数据表MoveSequencePoint参数。如上述ST例子。

7.10.33 LS_4AxisMoveSequence(PMC600)

指定 Axis0-Axis2 做连续插补运动，Axis3 做跟随运动，位置坐标为绝对坐标

- **指令格式**

指令	名称	FB/FC	LD表现	ST表现	库

LS_4AxisMoveSequence	四轴连续插补	FB		LS_4AxisMoveSequence(Axis0:=, Axis1:=, Axis2:=, Axis3:=, Exec:=, Halt:=, Stop:=, IpoCycle:=, MoveSequencePoint:=, PointsNum:=, Velocity Mode:=, Jerk:=, LimitMax Acc:=, LimitMax AccJerk:=, RoundDiameter:=, Vel=>, Done=>, Busy=>, Command Aborted=>, Command Halt=>, Error=>, ErrorID=>, LineCurrent=>);	PMC_IpoLib

- **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	初始值	描述
Axis0	轴0	AXIS_REF_VIRTUAL_SM3	-	-	插补X轴
Axis1	轴1	AXIS_REF_VIRTUAL_SM3	-	-	插补Y轴
Axis2	轴2	AXIS_REF_VIRTUAL_SM3	-	-	插补Z轴
Axis3	轴3	AXIS_REF_VIRTUAL_SM3	-	-	插补U轴，跟随轴，

					速度不连续；

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Exec	启动	BOOL	TRUE-FALSE	FALSE	启动命令，上升沿有效；
Halt	暂停	BOOL	TRUE-FALSE	FALSE	暂停命令，当该输入量为True时，当前插补运动暂停；当状态再次切换到False时，插补模块继续之前的未完成的插补任务。
Stop	停止	BOOL	TRUE-FALSE	FALSE	停止命令，该命令为True时，插补模块停止。
IpoCycle	插补周期	DWORD	-	2000	插补周期，单位：us，必须和所在任务周期一致；
MoveSequencePoint	插补数据表	ARRAY[0..MoveSequence_MAX]OFMoveSequence	-	-	输入的数据点
Points Num	点数	UINT	正数	0	输入的数据点数
Velocity Mode	速度模式	SMC_INT_VELMODE	0-3	SIGMOID	速度模式：梯形-0（TRAPEZOID）；S形-1（SIGMOID）；四次方-3（QUADRATIC）
Jerk	加加速度	LREAL	正数	90000000	加加速度，速度模式为3时需要设置此参数，该参数值不能为0。
Limit MaxAcc	最大加速度	LREAL	正数	90000000	拐角过渡圆弧最大加速度限制
Limit MaxAccJerk	最大加加速度	LREAL	正数	90000000	拐角过渡圆弧最大加加速度限制

Round Diameter	拐角半径	LREAL	正数	100	拐角过渡圆弧半径，单位：Pulse

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
Vel	合速度	LREAL	0，正数	0	当前插补运动合成速度
Done	完成	BOOL	TRUE-FALSE	FALSE	True表示插补模块已完成插补任务
Busy	运动中	BOOL	TRUE-FALSE	FALSE	True表示插补运动正在进行之中。
Command Aborted	运动中断	BOOL	TRUE-FALSE	FALSE	当Stop有效时，该变量为True，表示插补模块非正常中止。
Command Halt	运动暂停	BOOL	TRUE-FALSE	FALSE	当Halt为True时，该变量为True，表示插补模块处于暂停状态。
Error	错误	BOOL	TRUE-FALSE	FALSE	错误状态；False-没有错误；
ErrorID	错误代码	DINT	0，正数	0	错误代码
LineCurrent	当前执行的数据行号	UINT	0，正数	0	当前执行的数据行号

	布尔	位串				整数								实数		时刻、持续时间、日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Axis0	AXIS_REF_VIRTUAL_SM3																			
Axis1	AXIS_REF_VIRTUAL_SM3																			
Axis2	AXIS_REF_VIRTUAL_SM3																			
Axis3	AXIS_REF_VIRTUAL_SM3																			
Exec	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Halt	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Stop	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
IpoCycle	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

MoveSequencePoint	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
PointsNum	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–
Velocity Mode	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Jerk	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–
LimitMax Acc	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–
LimitMax AccJerk	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–
Round Diameter	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–
Vel	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–
Done	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Busy	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Command Aborted	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Command Halt	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Error	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
ErrorID	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–
LineCurrent	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–

- **功能说明：**

  - 指定Axis0-Axis2做连续插补运动，Axis3做跟随运动，位置坐标为绝对坐标，一次最多可以输入200个插补运动点。

  - 其他参数的设置和LS_3AxisMoveSequence指令相同。

- **程序示例**

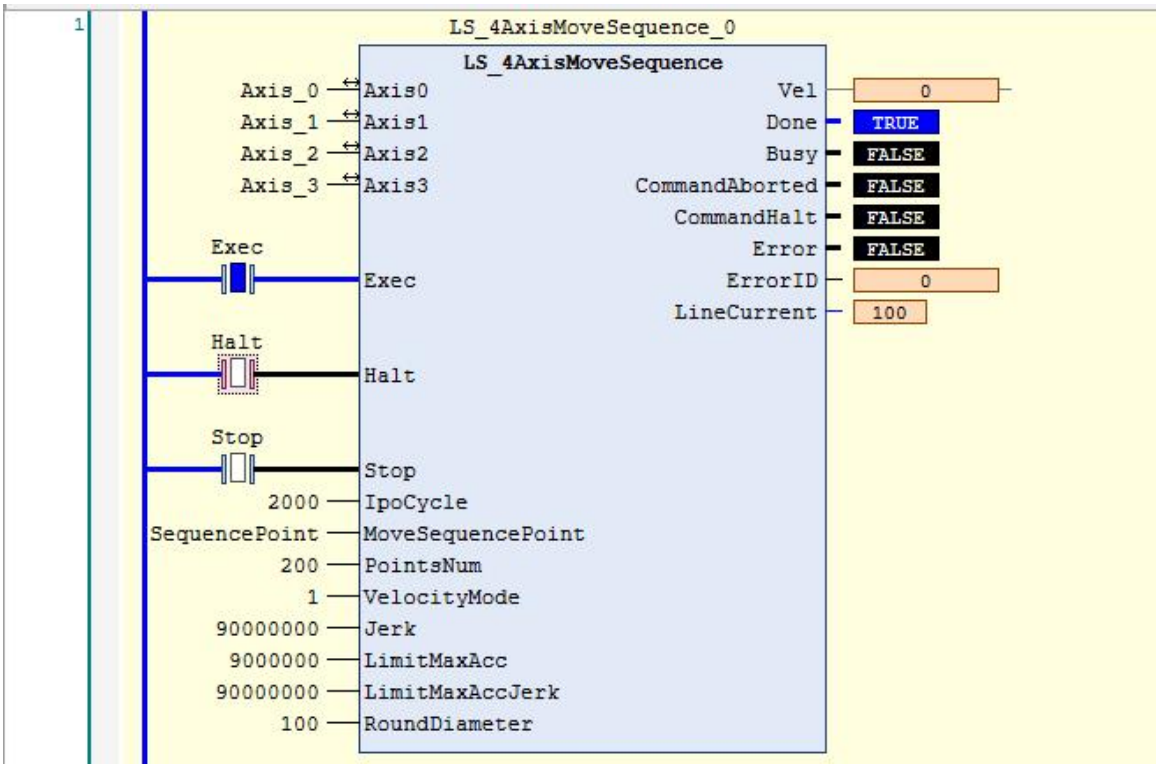
  ST：当Exec变为True时，执行四轴连续插补指令；程序如下：

```
LS_4AxisMoveSequence_0(
    Axis0:= Axis_0,
    Axis1:= Axis_1,
    Axis2:= Axis_2,
    Axis3:= Axis_3,
    Exec TRUE := Exec TRUE,
    Halt FALSE := Halt FALSE,
    Stop FALSE := Stop FALSE,
    IpoCycle 2000 := 2000,
    MoveSequencePoint:= SequencePoint,
    PointsNum 150 := 150,
    VelocityMode SIGMOID := 1,
    Jerk 9E+07 := 90000000,
    LimitMaxAcc 9E+06 := 9000000,
    LimitMaxAccJerk 9E+07 := 90000000,
    RoundDiameter 100 := 100,
    Vel 935 => Vel 935 ,
    Done FALSE => Done FALSE,
    Busy TRUE => Busy TRUE,
    CommandAborted FALSE => CommandAborted FALSE,
    CommandHalt FALSE => CommandHalt FALSE,
    Error FALSE => err FALSE,
    ErrorID 0 => errid 0 ,
    LineCurrent 82 => LineCurrent 82 );


IF aaa TRUE THEN            //配置连续插补数据点参数
    FOR i 101 :=0 TO 100 DO
        SequencePoint[i 101 ].Acc 0 :=2000;        //设置插补运动加速度2000
        SequencePoint[i 101 ].DataType 0 :=1;      //设置插补运动模式：1表示直线插补
        SequencePoint[i 101 ].LineNum 0 :=i 101 ;      //执行的数据行号
        SequencePoint[i 101 ].Vel 0 :=1000;        //设置插补运动合速度

        SequencePoint[i 101 ].X_EndPos 0 :=300+100*i 101 ;     //设置X终点坐标
        SequencePoint[i 101 ].Y_EndPos 0 :=200+200*i 101 ;     //设置Y终点坐标
        SequencePoint[i 101 ].Z_EndPos 0 :=300+300*i 101 ;     //设置Z终点坐标
        SequencePoint[i 101 ].U_EndPos 0 :=400+400*i 101 ;     //设置U终点坐标
    END_FOR
END_IF RETURN
```

LD：当Exec变为True时，执行四轴连续插补指令；程序如下：



- **注意事项**

➢ 这个指令由“PMC_Ipolib”库实现。

➢ IpoCycle参数需要与运动指令所在的任务周期时间保持一致，否则指令执行时可能会出现Error报错。

➢ 参数IpoCycle、Jerk、Velocity、Acceleration、Deceleration不可以设置为0。

➢ 在指令运行期间，四个轴都不能再被其它运动指令所调用。

➢ Velocity、Acceleration、Deceleration、Jerk大小不能超过“轴设置-SoftMotion驱动：通用”里面的速度，加速度，减速度，加加速度的动态限制，否则指令执行时可能会出现Error报错。

➢ 当选择的插补类型为直线插补时，如果相邻两段插补运动的轨迹相切或拐角时，规划出来的合速度便会是连续的，如果拐角太大，则速度不连续；当选择的插补类型为圆弧插补时，相邻两段插补运动的轨迹在交接点处如果是相切的，则速度连续，如果不相切时，则速度不连续。

➢ 启动连续插补指令后，指令内部需要先进行压缓存的操作，插补点数越多，压缓存的时

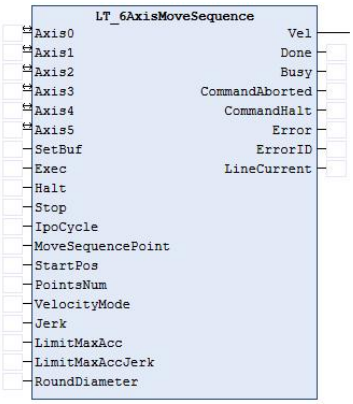
间会越长。

➢ 在执行连续插补指令前要先配置插补数据表MoveSequencePoint参数。如上述ST例子。

7.10.34 LT_6AxisMoveSequence(PMC600)

指定的六个轴做连续插补运动，位置坐标为绝对坐标

- **指令格式**

指令	名称	FB/FC	LD表现	ST表现	库
LT_6AxisMoveSequence	六轴连续插补	FB		LT_6AxisMoveSequence(Axis0:=, Axis1:=, Axis2:=, Axis3:=, Axis4:=, Axis5:=, SetBuf:=, Exec:=, Halt:=, Stop:=, IpoCycle:=, MoveSequencePoint:=, StartPos:=, PointsNum:=, VelocityMode:=, Jerk:=, LimitMaxAcc:=, LimitMaxAcc Jerk:=, Round Diameter:=, Vel=>, Done=>, Busy=>, Command Aborted=>, CommandHalt=>,	PMC_IpoLib

				Error=>, ErrorID=>, LineCurrent=>);	

- 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	初始值	描述
Axis0	轴0	AXIS_REF_VIRTUAL_SM3	-	-	插补X轴
Axis1	轴1	AXIS_REF_VIRTUAL_SM3	-	-	插补Y轴
Axis2	轴2	AXIS_REF_VIRTUAL_SM3	-	-	插补Z轴
Axis3	轴3	AXIS_REF_VIRTUAL_SM3	-	-	插补U轴，跟随轴，速度不连续；
Axis4	轴4	AXIS_REF_VIRTUAL_SM3	-	-	插补V轴，跟随轴，速度不连续
Axis5	轴5	AXIS_REF_VIRTUAL_SM3	-	-	插补W轴，跟随轴，速度不连续；

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
SetBuf	插补数据压入缓存	BOOL	TRUE-FALSE	FALSE	TRUE-插补数据压入缓存；
Exec	启动	BOOL	TRUE-FALSE	FALSE	启动命令，上升沿有效；
Halt	暂停	BOOL	TRUE-FALSE	FALSE	暂停命令，当该输入量为 True 时，当前插补运动暂停；当状态再次切换到 False 时，插补模块继续之前的未完成的插补任务。
Stop	停止	BOOL	TRUE-FALSE	FALSE	停止命令，该命令为 True 时，插补模块停止。
IpoCycle	插补周期	DWORD	-	2000	插补周期，单位：us，必须和所在任务周期

					一致；
MoveSequencePoint	插补数据表	ARRAY[0..MoveSequence_MAX]OFMoveSequence	-	-	输入的数据点
StartPos	轴起点位置	ARRAY[0..5]OFLREAL	负数，0 正数	-	XYZUVW 轴起点位置，单位：Pulse
PointsNum	点数	UINT	正数	0	输入的数据点数
VelocityMode	速度模式	SMC_INT_VELMODE	0-3	SIGMOID	速度模式：梯形-0（TRAPEZOID）；S形-1（SIGMOID）；四次方-3（QUADRATIC）
Jerk	加加速度	LREAL	正数	90000000	加加速度，速度模式为3 时需要设置此参数，该参数值不能为 0。
LimitMaxAcc	最大加速度	LREAL	正数	90000000	拐角过渡圆弧最大加速度限制
LimitMaxAccJerk	最大加加速度	LREAL	正数	90000000	拐角过渡圆弧最大加加速度限制
RoundDiameter	拐角半径	LREAL	正数	100	拐角过渡圆弧半径，单位：Pulse

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
Vel	合速度	LREAL	0，正数	0	当前插补运动合成速度
Done	完成	BOOL	TRUE-FALSE	FALSE	True表示插补模块已完成插补任务
Busy	运动中	BOOL	TRUE-FALSE	FALSE	True表示插补运动正在进行之中。
CommandAborted	运动中断	BOOL	TRUE-FALSE	FALSE	当Stop有效时，该变量为True，表示插补模块非正常中止。
CommandHalt	运动暂停	BOOL	TRUE-FALSE	FALSE	当Halt为True时，该变量为True，表示插补模

					块处于暂停状态。
Error	错误	BOOL	TRUE-FALSE	FALSE	错误状态；False-没有错误；
ErrorID	错误代码	DINT	0，正数	0	错误代码
Line Current	当前执行的数据行号	UINT	0，正数	0	当前执行的数据行号

	布尔	位串				整数								实数		时刻、持续时间、日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Axis0	AXIS_REF_VIRTUAL_SM3																			
Axis1	AXIS_REF_VIRTUAL_SM3																			
Axis2	AXIS_REF_VIRTUAL_SM3																			
Axis3	AXIS_REF_VIRTUAL_SM3																			
Axis4	AXIS_REF_VIRTUAL_SM3																			
Axis5	AXIS_REF_VIRTUAL_SM3																			
SetBuf	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Exec	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Halt	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Stop	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
IpoCycle	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
MoveSequencePoint	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_
StartPos	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
PointsNum	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–
VelocityMode	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Jerk	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–
LimitMaxAcc	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–
LimitMaxAccJerk	_	_	_	_	_	_	_	_	_	_	_	_	_	_	√	_	_	_	_	_
RoundDiameter	_	_	_	_	_	_	_	_	_	_	_	_	_	_	√	_	_	_	_	_

Vel	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–
Done	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Busy	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
CommandAborted	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
CommandHalt	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Error	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
ErrorID	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–
LineCurrent	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–

- **功能说明：**

  - 指定Axis0-Axis2做连续插补运动，Axis3-Axis5做跟随运动，位置坐标为绝对坐标，一次最多可以输入200个插补运动点。

  - 其他参数的设置和LS_3AxisMoveSequence指令相同。

- **程序示例**

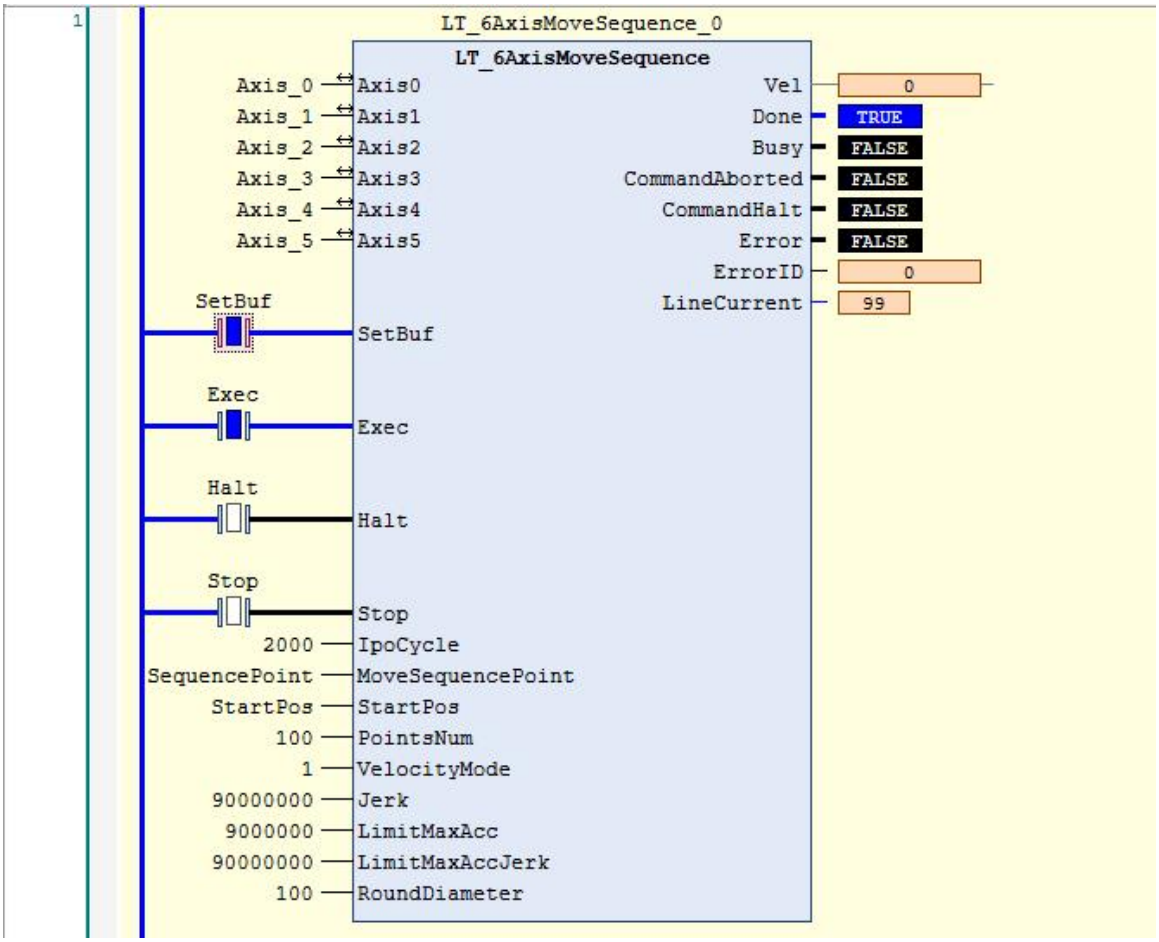
  ST：当 Exec 变为 True 时，执行六轴连续插补指令；程序如下：

```
LT_6AxisMoveSequence_0(
    Axis0:= Axis_0,
    Axis1:= Axis_1,
    Axis2:= Axis_2,
    Axis3:= Axis_3,
    Axis4:= Axis_4,
    Axis5:= Axis_5,
    SetBuf TRUE := SetBuf TRUE,
    Exec TRUE := Exec TRUE,
    Halt FALSE := Halt FALSE,
    Stop FALSE := Stop FALSE,
    IpoCycle 2000 := 2000,
    MoveSequencePoint:= SequencePoint,
    StartPos:= StartPos,
    PointsNum 200 := 200,
    VelocityMode SIGMOID := 1,
    Jerk 9E+07 := 90000000,
    LimitMaxAcc 9E+06 := 9000000,
    LimitMaxAccJerk 9E+07 := 90000000,
    RoundDiameter 100 := 100,
    Vel 935 => Vel 935 ,
    Done FALSE => Done FALSE ,
    Busy TRUE => Busy TRUE ,
    CommandAborted FALSE => CommandAborted FALSE ,
    CommandHalt FALSE => CommandHalt FALSE ,
    Error FALSE => err FALSE ,
    ErrorID 0 => errid 0 ,
    LineCurrent 39 => LineCurrent 39 );

IF aaa TRUE THEN            //配置连续插补数据点参数
    FOR i 101 :=0 TO 100 DO
        SequencePoint[i 101 ].Acc 0 :=2000;        //设置插补运动加速度2000
        SequencePoint[i 101 ].DataType 0 :=1;      //设置插补运动模式：1表示直线插补
        SequencePoint[i 101 ].LineNum 0 :=i 101 ;      //执行的数据行号
        SequencePoint[i 101 ].Vel 0 :=1000;        //设置插补运动合速度

        SequencePoint[i 101 ].X_EndPos 0 :=100+100*i 101 ;     //设置X终点坐标
        SequencePoint[i 101 ].Y_EndPos 0 :=200+200*i 101 ;     //设置Y终点坐标
        SequencePoint[i 101 ].Z_EndPos 0 :=300+300*i 101 ;     //设置Z终点坐标
        SequencePoint[i 101 ].U_EndPos 0 :=400+400*i 101 ;     //设置U终点坐标
        SequencePoint[i 101 ].V_EndPos 0 :=500+400*i 101 ;     //设置U终点坐标
        SequencePoint[i 101 ].W_EndPos 0 :=600+400*i 101 ;     //设置U终点坐标
    END_FOR
END_IFRETURN
```

LD：当Exec变为True时，执行六轴连续插补指令；程序如下：



- **注意事项**

➢ 这个指令由“PMC_Ipolib”库实现。

➢ IpoCycle参数需要与运动指令所在的任务周期时间保持一致，否则指令执行时可能会出现Error报错。

➢ 参数IpoCycle、Jerk、Velocity、Acceleration、Deceleration不可以设置为0。

➢ 在指令运行期间，六个轴都不能再被其它运动指令所调用。

➢ Velocity、Acceleration、Deceleration、Jerk大小不能超过“轴设置-SoftMotion驱动：通用”里面的速度，加速度，减速度，加加速度的动态限制，否则指令执行时可能会出现Error报错。

➢ 当选择的插补类型为直线插补时，如果相邻两段插补运动的轨迹相切或拐角时，规划出来的合速度便会是连续的，如果拐角太大，则速度不连续；当选择的插补类型为圆弧插补时，相邻两段插补运动的轨迹在交接点处如果是相切的，则速度连续，如果不相切时，

则速度不连续。

- 启动连续插补指令后，先将SetBuf置TRUE，进行插补数据压入缓存的操作，插补点数越多，压缓存的时间会越长。
- 在执行连续插补指令前要先配置插补数据表MoveSequencePoint参数。如上述ST例子。

7.11 雷赛单轴运动控制指令

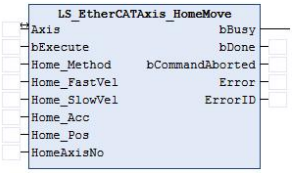
7.11.1 指令列表

指令类别	名称	FB/FC	功能
雷赛单轴运动控制指令	LS_EtherCATAxis_HomeMove	FB	回零（驱动器回零）
	LS_MoveAbsChangePosVel	FB	在线变速变位置
	LS_MoveChangeVel	FB	在线变速
	LS_MoveAbs	FB	单轴起跳速度绝对运动
	LS_MoveRel	FB	单轴起跳速度相对运动
	LS_MoveVel	FB	单轴起跳速度恒速运动
	LS_ResetVirtualAxis	FB	单轴复位功能

7.11.2 LS_EtherCATAxis_HomeMove

控制器通过 SDO 服务向驱动器写入回零参数。

- **指令格式**

指令	名称	FB/FC	LD表现	ST表现	库
LS_EtherCATAxis_HomeMove	回零指令	FB		LS_EtherCATAxis_HomeMove(Axis:=, bExecute:=, Home_Method:=, Home_FastVel:=, Home_SlowVel:=, Home_Acc:=, Home_Pos:=, HomeAxisNo:=, bBusy=>, bDone=>, bCommandAborted=>, Error=>, ErrorID=>);	PMC_SingleAxisLib

- **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	初始值	描述
Axis	轴	AXIS_REF_VIRTUAL_SM3	-	-	执行回零的轴

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
bExecute	启动	BOOL	TRUE-FALSE	FALSE	启动命令，回零过程中该信号需要保持，回零完成再清除；
Home_Method	方式	DWORD	-	21	回零方式，请参考各驱动支持的回零方式（回零方式对象字典索引为0x6098，子索引0x00）；
Home_FastVel	快速	DWORD	0，正数	500	回零快速，对象字典索引为0x6099，子索引0x01；
Home_SlowVel	慢速	DWORD	0，正数	100	回零慢速，对象字典索引为0x6099，子索引0x02；
Home_Acc	加速度	DWORD	0，正数	1000	回零加减速，对象字典索引为0x609A，子索引0x00；
Home_Pos	偏移值	DWORD	负数，0，正数	0	回零后偏移值，对象字典索引为0x607C，子索引0x00；
HomeAxisNo	驱动下轴号	DWORD	0，1	0	轴编号。0—驱动或模块下0轴；1—驱动或模块下1轴；

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
bBusy	执行中	BOOL	TRUE-FALSE	FALSE	True表示回零过程中。
bDone	完成	BOOL	TRUE-FALSE	FALSE	True表示回零完成。
bCommandAborted	回零中断	BOOL	TRUE-FALSE	FALSE	回零取消。
Error	错误	BOOL	TRUE-FALSE	FALSE	错误状态；False-没有错误；
ErrorID	错误代码	UDINT	0，正数	0	错误代码

	布尔	位串	整数	实数	时刻、持续时间、日期、字符串

	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Axis	AXIS_REF_VIRTUAL_SM3																			
bExecute	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Home_ Method	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Home_ FastVel	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Home_ SlowVel	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Home _Acc	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Home _Pos	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Home AxisNo	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
bBusy	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
bDone	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
bCommand Aborted	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Error	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
ErrorID	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–

- **功能说明：**

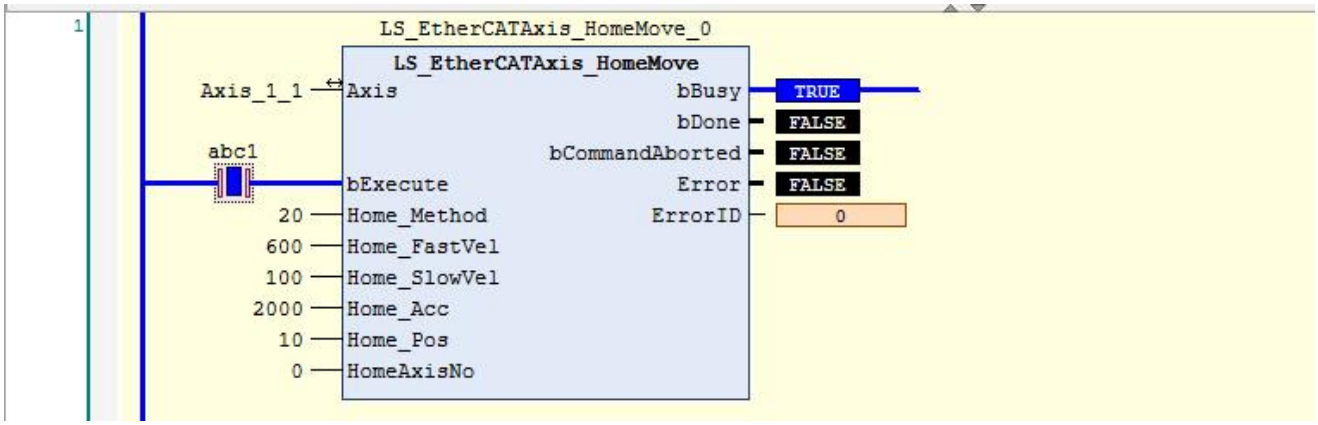
  - 控制器通过SDO服务向驱动器写入回零参数。此回零指令的回零模式由连接在总线上的从站决定，控制端只是将需要的回零参数发给驱动器，具体的回零动作，在驱动器端完成。

- **程序示例**

ST：当abc变为True时，执行回零指令；程序如下：

```
LS_EtherCATAxis_HomeMove_0(
    Axis:= Axis_1_1,
    bExecute TRUE := abc TRUE,
    Home_Method 19 := 19,
    Home_FastVel 1000 := 1000,
    Home_SlowVel 200 := 200,
    Home_Acc 10000 := 10000,
    Home_Pos 0 := 0,
    HomeAxisNo 0 := 0,
    bBusy TRUE => bBusy TRUE,
    bDone FALSE => bDone FALSE,
    bCommandAborted FALSE => Aborted FALSE,
    Error FALSE => err FALSE,
    ErrorID 0 => errid 0 );RETURN
```

LD：当abc1变为True时，执行回零指令；程序如下：



- **注意事项**

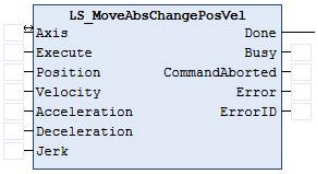
➢ 这个指令由“PMC_SingleAxisLib”库实现。

➢ 调用该指令时，需要正确配置对象字典：回零模式6098、回零速度6099、回零加减速609A、回零偏移值607C，具体的参数说明需要参考所使用的总线驱动。

7.11.3 LS_MoveAbsChangePosVel

单轴绝对运动，可以在线变速、变位置（绝对位置）。

- **指令格式**

指令	名称	数据类型	有效范围	初始值	描述
LS_MoveAbsChangePosVel	在线变速变位置指令	FB		LS_MoveAbsChangePosVel(Axis:=, Execute:=, Position:=,	PMC_SingleAxisLib

				Velocity:=, Acceleration:=, Deceleration:=, Jerk:=, Done=>, Busy=>, CommandAborted=>, Error=>, ErrorID=>);	

- 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	初始值	描述
Axis	轴	AXIS_REF_VIRTUAL_SM3	-	-	指定轴

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Execute	启动	BOOL	TRUE-FALSE	FALSE	启动命令，上升沿触发指令执行；
Position	目标位置	LREAL	负数，0，正数	0	目标位置（绝对坐标）
Velocity	目标速度	LREAL	负数，0，正数	0	目标速度
Acceleration	目标加速度	LREAL	负数，0，正数	0	加速度值
Deceleration	目标减速度	LREAL	负数，0，正数	0	减速度值
Jerk	目标加加速度	LREAL	负数，0，正数	0	加加速度值

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
Done	完成	BOOL	TRUE-FALSE	FALSE	True表示指令执行完成。
Busy	执行中	BOOL	TRUE-FALSE	FALSE	True表示指令执行中。
CommandAborted	指令被中	BOOL	TRUE-FALSE	FALSE	当功能块执行被中断

