INOVANCE

FORWARD, ALWAYS PROGRESSING







Medium-sized PLC Instructions

User Guide

Industrial Automation | Intelligent Elevator | New Energy Vehicle | Industrial Robot | Rail Transit

Data code 19012377A00

Preface

Introduction

This guide covers medium-sized programmable logic controllers (PLCs) of Inovance, including the AM and AC series.

This guide describes instructions and programs used by the medium-sized PLC programming software, InoProShop, to program controllers. This guide provides examples of instruction usage to help you get started quickly.

More Documents

Data Code	Doc Name	Description
19012378	Medium-Sized PLC Programming Guide (Motion Control)	Describes the composition of the PLC motion control system, mechanism of the motion control program, MC instructions, and simulation and debugging related operations.
19010980	Medium-Sized PLC Programming Software User Guide	Describes the basic functions, quick start, network settings, and programming basics of the medium-sized PLC programming software.

Revision History

Date	Version	Description
Sept 2024	A00	First release

Access to the Guide

This guide is not delivered with the product. The version may be updated without further notice. You can obtain the latest version in the following ways:

- Log in to Inovance's website (www.inovance.com), choose "Support" > "Download", search by keyword, and then download the PDF file.
- Scan the QR code on the product with your smart phone.
- Scan the QR code below to install the Inovance APP, and search for the file in the APP.

Warranty

For faults and damage incurred during normal use in the warranty period, Inovance provides free repair service. (For details of the warranty period, see the purchase order.)

A maintenance fee will be charged out of the warranty period.

Even in the warranty period, a maintenance fee will be charged for repair of the following damage:

- Damage caused by operations not following the instructions in the guide
- Damage caused by fire, flood, or abnormal voltage
- Damage caused by unintended use of the product
- Damage caused by use beyond the specified scope of application of the product
- Damage or secondary damage caused by force majeure (natural disaster, earthquake, and lightning strike)

The maintenance fee will be charged according to the latest Price List of Inovance if not otherwise agreed upon.

For details, see the Product Warranty Card.

Contents

Preface 1

Introduction 1

More Documents 1

Revision History 1

Access to the Guide 2

Warranty 3

Fundamental Safety Instructions 18

Chapter 1 Overview of Basic Instructions 22

1. 1 Overview 22

1. 2 Standard Data Type 22

Chapter 2 Instruction List 23

Chapter 3 Basic Instructions 31

3. 1 Comparison Instructions 31

3.1. 1 Instruction List 31

3.1. 2 GT 31

3.1. 3 LT 33

3.1. 4 GE 34

3.1. 5 LE 35

3.1. 6 EQ 36

3.1. 7 NE 37

3.1. 8 ZoneCmp 39

3.1. 9 TableCmp 41

3.1. 10 AryCmpEQ 43

3.1. 11 AryCmpNE 45

3.1. 12 AryCmpEQV 46

3.1. 13 AryCmpNEV 48

3.1. 14 AryCmpLT 50

3.1. 15 AryCmpLE 52

3.1. 16 AryCmpGT 54

3.1. 17 AryCmpGE 56

3.1. 18 AryCmpLTV 58

3.1. 19 AryCmpLEV 60

3.1. 20 AryCmpGTV 62
3.1. 21 AryCmpGEV 64
3. 2 Selection Instructions 66
3.2. 1 Instruction List 66
3.2. 2 AryMax/AryMin 67
3.2. 3 ArySearch 69
3.2. 4 ArySearch2 72
3.2. 5 SEL 74
3.2. 6 MUX 76
3.2. 7 MAX 78
3.2. 8 MIN 79
3.2. 9 LIMIT 80
3. 3 Counter Instructions 81
3.3. 1 Instruction List 81
3.3. 2 CTD_** 82
3.3. 3 CTU_** 84
3.3. 4 CTUD_** 87
3.3. 5 CTD 90
3.3. 6 CTU 92
3.3. 7 CTUD 94
3. 4 Timer Instructions 97
3.4. 1 Instruction List 97
3.4. 2 AccumulationTimer 97
3.4. 3 Timer 99
3.4. 4 TP 101
3.4. 5 TON 103
3.4. 6 TOF 105
3.4. 7 RTC 106
3. 5 Bit and Word Logic Instructions 108
3.5. 1 Instruction List 108
3.5. 2 AND 109
3.5. 3 OR 111
3.5. 4 NOT 112
3.5. 5 XOR 113
3.5. 6 PLS 114

3.5. 7 PLF 116
3.5. 8 ALT 117
3.5. 9 BOUT 118
3.5. 10 BSET 121
3.5. 11 BRST 122
3.5. 12 SR 123
3.5. 13 RS 125
3.5. 14 R_TRIG 126
3.5. 15 F_TRIG 127
3.5. 16 Up 129
3.5. 17 Down 130
3.5. 18 EXTRACT 131
3.5. 19 PUTBIT 132
3.5. 20 PACK 134
3.5. 21 UNPACK 135
3.5. 22 BIT_AS_BYTE 137
3.5. 23 BYTE_AS_BIT 139
3.5. 24 BIT_AS_WORD 141
3.5. 25 WORD_AS_BIT 143
3.5. 26 BIT_AS_DWORD 146
3.5. 27 DWORD_AS_BIT 148
3.5. 28 AryAnd, AryOr, AryXor, and AryXorN 150
3. 6 Data Shift Instructions 154
3.6. 1 Instruction List 154
3.6. 2 SHL and SHR 154
3.6. 3 ArySHL and ArySHR 156
3.6. 4 AryShiftReg 159
3.6. 5 AryShiftRegLR 161
3.6. 6 ROL and ROR 163
3.6. 7 RCL and RCR 165
3.6. 8 SFTL and SFTR 167
3.6. 9 WSFL and WSFR 170
3.6. 10 SFRD and SFWR 173
3.6. 11 NSHLC and NSHRC 177
3. 7 Data Type Conversion Instructions 179

3.7. 1 Instruction List 179
3.7. 2 BOOL_TO_<TYPE> 180
3.7. 3 BYTE_TO_<TYPE> 181
3.7. 4 WORD_TO_<TYPE> 182
3.7. 5 DWORD_TO_<TYPE> 183
3.7. 6 INT_TO_<TYPE> 184
3.7. 7 SINT_TO_<TYPE> 185
3.7. 8 DINT_TO_<TYPE> 186
3.7. 9 UDINT_TO_<TYPE> 187
3.7. 10 REAL_TO_<TYPE> 188
3.7. 11 STRING_TO_<TYPE> 189
3.7. 12 TIME_TO_<TYPE> 190
3.7. 13 TOD_TO_<TYPE> 191
3.7. 14 DATE_TO_<TYPE> 192
3.7. 15 DT_TO_<TYPE> 193
3.7. 16 DtToString 194
3.7. 17 DateToString 196
3.7. 18 TodToString 197
3.7. 19 EnumToNum 198
3.7. 20 NumToEnum 200
3. 8 Data Processing Instructions 201
3.8. 1 Instruction List 201
3.8. 2 WriteNbit_** 203
3.8. 3 ReadNbit_** 205
3.8. 4 TransBits 207
3.8. 5 Encoder 209
3.8. 6 Decoder 210
3.8. 7 NumToDecString and NumToHexString 212
3.8. 8 HexStringToNum_** 215
3.8. 9 AryToString 216
3.8. 10 StringToAry 218
3.8. 11 Clear 220
3.8. 12 SetBlock 221
3.8. 13 MOVE 223
3.8. 14 BMOV 224

3.8. 15 FMOV 226
3.8. 16 BON 228
3.8. 17 SUM 229
3.8. 18 BTOW 230
3.8. 19 SWAP 232
3.8. 20 SWAP16 234
3.8. 21 SWAP32 235
3.8. 22 XCH 236
3.8. 23 HEXinASCII_TO_BYTE 238
3.8. 24 BYTE_TO_HEXinASCII 239
3.8. 25 WORD_AS_STRING 240
3.8. 26 Byte_To_HexString 243
3.8. 27 Word_To_HexString 244
3.8. 28 Dword_To_HexString 245
3.8. 29 CopyRealToNum 247
3.8. 30 CopyByteToNum 248
3.8. 31 CopyDwordToReal 250
3.8. 32 CopyNumToByte 251
3.8. 33 CopyNumToReal 253
3.8. 34 CopyRealToDword 254
3.8. 35 BitCnt 256
3.8. 36 AryByteTo 257
3.8. 37 DispartReal 263
3.8. 38 UniteReal 264
3.8. 39 GRAY 266
3.8. 40 ColmToLine_** 269
3.8. 41 LineToColm 271
3.8. 42 FixNumToString 273
3.8. 43 StringToFixNum 275
3.8. 44 DispartDigit 277
3.8. 45 UniteDigit_** 278
3.8. 46 Dispart8Bit 280
3.8. 47 Unite8Bit_** 281
3.8. 48 ToAryByte 283
3.8. 49 AryToBCD 287

3.8. 50 AryToBin 289
3.8. 51 MoveDigit 290
3.8. 52 Exchange 292
3.8. 53 AryExchange 293
3.8. 54 AryMove 295
3.8. 55 SizeOfAry 297
3.8. 56 GrayToBin_** 298
3.8. 57 BinToGray_** 300
3. 9 Math Instructions 301
3.9. 1 Instruction List 301
3.9. 2 ADD 303
3.9. 3 SUB 305
3.9. 4 MUL 306
3.9. 5 DIV 307
3.9. 6 MOD 308
3.9. 7 ABS 309
3.9. 8 SQRT 310
3.9. 9 LN 311
3.9. 10 LOG 312
3.9. 11 EXP 314
3.9. 12 EXPT 315
3.9. 13 SIN 316
3.9. 14 COS 317
3.9. 15 TAN 318
3.9. 16 ASIN 319
3.9. 17 ACOS 320
3.9. 18 ATAN 321
3.9. 19 RAD and DEG 322
3.9. 20 SIZEOF 324
3.9. 21 ArySD 325
3.9. 22 RoundUp 327
3.9. 23 MovingAverage 328
3.9. 24 Inc and Dec 333
3.9. 25 AryAddV 334
3.9. 26 ArySubV 336

3.9. 27 CheckReal 338
3.9. 28 AryMean 340
3.9. 29 ModReal 342
3.9. 30 ModReal_LR 344
3.9. 31 Rand 346
Chapter 4 Extended Instructions 329
4. 1 Time and Time of Day Instructions 329
4.1. 1 Instruction List 329
4.1. 2 MULTIME 330
4.1. 3 DIVTIME 331
4.1. 4 TruncTime 333
4.1. 5 TruncDt 334
4.1. 6 TruncTod 335
4.1. 7 ChkLeapYear 337
4.1. 8 GetDaysOfMonth 338
4.1. 9 GetDayOfWeek 339
4.1. 10 GetWeekOfYear 340
4.1. 11 ADD_TOD_TIME 341
4.1. 12 ADD_DT_TIME 343
4.1. 13 SUB_TOD_TIME 344
4.1. 14 SUB_DT_TIME 345
4.1. 15 SUB_TOD_TOD 346
4.1. 16 SUB_DATE_DATE 347
4.1. 17 SUB_DT_DT 349
4.1. 18 DT_TO_DATE 350
4.1. 19 DT_TO_TOD 351
4.1. 20 DtToDateStruct 352
4.1. 21 DateStructToDt 354
4.1. 22 GetSystemDate_sDt 355
4.1. 23 GetSystemTime 357
4.1. 24 SysHC_SetSystemDate 358
4.1. 25 SysHC_GetSystemDate 360
4.1. 26 DtToSec 362
4.1. 27 DateToSec 363
4.1. 28 TodToSec 364

4.1. 29 SecToDt 366
4.1. 30 SecToDate 367
4.1. 31 SecToTod 368
4.1. 32 TimeToNanoSec 369
4.1. 33 TimeToSec 370
4.1. 34 NanoSecToTime 371
4.1. 35 SecToTime 373
4.1. 36 DaysToMonth 374
4. 2 Text String Instructions 375
4.2. 1 Instruction List 375
4.2. 2 AddDelimiter 376
4.2. 3 AddDelimiter_LR 378
4.2. 4 SubDelimiter 380
4.2. 5 SubDelimiter_LR 383
4.2. 6 ToUCase and ToLCase 386
4.2. 7 StringSum 388
4.2. 8 LEN 389
4.2. 9 LEFT 391
4.2. 10 RIGHT 392
4.2. 11 MID 394
4.2. 12 CONCAT 396
4.2. 13 INSERT 397
4.2. 14 DELETE 399
4.2. 15 FIND 400
4.2. 16 REPLACE 402
4.2. 17 GetByteLen 403
4.2. 18 ClearString 405
4.2. 19 TrimL and TrimR 406
4. 3 Address Operation Instructions 408
4.3. 1 Instruction List 408
4.3. 2 ADR and ^ 408
4.3. 3 BITADR 409
4. 4 Queue Instructions 411
4.4. 1 Instruction List 411
4.4. 2 FIFO 411

4.4. 3 StackFIFO 415
4.4. 4 StackPush 417
4.4. 5 StackLIFO 420
4.4. 6 StackIns 422
4.4. 7 StackDel 425
4. 5 Table and Range Instructions 427
4.5. 1 Instruction List 427
4.5. 2 BZAND_TAB 428
4.5. 3 MEAN_TAB 431
4.5. 4 ZONE_TAB 432
4.5. 5 ZRST_TAB 435
4.5. 6 SCL_TAB 437
4.5. 7 SORT_TAB 440
4.5. 8 RAMP_TAB 445
4.5. 9 WSUM_TAB 448
4.5. 10 RecSearch 451
4.5. 11 RecRangeSearch 454
4.5. 12 RecSort 459
4.5. 13 RecNum 462
4.5. 14 RecMax and RecMin 464
4. 6 File Operation Instructions 467
4.6. 1 Instruction List 467
4.6. 2 FileOpen 467
4.6. 3 FileClose 470
4.6. 4 FileSeek 472
4.6. 5 FileRead 475
4.6. 6 FileWrite 478
4.6. 7 FilePuts 481
4.6. 8 FileGets 484
4.6. 9 FileWriteVar 486
4.6. 10 FileReadVar 489
4.6. 11 FileCopy 492
4.6. 12 FileRemove 494
4.6. 13 FileRename 496
4.6. 14 DirCreate 499

4.6. 15 DirRemove 502
4.6. 16 FileOpen2 504
4.6. 17 FileClose2 508
4.6. 18 FileSeek2 511
4.6. 19 FileRead2 514
4.6. 20 FileWrite2 518
4.6. 21 FilePuts2 521
4.6. 22 FileGets2 524
4.6. 23 FileWriteVar2 528
4.6. 24 FileReadVar2 532
4.6. 25 FileCopy2 535
4.6. 26 FileRemove2 539
4.6. 27 FileRename2 542
4.6. 28 DirCreate2 545
4.6. 29 DirRemove2 548
4.6. 30 FileTell 552
4.6. 31 FileFindList 555
4. 7 Analog Calculation Instructions 560
4.7. 1 Instruction List 560
4.7. 2 PD 560
4.7. 3 PID 563
4.7. 4 PID_FIXCYCLE 566
4.7. 5 PID_AT 570
4.7. 6 PID_ATC 594
4. 8 BCD Conversion Instructions 619
4.8. 1 Instruction List 619
4.8. 2 BCD_TO_INT 619
4.8. 3 INT_TO_BCD 621
4.8. 4 BCD_TO_BYTE 622
4.8. 5 BYTE_TO_BCD 623
4.8. 6 BCD_TO_WORD 625
4.8. 7 WORD_TO_BCD 626
4.8. 8 BCD_TO_DWORD 627
4.8. 9 DWORD_TO_BCD 629
4. 9 Filter Instructions 630

4.9. 1 Instruction List 630
4.9. 2 LimitingFilter 630
4.9. 3 MedianFilter 634
4.9. 4 ArithmeticAverageFilter 637
4.9. 5 RecursiveAverageFilter 641
4.9. 6 MedianAverageFilter 645
4.9. 7 LimitingAverageFilter 648
4.9. 8 FirstOrderLagFilter 652
4.9. 9 WeightRecursiveAverageFilter 656
4.9. 10 DebounceFilter 659
4.9. 11 LimitingDebounceFilter 663
4. 10 System Instructions 667
4.10. 1 Instruction List 667
4.10. 2 SysHC_HWInfo 668
4.10. 3 SysHC_SWInfo 670
4.10. 4 SysHC_CPUInfo 672
4.10. 5 SysHC_CPUDiagnose 675
4.10. 6 DiagnosticMessage 677
4.10. 7 SysHC_NetworkConfig 680
4.10. 8 SysHC_NetworkInfo 683
4.10. 9 SysHC_UDiskPath 685
4.10. 10 SysHC_SetSerialParam 687
4.10. 11 SysHC_SetSerialSendRecvParam 689
4.10. 12 SysHC_GatewayConfig2 692
4.10. 13 GetFPGALogicVersion 693
4.10. 14 GetFPGASoftwareVersion 694
4.10. 15 GetBootVersion 696
4.10. 16 GetPLCVersion 697
4.10. 17 GetProductName 698
4.10. 18 GetRuntimeVersion 699
4.10. 19 GetSerialNumber 700
4.10. 20 GET_CPU_IOMODULE_DIAGNOSE 701
4.10. 21 SysHC_GetAllGL20ModuleInfo 703
4.10. 22 SysHC_GetGL20ModuleInfo 705

4.10. 23 SysHC_NTPClient 706
4.10. 24 SysHC_Appbackup 708
4. 11 Analog Waveform Instructions 711
4.11. 1 Instruction List 711
4.11. 2 BLINK 711
4.11. 3 GEN 713
4.11. 4 FREQ_MEASURE 718
Chapter 5 High-Speed I/O Instructions 646
5. 1 AM300/AM500/AC700 Local Counter Instructions 646
5.1. 1 Instruction List 646
5.1. 2 HC_Counter 646
5.1. 3 HC_Preset 649
5.1. 4 HC_TouchProbe 653
5.1. 5 HC_Compare 658
5.1. 6 HC_ArrayCompare 661
5.1. 7 HC_StepCompare 664
5.1. 8 HC_VirtualTouchProbe 669
5.1. 9 HC_ChangeGearingRatio 671
5.1. 10 HC_PWM 673
5.1. 11 HC_ReadStatus 674
5. 2 AM600 Local Counter Instructions 678
5.2. 1 Instruction List 678
5.2. 2 HC_Counter 678
5.2. 3 HC_SetCompare 681
5.2. 4 HC_PresetValue 682
5.2. 5 HC_EnableInterrupt 685
5.2. 6 HC_TouchProbe 686
5.2. 7 HC_MeasurePulseWidth 689
5.2. 8 HC_Sample 690
5.2. 9 HC_ReadBoolParameter 693
5.2. 10 HC_WriteBoolParameter 694
5.2. 11 HC_SetCompareM 696
5.2. 12 HC_SetRing 699
5.2. 13 HC_ResetCmpOutput 700
5.2. 14 HC_WriteInterruptParameter 701

5. 3 AM600 Local Pulse Output Instructions 704
5.3. 1 Instruction List 704
5.3. 2 MC_Jog_P 704
5.3. 3 MC_Home_P 706
5.3. 4 MC_MoveAbsolute_P 710
5.3. 5 MC_MoveRelative_P 713
5.3. 6 MC_MoveVelocity_P 716
5.3. 7 MC_Stop_P 718
5.3. 8 MC_Reset_P 720
5.3. 9 MC_Power_P 721
5.3. 10 MC_ReadParameter_P 722
5.3. 11 MC_WriteParameter_P 724
5.3. 12 MC_SetPosition_P 726
5.3. 13 MC_ReadStatus_P 727
5. 4 EtherCAT Remote Counter Instructions 732
5.4. 1 Instruction List 732
5.4. 2 HC_Counter_ETC 733
5.4. 3 HC_SetCompare_ETC 736
5.4. 4 HC_Presetvalue_ETC 738
5.4. 5 HC_TouchProbe_ETC 740
5.4. 6 HC_Reset_ETC 744
5.4. 7 Counter_ETC 745
5.4. 8 Counter_SetCompare_ETC 747
5.4. 9 Counter_Presetvalue_ETC 749
5.4. 10 Counter_TouchProbe_ETC 751
5.4. 11 Counter_Reset_ETC 754
Chapter 6 Communication Instructions 740
6. 1 Free TCP Communication Instructions 740
6.1. 1 Instruction List 740
6.1. 2 TCP_Server 740
6.1. 3 TCP_Client 741
6.1. 4 TCP_Connect 742
6.1. 5 TCP_Receive 744
6.1. 6 TCP_Send 745
6.1. 7 TCP Communication Instruction Examples 746

6. 2 Free UDP Communication Instructions 756
6.2. 1 Instruction List 756
6.2. 2 UDP_Peer 756
6.2. 3 UDP_Receive 757
6.2. 4 UDP_Send 759
6.2. 5 UDP Communication Instruction Examples 760
6. 3 CANopen Communication Instructions 766
6.3. 1 Instruction List 766
6.3. 2 NMT 766
6.3. 3 RECV_EMCY 770
6.3. 4 RECV_EMCY_DEV 773
6.3. 5 GET_LOCAL_NODE_ID 776
6.3. 6 GET_CANOPEN_KERNEL_STATE 779
6.3. 7 GET_STATE 782
6.3. 8 SDO_READ 785
6.3. 9 SDO_READ4 789
6.3. 10 SDO_WRITE 792
6.3. 11 SDO_WRITE4 796
6.3. 12 CANOPEN_COUNT 799
6.3. 13 GET_MST_STATISTICS 801
6.3. 14 COP_NMT 802
6.3. 15 COP_RECV_EMCY 806
6.3. 16 COP_RECV_EMCY_DEV 809
6.3. 17 COP_GET_LOCAL_NODE_ID 813
6.3. 18 COP_GET_CANOPEN_KERNEL_STATE 815
6.3. 19 COP_GET_STATE 818
6.3. 20 COP_SDO_READ 822
6.3. 21 COP_SDO_READ4 826
6.3. 22 COP_SDO_WRITE 830
6.3. 23 COP_SDO_WRITE4 834
6.3. 24 COP_CANOPEN_COUNT 837
6.3. 25 COP_GET_MST_STATISTICS 839
6. 4 CAN Free Protocol 840
6.4. 1 Instruction List 840
6.4. 2 CANBus_CtrlDevState 840

6.4. 3 CANBus_DevRxMsg 843
6.4. 4 CANBus_DevRxMultiMsg 845
6.4. 5 CANBus_DevTxMsg 848
6.4. 6 CANBus_GetDevDiagInfo 851
6.4. 7 CANBus_CetIoDrvVer 853
6.4. 8 CANBus_SetDevBaud 855
6.4. 9 Example of Using Function Blocks 858
6. 5 EtherCAT Communication Instructions 860
6.5. 1 Instruction List 860
6.5. 2 ETC_CO_SdoReadDWord 860
6.5. 3 ETC_CO_SdoRead4 863
6.5. 4 ETC_CO_SdoRead 866
6.5. 5 ETC_CO_SdoWriteDWord 869
6.5. 6 ETC_CO_SdoWrite4 872
6.5. 7 ETC_CO_SdoWrite 875
6.5. 8 IoDrvEtherCAT 878
6.5. 9 ETCSlave 880
6.5. 10 ETC_GetMasterInfo 881
6.5. 11 ETC_GetSlaveInfo 884
6. 6 EtherNet/IP Communication Instructions 887
6.6. 1 Instruction List 887
6.6. 2 IoDrvEtherNetIP 888
6.6. 3 RemoteAdapter 888
6.6. 4 Generic_Service 889
6.6. 5 Get_Attributes_All 890
6.6. 6 Get_Attribute_Single 891
6.6. 7 Set_Attributes_All 892
6.6. 8 Set_Attribute_Single 893
6.6. 9 Instances for Ethernet/IP Communication Reset Instructions 894
6.6. 10 Service Data Communication Examples for EtherNet/IP Master 897
6. 7 Modbus RTU Communication Instructions 903
6.7. 1 Instruction List 903
6.7. 2 MB_Connect 903
6.7. 3 MB_Master 906
6.7. 4 MB_Slave 910

6.7. 5 MB_TCP_Connect 913
6.7. 6 MB_Client 915
6.7. 7 MB_Server 919
6.7. 8 ModbusRTUChannel 923
6.7. 9 ModbusRTUSlave 925
6.7. 10 sysHC_ModbusRtuDeviceDiagnose 928
6.7. 11 sysHC_ModbusRtuSlaveDiagnose 929
6.7. 12 Examples of Modbus Communication Instructions 930
6. 8 Free Serial Communication Instructions 944
6. 9 Modbus TCP Communication Instructions 953
Chapter 7 CANopen Axis Control Instructions 874
7. 1 Instruction List 874
7. 2 MC_Power_CO 874
7. 3 MC_MoveAbsolute_CO 877
7. 4 MC_MoveRelative_CO 879
7. 5 MC_MoveVelocity_CO 881
7. 6 MC_Home_CO 883
7. 7 MC_Stop_CO 884
7. 8 MC_Halt_CO 886
7. 9 MC_Reset_CO 887
7. 10 MC_WriteParameter_CO 889
7. 11 MC_ReadParameter_CO 890
7. 12 MC_ReadStatus_CO 892
7. 13 MC_Jog_CO 894

Fundamental Safety Instructions

Safety Disclaimer

- This chapter presents essential safety instructions for proper use of the equipment. Before operating the equipment, read through the guide and comprehend all the safety instructions. Failure to comply with the safety precautions may result in equipment damage, severe physical injuries, or even death.
- "CAUTION", "WARNING", and "DANGER" messages in the guide are only examples and do not cover all safety precautions.
- Use this equipment according to the designated environment requirements. Damage caused by improper use is not covered by warranty.
- Inovance shall not be held liable for any physical injuries or property damage caused by improper use.

Safety Levels and Definitions



"DANGER" indicates that failure to comply with the notice will result in severe physical injury or even death.



"WARNING" indicates that failure to comply with the notice may result in severe physical injuries or even death.



"CAUTION" indicates that failure to comply with the notice may result in minor or moderate physical injuries or equipment damage.

General Safety Precautions

Unpacking
CAUTION ◆ Before unpacking, check whether the packing is intact without damage, water seepage, damp, and deformation. ◆ Unpack the package layer by layer. Do not strike the package violently. ◆ Check the surface of the equipment and accessories for any damage or rust. ◆ Check the equipment, accessories, and materials in the package against the packing list to ensure that no item is missing.
WARNING ◆ Do not install the equipment if you find damage, rust, or signs of use on the equipment or accessories upon unpacking. ◆ Do not install the equipment if you find water seepage or missing or damaged components upon unpacking. ◆ Do not install the equipment if you find the packing list does not conform to the equipment you received.
Storage and Transportation

CAUTION

- Handle the equipment with care and mind your steps. Failure to comply may result in physical injuries or equipment damage.
- When carrying the equipment with bare hands, hold the equipment casing firmly with care to prevent parts from falling. Failure to comply may result in physical injuries.
- Store and transport the equipment based on the storage and transportation requirements. Failure to comply will result in equipment damage.
- Avoid storing or transporting the equipment in environments with water splash, rain, direct sunlight, strong electric field, strong magnetic field, and strong vibration.
- Avoid storing the equipment for more than three months. Long-term storage requires stricter protection and necessary inspections.
- Pack the equipment properly before transportation by vehicle. Use an enclosed box for long-distance transportation.
- Never transport the equipment with other equipment or materials that may harm or have negative impacts on this equipment.

WARNING

- Large-scale or heavy equipment must be transported by qualified professionals using specialized hoisting equipment. Failure to comply may result in physical injuries or equipment damage.
- Before hoisting the equipment, ensure the equipment components such as the front cover and terminal blocks are secured firmly with screws. Loosely-connected components may fall off and result in physical injuries or equipment damage.
- Never stand or stay below the equipment when the equipment is being hoisted.
- When hoisting the equipment with a steel rope, ensure the equipment is hoisted at a constant speed without suffering from vibration or shock. Do not turn the equipment over or let the equipment stay hanging in the air. Failure to comply may result in physical injuries or equipment damage.

Installation

WARNING

- Read through the guide and safety instructions before installation.
- Do not install this equipment in places with strong electric or magnetic fields.
- Before installation, check that the mechanical strength of the installation site can bear the weight of the equipment. Failure to comply will result in mechanical hazards.
- Do not wear loose clothes or accessories during installation. Failure to comply may result in electric shock.
- When installing the equipment in a closed environment (such as a cabinet or casing), use a cooling device (such as a fan or air conditioner) to cool the environment down to the required temperature. Failure to comply may result in equipment over-temperature or fire.
- Do not retrofit the equipment.
- Do not fiddle with the bolts used to fix equipment components or the bolts marked in red.
- When the equipment is installed in a cabinet or final assembly, a fireproof enclosure providing both electrical and mechanical protections must be provided. The IP rating must meet IEC standards and local laws and regulations.
- Before installing equipment with strong electromagnetic interference, such as a transformer, install a shielding device for the equipment to prevent malfunction of this product.
- Install the equipment on metal or other incombustible objects. Keep the equipment away from combustible objects. Failure to comply will result in fire.

DANGER

- The equipment must be operated only by professionals with electrical knowledge. Non-professionals are not allowed to operate the equipment.

CAUTION

- Cover the top of the equipment with a piece of cloth or paper during installation. This is to prevent unwanted objects such as metal chippings, oil, and water from falling into the equipment and causing faults. After installation, remove the cloth or paper on the top of the equipment to prevent over-temperature caused by poor ventilation due to blocked ventilation holes.
- Resonance may occur when a machine supposed to run at a constant speed is running at variable speeds. In this case, install the vibration-proof rubber under the motor frame or use the vibration suppression function to reduce resonance.

Wiring

DANGER

- Do not allow non-professionals to perform equipment installation, wiring, maintenance, inspection, or parts replacement.
- Before wiring, cut off all the power supplies of the equipment. Wait for at least the time designated on the equipment warning label before further operations because residual voltage still exists after power-off. After waiting for the designated time, measure the DC voltage in the main circuit to ensure the DC voltage is within the safe voltage range. Failure to comply will result in electric shock.
- Do not perform wiring, remove the equipment cover, or touch the circuit board while power is on. Failure to comply will result in electric shock.
- Ensure that the equipment is well grounded. Failure to comply will result in an electric shock.

WARNING

- Do not connect the input power supply to the output end of the equipment. Failure to comply may result in equipment damage or even fire.
- When connecting a drive to the motor, make sure that the phase sequences of the drive and motor terminals are consistent to prevent reverse motor rotation.
- Use cables with required diameter and shield. Properly ground one end of the shield if a shielded cable is used.
- Fix the terminal screws with the tightening torque specified in the user guide. Improper tightening torque may overheat or damage the connecting part, resulting in fire.
- After wiring is done, check that all cables are connected properly and no screws, washers, or exposed cables are left inside the equipment. Failure to comply may result in electric shock or equipment damage.

CAUTION

- During wiring, follow the proper electrostatic discharge (ESD) procedures and wear an anti-static wrist strap. Failure to comply may result in damage to internal circuits of the equipment.
- Use shielded twisted pairs for the control circuit. Connect the shield to the grounding terminal of the equipment for grounding purpose. Failure to comply will result in equipment malfunction.

Power-on

DANGER

- Before power-on, check that the equipment is installed properly with reliable wiring and the motor can be restarted.
- Check that the power supply meets equipment requirements before power-on to prevent equipment damage or fire.
- After power-on, do not open the cabinet door or protective cover of the equipment, touch any terminal, or disassemble any unit or component of the equipment. Failure to comply will result in electric shock.

WARNING

- Perform a trial run after wiring and parameter setting to ensure the equipment operates safely. Failure to comply may result in physical injuries or equipment damage.
- Before power-on, check that the rated voltage of the equipment is consistent with that of the power supply. Failure to comply may result in fire.
- Before power-on, check that no one is near the equipment, motor, or machine. Failure to comply may result in physical injuries or even death.