	断				时为TRUE。
Error	错误	BOOL	TRUE-FALSE	FALSE	错误状态；False-没有错误；
ErrorID	错误代码	UDINT	0，正数	0	错误代码

	布尔	位串				整数								实数		时刻、持续时间、日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Axis	AXIS_REF_VIRTUAL_SM3																			
Execute	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Position	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-
Velocity	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-
Acceleration	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-
Deceleration	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-
Jerk	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-
Done	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Busy	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
CommandAborted	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Error	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
ErrorID	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-

- **功能说明：**

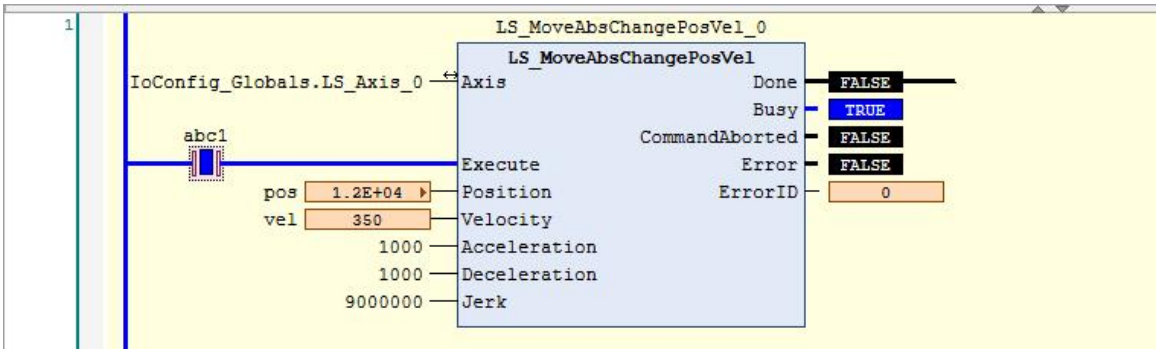
  - 单轴绝对运动，可以在线变速、变位置（绝对位置）。

- **程序示例**

ST：当abc变为True时，执行单轴绝对运动（可在线变速变位置）指令；程序如下：

```
LS_MoveAbsChangePosVel_0(
    Axis:= IoConfig_Globals.LS_Axis_1,
    Execute TRUE := abc TRUE,
    Position 1E+04 := Position 1E+04,
    Velocity 200 := Velocity 200,
    Acceleration 2E+03 := 2000,
    Deceleration 2E+03 := 2000,
    Jerk 9E+06 := 9000000,
    Done TRUE => Done TRUE,
    Busy FALSE => Busy FALSE,
    CommandAborted FALSE => CommandAborted FALSE,
    Error FALSE => err FALSE,
    ErrorID 0 => errid 0);
```

LD：当 abc1 变为 True 时，执行单轴绝对运动（可在线变速变位置）指令；程序如下：



- **注意事项**

➢ 这个指令由“PMC_SingleAxisLib”库实现。

➢ 当Execute为上升沿时，启动模块执行，且该信号需要保持。在Execute为True状态，改变Position或Velocity，数据立即生效，按照新设置的值执行；如果Execute为False，改变的值不起作用。

➢ 如果运动完成后，即Done为True，再改变Position或Velocity，新的运动不会执行。

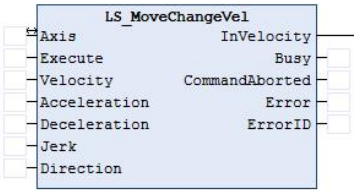
➢ 轴运动的停止，需要单独调用MC_Stop功能块。

7.11.4 LS_MoveChangeVel

在运动过程中，修改运动方向、速度、加速度、减速度、Jerk，立即生效。

- **指令格式**

指令	名称	FB/FC	LD表现	ST表现	库

LS_MoveChangeVel	在线变速指令	FB		LS_MoveChangeVel(Axis:=, Execute:=, Velocity:=, Acceleration:=, Deceleration:=, Jerk:=, Direction:=, InVelocity=>, Busy=>, CommandAborted=>, Error=>, ErrorID=>);	PMC_SingleAxisLib

- **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	初始值	描述
Axis	轴	AXIS_REF_VIRTUAL_SM3	-	-	指定轴

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Execute	启动	BOOL	TRUE-FALSE	FALSE	启动命令，上升沿触发指令执行；
Velocity	目标速度	LREAL	负数，0，正数	0	修改的速度
Acceleration	目标加速度	LREAL	负数，0，正数	0	加速度值
Deceleration	目标减速度	LREAL	负数，0，正数	0	减速度值
Jerk	目标加加速度	LREAL	负数，0，正数	0	加加速度值
Direction	方向	INT	-1，1	1	运动方向。1：正方向；-1：负方向；

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
InVelocity	到达设定速度	BOOL	TRUE-FALSE	FALSE	TRUE表示速度达到设定速度；
Busy	执行中	BOOL	TRUE-FALSE	FALSE	True表示指令执行中。
Command Aborted	指令被中断	BOOL	TRUE-FALSE	FALSE	当功能块执行被中断时为TRUE。
Error	错误	BOOL	TRUE-FALSE	FALSE	错误状态；False-没有错误；
ErrorID	错误代码	UDINT	0，正数	0	错误代码

	布尔	位串				整数								实数		时刻、持续时间、日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Axis	AXIS_REF_VIRTUAL_SM3																			
Execute	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Velocity	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-
Acceleration	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-
Deceleration	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-
Jerk	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-
Direction	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-
InVelocity	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Busy	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
CommandAborted	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Error	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
ErrorID	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-

- **功能说明：**

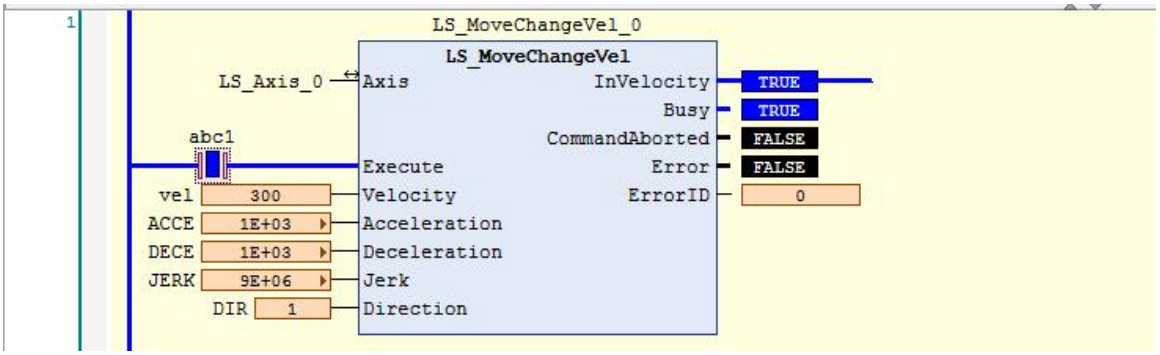
  - 本指令的主要功能，是在运动过程中，修改运动方向、速度、加速度、减速度、Jerk，立即生效。

- **程序示例**

ST：当abc变为True时，执行单轴恒速运动（可在线变速）指令；程序如下：

```
LS_MoveChangeVel_0(
    Axis:= LS_Axis_1,
    Execute TRUE := abc TRUE,
    Velocity 1E+03 := Velocity 1E+03,
    Acceleration 2E+04 := Acceleration 2E+04,
    Deceleration 2E+04 := Deceleration 2E+04,
    Jerk 9E+07 := Jerk 9E+07,
    Direction 1 := Direction 1,
    InVelocity TRUE => InVelocity TRUE,
    Busy TRUE => Busy TRUE,
    CommandAborted FALSE => CommandAborted FALSE,
    Error FALSE => err FALSE,
    ErrorID 0 => errid 0);
```

LD：当abc1变为True时，执行单轴恒速运动（可在线变速）指令；程序如下：



- **注意事项**

➢ 这个指令由“PMC_SingleAxisLib”库实现。

➢ 当Execute为上升沿时，启动模块执行，且该信号需要保持。在Execute为True状态，改变运动方向、速度、加速度、减速度、Jerk，数据立即生效，按照新设置的值执行；如果Execute为False，改变的值不起作用。

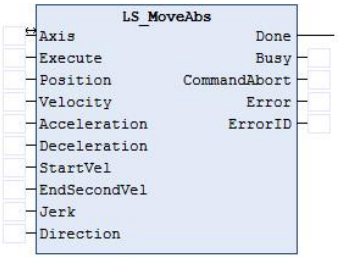
➢ 轴运动的停止，需要单独调用MC_Stop功能块。

7.11.5 LS_MoveAbs

控制轴按照设定参数运动到指定的绝对位置，只支持 T 型加减速。

- **指令格式**

指令	名称	FB/FC	LD表现	ST表现	库

LS_MoveAbs	单轴起跳速度绝对运动指令	FB		LS_MoveAbs(Axis:=, Execute:=, Position:=, Velocity:=, Acceleration:=, Deceleration:=, StartVel:=, EndSecondVel:=, Jerk:=, Direction:=, Done=>, Busy=>, CommandAbort=>, Error=>, ErrorID=>);	PMC_SingleAxisLib

- **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	初始值	描述
Axis	轴	AXIS_REF_VIRTUAL_SM3	-	-	指定轴

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Execute	启动	BOOL	TRUE-FALSE	FALSE	启动命令，上升沿触发指令执行；
Position	目标位置	LREAL	负数，0，正数	0	运动的目标位置；
Velocity	目标速度	LREAL	0，正数	0	目标速度；
Acceleration	目标加速度	LREAL	0，正数	0	加速度值；
Deceleration	目标减速度	LREAL	0，正数	0	减速度值；
StartVel	起跳速度	LREAL	>0.001	20	起跳速度，必须为正值，最小值为0.001
EndSecondVel	停止速度	LREAL	>0.001	20	停止前缓冲速度，最小值为0.001
Jerk	加加速度	LREAL	0，正数	0	加加速度值
Direction	方向	INT	-1,1		运动方向。1：正；-1：负；

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
Done	完成	BOOL	TRUE-FALSE	FALSE	True表示达到最终位置。
Busy	执行中	BOOL	TRUE-FALSE	FALSE	True表示指令执行中。
CommandAborted	指令被中断	BOOL	TRUE-FALSE	FALSE	当功能块执行被中断时为TRUE。
Error	错误	BOOL	TRUE-FALSE	FALSE	错误状态；False-没有错误；
ErrorID	错误代码	UDINT	0，正数	0	错误代码

	布尔	位串				整数								实数		时刻、持续时间、日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Axis	AXIS_REF_VIRTUAL_SM3																			
Execute	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Position	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-
Velocity	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-
Acceleration	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-
Deceleration	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-
StartVel	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-
EndSecondVel	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-
Jerk	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-
Direction	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-
Done	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Busy	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
CommandAborted	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Error	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
ErrorID	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-

- **功能说明：**

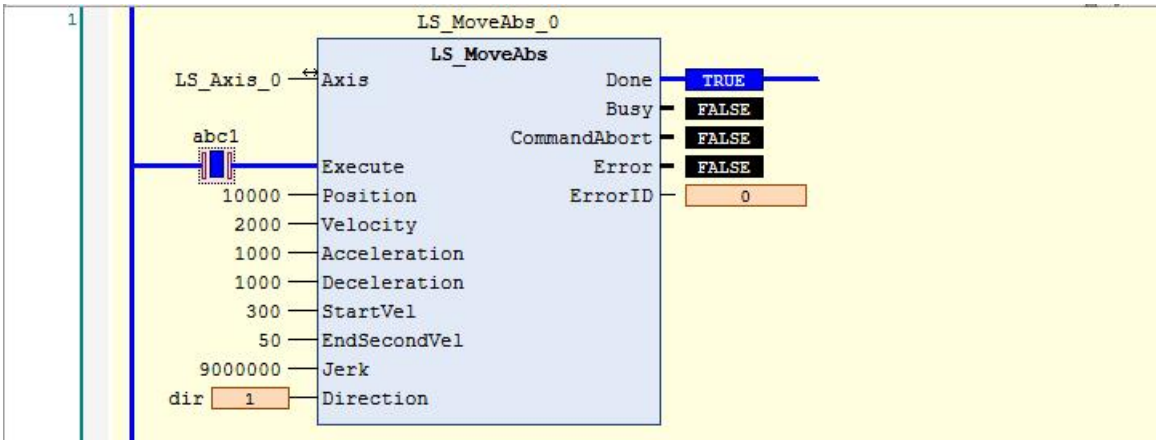
  - 控制轴按照设定参数运动到指定的绝对位置，只支持T型加减速。

- **程序示例**

ST：当abc变为True时，执行单轴起跳速度绝对运动指令；程序如下：

```
LS_MoveAbs_0(
    Axis:= LS_Axis_1,
    Execute TRUE := abc TRUE,
    Position 9E+04 := Position 9E+04,
    Velocity 1E+03 := Velocity 1E+03,
    Acceleration 1E+04 := Acceleration 1E+04,
    Deceleration 1E+04 := Deceleration 1E+04,
    StartVel 500 := StartVel 500,
    EndSecondVel 200 := EndSecondVel 200,
    Jerk 9E+07 := 90000000,
    Direction negative := Direction -1,
    Done FALSE => Done FALSE,
    Busy TRUE => Busy TRUE,
    CommandAbort FALSE => CommandAbort FALSE,
    Error FALSE => err FALSE,
    ErrorID 0 => errid 0);
```

LD：当abc1变为True时，执行单轴起跳速度绝对运动指令；程序如下：



- **注意事项**

➢ 这个指令由“PMC_SingleAxisLib”库实现。

➢ 如果是线性轴的绝对点位运动，方向值将被忽略。

➢ 起跳速度和停止缓冲速度不能为负。

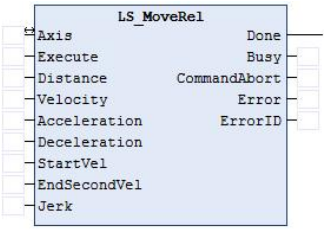
➢ 如果距离过短，可能达不到最大速度。

➢ 将加减速度设定为0后，将不做加减速而直接达到目标速度。

7.11.6 LS_MoveRel

控制轴按照设定参数运动一段相对距离，只支持 T 型加减速。

- **指令格式**

指令	名称	FB/FC	LD表现	ST表现	库
LS_MoveRel	单轴起跳速度相对运动指令	FB		LS_MoveRel(Axis:=, Execute:=, Distance:=, Velocity:=, Acceleration:=, Deceleration:=, StartVel:=, EndSecondVel:=, Jerk:=, Done=>, Busy=>, CommandAbort=>, Error=>, ErrorID=>);	PMC_SingleAxisLib

- **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	初始值	描述
Axis	轴	AXIS_REF_VIRTUAL_SM3	-	-	指定轴

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Execute	启动	BOOL	TRUE-FALSE	FALSE	启动命令，上升沿触发指令执行；
Distance	距离	LREAL	负数，0，正数	0	运动的距离；
Velocity	目标速度	LREAL	0，正数	0	目标速度；
Acceleration	加速度	LREAL	0，正数	0	加速度值；
Deceleration	减速度	LREAL	0，正数	0	减速度值；
StartVel	起跳速度	LREAL	>0.001	20	起跳速度，必须为正值，最小值为0.001
EndSecondVel	停止速度	LREAL	>0.001	20	停止前缓冲速度，最小值为0.001

Jerk	加加速度	LREAL	0，正数	0	加加速度值

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
Done	完成	BOOL	TRUE-FALSE	FALSE	True表示指令执行完成。
Busy	执行中	BOOL	TRUE-FALSE	FALSE	True表示指令执行中。
CommandAborted	指令被中断	BOOL	TRUE-FALSE	FALSE	当功能块执行被中断时为TRUE。
Error	错误	BOOL	TRUE-FALSE	FALSE	错误状态；False-没有错误；
ErrorID	错误代码	UDINT	0，正数	0	错误代码

	布尔	位串				整数								实数		时刻、持续时间、日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Axis	AXIS_REF_VIRTUAL_SM3																			
Execute	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Distance	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-
Velocity	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-
Acceleration	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-
Deceleration	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-
StartVel	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-
EndSecondVel	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-
Jerk	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-
Done	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Busy	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
CommandAborted	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Error	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
ErrorID	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-

- **功能说明：**

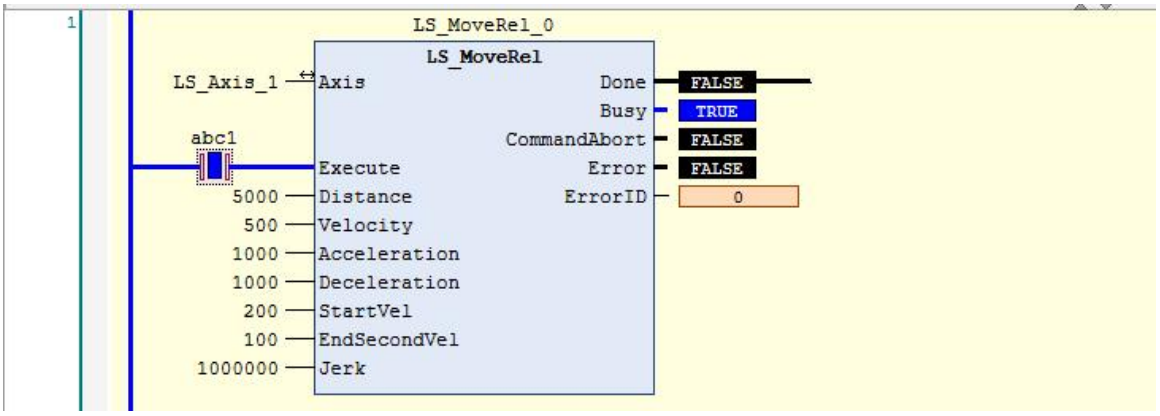
  - 控制轴按照设定参数运动一段相对距离，只支持T型加减速。

- **程序示例**

ST：当abc变为True时，执行单轴起跳速度相对运动指令；程序如下：

```
LS_MoveRel_0(
    Axis:= LS_Axis_0,
    Execute TRUE := abc TRUE,
    Distance 1E+04 := 10000,
    Velocity 1E+03 := 1000,
    Acceleration 2E+03 := 2000,
    Deceleration 2E+03 := 2000,
    StartVel 200 := 200,
    EndSecondVel 200 := 200,
    Jerk 1E+06 := 1000000,
    Done TRUE => Done TRUE,
    Busy FALSE => Busy FALSE,
    CommandAbort FALSE => CommandAbort FALSE,
    Error FALSE => err FALSE,
    ErrorID 0 => errid 0 );
    RETURN
```

LD：当abc1变为True时，执行单轴起跳速度相对运动指令；程序如下：



- **注意事项**

➢ 这个指令由“PMC_SingleAxisLib”库实现。

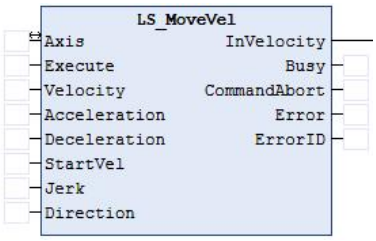
➢ 此功能块不支持在线变速变位置的功能。

7.11.7 LS_MoveVel

控制电机以指定的起跳速度和加速度加速到目标速度，然后以该最大速度一直运行。

- **指令格式**

指令	名称	FB/FC	LD表现	ST表现	库

LS_MoveVel	单轴起跳速度恒速运动指令	FB		LS_MoveVel(Axis:=, Execute:=, Velocity:=, Acceleration:=, Deceleration:=, StartVel:=, Jerk:=, Direction:=, InVelocity=>, Busy=>, CommandAbort=>, Error=>, ErrorID=>);	PMC_SingleAxisLib

- **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	初始值	描述
Axis	轴	AXIS_REF_VIRTUAL_SM3	-	-	指定轴

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Execute	启动	BOOL	TRUE-FALSE	FALSE	启动命令，上升沿触发指令执行；
Velocity	目标速度	LREAL	0，正数	0	目标速度；
Acceleration	目标加速度	LREAL	0，正数	0	加速度值；
Deceleration	目标减速度	LREAL	0，正数	0	减速度值；
StartVel	起跳速度	LREAL	>0.001	20	起跳速度，必须为正值，最小值为0.001
Jerk	加加速度	LREAL	0，正数	0	加加速度值
Direction	方向	INT	-1,1	0	运动方向。1：正；-1：负；

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
InVelocity	速度到达	BOOL	TRUE-FALSE	FALSE	True表示速度达到设定值；

Busy	执行中	BOOL	TRUE-FALSE	FALSE	True表示指令执行中。
CommandAborted	指令被中断	BOOL	TRUE-FALSE	FALSE	当功能块执行被中断时为TRUE。
Error	错误	BOOL	TRUE-FALSE	FALSE	错误状态；False-没有错误；
ErrorID	错误代码	UDINT	0，正数	0	错误代码

	布尔	位串				整数								实数		时刻、持续时间、日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Axis	AXIS_REF_VIRTUAL_SM3																			
Execute	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Velocity	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-
Acceleration	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-
Deceleration	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-
StartVel	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-
Jerk	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-
Direction	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-
InVelocity	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Busy	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
CommandAborted	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Error	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
ErrorID	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-

- **功能说明：**

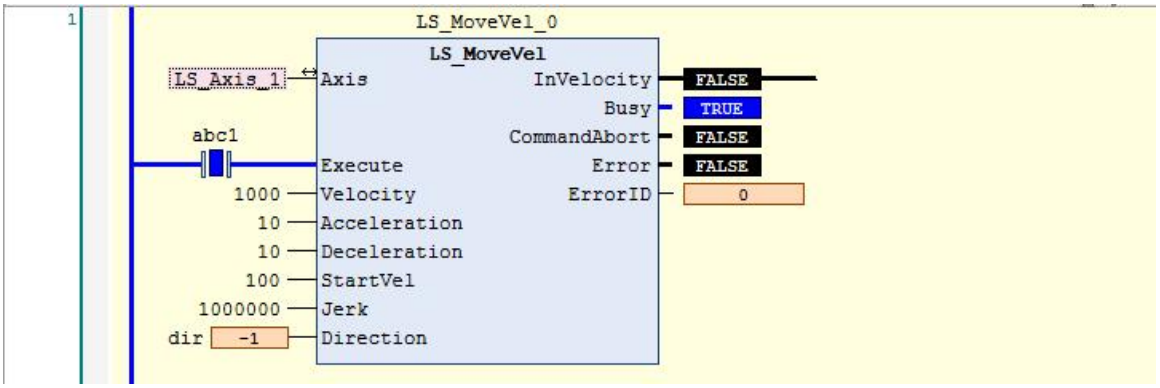
  - 控制电机以指定的起跳速度和加速度加速到目标速度，然后以该最大速度一直运行。直到调用停止指令或者其他的中断指令中断该指令，只支持梯形加减速。

- **程序示例**

ST：当 abc 变为 True 时，执行单轴起跳速度恒速运动指令；程序如下：

```
LS_MoveVel_0(
    Axis:= LS_Axis_0,
    Execute TRUE := abc TRUE,
    Velocity 2E+03 := 2000,
    Acceleration 100 := 100,
    Deceleration 100 := 100,
    StartVel 500 := 500,
    Jerk 1E+06 := 1000000,
    Direction positive := Direction 1,
    InVelocity TRUE => InVelocity TRUE,
    Busy TRUE => Busy TRUE,
    CommandAbort FALSE => CommandAbort FALSE,
    Error FALSE => err FALSE,
    ErrorID 0 => errid 0);
    RETURN
```

LD：当 abc1 变为 True 时，执行单轴起跳速度恒速运动指令；程序如下：



- **注意事项**

➢ 这个指令由“PMC_SingleAxisLib”库实现。

➢ 负速度×负方向=正速度。

➢ 起跳速度必须为正。

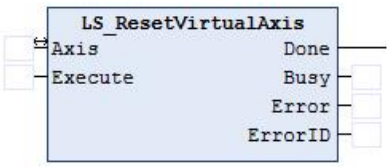
➢ 轴运动的停止，需要单独调用MC_Stop功能块。

7.11.8 LS_ResetVirtualAxis

单轴复位功能。

- **指令格式**

指令	名称	FB/FC	LD表现	ST表现	库

LS_ResetVirtualAxis	单轴复位功能指令	FB		LS_ResetVirtualAxis(Axis:=, Execute:=, Done=>, Busy=>, Error=>, ErrorID=>);	PMC_SingleAxisLib

- **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	初始值	描述
Axis	轴	AXIS_REF_VIRTUAL_SM3	-	-	指定轴

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Execute	启动	BOOL	TRUE-FALSE	FALSE	启动命令，上升沿触发指令执行；

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
Done	完成	BOOL	TRUE-FALSE	FALSE	True表示指令执行完成。
Busy	执行中	BOOL	TRUE-FALSE	FALSE	True表示指令执行中。
Error	错误	BOOL	TRUE-FALSE	FALSE	错误状态；False-没有错误；
ErrorID	错误代码	UDINT	0，正数	0	错误代码

	布尔	位串				整数								实数		时刻、持续时间、日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Axis	AXIS_REF_VIRTUAL_SM3																			
Execute	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Done	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Busy	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

Error	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
ErrorID	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-

- **功能说明：**

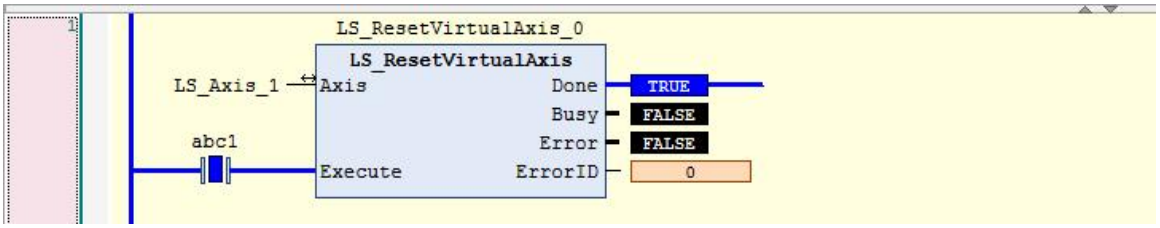
  ➢ 读取轴状态为错误停止ErrorStop和轴结构体错误FBErrorOccured时，可调用此功能解除错误和报错信息，恢复轴状态为StandStill。

- **程序示例**

ST：当abc变为True时，执行单轴复位功能指令；程序如下：

```
LS_ResetVirtualAxis_0(
    Axis:= LS_Axis_0,
    Execute TRUE := abc TRUE,
    Done TRUE => Done TRUE,
    Busy FALSE => Busy FALSE,
    Error FALSE => err FALSE,
    ErrorID 0 => errid 0);
    RETURN
```

LD：当abc1变为True时，执行单轴复位功能指令；程序如下：



- **注意事项**

➢ 这个指令由“PMC_SingleAxisLib”库实现。

7.12 脉冲轴控制

7.12.1 指令列表

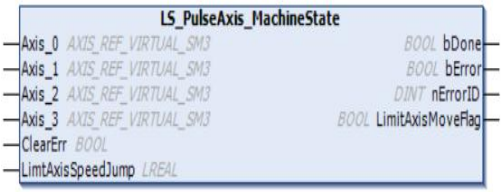
指令类别	名称	FB/FC	功能
脉冲控制指令	LS_PulseAxis_MachineState	FC	轴脉冲输出控制
	LS_PulseAxis_SetPulseMode	FC	设置轴脉冲模式
	LS_PulseAxis_GetPulseMode	FC	读取轴脉冲模式

	LS_PulseAxis_SetSpecialIOLogicalState	FC	设置轴特殊 IO 信号
	LS_PulseAxis_GetSpecialIOLogicalState	FC	读取轴特殊 IO 信号
	LS_PulseAxis_GetSpecialIOState	FC	获取轴特殊 IO 状态
	LS_PulseAxis_SetRatio	FC	设置轴电子齿轮比
	LS_PulseAxis_ReadRatio	FC	读取轴电子齿轮比

7.12.2 LS_PulseAxis_MachineState

控制 PMC600/PMC610 主机脉冲轴的脉冲输出。在使用脉冲轴前必须调用，否则无法输出脉冲。

- **指令格式**

指令	名称	FB/FC	LD表现	ST表现	库
LS_PulseAxis_MachineState	轴脉冲输出控制模块	FC		LS_PulseAxis_MachineState(Axis_0:=, Axis_1:=, Axis_2:=, Axis_3:=, ClearErr:=, LimtAxisSpeedJump:=, bDone=>, bError=>, nErrorID=>, LimitAxisMoveFlag=>);	PMC_Controller

- **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	初始值	描述
Axis_0	轴0	AXIS_REF_VIRTUAL_SM3	-	-	指定轴0
Axis_1	轴1	AXIS_REF_VIRTUAL_SM3	-	-	指定轴1
Axis_2	轴2	AXIS_REF_VIRTUAL_SM3	-	-	指定轴2
Axis_3	轴3	AXIS_REF_VIRTUAL_SM3	-	-	指定轴3

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
ClearErr	清除报错	BOOL	TRUE-FALSE	FALSE	清除错误，上升沿有效。（该参数仅用于清除模块内的错误标记和错误码）
LimtAxisSpeedJump	速度跳变限制	LREAL	正数	50000	限制轴实际输出时单周期速度跳变，当速度跳变大于该值时，模块报错并限制轴脉冲输出

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
bDone	完成	BOOL	TRUE-FALSE	FALSE	TRUE：表示指令执行完成； FALSE：表示指令未执行；
bError	错误	BOOL	TRUE-FALSE	FALSE	错误状态；False-没有错误；
nErrorID	错误代码	DINT	0，正数	0	错误代码
LimitAxisMoveFlag	限制运动标记	BOOL	TRUE-FALSE	FALSE	限制轴运动标记

	布尔	位串				整数								实数		时刻、持续时间、日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Axis_0	AXIS_REF_VIRTUAL_SM3																			
Axis_1	AXIS_REF_VIRTUAL_SM3																			
Axis_2	AXIS_REF_VIRTUAL_SM3																			
Axis_3	AXIS_REF_VIRTUAL_SM3																			
ClearErr	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
LimtAxisSpeedJump	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-
bDone	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
bError	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
nErrorID	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-
LimitAxisMoveFlag	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

- **功能说明：**

  - 控制PMC600/PMC610主机脉冲轴的脉冲输出。在使用脉冲轴前必须调用，否则无法

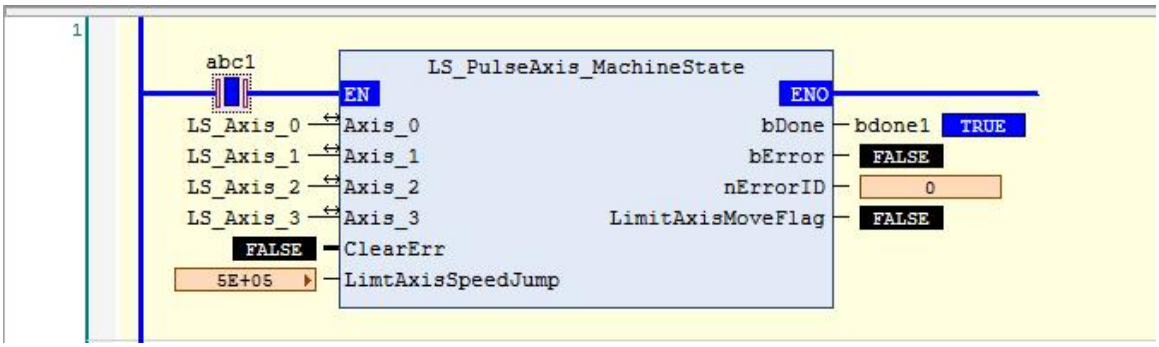
输出脉冲。

- **程序示例**

ST：当abc变为True时，使能轴脉冲输出控制模块；程序如下：

```
LS_PulseAxis_MachineState(
    Axis_0:= LS_Axis_0,
    Axis_1:= LS_Axis_1,
    Axis_2:= LS_Axis_2,
    Axis_3:= LS_Axis_3,
    ClearErr FALSE := ClearErr FALSE ,
    LimtAxisSpeedJump:= ,
    bDone TRUE => bDone TRUE ,
    bError FALSE => err FALSE ,
    nErrorID 0 => errid 0 ,
    LimitAxisMoveFlag FALSE => LimitAxisMoveFlag FALSE );
```