Operation

DANGER

- The equipment must be operated only by professionals. Failure to comply will result in physical injuries or even death.
- Do not touch any connecting terminals or disassemble any unit or component of the equipment during operation. Failure to comply will result in electric shock.

WARNING

- Do not touch the equipment casing, fan, or resistor to check the temperature. Failure to comply may result in burns.
- Prevent metal or other objects from falling into the equipment during operation. Failure to comply may result in fire or equipment damage.

Maintenance

DANGER

- Do not allow non-professionals to perform equipment installation, wiring, maintenance, inspection, or parts replacement.
- Do not maintain the equipment while power is on. Failure to comply will result in electric shock.
- Before maintenance, cut off all the power supplies of the equipment and wait for at least the time designated on the equipment warning label.

WARNING

- Perform routine and periodic inspection and maintenance on the equipment according to maintenance requirements and keep a maintenance record.

Repair

DANGER

- Do not allow non-professionals to perform equipment installation, wiring, maintenance, inspection, or parts replacement.
- Do not repair the equipment while power is on. Failure to comply will result in electric shock.
- Cut off all power supplies to the equipment and wait for at least 10 minutes before equipment inspection or repair.

WARNING

- Submit the repair request according to the warranty agreement.
- When the equipment fails or is damaged, designate qualified technicians to troubleshoot and repair the equipment in accordance with the maintenance instructions and keep a maintenance record.
- Replace quick-wear parts of the equipment according to the replacement instructions.
- Do not use a damaged machine. Failure to comply may result in further damage.
- After the equipment is replaced, check the wiring and set parameters again.

Disposal

WARNING

- Dispose of retired equipment in accordance with local regulations and standards. Failure to comply may result in property damage, physical injuries, or even death.
- Dispose of or recycle retired equipment by following industry waste disposal standards to avoid environmental pollution.

Safety Labels

For safe operation and maintenance, follow the instructions on safety labels on the equipment. Do not stain or remove the safety labels. The following table describes the safety labels.

Safety Label	Description
	◆ Read the user guide before installation and operation. Failure to comply will result in an electric shock. ◆ Do not remove the cover during power-on or within 10 minutes after power-off. ◆ Before maintenance, inspection, and wiring, cut off the input and output power and wait for 10 minutes until the power indicator goes off.

Chapter 1 Overview of Basic Instructions

1. 1 Overview

This guide covers medium-sized programmable logic controllers (PLCs) of Inovance, including the AM and AC series.

This guide describes instructions and programs used by the medium-sized PLC programming software, InoProShop, to program controllers. This guide provides examples of instruction usage to help you get started quickly.

1. 2 Standard Data Type

Data Category	Data Type	Keyword	Memory Occupancy (Bit)	Value Range
Boolean	Boolean	BOOL	8	FALSE (0) or TRUE (1)
Integer	Byte	BYTE	8	0 to 255
	Word	WORD	16	0 to 65535
	Doubleword	DWORD	32	0 to 4294967295
	Long word	LWORD	64	0 to (2^64 − 1)
	Short integer	SINT	8	−128 to +127
	Unsigned short integer	USINT	8	0 to 255
	Integer	INT	16	−32768 to +32767
	Unsigned integer	UINT	16	0 to 65535
	Double integer	DINT	32	−2147483648 to +2147483647
	Unsigned double integer	UDINT	32	0 to 4294967295
	Long integer	LINT	64	−2^63 to (2^63 − 1)
	Unsigned long integer	ULINT	64	0 to (2^64 − 1)
Floating point	Single precision	REAL	32	1.175494351E-38 to 3.402823466E+38
	Double precision	LREAL	64	2.2250738585072014E-308 to 1.7976931348623158E+308
String	String	STRING	8 x N	-
	Unicode text string	WSTRING	16 x N	-
Time data	Time	TIME	32	T#0d0h0m0s0 ms to T#49d17h2m47s295 ms
		TIME_OF_DAY	32	TOD#0:0:0 to TOD#1193:02:47.295
		DATE	32	D#1970-1-1 to D#2106-02-06
		DATE_AND_TIME	32	DT#1970-1-1-0:0:0 to DT#2106-02-06-06:28:15
		LTIME	64	LTIME#0NS to LTIME#213503D23H34M33S709MS551US615NS

Note:

*N: Defined size of text string + 1

Chapter 2 Instruction List

Instruction Category	Name	FB/FC	Function
Comparison instructions	GT	FC	Greater than
	LT	FC	Less than
	GE	FC	Greater than or equal to
	LE	FC	Less than or equal to
	EQ	FC	Equal to
	NE	FC	Not equal to
	ZoneCmp	FC	Zone comparison
	TableCmp	FC	Table comparison
	AryCmpEQ	FC	Array comparison equal
	AryCmpNE	FC	Array comparison not equal
	AryCmpEQV	FC	Array value comparison equal
	AryCmpNEV	FC	Array value comparison not equal
	AryCmpLT	FC	Array comparison less than
	AryCmpLE	FC	Array comparison less than or equal
	AryCmpGT	FC	Array comparison greater than
	AryCmpGE	FC	Array comparison greater than or equal
	AryCmpLTV	FC	Array value comparison less than
	AryCmpLEV	FC	Array value comparison less than or equal
	AryCmpGTV	FC	Array value comparison greater than
	AryCmpGEV	FC	Array value comparison greater than or equal
Selection instructions	AryMax	FC	Array maximum
	AryMin	FC	Array minimum
	ArySearch	FC	Array search
	SEL	FC	Binary selection
	MUX	FC	Multiplexer
	MAX	FC	Maximum
	MIN	FC	Minimum
	LIMIT	FC	Limiter
Counter instructions	CTD_**	FB	Down-counter group
	CTU_**	FB	Up-counter group
	CTUD_**	FB	Up-down counter group
	CTD	FB	Down-counter
	CTU	FB	Up-counter
	CTUD	FB	Up-down counter
Timer instructions	AccumulationTimer	FB	Accumulation timer
	Timer	FC	Hundred-ms timer
	TP	FB	Timer pulse
	TON	FB	On-delay timer
	TOF	FB	Off-delay timer
	RTC	FB	Real-time clock

Instruction Category	Name	FB/FC	Function
Bit and word logic instructions	AND	FC	Logical AND
	OR	FC	Logical OR
	NOT	FC	Bit reversal
	XOR	FC	Logical exclusive OR
	PLS	FC	Rising edge output
	PLF	FC	Falling edge output
	ALT	FC	Alternate output
	BOUT	FC	Bit data output
	BSET	FC	Bit data set
	BRST	FC	Bit data reset
	AryAnd	FC	Array logical AND
	AryOr	FC	Array logical OR
	AryXor	FC	Array logical exclusive OR
	AryXorN	FC	Array logical exclusive NOR
	SR	FB	Set-priority keep
	RS	FB	Reset-priority keep
	R_TRIG	FB	Up trigger
	F_TRIG	FB	Down trigger
	EXTRACT	FC	Bit extraction
	PUTBIT	FC	Bit assignment
	PACK	FC	Bit packing
	UNPACK	FC	Bit unpacking
	BIT_AS_BYTE	FC	8-bit packing into byte
	BYTE_AS_BIT	FC	Byte unpacking to bits
	BIT_AS_WORD	FC	16-bit packing into word
	WORD_AS_BIT	FC	Word unpacking to bits
	BIT_AS_DWORD	FC	32-bit packing into doubleword
	DWORD_AS_BIT	FC	Doubleword unpacking to bits
	AryAnd	FC	Array logical AND
	AryOr	FC	Array logical OR
	AryXor	FC	Logical exclusive OR
	AryXorN	FC	Array logical exclusive NOR
Shift instructions	SHL	FC	Left shift
	SHR	FC	Right shift
	ArySHL	FC	Array N-element left shift
	ArySHR	FC	Array N-element right shift
	AryShiftReg	FC	Left shift register
	AryShiftRegLR	FC	Reversible shift register
	ROL	FC	Rotate left
	ROR	FC	Rotate right
	RCL	FC	Rotate left with carry
	RCR	FC	Rotate right with carry
	SFTL	FC	Bit shift left
	SFTR	FC	Bit shift right
	WSFL	FC	Word shift left
	WSFR	FC	Word shift right
	SFRD	FC	Shift read in FIFO mode
	SFWR	FC	Shift write in FIFO mode
	NSHLC	FC	Shift N-bits left with carry
	NSHRC	FC	Shift N-bits right with carry

Instruction Category	Name	FB/FC	Function
Data type conversion instructions	BOOL_TO_<TYPE>	FC	Conversion from bool to a type
	BYTE_TO_<TYPE>	FC	Conversion from byte to a type
	WORD_TO_<TYPE>	FC	Conversion from word to a type
	DWORD_TO_<TYPE>	FC	Conversion from doubleword to a type
	INT_TO_<TYPE>	FC	Conversion from integer to a type
	SINT_TO_<TYPE>	FC	Conversion from short integer to a type
	DINT_TO_<TYPE>	FC	Conversion from long integer to a type
	UDINT_TO_<TYPE>	FC	Conversion from unsigned long integer to a type
	REAL_TO_<TYPE>	FC	Conversion from real number to a type
	STRING_TO_<TYPE>	FC	Conversion from text string to a type
	TIME_TO_<TYPE>	FC	Conversion from time to a type
	TOD_TO_<TYPE>	FC	Conversion from time of day to a type
	DATE_TO_<TYPE>	FC	Conversion from date to a type
	DT_TO_<TYPE>	FC	Conversion from date and time to a type
	DtToString	FC	DT-to-text string conversion
	DateToString	FC	Date-to-text string conversion
	TodToString	FC	Time of day-to-text string conversion
	EnumToNum	FC	Enumeration-to-integer conversion
	NumToEnum	FC	Integer-to-enumeration conversion

Instruction Category	Name	FB/FC	Function
Data processing instructions	WriteNbit_**	FC	N-bit write group
	ReadNbit_**	FC	N-bit read group
	TransBits	FC	Move bits
	Encoder	FC	Bit encoder
	Decoder	FC	Bit decoder
	NumToDecString	FC	Integer-to-fixed-length decimal text string conversion
	NumToHexString	FC	Integer-to-fixed-length hexadecimal text string conversion
	HexStringToNum_**	FC	Hexadecimal text string-to-number conversion group
	AryToString	FC	Array-to-text string conversion
	StringToAry	FC	Text string-to-array conversion
	Clear	FC	Initialization
	SetBlock	FC	Block set
	MOVE	FC	Move
	BMOV	FC	Batch move
	FMOV	FC	Multi-point move
	BON	FC	Bit state check
	SUM	FC	Sum of ON bits
	BTOW	FC	Byte to word
	SWAP	FC	Swap bytes
	XCH	FC	Data exchange
	HEXinASCII_TO_BYTE	FC	ASCII-to-byte conversion
	BYTE_TO_HEXinASCII	FC	Byte-to-ASCII conversion
	WORD_AS_STRING	FC	Word-to-string conversion
	Byte_To_HexString	FC	Byte-to-hexadecimal text string conversion
	Word_To_HexString	FC	Word-to-hexadecimal text string conversion
	Dword_To_HexString	FC	Doubleword-to-hexadecimal text string conversion
	CopyRealToNum	FC	Bit pattern copy (real number to signed integer)
	CopyByteToNum	FC	Bit pattern copy (bit string to signed integer)
	CopyDwordToReal	FC	Bit pattern copy (bit string to real number)
	CopyNumToByte	FC	Bit pattern copy (signed integer to bit string)
	CopyNumToReal	FC	Bit pattern copy (signed integer to real number)
	CopyRealToDword	FC	Bit pattern copy (real number to bit string)
	BitCnt	FC	Bit counter
	AryByteTo	FC	Conversion from byte array
	DispartReal	FC	Separate mantissa and exponent
	UniteReal	FC	Combine real number mantissa and exponent
	Gray	FC	Gray code conversion
	ColmToLine_**	FC	Column-to-line conversion group
	LineToColm	FC	Line-to-column conversion
	FixNumToString	FC	Fixed-decimal number-to-text string conversion
	StringToFixNum	FC	Text string-to-fixed-decimal number conversion
	DispartDigit	FC	Four-bit separation
	UniteDigit_**	FC	Four-bit join group
	Dispart8Bit	FC	Byte data separation
	Unite8Bit_**	FC	Byte data join group
	ToAryByte	FC	Conversion to byte array
	AryToBCD	FC	Array-to-BCD conversion
	AryToBin	FC	Array-to-bit string conversion
	MoveDigit	FC	Move digit
	Exchange	FC	Data exchange
	AryExchange	FC	Array data exchange
	AryMove	FC	Array move
	SizeOfAry	FC	Get number of array elements
	GrayToBin_**	FC	Gray code-to-binary code conversion group
	BinToGray_**	FC	Binary code-to-gray code conversion
	ADD	FC	Addition
	SUB	FC	Subtraction
	MUL	FC	Multiplication
	DIV	FC	Division
	MOD	FC	Modulo-division
	ABS	FC	Absolute value
	SQRT	FC	Square root
	LN	FC	Natural logarithm
	LOG	FC	Logarithm base 10
	EXP	FC	Natural exponential operation
	EXPT	FC	Exponentiation
	SIN	FC	Sine
	COS	FC	Cosine
	TAN	FC	Tangent

Instruction Category	Name	FB/FC	Function
Time and time of day instructions	MULTIME	FC	Multiply time
	DIVTIME	FC	Divide time
	TruncTime	FC	Truncate time
	TruncDt	FC	Truncate date and time
	TruncTod	FC	Truncate time of day
	ChkLeapYear	FC	Check for leap year
	GetDaysOfMonth	FC	Get days in month
	GetDayOfWeek	FC	Get day of week
	GetWeekOfYear	FC	Get week number
	ADD_TOD_TIME	FC	Add time to time of day
	ADD_DT_TIME	FC	Add time to date and time
	SUB_TOD_TIME	FC	Subtract time from time of day
	SUB_TOD_TOD	FC	Subtract time of day
	SUB_DATE_DATE	FC	Subtract date
	DtToDateStruct	FC	Break down date and time
	DateStructToDt	FC	Join time
	GetSystemDate_sDT	FC	Get system date struct
	GetSystemTime	FB	Get system time
	SysHC_SetSystem-Date	FB	Set system date and timezone
	SysHC_GetSystem-Date	FB	Get system date and timezone
	DtToSec	FC	Convert date and time to seconds
	DateToSec	FC	Convert date to seconds
	TodToSec	FC	Convert time of day to seconds
	SecToDt	FC	Convert seconds to date and time
	SecToDate	FC	Convert seconds to date
	SecToTod	FC	Convert seconds to time of day
	TimeToNanoSec	FC	Convert time to nanoseconds
	TimeToSec	FC	Convert time to seconds
	NanoSecToTime	FC	Convert nanoseconds to time
	SecToTime	FC	Convert seconds to time
	DaysToMonth	FC	Convert days to month
Text string instructions	AddDelimiter	FC	Put REAL to text strings with delimiters
	AddDelimiter_LR	FC	Put LREAL to text strings with delimiters
	SubDelimiter	FC	Get text strings to LREAL minus delimiters
	SubDelimiter_LR	FC	Get text strings to LREAL minus delimiters
	ToUCase	FC	Convert to uppercase
	ToLCase	FC	Convert to lowercase
	StringSum	FC	Checksum calculation
	LEN	FC	Get characters count of string
	LEFT	FC	Intercept string, start from left
	RIGHT	FC	Intercept string, start from right
	MID	FC	Intercept string, start from specification
	CONCAT	FC	Contact two strings
	INSERT	FC	Insert string
	DELETE	FC	Delete chars form string
	FIND	FC	Get index of string in searched string
	REPLACE	FC	Replace some characters in a string
	GetByteLen	FC	Get byte length
	ClearString	FC	Clear string
	TrimL	FC	Trim string left
	TrimR	FC	Trim string right
Address operation instructions	ADR	FC	Get address
	^	FC	Get address content
	BITADR	FC	Get bit address
Queue instructions	StackPush	FC	Push onto stack
	StackFIFO	FC	First in first out
	StackLIFO	FC	Last in first out
	StackIns	FC	Insert into stack
	StackDel	FC	Delete from stack
	FIFO	FB	First in first out ring queue

Instruction Category	Name	FB/FC	Function
Table and range instructions	BZAND_TAB	FC	Dead zone control
	MEAN_TAB	FC	Average calculation
	ZONE_TAB	FC	Regional control
	ZRST_TAB	FC	Reset all
	SCL_TAB	FC	Fixed coordinates (different point coordinate)
	SORT_TAB	FC	Data sorting
	RAMP_TAB	FB	Ramp instruction
	WSUM_TAB	FC	Calculate the total value
	RecSearch	FC	Record search
	RecRangeSearch	FC	Range record search
	RecSort	FB	Record sort
	RecNum	FC	Get number of records
	RecMax	FC	Maximum record search
	RecMin	FC	Minimum record search
File operation instructions	FileOpen	FB	Open file
	FileClose	FB	Close file
	FileRead	FB	Read file
	FileWrite	FB	Write file
	FileCopy	FB	Copy file
	FileRemove	FB	Delete file
	FileRename	FB	Change file name
	FileWriteVar	FB	Write variable to file
	FileReadVar	FB	Read variable from file
	FileSeek	FB	Seek file
	FileGets	FB	Get text string
	FilePuts	FB	Put text string
	DirCreate	FB	Create directory
	DirRemove	FB	Delete directory
Analog calculation instructions	PD	FB	Universal PD control
	PID	FB	Universal PID control
	PID_FIXCYCLE	FB	Universal PID control that can manually set the cycle time
BCD conversion instructions	BCD_TO_INT	FC	BCD to signed integer
	INT_TO_BCD	FC	Signed integer to BCD
	BCD_TO_BYTE	FC	BCD to BYTE
	BYTE_TO_BCD	FC	BYTE to BCD
	BCD_TO_WORD	FC	BCD to word
	WORD_TO_BCD	FC	Word to BCD
	BCD_TO_DWORD	FC	BCD to double word
	DWORD_TO_BCD	FC	Double word to BCD
Filter instructions	LimitingFilter	FB	Limiting filter
	MedianFilter	FB	Median value filtering
	ArithmeticAverageFilter	FB	One-dimensional arithmetic mean filtering
	RecursiveAverageFilter	FB	Recursive averaging filter
	MedianAverageFilter	FB	Median average filtering
	LimitingAverageFilter	FB	Limiting average filtering
	FirstOrderLagFilter	FB	First order lag filtering
	WeightRecursiveAverageFilter	FB	Weighted recursive average filtering
	DebounceFilter	FB	Debounce filter
	LimitingDebounceFilter	FB	Limit debounce filter

Instruction Category	Name	FB/FC	Function
System instructions	SysHC_HWInfo	FB	Get PLC hardware information
	SysHC_SWInfo	FB	Get PLC software information
	SysHC_CPUInfo	FB	Get the CPU, memory, and boot time information of the PLC
	SysHC_CPUDiagnose	FB	Get the CPU hardware diagnosis information
	DiagnosticMessage	FB	Get CPU diagnostic message
	SysHC_NetworkConfig	FB	Configure PLC network ports
	SysHC_NetworkInfo	FB	Obtain PLC network configuration
	SysHC_UDiskPath	FB	Obtain the path of the U disk
	SysHC_SetSerial-Param	FB	Parameter setting for serial port free protocol
	SysHC_SetSerialSen-dRecvParam	FB	Send/Receive parameter setting for serial port free protocol
	SysHC_GatewayCon-fig2	FB	Gateway setting
	GetFPGALogicVersion	FB	Get the FPGA logic version number
	GetFPGASoftwareVer-sion	FB	Get the FPGA software version number
	GetBootVersion	FB	Get the boot version
	GetPLCVersion	FB	Get the PLC firmware version
	GetProductName	FB	Get the PLC name
	GetRuntimeVersion	FB	Get runtime version
	GetSerialNumber	FB	Get the PLC serial number (unique)
	GET_CPU_IOMOD-ULE_DIAGNOSE	FB	Get diagnostic information of CPU local modules
Analog waveform instructions	BLINK	FB	Simulate a blinking signal
	GEN	FB	Analog waveform generator
	FREQ_MEASURE	FB	Analog frequency measurement
AM300/ AM500/ AC700 local counter instructions	HC_Counter	FB	High-speed counter enable
	HC_Preset	FB	High-speed counter preset value
	HC_TouchProbe	FB	High-speed counter probe
	HC_Compare	FB	High-speed counter comparison
	HC_ArrayCompare	FB	High-speed counter array comparison
	HC_StepCompare	FB	High-speed counter equal distance comparison
	HC_VirtualTouch-Probe	FB	Virtual probe
	HC_ChangeGearin-gRatio	FB	High-speed counter set gearing ratio
	HC_PWM	FB	High-speed counter PWM
	HC_ReadStatus	FB	High-speed counter read status
AM600 local counter instructions	HC_Counter	FB	Enable counter
	HC_SetCompare	FB	Set counter comparison output
	HC_PresetValue	FB	Set counter value from preset value
	HC_EnableInterrupt	FB	Interrupt counter
	HC_TouchProbe	FB	Counter probe latch
	HC_MeasurePulse-Width	FB	Measure counter pulse width
	HC_Sample	FB	Counter start to sample
	HC_ReadBoolParame-ter	FB	Read counter boolean parameters
	HC_WriteBoolParame-ter	FB	Write counter boolean parameters
	HC_SetCompareM	FB	Counter multi-segment comparison
	HC_SetRing	FB	Set counter mode
	HC_ResetCmpOutput	FB	Reset comparison output
	HC_WriteInterruptPa-rameter	FB	Write counter interrupt parameters

Instruction Category	Name	FB/FC	Function
AM600 local pulse output instructions	MC_Jog_P	FB	Jog local pulse axis
	MC_Home_P	FB	Local pulse axis homing
	MC_MoveAbsolute_P	FB	Local pulse axis move absolute
	MC_MoveRelative_P	FB	Local pulse axis move relative
	MC_MoveVelocity_P	FB	Local pulse axis move velocity
	MC_Stop_P	FB	Stop pulse axis motion
	MC_Reset_P	FB	Reset pulse axis errors
	MC_Power_P	FB	Enable local pulse axis
	MC_ReadParameter_P	FB	Read pulse axis parameters
	MC_WriteParameter_P	FB	Write pulse axis parameters
	MC_SetPosition_P	FB	Set pulse axis position
	MC_ReadStatus_P	FB	Read pulse axis status
GR10-2HCE remote counter instructions	HC_Counter_ETC	FB	Enable 2HCE counter
	HC_SetCompare_ETC	FB	2HCE counter comparison output
	HC_Presetvalue_ETC	FB	Preset 2HCE counter value
	HC_TouchProbe_ETC	FB	2HCE counter probe
	HC_Reset_ETC	FB	Reset 2HCE counter
GR10-8PBE remote counter instructions	Counter_ETC	FB	Enable 2HCE counter
	Counter_SetCompare_ETC	FB	2HCE counter comparison output
	Counter_Presetvalue_ETC	FB	Preset 2HCE counter value
	Counter_TouchProbe_ETC	FB	2HCE counter probe
	Counter_Reset_ETC	FB	Reset 2HCE counter
Free TCP communication instructions	TCP_Server	FB	Create TCP server communication
	TCP_Client	FB	Create TCP client communication
	TCP_Connect	FB	Create TCP client, and connect to server
	TCP_Receive	FB	Receive data
	TCP_Send	FB	Send data
Free UDP communication instructions	UDP_Peer	FB	Activate UDP peer
	UDP_Receive	FB	Receive UDP data
	UDP_Send	FB	Send UDP data
CANopen communication instructions	NMT	FB	NMT services
	RECV_EMCY	FB	Receive emergency object from any device
	RECV_EMCY_DEV	FB	Receive emergency object from input device
	GET_LOCAL_NODE_ID	FB	Get the CANopen NodeID of the local device
	GET_CANOPEN_KERNEL_STATE	FB	Get the current state of the CANopen kernel
	GET_STATE	FB	Get the CANopen state of device
	SDO_READ	FB	Read specific object from object dictionary of device
	SDO_READ4	FB	Read specific object up to 4 bytes from object dictionary of device
	SDO_WRITE	FB	Write specific object in object dictionary of device
	SDO_WRITE4	FB	Write specific object up to 4 bytes in object dictionary of device
	CANOPEN_COUNT	FB	CANopen count
	GET_MST_STATISTICS	FB	Get MST statistics
EtherCAT communication instructions	ETC_CO_SdoReadDWord	FB	Read EtherCAT slave parameters as DWORD
	ETC_CO_SdoRead4	FB	Read EtherCAT slave parameters that are no longer than 4 bytes
	ETC_CO_SdoRead	FB	Read EtherCAT slave parameters
	ETC_CO_SdoWriteDWord	FB	Write EtherCAT slave parameters as DWORD
	ETC_CO_SdoWrite4	FB	Write EtherCAT slave parameters that are no longer than 4 bytes
	ETC_CO_SdoWrite	FB	Write EtherCAT slave parameters
	IoDrvEtherCAT	FB	Implicit instance of EtherCAT master
	ETCSlave	FB	Implicit instance of EtherCAT slave

Instruction Category	Name	FB/FC	Function
EtherNet/IP communication instructions	IoDrvEtherNetIP	FB	Implicit instance of EtherNet/IP
	RemoteAdapter	FB	Remote EtherNet/IP adapter
	Generic_Service	FB	Performs generic service at EtherNet/IP adapter
	Get_Attributes_All	FB	Get attributes of specific instance of all CIP object
	Get_Attribute_Single	FB	Get attribute of specific instance of single CIP object
	Set_Attributes_All	FB	Set attributes of specific instance of all CIP object
	Set_Attribute_Single	FB	Set attribute of specific instance of single CIP object
Modbus RTU communication instructions	ModbusRTUChannel	FB	Trigger channel of ModbusRTU master
	sysHC_ModbusRtuDeviceDiagnose	FB	Get diagnostic information of ModbusRTU slave devices
	sysHC_ModbusRtuSlaveDiagnose	FB	Get diagnostic data of ModbusRTU master accessing slaves
Modbus TCP communication instructions	ModbusTCPChannel	FB	Trigger channel of ModbusTcp master
	ModbusTCPServerConfig	FB	Modbus TCP slave configuration
	ModbusTCPSlaveDisable	FB	Modbus TCP slave disable
	ModbusTCPSlaveSetIPAddr	FB	Modbus TCP slave IP address configuration
	sysHC_ModbusTcpDeviceDiagnose	FB	Get diagnostic information of ModbusTcp slave devices
	sysHC_ModbusTcpSlaveDiagnose	FB	Get diagnostic data of ModbusTcp master accessing slaves
CANopen axis control instructions	MC_Power_CO	FB	Enable axis
	MC_MoveAbsolute_CO	FB	Move absolute position
	MC_MoveRelative_CO	FB	Move relative position
	MC_MoveVelocity_CO	FB	Move in specified speed
	MC_Home_CO	FB	Move homing
	MC_Stop_CO	FB	Stop motion
	MC_Halt_CO	FB	Stop motion and motion can be interrupted
	MC_Reset_CO	FB	Reset status
	MC_WriteParameter_CO	FB	Write specific axis parameters
	MC_ReadParameter_CO	FB	Read specific axis parameters
	MC_ReadStatus_CO	FB	Read motion status
	MC_Halt_CO	FB	Stop motion and motion can be interrupted

Chapter 3 Basic Instructions

3. 1 Comparison Instructions

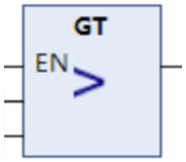
3.1. 1 Instruction List

Instruction Category	Name	FB/FC	Function
Comparison instructions	GT	FC	Greater than
	LT	FC	Less than
	GE	FC	Greater than or equal to
	LE	FC	Less than or equal to
	EQ	FC	Equal to
	NE	FC	Not equal to
	ZoneCmp	FC	Zone comparison
	TableCmp	FC	Table comparison
	AryCmpEQ	FC	Array comparison equal
	AryCmpNE	FC	Array comparison not equal
	AryCmpEQV	FC	Array value comparison equal
	AryCmpNEV	FC	Array value comparison not equal
	AryCmpLT	FC	Array comparison less than
	AryCmpLE	FC	Array comparison less than or equal
	AryCmpGT	FC	Array comparison greater than
	AryCmpGE	FC	Array comparison greater than or equal
	AryCmpLTV	FC	Array value comparison less than
	AryCmpLEV	FC	Array value comparison less than or equal
	AryCmpGTV	FC	Array value comparison greater than
	AryCmpGEV	FC	Array value comparison greater than or equal

3.1. 2 GT

This instruction compares two input values. When the first input value is greater than the second input value, the output is TRUE. Otherwise, the output is FALSE.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
GT	Greater than	FC		>

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
In1	Input 1	-	Depends on data types	0	Data 1
In2	Input 2	-	Depends on data types	0	Data 2

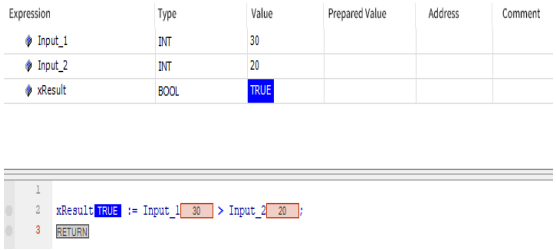
Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
Out	Return value	BOOL	[FALSE, TRUE]	FALSE	Comparison result

	Bool-ean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In1	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√
In2	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√
Out	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

■ Program example

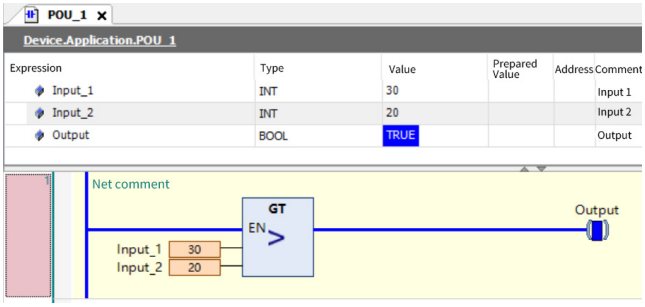
ST: When Input_1 is greater than Input_2, the output is TRUE. Otherwise, the output is FALSE.



Expression	Type	Value	Prepared Value	Address	Comment
Input_1	INT	30			
Input_2	INT	20			
xResult	BOOL	TRUE			

```
1
2  xResult TRUE := Input_1 30 > Input_2 20 ;
3  RETURN
```

LD: When Input_1 is greater than Input_2, the output is TRUE. Otherwise, the output is FALSE.



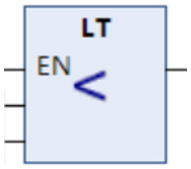
■ Precautions

When the data types of two input variables are inconsistent, a compilation error occurs.

3.1. 3 LT

This instruction compares two input values. When the first input value is less than the second input value, the output is TRUE. Otherwise, the output is FALSE.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
LT	Less than	FC		<

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
In1	Input 1	-	Depends on data types	0	Data 1
In2	Input 2	-	Depends on data types	0	Data 2

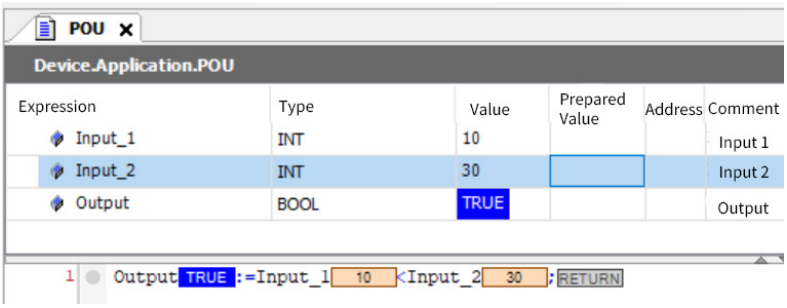
Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
Out	Return value	BOOL	[FALSE, TRUE]	FALSE	Comparison result

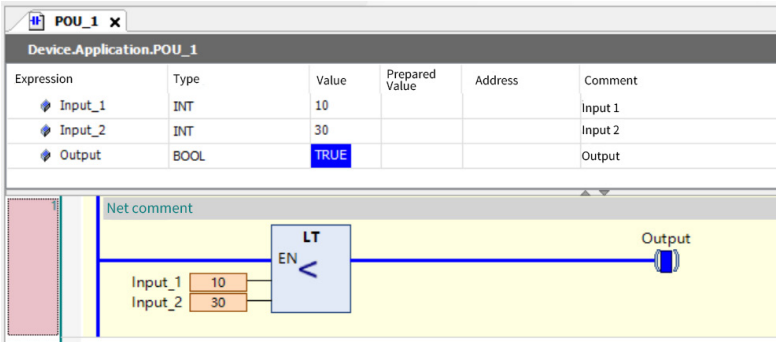
	Bool-ean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In1	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√
In2	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√
Out	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

■ Program example

ST: When Input_1 is less than Input_2, the output is TRUE. Otherwise, the output is FALSE.