LD：当abc1变为True时，使能轴脉冲输出控制模块；程序如下：



- **注意事项**

➢ 这个指令由“PMC_Controller”库实现。

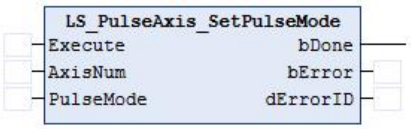
➢ 在使用该模块的时候，需要将模块放置到主任务之中。

7.12.3 LS_PulseAxis_SetPulseMode

设置 PMC600/PMC610PLC 本体脉冲轴的脉冲输出模式。

- **指令格式**

指令	名称	FB/FC	LD表现	ST表现	库

LS_PulseAxis_SetPulseMode	设置轴脉冲模式指令	FC		LS_PulseAxis_SetPulseMode(Execute:=, AxisNum:=, PulseMode:=, bDone=>, bError=>, dErrorID=>);	PMC_Controller

- **相关变量**

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Execute	启动	BOOL	TRUE-FALSE	FALSE	启动信号，TRUE时使能功能块，上升沿触发；
AxisNum	轴号	UINT	0-3	0	轴号：0-3；
PulseMode	脉冲输出模式	UINT	0-6	0	设置轴的脉冲输出模式。0-3：脉冲方向模式；4-5：双脉冲模式；6：AB相模式；

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
bDone	返回值	BOOL	TRUE-FALSE	FALSE	TRUE：指令执行完成； FALSE：指令未执行；
bError	错误	BOOL	TRUE-FALSE	FALSE	错误状态；False-没有错误；
dErrorID	错误代码	DINT	遵照数据类型	0	错误代码

	布尔	位串				整数								实数		时刻、持续时间、日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Execute	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
AxisNum	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-
Pulse Mode	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-
bDone	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
bError	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

dErrorID	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-

- **功能说明：**

  - 本指令用于设置脉冲轴的脉冲输出模式。

  - 脉冲输出模式总共有6种：0-3：脉冲方向模式；4-5：双脉冲模式；6：AB相模式。详见下表。

脉冲输出模式	正方向脉冲		负方向脉冲	
	PULSE 输出	DIR 输出端	PULSE 输出	DIR 输出端
0		高电平		低电平
1		高电平		低电平
2		低电平		高电平
3		低电平		高电平
4		高电平	高电平	
5		低电平	低电平	
6	A 相 B 相		B 相 A 相	

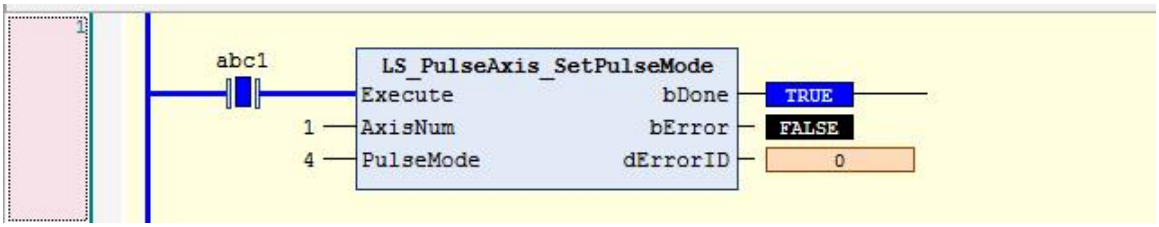
脉冲输出模式

- **程序示例**

ST：当启动信号abc变为True时，将对应脉冲轴预设的脉冲输出模式写进控制器。

```
1  LS_PulseAxis_SetPulseMode(
2      Execute TRUE := abc TRUE,
3      AxisNum 0 := 0,
4      PulseMode 2 := 2,
5      bDone TRUE => bdone TRUE,
6      bError FALSE => err FALSE,
7      dErrorID 0 => errid 0 );
8
```

LD：当启动信号abc1变为True时，将对应脉冲轴预设的脉冲输出模式写进控制器。



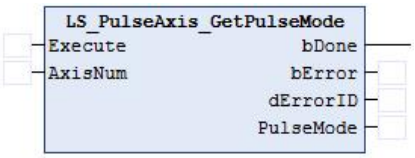
- **注意事项**

- 本指令由“PMC_Controller”库实现。
- 脉冲输出模式默认是模式0，如果实际应用不是使用这种模式，则在调用运动函数输出脉冲之前，一定要根据驱动器接收脉冲的模式，调用脉冲模式设置函数设置控制器脉冲输出模式。

7.12.4 LS_PulseAxis_GetPulseMode

读取脉冲轴的脉冲输出模式。

- **指令格式**

指令	名称	FB/FC	LD表现	ST表现	库
LS_PulseAxis_GetPulseMode	读取轴脉冲模式指令	FC		LS_PulseAxis_GetPulseMode(Execute:=, AxisNum:=, bDone=>, bError=>, dErrorID=>, PulseMode=>);	PMC_Controller库

- **相关变量**

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Execute	启动	BOOL	TRUE-FALSE	FALSE	启动信号，TRUE时使能功能块，上升沿触发；
AxisNum	轴号	UINT	0-3	0	轴号：0-3；

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
bDone	返回值	BOOL	TRUE-FALSE	FALSE	TRUE：指令执行完成； FALSE：指令未执行；
bError	错误	BOOL	TRUE-FALSE	FALSE	错误状态；False-没有错误；
dErrorID	错误代码	DINT	遵照数据类型	0	错误代码

PulseMode	脉冲输出模式	UINT	0-6	0	轴的脉冲输出模式。0-3：脉冲方向模式；4-5：双脉冲模式；6：AB相模式

	布尔	位串				整数								实数		时刻、持续时间、日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Execute	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
AxisNum	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-
bDone	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
bError	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
dErrorID	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-
PulseMode	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-

- **功能说明：**

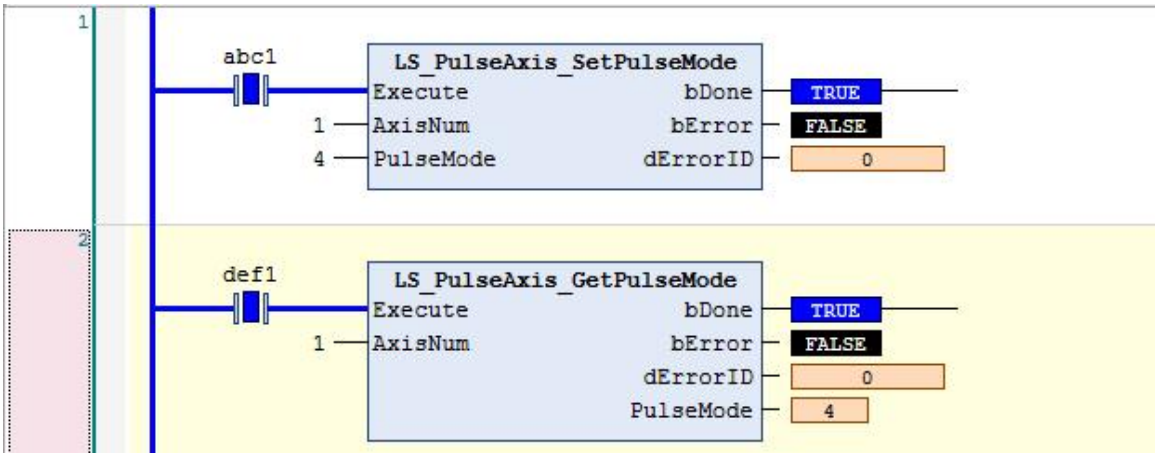
  - 指令用来读取脉冲轴的脉冲输出模式，与“LS_PulseAxis_SetPulseMode”指令配合使用。

- **程序示例**

ST：当启动信号def变为True时，读取对应脉冲轴的脉冲输出模式。

```
1  LS_PulseAxis_SetPulseMode(
2      Execute TRUE := abc TRUE,
3      AxisNum 0 := 0,
4      PulseMode 2 := 2,
5      bDone TRUE => bdone TRUE,
6      bError FALSE => err FALSE,
7      dErrorID 0 => errid 0 );
8
9  LS_PulseAxis_GetPulseMode(
10     Execute TRUE := def TRUE,
11     AxisNum 0 := 0,
12     bDone TRUE => bDone1 TRUE,
13     bError FALSE => err1 FALSE,
14     dErrorID 0 => errid1 0,
15     PulseMode 2 => PulseMode 2 );
```

LD：当启动信号def1变为True时，读取对应脉冲轴的脉冲输出模式。



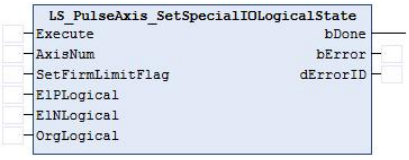
- **注意事项**

➢ 本指令由“PMC_Controller”库实现。

7.12.5 LS_PulseAxis_SetSpecialIOLogicalState

设置脉冲轴的特殊 IO 信号的启用与逻辑电平。

- **指令格式**

指令	名称	FB/FC	LD表现	ST表现	库
LS_PulseAxis_SetSpecialIOLogicalState	设置轴特殊IO信号指令	FC		LS_PulseAxis_SetSpecialIOLogicalState(Execute:=, AxisNum:=, SetFirmLimitFlag:=, ElPLogical:=, ElNLogical:=, OrgLogical:=, bDone=>, bError=>, dErrorID=>);	PMC_Controller

- **相关变量**

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述

Execute	启动	BOOL	TRUE-FALSE	FALSE	启动信号，TRUE时使能功能块，上升沿触发；
AxisNum	轴号	UINT	0-3	0	轴号：0-3
SetFirmLimitFlag	硬件限位有效标记	BOOL	TRUE-FALSE	FALSE	设置硬件限位是否有效，false表示不处理硬件限位；true表示处理硬件限位；
ElPLogical	正限位逻辑电平	BOOL	TRUE-FALSE	FALSE	设置正限位端口逻辑电平
ElNLogical	负限位逻辑电平	BOOL	TRUE-FALSE	FALSE	设置负限位端口逻辑电平
OrgLogical	原点逻辑电平	BOOL	TRUE-FALSE	FALSE	设置原点端口逻辑电平

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
bDone	返回值	BOOL	TRUE-FALSE	FALSE	TRUE：表示指令执行完成； FALSE：表示指令未执行；
bError	错误	BOOL	TRUE-FALSE	FALSE	错误状态；False-没有错误；
dErrorID	错误代码	DINT	遵照数据类型	0	错误代码

	布尔	位串				整数								实数		时刻、持续时间、日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Execute	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
AxisNum	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–
SetFirmLimitFlag	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
ElPLogical	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
ElNLogical	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
OrgLogical	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
bDone	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
bError	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
dErrorID	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–

- **功能说明：**

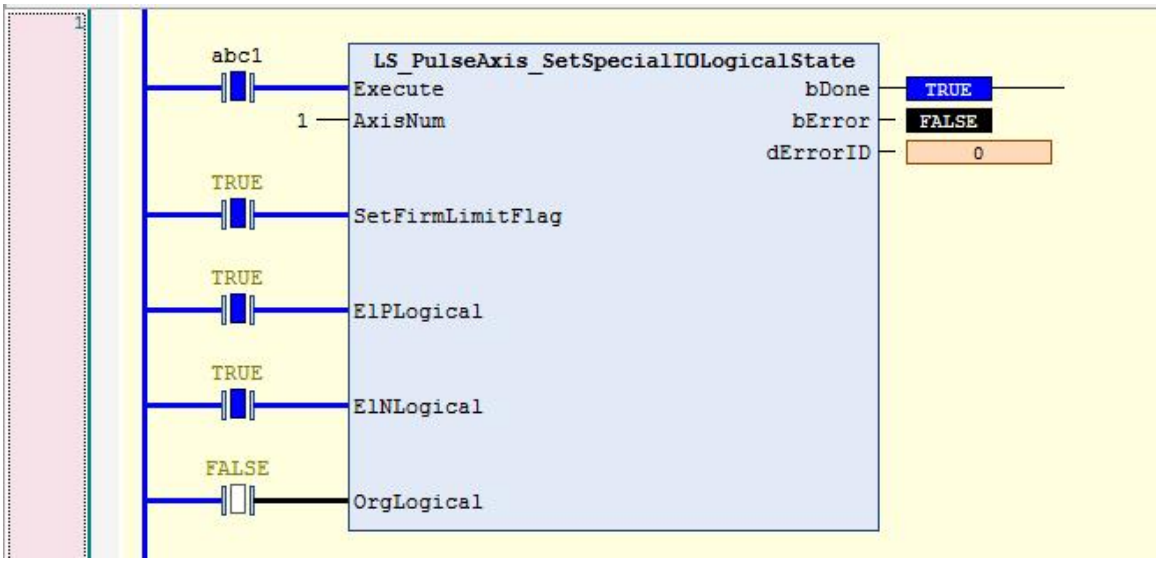
➢ 本指令用于设置脉冲轴的特殊IO信号的启用与逻辑电平。

- **程序示例**

ST：当启动信号Exec变为True时，将预设的比较器参数写入对应的通道号0。

```
LS_PulseAxis_SetSpecialIOLogicalState(
    Execute TRUE := abc TRUE,
    AxisNum 0 := 0,
    SetFirmLimitFlag TRUE := TRUE,
    ElPLogical FALSE := FALSE,
    ElNLogical FALSE := FALSE,
    OrgLogical FALSE := FALSE,
    bDone TRUE => bDone TRUE,
    bError FALSE => err FALSE,
    dErrorID 0 => errid 0 );
```

LD：当启动信号Exec1变为True时，将预设的比较器参数写入对应的通道号1。



- **注意事项**

➢ 本指令由“PMC_Controller”库实现。

➢ PMC600/PMC610为用户提供了设置硬件限位功能，用户必须根据设备的限位开关硬件接线，控制器内部限位开关的有效电平为低电平。

➢ 通过指令去设置每个轴的特殊IO信号（正限位，负限位，原点）的内部读取状态；即设置有效电平。例如：

➢ 设置ElPLogical为False时，表示正限位端口在高电平的状态下，内部读取的状态值为FALSE，正限位端口在低电平状态下，读取的状态值为TRUE；设置ElPLogical为TRUE

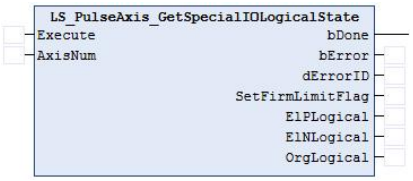
时，表示正限位端口在高电平的状态下，内部读取的状态值为TRUE。

- 当设置某个轴的硬件限位有效时，正限位只对正向运动起作用，也就是说轴在正向运动时负限位不起作用，同理负向运动正限位也不会起作用。

7.12.6 LS_PulseAxis_GetSpecialIOLogicalState

读取每个轴的限位是否有效，以及特殊IO信号（正限位，负限位，原点）的状态。

- **指令格式**

指令	名称	FB/FC	LD表现	ST表现	库
LS_PulseAxis_GetSpecialIOLogicalState	读取轴特殊IO信号指令	FC		LS_PulseAxis_GetSpecialIOLogicalState(Execute:=, AxisNum:=, bDone=>, bError=>, dErrorID=>, SetFirmLimitFlag=>, ElPLogical=>, ElNLogical=>, OrgLogical=>);	PMC_Controller

- **相关变量**

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Execute	启动	BOOL	TRUE-FALSE	FALSE	启动信号，TRUE时使能功能块，上升沿触发；
AxisNum	轴号	UINT	0-3	0	轴号：0-3

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
bDone	返回值	BOOL	TRUE-FALSE	FALSE	完成信号；False-没有完成；
bError	错误	BOOL	TRUE-FALSE	FALSE	错误状态；False-没有错误；
dErrorID	错误代码	DINT	遵照数据类型	0	错误代码
SetFirmLimitFlag	硬件限位有效标记	BOOL	TRUE-FALSE	FALSE	系统设置的硬件限位是否有效，false表示不处理硬件限位；true表示处理硬件限位；
ElP Logical	正限位逻辑电平	BOOL	TRUE-FALSE	FALSE	系统设置的正限位端口逻辑电平
ElN Logical	负限位逻辑电平	BOOL	TRUE-FALSE	FALSE	系统设置的负限位端口逻辑电平

Org Logical	原点逻辑电平	BOOL	TRUE-FALSE	FALSE	系统设置的原点端口逻辑电平

	布尔	位串				整数								实数		时刻、持续时间、日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Execute	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
AxisNum	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–
bDone	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
bError	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
dErrorID	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–
SetFirmLimitFlag	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
ElPLogical	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
ElNLogical	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
OrgLogical	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

- **功能说明：**

  ➢ 读取每个轴的限位是否有效，以及特殊IO信号（正限位，负限位，原点）的状态。

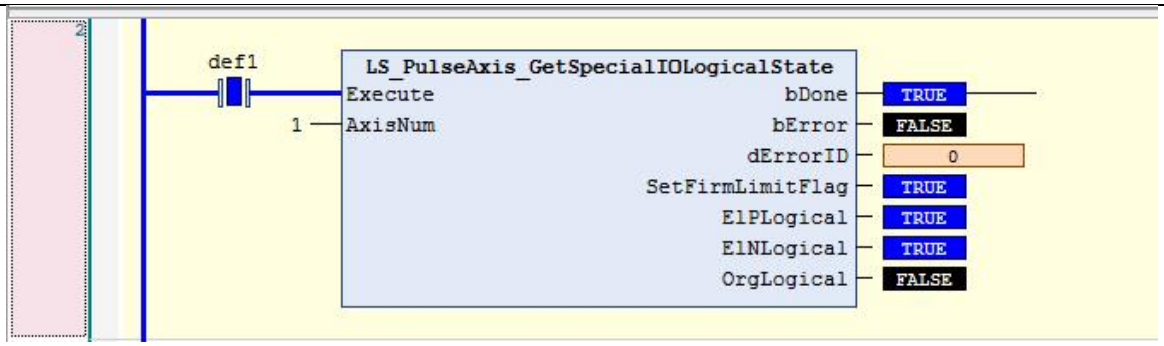
  与“LS_PulseAxis_SetSpecialIOLogicalState”指令配合进行使用。

- **程序示例**

  ST：当启动信号def变为True时，读取对应轴号0的特殊IO信号。

```
LS_PulseAxis_GetSpecialIOLogicalState(
    Execute TRUE := def TRUE,
    AxisNum 0 := 0,
    bDone TRUE => bDone1 TRUE,
    bError FALSE => err1 FALSE,
    dErrorID 0 => errid1 0,
    SetFirmLimitFlag TRUE => SetFirmLimitFlag TRUE,
    ElPLogical FALSE => ElPLogical FALSE,
    ElNLogical FALSE => ElNLogical FALSE,
    OrgLogical TRUE => OrgLogical TRUE);
```

  LD：当启动信号def1变为True时，读取对应轴号0的特殊IO信号。



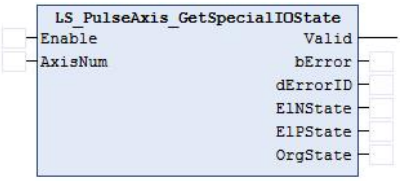
- **注意事项**

➢ 本指令由“PMC_Controller”库实现。

7.12.7 LS_PulseAxis_GetSpecialIOState

获取脉冲轴的硬件限位、原点信号的状态值。

- **指令格式**

指令	名称	FB/FC	LD表现	ST表现	库
LS_PulseAxis_GetSpecialIOState	获取轴特殊IO状态指令	FC		LS_PulseAxis_GetSpecialIOState(Enable:=, AxisNum:=, Valid=>, bError=>, dErrorID=>, ElNState=>, ElPState=>, OrgState=>);	PMC_Controller

- **相关变量**

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Enable	启动	BOOL	TRUE-FALSE	FALSE	启动信号，TRUE时使能功能块，True状态需要保持；
AxisNum	轴号	UINT	0-3	0	轴号：0-3

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
Valid	返回值	BOOL	TRUE-FALSE	FALSE	TRUE：指令执行生效中； FALSE：指令未执行；
bError	错误	BOOL	TRUE-FALSE	FALSE	错误状态；False-没有错误；
dErrorID	错误代码	DINT	遵照数据类型	0	错误代码
ElNState	负限位信号	BOOL	TRUE-FALSE	FALSE	负限位有效状态，false表示无效，true表示有效；
ElPState	正限位信号	BOOL	TRUE-FALSE	FALSE	正限位有效状态，false表示无效，true表示有效；
OrgState	原点信号	BOOL	TRUE-FALSE	FALSE	原点有效状态，false表示无效，true表示有效；

	布尔	位串				整数								实数		时刻、持续时间、日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Enable	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
AxisNum	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–
Valid	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
bError	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
dErrorID	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–
ElNState	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
ElPState	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
OrgState	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

- **功能说明：**

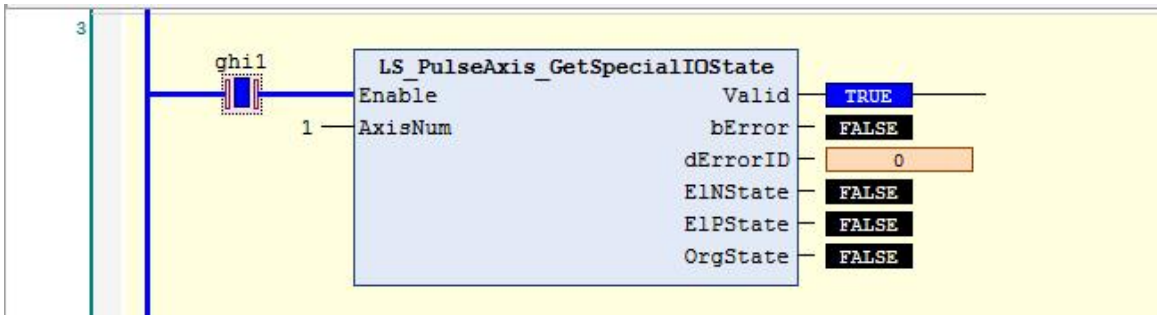
➢ 获取脉冲轴的硬件限位、原点信号的状态值。

- **程序示例**

ST：当启动信号ghi变为True时，获取对应轴号0的特殊IO信号状态值。

```
23  LS_PulseAxis_GetSpecialIOState(
24      Enable TRUE := ghi TRUE,
25      AxisNum 0 := 0,
26      Valid TRUE => Valid2 TRUE,
27      bError FALSE => err2 FALSE,
28      dErrorID 0 => errid2 0,
29      ElNState FALSE => ElNState FALSE,
30      ElPState FALSE => ElPState FALSE,
31      OrgState FALSE => OrgState FALSE);
32
```

LD：当启动信号ghi1变为True时，获取对应轴号1的特殊IO信号状态值。



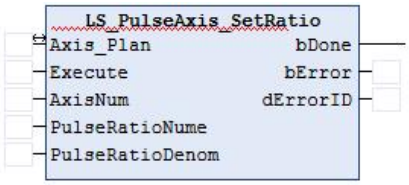
- **注意事项**

  - 本指令由“PMC_Controller”库实现。

7.12.8 LS_PulseAxis_SetRatio

设置脉冲轴的脉冲当量。

- **指令格式**

指令	名称	FB/FC	LD表现	ST表现	库
LS_PulseAxis_SetRatio	设置轴电子齿轮比指令	FC		LS_PulseAxis_SetRatio(Axis_Plan:=, Execute:=, AxisNum:=, PulseRatioNume:=, PulseRatioDenom:=, bDone=>, bError=>, dErrorID=>);	PMC_Controller

- **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	初始值	描述
Axis_Plan	脉冲规划轴	AXIS_REF_VIRTUAL_SM3	-	-	脉冲规划轴

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Execute	启动	BOOL	TRUE-FALSE	FALSE	启动信号，TRUE时使能功能块，上升沿触发；
AxisNum	轴号	UINT	0-3	0	设置规划轴号：0-3
PulseRatioNume	轴传动比分子	UDINT	正数	1	设置轴传动比分子
PulseRatioDenom	轴传动比分母	UDINT	正数	1	设置轴传动比分母

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
bDone	返回值	BOOL	TRUE-FALSE	FALSE	TRUE：表示指令执行完成； FALSE：表示指令未执行；
bError	错误	BOOL	TRUE-FALSE	FALSE	错误状态；False-没有错误；
dErrorID	错误代码	DINT	遵照数据类型	0	错误代码

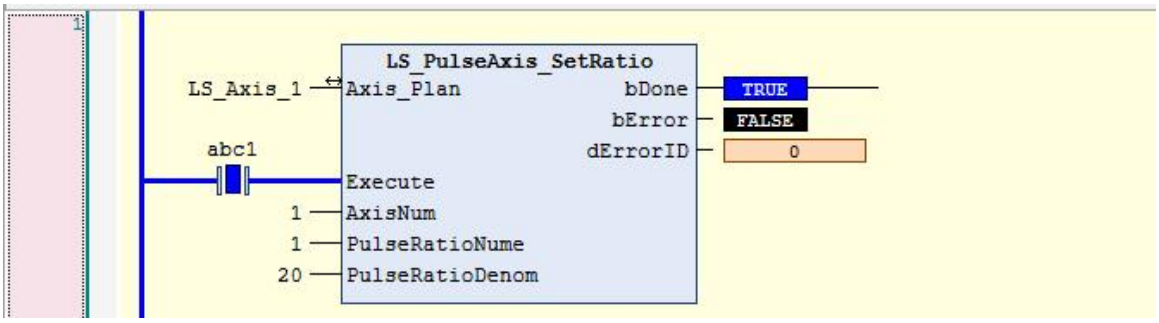
	布尔	位串				整数								实数		时刻、持续时间、日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Axis_Plan	AXIS_REF_VIRTUAL_SM3																			
Execute	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
AxisNum	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-
PulseRatioNume	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-
PulseRatioDenom	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-
bDone	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
bError	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
dErrorID	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-

- **功能说明：**
  - 本指令用于设置脉冲轴的脉冲当量。与总线轴的传动比类似，即：
  - 伺服轴传动比=传动比的分子/传动比的分母
  - 如轴0的传动比可表示为：Axis_Ratio_Nume[0]/Axis_Ratio_Denom[0]
  - 所有轴的轴传动比默认值是1。
- **程序示例**

ST：当启动信号abc变为True时，执行设置轴电子齿轮比指令。

```
LS_PulseAxis_SetRatio(
    Axis_Plan:= LS_Axis_0,
    Execute TRUE := abc TRUE,
    AxisNum 0 := 0,
    PulseRatioNume 1 := 1,
    PulseRatioDenom 10 := 10,
    bDone TRUE => bDone TRUE,
    bError FALSE => err FALSE,
    dErrorID 0 => errid 0 );
```

LD：当启动信号abc1变为True时，执行设置轴电子齿轮比指令。



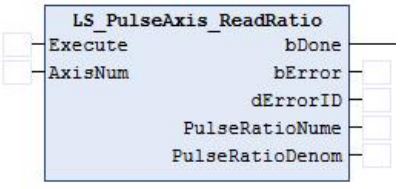
- **注意事项**

➢ 本指令由“PMC_Controller”库实现。

7.12.9 LS_PulseAxis_ReadRatio

获取脉冲轴的脉冲当量值。

- **指令格式**

指令	名称	FB/FC	LD表现	ST表现	库
LS_PulseAxis_ReadRatio	读取轴电子齿轮比指令	FC		LS_PulseAxis_ReadRatio(Execute:=, AxisNum:=, bDone=>, bError=>, dErrorID=>, PulseRatioNume=>, PulseRatioDenom=>);	PMC_Controller

- **相关变量**

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Execute	启动	BOOL	TRUE-FALSE	FALSE	启动信号，TRUE时使能功能块，上升沿触发；
AxisNum	轴号	UINT	0-3	0	轴号：0-3

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
bDone	返回值	BOOL	TRUE-FALSE	FALSE	TRUE：指令执行完成； FALSE：指令未执行；
bError	错误	BOOL	TRUE-FALSE	FALSE	错误状态；False-没有错误；
dErrorID	错误代码	DINT	遵照数据类型	0	错误代码
PulseRatioNume	轴传动比分子	UDINT	正数	1	轴传动比分子
PulseRatioDenom	轴传动比分母	UDINT	正数	1	轴传动比分母

	布尔	位串				整数								实数		时刻、持续时间、日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Execute	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
AxisNum	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-
bDone	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
bError	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
dErrorID	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-
PulseRatioNume	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-
PulseRatioDenom	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-

- **功能说明：**

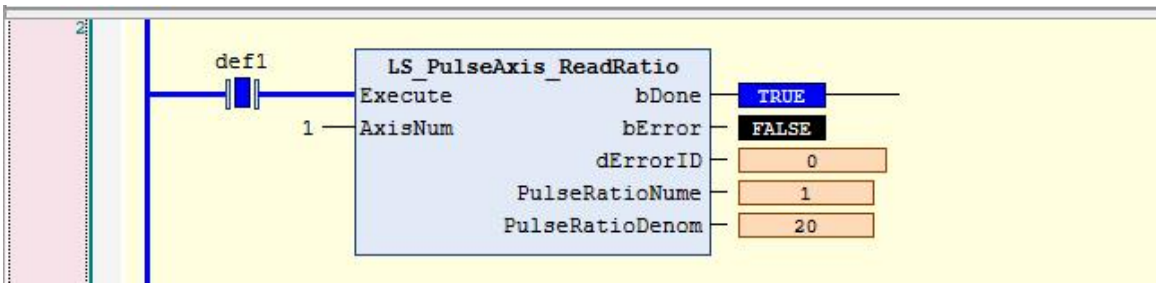
  - 获取脉冲轴的脉冲当量值。

- **程序示例**

ST：当启动信号def变为True时，获取对应轴号0的脉冲当量值。

```
11  LS_PulseAxis_ReadRatio(
12      Execute TRUE := def TRUE,
13      AxisNum 0 := 0,
14      bDone TRUE => bDone1 TRUE,
15      bError FALSE => err1 FALSE,
16      dErrorID 0 => erridl 0,
17      PulseRatioNume 1 => PulseRatioNume 1,
18      PulseRatioDenom 10 => PulseRatioDenom 10 );
19
```

LD：当启动信号def1变为True时，获取对应轴号1的脉冲当量值。



- **注意事项**

➢ 本指令由“PMC_Controller”库实现。

7.13 编码器

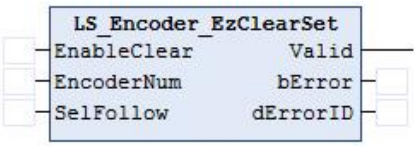
7.13.1 指令列表

指令类别	名称	FB/FC	功能
编码器指令	LS_Encoder_EzClearSet	FC	EZ 清零参数设置
	LS_Encoder_EzClearRead	FC	EZ 清零配置读取
	LS_Encoder_EzSetFilterTime	FC	EZ 滤波参数设置
	LS_Encoder_EzGetFilterTime	FC	EZ 滤波参数读
	LS_Encoder_SetVal	FC	设置编码器数值
	LS_Encoder_GetVal	FC	读取编码器数值
	LS_Encoder_SetWorkMode	FC	设置编码器工作模式
	LS_Encoder_SetAxisMove	FC	编码器数值转换轴运动

7.13.2 LS_Encoder_EzClearSet

选择上升沿或下降沿进行编码器 Z 相清零。

- **指令格式**

指令	名称	FB/FC	LD表现	ST表现	库
LS_Encoder_EzClearSet	EZ清零参数设置指令	FC		LS_Encoder_EzClearSet(EnableClear:=, EncoderNum:=, SelFollow:=, Valid=>, bError=>, dErrorID=>);	PMC_Controller