LD: When Input_1 is less than Input_2, the output is TRUE. Otherwise, the output is FALSE.



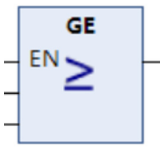
■ Precautions

When the data types of two input variables are inconsistent, a compilation error occurs.

3.1.4 GE

This instruction compares two input values. When the first input value is greater than or equal to the second input value, the output is TRUE. Otherwise, the output is FALSE.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
GE	Greater than or equal to	FC		>=

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
In1	Input 1	-	Depends on data types	0	Data 1
In2	Input 2	-	Depends on data types	0	Data 2

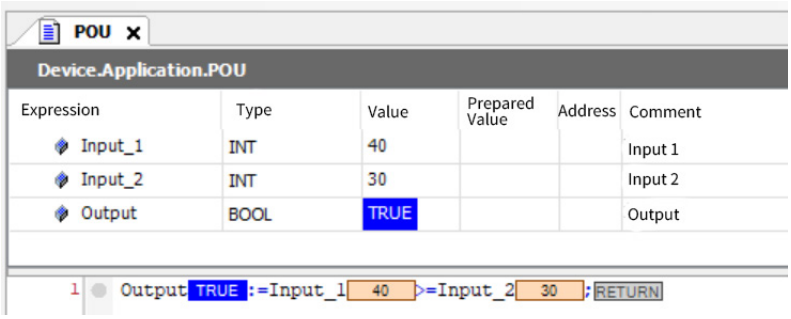
Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
Out	Return value	BOOL	[FALSE, TRUE]	FALSE	Comparison result

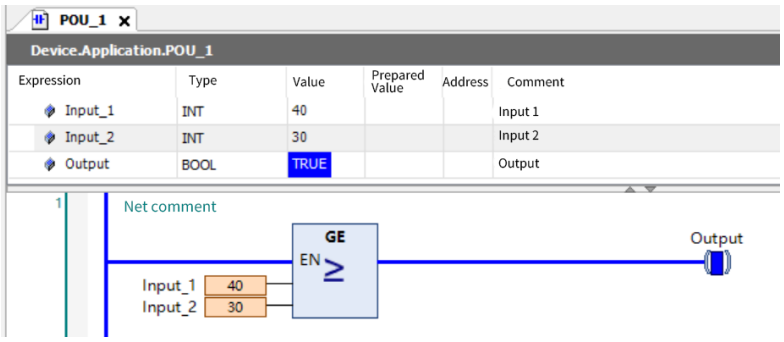
	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In1	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√
In2	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√
Out	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

■ Program example

ST: When Input_1 is greater than or equal to Input_2, the output is TRUE. Otherwise, the output is FALSE.



LD: When Input_1 is greater than or equal to Input_2, the output is TRUE. Otherwise, the output is FALSE.



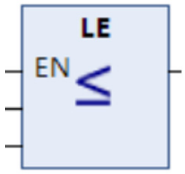
■ Precautions

When the data types of two input variables are inconsistent, a compilation error occurs.

3.1.5 LE

This instruction compares input values. When the first input value is less than or equal to the second input value, the output is TRUE. Otherwise, the output is FALSE.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
LE	Less than or equal to	FC		<=

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
In1	Input 1	-	Depends on data types	0	Data 1
In2	Input 2	-	Depends on data types	0	Data 2

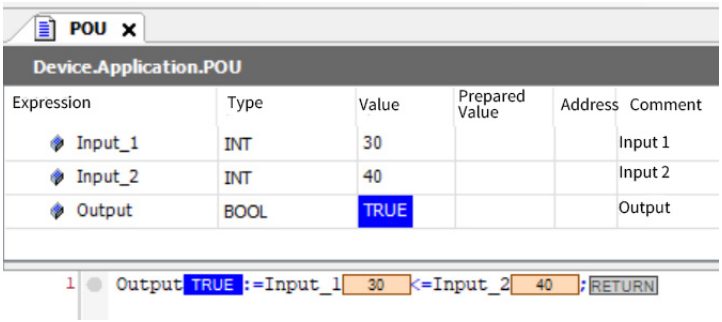
Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
Out	Return value	BOOL	[FALSE, TRUE]	FALSE	Comparison result

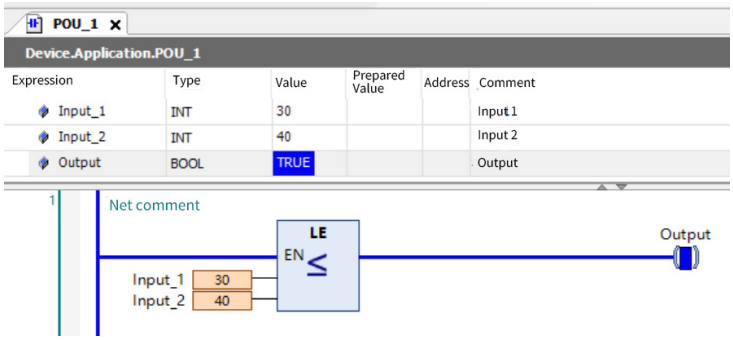
	Bool-ean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In1	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√
In2	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√
Out	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

■ Program example

ST: When Input_1 is less than or equal to Input_2, the output is TRUE. Otherwise, the output is FALSE.



LD: When Input_1 is less than or equal to Input_2, the output is TRUE. Otherwise, the output is FALSE.



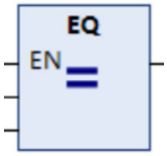
■ Precautions

When the data types of two input variables are inconsistent, a compilation error occurs.

3.1. 6 EQ

This instruction compares two input values. When the first input value is equal to the second input value, the output is TRUE. Otherwise, the output is FALSE.

■ Instruction description

Instruction	Name	FB/FC	LD Expression	ST Expression
EQ	Equal to	FC		=

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
In1	Input 1	-	Depends on data types	0	Data 1
In2	Input 2	-	Depends on data types	0	Data 2

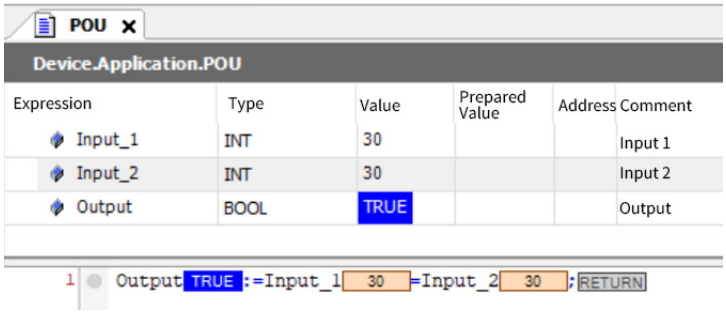
Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
Out	Return value	BOOL	[FALSE, TRUE]	FALSE	Comparison result

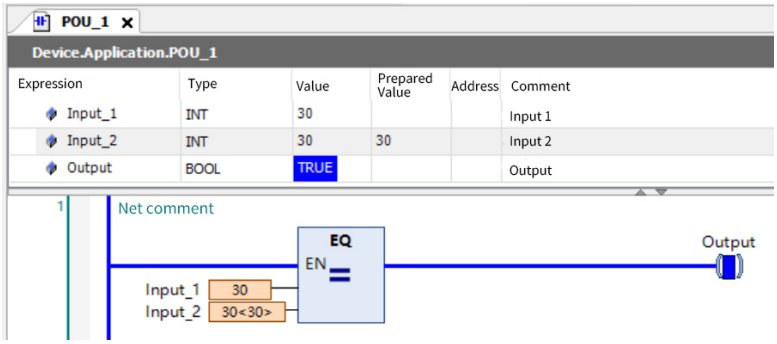
	Bool-ean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In1	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√
In2	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√
Out	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

■ Program example

ST: When Input_1 is equal to Input_2, the output is TRUE. Otherwise, the output is FALSE.



LD:

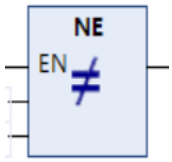


When Input_1 is equal to Input_2, the output is TRUE. Otherwise, the output is FALSE.

3.1.7 NE

This instruction compares two input values. When the first input value is not equal to the second input value, the output is TRUE. Otherwise, the output is FALSE.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
NE	Not equal to	FC		q:=x1<>x2;

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
In1	Input 1	-	Depends on data types	0	Data 1
In2	Input 2	-	Depends on data types	0	Data 2

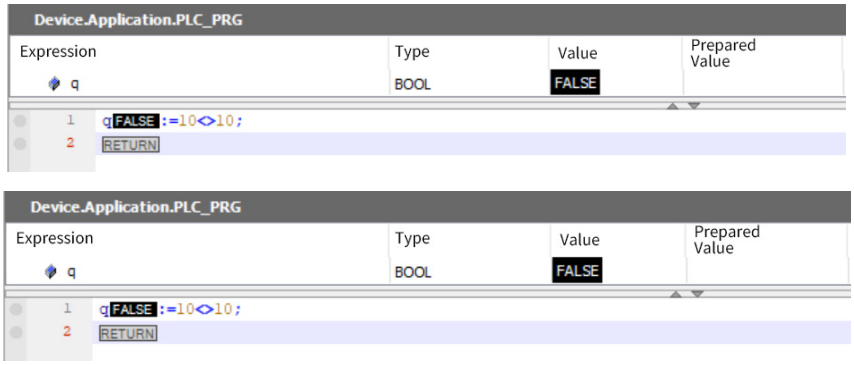
Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
Out	Return value	BOOL	[FALSE, TRUE]	FALSE	Comparison result

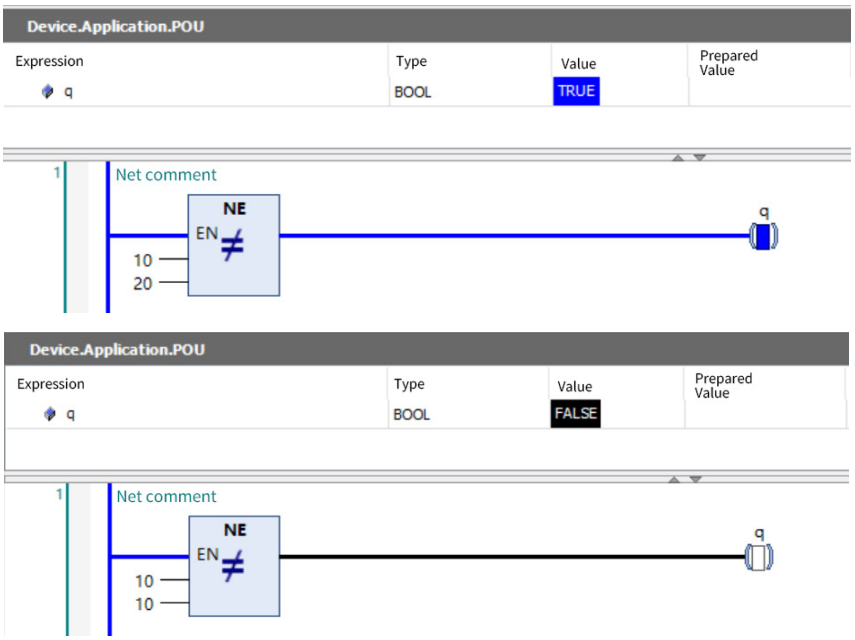
	Bool-ean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In1	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√
In2	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√
Out	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

■ Program example

ST: When Input_1 is not equal to Input_2, the output is TRUE. Otherwise, the output is FALSE.



LD: When Input_1 is not equal to Input_2, the output is TRUE. Otherwise, the output is FALSE.



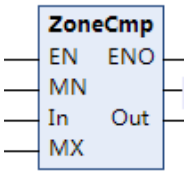
■ Precautions

When the data types of two input variables are inconsistent, a compilation error occurs.

3.1. 8 ZoneCmp

This instruction determines whether the comparison data is within the specified maximum and minimum values.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
ZoneCmp	Zone comparison	FC		ZoneCmp(MN,In,MX);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
MN	Upper limit	-	Depends on data types	-	Upper limit
In	Compared data	-	Depends on data types	-	Value to compare
MX	Lower limit	-	Depends on data types	-	Lower limit

Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
Out	Comparison result	BOOL	[FALSE, TRUE]	FALSE	Comparison result

	Bool-ean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
MN	-	-	-	-	-	√	√	√	√	√	√	√	√	√	√	√	√	√	√	-
In	-	-	-	-	-	√	√	√	√	√	√	√	√	√	√	√	√	√	√	-
MX	-	-	-	-	-	√	√	√	√	√	√	√	√	√	√	√	√	√	√	-
Out	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

■ Function

This instruction determines whether comparison data In is between maximum value MX and minimum value MN.

If MX ⩾ In ⩾ MN, Out is TRUE. Otherwise, Out is FALSE.

The relationships between values with data types that are not integers or real numbers are determined as given in the following table.

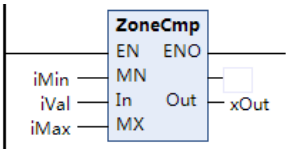
Data Type	Relationship
TIME	The numerically larger value is considered to be larger.
DATE, TOD, or DT	Later dates or TOD are considered to be larger.

■ Program example

The following example shows MN is INT#10, In is INT#20, and MX is INT#30.

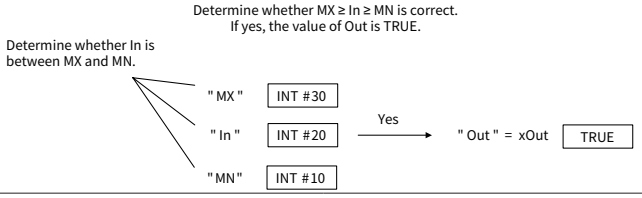
	LD	ST
Defined variable		
Program		
Running result		
Work principle		

```
VAR
    iMin : INT := 10;
    iVal : INT := 20;
    iMax : INT := 30;
    xOut : BOOL;
END_VAR
```




```
ZoneCmp(
    MN := iMin,
    In := iVal,
    MX := iMax,
    Out => xOut
);
```

iMin	INT	10
iVal	INT	20
iMax	INT	30
xOut	BOOL	TRUE



■ Precautions

· When the data types of In, MX, and MN are different, an error occurs during compilation.

· If In, MX, and MN are real numbers that contain non-terminating decimal numbers, a rounding error may cause unexpected processing results.

· Two values that are positive infinity or two values that are negative infinity are equivalent.

· When the value of In is nonnumeric data, the value of Out is FALSE.

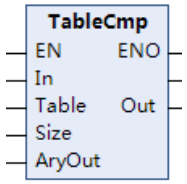
· When the value of MN is greater than the value of MX, an error occurs and the return value is FALSE.

· If either MX or MN contains nonnumeric data, an error occurs and the return value is FALSE.

3.1. 9 TableCmp

This instruction compares the comparison data with multiple defined ranges in a comparison table.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
TableCmp	Table comparison	FC		TableCmp (In :=, Table :=, Size :=, AryOut :=, Out =>,);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
In	Compared data	-	Depends on data types	-	Value to compare
Table[] (two-dimensional array)	Comparison table	-	Depends on data types	-	Two-dimensional array that contains the elements for the defined ranges
Size	Comparison size	UINT	Depends on data types	1	Number of elements in Table[] to be compared with In
AryOut[] (array)	Individual comparison result array	BOOL	[FALSE, TRUE]	FALSE	Comparison results for Table[] elements TRUE: Consistent FALSE: Inconsistent

Output Variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
Out	Return value	BOOL	[FALSE, TRUE]	FALSE	Comparison result

	Bool-ean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In	-	-	-	-	-	√	√	√	√	√	√	√	√	√	√	-	-	-	-	-
Table[]	Two-dimensional array with elements that have the same data type as In																			
Size	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-
AryOut[]	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

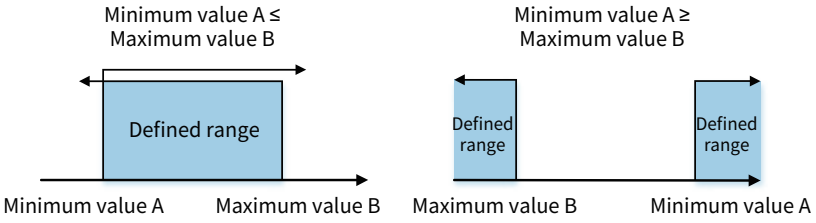
■ Function

This instruction compares the comparison data In with defined ranges of Size specified in a comparison table Table[].

Table[] is a two-dimensional array. The first dimension contains the numbers of the defined ranges. In the second dimension, element 0 is the minimum value A of the defined range, and element 1 is the maximum value B of the defined range.

Defined Range	Minimum Value A	Maximum Value B
Range 0	Table[0,0]	Table[0,1]
Range 1	Table[1,0]	Table[1,1]
...	...	...
Range Size - 1	Table[Size-1,0]	Table[Size-1,1]

Use the minimum value A and maximum value B to specify the defined range. Both values are included in the defined range.

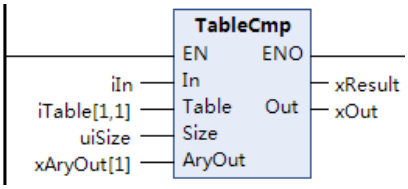


The results of comparing In and Table[] are stored in individual comparison result array AryOut[]. If In is within the defined range for element i, AryOut[i] is TRUE. If it is not within the range, AryOut[i] is FALSE. If all Size elements of AryOut[] are TRUE, the comparison result Out is TRUE. Otherwise, it is FALSE.

■ Program example

	LD	ST
Defined variable		
Program		
Running result		

```
VAR
    iIn     : INT := 5;
    iTable  : ARRAY [1..3,1..2] OF INT := [0,2,0,10,10,2];
    uiSize  : UINT := 3;
    xAryOut : ARRAY [1..3] OF BOOL;
    xOut    : BOOL;
    xResult : BOOL;
END_VAR
```




```
xResult := TableCmp(In := iIn,
        Table := iTable[1,1],
        Size := uiSize,
        AryOut := xAryOut[1],
        Out => xOut
        );
```

iIn	INT	5
iTable	ARRAY [1..3, 1..2] OF INT	
iTable[1, 1]	INT	0
iTable[1, 2]	INT	2
iTable[2, 1]	INT	0
iTable[2, 2]	INT	10
iTable[3, 1]	INT	10
iTable[3, 2]	INT	2
uiSize	UINT	3
xAryOut	ARRAY [1..3] OF BOOL	
xAryOut[1]	BOOL	FALSE
xAryOut[2]	BOOL	TRUE
xAryOut[3]	BOOL	FALSE
xOut	BOOL	FALSE
xResult	BOOL	TRUE

■ Precautions

· Use the same data type for In and Table[]. Otherwise, the return value is FALSE.

· Use a two-dimensional array for Table[] and ensure that the size of the second dimension is 2. Otherwise, the output is unexpected.

· If the AryOut[] array is greater than the value of Size, the comparison results are stored in AryOut[0] to AryOut[Size − 1]. Other elements of the array do not change.

· If real numbers that contain non-terminating decimal numbers are compared, a rounding error may cause unexpected processing results.

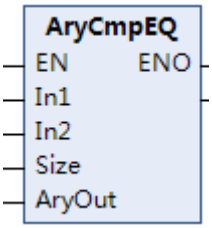
· When the value of Size is 0, the value of Out is FALSE and AryOut[] does not change.

· When the value of Size exceeds the size of the AryOut[] array or the size of the first dimension of the Table[] array, an error may occur in program execution and even the PLC may break down.

3.1. 10 AryCmpEQ

This instruction compares the values of elements in two arrays.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
AryCmpEQ	Array comparison equal	FC		AryCmpEQ (In1 :=, In2 :=, Size := , AryOut :=);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
In1[] (array)	Comparison array 1	-	Depends on data types	-	Comparison array 1
In2[] (array)	Comparison array 2	-	Depends on data types	-	Comparison array 2
Size	Number of comparison elements	UINT	Depends on data types	1	Number of elements to compare

In-out variables

In-Out Variable	Name	Data Type	Value Range	Initial Value	Description
AryOut[] (array)	Comparison result array	-	Depends on data types	-	Comparison result array

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In1[] (array)	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	-	-	-	-	-
In2[] (array)	Array with the same data type as In1[]																			
Size	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-

AryOut[] (array)	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

■ Function

This instruction compares the values of the elements with the same element numbers in two arrays (In1[0] to In1[Size − 1] and In2[0] to In2[Size − 1]). The comparison results are stored in the elements with the corresponding element numbers (AryOut[0] to AryOut[Size − 1]).

When In1[i] is equal to In2[i], AryOut[i] is TRUE. Otherwise, it is FALSE.

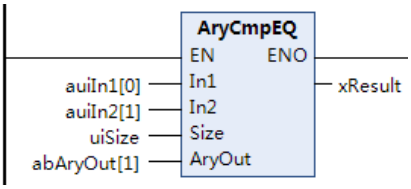
■ Program example

LD

Defined variable

```
VAR
    auiIn1      : ARRAY [0..4] OF UINT := [11,22,33,44,55];
    auiIn2      : ARRAY [0..4] OF UINT := [11,21,22,33,55];
    abAryOut    : ARRAY [0..4] OF BOOL;
    uiSize      : UINT := 4;
    xResult     : BOOL;
END_VAR
```

Program



ST

```
xResult := AryCmpEQ(In1:= auiIn1[0],
    In2:= auiIn2[1],
    Size:= uisize,
    AryOut:= abAryOut[1]);
```

Running result

auiIn1	ARRAY [0..4] OF UINT	
auiIn1[0]	UINT	11
auiIn1[1]	UINT	22
auiIn1[2]	UINT	33
auiIn1[3]	UINT	44
auiIn1[4]	UINT	55
auiIn2	ARRAY [0..4] OF UINT	
auiIn2[0]	UINT	11
auiIn2[1]	UINT	21
auiIn2[2]	UINT	22
auiIn2[3]	UINT	33
auiIn2[4]	UINT	55
abAryOut	ARRAY [0..4] OF BOOL	
abAryOut[0]	BOOL	FALSE
abAryOut[1]	BOOL	FALSE
abAryOut[2]	BOOL	TRUE
abAryOut[3]	BOOL	TRUE
abAryOut[4]	BOOL	FALSE
uiSize	UINT	4
xResult	BOOL	TRUE

■ Precautions

· Use the same data type for In1[] and In2[]. If they are different, an error occurs during compilation.

· If In1[] and In2[] are real numbers that contain non-terminating decimal numbers, a rounding error may cause unexpected processing results.

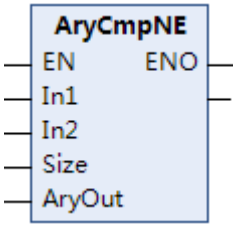
· When the value of Size is 0, the return value is TRUE and AryOut[] does not change.

· When the value of Size exceeds the In1[], In2[], or AryOut[] array, the return value is FALSE and AryOut[] does not change.

3.1. 11 AryCmpNE

This instruction compares the values of elements in two arrays.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
AryCmpNE	Array comparison not equal	FC		AryCmpNE (In1 :=, In2 :=, Size := , AryOut :=);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
In1[] (array)	Comparison array 1	-	Depends on data types	-	Comparison array 1
In2[] (array)	Comparison array 2	-	Depends on data types	-	Comparison array 2
Size	Number of comparison elements	UINT	Depends on data types	1	Number of elements to compare

In-out variables

In-Out Variable	Name	Data Type	Value Range	Initial Value	Description
AryOut[] (array)	Comparison result array	-	Depends on data types	-	Comparison result array

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In1[] (array)	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	-	-	-	-	-
In2[] (array)	Array with the same data type as In1[]																			
Size	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-
AryOut[] (array)	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

■ Function

This instruction compares the values of the elements with the same element numbers in two arrays (In1[0] to In1[Size − 1] and In2[0] to In2[Size − 1]). The comparison results are stored in the elements with the corresponding element numbers (AryOut[0] to AryOut[Size − 1]).

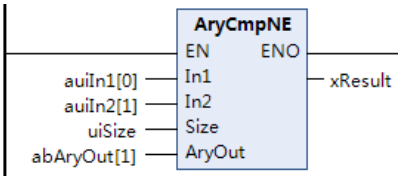
When In1[i] is not equal to In2[i], AryOut[i] is TRUE. Otherwise, it is FALSE.

■ Program example

<table>
<tr><th></th><th>LD</th><th>ST</th></tr>
<tr><td>Defined variable</td><td colspan="2">

```
VAR
    auiIn1      : ARRAY [0..4] OF UINT := [11,22,33,44,55];
    auiIn2      : ARRAY [0..4] OF UINT := [11,21,22,33,55];
    abAryOut    : ARRAY [0..4] OF BOOL;
    uiSize      : UINT := 4;
    xResult     : BOOL;
END_VAR
```

</td></tr>
<tr><td>Program</td><td>



</td><td>

```
xResult := AryCmpNE(In1:= auiIn1[0],
    In2:= auiIn2[1],
    Size:= uisize,
    AryOut:= abAryOut[1]);
```

</td></tr>
<tr><td>Running result</td><td colspan="2">

auiIn1	ARRAY [0..4] OF UINT	
auiIn1[0]	UINT	11
auiIn1[1]	UINT	22
auiIn1[2]	UINT	33
auiIn1[3]	UINT	44
auiIn1[4]	UINT	55
auiIn2	ARRAY [0..4] OF UINT	
auiIn2[0]	UINT	11
auiIn2[1]	UINT	21
auiIn2[2]	UINT	22
auiIn2[3]	UINT	33
auiIn2[4]	UINT	55
abAryOut	ARRAY [0..4] OF BOOL	
abAryOut[0]	BOOL	FALSE
abAryOut[1]	BOOL	TRUE
abAryOut[2]	BOOL	FALSE
abAryOut[3]	BOOL	FALSE
abAryOut[4]	BOOL	TRUE
uiSize	UINT	4
xResult	BOOL	TRUE

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

■ Precautions

· Use the same data type for In1[] and In2[]. If they are different, an error occurs during compilation.

· If In1[] and In2[] are real numbers that contain non-terminating decimal numbers, a rounding error may cause unexpected processing results.

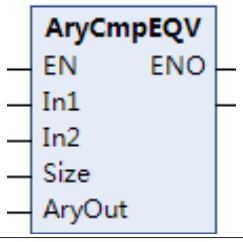
· When the value of Size is 0, the return value is TRUE and AryOut[] does not change.

· When the value of Size exceeds the In1[], In2[], or AryOut[] array, the return value is FALSE and AryOut[] does not change.

3.1. 12 AryCmpEQV

This instruction compares array elements with values.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
AryCmpEQV	Array value comparison equal	FC		AryCmpEQV (In1 :=, In2 :=, Size := , AryOut :=);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
In1[] (array)	Comparison array	-	Depends on data types	-	Comparison array
In2[] (array)	Comparison variable	-	Depends on data types	-	Variable to compare
Size	Number of comparison elements	UINT	Depends on data types	1	Number of elements to compare

In-out variables

In-Out Variable	Name	Data Type	Value Range	Initial Value	Description
AryOut[] (array)	Comparison result array	-	Depends on data types	-	Comparison result array

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In1[] (array)	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	-	-	-	-	-
In2[] (array)	Array with the same data type as the elements of In1[]																			
Size	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-
AryOut[] (array)	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

■ Function

This instruction compares the comparison array (In1[0] to In1[Size-1]) with the comparison variable In2. The comparison results are stored in the elements with the corresponding element numbers (AryOut[0] to AryOut[Size − 1]).

When In1[i] is equal to In2, AryOut[i] is TRUE. Otherwise, it is FALSE.

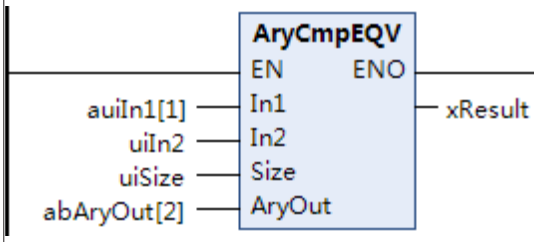
■ Program example

	LD	ST

Defined variable

```
VAR
    auiIn1      : ARRAY [0..4] OF UINT := [11,22,33,44,55];
    uiIn2       : UINT := 33;
    abAryOut    : ARRAY [0..4] OF BOOL;
    uiSize      : UINT := 3;
    xResult     : BOOL;
END_VAR
```

Program




```
xResult := AryCmpEQV(In1:= auiIn1[1],
    In2:= uiIn2,
    Size:= uisize,
```

Running result

auiIn1	ARRAY [0..4] OF UINT	
auiIn1[0]	UINT	11
auiIn1[1]	UINT	22
auiIn1[2]	UINT	33
auiIn1[3]	UINT	44
auiIn1[4]	UINT	55
uiIn2	UINT	33
abAryOut	ARRAY [0..4] OF BOOL	
abAryOut[0]	BOOL	FALSE
abAryOut[1]	BOOL	FALSE
abAryOut[2]	BOOL	FALSE
abAryOut[3]	BOOL	TRUE
abAryOut[4]	BOOL	FALSE
uiSize	UINT	3
xResult	BOOL	TRUE

■ Precautions

· Use the same data type for In1[] and In2. If they are different, an error occurs during compilation.

· If In1[] and In2 are real numbers that contain non-terminating decimal numbers, a rounding error may cause unexpected processing results.

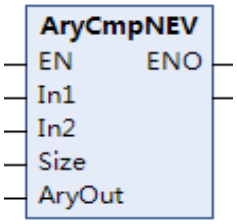
· When the value of Size is 0, the return value is TRUE and AryOut[] does not change.

· When the value of Size exceeds the In1[] or AryOut[] array, the return value is FALSE and AryOut[] does not change.

3.1. 13 AryCmpNEV

This instruction compares array elements with values.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
AryCmpNEV	Array value comparison not equal	FC		AryCmpNEV (In1 :=, In2 :=, Size := , AryOut :=);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
In1[] (array)	Comparison array	-	Depends on data types	-	Comparison array
In2	Comparison variable	-	Depends on data types	-	Variable to compare
Size	Number of comparison elements	UINT	Depends on data types	1	Number of elements to compare

In-out variables

In-Out Variable	Name	Data Type	Value Range	Initial Value	Description
AryOut[] (array)	Comparison result array	-	Depends on data types	-	Comparison result array

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In1[] (array)	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	-	-	-	-	-
In2[] (array)	Array with the same data type as the elements of In1[]																			
Size	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-
AryOut[] (array)	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

■ Function

This instruction compares the comparison array (In1[0] to In1[Size-1]) with the comparison variable In2. The comparison results are stored in the elements with the corresponding element numbers (AryOut[0] to AryOut[Size − 1]).

When In1[i] is not equal to In2, AryOut[i] is TRUE. Otherwise, it is FALSE.