- **相关变量**

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
EnableClear	启动	BOOL	TRUE-FALSE	FALSE	启动信号，TRUE时使能功能块，TRUE状态需要保持；
EncoderNum	编码器号	UINT	0-2	0	编码器号：0-2；

SelFollow	触发沿	UINT	0，1	0	选择清零触发沿：0-下降沿清零;1-上升沿清零。（编码器Z相信号状态从无到有为上升沿）；

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
Valid	返回值	BOOL	TRUE-FALSE	FALSE	TRUE：指令执行生效中； FALSE：指令未执行；
bError	错误	BOOL	TRUE-FALSE	FALSE	错误状态；False-没有错误；
dErrorID	错误代码	DINT	遵照数据类型	0	错误代码

	布尔	位串				整数								实数		时刻、持续时间、日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
EnableClear	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Encoder Num	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–
SelFollow	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–
Valid	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
bError	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
dErrorID	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–

- **功能说明：**

  - 编码器Z相清零，是当编码器在走到Z相信号有效的位置时，编码器数值会被清零。可选择上升沿或下降沿进行数值清零。

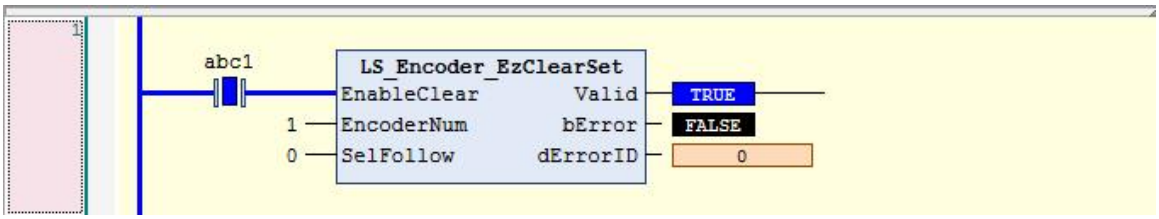
  - 当指令的“EnableClear”信号持续为TRUE时，执行编码器Z相清零。

- **程序示例**

ST：当启动信号abc变为True时，执行EZ清零参数设置指令。

```
LS_Encoder_EzClearSet(
    EnableClear TRUE := abc TRUE,
    EncoderNum 0 := 0,
    SelFollow 1 := 1,
    Valid TRUE => Valid TRUE,
    bError FALSE => err FALSE,
    dErrorID 0 => errid 0 );
```

LD：当启动信号abc1变为True时，执行EZ清零参数设置指令。



- **注意事项**

➢ 本指令由“PMC_Controller”库实现。

➢ EnableClear需要保持TRUE状态，当编码器走到Z相信号有效的位置时，编码器数值清零。

7.13.3 LS_Encoder_EzClearRead

读取编码器 Z 相清零功能的参数设置。

- **指令格式**

指令	名称	FB/FC	LD表现	ST表现	库
LS_Encoder_EzClearRead	EZ清零配置读取指令	FC	LS_Encoder_EzClearRead Execute EncoderNum bDone bError dErrorID Enable_Val SelFollow_Val	LS_Encoder_EzClearRead(Execute:=, EncoderNum:=, bDone=>, bError=>, dErrorID=>, Enable_Val=>, SelFollow_Val=>);	PMC_Controller

- **相关变量**

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述

Execute	启动	BOOL	TRUE-FALSE	FALSE	启动信号，TRUE时使能功能块，上升沿触发；
Encoder Num	编码器号	UINT	0-2	0	编码器号：0-2；

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
bDone	返回值	BOOL	TRUE-FALSE	FALSE	TRUE：指令执行完成； FALSE：指令未执行；
bError	错误	BOOL	TRUE-FALSE	FALSE	错误状态；False-没有错误；
dErrorID	错误代码	DINT	遵照数据类型	0	错误代码
Enable_Val	使能状态	BOOL	TRUE-FALSE	FALSE	读取EZ清零使能信号；TRUE表示EZ清零功能已使能；
SelFollow_Val	触发沿	UINT	0,1	0	读取配置的启动Ez清零的触发沿

	布尔	位串				整数								实数		时刻、持续时间、日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Execute	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Encoder Num	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–
bDone	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
bError	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
dErrorID	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–
Enable_Val	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
SelFollow_Val	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–

- **功能说明：**

  - 指令用来读取编码器Z相清零功能的参数设置，与“LS_Encoder_EzClearSet”指令配合使用。

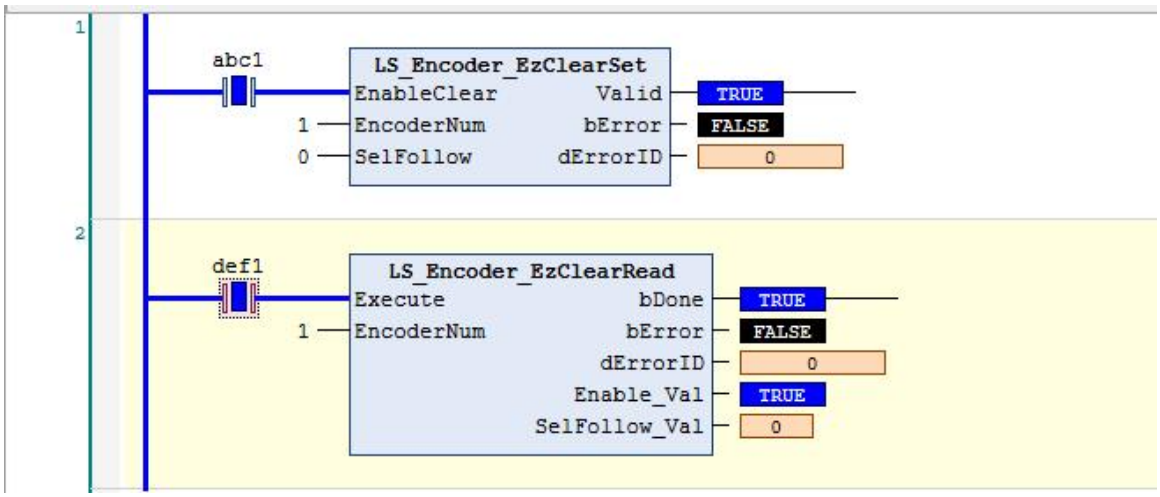
- **程序示例**

ST：当启动信号def变为True时，读取0通道编码器Z相清零功能的设置参数。

```
LS_Encoder_EzClearSet(
    EnableClear TRUE := abc TRUE,
    EncoderNum 0 := 0,
    SelFollow 1 := 1,
    Valid TRUE => Valid TRUE,
    bError FALSE => err FALSE,
    dErrorID 0 => errid 0);

LS_Encoder_EzClearRead(
    Execute TRUE := def TRUE,
    EncoderNum 0 := 0,
    bDone TRUE => bdone1 TRUE,
    bError FALSE => err1 FALSE,
    dErrorID 0 => errid1 0,
    Enable_Val TRUE => Enable_Val TRUE,
    SelFollow_Val 1 => SelFollow_Val 1);
```

LD：当启动信号def1变为True时，读取1通道编码器Z相清零功能的设置参数。



- **注意事项**

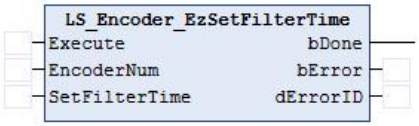
➢ 本指令由“PMC_Controller”库实现。

7.13.4 LS_Encoder_EzSetFilterTime

设置编码器 Z 相信号的滤波参数。

- **指令格式**

指令	名称	FB/FC	LD表现	ST表现	库

LS_Encoder_EzSetFilterTime	EZ滤波参数设置指令	FC		LS_Encoder_EzSetFilterTime(Execute:=, EncoderNum:=, SetFilterTime:=, bDone=>, bError=>, dErrorID=>);	PMC_Controller

- **相关变量**

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Execute	启动	BOOL	TRUE-FALSE	FALSE	启动信号，TRUE时使能功能块，上升沿触发；
EncoderNum	编码器号	UINT	0-2	0	编码器号：0-2；
SetFilterTime	滤波时间	UDINT	正数	400	设置滤波时间，单位us。建议值大于400us；

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
bDone	返回值	BOOL	TRUE-FALSE	FALSE	TRUE：指令执行完成； FALSE：指令未执行；
bError	错误	BOOL	TRUE-FALSE	FALSE	错误状态；False-没有错误；
dErrorID	错误代码	DINT	遵照数据类型	0	错误代码

	布尔	位串				整数								实数		时刻、持续时间、日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Execute	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Encoder Num	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–
SetFilter Time	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–
bDone	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
bError	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

dErrorID	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–

- **功能说明：**

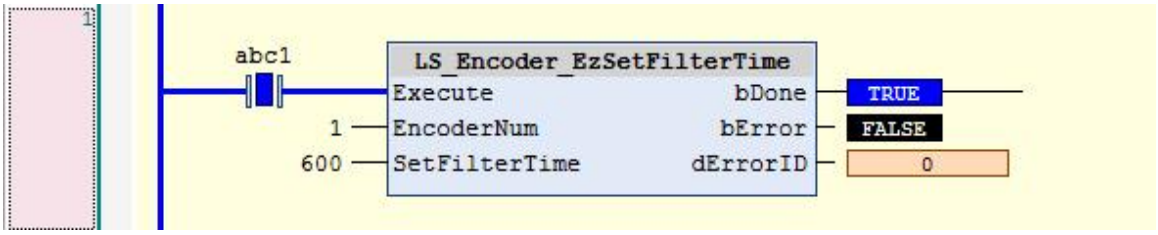
  ➢ 指令用于设置编码器Z相信号的滤波参数。

- **程序示例**

ST：当启动信号abc变为True时，执行EZ滤波参数设置指令。

```
LS_Encoder_EzSetFilterTime(
    Execute TRUE := abc TRUE,
    EncoderNum 0 := 0,
    SetFilterTime 500 := 500,
    bDone TRUE => bdone TRUE,
    bError FALSE => err FALSE,
    dErrorID 0 => errid 0);
```

LD：当启动信号abc1变为True时，执行EZ滤波参数设置指令。



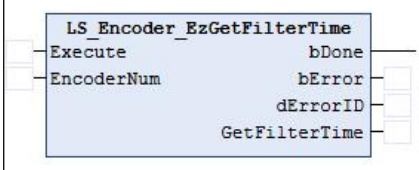
- **注意事项**

➢ 本指令由“PMC_Controller”库实现。

7.13.5 LS_Encoder_EzGetFilterTime

获取编码器端口的滤波时间参数。

- **指令格式**

指令	名称	FB/FC	LD表现	ST表现	库
LS_Encoder_EzGetFilterTime	EZ滤波参数读取指令	FC		LS_Encoder_EzGetFilterTime(Execute:=, EncoderNum:=, bDone=>, bError=>,	PMC_Controller

				dErrorID=>, GetFilterTime=>);	

- **相关变量**

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Execute	启动	BOOL	TRUE-FALSE	FALSE	启动信号，TRUE时使能功能块，上升沿触发；
EncoderNum	编码器号	UINT	0-2	0	编码器号：0-2；

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
bDone	返回值	BOOL	TRUE-FALSE	FALSE	TRUE：指令执行完成； FALSE：指令未执行；
bError	错误	BOOL	TRUE-FALSE	FALSE	错误状态；False-没有错误；
dErrorID	错误代码	DINT	遵照数据类型	0	错误代码
GetFilterTime	滤波时间	UDINT	正数	400	读取滤波时间，单位us；

	布尔	位串				整数								实数		时刻、持续时间、日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Execute	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Encoder Num	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–
bDone	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
bError	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
dErrorID	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–
GetFilterTime	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–

- **功能说明：**

  - 指令用于获取编码器端口的滤波时间参数。与指令“LS_Encoder_EzSetFilterTime”配

合使用。

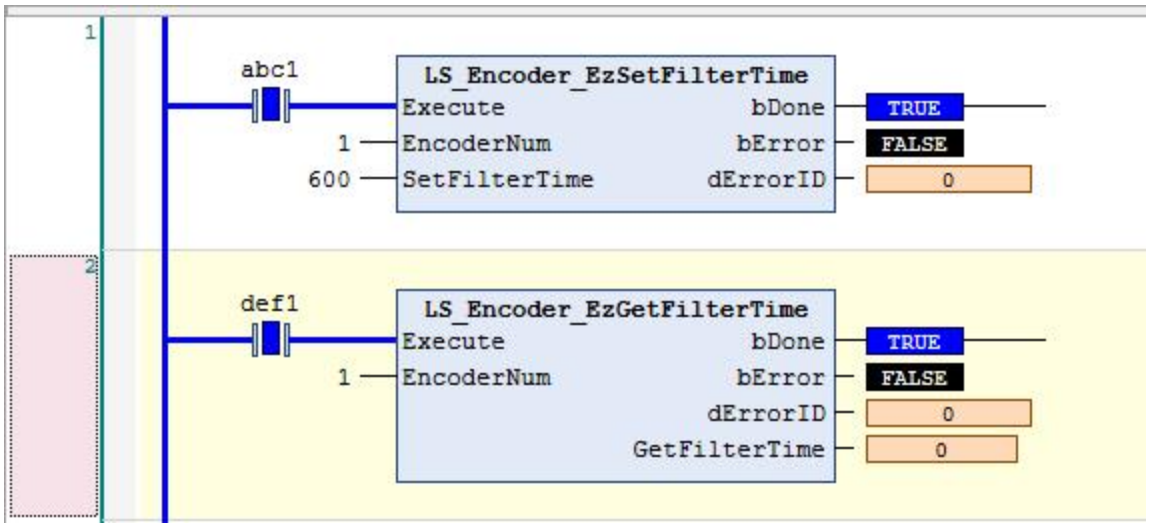
- **程序示例**

ST：当启动信号def变为True时，读取0通道编码器端口的滤波时间参数。

```
LS_Encoder_EzSetFilterTime(
    Execute TRUE := abc TRUE,
    EncoderNum 0 := 0,
    SetFilterTime 500 := 500,
    bDone TRUE => bdone TRUE,
    bError FALSE => err FALSE,
    dErrorID 0 => errid 0);

LS_Encoder_EzGetFilterTime(
    Execute TRUE := def TRUE,
    EncoderNum 0 := 0,
    bDone TRUE => bdone1 TRUE,
    bError FALSE => err1 FALSE,
    dErrorID 0 => erridl 0,
    GetFilterTime 0 => GetFilterTime 0);
    RETURN
```

LD：当启动信号def1变为True时，读取1通道编码器端口的滤波时间参数。



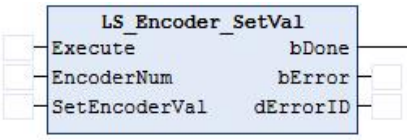
- **注意事项**

➢ 本指令由“PMC_Controller”库实现。

7.13.6 LS_Encoder_SetVal

设置编码器数值。

- **指令格式**

指令	名称	FB/FC	LD表现	ST表现	库
LS_Encoder_SetVal	设置编码器数值指令	FC		LS_Encoder_SetVal(Execute:=, EncoderNum:=, SetEncoderVal:=, bDone=>, bError=>, dErrorID=>);	PMC_Controller

- **相关变量**

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Execute	启动	BOOL	TRUE-FALSE	FALSE	启动信号，TRUE时使能功能块，上升沿触发；
EncoderNum	编码器号	UINT	0-2	0	编码器号：0—2；
SetEncoderVal	编码器数值	DINT	负数，0 正数	0	当前需要写入编码器的值，有效范围[-2147483648，2147483647]

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
bDone	返回值	BOOL	TRUE-FALSE	FALSE	完成信号；TRUE-指令执行完成；
bError	错误	BOOL	TRUE-FALSE	FALSE	错误状态；False-没有错误；
dErrorID	错误代码	DINT	遵照数据类型	0	错误代码

	布尔	位串				整数								实数		时刻、持续时间、日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Execute	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Encoder Num	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-
SetEncoderVal	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-

bDone	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
bError	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
dErrorID	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-

- **功能说明：**

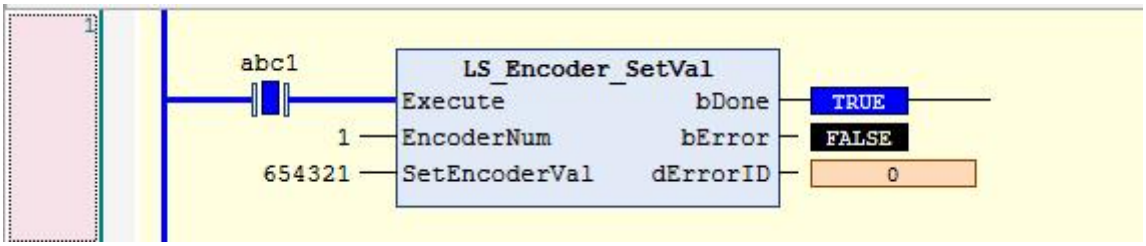
  - 指令用于设置编码器数值。

- **程序示例**

ST：当启动信号abc变为True时，执行设置编码器数值指令。

```
LS_Encoder_SetVal(
    Execute:= abc,
    EncoderNum:= 0,
    SetEncoderVal:= -12345,
    bDone=> bdone,
    bError=> err,
    dErrorID=> errid);
```

LD：当启动信号abc1变为True时，执行设置编码器数值指令。



- **注意事项**

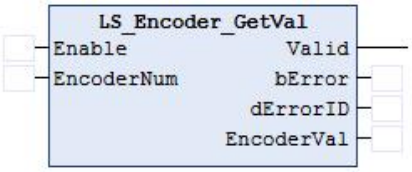
➢ 本指令由“PMC_Controller”库实现。

7.13.7 LS_Encoder_GetVal

读取编码器数值。

- **指令格式**

指令	名称	FB/FC	LD表现	ST表现	库

LS_Encoder_GetVal	读取编码器数值指令	FC		LS_Encoder_GetVal (Enable:=, EncoderNum:=, Valid=>, bError=>, dErrorID=>, EncoderVal=>);	PMC_Controller

- **相关变量**

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Enable	启动	BOOL	TRUE-FALSE	FALSE	启动信号，TRUE时使能功能块，TRUE状态需要保持；
Encoder Num	编码器号	UINT	0-2	0	编码器号：0-2；

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
Valid	返回值	BOOL	TRUE-FALSE	FALSE	TRUE：指令执行生效中； FALSE：指令未执行；
bError	错误	BOOL	TRUE-FALSE	FALSE	错误状态；False-没有错误；
dErrorID	错误代码	DINT	遵照数据类型	0	错误代码
Encoder Val	编码器数值	DINT	负数，0 正数	0	当前需要写入编码器的值，有效范围[-2147483648，2147483647]；

	布尔	位串				整数								实数		时刻、持续时间、日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Enable	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Encoder Num	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–
Valid	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
bError	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

dErrorID	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–
EncoderVal	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–

- **功能说明：**

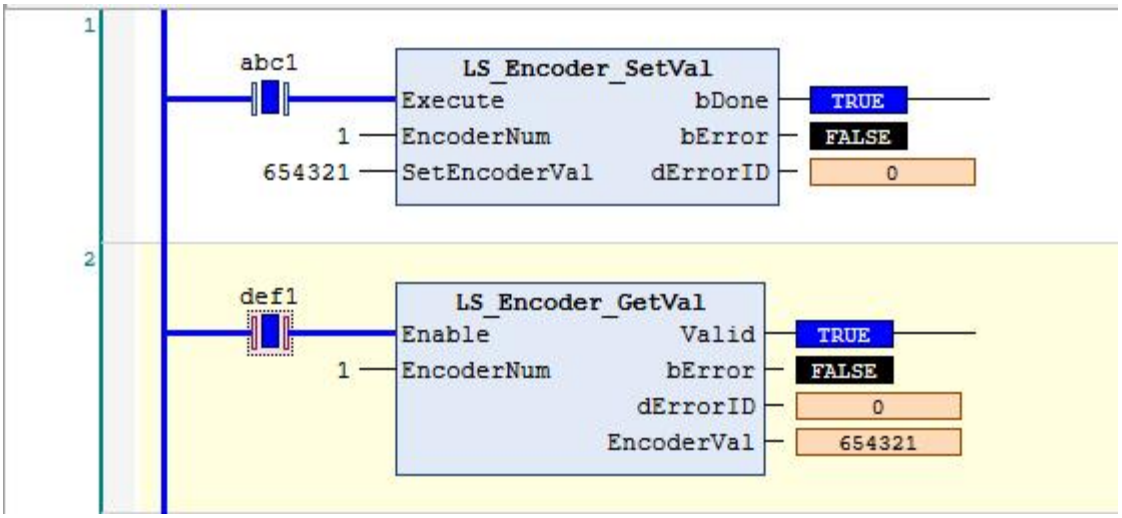
  - 指令用于读取编码器数值。与指令“LS_Encoder_SetVal”配合进行使用。

- **程序示例**

ST：当启动信号def变为True时，读取0通道编码器数值。

```
1  LS_Encoder_SetVal(
2      Execute TRUE := abc TRUE,
3      EncoderNum 0 := 0,
4      SetEncoderVal -12345 := -12345,
5      bDone TRUE => bdone TRUE,
6      bError FALSE => err FALSE,
7      dErrorID 0 => errid 0);
8
9  LS_Encoder_GetVal(
10     Enable TRUE := def TRUE,
11     EncoderNum 0 := 0,
12     Valid TRUE => valid1 TRUE,
13     bError FALSE => err1 FALSE,
14     dErrorID 0 => errid1 0,
15     EncoderVal -12345 => EncoderVal -12345);
16
```

LD：当启动信号def1变为True时，读取1通道编码器数值。



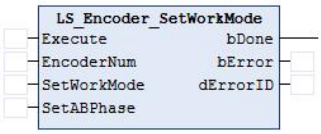
- **注意事项**

  - 本指令由“PMC_Controller”库实现。

7.13.8 LS_Encoder_SetWorkMode

设置编码器的工作模式与相位参数。

- **指令格式**

指令	名称	FB/FC	LD表现	ST表现	库
LS_Encoder_SetWorkMode	设置编码器工作模式指令	FC		LS_Encoder_SetWorkMode(Execute:=, EncoderNum:=, SetWorkMode:=, SetABPhase:=, bDone=>, bError=>, dErrorID=>);	PMC_Controller

- **相关变量**

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Execute	启动	BOOL	TRUE-FALSE	FALSE	启动信号，TRUE时使能功能块，上升沿触发；
Encoder Num	编码器号	UINT	0-2	0	编码器号：0—2；
SetWork Mode	工作模式	SINT	0-2	0	编码器的工作模式：0-四倍频模式（AB相模式）;1-脉冲方向模式;2-表示单向计数；
SetAB Phase	相位	SINT	0，1	0	设置编码器反馈的相位：0-在正方向A相比B相超前90度；1-在正方向B相比A相超前90度

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
bDone	返回值	BOOL	TRUE-FALSE	FALSE	TRUE：表示指令执行完成；FALSE：表示指令未执行；
bError	错误	BOOL	TRUE-FALSE	FALSE	错误状态；False-没有错误；
dErrorID	错误代码	DINT	遵照数据类型	0	错误代码

	布尔	位串				整数								实数		时刻、持续时间、日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Execute	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Encoder Num	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-
SetWork Mode	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-
SetAB Phase	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-
bDone	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
bError	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
dError ID	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-

- **功能说明：**

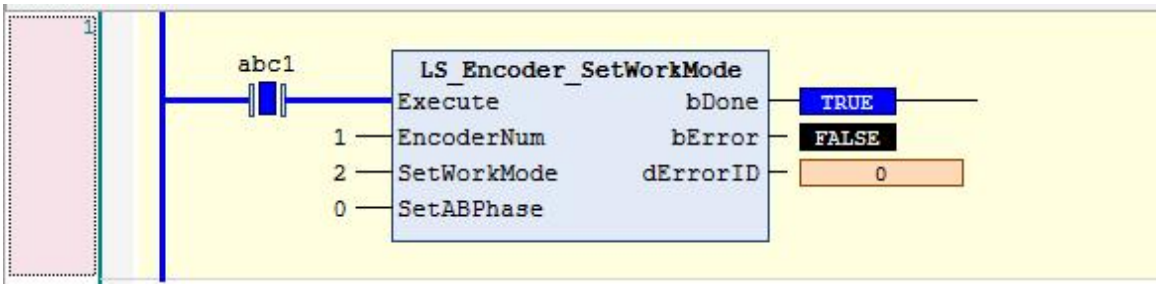
  - 原指令用于设置编码器的工作模式与相位参数，以适应与不同的脉冲型伺服、步进驱动器之间的配合使用。

- **程序示例**

ST：当启动信号abc变为True时，执行设置编码器工作模式指令。

```
LS_Encoder_SetWorkMode(
    Execute TRUE := abc TRUE,
    EncoderNum 0 := 0,
    SetWorkMode 0 := 0,
    SetABPhase 1 := 1,
    bDone TRUE => bDone TRUE,
    bError FALSE => err FALSE,
    dErrorID 0 => errid 0);
```

LD：当启动信号abc1变为True时，执行设置编码器工作模式指令。



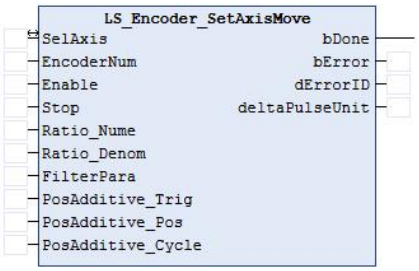
- **注意事项**

➢ 本指令由“PMC_Controller”库实现。

7.13.9 LS_Encoder_SetAxisMove

将编码器接收到的位置值，转化成为指定轴的运动。

- **指令格式**

指令	名称	FB/FC	LD表现	ST表现	库
LS_Encoder_SetAxisMove	编码器数值转换轴运动	FC		LS_Encoder_SetAxisMove(SelAxis:=, EncoderNum:=, Enable:=, Stop:=, Ratio_Nume:=, Ratio_Denom:=, FilterPara:=, PosAdditive_Trig:=, PosAdditive_Pos:=, PosAdditive_Cycle:=, bDone=>, bError=>, dErrorID=>, deltaPulseUnit=>);	PMC_IpoLib

- **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	初始值	描述
SelAxis	轴	AXIS_REF_VIRTUAL_SM3	-	-	需要执行运动的轴

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
EncoderNum	编码器号	UINT	0-2	0	编码器号：0-2；
Enable	启动	BOOL	TRUE-FALSE	FALSE	TRUE：持续将编码器信号转换为轴运动；FALSE：不执行指令；
Stop	停止	BOOL	TRUE-FALSE	FALSE	上升沿触发指令的停止执行；
Ratio_Nume	放大倍率分子	INT	1-10000	1	配置放大倍率，放大倍率分子；
Ratio_Denom	放大倍率分母	INT	1-10000	1	配置放大倍率，放大倍率分母；

FilterPara	平滑系数	UINT	1-100	5	平滑系数，参数越大，数据平滑增强；
PosAdditive_Trig	位置叠加触发	BOOL	TRUE-FALSE	FALSE	位置叠加触发信号，上升沿有效。
PosAdditive_Pos	叠加位置	LREAL	负数，0 正数	0	叠加的位置。叠加的位置不参与倍率运算
PosAdditive_Cycle	叠加周期	UDINT	正数	1	叠加的位置在几个周期内完成

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
bDone	完成	BOOL	TRUE-FALSE	FALSE	True表示指令执行完成
bError	错误	BOOL	TRUE-FALSE	FALSE	错误状态；False-没有错误；
dErrorID	错误代码	DINT	0，正数	0	错误代码
deltaPulseUnit	每周期输出的脉冲	LREAL	0，正数	0	每周期输出的脉冲

	布尔	位串				整数								实数		时刻、持续时间 日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
SelAxis	AXIS_REF_VIRTUAL_SM3																			
Encoder Num	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-
Enable	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Stop	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Ratio_ Nume	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-
Ratio_ Denom	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-
FilterPara	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-
PosAdditive_Trig	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
PosAdditive_Pos	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-
PosAdditive_Cycle	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-
bDone	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
bError	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
dErrorID	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-
deltaPulseUnit	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-

- **功能说明：**

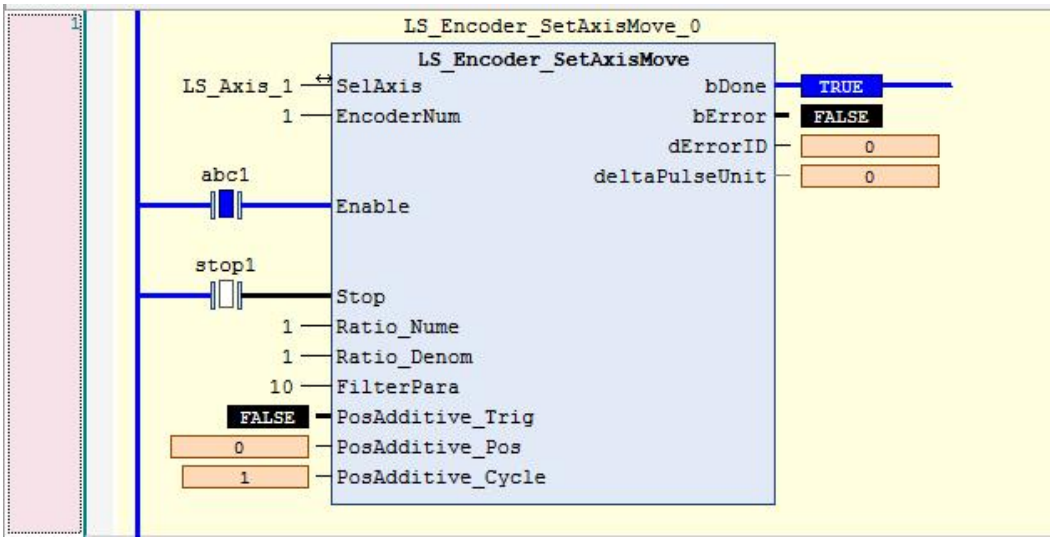
  - 将编码器接收到的位置值，转化成为指定轴的运动。
  - 可以通过调整“Ratio_Nume”和“Ratio_Denom”参数，来配置编码器位置值和轴运动的位置值的比例。
  - 可以设置“FilterPara”过滤器参数，参数值越大，数据平滑增强。
  - 有位置叠加功能，通过触发“PosAdditive_Trig”信号，让跟随轴在参数“PosAdditive_Cycle”指定的周期内，进行位置值为“PosAdditive_Pos”的位置叠加。

- **程序示例**

ST：当abc变为True时，执行编码器数值转换轴运动指令；程序如下：

```
LS_Encoder_SetAxisMove_0(
    SelAxis:= LS_Axis_0,
    EncoderNum:= 0,
    Enable:= abc,
    Stop:= stop,
    Ratio_Nume:= 1,
    Ratio_Denom:= 1,
    FilterPara:= 20,
    PosAdditive_Trig:= ,
    PosAdditive_Pos:= ,
    PosAdditive_Cycle:= ,
    bDone=> bdone,
    bError=> err,
    dErrorID=> errid,
    deltaPulseUnit=> deltaPulseUnit);
```

LD：当abc1变为True时，执行编码器数值转换轴运动指令；程序如下：



- **注意事项**

- 这个指令由“PMC_Ipolib”库实现。

7.14 高速比较

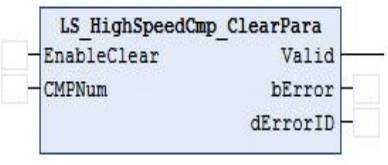
7.14.1 指令列表

指令类别	名称	FB/FC	功能
高速比较控制指令	LS_HighSpeedCmp_ClearPara	FC	清除比较器指令
	LS_HighSpeedCmp_SetWorkPara	FC	设置比较器参数指令
	LS_HighSpeedCmp_GetWorkPara	FC	读取比较器参数指令
	LS_HighSpeedCmp_SetCmpPos	FC	设置比较位置指令
	LS_HighSpeedCmp_SetLinearPara	FC	设置线性模式参数指令
	LS_HighSpeedCmp_GetLinearPara	FC	读取线性模式参数指令
	LS_HighSpeedCmp_SetFIFOPos	FC	设置 FIFO 模式 1 比较位置指令
	LS_HighSpeedCmp_SetFIFOPosType2	FC	设置 FIFO 模式 2 比较位置指令
	LS_HighSpeedCmp_GetCurState	FC	读取比较器当前状态指令