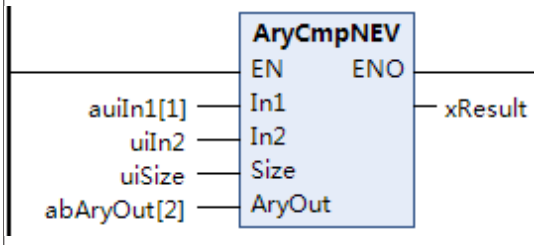
■ Program example

	LD	ST

Defined variable

```
VAR
    auiIn1      : ARRAY [0..4] OF UINT := [11,22,33,44,55];
    uiIn2       : UINT := 33;
    abAryOut    : ARRAY [0..4] OF BOOL;
    uiSize      : UINT := 3;
    xResult     : BOOL;
END_VAR
```

Program




```
xResult := AryCmpNEV(In1:= auiIn1[1],
    In2:= uiIn2,
    Size:= uisize,
    AryOut:= abAryOut[2]);
```

Running result

auiIn1	ARRAY [0..4] OF UINT	
auiIn1[0]	UINT	11
auiIn1[1]	UINT	22
auiIn1[2]	UINT	33
auiIn1[3]	UINT	44
auiIn1[4]	UINT	55
uiIn2	UINT	33
abAryOut	ARRAY [0..4] OF BOOL	
abAryOut[0]	BOOL	FALSE
abAryOut[1]	BOOL	FALSE
abAryOut[2]	BOOL	TRUE
abAryOut[3]	BOOL	FALSE
abAryOut[4]	BOOL	TRUE
uiSize	UINT	3
xResult	BOOL	TRUE

■ Precautions

· Use the same data type for In1[] and In2. If they are different, an error occurs during compilation.

· If In1[] and In2 are real numbers that contain non-terminating decimal numbers, a rounding error may cause unexpected processing results.

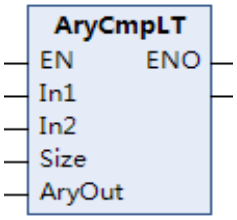
· When the value of Size is 0, the return value is TRUE and AryOut[] does not change.

· When the value of Size exceeds the In1[] or AryOut[] array, the return value is FALSE and AryOut[] does not change.

3.1. 14 AryCmpLT

This instruction compares (less than) the values of elements in two arrays.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
AryCmpLT	Array value comparison less than	FC		AryCmpLT (In1 :=, In2 :=, Size := , AryOut :=);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
In1[] (array)	Comparison array	-	Depends on data types	-	Comparison array 1
In2[] (array)	Comparison variable	-	Depends on data types	-	Comparison array 2
Size	Number of comparison elements	UINT	Depends on data types	1	Number of elements to compare

In-out variables

In-Out Variable	Name	Data Type	Value Range	Initial Value	Description
AryOut[] (array)	Comparison result array	-	Depends on data types	-	Comparison result array

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In1[] (array)	-	-	-	-	-	√	√	√	√	√	√	√	√	√	√	-	-	-	-	-
In2[] (array)	Array with the same data type as In1[]																			
Size	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-
AryOut[] (array)	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

■ Function

This instruction compares (less than) the values of the elements with the same element numbers in two arrays (In1[0] to In1[Size − 1] and In2[0] to In2[Size − 1]). The comparison results are stored in the elements with the corresponding element numbers (AryOut[0] to AryOut[Size − 1]).

When In1[i] is less than In2[i], AryOut[i] is TRUE. Otherwise, it is FALSE.

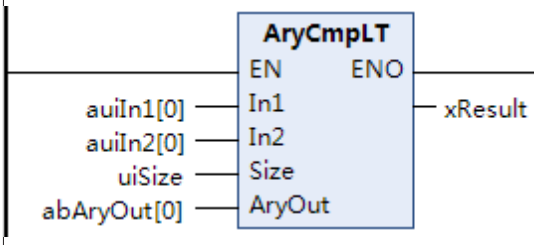
■ Program example

	LD	ST

Defined variable

```
VAR
    auiIn1      : ARRAY [0..4] OF UINT := [11,22,33,44,55];
    auiIn2      : ARRAY [0..4] OF UINT := [10,22,34,43,55];
    abAryOut    : ARRAY [0..4] OF BOOL;
    uiSize      : UINT := 5;
    xResult     : BOOL;
END_VAR
```

Program




```
xResult := AryCmpLT(In1:= auiIn1[0],
    In2:= auiIn2[0],
    Size:= uiSize,
    AryOut:= abAryOut[0]);
```

Running result

auiIn1	ARRAY [0..4] OF UINT	
auiIn1[0]	UINT	11
auiIn1[1]	UINT	22
auiIn1[2]	UINT	33
auiIn1[3]	UINT	44
auiIn1[4]	UINT	55
auiIn2	ARRAY [0..4] OF UINT	
auiIn2[0]	UINT	10
auiIn2[1]	UINT	22
auiIn2[2]	UINT	34
auiIn2[3]	UINT	43
auiIn2[4]	UINT	55
abAryOut	ARRAY [0..4] OF BOOL	
abAryOut[0]	BOOL	FALSE
abAryOut[1]	BOOL	FALSE
abAryOut[2]	BOOL	TRUE
abAryOut[3]	BOOL	FALSE
abAryOut[4]	BOOL	FALSE
uiSize	UINT	5
xResult	BOOL	TRUE

■ Precautions

· Use the same data type for In1[] and In2[]. If they are different, an error occurs during compilation.

· If In1[] and In2[] are real numbers that contain non-terminating decimal numbers, a rounding error may cause unexpected processing results.

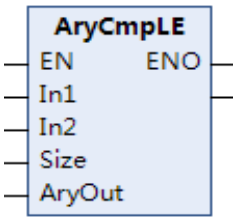
· When the value of Size is 0, the return value is TRUE and AryOut[] does not change.

· When the value of Size exceeds the In1[], In2[], or AryOut[] array, the return value is FALSE and AryOut[] does not change.

3.1. 15 AryCmpLE

This instruction compares (less than or equal to) the values of elements in two arrays.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
AryCmpLE	Array comparison less than or equal	FC		AryCmpLE (In1 :=, In2 :=, Size := , AryOut :=);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
In1[] (array)	Comparison array 1	-	Depends on data types	-	Comparison array 1
In2[] (array)	Comparison array 2	-	Depends on data types	-	Comparison array 2
Size	Number of comparison elements	UINT	Depends on data types	1	Number of elements to compare

In-out variables

In-Out Variable	Name	Data Type	Value Range	Initial Value	Description
AryOut[] (array)	Comparison result array	-	Depends on data types	-	Comparison result array

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In1[] (array)	-	-	-	-	-	√	√	√	√	√	√	√	√	√	√	-	-	-	-	-
In2[] (array)	Array with the same data type as In1[]																			
Size	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-
AryOut[] (array)	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

■ Function

This instruction compares the comparison array (In1[0] to In1[Size-1]) with the comparison variable In2. The comparison results are stored in the elements with the corresponding element numbers (AryOut[0] to AryOut[Size − 1]).

When In1[i] is not equal to In2, AryOut[i] is TRUE. Otherwise, it is FALSE.

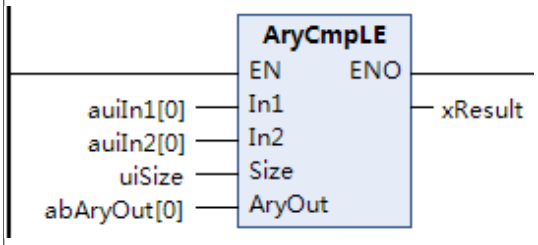
■ Program example

	LD	ST

Defined variable

```
VAR
    auiIn1      : ARRAY [0..4] OF UINT := [11,22,33,44,55];
    auiIn2      : ARRAY [0..4] OF UINT := [10,22,34,43,55];
    abAryOut    : ARRAY [0..4] OF BOOL;
    uiSize      : UINT := 5;
    xResult     : BOOL;
END_VAR
```

Program




```
xResult := AryCmpLE(In1:= auiIn1[0],
    In2:= auiIn2[0],
    Size:= uiSize,
    AryOut:= abAryOut[0]);
```

Running result

auiIn1	ARRAY [0..4] OF UINT	
auiIn1[0]	UINT	11
auiIn1[1]	UINT	22
auiIn1[2]	UINT	33
auiIn1[3]	UINT	44
auiIn1[4]	UINT	55
auiIn2	ARRAY [0..4] OF UINT	
auiIn2[0]	UINT	10
auiIn2[1]	UINT	22
auiIn2[2]	UINT	34
auiIn2[3]	UINT	43
auiIn2[4]	UINT	55
abAryOut	ARRAY [0..4] OF BOOL	
abAryOut[0]	BOOL	FALSE
abAryOut[1]	BOOL	TRUE
abAryOut[2]	BOOL	TRUE
abAryOut[3]	BOOL	FALSE
abAryOut[4]	BOOL	TRUE
uiSize	UINT	5
xResult	BOOL	TRUE

■ Precautions

· Use the same data type for In1[] and In2[]. If they are different, an error occurs during compilation.

· If In1[] and In2[] are real numbers that contain non-terminating decimal numbers, a rounding error may cause unexpected processing results.

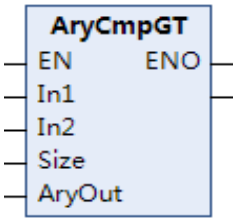
· When the value of Size is 0, the return value is TRUE and AryOut[] does not change.

· When the value of Size exceeds the In1[], In2[], or AryOut[] array, the return value is FALSE and AryOut[] does not change.

3.1. 16 AryCmpGT

This instruction compares (greater than) the values of elements in two arrays.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
AryCmpGT	Array comparison greater than	FC		AryCmpGT (In1 :=, In2 :=, Size := , AryOut :=);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
In1[] (array)	Comparison array 1	-	Depends on data types	-	Comparison array 1
In2[] (array)	Comparison array 2	-	Depends on data types	-	Comparison array 2
Size	Number of comparison elements	UINT	Depends on data types	1	Number of elements to compare

In-out variables

In-Out Variable	Name	Data Type	Value Range	Initial Value	Description
AryOut[] (array)	Comparison result array	-	Depends on data types	-	Comparison result array

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In1[] (array)	-	-	-	-	-	√	√	√	√	√	√	√	√	√	√	-	-	-	-	-
In2[] (array)	Array with the same data type as In1[]																			
Size	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-
AryOut[] (array)	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

■ Function

This instruction compares (greater than) the values of the elements with the same element numbers in two arrays (In1[0] to In1[Size − 1] and In2[0] to In2[Size − 1]). The comparison results are stored in the elements with the corresponding element numbers (AryOut[0] to AryOut[Size − 1]).

When In1[i] is greater than In2[i], AryOut[i] is TRUE. Otherwise, it is FALSE.

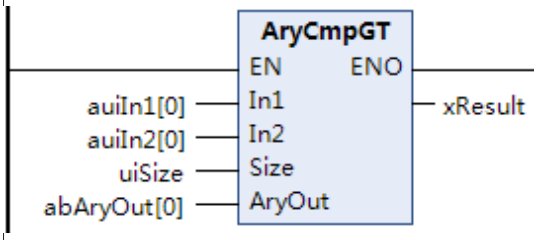
■ Program example

	LD	ST

Defined variable

```
VAR
    auiIn1      : ARRAY [0..4] OF UINT := [11,22,33,44,55];
    auiIn2      : ARRAY [0..4] OF UINT := [10,22,34,43,55];
    abAryOut    : ARRAY [0..4] OF BOOL;
    uiSize      : UINT := 5;
    xResult     : BOOL;
END_VAR
```

Program




```
xResult := AryCmpGT(In1:= auiIn1[0],
    In2:= auiIn2[0],
    Size:= uiSize,
    AryOut:= abAryOut[0]);
```

Running result

auiIn1	ARRAY [0..4] OF UINT	
auiIn1[0]	UINT	11
auiIn1[1]	UINT	22
auiIn1[2]	UINT	33
auiIn1[3]	UINT	44
auiIn1[4]	UINT	55
auiIn2	ARRAY [0..4] OF UINT	
auiIn2[0]	UINT	10
auiIn2[1]	UINT	22
auiIn2[2]	UINT	34
auiIn2[3]	UINT	43
auiIn2[4]	UINT	55
abAryOut	ARRAY [0..4] OF BOOL	
abAryOut[0]	BOOL	TRUE
abAryOut[1]	BOOL	FALSE
abAryOut[2]	BOOL	FALSE
abAryOut[3]	BOOL	TRUE
abAryOut[4]	BOOL	FALSE
uiSize	UINT	5
xResult	BOOL	TRUE

■ Precautions

Use the same data type for In1[] and In2[]. If they are different, an error occurs during compilation.

If In1[] and In2[] are real numbers that contain non-terminating decimal numbers, a rounding error may cause unexpected processing results.

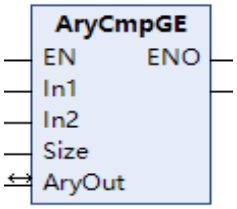
When the value of Size is 0, the return value is TRUE and AryOut[] does not change.

When the value of Size exceeds the In1[], In2[], or AryOut[] array, the return value is FALSE and AryOut[] does not change.

3.1. 17 AryCmpGE

This instruction compares (greater than or equal to) the values of elements in two arrays.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
AryCmpGE	Array comparison greater than or equal	FC		AryCmpGE (In1 :=, In2 :=, Size := , AryOut :=);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
In1[] (array)	Comparison array 1	-	Depends on data types	-	Comparison array 1
In2[] (array)	Comparison array 2	-	Depends on data types	-	Comparison array 2
Size	Number of comparison elements	UINT	Depends on data types	1	Number of elements to compare

In-out variables

In-Out Variable	Name	Data Type	Value Range	Initial Value	Description
AryOut[] (array)	Comparison result array	-	Depends on data types	-	Comparison result array

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In1[] (array)	-	-	-	-	-	√	√	√	√	√	√	√	√	√	√	-	-	-	-	-
In2[] (array)	Array with the same data type as In1[]																			
Size	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-
AryOut[] (array)	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

■ Function

This instruction compares (greater than or equal to) the values of the elements with the same element numbers in two arrays (In1[0] to In1[Size − 1] and In2[0] to In2[Size − 1]). The comparison results are stored in the elements with the corresponding element numbers (AryOut[0] to AryOut[Size − 1]).

When In1[i] is greater than or equal to In2[i], AryOut[i] is TRUE. Otherwise, it is FALSE.

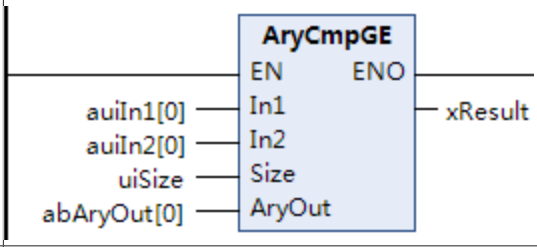
■ Program example

	LD	ST

Defined variable

```
VAR
    auiIn1      : ARRAY [0..4] OF UINT := [11,22,33,44,55];
    auiIn2      : ARRAY [0..4] OF UINT := [10,22,34,43,55];
    abAryOut    : ARRAY [0..4] OF BOOL;
    uiSize      : UINT := 5;
    xResult     : BOOL;
END_VAR
```

Program




```
xResult := AryCmpGE(In1:= auiIn1[0],
    In2:= auiIn2[0],
    Size:= uiSize,
    AryOut:= abAryOut[0]);
```

Running result

auiIn1	ARRAY [0..4] OF UINT	
auiIn1[0]	UINT	11
auiIn1[1]	UINT	22
auiIn1[2]	UINT	33
auiIn1[3]	UINT	44
auiIn1[4]	UINT	55
auiIn2	ARRAY [0..4] OF UINT	
auiIn2[0]	UINT	10
auiIn2[1]	UINT	22
auiIn2[2]	UINT	34
auiIn2[3]	UINT	43
auiIn2[4]	UINT	55
abAryOut	ARRAY [0..4] OF BOOL	
abAryOut[0]	BOOL	TRUE
abAryOut[1]	BOOL	TRUE
abAryOut[2]	BOOL	FALSE
abAryOut[3]	BOOL	TRUE
abAryOut[4]	BOOL	TRUE
uiSize	UINT	5
xResult	BOOL	TRUE

■ Precautions

Use the same data type for In1[] and In2[]. If they are different, an error occurs during compilation.

If In1[] and In2[] are real numbers that contain non-terminating decimal numbers, a rounding error may cause unexpected processing results.

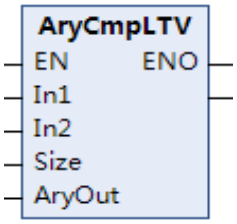
When the value of Size is 0, the return value is TRUE and AryOut[] does not change.

When the value of Size exceeds the In1[], In2[], or AryOut[] array, the return value is FALSE and AryOut[] does not change.

3.1. 18 AryCmpLTV

This instruction compares (less than) the values of elements in arrays.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
AryCmpLTV	Array value comparison less than	FC		AryCmpLTV (In1 :=, In2 :=, Size := , AryOut :=);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
In1[] (array)	Comparison array	-	Depends on data types	-	Comparison array
In2[] (array)	Comparison variable	-	Depends on data types	-	Variable to compare
Size	Number of comparison elements	UINT	Depends on data types	1	Number of elements to compare

In-out variables

In-Out Variable	Name	Data Type	Value Range	Initial Value	Description
AryOut[] (array)	Comparison result array	-	Depends on data types	-	Comparison result array

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In1[] (array)	-	-	-	-	-	√	√	√	√	√	√	√	√	√	√	-	-	-	-	-
In2[] (array)	Array with the same data type as the elements of In1[]																			
Size	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-
AryOut[] (array)	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

■ Function

This instruction compares (less than) the comparison array (In1[0] to In1[Size-1]) with the comparison variable In2. The comparison results are stored in the elements with the corresponding element numbers (AryOut[0] to AryOut[Size − 1]).

When In1[i] is less than In2, AryOut[i] is TRUE. Otherwise, it is FALSE.

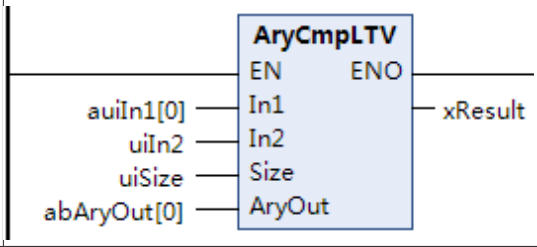
■ Program example

	LD	ST

Defined variable

```
VAR
    auiIn1      : ARRAY [0..4] OF UINT := [11,22,33,44,55];
    uiIn2       : UINT := 33;
    abAryOut    : ARRAY [0..4] OF BOOL;
    uiSize      : UINT := 5;
    xResult     : BOOL;
END_VAR
```

Program




```
xResult := AryCmpLTV(In1:= auiIn1[0],
    In2:= uiIn2,
    Size:= uiSize,
    AryOut:= abAryOut[0]);
```

Running result

auiIn1	ARRAY [0..4] OF UINT	
auiIn1[0]	UINT	11
auiIn1[1]	UINT	22
auiIn1[2]	UINT	33
auiIn1[3]	UINT	44
auiIn1[4]	UINT	55
uiIn2	UINT	33
abAryOut	ARRAY [0..4] OF BOOL	
abAryOut[0]	BOOL	TRUE
abAryOut[1]	BOOL	TRUE
abAryOut[2]	BOOL	FALSE
abAryOut[3]	BOOL	FALSE
abAryOut[4]	BOOL	FALSE
uiSize	UINT	5
xResult	BOOL	TRUE

■ Precautions

Use the same data type for In1[] and In2. If they are different, an error occurs during compilation.

If In1[] and In2 are real numbers that contain non-terminating decimal numbers, a rounding error may cause unexpected processing results.

When the value of Size is 0, the return value is TRUE and AryOut[] does not change.

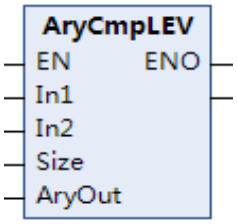
When the value of Size exceeds the In1[] or AryOut[] array, the return value is FALSE and AryOut[] does not change.

3.1. 19 AryCmpLEV

This instruction compares (less than or equal to) the values of elements in arrays.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression

AryCmpLEV	Array value comparison less than or equal	FC		AryCmpLEV (In1 :=, In2 :=, Size := , AryOut :=);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
In1[] (array)	Comparison array	-	Depends on data types	-	Comparison array
In2[] (array)	Comparison variable	-	Depends on data types	-	Variable to compare
Size	Number of comparison elements	UINT	Depends on data types	1	Number of elements to compare

In-out variables

In-Out Variable	Name	Data Type	Value Range	Initial Value	Description
AryOut[] (array)	Comparison result array	-	Depends on data types	-	Comparison result array

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In1[] (array)	-	-	-	-	-	√	√	√	√	√	√	√	√	√	√	-	-	-	-	-
In2[] (array)	Array with the same data type as the elements of In1[]																			
Size	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-
AryOut[] (array)	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

■ Function

This instruction compares (less than or equal to) the comparison array (In1[0] to In1[Size-1]) with the comparison variable In2. The comparison results are stored in the elements with the corresponding element numbers (AryOut[0] to AryOut[Size − 1]).

When In1[i] is less than or equal to In2, AryOut[i] is TRUE. Otherwise, it is FALSE.

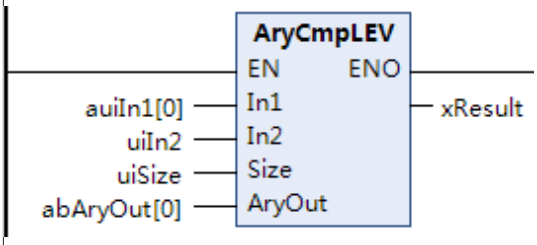
■ Program example

	LD	ST

Defined variable

```
VAR
    auiIn1      : ARRAY [0..4] OF UINT := [11,22,33,44,55];
    uiIn2       : UINT := 33;
    abAryOut    : ARRAY [0..4] OF BOOL;
    uiSize      : UINT := 3;
    xResult     : BOOL;
END_VAR
```

Program




```
xResult := AryCmpLEV(In1:= auiIn1[0],
    In2:= uiIn2,
    Size:= uiSize,
    AryOut:= abAryOut[0]);
```

Running result

auiIn1	ARRAY [0..4] OF UINT	
auiIn1[0]	UINT	11
auiIn1[1]	UINT	22
auiIn1[2]	UINT	33
auiIn1[3]	UINT	44
auiIn1[4]	UINT	55
uiIn2	UINT	33
abAryOut	ARRAY [0..4] OF BOOL	
abAryOut[0]	BOOL	TRUE
abAryOut[1]	BOOL	TRUE
abAryOut[2]	BOOL	TRUE
abAryOut[3]	BOOL	FALSE
abAryOut[4]	BOOL	FALSE
uiSize	UINT	5
xResult	BOOL	TRUE

■ Precautions

Use the same data type for In1[] and In2. If they are different, an error occurs during compilation.

If In1[] and In2 are real numbers that contain non-terminating decimal numbers, a rounding error may cause unexpected processing results.

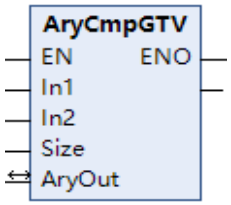
When the value of Size is 0, the return value is TRUE and AryOut[] does not change.

When the value of Size exceeds the In1[] or AryOut[] array, the return value is FALSE and AryOut[] does not change.

3.1. 20 AryCmpGTV

This instruction compares (greater than) the values of elements in arrays.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
AryCmpGTV	Array value comparison greater than	FC		AryCmpGTV (In1 :=, In2 :=, Size := , AryOut :=);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
In1[] (array)	Comparison array	-	Depends on data types	-	Comparison array
In2[] (array)	Comparison variable	-	Depends on data types	-	Variable to compare
Size	Number of comparison elements	UINT	Depends on data types	1	Number of elements to compare

In-out variables

In-Out Variable	Name	Data Type	Value Range	Initial Value	Description
AryOut[] (array)	Comparison result array	-	Depends on data types	-	Comparison result array

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In1[] (array)	-	-	-	-	-	√	√	√	√	√	√	√	√	√	√	-	-	-	-	-
In2[] (array)	Array with the same data type as the elements of In1[]																			
Size	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-
AryOut[] (array)	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

■ Function

This instruction compares (greater than) the comparison array (In1[0] to In1[Size-1]) with the comparison variable In2. The comparison results are stored in the elements with the corresponding element numbers (AryOut[0] to AryOut[Size − 1]).

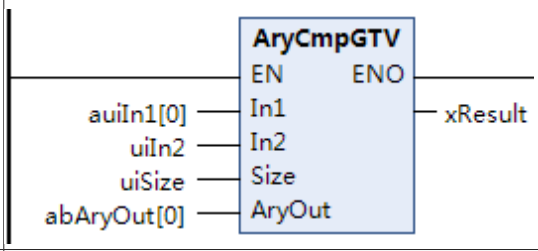
When In1[i] is greater than In2, AryOut[i] is TRUE. Otherwise, it is FALSE.

■ Program example

<table>
<tr><th></th><th>LD</th><th>ST</th></tr>
<tr><td>Defined variable</td><td colspan="2">

```
VAR
    auiIn1      : ARRAY [0..4] OF UINT := [11,22,33,44,55];
    uiIn2       : UINT := 33;
    abAryOut    : ARRAY [0..4] OF BOOL;
    uiSize      : UINT := 5;
    xResult     : BOOL;
END_VAR
```

</td></tr>
<tr><td>Program</td><td>



</td><td>

```
xResult := AryCmpGTV(In1:= auiIn1[0],
    In2:= uiIn2,
    Size:= uiSize,
    AryOut:= abAryOut[0]);
```

</td></tr>
<tr><td>Running result</td><td colspan="2">

auiIn1	ARRAY [0..4] OF UINT	
auiIn1[0]	UINT	11
auiIn1[1]	UINT	22
auiIn1[2]	UINT	33
auiIn1[3]	UINT	44
auiIn1[4]	UINT	55
uiIn2	UINT	33
abAryOut	ARRAY [0..4] OF BOOL	
abAryOut[0]	BOOL	FALSE
abAryOut[1]	BOOL	FALSE
abAryOut[2]	BOOL	FALSE
abAryOut[3]	BOOL	TRUE
abAryOut[4]	BOOL	TRUE
uiSize	UINT	5
xResult	BOOL	TRUE

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

■ Precautions

Use the same data type for In1[] and In2. If they are different, an error occurs during compilation.

If In1[] and In2 are real numbers that contain non-terminating decimal numbers, a rounding error may cause unexpected processing results.

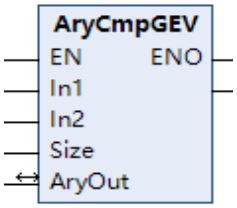
When the value of Size is 0, the return value is TRUE and AryOut[] does not change.

When the value of Size exceeds the In1[] or AryOut[] array, the return value is FALSE and AryOut[] does not change.

3.1. 21 AryCmpGEV

This instruction compares (greater than or equal to) the values of elements in arrays.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
AryCmpGEV	Array value comparison greater than or equal	FC		AryCmpGEV (In1 :=, In2 :=, Size := , AryOut :=);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
In1[] (array)	Comparison array	-	Depends on data types	-	Comparison array
In2[] (array)	Comparison variable	-	Depends on data types	-	Variable to compare
Size	Number of comparison elements	UINT	Depends on data types	1	Number of elements to compare

In-out variables

In-Out Variable	Name	Data Type	Value Range	Initial Value	Description
AryOut[] (array)	Comparison result array	-	Depends on data types	-	Comparison result array

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In1[] (array)	-	-	-	-	-	√	√	√	√	√	√	√	√	√	√	-	-	-	-	-
In2[] (array)	Array with the same data type as the elements of In1[]																			
Size	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-
AryOut[] (array)	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

■ Function

This instruction compares (greater than or equal to) the comparison array (In1[0] to In1[Size-1]) with the comparison variable In2. The comparison results are stored in the elements with the corresponding element numbers (AryOut[0] to AryOut[Size − 1]).

When In1[i] is greater than or equal to In2, AryOut[i] is TRUE. Otherwise, it is FALSE.

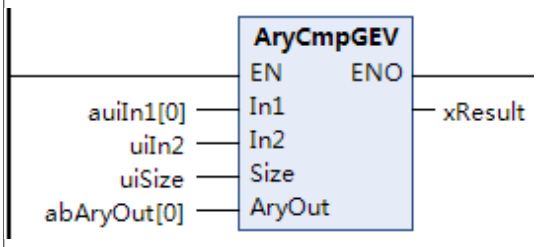
■ Program example

	LD	ST

Defined variable

```
VAR
    auiIn1      : ARRAY [0..4] OF UINT := [11,22,33,44,55];
    uiIn2       : UINT := 33;
    abAryOut    : ARRAY [0..4] OF BOOL;
    uiSize      : UINT := 5;
    xResult     : BOOL;
END_VAR
```

Program




```
xResult := AryCmpGEV(In1:= auiIn1[0],
    In2:= uiIn2,
    Size:= uiSize,
    AryOut:= abAryOut[0]);
```

Running result

auiIn1	ARRAY [0..4] OF UINT	
auiIn1[0]	UINT	11
auiIn1[1]	UINT	22
auiIn1[2]	UINT	33
auiIn1[3]	UINT	44
auiIn1[4]	UINT	55
uiIn2	UINT	33
abAryOut	ARRAY [0..4] OF BOOL	
abAryOut[0]	BOOL	FALSE
abAryOut[1]	BOOL	FALSE
abAryOut[2]	BOOL	TRUE
abAryOut[3]	BOOL	TRUE
abAryOut[4]	BOOL	TRUE
uiSize	UINT	5
xResult	BOOL	TRUE

■ Precautions

Use the same data type for In1[] and In2. If they are different, an error occurs during compilation.

If In1[] and In2 are real numbers that contain non-terminating decimal numbers, a rounding error may cause unexpected processing results.

When the value of Size is 0, the return value is TRUE and AryOut[] does not change.

When the value of Size exceeds the In1[] or AryOut[] array, the return value is FALSE and AryOut[] does not change.

3. 2 Selection Instructions

3.2. 1 Instruction List

Instruction Category	Name	FB/FC	Function

Selection instruction	AryMax	FC	Array maximum
	AryMin	FC	Array minimum
	ArySearch	FC	Array search
	ArySearch2	FC	Array search
	SEL	FC	Either one
	MUX	FC	One from multiple
	MAX	FC	Maximum value
	MIN	FC	Minimum value
	LIMIT	FC	Limit

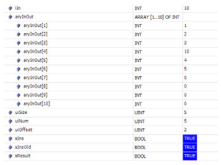
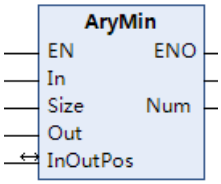
3.2. 2 AryMax/AryMin

These instructions search for the maximum value and minimum value in a one-dimensional array.

AryMax: Searches for the maximum value of an element in a one-dimensional array.

AryMin: Searches for the minimum value of an element in a one-dimensional array.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
AryMax	Maximum value retrieval for an array	FC		Out:=AryMax(In :=, Size :=, Out :=, InOutPos :=, Num =>);
AryMin	Minimum value retrieval for an array	FC		Out:=AryMin(In :=, Size :=, Out :=, InOutPos :=, Num =>);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
In[] (array)	Array to search	-	Depends on data types	(*)	Array to search
Size	Number of elements to search	UINT	Depends on data types	1	Number of elements in In[] to search
Out	Search result	-	Depends on data types	(*)	Search result

In-out variables

In-Out Variable	Name	Data Type	Value Range	Initial Val-ue	Description
InOutPos	Found element	UINT	Depends on data types	-	Array element number where the value is found

Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
Num	Number found	UINT	Depends on data types	-	Number found

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In[]	-	-	-	-	-	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√
Size	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-
InOutPos	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-
Out	Data with the same data type as elements in In[]																			
Num	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-

■ Function

These instructions search for Size elements starting from In[0] in the In[] array.