7.14.2 LS_HighSpeedCmp_ClearPara

清除高速比较器的参数和所有状态值，包括比较位置，FIFO 数据等。

- **指令格式**

指令	名称	FB/FC	LD表现	ST表现	库
LS_HighSpeedCmp_ClearPara	清除比较器指令	FC		LS_HighSpeedCmp_ClearPara(EnableClear:= , CMPNum:= , Valid=> , bError=> , dErrorID=>);	PMC_Controller库

- **相关变量**

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Enable Clear	比较器使能	BOOL	FALSE,TRUE	FALSE	TRUE：持续清除比较器

					FALSE：不执行指令
CMPNum	比较器通道	SINT	0-3	0	比较器通道号：0-3

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
Valid	返回值	BOOL	FALSE,TRUE	FALSE	完成信号；False-指令未执行；True-指令执行生效；
bError	错误	BOOL	FALSE,TRUE	FALSE	错误状态；False-没有错误；
dErrorID	错误代码	DINT	遵照数据类型	0	错误代码

	布尔	位串				整数								实数		时刻、持续时间、日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Enable Clear	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
CMP Num										√	-	-	-	-	-	-	-	-	-	-
Valid	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
bError	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
dErrorID	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-

- **功能说明：**

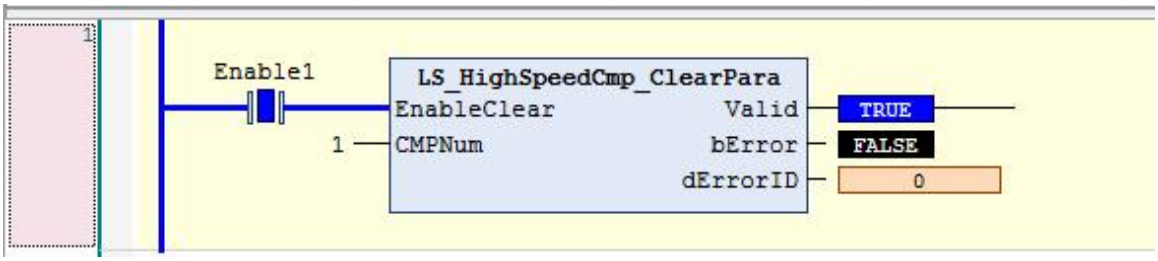
  - 清除高速比较器的参数和所有状态值，包括比较位置，FIFO数据等。

- **程序示例**

ST：当启动信号Enable变为True时，清除高速比较器通道0的参数和所有状态值。

```
LS_HighSpeedCmp_ClearPara(
    EnableClear TRUE := Enable TRUE,
    CMPNum 0 := 0,
    Valid TRUE =>valid TRUE ,
    bError FALSE => err0 FALSE,
    dErrorID 0 => errid0 0 );
RETURN
```

LD：当启动信号Enable1变为True时，清除高速比较器通道1的参数和所有状态值。



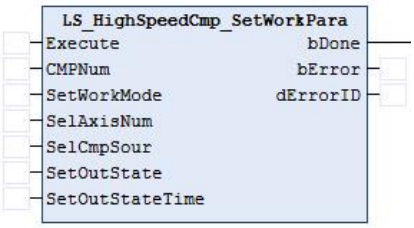
- **注意事项**

  - 本指令由“PMC_Controller”库实现。
  - 当使用对应的通道作为高速比较输出时，需先在硬件端口配置中将对应通道的功能勾选上高速比较输出；否则运行指令可能报错，错误代码50302。

7.14.3 LS_HighSpeedCmp_SetWorkPara

设置一维高速比较器的参数。

- **指令格式**

指令	名称	FB/FC	LD表现	ST表现	库
LS_HighSpeedCmp_SetWorkPara	设置比较器参数指令	FC		LS_HighSpeedCmp_SetWorkPara(Execute:= , CMPNum:= , SetWorkMode:= , SelAxisNum:= , SelCmpSour:= , SetOutState:= , SetOutStateTime:= , bDone=> , bError=> , dErrorID=>);	PMC_Controller 库

- **相关变量**

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Execute	启动	BOOL	FALSE,TRUE	FALSE	启动信号，TRUE时使能功能块，上升沿触发；
CMP Num	比较器通道	SINT	0-3	0	比较器通道：0-3
SetWorkMode	比较模式	UINT	0-6	0	设置比较器工作模式：0-不工作；1-计数器的值等于比较器；2-计数器的值小于比较器；3-计数器的值大于比较器；4-FIFO模式;5-Linear模式;6-FIFO模式2。
SelAxisNum	比较轴号	UINT	0-3	0	选择需要比较的轴号：0-3
SelCmpSour	比较源	UINT	0，1	0	选择需要比较的数据源：0-指令位置；1-反馈位置
SetOut State	输出电平逻辑	BOOL	FALSE,TRUE	FALSE	设置输出端口的有效状态：False-条件成立输出false，指示灯灭；True-条件成立输出

					True，指示灯亮；
SetOut State Time	输出时间	UINT	0-20000000	0	在FIFO模式和Linear模式下才使用。用于控制输出端口保持有效状态时间，单位us：范围0-20000000；

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
bDone	返回值	BOOL	FALSE,TRUE	FALSE	完成信号；False-没有完成；
bError	错误	BOOL	FALSE,TRUE	FALSE	错误状态；False-没有错误；
dErrorID	错误代码	DINT	遵照数据类型	0	错误代码

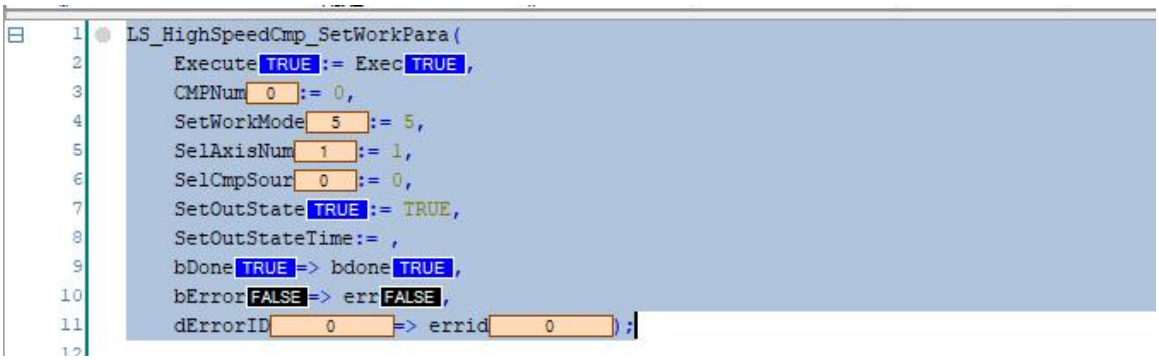
	布尔	位串				整数								实数		时刻、持续时间、日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Execute	√	-	-	-	-	–	–	–	–	–	–	–	–	–	–	–	-	-	-	-
CMPNum	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	-	-	-	-
SetWork Mode	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–
SelAxis Num	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–
SelCmp Sour	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–
SetOut State	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
SetOut State Time	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–
bDone	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
bError	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
dErrorID	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–

- **功能说明：**

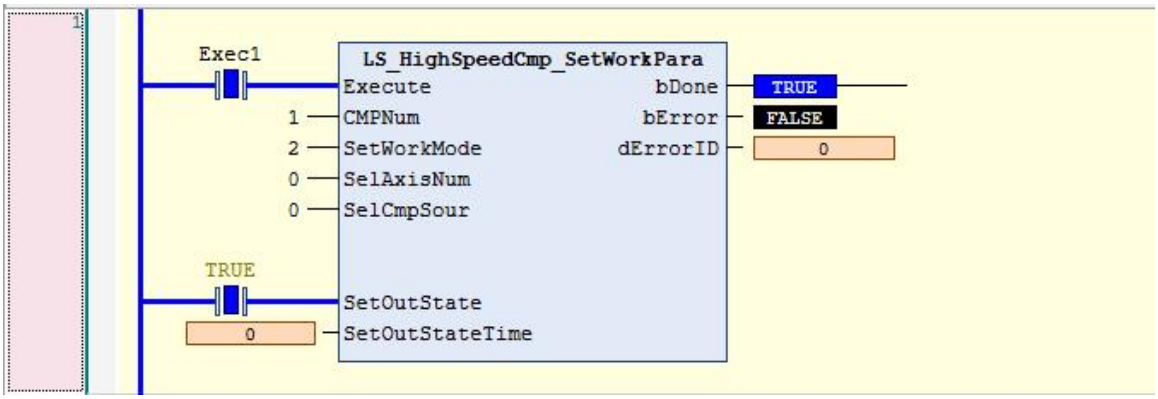
  - 设置一维高速比较器的参数；包括比较模式，比较轴号，比较源，输出电平逻辑，输出时间等。

- 程序示例

ST：当启动信号Exec变为True时，将预设的比较器参数写入对应的通道号0。



LD：当启动信号Exec1变为True时，将预设的比较器参数写入对应的通道号1。



- **注意事项**

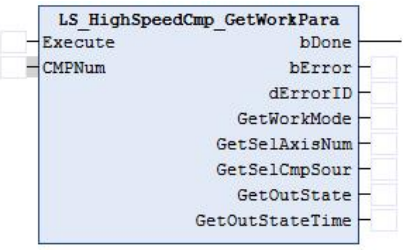
  - 本指令由“PMC_Controller”库实现。
  - PMC610有4个高速位置比较器通道，每个高速比较输出端口都可以与任意轴关联，支持多种比较模式。4个比较器通道对应的高速口分别是OUT16-OUT19，电路中使用的光耦响应频率可达200KHz。
  - 每个比较器的位置比较都是独立进行的。
  - 高速比较功能支持7种比较模式：0--不工作；1--计数器的值等于比较器；2--计数器的值小于比较器；3--计数器的值大于比较器；4--FIFO模式1（设置输出时间）；5--Linear模式；6--FIFO模式2（设置输出电平的高低）。
  - 在模式1-3中，当比较器满足设置的位置“等于”、“小于”和|“大于”的参数条件后，设定的高速比较输出口会持续保持输出状态。
  - 在FIFO模式1和Linear模式下，比较器满足条件后，SetOutStateTime控制输出端口保持有效状态的时间。
  - Linear模式的比较器的位置增量和比较点数，在指令“LS_HighSpeedCmp_SetLinearPara”中进行设置。
  - FIFO模式1的比较器点数和位置，在指令“LS_HighSpeedCmp_SetFIFOPos”中进行设置。
  - 在FIFO模式2中，比较器满足条件后，输出状态一直保持，直到遇到下一比较点。比较器的位置和输出状态，在指令“LS_HighSpeedCmp_SetFIFOPosType2”中进行设置。
  - 当使用对应的通道作为高速比较输出时，需先在硬件端口配置中将对应通道的功能勾选上高速比较输出；否则运行指令可能报错，错误代码50302。
  - 在两种FIFO及线性Linear比较模式中，执行位置比较时，每个比较点的触发是按照添加的比较点顺序执行的，即如果有一个比较点没有被触发比较动作，那么后面的比较点是不会被触发的。

➢ 高速比较输出的最大误差为扫描周期与运动速度的乘积。

7.14.4 LS_HighSpeedCmp_GetWorkPara

获取一维高速比较器的参数。

- **指令格式**

指令	名称	FB/FC	LD表现	ST表现	库
LS_HighSpeedCmp_GetWorkPara	读取比较器参数指令	FC		LS_HighSpeedCmp_GetWorkPara(Execute:= , CMPNum:= , bDone=> , bError=> , dErrorID=> , GetWorkMode=> , GetSelAxisNum=> , GetSelCmpSour=> , GetOutState=> , GetOutStateTime=>);	PMC_Controller库

- **相关变量**

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Execute	启动	BOOL	FALSE,TRUE	FALSE	启动信号，TRUE时使能功能块，上升沿触发；
CMP Num	比较器通道	SINT	0-3	0	比较器通道号：0-3

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
bDone	返回值	BOOL	FALSE,TRUE	FALSE	完成信号；False-没有完成；
bError	错误	BOOL	FALSE,TRUE	FALSE	错误状态；False-没有错误；
dErrorID	错误代码	DINT	遵照数据类型	0	错误代码
GetWork	比较模式	UINT	0-6	0	比较器工作模式：0-6。

Mode					
GetSelAxisNum	比较轴号	UINT	0-3	0	需要比较的轴号：0-3
GetSel CmpSour	比较源	UINT	0，1	0	需要比较的数据源：0-指令位置；1-反馈位置
GetOut State	输出电平逻辑	BOOL	FALSE,TRUE	FALSE	输出端口的有效状态：False-条件成立输出false，指示灯灭；True-条件成立输出True，指示灯亮；
GetOut StateTime	输出时间	UINT	0-20000000	0	在FIFO模式和Linear模式下才使用。用于控制输出端口保持有效状态时间，单位us：范围0-20000000；

	布尔	位串				整数								实数		时刻、持续时间、日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Execute	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
CMP Num	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–
bDone	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
bError	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
dErrorID	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–
GetWorkMode	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–
GetSel AxisNum	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–
GetSel CmpSour	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–
GetOut State	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
GetOut StateTime	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–

- **功能说明：**

  - 获取一维高速比较器的参数；包括比较模式，比较轴号，比较源，输出电平逻辑，输出时间等。与“LS_HighSpeedCmp_SetWorkPara”指令配合进行使用。

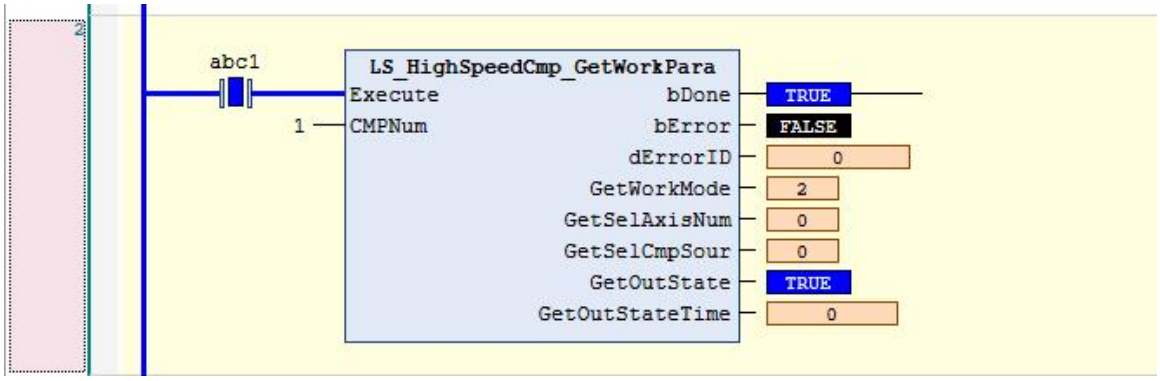
- 程序示例

ST：当启动信号abc变为True时，读取对应通道号0比较器的参数。

```
LS_HighSpeedCmp_SetWorkPara(
    Execute TRUE := Exec TRUE,
    CMPNum 0 := 0,
    SetWorkMode 5 := 5,
    SelAxisNum 1 := 1,
    SelCmpSour 0 := 0,
    SetOutState TRUE := TRUE,
    SetOutStateTime:= ,
    bDone TRUE => bdone TRUE,
    bError FALSE => err FALSE,
    dErrorID 0 => errid 0 );

LS_HighSpeedCmp_GetWorkPara(
    Execute TRUE := abc TRUE,
    CMPNum 0 := 0,
    bDone TRUE => bdone1 TRUE,
    bError FALSE => err1 FALSE,
    dErrorID 0 => erridl 0,
    GetWorkMode 5 => mode 5,
    GetSelAxisNum 1 => axisnum 1,
    GetSelCmpSour 0 => sour 0,
    GetOutState TRUE => outstate TRUE,
    GetOutStateTime 0 => statetime 0 );
```

LD：当启动信号abc1变为True时，读取对应通道号1比较器的参数。



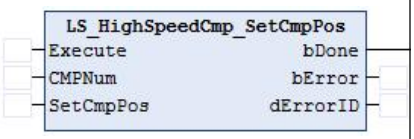
- **注意事项**

  - 本指令由“PMC_Controller”库实现。
  - 当使用对应的通道作为高速比较输出时，需先在硬件端口配置中将对应通道的功能勾选上高速比较输出；否则运行指令可能报错，错误代码50302。

7.14.5 LS_HighSpeedCmp_SetCmpPos

设置或更新比较器的比较位置。

- **指令格式**

指令	名称	FB/FC	LD 表现	ST 表现	库
LS_HighSpeedCmp_SetCmpPos	设置比较位置指令	FC		LS_HighSpeedCmp_SetCmpPos(Execute:= , CMPNum:= , SetCmpPos:= , bDone=> , bError=> , dErrorID=>);	PMC_Controller库

- **相关变量**

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Execute	启动	BOOL	FALSE,TRUE	FALSE	启动信号，TRUE时使能功能块，上升沿触发；
CMP Num	比较器通道	SINT	0-3	0	比较器通道号：0-3
SetCmp Pos	比较位置	DINT	遵照数据类型	0	设置比较位置

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
bDone	返回值	BOOL	FALSE,TRUE	FALSE	完成信号；False-没有完成；
bError	错误	BOOL	FALSE,TRUE	FALSE	错误状态；False-没有错误；
dErrorID	错误代码	DINT	遵照数据类型	0	错误代码

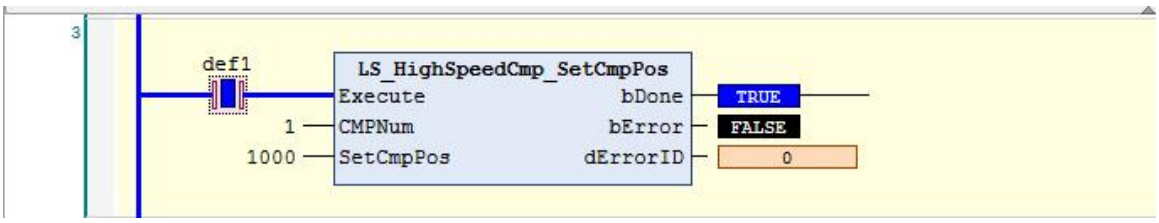
	布尔	位串				整数								实数		时刻、持续时间、日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Execute	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
CMP Num	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-
SetCmp Pos	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-
bDone	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
bError	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
dErrorID	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-

- **功能说明：**
  - 设置或更新比较器的比较位置，仅在比较模式1/2/3/5有效（除FIFO模式）。
  - 其中，在模式5线性模式下，用来更新起始比较位置；在其他模式下，更新比较位置。
- **程序示例**

ST：当启动信号def变为True时，将预设的比较位置（3000）写入对应的通道号0。

```
24  LS_HighSpeedCmp_SetCmpPos(
25      Execute TRUE := def TRUE,
26      CMPNum 0 := 0,
27      SetCmpPos 3000 := 3000,
28      bDone TRUE => bdone2 TRUE,
29      bError FALSE => err2 FALSE,
30      dErrorID 0 => errid2 0);
```

LD：当启动信号def1变为True时，将预设的比较位置（1000）写入对应的通道号1。



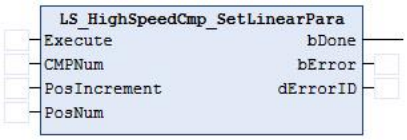
- **注意事项**

- 本指令由“PMC_Controller”库实现。
- 高速比较功能支持7种比较模式：0--不工作；1--计数器的值等于比较器；2--计数器的值小于比较器；3--计数器的值大于比较器；4--FIFO模式1（设置输出时间）；5--Linear模式；6--FIFO模式2（设置输出电平的高低）。
- FIFO模式1的比较器点数和比较位置，在指令“LS_HighSpeedCmp_SetFIFOPos”中进行设置。
- 在FIFO模式2中，比较器满足条件后，输出状态一直保持，直到遇到下一比较点。比较器的比较位置和输出状态，在指令“LS_HighSpeedCmp_SetFIFOPosType2”中进行设置。
- 当使用对应的通道作为高速比较输出时，需先在硬件端口配置中将对应通道的功能勾选上高速比较输出；否则运行指令可能报错，错误代码50302。

7.14.6 LS_HighSpeedCmp_SetLinearPara

设置一维高速比较器模式 5Linear 模式的参数。

- **指令格式**

指令	名称	FB/FC	LD 表现	ST 表现	库
LS_HighSpeedCmp_SetLinearPara	设置线性模式参数指令	FUN		LS_HighSpeedCmp_SetLinearPara(Execute:= , CMPNum:= , PosIncrement:= , PosNum:= , bDone=> , bError=> , dErrorID=>);	PMC_Controller库

- **相关变量**

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Execute	启动	BOOL	FALSE,TRUE	FALSE	启动信号，TRUE时使能功能块，上升沿触发；
CMPNum	比较器通道	SINT	0-3	0	比较器通道号：0-3
PosIncrement	位置增量	DINT	负数，0 正数	0	设置Linear（线性）模式下比较点位置增量；
PosNum	比较点数	DINT	0-128	0	设置Linear（线性）模式下比较点个数；

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
bDone	返回值	BOOL	FALSE,TRUE	FALSE	完成信号；False-没有完成；
bError	错误	BOOL	FALSE,TRUE	FALSE	错误状态；False-没有错误；
dErrorID	错误代码	DINT	遵照数据类型	0	错误代码

	布尔	位串				整数								实数		时刻、持续时间、日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Execute	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
CMP Num	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-
PosIncrement	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-
PosNum	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-
bDone	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
bError	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
dErrorID	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-

- **功能说明：**

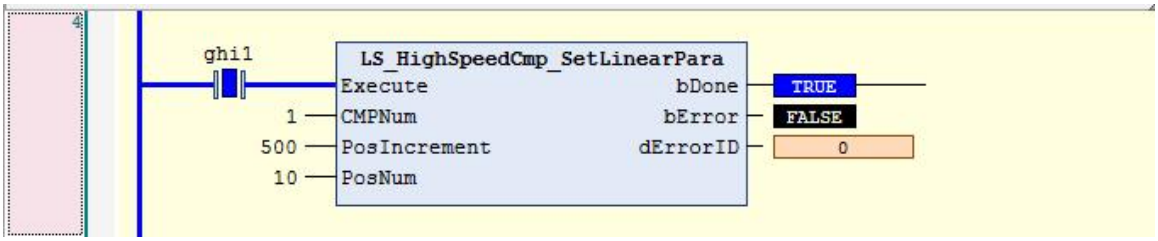
  - 设置一维高速比较器模式5Linear模式的参数，包括位置增量和比较点数。

- **程序示例**

ST：当启动信号ghi变为True时，将预设的5Linear模式的参数写入对应的通道号0。

```
LS_HighSpeedCmp_SetLinearPara(
    Execute TRUE := ghi TRUE,
    CMPNum 0 := 0,
    PosIncrement 800 := 800,
    PosNum 30 := 30,
    bDone TRUE => bdone3 TRUE,
    bError FALSE =>err3 FALSE ,
    dErrorID 0 => errid3 0 );
```

LD：当启动信号ghi1变为True时，将预设的5Linear模式的参数写入对应的通道号1。



- **注意事项**

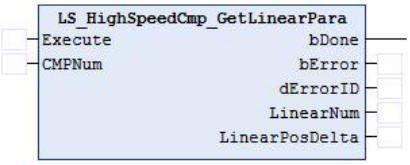
  - 本指令由“PMC_Controller”库实现。

- 当使用对应的通道作为高速比较输出时，需先在硬件端口配置中将对应通道的功能勾选上高速比较输出；否则运行指令可能报错，错误代码50302。
- 在两种FIFO及线性Linear比较模式中，执行位置比较时，每个比较点的触发是按照添加的比较点顺序执行的，即如果有一个比较点没有被触发比较动作，那么后面的比较点是不会被触发的。
- 当比较点数设置为0或1时，都会有且只有输出一个比较点，对应指令“LS_HighSpeedCmp_SetCmpPos”所设置的比较位置。
- 当设置的比较点个数大于1，且位置增量也不等于0时，则输出的高速比较点将从“LS_HighSpeedCmp_SetCmpPos”指令所设置的比较位置开始输出第1个比较点，之后如果运动方向保持，将以增量“PosIncrement”，输出剩下的“PosNum-1”个比较点。
- 如果两次高速比较输入点相距很近，则自动重合为一个脉冲。

7.14.7 LS_HighSpeedCmp_GetLinearPara

获取一维高速比较器 Linear 模式下参数。

- **指令格式**

指令	名称	FB/FC	LD表现	ST表现	库
LS_HighSpeedCmp_GetLinearPara	读取线性模式参数指令	FC		LS_HighSpeedCmp_GetLinearPara(Execute:= , CMPNum:= , bDone=> , bError=> , dErrorID=> , LinearNum=> , LinearPosDelta=>);	PMC_Controller库

- **相关变量**

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Execute	启动	BOOL	FALSE,TRUE	FALSE	启动信号，TRUE时使能功能块，上升沿触发；
CMP Num	比较器通道	SINT	0-3	0	比较器通道号：0-3

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
bDone	返回值	BOOL	FALSE,TRUE	FALSE	完成信号；False-没有完成；
bError	错误	BOOL	FALSE,TRUE	FALSE	错误状态；False-没有错误；
dErrorID	错误代码	DINT	遵照数据类型	0	错误代码
LinearNum	比较点数	DINT	0-128	0	读取Linear（线性）模式下比较点个数；
LinearPosDelta	位置增量	DINT	负数，0 正数	0	读取Linear（线性）模式下比较点位置增量；

	布尔	位串				整数								实数		时刻、持续时间、日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Execute	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
CMP Num	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-
bDone	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
bError	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
dErrorID	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-
Linear Num	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-
LinearPosDelta	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-

- **功能说明：**

  - 获取一维高速比较器Linear模式下参数；包括比较点数和位置增量。与“LS_HighSpeedCmp_SetLinearPara”指令配合进行使用。

- **程序示例**

ST：当启动信号jkl变为True时，读取对应通道号0比较器Linear模式下参数。

```
31
32  LS_HighSpeedCmp_SetLinearPara(
33      Execute TRUE := ghi TRUE,
34      CMPNum 0 := 0,
35      PosIncrement 800 := 800,
36      PosNum 30 := 30,
37      bDone TRUE => bdone3 TRUE,
38      bError FALSE =>err3 FALSE ,
39      dErrorID 0 => errid3 0 );
40
41  LS_HighSpeedCmp_GetLinearPara(
42      Execute TRUE := jkl TRUE,
43      CMPNum 0 := 0,
44      bDone TRUE => bdone4 TRUE,
45      bError FALSE => err4 FALSE,
46      dErrorID 0 => errid4 0 ,
47      LinearNum 30 => LinearNum 30 ,
48      LinearPosDelta 800 => LinearPosDelta 800 );
49
```

LD：当启动信号jkl1变为True时，读取对应通道号1比较器Linear模式下参数。



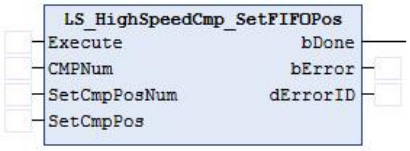
- **注意事项**

  - 本指令由“PMC_Controller”库实现。
  - 当使用对应的通道作为高速比较输出时，需先在硬件端口配置中将对应通道的功能勾选上高速比较输出；否则运行指令可能报错，错误代码50302。

7.14.8 LS_HighSpeedCmp_SetFIFOPos

设置一维高速比较器 FIFO 模式 1 的比较位置和比较点数。

- **指令格式**

指令	名称	FB/FC	LD表现	ST表现	库
LS_HighSpeedCmp_SetFIFOPos	设置FIFO模式1比较位置指令	FC		LS_HighSpeedCmp_SetFIFOPos(Execute:= , CMPNum:= , SetCmpPosNum:= , SetCmpPos:= , bDone=> , bError=> , dErrorID=>);	PMC_Controller库

- **相关变量**

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Execute	启动	BOOL	FALSE,TRUE	FALSE	启动信号，TRUE时使能功能块，上升沿触发；
CMPNum	比较器通道	SINT	0-3	0	比较器通道号：0-3
SetCmpPosNum	比较点数	DINT	0-128	0	设置FIFO模式1下高速比较点个数；
SetCmpPos	比较位置数组	ARRAY[0..127] OF DINT	遵照数据类型	-	设置FIFO模式1下高速比较的位置数组

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
bDone	返回值	BOOL	FALSE,TRUE	FALSE	完成信号；False-没有完成；
bError	错误	BOOL	FALSE,TRUE	FALSE	错误状态；False-没有错误；
dErrorID	错误代码	DINT	遵照数据类型	0	错误代码

	布尔	位串				整数								实数		时刻、持续时间、日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Execute	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
CMP Num	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-
SetCmp PosNum	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-
SetCmp Pos	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
bDone	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
bError	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
dErrorID	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-

- **功能说明：**

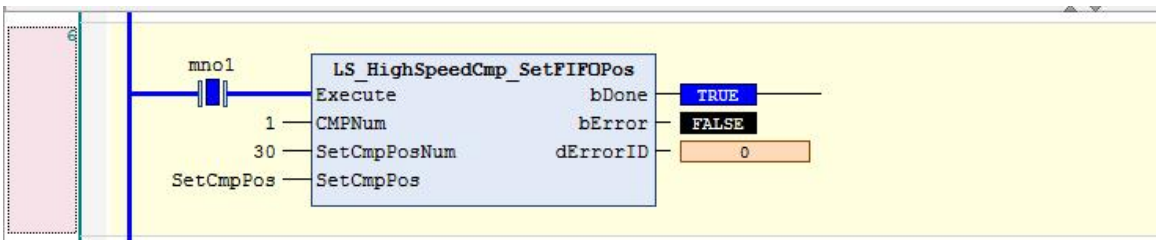
  - 设置一维高速比较器FIFO模式1的比较位置和比较点数。用于在比较模式4，即FIFO模式1（设置输出时间）下，写入多个位置点，最多可以输入128个比较点。

- **程序示例**

ST：当启动信号mno变为True时，将预设的4FIFO模式1的参数写入对应的通道号0。

```
51  LS_HighSpeedCmp_SetFIFOPos(
52      Execute TRUE := mno TRUE,
53      CMPNum 0 := 0,
54      SetCmpPosNum 50 := 50,
55      SetCmpPos:= SetCmpPos,
56      bDone TRUE => bdone5 TRUE,
57      bError FALSE => err5 FALSE,
58      dErrorID 0 => errid5 0 );
59
```

LD：当启动信号mno1变为True时，将预设的4FIFO模式1的参数写入对应的通道号1。



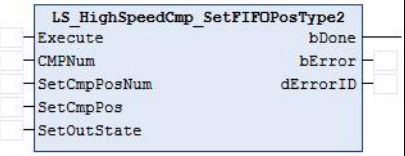
- **注意事项**

  - 本指令由“PMC_Controller”库实现。
  - 当使用对应的通道作为高速比较输出时，需先在硬件端口配置中将对应通道的功能勾选上高速比较输出；否则运行指令可能报错，错误代码50302。
  - 在两种FIFO及线性Linear比较模式中，执行位置比较时，每个比较点的触发是按照添加的比较点顺序执行的，即如果有一个比较点没有被触发比较动作，那么后面的比较点是不会被触发的。
  - 如果两次高速比较输入点相距很近，则自动重合为一个脉冲。
  - FIFO模式1是让高速比较输出电平保持一定的时间后自动关闭；FIFO模式2是设置高速比较输出电平的高低。