The value that is found is assigned to Out, the element number where it is found is assigned to InOutPos, and the number of times the value is found is assigned to Num. If Num is greater than 1, the value of InOutPos changes to the element number of the least-significant bit in the value that is found.

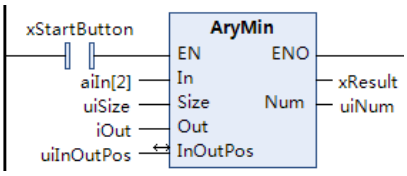
The relationship between values with data types that are not integers or real numbers is determined as given in the following table.

Data Type	Relationship
TIME	The numerically larger value is considered to be larger.
DATE, TOD, or DT	Later dates or TOD are considered to be larger.
STRING	First, the first character codes in all of the text strings are compared. If the first character codes are different, the size of each character code is the size of the corresponding text string. If the first character codes are the same, comparison continues in order to the other characters until a different character code is found. If the lengths of the text strings are different, NULL characters (16#00) are added to the shorter text string to complete the comparison.

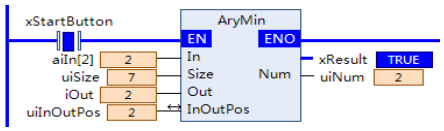
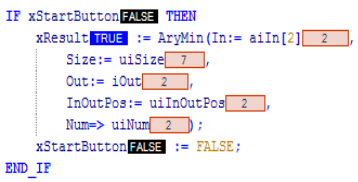
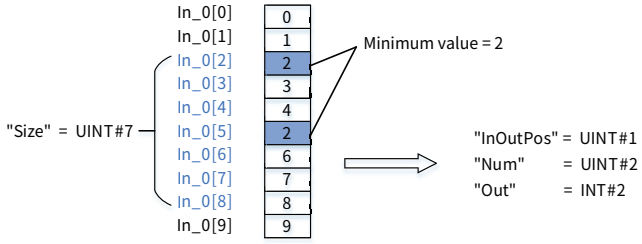
■ Program example

	LD	ST
Defined variable		
Program		

```
VAR
    xStartButton    : BOOL;
    aiIn            : ARRAY [0..9] OF INT := [0,1,2,3,4,2,6,7,8,9];
    uiSize          : UINT := 7;
    iOut            : INT;
    uiInOutPos      : UINT;
    uiNum           : UINT;
    xResult         : BOOL;
END_VAR
```




```
IF xStartButton THEN
    xResult := AryMin(In:= aiIn[2],
        Size:= uiSize,
        Out:= iOut,
        InOutPos:= uiInOutPos,
        Num=> uiNum);
    xStartButton := FALSE;
END_IF
```

Running result		
Work principle		

■ Precautions

· When the value of Size exceeds the In[] array, the return value is FALSE and Out does not change.

· When In[] is not a one-dimensional array, an error occurs during compilation.

· Ensure that the data type of Out is the same as that of the In[] array to search. Otherwise, an error occurs during compilation.

· The found element number InOutPos is in the In array to search.

· When In[] is a real number, an error may cause unexpected results.

· When the value of Size is 0, the value of Num changes to 0, the values of InOutPos and Out do not change, and the return value is TRUE.

· When the data type of In[] is not supported, an error occurs during compilation.

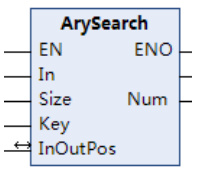
· When the data type of In[] and Out is STRING, the size of In[] should not exceed the size of Out. Otherwise, the return value is FALSE and Out does not change.

· When the data type of In[] and Key is STRING, the value range supports a maximum of 1985 characters. Extra characters are truncated.

3.2. 3 ArySearch

This instruction searches for a specified value in a one-dimensional array.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
ArySearch	Array search	FC		Out := ArySearch(In :=, Size :=, Key :=, InOutPos :=, Num =>);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
In[] (array)	Array to search	-	Depends on data types	(*)	Array to search

Input Variable	Name	Data Type	Value Range	Initial Value	Description
Size	Number of elements to search	UINT	Depends on data types	1	Number of elements to search
Key	Search keyword	-	Depends on data types	(*)	Search value

In-out variables

In-Out Variable	Name	Data Type	Value Range	Initial Value	Description
InOutPos	Found element number	UINT	Depends on data types	-	Element number found in the In[] array with the size specified by Size

Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
Num	Number found	UINT	Depends on data types	0	Number found

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In[]	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√
	Arrays of enumerations can also be specified.																			
Size	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-
Key	Data with the same data type as elements in In[]																			
InOutPos	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-
Num	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-

■ Function

This instruction searches Size elements starting from the first element In[0] in the In[] one-dimensional array for elements with the same value as the search keyword Key.

Values of the found element number InOutPos and the found number Num are listed in the following table.

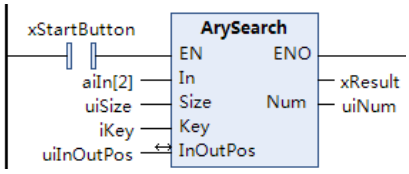
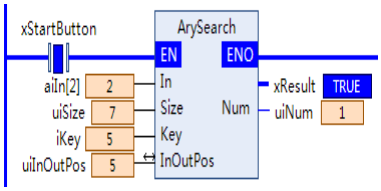
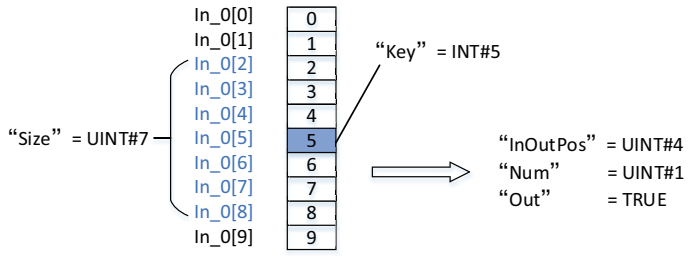
Element with Same Value As Key	InOutPos	Num
Existing	Element number of the least-significant bit for the element with the same value as Key	Number of elements with same value as Key
Not existing	Unchanged	0

The relationship between values with data types that are not integers or real numbers is determined as given in the following table.

Data Type	Relationship
TIME	The numerically larger value is considered to be larger.
DATE	Later dates are considered to be larger.

Data Type	Relationship
STRING	First, the first character codes in all of the text strings are compared. If the first character codes are different, the size of each character code is the size of the corresponding text string. If the first character codes are the same, comparison continues in order to the other characters until a different character code is found. If the lengths of the text strings are different, NULL characters (16#00) are added to the shorter text string to complete the comparison.

■ Program example

	LD	ST
Defined variable		
Program		
Running result		
Work principle		

```
VAR
    xStartButton    : BOOL;
    aiIn            : ARRAY [0..9] OF INT := [0,1,2,3,4,5,6,7,8,9];
    uiSize          : UINT := 7;
    iKey            : INT := 5;
    uiInOutPos      : UINT;
    uiNum           : UINT;
    xResult         : BOOL;
END_VAR
```

```
IF xStartButton THEN
    xResult := ArySearch(In:= aiIn[2],
        Size:= uiSize,
        Key:= iKey,
        InOutPos:= uiInOutPos,
        Num=> uiNum);
    xStartButton := FALSE;
END_IF
```



```
IF xStartButton FALSE THEN
    xResult TRUE := ArySearch(In:= aiIn[2] 2,
        Size:= uiSize 7,
        Key:= iKey 5,
        InOutPos:= uiInOutPos 5,
        Num=> uiNum 1);
    xStartButton FALSE := FALSE;
END_IF
```


■ Precautions

· When the value of Size exceeds the AryOut[] array, the return value is FALSE and the value of Num is initialized to 0.

· When In[] is not a one-dimensional array, an error occurs during compilation.

· Ensure that the data type of Key is the same as that of the In[] array to search. Otherwise, an error occurs during compilation.

· Ensure that the input parameter of Key is a variable.

· The found element number InOutPos is in the In array to search.

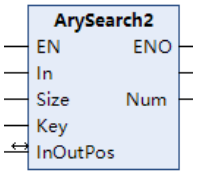
· When the value of Size is 0, the value of Num is initialized to 0, the value of InOutPos does not change, and the return value is TRUE.

· When the data type of In[] and Key is STRING, the value range supports up to 1985 characters. When the value exceeds the value range, the return value is FALSE and the value of Num is initialized to 0.

3.2. 4 ArySearch2

This instruction searches for a specified value in a one-dimensional array.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
ArySearch	Array search	FC		Out := ArySearch2(In :=, Size :=, Key :=, InOutPos :=, Num =>);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
In[] (array)	Array to search	-	Depends on data types	(*)	Array to search
Size	Number of elements to search	UINT	Depends on data types	1	Number of elements to search
Key	Search keyword	-	Depends on data types	(*)	Search value

In-out variables

In-Out Variable	Name	Data Type	Value Range	Initial Value	Description
InOutPos	Found element number	UINT	Depends on data types	-	Element number found in the In[] array with the size specified by Size

Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
Num	Number found	UINT	Depends on data types	0	Number found

	Bool-ean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In[ ]	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√
	Arrays of enumerations can also be specified.																			
Size	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-
Key	Data with the same data type as elements in In[]																			
InOutPos	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-
Num	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-

■ Function

This instruction searches Size elements starting from the first element In[0] in the In[] one-dimensional array for elements with the same value as the search keyword Key.

Values of the found element number InOutPos and the found number Num are listed in the following table.

Element with Same Value As Key	InOutPos	Num
Existing	Element number of the least-significant bit for the element with the same value as Key	Number of elements with same value as Key
Not existing	Unchanged	0

The relationship between values with data types that are not integers or real numbers is determined as given in the following table.

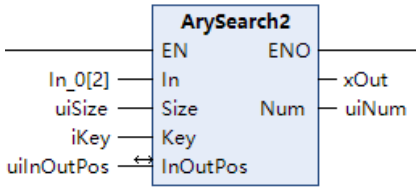
Data Type	Relationship
TIME	The numerically larger value is considered to be larger.
DATE	Later dates are considered to be larger.
STRING	First, the first character codes in all of the text strings are compared. If the first character codes are different, the size of each character code is the size of the corresponding text string. If the first character codes are the same, comparison continues in order to the other characters until a different character code is found. If the lengths of the text strings are different, NULL characters (16#00) are added to the shorter text string to complete the comparison.

■ Program example

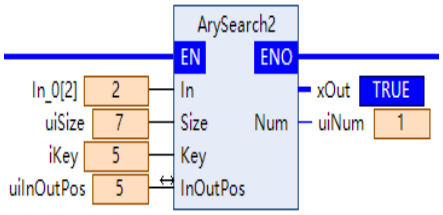
	LD	ST

<table>
<tr><td>Defined variable</td><td colspan="2">

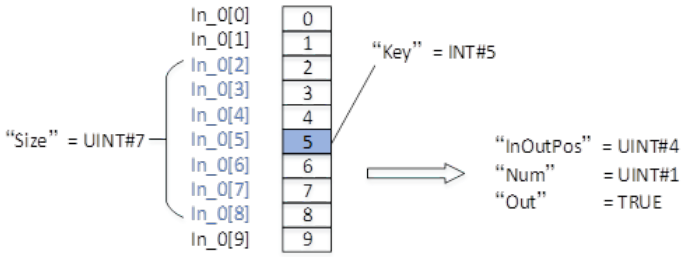
```
VAR
    In_0        :ARRAY[0..9] OF INT := [0,1,2,3,4,5,6,7,8,9];
    uiSize      :UINT := 0;
    iKey        :INT := 5;
    xOut        :BOOL;
    uiNum       :UINT;
    uiInOutPos  :UINT;
END_VAR
```

</td></tr>
<tr><td>Program</td><td> 
</td><td>

```
xOut := ArySearch2(In:= In_0[2],
                   Size:= uiSize,
                   Key:= iKey,
                   InOutPos:= uiInOutPos,
                   Num=> uiNum);
```

</td></tr>
<tr><td>Running result</td><td> 
</td><td>

```
xOut TRUE := ArySearch2(In:= In_0[2] 2 ,
                        Size:= uiSize 7 ,
                        Key:= iKey 5 ,
                        InOutPos:= uiInOutPos 5 ,
                        Num=> uiNum 1 );
```

</td></tr>
<tr><td>Work principle</td><td colspan="2"> 
</td></tr>
</table>

■ Precautions

· When the value of Size exceeds the AryOut[] array, the return value is FALSE and the value of Num is initialized to 0.

· When In[] is not a one-dimensional array, an error occurs during compilation.

· Ensure that the data type of Key is the same as that of the In[] array to search. Otherwise, an error occurs during compilation.

· Ensure that the input parameter of Key is a variable.

· The found element number InOutPos is in the In array to search.

· When the value of Size is 0, the value of Num is initialized to 0, the value of InOutPos does not change, and the return value is FALSE.

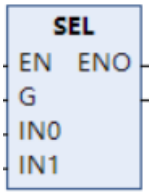
· When the data type of In[] and Key is STRING, the value range supports up to 1,985 characters. When the value exceeds the value range, the return value is FALSE and the value of Num is initialized to 0.

· Only when Key and In[] contain the same element, the return value is TRUE. Otherwise, it is FALSE.

3.2. 5 SEL

This instruction outputs IN0 when G is FALSE and IN1 when G is TRUE.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
SEL	Binary selection	FC		SEL

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
G	Input Signal	BOOL	[FALSE, TRUE]	FALSE	The output is IN0 when G is FALSE. The output is IN1 when G is TRUE.
IN0	Input signal	INT	–32767 to +32767	0	The output is IN0 when G is FALSE.
IN1	Input signal	INT	–32767 to +32767	0	The output is IN0 when G is FALSE.

Output variables

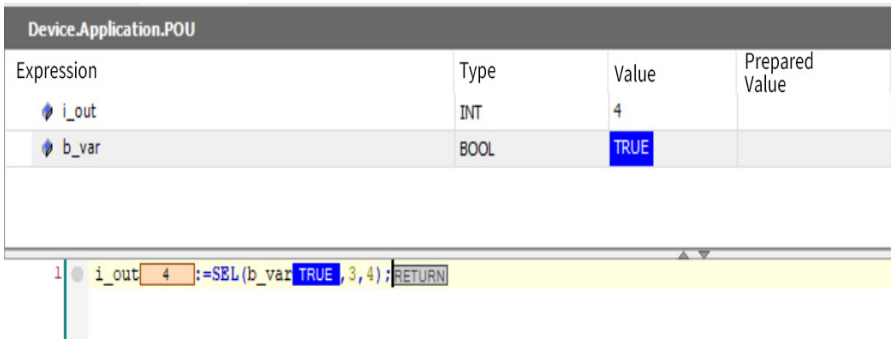
Output Variable	Name	Data Type	Value Range	Initial Value	Description
OUT	Output signal	INT	–32767 to +32767	0	The output is IN0 or IN1 based on the G status.

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
G	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
IN0	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-
IN1	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-
OUT	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-
Num	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-

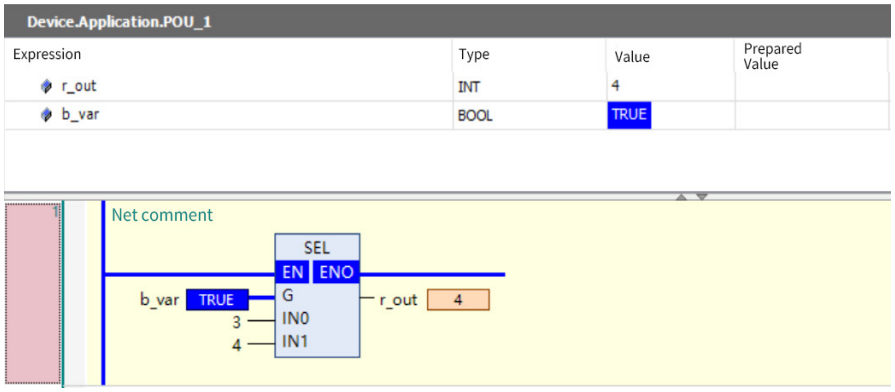
■ Program example

ST:

The values of input variables IN0 and IN1 are 3 and 4 respectively. When the input variable G is FALSE, IN0 is output. When the input variable G is TRUE, IN1 is output.



LD



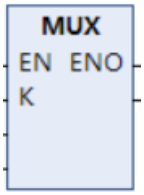
■ Precautions

When G is TRUE, CODESYS does not calculate the expression before IN0. When G is FALSE, CODESYS does not calculate the expression before IN1.

3.2. 6 MUX

This instruction selects the K value from a group of values. If the K value is greater than the number of input data entries, the result is the value of the last input data entry. If K is 0, the result is the value of the first input data entry.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
MUX	Multiplexer	FC		MUX

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
K	Selector	Any type	-	-	Digit bit to select
IN0	Selections	Any type	-	-	Data source to select
IN1	Selections	Any type	-	-	Data source to compare

Output variables

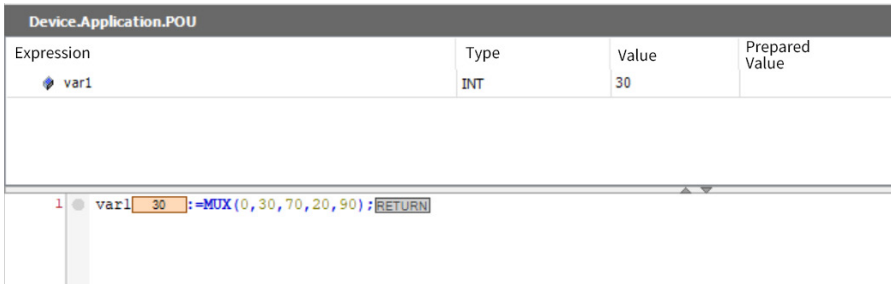
Output Variable	Name	Data Type	Value Range	Initial Value	Description
OUT	Comparison result	-	-	0	Output comparison result

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
K	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√
IN0	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√
IN1	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√
OUT	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√

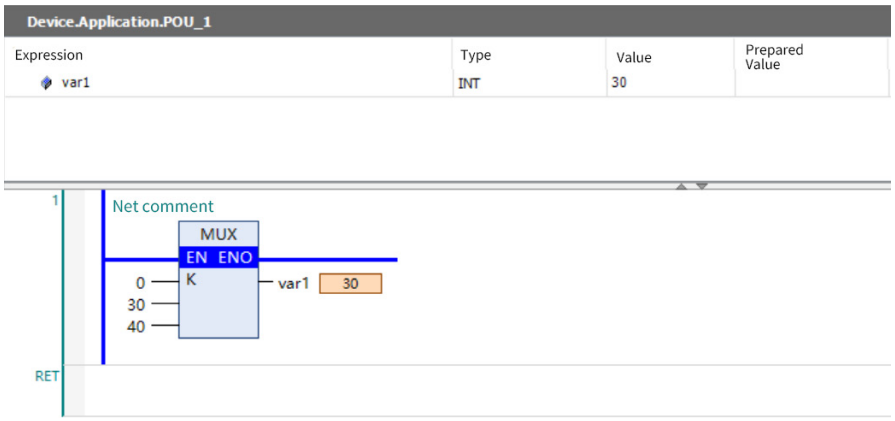
■ Program example

A random set of data (30, 70, 20, 90) is provided. When K is 0, the output is 30.

ST



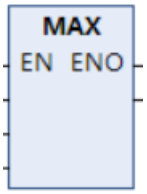
LD



3.2. 7 MAX

This instruction finds the maximum value from multiple input values, and outputs the maximum value from the right side.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
MAX	Maximum	FC		MAX

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
IN0	Selections	-	-	0	Number to compare
IN1	Selections	-	-	0	Number to compare

Output variables

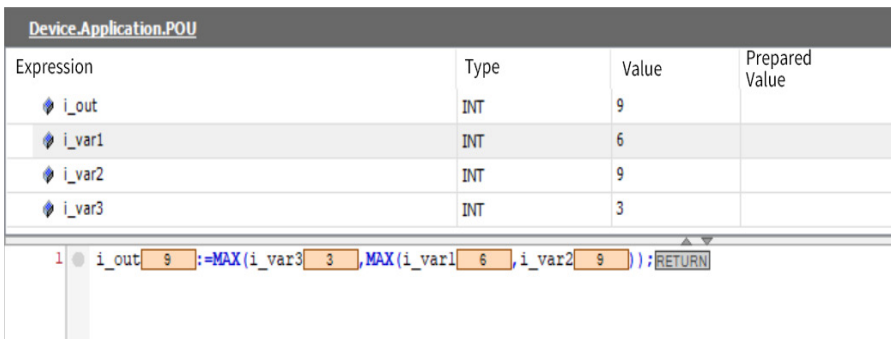
Output Variable	Name	Data Type	Value Range	Initial Value	Description
OUT	Comparison result	-	-	0	Output of the larger one from two selections

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
IN0	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√
IN1	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√
OUT	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√

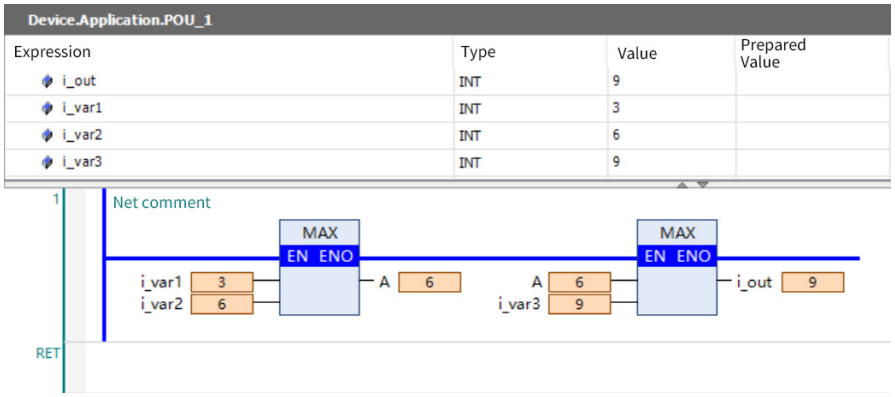
■ Program example

This instruction finds the maximum value in two given values.

ST



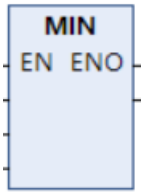
LD



3.2. 8 MIN

This instruction finds the minimum value from multiple input values, and outputs the minimum value from the right side.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
MIN	Minimum	FC		MIN

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
IN0	Selections	-	-	0	Number to compare
IN1	Selections	-	-	0	Number to compare

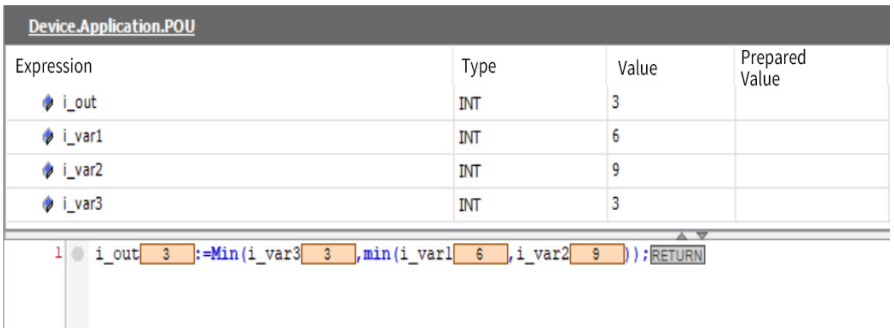
Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
OUT	Comparison result	-	-	0	Output of the smaller one from two selections

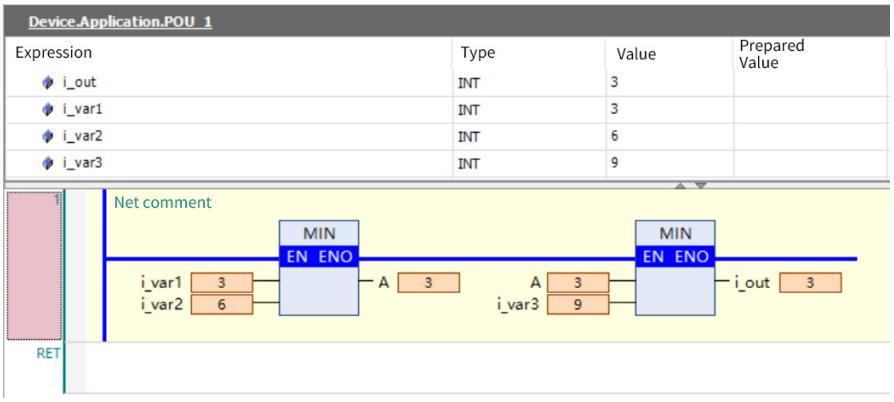
	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
IN0	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√
IN1	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√
OUT	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√

■ Program example

ST



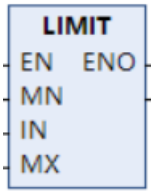
LD



3.2. 9 LIMIT

This instruction limits the maximum value (MX) and minimum value (MN) for the result. If the value of IN is greater than MX, the output calculation result is MX. If the value of IN is less than MN, the output calculation result is MN. When the value of IN is within the range of MN and MX, the output value is the input value of IN.

■ Instruction description

Instruction	Name	FB/FC	LD Expression	ST Expression
LIMIT	Limiter	FC		LIMIT

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
MN	Input signal	All types	-	-	Minimum value
MX	Input signal	All types	-	-	Maximum value
IN	Input signal	All types	-	-	Comparison value

Output variables

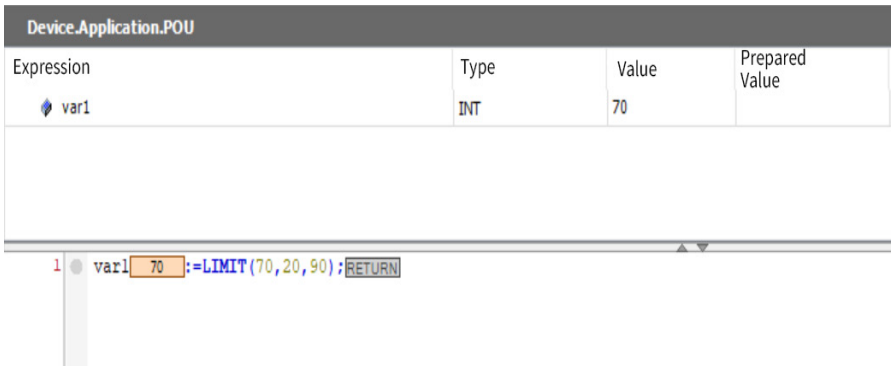
Output Variable	Name	Data Type	Value Range	Initial Value	Description
OUT	Output result	All types	-	-	For details, see the program example.

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
MIN	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√
MX	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√
IN	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√
OUT	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√

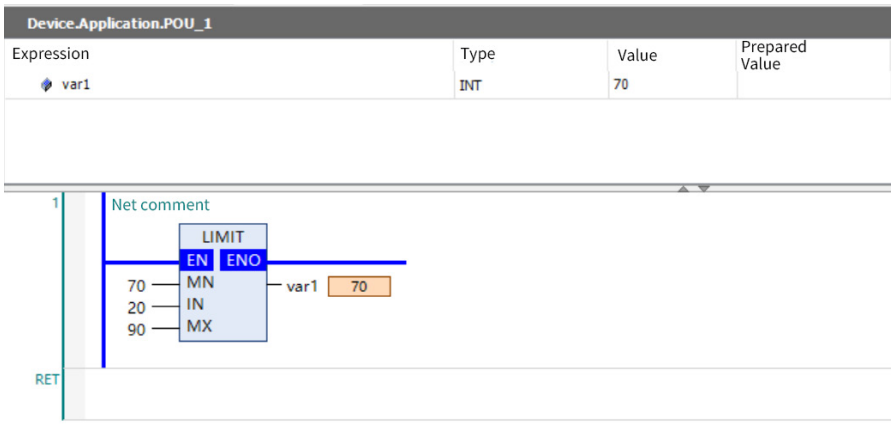
■ Program example

This instruction sets MN to 70, MX to 90, and IN to 20. Then the output is 70.

ST



LD



■ Precautions for Use

Use the same data type for MX and MN.

3. 3 Counter Instructions

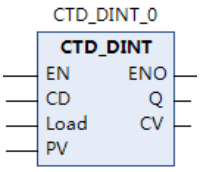
3.3. 1 Instruction List

Instruction Category	Name	FB/FC	Function
Counter instructions	CTD_**	FB	Down-counter group
	CTU_**	FB	Up-counter group
	CTUD_**	FB	Up-down counter group
	CTD	FB	Down-counter
	CTU	FB	Up-counter
	CTUD	FB	Up-down counter

3.3. 2 CTD_**

This instruction decrements the count value when the counter input signal is received. The preset value and count value must be one of the following data types: DINT, LINT, UDINT, or ULINT.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
CTD_**	Down-counter group	FB	 DINT can be replaced by LINT, UDINT, or ULINT.	CTD_DINT_0(CD:=, Load :=, PV :=, Q =>, CV =>); DINT can be replaced by LINT, UDINT, or ULINT.

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
CD	Counter input	BOOL	[FALSE, TRUE]	FALSE	Counter input
Load	Load signal	BOOL	[FALSE, TRUE]	FALSE	TRUE: Set CV to PV
PV	Preset value	-	Depends on data types	0	Preset value of the counter

Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
Q	Counter output	BOOL	[FALSE, TRUE]	FALSE	TRUE: Counter output ON FALSE: Counter output OFF
CV	Count value	-	Depends on data types	-	Current value of the counter

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	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	Value with the same data type as PV																			

■ Function

This instruction creates a down counter. The preset value and count value must be one of the following data types: DINT, LINT, UDINT, or ULINT.

The instruction name is determined by the data types of PV and CV. For example, when their data types are DINT, the instruction name is CTD_DINT.

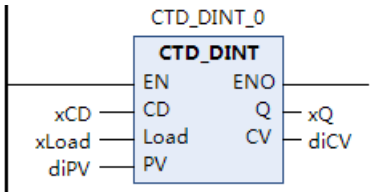
When the load signal Load changes to TRUE, the count value CV is set to the preset value PV and the counter output Q changes to FALSE.

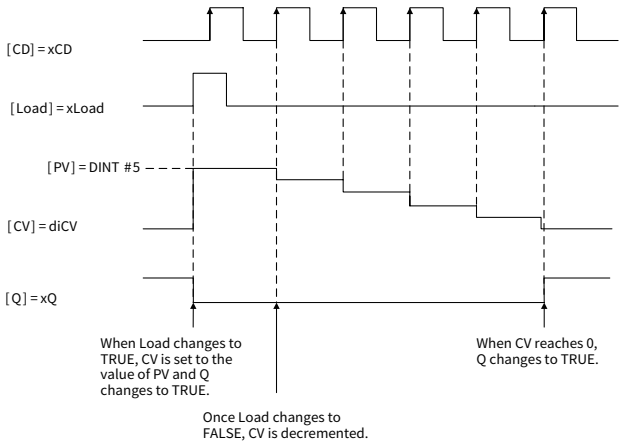
When the counter input signal CD changes to TRUE, CV is decremented. When the value of CV is less than 0, the value of Q changes to TRUE.

When the value of CV is less than 0, CV is not decremented even if CD changes to TRUE.

■ Program example

The following figure shows a programming example and timing chart for a PV of DINT#5 when the instruction is CTD_DINT.

	LD	ST
Defined variable		VAR CTD_DINT_0: CTD_DINT; xCD : BOOL; xLoad : BOOL; diPV : DINT := 5; diCV : DINT; xQ : BOOL; END_VAR
Program		CTD_DINT_0(CD := xCD, Load := xLoad, PV := diPV, Q => xQ, CV => diCV);



■ Precautions

· Change Load to TRUE and then back to FALSE to restart a counter that has completed counting down.

· Set the same data type for PV and CV.

· When PV is set to a negative value and the value of Load changes to TRUE, CV is set to the value of PV. The value of CV is less than 0, so the value of Q changes to TRUE immediately. After that, the value of CV is not decremented even if the value of CD changes.

· If the value of PV exceeds the value range, it is implicitly converted into an incorrect value and no function error occurs.

· When the value of CD is FALSE and the power supply is interrupted or the operating mode is changed to program control, the value of CV is decremented once if the value of CD changes to TRUE upon instruction re-execution.

· If this instruction is used in a ladder diagram, the value of Q changes to FALSE if an error occurs in the previous instruction on the rung.

3.3. 3 CTU_**

This instruction increments the count value when the counter input signal is received. The preset value and count value must be one of the following data types: DINT, LINT, UDINT, or ULINT.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
CTU_**	Up-counter group	FB	CTU_DINT_0 CTU_DINT EN ENO CU Q Reset CV PV DINT can be replaced by LINT, UDINT, or ULINT.	CTU_DINT_0(CU :=, Reset :=, PV :=, Q =>, CV =>); DINT can be replaced by LINT, UDINT, or ULINT. DINT_0(CD,Load,PV,Q,CV); DINT can be replaced by LINT, UDINT, or ULINT.

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
CD	Counter input	BOOL	[FALSE, TRUE]	FALSE	Counter input
Reset	Reset signal	BOOL	[FALSE, TRUE]	FALSE	TRUE: Reset CV to 0
PV	Preset value	-	Depends on data types	0	Preset value of the counter

Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
Q	Counter output	BOOL	[FALSE, TRUE]	FALSE	TRUE: Counter output ON FALSE: Counter output OFF
CV	Count value	-	Depends on data types	-	Current value of the counter

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	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	Value with the same data type as PV																			

■ Function

This instruction creates an up counter. The preset value and count value must be one of the following data types: DINT, LINT, UDINT, or ULINT.

The instruction name is determined by the data types of PV and CV. For example, when their data types are DINT, the instruction name is CTU_DINT.

When the reset signal Reset changes to TRUE, the count value CV changes to 0 and the counter output Q changes to FALSE.

When the counter input signal CU changes to TRUE, CV is incremented. When the value of CV is greater than the preset value PV, the value of Q changes to TRUE.

When the value of CV is greater than the value of PV, the value of CV does not change even if the input value of CU is higher.

CU is ignored while Reset is TRUE. CV is not incremented.

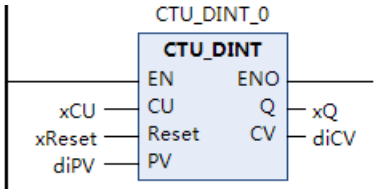
■ Program example

The following figure shows a programming example and timing chart for a PV of DINT#5 when the instruction is CTU_DINT.

<table>
<tr><th></th><th>LD</th><th>ST</th></tr>
<tr><td>Defined variable</td><td colspan="2">

```
VAR
    CTU_DINT_0: CTU_DINT;
    xCU     : BOOL;
    xReset  : BOOL;
    diPV    : DINT := 5;
    diCV    : DINT;
    xQ      : BOOL;
END_VAR
```

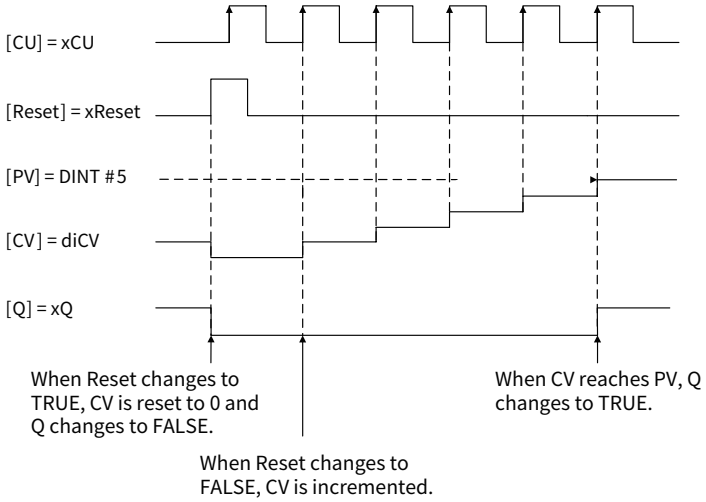
</td></tr>
<tr><td>Program</td><td>



</td><td>

```
CTU_DINT_0(
    CU := xCU,
    Reset := xReset,
    PV := diPV,
    Q => xQ,
    CV => diCV
);
```

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



■ Precautions

· Change Reset to TRUE and then back to FALSE to restart a counter that has completed counting down.

· When PV is set to a negative value and the value of Reset changes to TRUE, the value of CV changes to 0. The value of CV is greater than the value of PV, so the value of Q changes to TRUE immediately. After that, the value of CV is not incremented even if the value of CU changes.

· If the value of PV exceeds the value range, it is implicitly converted into an incorrect value and no function error occurs.

· Set the same data type for PV and CV.

· The following operation is performed if the value of PV changes while the value of Reset is FALSE.

Value of PV	Meaning
Greater than the current value of CV	The count operation is continued.
Less than or equal to the current value of CV	The count operation is ended. The value of Q changes to TRUE. The current value of CV is retained.

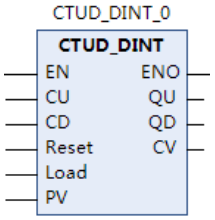
When the value of CU is FALSE and the power supply is interrupted or the operating mode is changed to program control, the value of CV is incremented once if the value of CU changes to TRUE upon instruction re-execution.

If this instruction is used in a ladder diagram, the value of Q changes to FALSE if an error occurs in the previous instruction on the rung.

3.3. 4 CTUD_**

This instruction creates an up-down counter that operates according to an up-counter input and a down-counter input. The preset value and count value must be one of the following data types: DINT, LINT, UDINT, or ULINT.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
CTUD_**	Up-down counter group	FB	 DINT can be replaced by LINT, UDINT, or ULINT.	CTUD_DINT_0(CU :=, CD :=, Reset :=, Load :=, PV :=, QU =>, QD =>, CV =>); DINT can be replaced by LINT, UDINT, or ULINT.

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
CU	Up-counter input	BOOL	[FALSE, TRUE]	FALSE	Up-counter input
CD	Down-counter input	BOOL	[FALSE, TRUE]	FALSE	Down-counter input
Reset	Reset signal	BOOL	[FALSE, TRUE]	FALSE	TRUE: Reset CV to 0
Load	Load signal	BOOL	[FALSE, TRUE]	FALSE	TRUE: Set CV to PV
PV	Preset value	-	Depends on data types	0	Preset value of the counter

Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
QU	Up-counter output	BOOL	[FALSE, TRUE]	FALSE	TRUE: Up-counter output ON FALSE: Up-counter output OFF
QD	Down-counter output	BOOL	[FALSE, TRUE]	FALSE	TRUE: Down-counter output ON FALSE: Down-counter output OFF
CV	Count value	-	Depends on data types	-	Current value of the counter

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
CU	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Reset	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

	Bool-ean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
PV	-	-	-	-	-	-	-	√	√	-	-	√	√	-	-	-	-	-	-	-
Q	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
CV	Value with the same data type as PV																			

■ Function

This instruction creates an up-down counter that operates according to up-counter and down-counter input signals.

It has the functions of both an up counter and a down counter.

The preset value and count value must be one of the following data types: DINT, LINT, UDINT, or ULINT.

The instruction name is determined by the data types of PV and CV. For example, when their data types are LINT, the instruction name is CTUD_LINT.

Operation as an up counter

When the reset signal Reset changes to TRUE, the count value CV changes to 0 and the up-counter output QU changes to FALSE. When the up-counter input signal CU changes to TRUE, CV is incremented. When the value of CV is greater than the preset value PV, the value of QU changes to TRUE.

When the value of CV is greater than the value of PV, the value of CV does not change even if the input value of CU is higher.

Operation as a down counter

When the load signal Load changes to TRUE, the count value CV is set to the preset value PV and the down-counter output QD changes to FALSE.

When the down-counter input signal CD changes to TRUE, CV is decremented. When the value of CV is less than 0, the value of QD changes to TRUE.

When the value of CV is less than 0, the value of CV does not change even if the input value of CD is higher.

Common operation for up and down counters

CU and CD are ignored while Load or Reset is TRUE. CV is not incremented or decremented.

When both CU and CD change to TRUE at the same time, CV does not change.

If Reset and Load are both TRUE, Reset has priority and the value of CV changes to 0.

If Reset changes to TRUE, the value of CV changes to 0, and so the value of QD changes to TRUE.

If Load changes to TRUE, the value of CV changes to PV, and so the value of QU changes to TRUE.

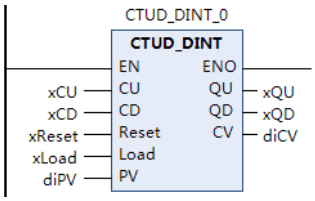
Reset	Load	CV	QU	QD	Action
FALSE	FALSE	Less than 0	FALSE	TRUE	Only the up counter operation is performed. CV is incremented when CU changes to TRUE. It is not decremented when CD changes to TRUE.
		Between 0 and PV	FALSE	FALSE	Both the up and down counter operations are performed. CV is incremented when CU changes to TRUE and decremented when CD changes to TRUE.
		Greater than PV	TRUE	FALSE	Only the down counter operation is performed. CV is decremented when CD changes to TRUE. It is not incremented when CU changes to TRUE.
TRUE	FALSE	0	FALSE	TRUE	The up counter is reset. The value of CV is set to 0.
FALSE	TRUE	PV	TRUE	FALSE	The down counter is reset. The value of CV is set to PV.
TRUE	TRUE	0	FALSE	TRUE	The up counter is reset. Reset takes priority over Load. The value of CV is set to 0.

■ Program example

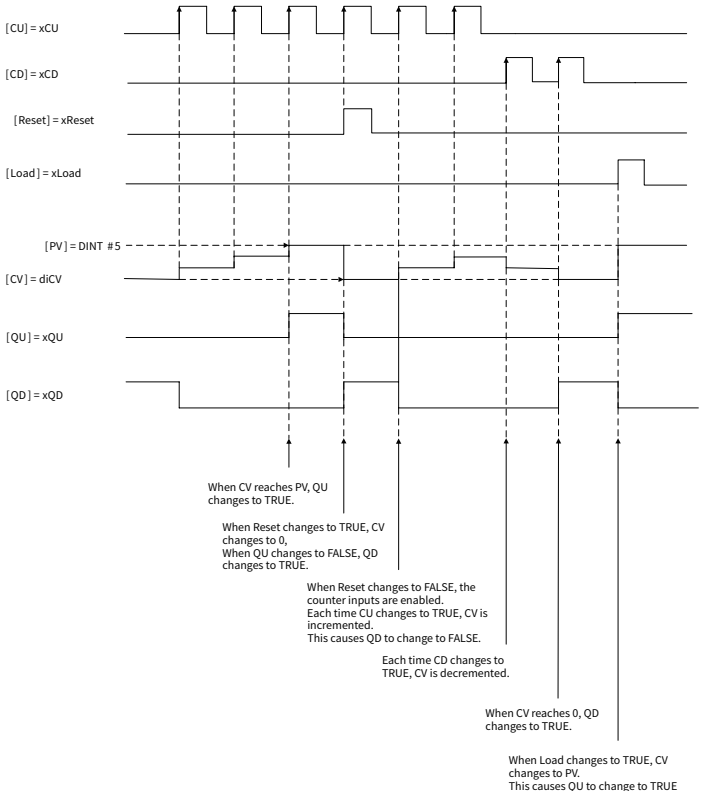
The following figure shows a programming example and timing chart for a PV of DINT#3 when the instruction is CTUD_DINT.

	LD	ST
Defined variable		
Program		

```
VAR
    CTUD_DINT_0: CTUD_DINT;
    xCU    : BOOL;
    xCD    : BOOL;
    xReset : BOOL;
    xLoad  : BOOL;
    diPV   : DINT := 3;
    diCV   : DINT;
    xQU    : BOOL;
    xQD    : BOOL;
END_VAR
```




```
xResult := ADD_TOD_TIME(In1 := todIn1,
        In2 := tIn2,
        Out => todOut
        );
```



■ Precautions

· If you change Reset to TRUE to reset the up counter, the value of QU changes to FALSE and the value of QD changes to TRUE.

· If you change Load to TRUE to reset the down counter, the value of QD changes to FALSE and the value of QU changes to TRUE.

· When PV is set to a negative value and the value of Load changes to TRUE, CV is set to the value of PV. The value of CV is less than 0, so the value of QD changes to TRUE immediately. After that, the value of CV is not decremented even if the value of CD changes. The value of Reset changes to TRUE, and the value of CV changes to 0. The value of CV is greater than the value of PV, so the value of QU changes to TRUE immediately. After that, the value of CV is not incremented even if the value of CU changes.

· If the value of PV exceeds the value range, it is implicitly converted into an incorrect value and no function error occurs.

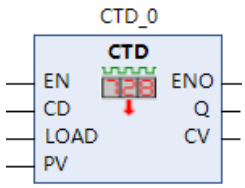
· Set the same data type for PV and CV.

· When the value of CU or CD is FALSE and the power supply is interrupted or the operating mode is changed to program control, the value of CV is incremented or decremented once if the value of CU or CD changes to TRUE upon instruction re-execution.

3.3. 5 CTD

This instruction constantly decrements the count value when the counter input signal is received. The data type of the preset value and count value is WORD.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
CTD	Down-counter	FB		CTD_0(CD := , Load := , PV := , Q => , CV =>);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
CU	Counter input	BOOL	[FALSE, TRUE]	FALSE	Counter input
Load	Load signal	BOOL	[FALSE, TRUE]	FALSE	TRUE: Set CV to PV
PV	Preset value	WORD	0 to 65535	0	Preset value of the counter

Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
Q	Counter output	BOOL	[FALSE, TRUE]	FALSE	TRUE: Counter output ON FALSE: Counter output OFF
CV	Count value	WORD	0 to 65535	0	Current value of the counter

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
CU	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Load	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
PV	-	-	-	-	-	-	-	√	√	-	-	√	√	-	-	-	-	-	-	-
Q	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
CV	Value with the same data type as PV																			

■ Function

This instruction creates a down counter. The data type of the preset value and count value is WORD.

When the load signal Load changes to TRUE, the count value CV is set to the preset value PV and the counter output Q changes to FALSE.

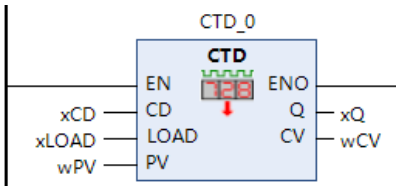
When counter input signal CD changes to TRUE, CV is constantly decremented. When the value of CV is less than 0, the value of Q changes to TRUE.

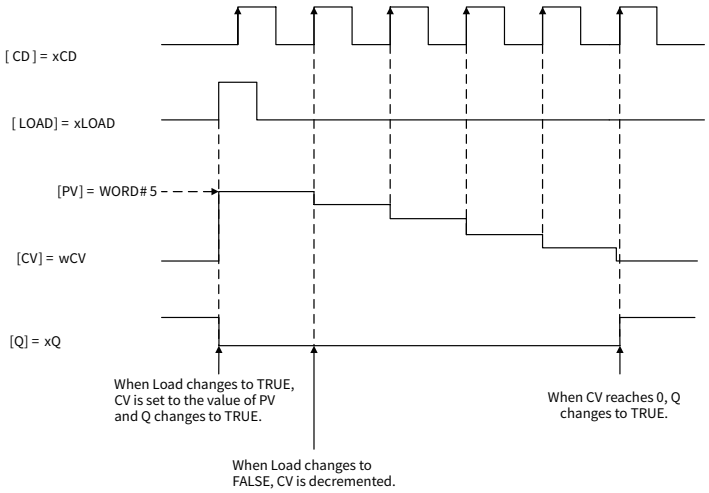
When the value of CV is less than 0, CV is not decremented even if CD changes to TRUE.

CD is ignored while Load is TRUE. CV is constantly decremented.

■ Program example

The following figure shows a programming example and timing chart for a PV of WORD#5.

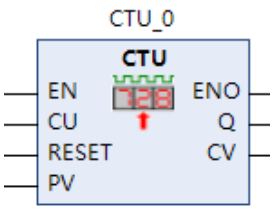
	LD	ST
Defined variable	VAR CTD_0 : CTD; xCD : BOOL; xLOAD : BOOL; wPV : WORD :=5; xQ : BOOL; wCV : WORD; END_VAR	
Program		CTD_0(CD := xCD, LOAD := xLOAD, PV := wPV, Q => xQ, CV => wCV);



3.3. 6 CTU

This instruction constantly increments the count value when the counter input signal is received. The data type of the preset value and count value is WORD.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
CTU	Up-counter	FB		CTU_0(CU := , Reset := , PV := , Q => , CV =>);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
CU	Counter input	BOOL	[FALSE, TRUE]	FALSE	Counter input
Reset	Reset signal	BOOL	[FALSE, TRUE]	FALSE	TRUE: Reset CV to 0
PV	Preset value	WORD	0 to 65535	0	Preset value of the counter

Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
Q	Counter output	BOOL	[FALSE, TRUE]	FALSE	TRUE: Counter output ON FALSE: Counter output OFF
CV	Count value	WORD	0 to 65535	0	Current value of the counter

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
CU	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Load	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
PV	-	-	-	-	-	-	-	√	√	-	-	√	√	-	-	-	-	-	-	-
Q	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
CV	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

■ Function

This instruction creates an up counter. The data type of the preset value and count value is WORD.

When the reset signal Reset changes to TRUE, the count value CV changes to 0 and the counter output Q changes to FALSE.

When counter input signal CU changes to TRUE, CV is constantly incremented. When the value of CV is greater than the preset value PV, the value of Q changes to TRUE.

When the value of CV is greater than the value of PV, the value of CV does not change even if the input value of CU is higher.

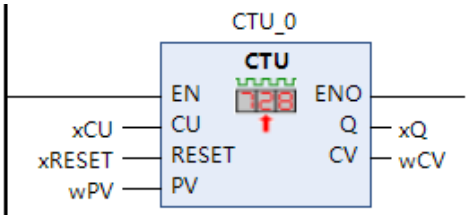
CU is ignored while Reset is TRUE. CV is not incremented.

■ Program example

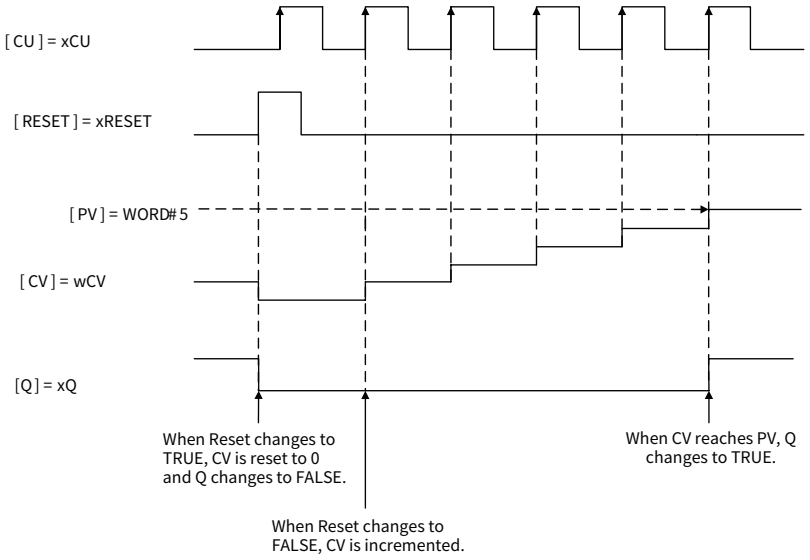
The following figure shows a programming example and timing chart for a PV of WORD#5.

	LD	ST
Defined variable		
Program		

```
VAR
    CTU_0  : CTU;
    xCU    : BOOL;
    xRESET : BOOL;
    wPV    : WORD := 5;
    xQ     : BOOL;
    wCV    : WORD;
END_VAR
```



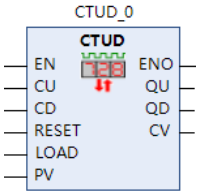

```
CTU_0(
    CU := xCU,
    RESET := xRESET,
    PV := wPV,
    Q => xQ,
    CV => wCV
);
```



3.3. 7 CTUD

This instruction creates an up-down counter that operates according to an up-counter input and a down-counter input. The data type of the preset value and count value is WORD.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
CTUD	Up-down counter	FB		CTUD_DINT_0(CU := , CD := , Reset := , Load := , PV := , QU => , QD => , CV =>);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
CU	Up-counter input	BOOL	[FALSE, TRUE]	FALSE	Up-counter input
CD	Down-counter input	BOOL	[FALSE, TRUE]	FALSE	Down-counter input
Reset	Reset signal	BOOL	[FALSE, TRUE]	FALSE	TRUE: Reset CV to 0
Load	Load signal	BOOL	[FALSE, TRUE]	FALSE	TRUE: Set CV to PV
PV	Preset value	WORD	0 to 65535	0	Preset value of the counter

Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
QU	Up-counter output	BOOL	[FALSE, TRUE]	FALSE	TRUE: Up-counter output ON FALSE: Up-counter output OFF
QD	Down-counter output	BOOL	[FALSE, TRUE]	FALSE	TRUE: Down-counter output ON FALSE: Down-counter output OFF
CV	Count value	WORD	0 to 65535	0	Current value of the counter

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	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	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
QU	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
CV	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

■ Function

This instruction creates an up-down counter that operates according to up-counter and down-counter input signals.

It has the functions of both an up counter and a down counter.

The data type of the preset value and count value is WORD.

Operation as an up counter

When the reset signal Reset changes to TRUE, the count value CV changes to 0 and the up-counter output QU changes to FALSE. When the up-counter input signal CU changes to TRUE, CV is incremented. When the value of CV is greater than the preset value PV, the value of QU changes to TRUE.

When the value of CV is greater than the value of PV, the value of CV does not change even if the input value of CU is higher.

Operation as a down counter

When the load signal Load changes to TRUE, the count value CV is set to the preset value PV and the down-counter output QD changes to FALSE.

When the down-counter input signal CD changes to TRUE, CV is decremented. When the value of CV is less than 0, the value of QD changes to TRUE.

When the value of CV is less than 0, the value of CV does not change even if the input value of CD is higher.

Common operation for up and down counters

CU and CD are ignored while Load or Reset is TRUE. CV is not incremented or decremented.

When both CU and CD change to TRUE at the same time, CV does not change.

If Reset and Load are both TRUE, Reset has priority and the value of CV changes to 0.

If Reset changes to TRUE, the value of CV changes to 0, and so the value of QD changes to TRUE.

If Load changes to TRUE, the value of CV changes to PV, and so the value of QU changes to TRUE.

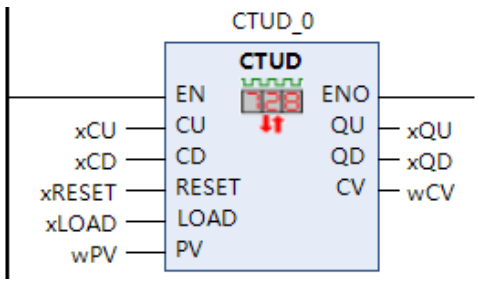
Reset	Load	CV	QU	QD	Action
FALSE	FALSE	Less than 0	FALSE	TRUE	Only the up counter operation is performed. CV is incremented when CU changes to TRUE. It is not decremented when CD changes to TRUE.
		Between 0 and PV	FALSE	FALSE	Both the up and down counter operations are performed. CV is incremented when CU changes to TRUE and decremented when CD changes to TRUE.
		Greater than PV	TRUE	FALSE	Only the down counter operation is performed. CV is decremented when CD changes to TRUE. It is not incremented when CU changes to TRUE.
TRUE	FALSE	0	FALSE	TRUE	The up counter is reset. The value of CV is set to 0.
FALSE	TRUE	PV	TRUE	FALSE	The down counter is reset. The value of CV is set to PV.
TRUE	TRUE	0	FALSE	TRUE	The up counter is reset. Reset takes priority over Load. The value of CV is set to 0.

■ Program example

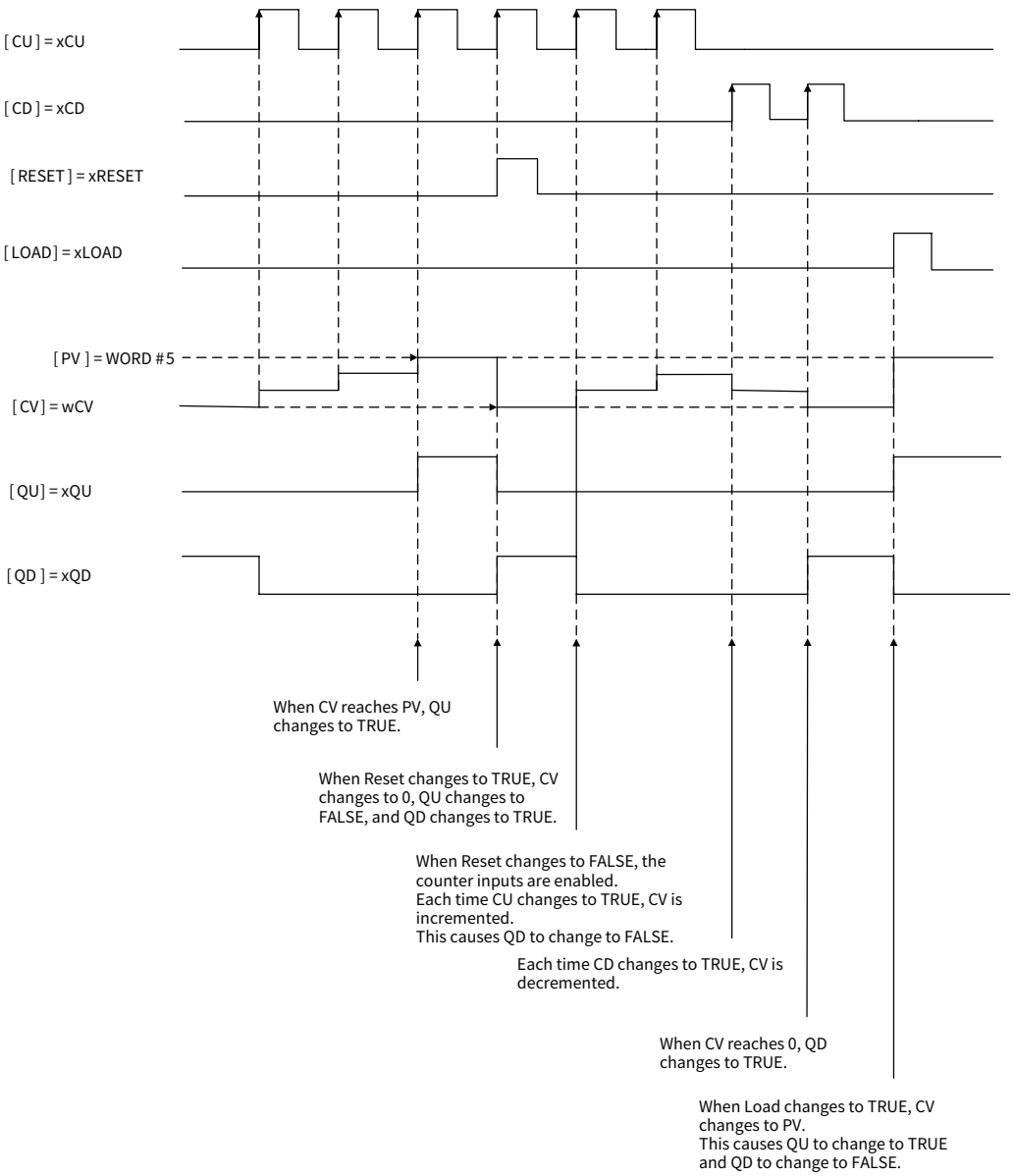
The following figure shows a programming example and timing chart for a PV of DINT#3.

	LD	ST
Defined variable		
Program		

```
VAR
    CTUD_0: CTUD;
    xCU     : BOOL;
    xCD     : BOOL;
    xRESET  : BOOL;
    xLOAD   : BOOL;
    wPV     : WORD := 3;
    xQU     : BOOL;
    xQD     : BOOL;
    wCV     : WORD;
END_VAR
```




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



3. 4 Timer Instructions

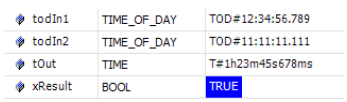
3.4. 1 Instruction List

Instruction Category	Name	FB/FC	Function
Timer instructions	AccumulationTimer	FB	Accumulation timer
	Timer	FC	Hundred-ms timer
	TP	FB	Timer pulse
	TON	FB	On-delay timer
	TOF	FB	Off-delay timer
	RTC	FB	Real-time clock

3.4. 2 AccumulationTimer

This instruction works as an accumulation timer.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
AccumulationTimer	Accumulation timer	FB		AccumulationTimer_0(In:=, PT:=, Reset:=, Q=>, ET=>);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
In	Timer input	BOOL	[FALSE, TRUE]	FALSE	TRUE: The timer operates FALSE: The timer stops
PT	Preset time	TIME	(*)	0	Maximum value for timing
Reset	Reset	BOOL	[FALSE, TRUE]	FALSE	TRUE: The timer is reset FALSE: The timer is not reset

Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
Q	Timer output	BOOL	[FALSE, TRUE]	-	TRUE: ET reaches PT FALSE: ET does not reach PT
ET	Total time	TIME	(*)	-	Total time

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
PT	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-
Reset	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Q	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
ET	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-

■ Function

This instruction totals the time that the timer input is TRUE. The time unit is ms and the time is accurate to 100 nanoseconds.

If Reset is FALSE, the timer starts when In changes to TRUE. Total time ET is incremented as time elapses.

The timer stops when In changes to FALSE. ET is held.

When In changes to TRUE again, the timer starts again. ET is incremented from the value that was previously held.

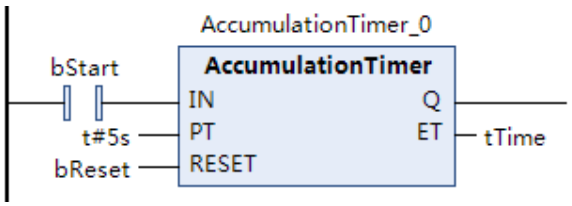
When ET reaches set time PT, timer output Q changes to TRUE. ET is not incremented after that.

The timer is reset when Reset changes to TRUE. The value of ET changes to 0 and Q changes to FALSE.

■ Timing diagram

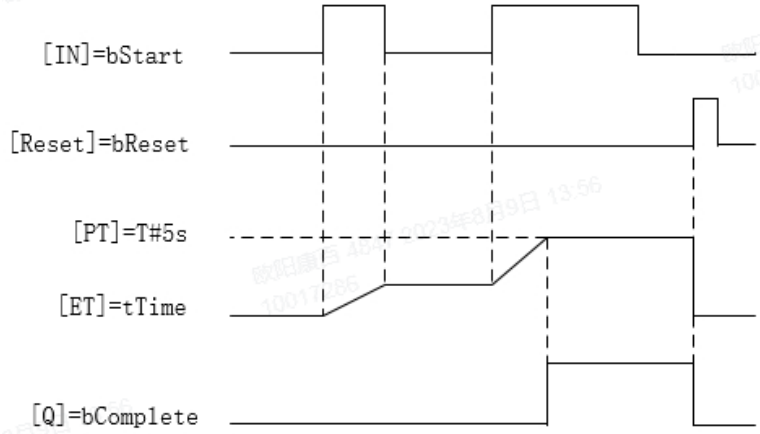
The following figure shows a programming example and timing chart for a PT of T#5s. Variable bComplete changes to TRUE when variable bStart is TRUE for a total of 5s.

LD



ST

```
AccumulationTimer_0(
    IN:=bStart,
    PT:=T#5S,
    RESET:=bReset,
    Q=>bComplete,
    ET=>tTime,
```



■ Precautions

· ET and Q are updated when this instruction is executed. Therefore, the condition for Q to change to TRUE is not that the accumulated time of the timer is equal to PT. Instead, the condition is met when the accumulated time of the timer reaches PT and the instruction is executed for the first time. Therefore, there is a delay of up to one task period.