7.14.9 LS_HighSpeedCmp_SetFIFOPosType2

设置一维高速比较器 FIFO 模式 2 的比较位置和比较点数。

- **指令格式**

指令	名称	FB/FC	LD表现	ST表现	库
LS_HighSpeedCmp_SetFIFOPosType2	设置FIFO模式2比较位置指令	FUN		LS_HighSpeedCmp_SetFIFOPosType2(Execute:= , CMPNum:= , SetCmpPosNum:= , SetCmpPos:= , SetOutState:= , bDone=> , bError=> , dErrorID=>);	PMC_Controller库

- **相关变量**

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述

Execute	启动	BOOL	FALSE,TRUE	FALSE	启动信号，TRUE时使能功能块，上升沿触发；
CMP Num	比较器通道	SINT	0-3	0	比较器通道号：0-3
SetCmpPosNum	比较点数	DINT	0，正数	0	设置FIFO模式2下高速比较点个数；
SetCmpPos	比较位置数组	ARRAY[0..127] OF DINT	负数，0 正数	-	设置FIFO模式2下高速比较的位置数组
SetOutState	输出口状态	ARRAY[0..127] OF BOOL	FALSE,TRUE	-	设置FIFO模式2下高速输出端口的有效状态：False-条件成立输出false，指示灯灭；True-条件成立输出True，指示灯亮；

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
bDone	返回值	BOOL	FALSE,TRUE	FALSE	完成信号；False-没有完成；
bError	错误	BOOL	FALSE,TRUE	FALSE	错误状态；False-没有错误；
dErrorID	错误代码	DINT	遵照数据类型	0	错误代码

	布尔	位串				整数								实数		时刻、持续时间、日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Execute	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
CMP Num	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-
SetCmp PosNum	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-
SetCmp Pos	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
SetOut State	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
bDone	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
bError	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
dErrorID	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-

- **功能说明：**

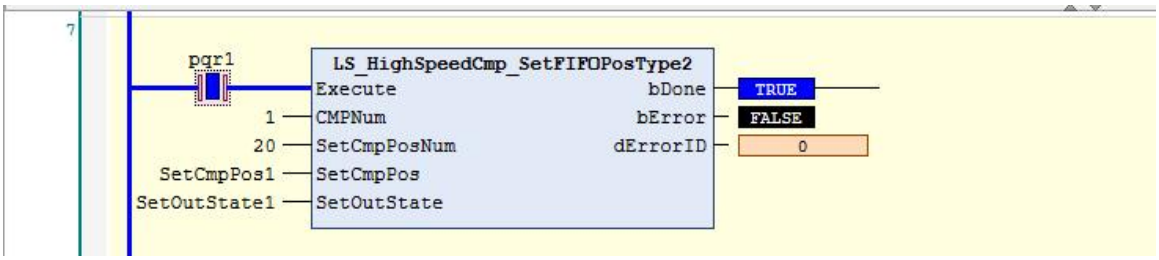
  - 设置一维高速比较器FIFO模式2的比较位置和比较点数。用于在模式6FIFO模式2（设置输出电平的高低）下，写入多个位置点。每个位置点，可以单独配置高速比较输出电平的高低状态。

- **程序示例**

ST：当启动信号pqr变为True时，将预设的6FIFO模式2的参数写入对应的通道号0。

```
60
61  LS_HighSpeedCmp_SetFIFOPosType2(
62      Execute TRUE := pqr TRUE,
63      CMPNum 0 := 0,
64      SetCmpPosNum 30 := 30,
65      SetCmpPos:= SetCmpPos1,
66      SetOutState:= SetOutState1,
67      bDone TRUE => bdone6 TRUE,
68      bError FALSE => err6 FALSE,
69      dErrorID 0 => errid6 0);
70
```

LD：当启动信号pqr1变为True时，将预设的6FIFO模式2的参数写入对应的通道号1。



- **注意事项**

  - 本指令由“PMC_Controller”库实现。

  - 在FIFO模式2中，比较器满足条件后，输出状态一直保持，直到遇到下一比较点。

  - 当使用对应的通道作为高速比较输出时，需先在硬件端口配置中将对应通道的功能勾选上高速比较输出；否则运行指令可能报错，错误代码50302。

  - 在两种FIFO及线性Linear比较模式中，执行位置比较时，每个比较点的触发是按照添加的比较点顺序执行的，即如果有一个比较点没有被触发比较动作，那么后面的比较点是不会被触发的。

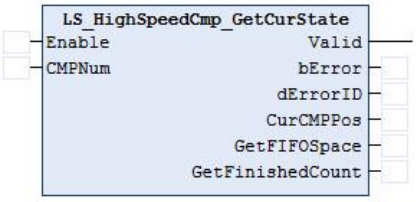
  - 如果两次高速比较输入点相距很近，则自动重合为一个脉冲。

➢ FIFO模式1是让高速比较输出电平保持一定的时间后自动关闭；FIFO模式2是设置高速比较输出电平的高低。

7.14.10 LS_HighSpeedCmp_GetCurState

读取比较器当前状态。

- **指令格式**

指令	名称	FB/FC	LD表现	ST表现	库
LS_HighSpeedCmp_GetCurState	读取比较器当前状态指令	FC		LS_HighSpeedCmp_GetCurState(Enable:= , CMPNum:= , Valid=> , bError=> , dErrorID=> , CurCMPPos=> , GetFIFOSpace=> , GetFinishedCount=>);	PMC_Controller库

- **相关变量**

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Enable	启动	BOOL	FALSE,TRUE	FALSE	启动信号，TRUE时使能功能块，TRUE状态需要保持；
CMPNum	比较器通道	SINT	0-3	0	比较器通道号：0-3

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
Valid	返回值	BOOL	FALSE,TRUE	FALSE	完成信号；False-没有完成；
bError	错误	BOOL	FALSE,TRUE	FALSE	错误状态；False-没有错误；
dErrorID	错误代码	DINT	遵照数据类型	0	错误代码
CurCMPPos	当前比较点	DINT	遵照数据类型	0	获取比较器当前比较点

GetFIFOSpace	FIFO剩余比较点数	UDINT	遵照数据类型	0	获取FIFO剩余比较点数
Get Finished Count	完成点数	UDINT	遵照数据类型	0	获取已经完成的比较点个数

	布尔	位串				整数								实数		时刻、持续时间、日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Enable	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
CMP Num	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-
Valid	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
bError	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
dErrorID	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-
CurCMP Pos	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-
GetFIFO Space	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-
Get FinishedCount	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-

- **功能说明：**

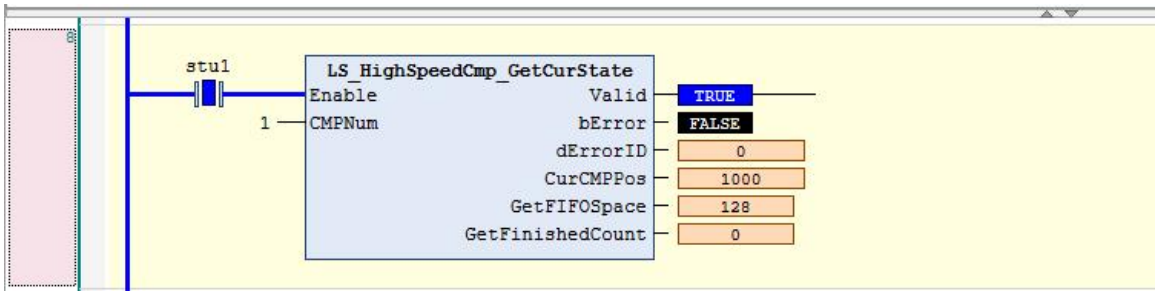
  - 获取一维高速比较器当前状态，包括当前目标比较点的位置、剩余比较点个数、已完成比较点个数等参数。

- **程序示例**

ST：当启动信号stu变为True时，读取对应通道号0高速比较器的当前状态。

```
71  LS_HighSpeedCmp_GetCurState(
72      Enable TRUE := stu TRUE,
73      CMPNum 0 := 0,
74      Valid TRUE => Valid2 TRUE,
75      bError FALSE => err7 FALSE,
76      dErrorID 0 => errid7 0,
77      CurCMPPos 5400 => CurCMPPos 5400,
78      GetFIFOSpace 128 => GetFIFOSpace 128,
79      GetFinishedCount 3 => GetFinishedCount 3 );
80
```

LD：当启动信号stu1变为True时，读取对应通道号1高速比较器的当前状态。



- **注意事项**

  - 本指令由“PMC_Controller”库实现。
  - 当使用对应的通道作为高速比较输出时，需先在硬件端口配置中将对应通道的功能勾选上高速比较输出；否则运行指令可能报错，错误代码50302。

7.15 锁存功能

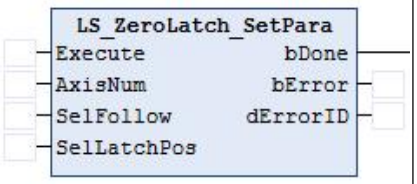
7.15.1 指令列表

指令类别	名称	FB/FC	功能
高速比较控制指令	LS_ZeroLatch_SetPara	FC	设置原点锁存参数指令
	LS_ZeroLatch_GetState	FC	读取原点锁存状态指令
	PMC_Controller.EzLatch_SetPara	FC	设置 EZ 锁存参数指令
	PMC_Controller.EzLatch_GetState	FC	读取 EZ 锁存状态指令
	LS_HighSpeedLatch_Clear	FC	清除高速比较锁存指令
	LS_HighSpeedLatch_SetSource	FC	设置高速锁存数据源指令
	LS_HighSpeedLatch_SetWorkMode	FC	设置高速锁存模式指令
	LS_HighSpeedLatch_ReadState	FC	读取高速锁存状态指令
	LS_HighSpeedLatch_ReadVal	FC	读取当前锁存值指令
	LS_HighSpeedLatch_FIFOReadVal	FC	读取 FIFO 模式锁存值指令
	LS_HighSpeedLatch_FIFOReadNumber	FC	读取 FIFO 模式锁存个数指令

7.15.2 LS_ZeroLatch_SetPara

设置原点锁存参数。

- **指令格式**

指令	名称	FB/FC	LD表现	ST表现	库
LS_ZeroLatch_SetPara	设置原点锁存参数指令	FC		LS_ZeroLatch_SetPara(Execute:= , AxisNum:= , SelFollow:= , SelLatchPos:= , bDone=> , bError=> , dErrorID=>);	PMC_Controller库

- **相关变量**

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Execute	启动	BOOL	FALSE,TRUE	FALSE	启动信号，TRUE时使能功能块，TRUE状态需要保持；
Axis Num	轴号	UINT	0-3	0	轴号：0—3；
Sel Follow	触发沿	UINT	0，1	0	选择原点锁存触发沿：0-下降沿锁存；1-上升沿锁存。（原点输入端口状态从FALSE到TRUE为上升沿）
SelLatchPos	锁存位置类型	UINT	0，1	0	选择原点锁存位置类型：0-锁存指令位置；1-锁存反馈位置；

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
bDone	返回值	BOOL	FALSE,TRUE	FALSE	完成信号；False-没有完成；
bError	错误	BOOL	FALSE,TRUE	FALSE	错误状态；False-没有错误；
dErrorID	错误代码	DINT	遵照数据类型	0	错误代码

	布尔	位串				整数								实数		时刻、持续时间、日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Execute	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
AxisNum	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–
Sel Follow	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–
SelLatch Pos	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–
bDone	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
bError	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
dErrorID	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–

- **功能说明：**

  - 原点锁存功能，用于在轴运动过程中，遇到原点信号，锁存此刻的轴位置。该

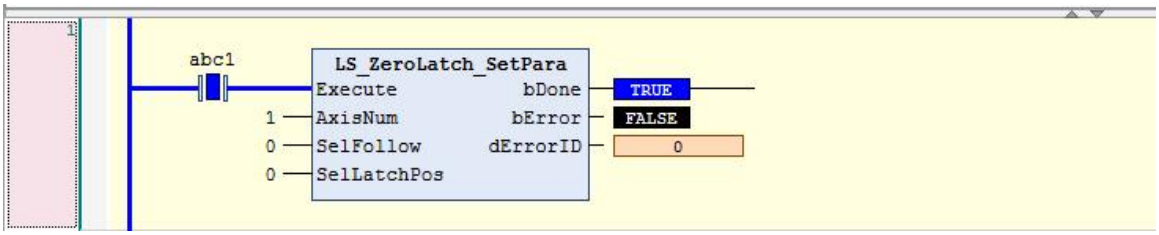
功能可以实现设备在触发原点信号时将当前位置锁存，精确回原点运动。指令用于设置原点锁存参数，设定轴号、触发沿和锁存位置类型。锁存参数设置完成后，触发指定轴号的原点信号，便能触发该轴的原点位置锁存。锁存完成信号与锁存的数据，可以在指令“LS_ZeroLatch_GetState”中获取。

- **程序示例**

ST：当启动信号abc变为True时，将预设的原点锁存参数写入控制器。

```
LS_ZeroLatch_SetPara(
    Execute TRUE := abc TRUE,
    AxisNum 0 := 0,
    SelFollow 1 := 1,
    SelLatchPos 0 := 0,
    bDone TRUE => bdone1 TRUE,
    bError FALSE => err1 FALSE,
    dErrorID 0 => erridl 0 );
```

LD：当启动信号abc1变为True时，将预设的原点锁存参数写入控制器。



- **注意事项**

➢ 本指令由“PMC_Controller”库实现。

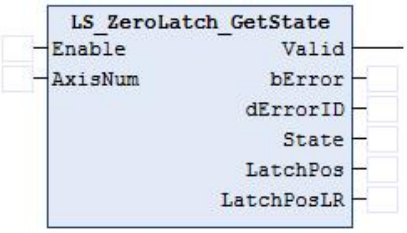
➢ 每个轴由各自的原点信号触发；锁存完成后，可以在指令“LS_ZeroLatch_GetState”中获取锁存位置。

7.15.3 LS_ZeroLatch_GetState

读取原点锁存状态参数。

- **指令格式**

指令	名称	FB/FC	LD表现	ST表现	库

LS_ZeroLatch_GetState	读取原点锁存状态指令	FC		LS_ZeroLatch_GetState(Enable:= , AxisNum:= , Valid=> , bError=> , dErrorID=> , State=> , LatchPos=> , LatchPosLR=>);	PMC_Controller库

- **相关变量**

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Enable	启动	BOOL	FALSE,TRUE	FALSE	启动信号，TRUE时使能功能块，TRUE状态需要保持；
Axis Num	轴号	UINT	0-3	0	轴号：0-3；

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
Valid	返回值	BOOL	FALSE,TRUE	FALSE	TRUE：指令执行生效中； FALSE：指令未执行；
bError	错误	BOOL	FALSE,TRUE	FALSE	错误状态；False-没有错误；
dErrorID	错误代码	DINT	遵照数据类型	0	错误代码
State	锁存状态	BOOL	FALSE,TRUE	FALSE	FALSE-锁存没有完成； True-锁存已经完成；
Latch Pos	锁存位置	DINT	负数，0 正数	0	获取锁存位置，当锁存完成标志位true时，该位置有效；（脉冲位置）
Latch PosLR	unit锁存位置	LREAL	负数，0 正数	0	包含脉冲当量的位置信息，以unit为单位；（除以脉冲当量后的值）

	布尔	位串	整数	实数	时刻、持续时间、日期、字符串

	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Enable	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
AxisNum	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–
Valid	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
bError	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
dErrorID	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–
State	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
LatchPos	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–
LatchPosLR	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–

- **功能说明：**

  - 用于获取对应轴号的原点锁存状态、锁存位置。当Enable信号持续为TRUE，则持续获取原点锁存信息。与“LS_ZeroLatch_SetPara”指令配合进行使用。

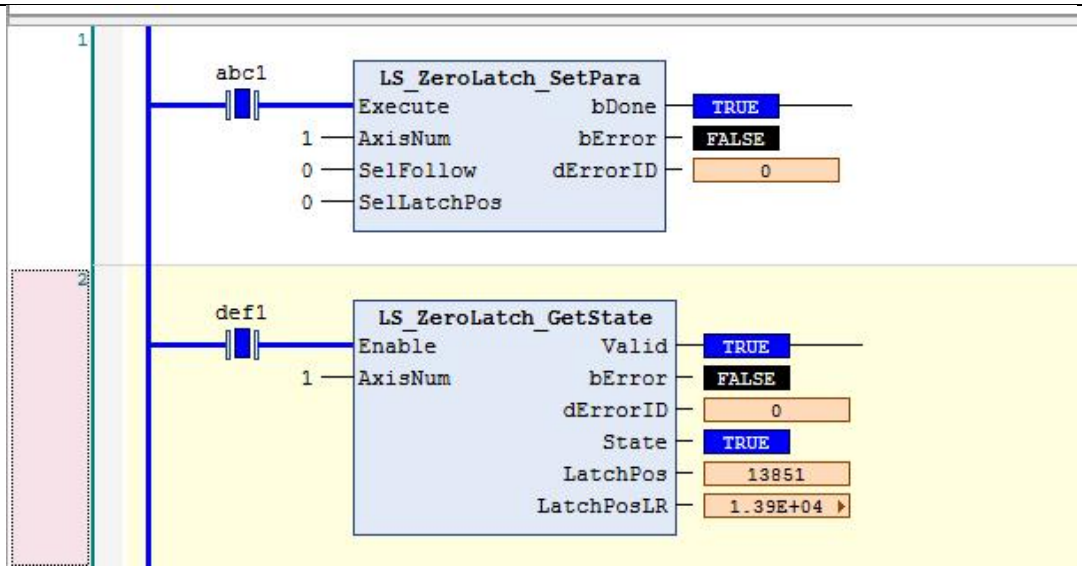
- **程序示例**

ST：当启动信号def变为True时，读取对应轴号0的原点锁存参数。

```
LS_ZeroLatch_SetPara(
    Execute:= abc,
    AxisNum:= 0,
    SelFollow:= 1,
    SelLatchPos:= 0,
    bDone=> bdone1,
    bError=> err1,
    dErrorID=> errid1);

LS_ZeroLatch_GetState(
    Enable:= def,
    AxisNum:= 0,
    Valid=> valid2,
    bError=> err2,
    dErrorID=> errid2,
    State=> State2,
    LatchPos=> LatchPos2,
    LatchPosLR=> LatchPosLR2);
```

LD：当启动信号def1变为True时，读取对应轴号1的原点锁存参数。



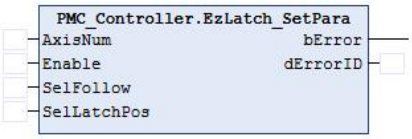
- **注意事项**

  - 本指令由“PMC_Controller”库实现。
  - 当State为TRUE时表示原点锁存完成，此时LatchPos，LatchPosLR锁存位置数据有效。
  - LatchPos锁存的值是脉冲数（单位是脉冲）；LatchPosLR锁存的值是脉冲除以脉冲当量后的值（单位是unit）

7.15.4 PMC_Controller.EzLatch_SetPara

设置 EZ 锁存参数。

- **指令格式**

指令	名称	FB/FC	LD表现	ST表现	库
PMC_Controller.EzLatch_SetPara	设置EZ锁存参数指令	FC		PMC_Controller.EzLatch_SetPara(AxisNum:= , Enable:= , SelFollow:= , SelLatchPos:= , bError=> , dErrorID=>);	PMC_Controller库

- **相关变量**

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
AxisNum	轴号	UINT	0-3	0	轴号：0—3；
Enable	启动	BOOL	FALSE,TRUE	FALSE	启动信号，TRUE时使能功能块，TRUE状态需要保持；
Sel Follow	触发沿	UINT	0，1	0	选择EZ锁存触发沿：0-下降沿锁存；1-上升沿锁存。（编码器Z相信号状态从FALSE到TRUE为上升沿）
SelLatchPos	锁存位置类型	UINT	0，1	0	选择EZ锁存位置类型：0-锁存指令位置；1-锁存反馈位置；

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
bError	错误	BOOL	FALSE,TRUE	FALSE	错误状态；False-没有错误；
dErrorID	错误代码	DINT	遵照数据类型	0	错误代码

	布尔	位串				整数								实数		时刻、持续时间、日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
AxisNum	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–
Enable	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
SelFollow	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–
SelLatch Pos	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–
bError	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
dErrorID	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–

- **功能说明：**

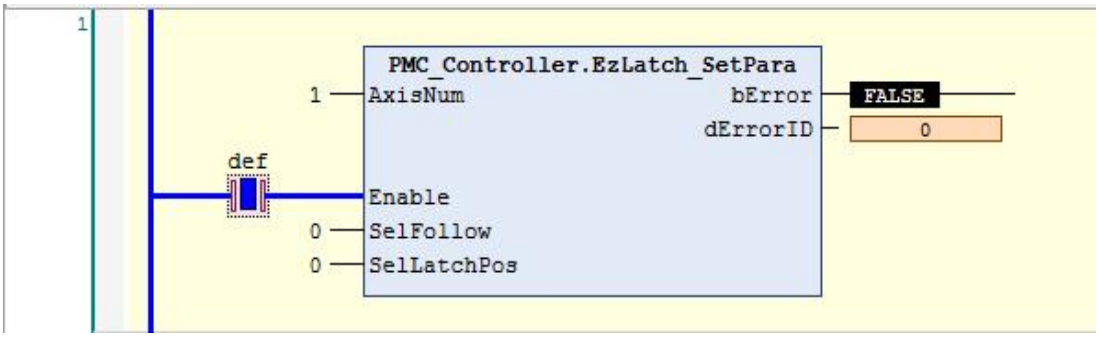
  - EZ锁存功能，用于在轴运动过程中，遇到编码器EZ信号有效则将当前位置锁存，实现精确回原点运动。指令用于设置EZ锁存参数，设定轴号、触发沿和锁存位置类型。锁存参数设置完成后，触发指定轴号的编码器Z相信号，便能触发该轴的EZ位置锁存。锁存完成信号，可以在指令“EzLatch_GetState”中获取。

- **程序示例**

ST：当启动信号abc变为True时，将预设的EZ锁存参数写入控制器。

```
PMC_Controller.EzLatch_SetPara(
    AxisNum 0 := 0,
    Enable TRUE := abc TRUE,
    SelFollow 1 := 1,
    SelLatchPos 0 := 0,
    bError FALSE => err FALSE,
    dErrorID 0 => errid 0);
```

LD：当启动信号def变为True时，将预设的EZ锁存参数写入控制器。



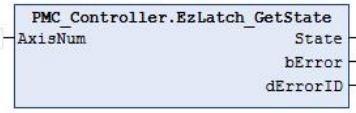
- **注意事项**

  - 本指令由“PMC_Controller”库实现。
  - 每个轴由各自的EZ信号触发。锁存完成后，锁存的位置保存于EzLatch_Axis_Pos数组中，如EzLatch_Axis_Pos[0]表示轴0的EZ锁存位置，其他轴以此类推。
  - 使用EZ锁存功能时必须外接驱动器EZ信号，未连接外部EZ信号时切不可在程序中读取EZ信号，它处于一个不稳定未知的状态。

7.15.5 PMC_Controller.EzLatch_GetState

读取 EZ 锁存状态参数。

- **指令格式**

指令	名称	FB/FC	LD表现	ST表现	库
PMC_Controller.EzLatch_GetState	读取EZ锁存状态指令	FC		PMC_Controller.EzLatch_GetState(AxisNum:= , State=> , bError=> , dErrorID=>);	PMC_Controller库

- **相关变量**

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Axis Num	轴号	UINT	0-3	0	轴号：0-3；

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
State	锁存状态	BOOL	FALSE,TRUE	FALSE	FALSE-锁存没有完成； True-锁存已经完成；
bError	错误	BOOL	FALSE,TRUE	FALSE	错误状态；False-没有错误；
dErrorID	错误代码	DINT	遵照数据类型	0	错误代码

	布尔	位串				整数								实数		时刻、持续时间、日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
AxisNum	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–
State	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
bError	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
dErrorID	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–

- **功能说明：**

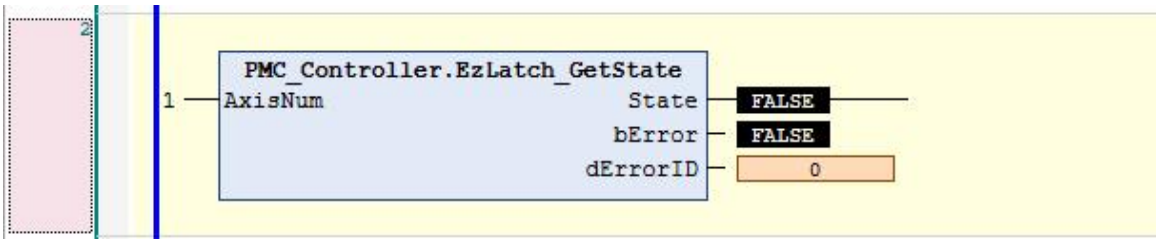
  - 用于获取对应轴号的EZ锁存状态。与“EzLatch_SetPara”指令配合进行使用。

- **程序示例**

ST：当启动程序时，读取对应轴号0的EZ锁存参数。

```
PMC_Controller.EzLatch_GetState(
    AxisNum 0 := 0,
    State FALSE => State FALSE,
    bError FALSE => err1 FALSE,
    dErrorID 0 => errid2 0);
```

LD：当启动程序时，读取对应轴号1的EZ锁存参数。



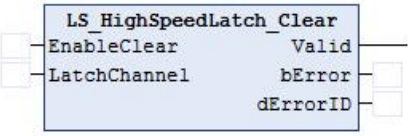
- **注意事项**

  - 本指令由“PMC_Controller”库实现。

7.15.6 LS_HighSpeedLatch_Clear

清除已经发生过的锁存及状态标记。

- **指令格式**

指令	名称	FB/FC	LD表现	ST表现	库
LS_HighSpeedLatch_Clear	清除高速比较锁存指令	FC		LS_HighSpeedLatch_Clear(EnableClear:= , LatchChannel:= , Valid=> , bError=> , dErrorID=>);	PMC_Controller库

- **相关变量**

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Enable Clear	启动	BOOL	FALSE,TRUE	FALSE	启动信号，TRUE时使能功能块，上升沿触发；
Latch Channel	通道号	UINT	0-3	0	通道号：0—3；

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
Valid	返回值	BOOL	FALSE,TRUE	FALSE	TRUE：指令执行生效中； FALSE：指令未执行；
bError	错误	BOOL	FALSE,TRUE	FALSE	错误状态；False-没有错误；
dErrorID	错误代码	DINT	遵照数据类型	0	错误代码

	布尔	位串				整数								实数		时刻、持续时间、日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Enable Clear	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Latch Channel	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–
Valid	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
bError	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
dErrorID	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–

- **功能说明：**

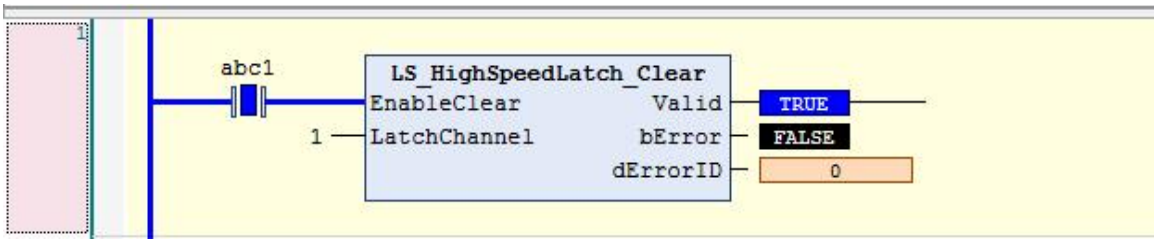
  - 清除已经发生过的锁存及状态标记，使其恢复到初始状态。调用高速锁存，必须先调用该指令，以避免锁存器的一些残留数据。

- **程序示例**

ST：当启动信号abc变为True时，执行清除高速比较锁存指令。

```
LS_HighSpeedLatch_Clear(
    EnableClear:= abc,
    LatchChannel:= 0,
    Valid=> Valid,
    bError=> err,
    dErrorID=> errid);
```

LD：当启动信号abc1变为True时，执行清除高速比较锁存指令。



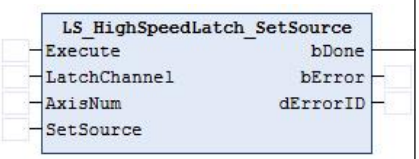
- **注意事项**

  - 本指令由“PMC_Controller”库实现。

7.15.7 LS_HighSpeedLatch_SetSource

设置高速锁存数据源。

- **指令格式**

指令	名称	FB/FC	LD表现	ST表现	库
LS_HighSpeedLatch_SetSource	设置高速锁存数据源指令	FC		LS_HighSpeedLatch_SetSource(Execute:= , LatchChannel:= , AxisNum:= , SetSource:= ,	PMC_Controller库

				bDone=> , bError=> , dErrorID=>);	

- **相关变量**

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Execute	启动	BOOL	FALSE,TRUE	FALSE	启动信号，TRUE时使能功能块，上升沿触发；
Latch Channel	通道号	UINT	0-3	0	通道号：0—3；
Axis Num	轴号	UINT	0-3	0	轴号：0—3；
Set Source	锁存源	UINT	0，1	0	设置锁存源：0-锁存指令位置；1-锁存编码器反馈位置；（编码器轴设置为1）

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
bDone	返回值	BOOL	FALSE,TRUE	FALSE	完成信号；False-没有完成；
bError	错误	BOOL	FALSE,TRUE	FALSE	错误状态；False-没有错误；
dErrorID	错误代码	DINT	遵照数据类型	0	错误代码

	布尔	位串				整数								实数		时刻、持续时间、日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Execute	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Latch Channel	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-
AxisNum	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-
Set Source	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-
bDone	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

bError	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
dErrorID	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-

- **功能说明：**

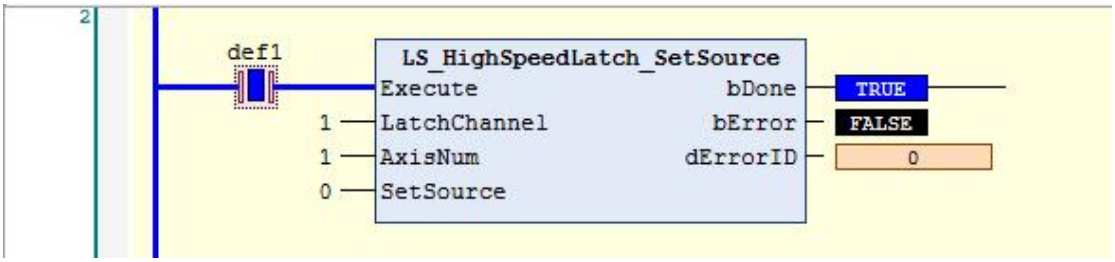
  - 设置高速锁存数据源，当SetSource为0时，则锁存源是指令位置，即当锁存触发时，锁存器里面记录的是指令位置；当SetSource为1时，则锁存源是编码器反馈位置，即当锁存触发时，锁存器里面记录的是编码器反馈位置。指令只需要调用一个周期，触发参数的执行。

- **程序示例**

ST：当启动信号def变为True时，将预设的高速锁存参数写入控制器。

```
LS_HighSpeedLatch_SetSource(
    Execute TRUE := def TRUE,
    LatchChannel 0 := 0,
    AxisNum 0 := 0,
    SetSource 0 := 0,
    bDone TRUE => bDone1 TRUE,
    bError FALSE => err1 FALSE,
    dErrorID 0 => erridl 0 );
```

LD：当启动信号def1变为True时，将预设的高速锁存参数写入控制器。



- **注意事项**

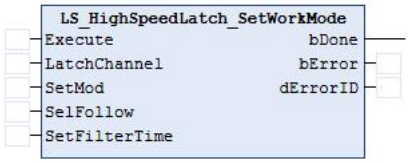
  - 本指令由“PMC_Controller”库实现。

  - 当使用对应的通道作为高速锁存输入时，需先在硬件端口配置中将对应通道的功能勾选上高速锁存输入；否则运行指令可能报错，错误代码50302。

7.15.8 LS_HighSpeedLatch_SetWorkMode

设置高速锁存的模式、触发沿等参数。

- **指令格式**

指令	名称	FB/FC	LD表现	ST表现	库
LS_HighSpeedLatch_SetWorkMode	设置高速锁存模式指令	FC		LS_HighSpeedLatch_SetWorkMode(Execute:= , LatchChannel:= , SetMod:= , SelFollow:= , SetFilterTime:= , bDone=> , bError=> , dErrorID=>);	PMC_Controller库