· PT and ET are set in ms, with the accuracy of 100 nanoseconds.

· When both In and Reset change to TRUE, Reset takes priority. That is, the value of ET changes to 0 and Q changes to FALSE.

· If the value of In is TRUE at the beginning of running, timing starts from the time.

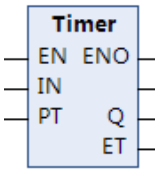
· When PT is set to T#0ms and the value of In changes to TRUE, Q changes to TRUE.

· Before the value of ET reaches the value of PT, the value of PT can be changed, but the cumulative time value of ET is still calculated based on the source value of PT.

3.4. 3 Timer

This instruction works as an accumulation timer.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
Timer	Hundred-ms timer	FC		Timer(In :=, PT :=, Q =>, ET =>);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
In	Timer input	BOOL	[FALSE, TRUE]	-	TRUE: The timer operates FALSE: The timer stops
PT	Preset time	UINT	Depends on data types	-	Maximum value for timing, in increments of 100 ms

Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
Q	Timer output	BOOL	[FALSE, TRUE]	-	TRUE: ET reaches PT FALSE: ET does not reach PT
ET	Remaining time	UINT	Depends on data types	-	Remaining time

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
PT	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-
Q	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
ET	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-
PV	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
QU	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
QU	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

■ Function

This instruction outputs TRUE when the set time elapses after the timer starts. The time is set in increments of 100 ms.

The timer is reset when timer input value of In changes to FALSE. Remaining time ET is set to set time PT, and timer output Q changes to FALSE.

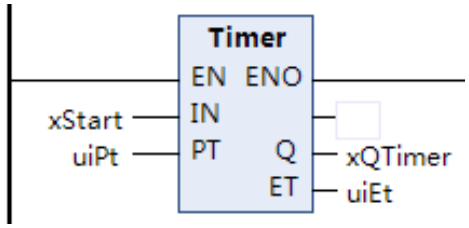
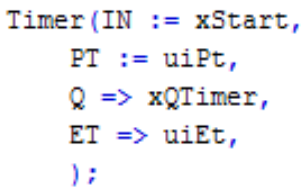
The timer starts when In changes to TRUE. The value of ET is constantly decremented as time elapses.

When ET reaches 0, timer output Q changes to TRUE. ET is not decremented after that.

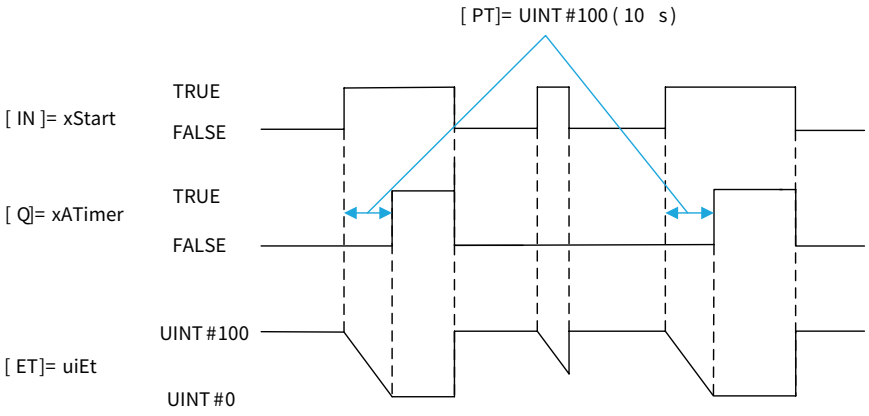
The timer is reset if In changes to FALSE after the timer starts but before ET reaches 0.

■ Timing diagram

The following figure shows a programming example and timing chart for a PT of UINT#100. The value of variable xQTimer changes to TRUE 100 x 100 ms (10s) after variable xStart changes to TRUE.

LD	ST
	


```
Timer(IN := xStart,
      PT := uiPt,
      Q => xQTimer,
      ET => uiEt,
      );
```



■ Precautions

· If multiple timers are called within the same Program Organization Unit (POU), program errors may occur.

· The timing error for which Q is TRUE for PT is one more task period. The timer judges whether the ET timing of each task period has reached PT. If so, there is a delay of one task period.

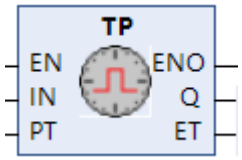
· The timer starts as soon as operation starts if the value of In has been TRUE.

· If the value of PT changes, the new value needs to be used from the next time when the timer is reset. The new value is not used during timing.

3.4. 4 TP

This instruction outputs the current elapsed time and timing status signal based on the input elapsed time.

■ Instruction description

Instruction	Name	LD Expression	ST Expression
TP	Timer pulse		`TP(IN:= , PT:= , Q=> , ET=> );`

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
IN	Input signal	BOOL	[TRUE, FALSE]	FALSE	When the value changes to TRUE, timing starts.
PT	Cumulative time storage start unit	TIME	0 to 4294967295	0	Preset time

Output variables

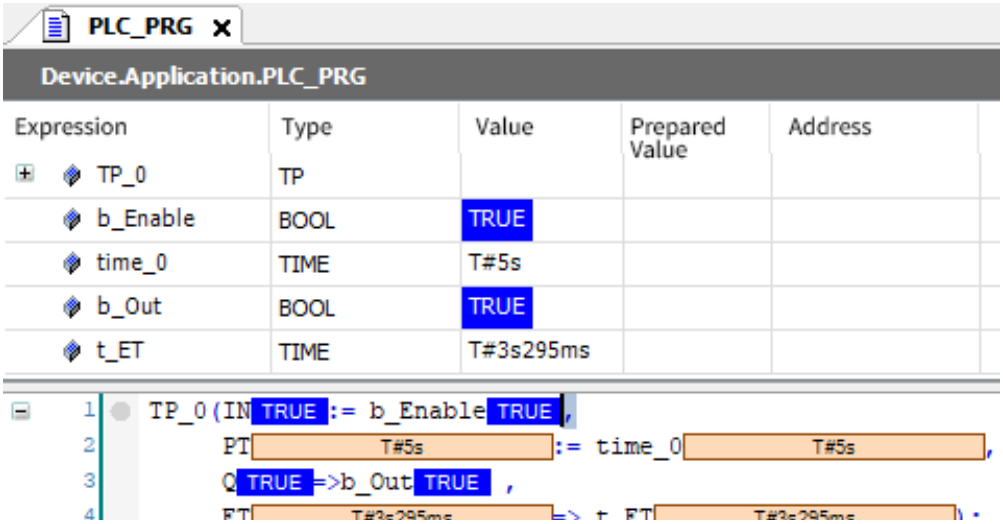
Output Variable	Name	Data Type	Value Range	Initial Value	Description
Q	Timing completed	BOOL	[TRUE, FALSE]	FALSE	When the value changes to FALSE, timing is completed.
ET	Elapsed time	TIME	0 to 4294967295	0	Preset time

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
IN	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
PT	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-
Q	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
ET	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-

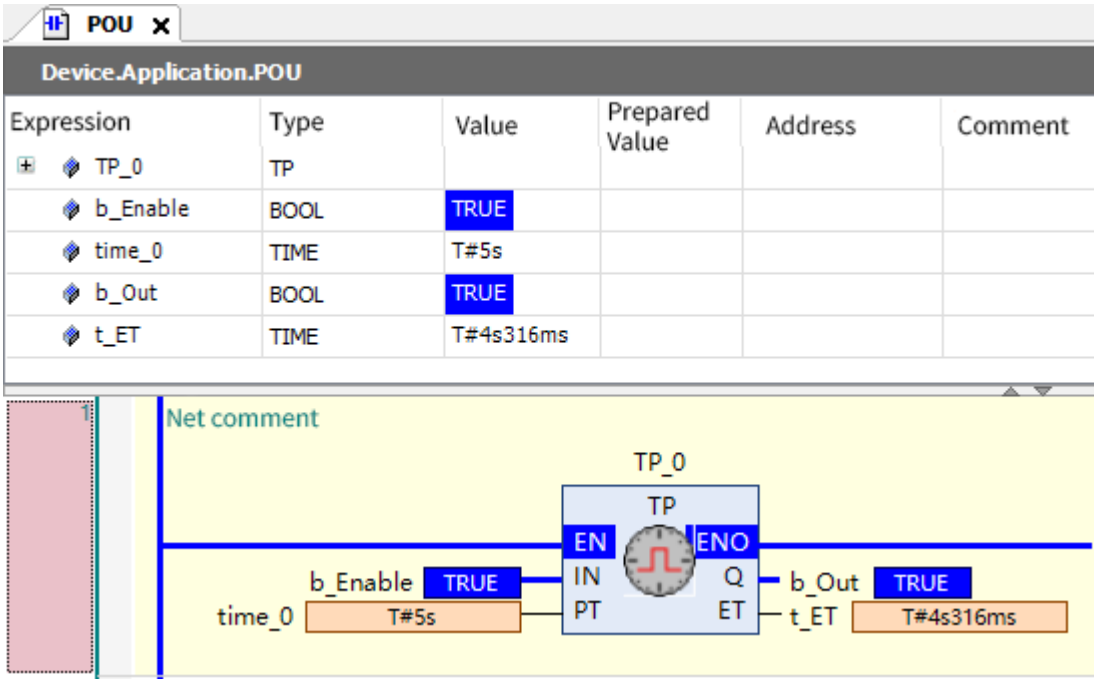
■ Program example

When In is TRUE, the Q output has a high-level signal for the time of PT. ET displays the current time. When ET is equal to PT, Q changes to FALSE.

ST



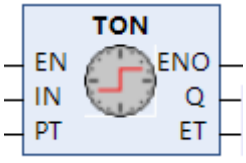
LD



3.4. 5 TON

This instruction generates an output signal indicating the completion of timing after a certain delay from power-on.

■ Instruction format

Instruction	Name	LD Expression	ST Expression
TON	On-delay timer		TON(IN:= , PT:= , Q=> , ET=>);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
IN	Input signal	BOOL	[TRUE, FALSE]	FALSE	When the value changes to TRUE, timing starts.
PT	Cumulative time storage start unit	TIME	0 to 4294967295	0	Preset time

Output variables

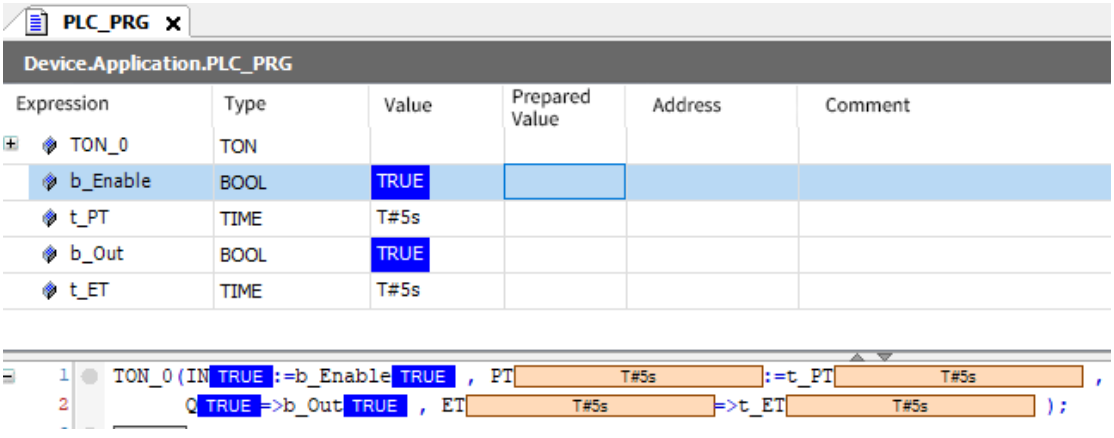
Output Variable	Name	Data Type	Value Range	Initial Value	Description
Q	Timing completed	BOOL	[TRUE, FALSE]	FALSE	When the value changes to TRUE, timing is completed.
ET	Elapsed time	TIME	0 to 4294967295	0	Preset time

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
IN	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
PT	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-
Q	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
ET	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-

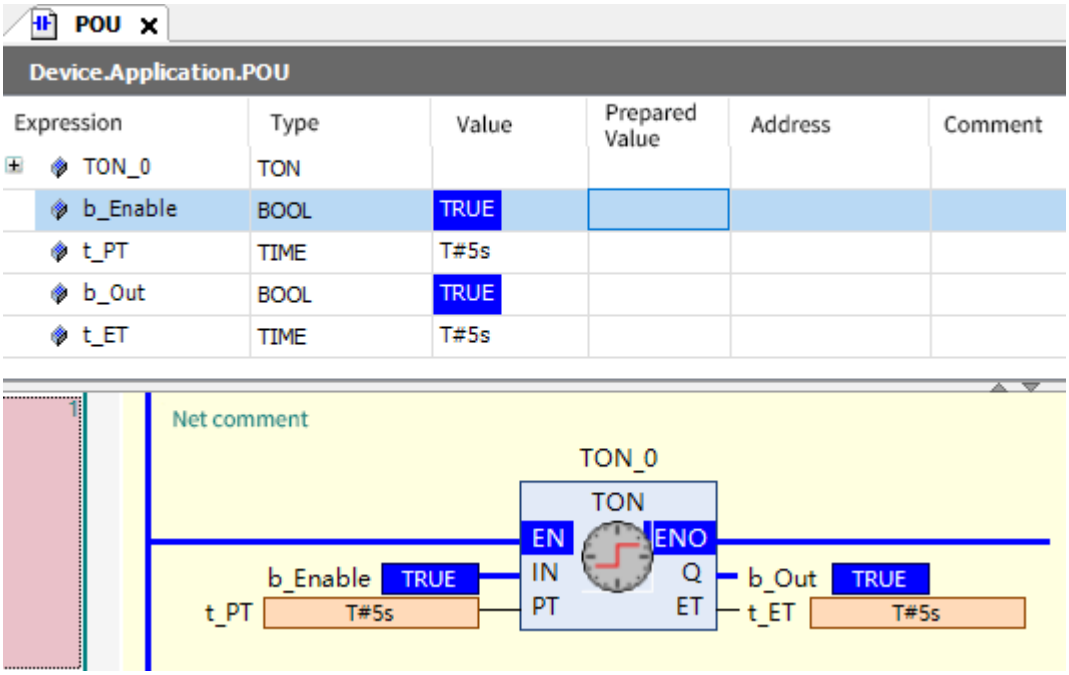
■ Program example

The input signal PT specifies the elapsed time. Timing starts when the input signal In changes to TRUE. After the timing is completed, Q is set to TRUE.

ST



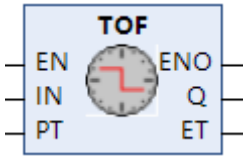
LD



3.4. 6 TOF

This instruction generates an output signal indicating the completion of timing after a certain delay from power-off.

■ Instruction format

Instruction	Name	LD Expression	ST Expression
TOF	Off-delay timer		TOF(IN:= , PT:= , Q=> , ET=>);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
IN	Input signal	BOOL	[TRUE, FALSE]	FALSE	When the value changes to FALSE, timing starts.
PT	Cumulative time storage start unit	TIME	0 to 4294967295	0	Preset time

Output variables

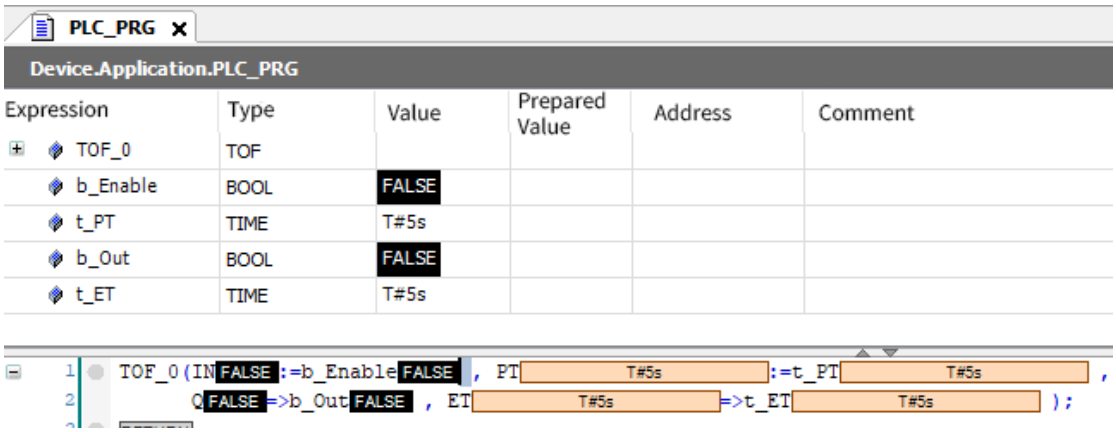
Output Variable	Name	Data Type	Value Range	Initial Value	Description
Q	Timing completed	BOOL	[TRUE, FALSE]	FALSE	When the value changes to TRUE, timing is completed.
ET	Elapsed time	TIME	0 to 4294967295	0	Preset time

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
IN	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
PT	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-
Q	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
ET	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-

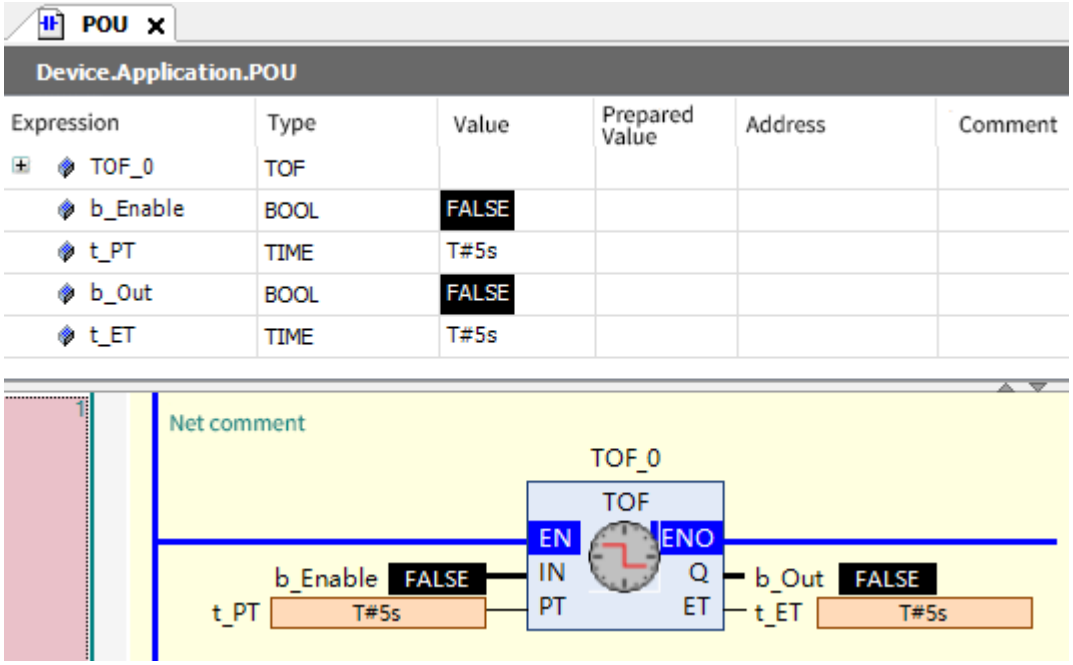
■ Program example

The input signal PT specifies the elapsed time. Timing starts when the input signal In changes to FALSE. After the timing is completed, Q is set to FALSE.

ST



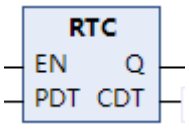
LD



3.4. 7 RTC

This instruction provides the clock function to start timing from the set time.

■ Instruction format

Instruction	Name	LD Expression	ST Expression
RTC	Real-time clock		RTC(EN:= , PDT:= , Q=> , CDT=>);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
PDT	Timing start time	DATE_AND_TIME	0 (1970-01-01, 00:00:00) to 4294967295 (2106-02-07, 06:28:15)	1970-01-01, 00:00:00	Timing startup

Output variables

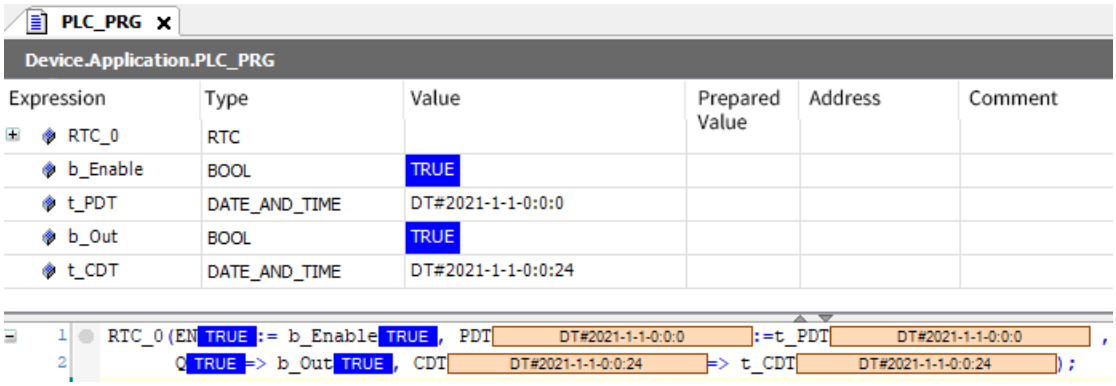
Output Variable	Name	Data Type	Value Range	Initial Value	Description
Q	Timing completed	BOOL	[TRUE, FALSE]	FALSE	When the value changes to FALSE, timing is valid.
CDT	Elapsed time	DATE_AND_TIME	0 (1970-01-01, 00:00:00) to 4294967295 (2106-02-07, 06:28:15)	1970-01-01, 00:00:00	Timing startup

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
PDT	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-
Q	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
CDT	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-

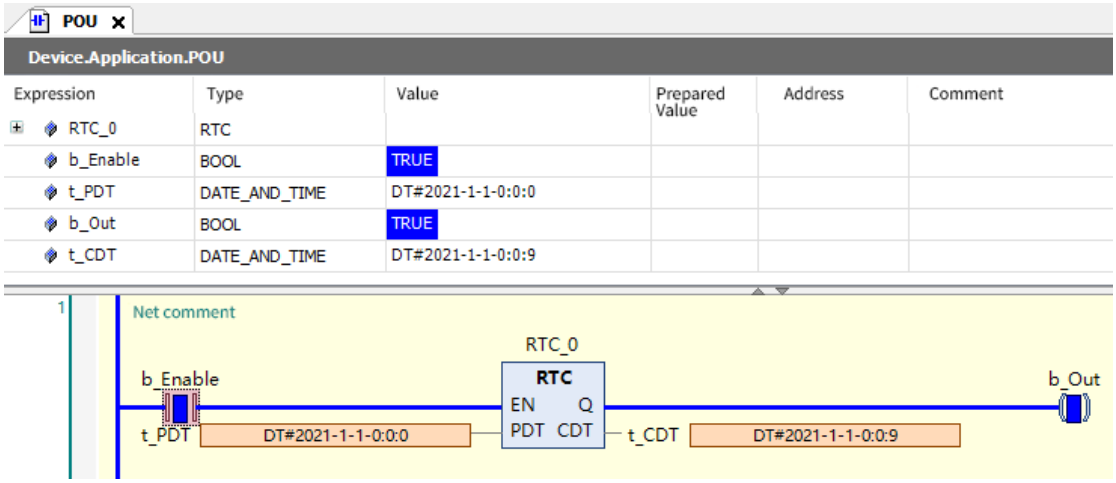
■ Program example

When the input value of EN is TRUE, timing starts from the set time PDT. The output value of Q is TRUE and the CDT outputs the current elapsed time.

ST



LD



3. 5 Bit and Word Logic Instructions

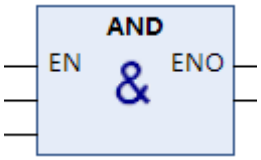
3.5. 1 Instruction List

Instruction Category	Name	FB/FC	Function
Bit and word logic instructions	AND	FC	Logical AND
	OR	FC	Logical OR
	NOT	FC	Bit reversal
	XOR	FC	Logical exclusive OR
	PLS	FB	Rising edge output
	PLF	FB	Falling edge output
	ALT	FB	Alternate output
	BOUT	FC	Bit data output
	BSET	FC	Bit data set
	BRST	FC	Bit data reset
	AryAnd	FC	Array logical AND
	AryOr	FC	Array logical OR
	AryXor	FC	Array logical exclusive OR
	AryXorN	FC	Array logical exclusive NOR
	SR	FB	Set-priority keep
	RS	FB	Reset-priority keep
	R_TRIG	FB	Up trigger
	F_TRIG	FB	Down trigger
	EXTRACT	FC	Bit extraction
	PUTBIT	FC	Bit assignment
	PACK	FC	Bit packing
	UNPACK	FC	Bit unpacking
	BIT_AS_BYTE	FC	8-bit packing into byte
	BYTE_AS_BIT	FC	Byte unpacking to bits
	BIT_AS_WORD	FC	16-bit packing into word
	WORD_AS_BIT	FC	Word unpacking to bits
	BIT_AS_DWORD	FC	32-bit packing into doubleword
	DWORD_AS_BIT	FC	Doubleword unpacking to bits
	AryAnd	FC	Array logical AND
	AryOr	FC	Array logical OR
	AryXor	FC	Logical exclusive OR
	AryXorN	FC	Array logical exclusive NOR

3.5. 2 AND

This instruction outputs 1 for the right output bit when the left two input bits are not 0, and outputs 0 when they are 0.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
AND	Logical AND	FC		AND

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
In1	Input 1	-	Depends on data types	-	Input 1
In2	Input 2	-	Depends on data types	-	Input 2

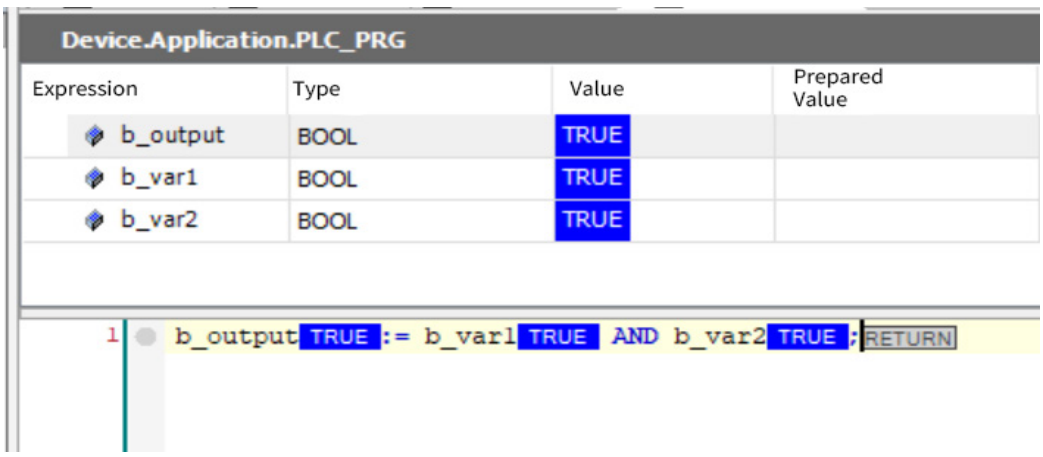
Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
Out	Output result	-	Depends on data types	-	Output result

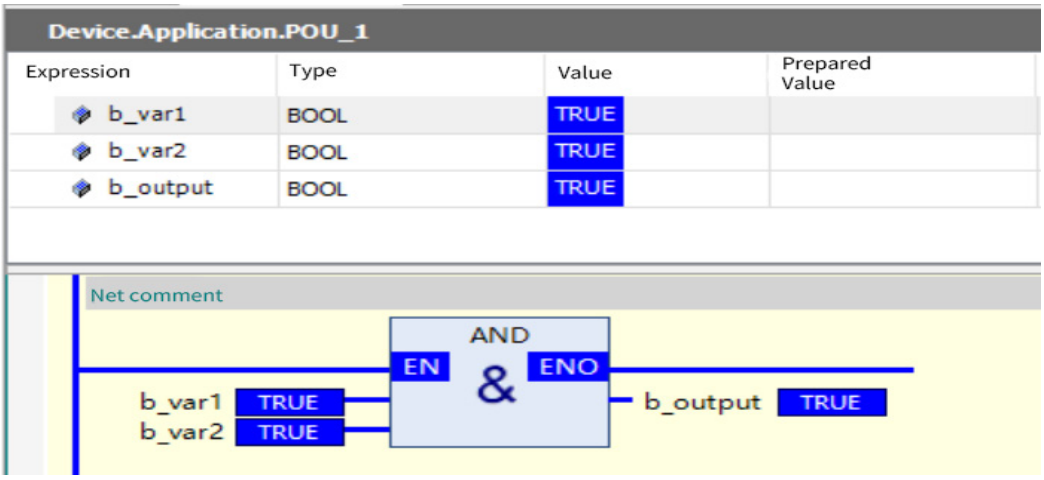
	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In1	√	√	√	√	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
In2	√	√	√	√	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Out	√	√	√	√	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

■ Program example

ST:



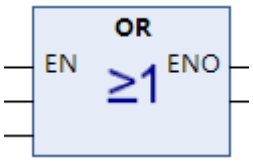
LD:



3.5. 3 OR

This instruction outputs 1 for the right output bit when the left two input bits contain one or no 0, and outputs 0 when both are 0.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
OR	Logical OR	FC		OR

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
In1	Input 1	-	Depends on data types	-	Input 1
In2	Input 2	-	Depends on data types	-	Input 2

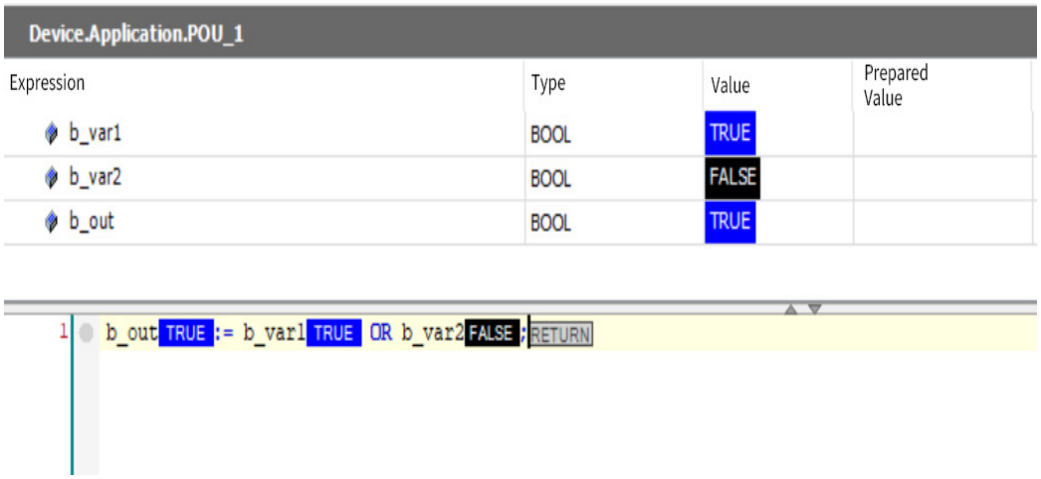
Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
Out	Output result	-	Depends on data types	-	Output result