- **相关变量**

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Execute	启动	BOOL	FALSE,TRUE	FALSE	启动信号，TRUE时使能功能块，上升沿触发；
LatchChannel	通道号	UINT	0-3	0	通道号：0-3
SetMod	锁存模式	UINT	0，1	0	设置锁存模式：0-单次锁存；1-连续锁存（系统底层FIFO数量为256）。参数值需要保持；
Sel Follow	锁存触发沿	UINT	0-2	0	选择高速锁存触发沿：0-下降沿锁存；1-上升沿锁存；2-任意沿锁存（上升沿或下降沿）。输入端口状态从FALSE到TRUE为上升沿；
SetFilterTime	-	-	-	-	参数保留

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
bDone	返回值	BOOL	FALSE,TRUE	FALSE	完成信号；False-没有完成；
bError	错误	BOOL	FALSE,TRUE	FALSE	错误状态；False-没有错误；
dErrorID	错误代码	DINT	遵照数据类型	0	错误代码

	布尔	位串				整数								实数		时刻、持续时间、日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Execute	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Latch Channel	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-
SetMod	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-
Sel Follow	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-
SetFilterTime	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
bDone	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
bError	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
dErrorID	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-

- **功能说明：**

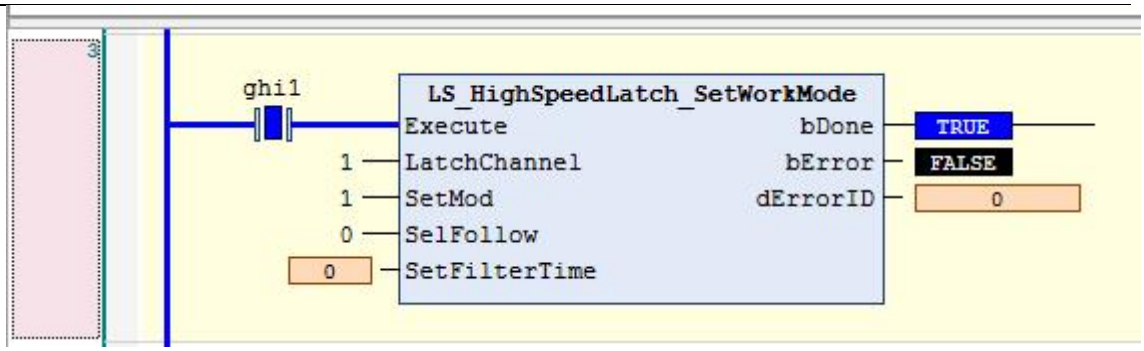
  - 设置高速锁存的模式、触发沿等参数。指令只需要调用一个周期，触发参数的执行。有两种锁存模式，SetMod为0是单次锁存，即设置完锁存器之后，锁存器只能够触发一次，且第一次有效，触发多次是不会产生更多的锁存数值的；SetMod为1时是连续锁存，即设置完锁存器之后，锁存器可以连续被触发，且最多为256个。

- **程序示例**

ST：当启动信号ghi变为True时，将预设的高速锁存参数写入对应的通道号0。

```
17  LS_HighSpeedLatch_SetWorkMode(
18      Execute TRUE := ghi TRUE,
19      LatchChannel 0 := 0,
20      SetMod 0 := 0,
21      SelFollow 1 := 1,
22      SetFilterTime:= ,
23      bDone TRUE => bDone2 TRUE,
24      bError FALSE => err2 FALSE,
25      dErrorID 0 => errid2 0 );
```

LD：当启动信号ghi1变为True时，将预设的高速锁存参数写入对应的通道号1。



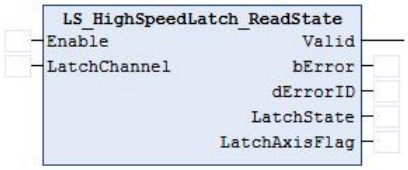
- **注意事项**

  - 本指令由“PMC_Controller”库实现。
  - 当使用对应的通道作为高速锁存输入时，需先在硬件端口配置中将对应通道的功能勾选上高速锁存输入；否则运行指令可能报错，错误代码50302。

7.15.9 LS_HighSpeedLatch_ReadState

读取单次锁存的锁存状态。

- **指令格式**

指令	名称	FB/FC	LD表现	ST表现	库
LS_HighSpeedLatch_ReadState	读取高速锁存状态指令	FC		LS_HighSpeedLatch_ReadState(Enable:= , LatchChannel:= , Valid=> , bError=> , dErrorID=> , LatchState=> , LatchAxisFlag=>);	PMC_Controller库

- **相关变量**

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Enable	启动	BOOL	FALSE,TRUE	FALSE	启动信号，TRUE时使能功能块，TRUE状态需要保持；
Latch Channel	通道号	UINT	0-3	0	通道号：0-3；

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
Valid	返回值	BOOL	FALSE,TRUE	FALSE	TRUE：指令执行生效中； FALSE：指令未执行；
bError	错误	BOOL	FALSE,TRUE	FALSE	错误状态；False-没有错误；
dErrorID	错误代码	DINT	遵照数据类型	0	错误代码
Latch State	锁存状态	BOOL	FALSE,TRUE	FALSE	FALSE-锁存没有完成； True-锁存已经完成；
Latch AxisFlag	锁存轴	UDINT	0-3	0	已经锁存有数据的轴标记

	布尔	位串				整数								实数		时刻、持续时间、日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Enable	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Latch Channel	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-
Valid	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
bError	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
dErrorID	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-
LatchState	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Latch AxisFlag	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-

- **功能说明：**

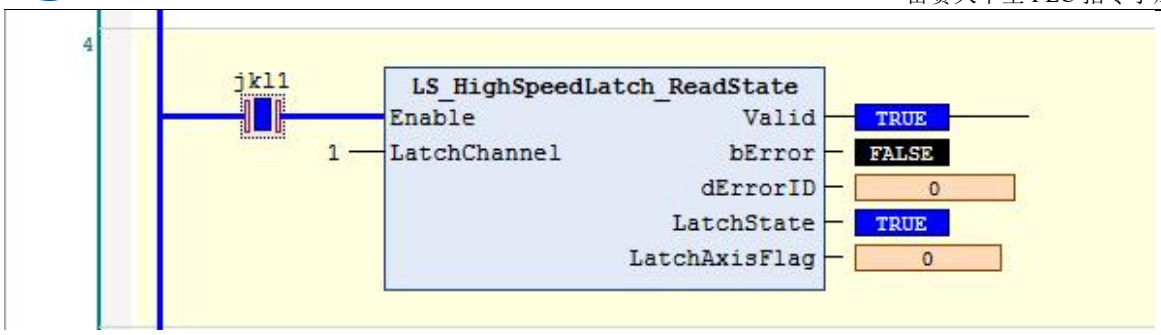
  - 用来读取单次锁存的锁存状态，不适用于连续锁存。锁存完成后，锁存状态LatchState将变成TRUE，且参数LatchAxisFlag也将显示当前锁存器锁存的轴号。

- **程序示例**

ST：当启动信号jkl变为True时，读取对应通道号0的原点锁存参数。

```
26
27  LS_HighSpeedLatch_ReadState(
28      Enable TRUE := jkl TRUE,
29      LatchChannel 0 := 0,
30      Valid TRUE => Valid2 TRUE,
31      bError FALSE => err3 FALSE,
32      dErrorID 0 => errid3 0,
33      LatchState FALSE => LatchState FALSE,
34      LatchAxisFlag 0 => LatchAxisFlag 0 );
35
```

LD：当启动信号jkl1变为True时，读取对应通道号1的原点锁存参数。



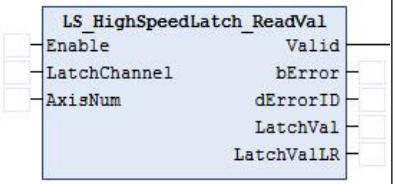
- **注意事项**

  - 本指令由“PMC_Controller”库实现。
  - 当使用对应的通道作为高速锁存输入时，需先在硬件端口配置中将对应通道的功能勾选上高速锁存输入；否则运行指令可能报错，错误代码50302。

7.15.10 LS_HighSpeedLatch_ReadVal

读取单次锁存的当前锁存值。

- **指令格式**

指令	名称	FB/FC	LD表现	ST表现	库
LS_HighSpeedLatch_ReadVal	读取当前锁存值指令	FC		LS_HighSpeedLatch_ReadVal(Enable:= , LatchChannel:= , AxisNum:= , Valid=> , bError=> , dErrorID=> , LatchVal=> , LatchValLR=>);	PMC_Controller库

- **相关变量**

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Enable	启动	BOOL	FALSE,TRUE	FALSE	启动信号，TRUE时使能功能块，TRUE状态需要保持；

Latch Channel	通道号	UINT	0-3	0	通道号：0-3；
Axis Num	轴号	UINT	0-3	0	轴号：0-3；

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
Valid	返回值	BOOL	FALSE,TRUE	FALSE	TRUE：指令执行生效中； FALSE：指令未执行；
bError	错误	BOOL	FALSE,TRUE	FALSE	错误状态；False-没有错误；
dErrorID	错误代码	DINT	遵照数据类型	0	错误代码
Latch Val	锁存值	DINT	遵照数据类型	0	返回锁存值（脉冲数）
Latch ValLR	锁存值	LREAL	遵照数据类型	0	返回锁存值（除以脉冲当量后的值）

	布尔	位串				整数								实数		时刻、持续时间、日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Enable	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Latch Channel	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-
AxisNum	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-
Valid	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
bError	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
dErrorID	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-
LatchVal	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-
LatchValLR	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-

- **功能说明：**

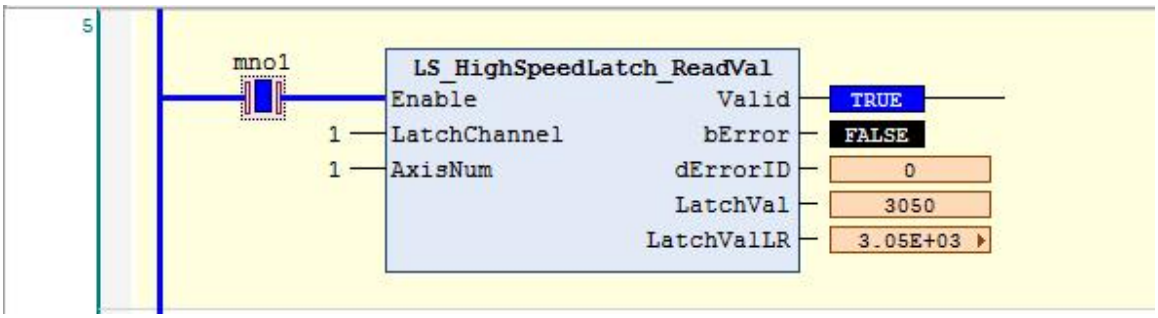
  - 用来读取单次锁存的当前锁存值，不适用于连续锁存。锁存完成后，将输出获取的锁存数值。

- **程序示例**

ST：当启动信号mno变为True时，读取对应通道号0的当前锁存值。

```
35
36  LS_HighSpeedLatch_ReadVal(
37      Enable TRUE := mno TRUE,
38      LatchChannel 0 := 0,
39      AxisNum 0 := 0,
40      Valid TRUE => Valid3 TRUE,
41      bError FALSE =>err4 FALSE ,
42      dErrorID 0 => errid4 0,
43      LatchVal 0 => LatchVal 0,
44      LatchValLR 0 => LatchValLR 0 );
45
```

LD：当启动信号mno1变为True时，读取对应通道号1的当前锁存值。



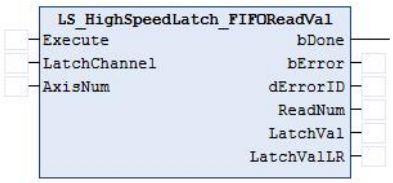
- **注意事项**

  - 本指令由“PMC_Controller”库实现。
  - 当使用对应的通道作为高速锁存输入时，需先在硬件端口配置中将对应通道的功能勾选上高速锁存输入；否则运行指令可能报错，错误代码50302。
  - LatchVal锁存的值是脉冲数（单位是脉冲）；LatchValLR锁存的值是脉冲除以脉冲当量后的值（单位是unit）。

7.15.11 LS_HighSpeedLatch_FIFOReadVal

读取连续锁存模式下当前的锁存值。

- **指令格式**

指令	名称	FB/FC	LD表现	ST表现	库
LS_HighSpeedLatch_FIFOReadVal	读取FIFO模式锁存值指令	FC		LS_HighSpeedLatch_FIFOReadVal(Execute:= , LatchChannel:= , AxisNum:= , bDone=> , bError=> , dErrorID=> , ReadNum=> , LatchVal=> , LatchValLR=>);	PMC_Controller库

- **相关变量**

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Execute	启动	BOOL	FALSE,TRUE	FALSE	启动信号，TRUE时使能功能块，上升沿触发；
Latch Channel	通道号	UINT	0-3	0	通道号：0-3；
Axis Num	轴号	UINT	0-3	0	轴号：0-3；

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
bDone	返回值	BOOL	FALSE,TRUE	FALSE	TRUE：指令执行完成； FALSE：指令未执行；
bError	错误	BOOL	FALSE,TRUE	FALSE	错误状态；False-没有错误；
dErrorID	错误代码	DINT	遵照数据类型	0	错误代码
ReadNum	读取锁存个数	UDINT	0，正数	0	返回本次读取的锁存个数
Latch	锁存值	ARRAY[0..31]	遵照数据类	0	返回锁存值（脉冲数）

Val		OF DINT	型		
Latch ValLR	锁存值	ARRAY[0..31] OF LREAL	遵照数据类型	0	返回锁存值（除以脉冲当量后的值）

	布尔	位串				整数								实数		时刻、持续时间、日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Execute	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Latch Channel	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–
AxisNum	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–
bDone	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
bError	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
dErrorID	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–
Read Num	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–
LatchVal	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Latch ValLR	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

- **功能说明：**

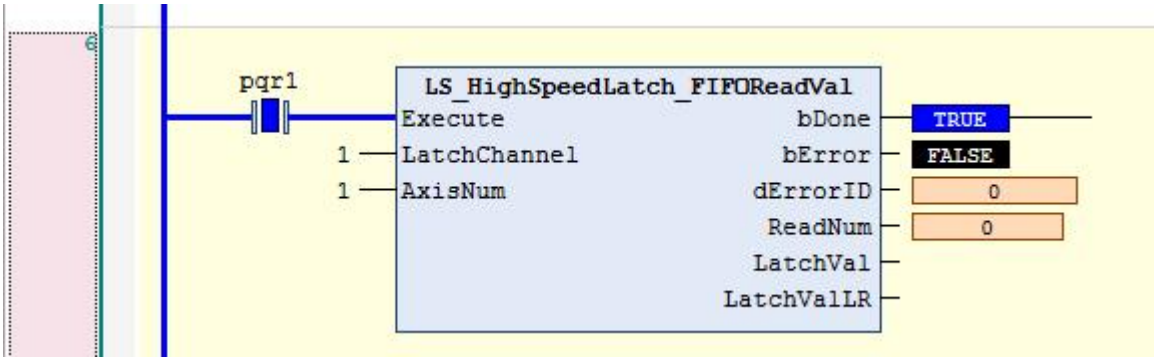
  - 用来读取连续锁存模式下当前的锁存值。当锁存缓存里有超过32个数值的时候，每触发一次Execute的上升沿，将从缓存里读取32个锁存值到LatchVal和LatchValLR之中，且保持先进先出的队列原则，即先锁存的数值将优先获取。如果缓存里的数值已经取完，则继续触发锁存值了，将获取到重复的数值。

- **程序示例**

ST：当启动信号pqr变为True时，读取FIFO模式的当前锁存值。

```
45
46  LS_HighSpeedLatch_FIFOReadVal(
47      Execute TRUE := pqr TRUE,
48      LatchChannel 0 := 0,
49      AxisNum 0 := 0,
50      bDone TRUE => bDone5 TRUE,
51      bError FALSE => err5 FALSE,
52      dErrorID 0 => errid5 0,
53      ReadNum 0 => ReadNum 0,
54      LatchVal=> LatchVal1,
55      LatchValLR=> LatchValLR1);
```

LD：当启动信号pqr1变为True时，读取FIFO模式的当前锁存值。



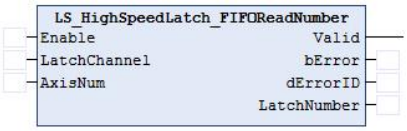
- **注意事项**

  - 本指令由“PMC_Controller”库实现。
  - 当使用对应的通道作为高速锁存输入时，需先在硬件端口配置中将对应通道的功能勾选上高速锁存输入；否则运行指令可能报错，错误代码50302。

7.15.12 LS_HighSpeedLatch_FIFOReadNumber

读取连续锁存当前的数据个数。

- **指令格式**

指令	名称	FB/FC	LD表现	ST表现	库
LS_HighSpeedLatch_FIFOReadNumber	读取FIFO模式锁存个数指令	FC		LS_HighSpeedLatch_FIFOReadNumber(Enable:= , LatchChannel:= , AxisNum:= , Valid=> , bError=> , dErrorID=> , LatchNumber=>);	PMC_Controller库

- **相关变量**

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Enable	启动	BOOL	FALSE,TRUE	FALSE	启动信号，TRUE时使能功能块，TRUE状态需要保持；
Latch Channel	通道号	UINT	0-3	0	通道号：0-3；
Axis Num	轴号	UINT	0-3	0	轴号：0-3；

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
Valid	返回值	BOOL	FALSE,TRUE	FALSE	TRUE：指令执行生效中； FALSE：指令未执行；

bError	错误	BOOL	FALSE,TRUE	FALSE	错误状态；False-没有错误；
dErrorID	错误代码	DINT	遵照数据类型	0	错误代码
LatchNumber	读取锁存个数	UDINT	0，正数	0	返回当前已经锁存的数据个数

	布尔	位串				整数								实数		时刻、持续时间、日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Enable	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Latch Channel	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–
AxisNum	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–
Valid	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
bError	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
dErrorID	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–
Latch Number	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–

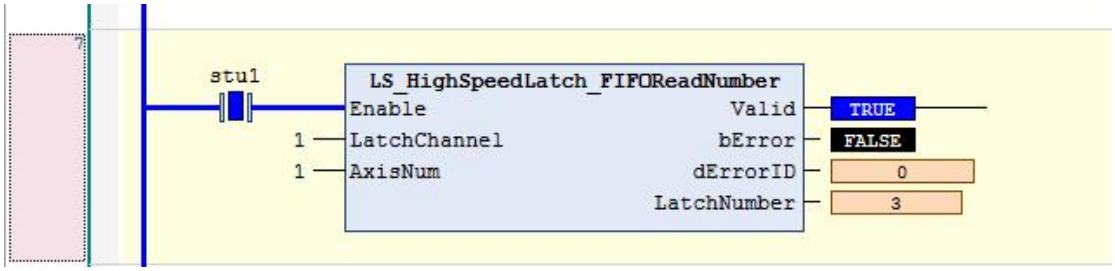
- **功能说明：**
  - 用来读取连续锁存当前的数据个数，可以通过指令实时获取当前连续锁存所运行的进度。
  - 只要指令的Enable为TRUE，将实时获取当前的锁存状态，一般在使用连续锁存时，可以持续调用指令获取数据。

- **程序示例**

ST：当启动信号stu变为True时，读取FIFO模式的当前锁存个数。

```
59      Enable TRUE := stu TRUE,
60      LatchChannel 1 := 1,
61      AxisNum 1 := 1,
62      Valid TRUE => Valid6 TRUE,
63      bError FALSE => err6 FALSE,
64      dErrorID 0 => errid6 0,
65      LatchNumber 1 => LatchNumber 1 );
```

LD：当启动信号stu1变为True时，读取FIFO模式的当前锁存个数。



- **注意事项**

  - 本指令由“PMC_Controller”库实现。
  - 当使用对应的通道作为高速锁存输入时，需先在硬件端口配置中将对应通道的功能勾选上高速锁存输入；否则运行指令可能报错，错误代码50302。

8 特殊指令

8.1SD卡文件操作指令

8.1.1 指令列表

指令类别	名称	FB/FC	功能
SD 卡文件操作	LS_CopyFromSDCard	FB	从 SD 卡拷贝文件至本地
	LS_CopyToSDCard	FB	将本地文件拷贝至 SD 卡
	LS_GetSDCardInformation	FB	获取 SD 卡下的文件信息

8.1.2 LS_CopyFromSDCard

从 SD 卡拷贝文件至本地。

- **指令格式**

指令	名称	FB/FC	LD表现	ST表现	库
LS_CopyFromSDCard	从 SD 卡拷贝文件至本地	FB		LS_CopyFromSDCard(xExecute:=, strSourceFileName:=, strDestFileName:=, xDone=>, xError=>, diErrorID=>, dwCopyedSizeBytes=>);	MC_SysLib

- **相关变量**

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述

xExecute	高电平有效	BOOL	TRUE-FALSE	FALSE	上升沿有效，状态从 FALSE 切换到 TRUE，执行功能块。
strSourceFileName	源文件名	STRING(32)	-	‘tmp.data’	源文件名。
strDestFileName	目标文件名	STRING(32)	-	‘tmp2.data’	目标文件名。

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
xDone	输出有效	BOOL	TRUE-FALSE	FALSE	TRUE表示写入数据完成。
xError	错误	BOOL	TRUE-FALSE	FALSE	FALSE——没有错误， TRUE——执行错误。
diErrorID	错误 ID	DINT	遵循数据类型	0	错误码。0表示没有错误，1表示找不到SD卡，2表示创建本地文件错误，3表示打开SD卡文件错误，4表示文件名异常，16表示文件不存在。
dwCopyedSizeBytes	拷贝的字节长度	DWORD	遵循数据类型	0	需要拷贝的字节长度。

- **功能说明**

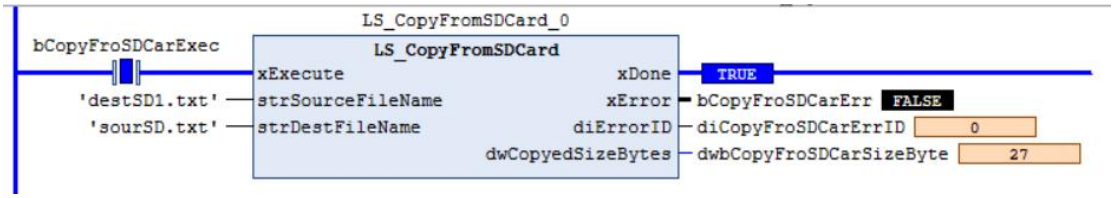
  - 将xEnable设为TRUE时，该功能块进入可运行状态；将xEnable设为FALSE，功能块不运行。
  - 从SD卡拷贝的文件保存在本地UsrData文件中。

- **程序实例**

ST：

```
LS_CopyFromSDCard_0(xExecute TRUE := bCopyFroSDCarExec TRUE,
        strSourceFileName 'destSD1.tx  := 'destSD1.txt',
        strDestFileName 'sourSD.txt' := 'sourSD.txt',
        xDone TRUE => bCopyFroSDCarDone TRUE,
        xError FALSE => bCopyFroSDCarErr FALSE,
        diErrorID 0 => diCopyFroSDCarErrID 0,
        dwCopyedSizeBytes 27 => dwbCopyFroSDCarSizeByte 27);
```

LD：



8.1.3 LS_CopyToSDCard

将本地文件拷贝至 SD 卡。

- **指令格式**

指令	名称	FB/FC	LD表现	ST表现	库
LS_CopyToSDCard	将本地文件拷贝至 SD 卡	FB		LS_CopyToSDCard(xExecute:=, strSourceFileName:=, strDestFileName:=, xDone=>, xError=>, diErrorID=>, dwCopyedSizeBytes=>);	MC_SysLib

- **相关变量**

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
xExecute	高电平有效	BOOL	TRUE-FALSE	FALSE	上升沿有效，状态从 FALSE 切换到 TRUE，执行功能块。
strSourceFileName	源文件名	STRING(32)	-	‘tmp.data’	源文件名。

strDestFileName	目标文件名	STRING(32)	-	‘tmp2.data’	目标文件名。

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
xDone	输出有效	BOOL	TRUE-FALSE	FALSE	TRUE 表示写入数据完成。
xError	错误	BOOL	TRUE-FALSE	FALSE	FALSE——没有错误， TRUE——执行错误。
diErrorID	错误 ID	DINT	遵循数据类型	0	错误码。0 表示没有错误，1 表示找不到 SD 卡，2 表示创建本地文件错误，3 表示打开 SD 卡文件错误，4 表示文件名异常，16 表示文件不存在。
dwCopyedSizeBytes	拷贝的字节长度	DWORD	遵循数据类型	0	拷贝的字节长度

- **功能说明**

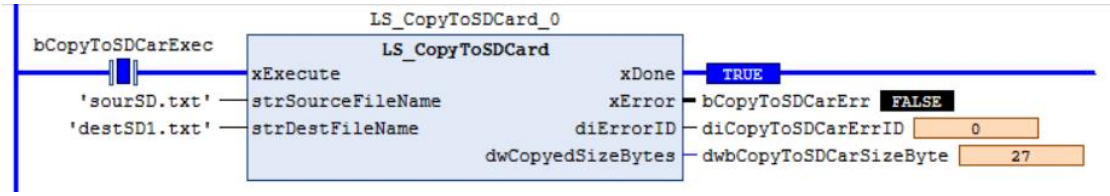
  - 将xEnable设为TRUE时，该功能块进入可运行状态；将xEnable设为FALSE，功能块不运行。
  - 执行功能块前，需将拷贝的文件放置在本地UsrData文件中。

- **程序实例**

ST：

```
LS_CopyToSDCard_0(xExecute TRUE := bCopyToSDCarExec TRUE ,
            strSourceFileName 'sourSD.txt' := 'sourSD.txt',
            strDestFileName 'destSD1.tx' := 'destSD1.txt',
            xDone TRUE => bCopyToSDCarDone TRUE,
            xError FALSE => bCopyToSDCarErr FALSE,
            diErrorID 0 => diCopyToSDCarErrID 0,
            dwCopyedSizeBytes 27 => dwbCopyToSDCarSizeByte 27 );
```

LD：



8.1.4 LS_GetSDCardInfomation

获取 SD 卡下的文件信息。

- **指令格式**

指令	名称	FB/FC	LD表现	ST表现	库
LS_GetSDCardInfomation	获取 SD 卡下的文件信息	FB		LS_GetSDCardInfomation(xExecute:=, strFileExtension:=, iMaxFileNums:=, pstFile_Information_Point:=, xDone=>, xError=>, diErrorID=>, iNumberOfFiles=>);	MC_SysLib

- **相关变量**

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
xExecute	高电平有效	BOOL	TRUE-FALSE	FALSE	上升沿有效，状态从 FALSE 切换到 TRUE，执行功能块。
strFileExtension	源文件名	STRING(32)	-	-	文件扩展名’.data’
iMaxFileNums	允许读取最大文件数	INT	-	20	允许读取最大文件数。
pstFile_Information_Point	文件信息指针	POINTER TO File_Information	-	-	文件信息指针。

输出变量

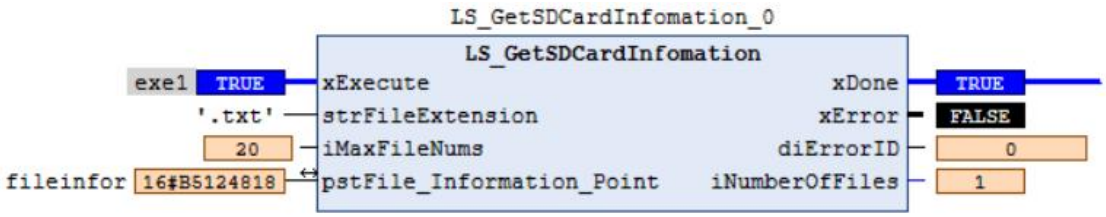
输出变量	名称	数据类型	有效范围	初始值	描述
xDone	输出有效	BOOL	TRUE-FALSE	FALSE	TRUE 表示写入数据完成。
xError	错误	BOOL	TRUE-FALSE	FALSE	FALSE——没有错误， TRUE——输入参数越界。
diErrorID	错误 ID	DINT	遵循数据类型	0	错误码。0 表示没有错误，1 表示找不到 SD 卡，2 表示文件读取个数超限，4 表示“文件信息指针”为空。
iNumberOfFiles	拷贝的字节长度	INT	遵循数据类型	0	读取的文件数。

- **功能说明**
  - 将xEnable设为TRUE时，该功能块进入可运行状态；将xEnable设为FALSE，功能块不运行。
  - pstFile_Information_Point输入输出引脚，需先定义文件信息结构体变量，建议用数组类型，当SD卡中有多个文件时，可以一并将文件信息读取。再用文件指针变量获取文件信息结构体的首地址，如下：
  - File_Infor:ARRAY[0..19] OF File_Information;
  - fileinfor: POINTER TO File_Information :=ADR(File_Infor[0])； 。
- **程序实例**

ST：

```
LS_GetSDCardInfomation_0(xExecute TRUE := exe1 TRUE,
            strFileExtension '.txt' := '.txt',
            iMaxFileNums:= ,
            pstFile_Information_Point 16#B5124818 := fileinfor 16#B5124818,
            xDone TRUE => done TRUE,
            xError FALSE => err FALSE,
            diErrorID=> ,
            iNumberofFiles 1 =>NumberofFiles 1 );RETURN
```

LD：


LS_GetSDCardInfomation_0
LS_GetSDCardInfomation
exe1
TRUE
xExecute
xDone
TRUE
'.txt'
strFileExtension
xError
FALSE
20
iMaxFileNums
diErrorID
0
fileinfor
16#B5124818
pstFile_Information_Point
iNumberOfFiles
1

8.2 系统指令

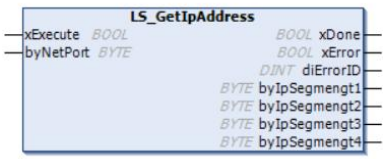
8.2.1 指令列表

指令类别	名称	FB/FC	功能
系统指令	LS_GetIpAddress	FB	获取 PLC IP 地址
	LS_SetIpAddress	FB	设置 PLC IP 地址
	GetPLCLoad	FC	获取 PLC 负载率
	GetPLCVersion	FC	获取 PLC 固件版本
	ColdResetApp	FC	冷复位 PLC
	WarmResetApp	FC	热复位 PLC
	LS_ReadDintDT	FB	获取系统时间（Dint 类型）
	LS_ReadStringDT	FB	获取系统时间（String 类型）
	LS_SetDintDT	FB	设置系统时间（Dint 类型）
	LS_SetStringDT	FB	设置系统时间（String 类型）

8.2.2 LS_GetIpAddress

获取 PLC 网络 IP 地址。通过外部库的方法，根据网络编号获取 PLC 网口的 IP 地址，分 4 段获取，分别为网络类型、网络范围、网络位以及主机位。

- **指令格式**

指令	名称	FB/FC	LD表现	ST表现	库
LS_GetIpAddress	获取网口 IP 地址	FB		LS_GetIpAddress(xExecute:=, byNetPort:=, xDone=>, xError=>, diErrorID=>, byIpSegmengt1=>, byIpSegmengt2=>, byIpSegmengt3=>, byIpSegmengt4=>);	MC_SysLib

- **相关变量**

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
xExecute	高电平有效	BOOL	TRUE-FALSE	FALSE	上升沿有效，状态从 FALSE 切换到 TRUE，读取 IP 地址。
byNetPort	网口 ID	BYTE	0-255	0	网口编号，从 0 号开始。

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
xDone	操作完成	BOOL	TRUE-FALSE	FALSE	功能块执行完成。
xError	错误	BOOL	TRUE-FALSE	FALSE	FALSE——没有错误， TRUE——执行错误。
diErrorID	错误 ID	DINT	遵循数据类型	0	错误码，02 表示 NetPort 值超出范围。
byIpSegment1	IP 地址第 1 段	BYTE	0-255	0	获取到的 IP 地址第 1 段，即网络类型。
byIpSegment2	IP 地址第 2 段	BYTE	0-255	0	获取到的 IP 地址第 2 段，即网络范围。
byIpSegment3	IP 地址第 3 段	BYTE	0-255	0	获取到的 IP 地址第 3 段，即网络位。
byIpSegment4	IP 地址第 4 段	BYTE	0-255	0	获取到的 IP 地址第 4 段，即主机位。

- **功能说明**

  - 将xExecute设为TRUE时，该功能块进入可运行状态；将xExecute设为FALSE，功能块不运行。
  - 根据主机配置的网口通道数，将需要获取的网口对应编号输入到byNetPort，默认从0开始，若主机只有1路以太网口，则输入0。
  - 如需将获取的IP地址转成字符串格式，先使用BYTE_TO_STRING指令把各段地址转

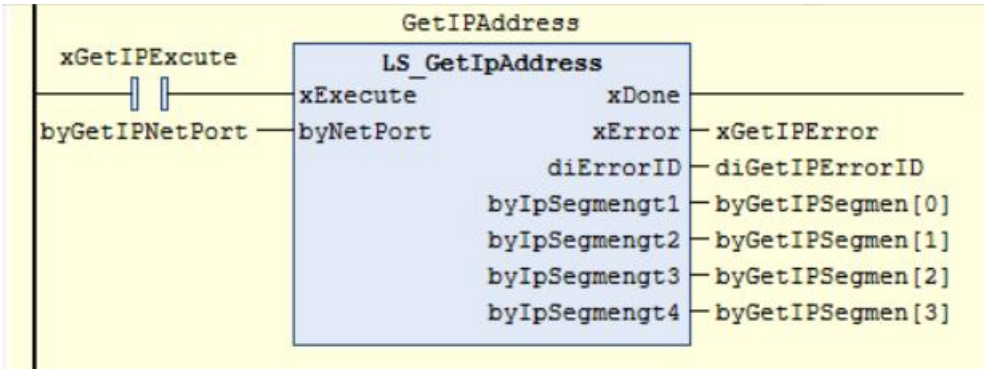
成字符串格式，再使用CONCAT字符串连接函数进行组合。

- **程序实例**

ST：

```
GetIPAddress(
    xExecute TRUE := xGetIPExcute TRUE,
    byNetPort 0 := byGetIPNetPort 0,
    xDone TRUE => xGetIPDone TRUE,
    xError FALSE => xGetIPError FALSE,
    diErrorID 0 => diGetIPErrorID 0,
    byIpSegmengt1 192 => byGetIPSegmen[0] 192,
    byIpSegmengt2 168 => byGetIPSegmen[1] 168,
    byIpSegmengt3 1 => byGetIPSegmen[2] 1,
    byIpSegmengt4 3 => byGetIPSegmen[3] 3 );RETURN
```

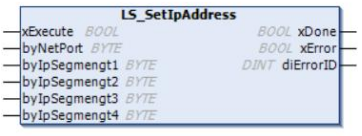
LD：



8.2.3 LS_SetIpAddress

配置 PLC 网络 IP 地址。通过外部库的方法，根据网络编号配置 PLC 网口的 IP 地址，分 4 段配置，分别为网络类型、网络范围、网络位以及主机位。

- **指令格式**

指令	名称	FB/FC	LD表现	ST表现	库
LS_SetIpAddress	配置网口 IP 地址	FB		LS_SetIPAddress(xExecute:=, byNetPort:=, byIpSegmengt1=, byIpSegmengt2=, byIpSegmengt3=, byIpSegmengt4=,	MC_SysLib

				xDone=>, xError=>, diErrorID=>);	

- **相关变量**

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
xExecute	高电平有效	BOOL	TRUE-FALSE	FALSE	上升沿有效，状态从 FALSE 切换到 TRUE，读取 IP 地址。
byNetPort	网口 ID	BYTE	0-255	0	网口编号，从 0 号开始。
ByIpSegment1	IP 地址第 1 段	BYTE	0-255	192	需要配置的 IP 地址第 1 段，即网络类型。
ByIpSegment2	IP 地址第 2 段	BYTE	0-255	168	需要配置的 IP 地址第 2 段，即网络范围。
ByIpSegment3	IP 地址第 3 段	BYTE	0-255	1	需要配置的 IP 地址第 3 段，即网络位。
ByIpSegment4	IP 地址第 4 段	BYTE	0-255	3	需要配置的 IP 地址第 4 段，即主机位。

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
xDone	操作完成	BOOL	TRUE-FALSE	FALSE	功能块执行完成。
xError	错误	BOOL	TRUE-FALSE	FALSE	FALSE——没有错误， TRUE——执行错误。
diErrorID	错误 ID	DINT	遵循数据类型	0	错误码，01 表示 IP 地址第 1 段数据值不能为 0，02 表示 NetPort 值超出范围。

- **功能说明**

  - 将xExecute设为TRUE时，该功能块进入可运行状态；将xExecute设为FALSE，功能块

不运行。

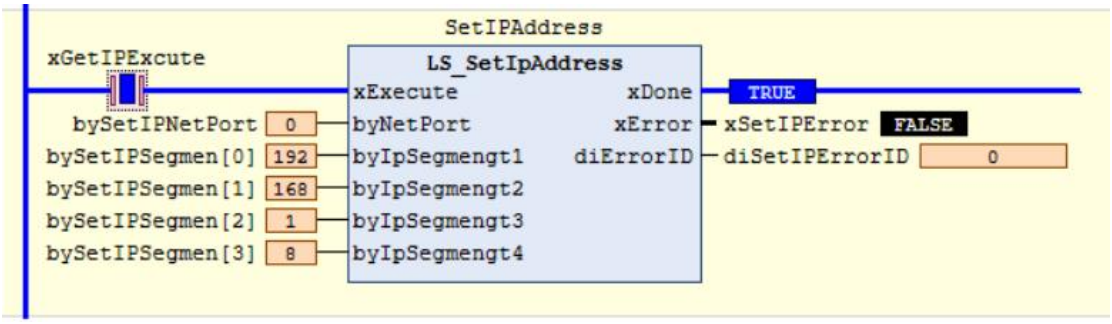
- 根据主机配置的网口通道数，将需要获取的网口对应编号输入到byNetPort，默认从0开始，若主机只有1路以太网口，则输入0。
- 将需要配置的IP地址分段写入到ByIpSegmengt1、ByIpSegmengt2、ByIpSegmengt3、ByIpSegmengt4中。
- 注意：成功配置新IP地址后，必须断电重启PLC才能生效。

- **程序实例**

ST：

```
SetIPAddress(
    xExecute TRUE := xSetIPExcute TRUE,  //写入IP
    byNetPort 0 := bySetIPNetPort 0,
    byIpSegmengt1 192 := bySetIPSegmen[0] 192,
    byIpSegmengt2 168 := bySetIPSegmen[1] 168,
    byIpSegmengt3 1 := bySetIPSegmen[2] 1,
    byIpSegmengt4 12 := bySetIPSegmen[3] 12,
    xDone TRUE => xSetIPDone TRUE,
    xError FALSE => xSetIPError FALSE,
    diErrorID 0 => diSetIPErrorID 0);
RETURN
```

LD：



8.2.4 GetPLCLoad

获取 PLC 的 CPU 负载率。将 CPU 负载率信息存储到内部属性变量中。

- **指令格式**

指令	名称	FB/FC	LD表现	ST表现	库

GetPLCLoad	获取 CPU 负载率	FC		GetPLCLoad(xEnable:=, xValid=>, xError=>, diErrorID=>, fLoad=>);	MC_SysLib

- **相关变量**

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
xEnable	高电平有效	BOOL	TRUE-FALSE	FALSE	高电平有效。

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
xValid	输出有效	BOOL	TRUE-FALSE	FALSE	TRUE 表示读取使能。
xError	错误	BOOL	TRUE-FALSE	FALSE	FALSE——没有错误， TRUE——执行错误。
diErrorID	错误 ID	DINT	遵循数据类型	0	错误码。
fLoad	CPU 负载率	REAL	遵循数据类型		获取的 CPU 负载率。

- **功能说明**

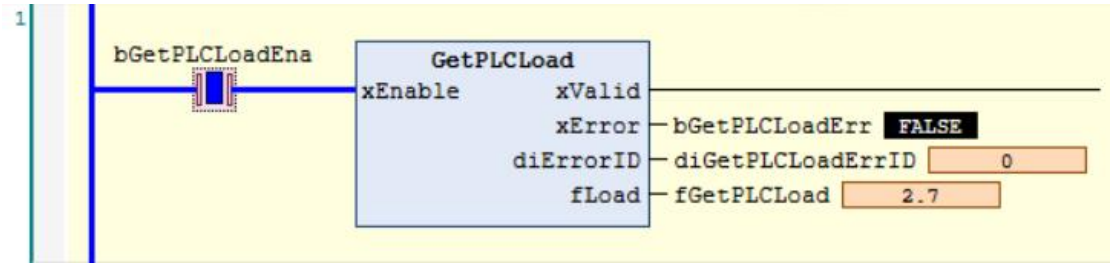
  - 将xEnable设为TRUE时，该功能块进入可运行状态；将xEnable设为FALSE，功能块不运行。
  - 若将xEnable保持为TRUE，fLoad则跟随系统CPU负载实时变化。

- **程序实例**

ST：

```
GetPLCLoad(xEnable:= bGetPLCLoadEna TRUE,
           xValid=>bGetPLCLoadValid TRUE,
           xError=>bGetPLCLoadErr FALSE,
           diErrorID=> diGetPLCLoadErrID 0,
           fLoad=> fGetPLCLoad 2.93);
RETURN
```

LD：



8.2.5 GetPLCVersion

获取 PLC 固件版本。将 PLC 固件版本信息存储到内部属性变量中。

- **指令格式**

指令	名称	FB/FC	LD表现	ST表现	库
GetPLCVersion	获取 CPU 固件版本	FC		GetPLCLoad(xEnable:=, xValid=>, xError=>, diErrorID=>, udInfo_CPU_ID=>, diHardware_Ver=>, diFirmwareH_Ver=>, diFirmwareS_Ver=>);	MC_SysLib

- **相关变量**

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
xEnable	高电平有效	BOOL	TRUE-FALSE	FALSE	必须设置为 TRUE，以激活功能块的处理。

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述

xValid	输出有效	BOOL	TRUE-FALSE	FALSE	TRUE 表示读取使能。
xError	错误	BOOL	TRUE-FALSE	FALSE	FALSE——没有错误， TRUE——执行错误。
diErrorID	错误 ID	DINT	遵循数据类型	0	错误码。
udInfo_CPU_ID	CPU ID	UDINT	遵循数据类型	-	CPU 的 ID 号。
diHardware_Ver	控制器型号	DINT	遵循数据类型	-	控制器型号。
diFirmwareH_Ver	FPGA 版本	DINT	遵循数据类型	-	需要用 16 进制显示。
diFirmwareS_Ver	ARM 版本	DINT	遵循数据类型	-	需要用 16 进制显示。

- **功能说明**

  - 将xEnable设为TRUE时，该功能块进入可运行状态；将xEnable设为FALSE，功能块不运行。

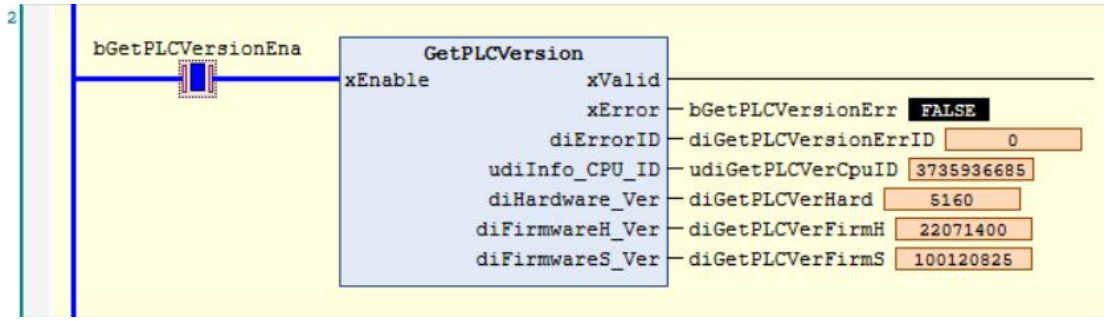
  - 若将xEnable保持为TRUE，fLoad则跟随系统CPU负载实时变化。

- **程序实例**

ST：

```
GetPLCVersion(xEnable:= bGetPLCLoadEna TRUE,
              xValid=> bGetPLCLoadValid TRUE,
              xError=> bGetPLCLoadErr FALSE,
              diErrorID=> diGetPLCLoadErrID 0,
              udiInfo_CPU_ID=> udiGetPLCVerCpuID 3735936685,
              diHardware_Ver=> diGetPLCVerHard 5160,
              diFirmwareH_Ver=> diGetPLCVerFirmH 22071400,
              diFirmwareS_Ver=> diGetPLCVerFirmS 100120825 );RETURN
```

LD：



8.2.6 ColdResetApp

冷复位 PLC。执行功能块后，PLC 从 RUN 状态切换到 STOP 状态，再恢复到 RUN 状态。复位后，程序中 VAR 类型变量、VAR RETAIN 类型变量将初始化值，VAR PERSISTENT RETAIN 或 VAR RETAIN PERSISTENT 类型变量会保持原值。

- **指令格式**

指令	名称	FB/FC	LD表现	ST表现	库
ColdReset App	冷复位 PLC	FC		ColdResetApp(xEnable:=);	MC_SysLib

- **相关变量**

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
xExecute	高电平有效	BOOL	TRUE-FALSE	FALSE	必须设置为 TRUE，以激活功能块的处理。

- **功能说明**

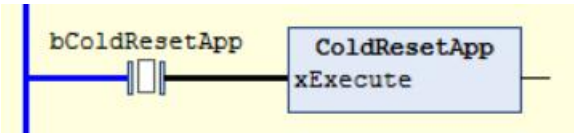
  - 将xEnable设为TRUE时，该功能块进入可运行状态；将xEnable设为FALSE，功能块不运行。
  - 执行功能块后，xEnable自动恢复为FALSE状态。

- **程序实例**

ST：

LD：

```
ColdResetApp(xExecute:= bColdResetApp FALSE );
```



8.2.7 WarmResetApp

热复位 PLC。执行功能块后，PLC 从 RUN 状态切换到 STOP 状态，再恢复到 RUN 状态。复位后，程序中 VAR 类型变量将初始化值，VAR RETAIN 类型变量和 VAR PERSISTENT RETAIN 或 VAR RETAIN PERSISTENT 类型变量会保持原值。

- **指令格式**

指令	名称	FB/FC	LD表现	ST表现	库
WarmResetApp	热复位 PLC	FC		WarmResetApp(xEnable:=);	MC_SysLib

- **相关变量**

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
xExecute	高电平有效	BOOL	TRUE-FALSE	FALSE	必须设置为 TRUE，以激活功能块的处理。

- **功能说明**

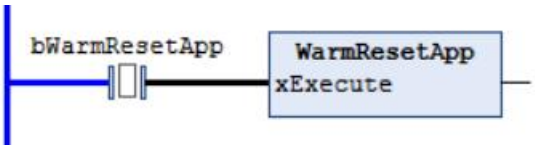
  - 将xEnable设为TRUE时，该功能块进入可运行状态；将xEnable设为FALSE，功能块不运行。
  - 执行功能块后，xEnable自动恢复为FALSE状态。

- **程序实例**

ST：

```
WarmResetApp(xExecute:= bWarmResetApp FALSE );RETURN
```

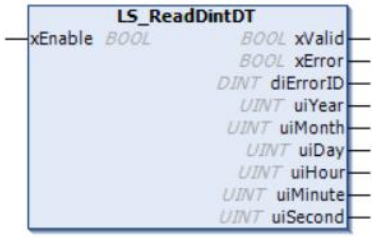
LD：



8.2.8 LS_ReadDintDT

获取 PLC 系统时间(Dint 类型)，将 PLC 系统时间信息存储到内部属性变量中。

- **指令格式**

指令	名称	FB/FC	LD表现	ST表现	库
LS_ReadDintDT	获取 PLC 系统时间	FB		LS_ReadDintDT(xEnable:=, xValid=>, xError=>, diErrorID=>, uiYear=>, uiMonth=>, uiDay=>, uiHour=>, uiMinute=>, uiSecond=>);	MC_SysLib

- **相关变量**

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
xEnable	高电平有效	BOOL	TRUE-FALSE	FALSE	必须设置为 TRUE，以激活功能块的处理。

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述

xValid	输出有效	BOOL	TRUE-FALSE	FALSE	TRUE 表示读取使能。
xError	错误	BOOL	TRUE-FALSE	FALSE	FALSE——没有错误， TRUE——执行错误。
diErrorID	错误 ID	DINT	遵循数据类型	0	错误码。
uiYear	年	UINT	遵循数据类型	1970～2106	年。
uiMonth	月	UINT	遵循数据类型	1～12	月。
uiDay	日	UINT	遵循数据类型	1～31	日。
uiHour	时	UINT	遵循数据类型	0～23	时。
uiMinute	分	UINT	遵循数据类型	0～59	分。
uiSecond	秒	UINT	遵循数据类型	0～59	秒。

- **功能说明**

  - 将xEnable设为TRUE时，该功能块进入可运行状态；将xEnable设为FALSE，功能块不运行。

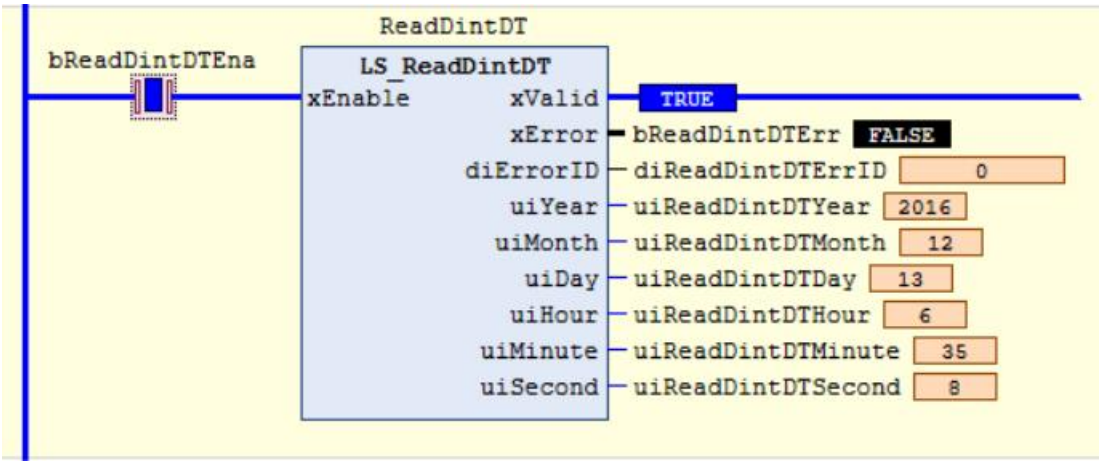
  - 若将xEnable保持为TRUE，则实时读取PLC系统时间到内部属性变量中。

- **程序实例**

  ST：

```
ReadDintDT(xEnable TRUE := bReadDintDTEna TRUE,
        xValid TRUE => bReadDintDTValid TRUE,
        xError FALSE => bReadDintDTErr FALSE,
        diErrorID 0 => diReadDintDTErrID 0,
        uiYear 2016 => uiReadDintDTYear 2016,
        uiMonth 12 => uiReadDintDTMonth 12,
        uiDay 13 => uiReadDintDTDay 13,
        uiHour 6 => uiReadDintDTHour 6,
        uiMinute 28 => uiReadDintDTMinute 28,
        uiSecond 22 => uiReadDintDTSecond 22);
```

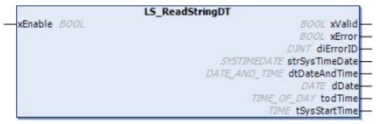
LD：



8.2.9 LS_ReadStringDT

获取 PLC 系统时间(String 类型)，将 PLC 系统时间信息存储到内部属性变量中。

- **指令格式**

指令	名称	FB/FC	LD表现	ST表现	库
LS_ReadStringDT	获取 PLC 系统时间	FB		LS_ReadStringDT(xEnable:=, xValid=>, xError=>, diErrorID=>, strSysTimeDate=>, dtDateAndTime=>, dDate=>, todTime=>, tSysStartTime=>);	MC_SysLib

- **相关变量**

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
xEnable	高电平有效	BOOL	TRUE-FALSE	FALSE	必须设置为 TRUE，以激活功能块的处理。

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
xValid	输出有效	BOOL	TRUE-FALSE	FALSE	TRUE 表示读取使能。
xError	错误	BOOL	TRUE-FALSE	FALSE	FALSE——没有错误， TRUE——执行错误。
diErrorID	错误 ID	DINT	遵循数据类型	0	错误码。
strSysTimeDate	系统时间结构体	SYSTIMEDATE	遵循数据类型		结构体变量包含：wYear、wMonth、wDay、wHour、wMinute、wSecond、wMilliseconds、wDayofWeek、wYday。
dtDateAndTime	系统时间	DATE_AND_TIME	遵循数据类型		格式：DT#2016-12-13-10：20：43。
dDate	系统日期	DATE	遵循数据类型		格式：D#2016-12-13。
todTime	系统时分秒	TIME_OF_DAT	遵循数据类型		格式：TOD#10：20：43。
tSysStartTime	系统运行时常	TIME	遵循数据类型		格式：T#4h22m22s333ms。

- **功能说明**

  - 将xEnable设为TRUE时，该功能块进入可运行状态；将xEnable设为FALSE，功能块不运行。
  - 若将xEnable保持为TRUE，则实时读取PLC系统时间到内部属性变量中。

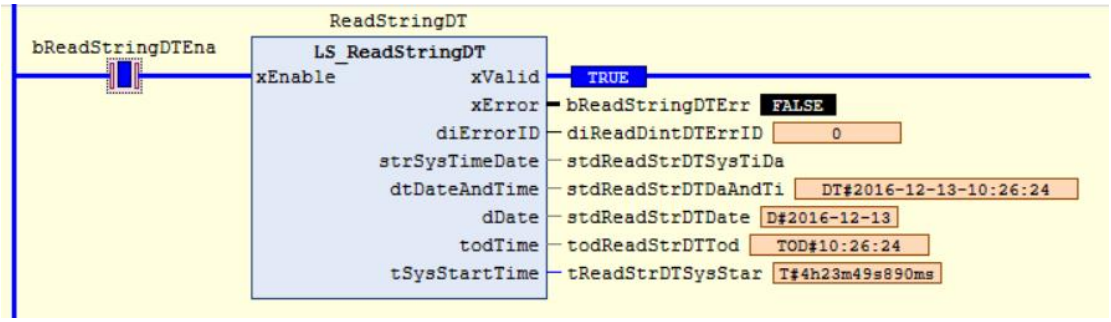
➢ SYSTIMEDATE结构体库文件路径：MC_SysLib.SysTimeRtc。

- **程序实例**

ST：

```
ReadStringDT(xEnable:= bReadStringDTEna,
        xValid=> bReadStrDTValid,
        xError=> bReadDintDTErr,
        diErrorID=> diReadStrDTErrID,
        strSysTimeDate=> stdReadStrDTSysTiDa,
        dtDateAndTime=> stdReadStrDTDaAndTi,
        dDate=> stdReadStrDTDate,
        todTime=> todReadStrDTTod,
        tSysStartTime=> tReadStrDTSysStar);
        RETURN
```

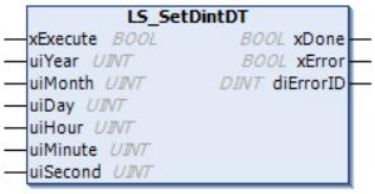
LD：



8.2.10 LS_SetDintDT

设置 PLC 系统时间(Dint 类型)，将内部属性变量中的信息写入到 PLC 系统时间。

- **指令格式**

指令	名称	FB/FC	LD表现	ST表现	库
LS_SetDintDT	设置 PLC 系统时间	FB		LS_SetDintDT(xExecute:=, uiYear:=, uiMonth:=, uiDay:=, uiHour:=, uiMinute:=, uiSecond:= xDone=>, xError=>, diErrorID=>);	MC_SysLib

- **相关变量**

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
xExecute	高电平有效	BOOL	TRUE-FALSE	FALSE	上升沿有效，状态从 FALSE 切换到 TRUE，执行功能块。
uiYear	年	UINT	1970～2106	2020	年。
uiMonth	月	UINT	1～12	4	月。
uiDay	日	UINT	1～31	17	日。
uiHour	时	UINT	0～23	10	时。
uiMinute	分	UINT	0～59	10	分。
uiSecond	秒	UINT	0～59	10	秒。

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
xDone	执行完成	BOOL	TRUE-FALSE	FALSE	上升沿有效，状态从 FALSE 切换到 TRUE，执行功能块。
xError	错误	BOOL	TRUE-FALSE	FALSE	FALSE——正常执行， TRUE——执行错误。
diErrorID	错误 ID	DINT	遵循数据类型	0	错误码。

- **功能说明**

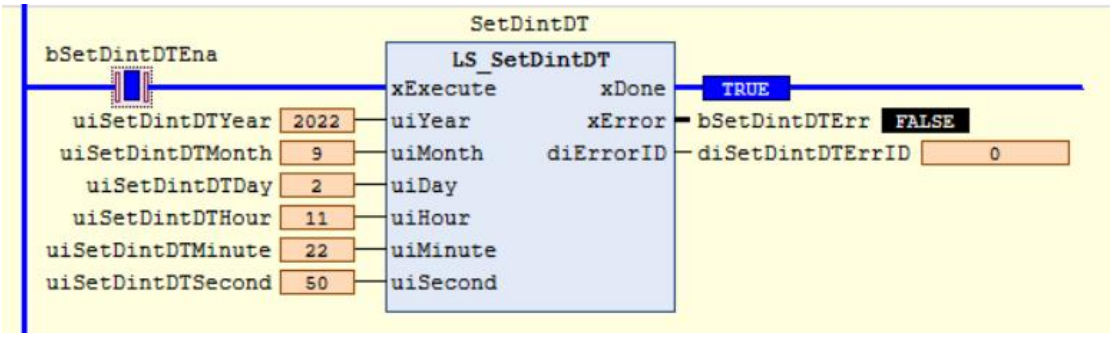
➢ 将xExecute设为TRUE时，该功能块进入可运行状态；将xExecute设为FALSE，功能块不运行。

- **程序实例**

ST：

```
SetDintDT(xExecute TRUE := bSetDintDTEna TRUE,
          uiYear 2022 := uiSetDintDTYear 2022,
          uiMonth 9 := uiSetDintDTMonth 9,
          uiDay 2 := uiSetDintDTDay 2,
          uiHour 11 := uiSetDintDTHour 11,
          uiMinute 22 := uiSetDintDTMinute 22,
          uiSecond 50 := uiSetDintDTSecond 50,
          xDone TRUE => bSetDintDTDone TRUE,
          xError FALSE => bSetDintDTErr FALSE,
          diErrorID 0 => diSetDintDTErrID 0);
```

LD：



8.2.11 LS_SetStringDT

设置 PLC 系统时间(String 类型)，将 PLC 系统时间信息存储到内部属性变量中。

- **指令格式**

指令	名称	FB/FC	LD表现	ST表现	库
LS_SetStringDT	获取 PLC 系统时间	FB		LS_SetStringDT(xExecute:=, diDateAndTime=, xDone=>, xError:=>, diErrorID:=>);	MC_SysLib

- **相关变量**

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
xExecute	高电平有效	BOOL	TRUE-FALSE	FALSE	必须设置为 TRUE，以激活功能块的处理。
diDateAndTime	需要设置的日期和时间	DATE_AND_TIME	遵循数据类型	2020-4-17-10:10:10	需要设置的日期和时间。

输出变量

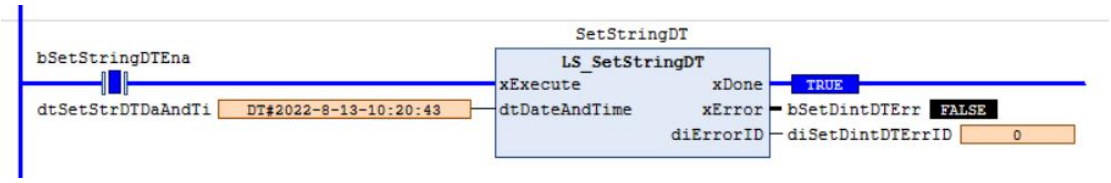
输出变量	名称	数据类型	有效范围	初始值	描述
xValid	输出有效	BOOL	TRUE-FALSE	FALSE	TRUE 表示读取使能。
xError	错误	BOOL	TRUE-FALSE	FALSE	FALSE——没有错误， TRUE——执行错误。
diErrorID	错误 ID	DINT	遵循数据类型	0	错误码。

- **功能说明**
  - 将xEnable设为TRUE时，该功能块进入可运行状态；将xEnable设为FALSE，功能块不运行。
  - diDateAndTime输入格式如DT#2022-8-13-10:20:43。
- **程序实例**

ST：

```
SetStringDT(xExecute TRUE := bSetStringDTEna TRUE,
        dtDateAndTime DT#2022-8-13-10:20:43 := dtSetStrDTDaAndTi DT#2022-8-13-10:20:43,
        xDone TRUE => bSetStrDTDone TRUE,
        xError FALSE => bSetStringDTErr FALSE,
        diErrorID 0 => diSetStringDTErrID 0);
    RETURN
```

LD：



客户咨询中心
目录索取・技术咨询・产品解惑
400-885-5521 销售热线
400-885-5501 技术热线

更多最新的雷赛资讯，请扫码关注！

公众号　　视频号

成就客户　共创共赢

深圳市雷赛智能控制股份有限公司
China Leadshine Technology Co., Ltd.

深圳市南山区沙河西路 3157 号南山智谷产业园 B 栋 15-20 层
邮 编：518052
电 话：400-885-5521　　传 真：0755-26402718
网 址：www.leisai.com　　E-Mail：marketing@leisai.com

上海分公司
上海市嘉定区金园五路 601 号

济南代表处
济南市天桥区滨河商务中心 D 座 2003 室

广佛代表处
广州市番禺区汉溪大道西 218 号李锦记大厦 A 塔 8032

川渝代表处
成都市武区人民南路四段 27 号商鼎国际 1 栋 1 单元 23 楼 A2309 房

温州代表处
浙江省温州市瓯海区潘桥街道宁波路阳光城愉景嘉园 8 幢 2604

杭州代表处
杭州市钱塘区白杨街道 6 号大街 260 号正泰中自科技园 19 幢 1006 室

大连代表处
辽宁省大连市沙河口区滨河街 60-1 号新星星海中心 A 座 1106 室

北京分公司
北京市大兴区绿地启航国际 3 号楼 1109 室

苏州代表处
江苏省苏州市苏州工业园区金尚路 1 号仙峰大厦南楼 7 层

东莞代表处
广东省东莞市南城区黄金路 1 号东莞天安数码城 F 区 3 栋 604

武汉代表处
湖北省武汉市东湖新技术开发区长城园路 2 号海贝孵化器 209

南京代表处
南京市江宁区科建路天韵南京科创产业园 1155 号 F 栋 403 室

长沙代表处
长沙市开福区湘江北路三段 1500 号北辰时代广场 A3 区 3426 房

中珠江代表处
中山市东区长江路 33 号汉宏盈基商务中心 9 层 906