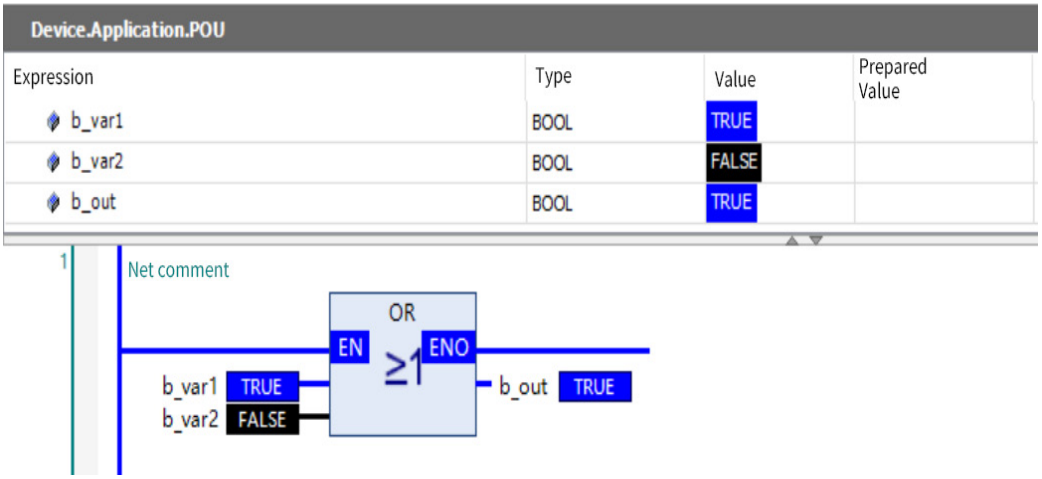
	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In1	√	√	√	√	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
In2	√	√	√	√	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Out	√	√	√	√	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

■ Program example

ST:



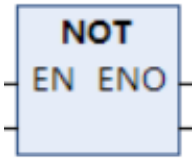
LD:



3.5. 4 NOT

This instruction outputs 1 for the right output bit when the left input bit is 0, and outputs 0 when the left input bit is 1.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
NOT	Bit reversal	FC		NOT

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
In1	Input 1	-	Depends on data types	-	Input 1
In2	Input 2	-	Depends on data types	-	Input 2

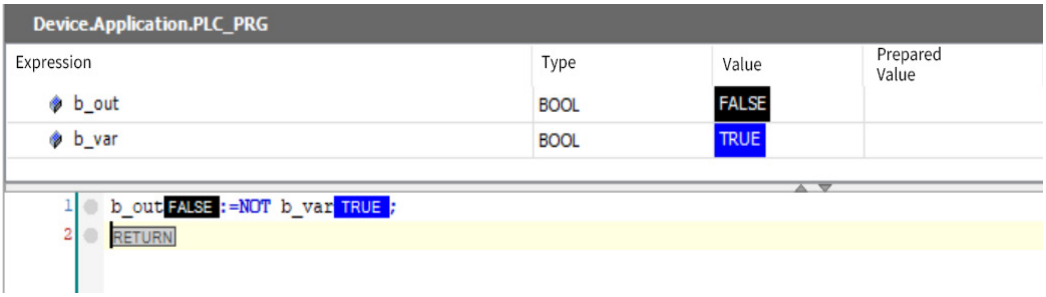
Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
Out	Output result	-	Depends on data types	-	Output result

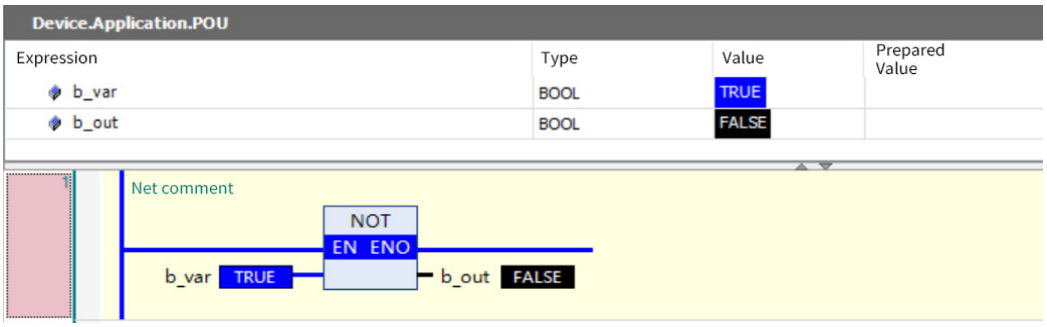
	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In1	√	√	√	√	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
In2	√	√	√	√	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Out	√	√	√	√	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

■ Program example

ST:



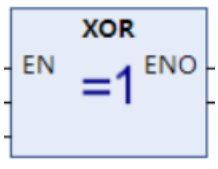
LD:



3.5. 5 XOR

This instruction outputs 1 when the left input bits are 1 and 0, and outputs 0 when they are both 1 or both 0.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
XOR	Logical exclusive OR	FC		XOR

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
In1	Input 1	-	Depends on data types	-	Input 1
In2	Input 2	-	Depends on data types	-	Input 2

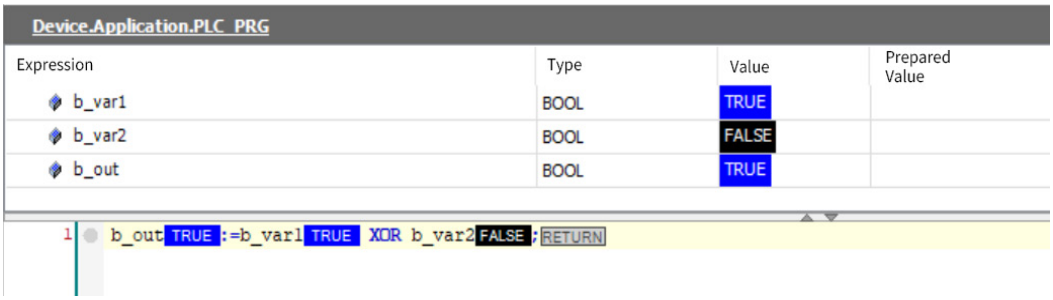
Output variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
Out	Output result	-	Depends on data types	-	Output result

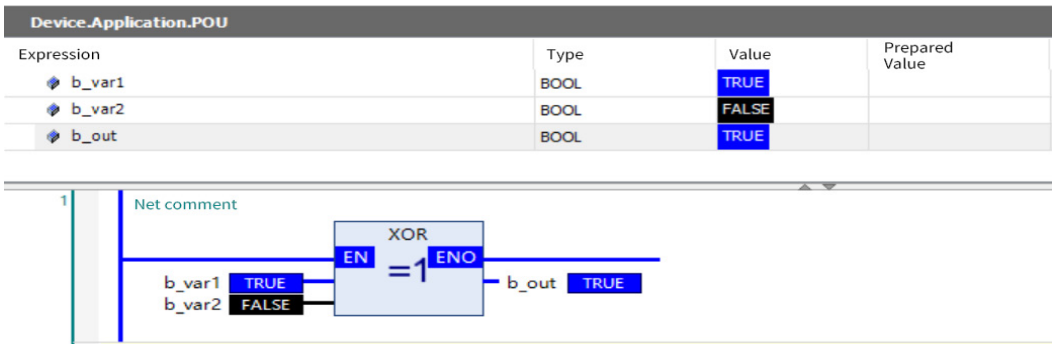
	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In1	√	√	√	√	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
In2	√	√	√	√	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Out	√	√	√	√	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

■ Program example

ST:



LD:



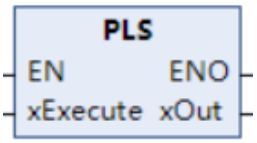
■ Precautions

· Note the following behavior of the XOR block in extended form (more than two inputs): CODESYS compares the inputs in pairs and then the corresponding results (according to the standard, but not necessarily according to expectations).

3.5. 6 PLS

This instruction sets xOut to TRUE when the xExecute variable is TRUE, and then to FALSE after one scan cycle.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
PLS	Rising edge output	FB		PLS(xExecute:= ,xOut=>);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
xExecute	Input signal	BOOL	[FALSE, TRUE]	FALSE	Measured boolean volume

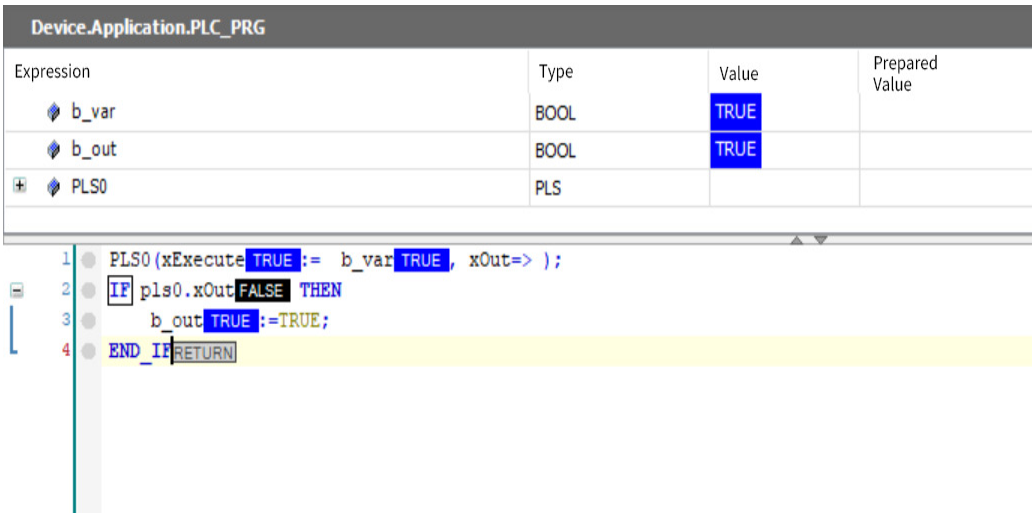
Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
Out	Output result	-	Depends on data types	-	Output result

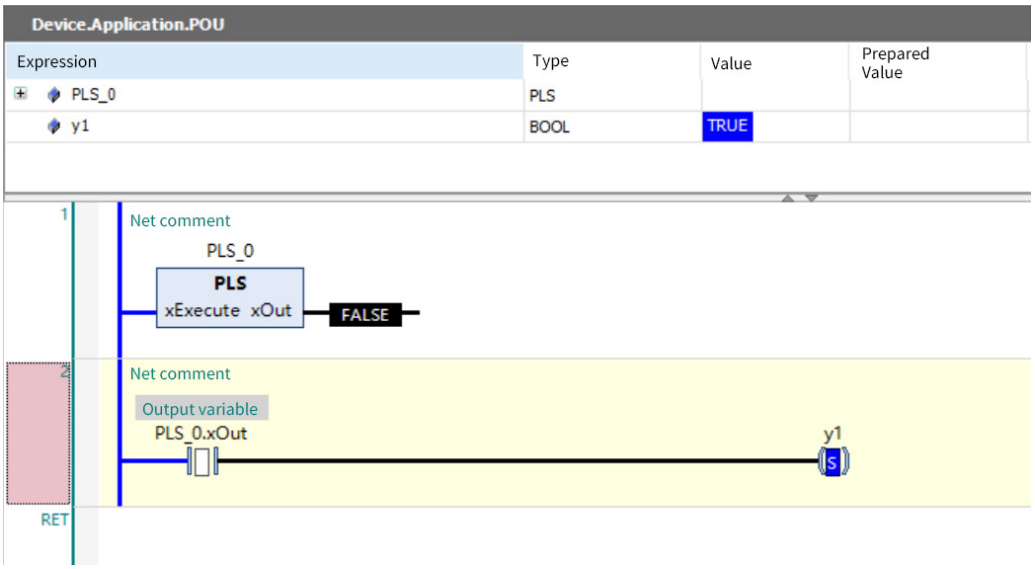
	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
xExecute	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Out	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

■ Program example

ST:



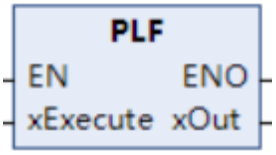
LD:



3.5. 7 PLF

This instruction sets xOut to TRUE when the xExecute variable is FALSE, and then to FALSE after one scan cycle.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
PLF	Falling edge output	FB		PLF(xExecute:= ,xOut=>);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
xExecute	Input signal	BOOL	[FALSE, TRUE]	FALSE	Measured boolean volume

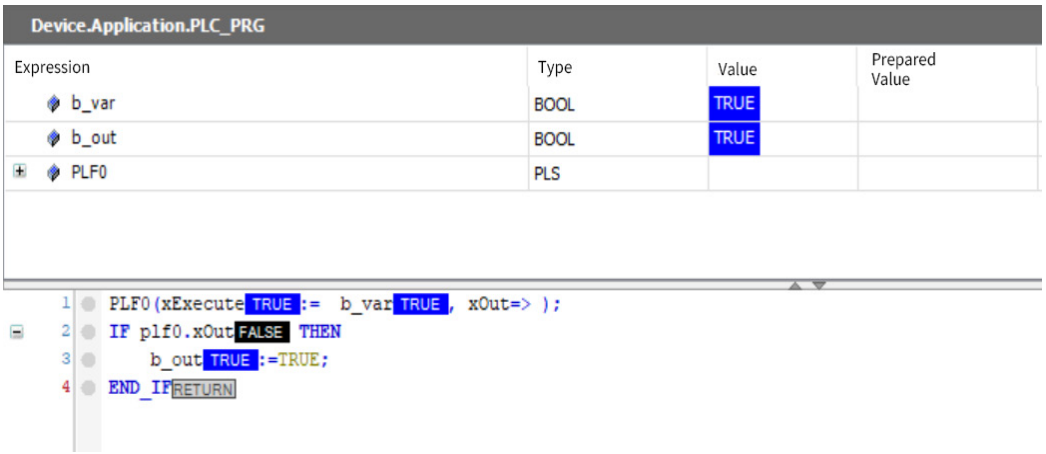
Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
xOut	Output signal	BOOL	[FALSE, TRUE]	FALSE	ON when the xExecute variable is FALSE

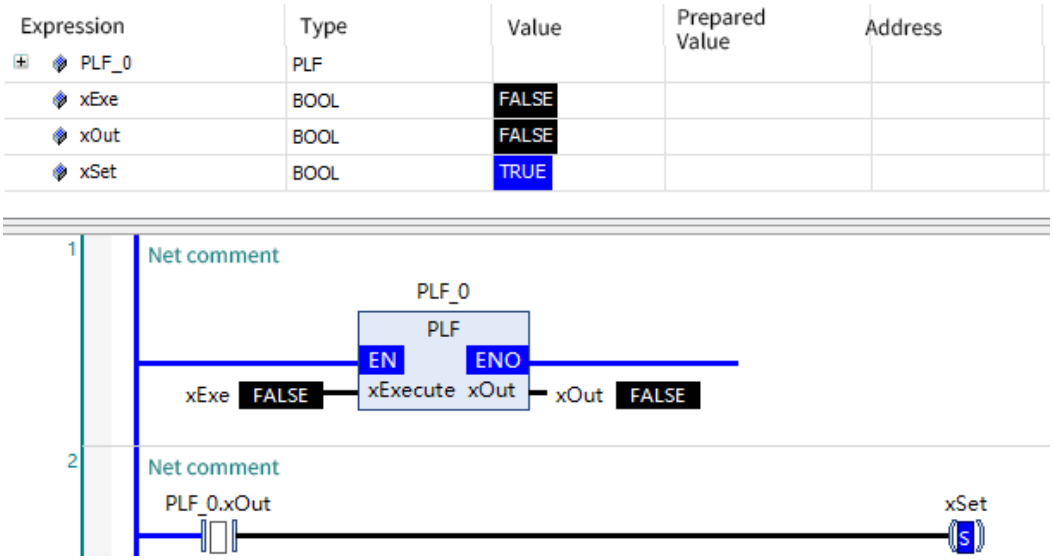
	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
xExecute	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Out	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

■ Program example

ST:



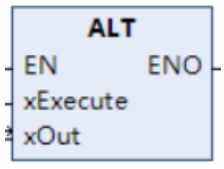
LD:



3.5. 8 ALT

This instruction sets xOut to reverse the status when the xExecute variable is TRUE.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
ALT	Alternate output	FB		ALT(xExecute:= ,xOut:=);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
xExecute	Input signal	BOOL	[FALSE, TRUE]	FALSE	Measured boolean volume

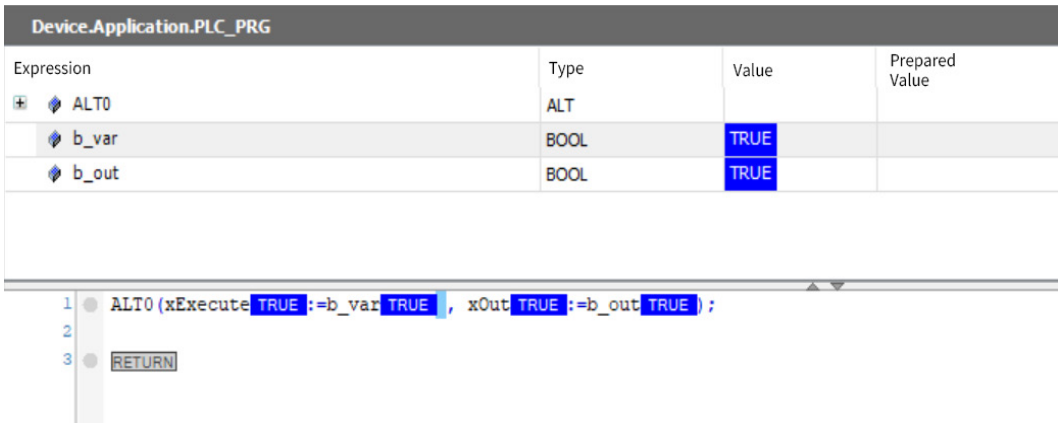
Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
xOut	Output signal	BOOL	[FALSE, TRUE]	FALSE	The status is alternated when the xExecute variable is TRUE.

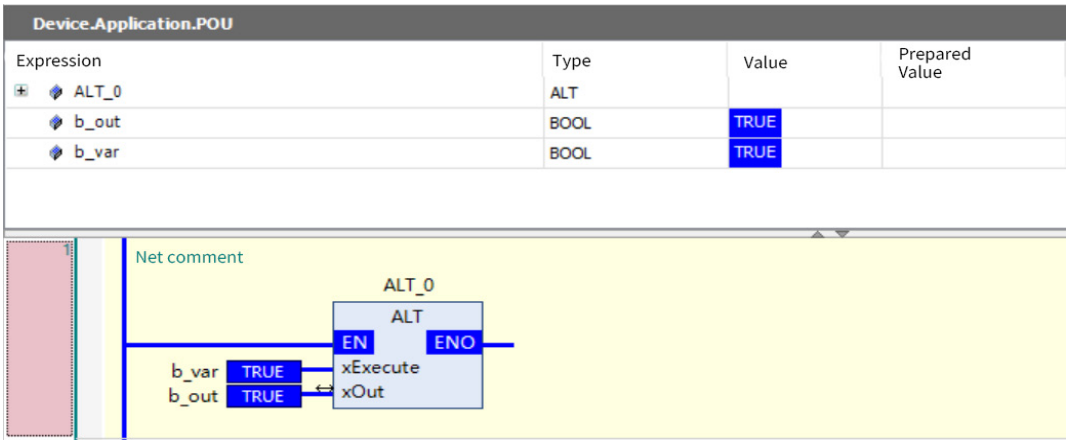
	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
xExecute	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Out	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

■ Program example

ST:



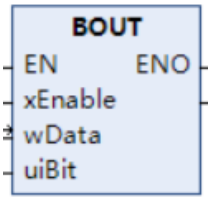
LD:



3.5. 9 BOUT

This instruction sets the status for a specified bit.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
BOUT	Bit data output	FC		BOUT(xEnable:= , wData:= , uiBit:=);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
xEnable	Input signal	BOOL	[FALSE, TRUE]	FALSE	Enabling bit triggered by level
uiBit	Input signal	UINT	Depends on data types	0	Bit to operate

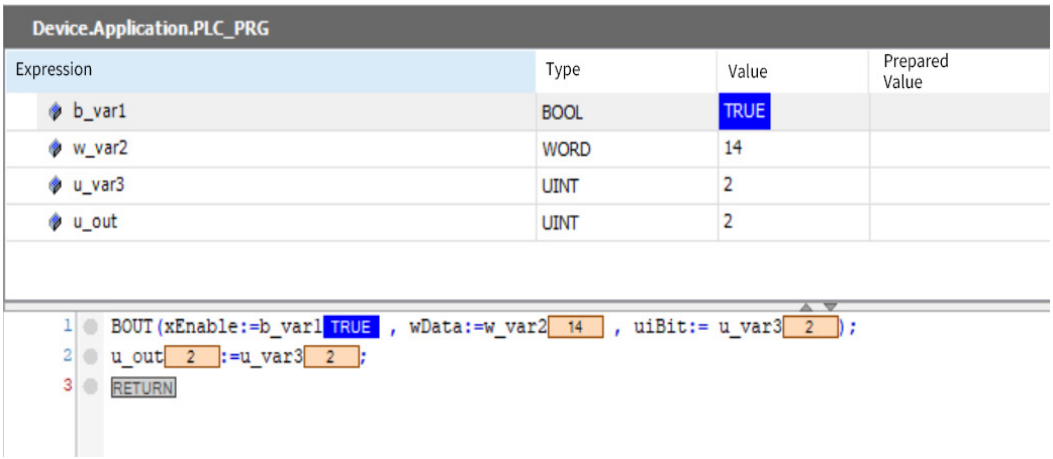
Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
wData	Input/Output signal	WORD	Depends on data types	0	Data to operate

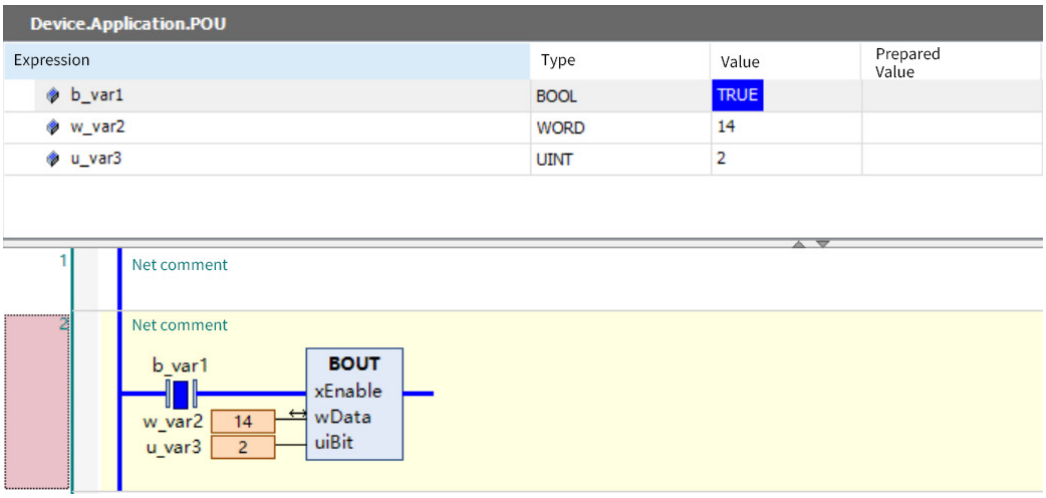
	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
xEnable	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
wData	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
uiBit	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-

■ Program example

ST: Set the initial value of wData to 10 (#1010 in binary mode). When uiBit is set to 2 and xEnable to TRUE, bit 2 of wData is set, resulting in wData of 14 (2#1110 in binary mode).



LD: Set the initial value of wData to 10 (#1010 in binary mode). When uiBit is set to 2 and xEnable to TRUE, bit 2 of wData is set, resulting in wData of 14 (2#1110 in binary mode).



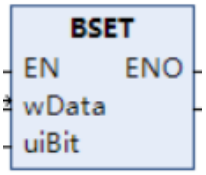
■ Precautions

· The initial value of wData is set in the variable table.

3.5. 10 BSET

This instruction sets the specified bit of data to ON. Regardless of whether the BSET instruction is still driven, the set bit remains ON. You can use the BRST instruction to set the bit to OFF.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
BSET	Bit data set	FC		BSET(wData:= , uiBit:=);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
uiBit	Input signal	UINT	Depends on data types	0	Bit to set to ON

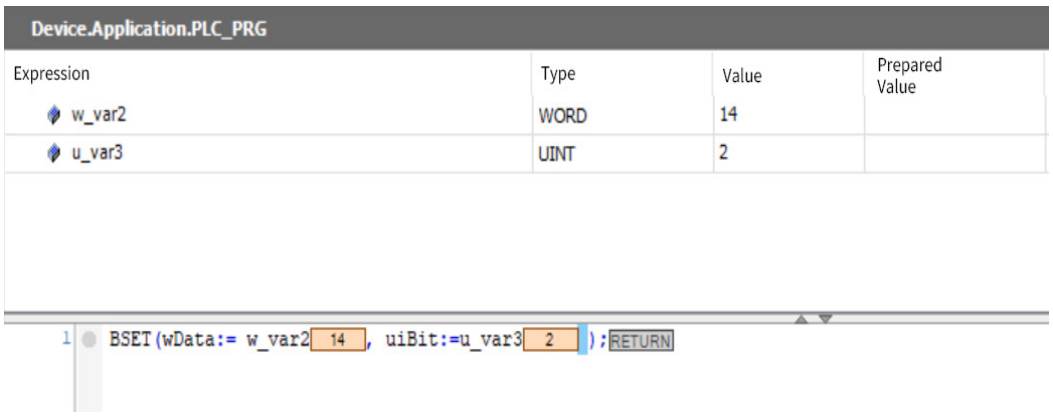
Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
wData	Input/Output signal	WORD	Depends on data types	0	Data to operate

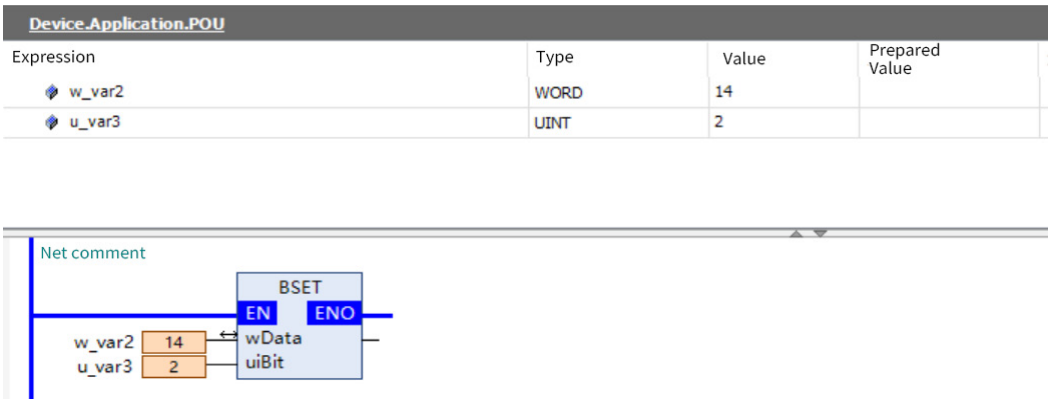
	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
wData	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
uiBit	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-

■ Program example

ST:



LD:



3.5. 11 BRST

This instruction resets the status of a specified bit to OFF.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
BRST	Bit data reset	FC	BRST EN ENO wData uiBit	BRST(wData:= , uiBit:=);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
wData	Input/Output signal	WORD	Depends on data types	0	Data to operate
uiBit	Input signal	UINT	Depends on data types	0	Bit to set to ON

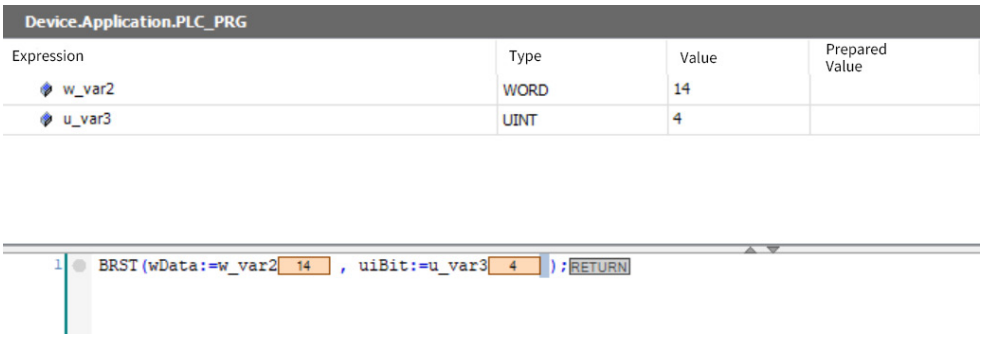
Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
wData	Input/Output signal	WORD	Depends on data types	0	Data to operate

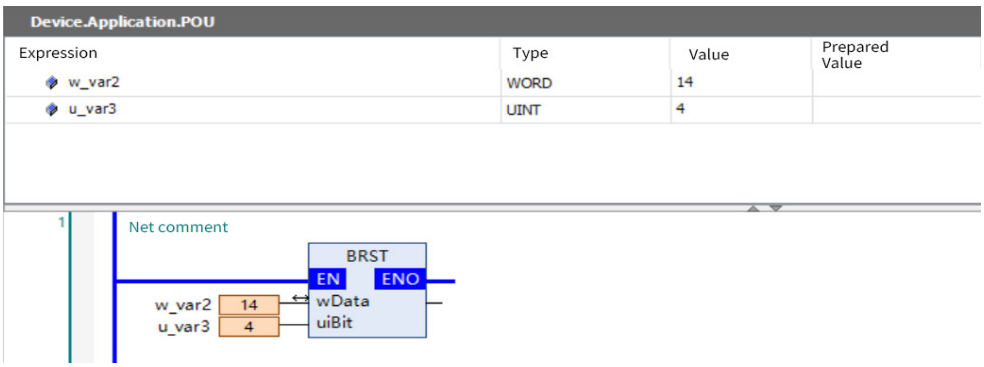
	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
wData	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
uiBit	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-

■ Program example

ST:



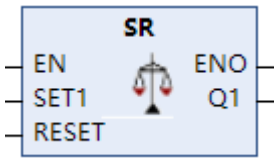
LD:



3.5. 12 SR

This instruction gives priority to the Set input if both the Set input and Reset input are TRUE.

■ Instruction description

Instruction	Name	LD Expression	ST Expression
SR	Set-priority keep		SR(SET1:= , RESET:= , Q1=>);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
SET1	Set signal	BOOL	[TRUE, FALSE]	FALSE	-
Reset	Reset signal	BOOL	[TRUE, FALSE]	FALSE	-

Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
Q1	Output signal	BOOL	[TRUE, FALSE]	FALSE	Set priority output

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
SET1	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Reset	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

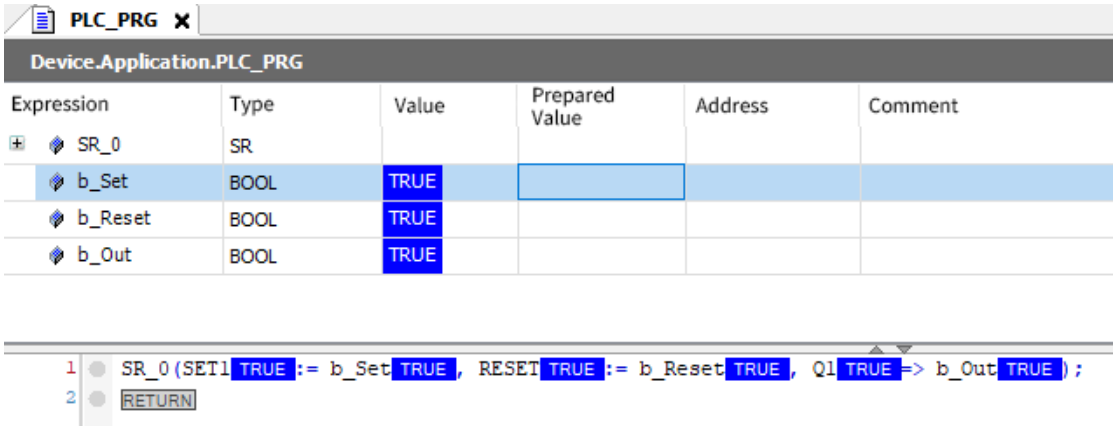
	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Q1	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

■ Program example

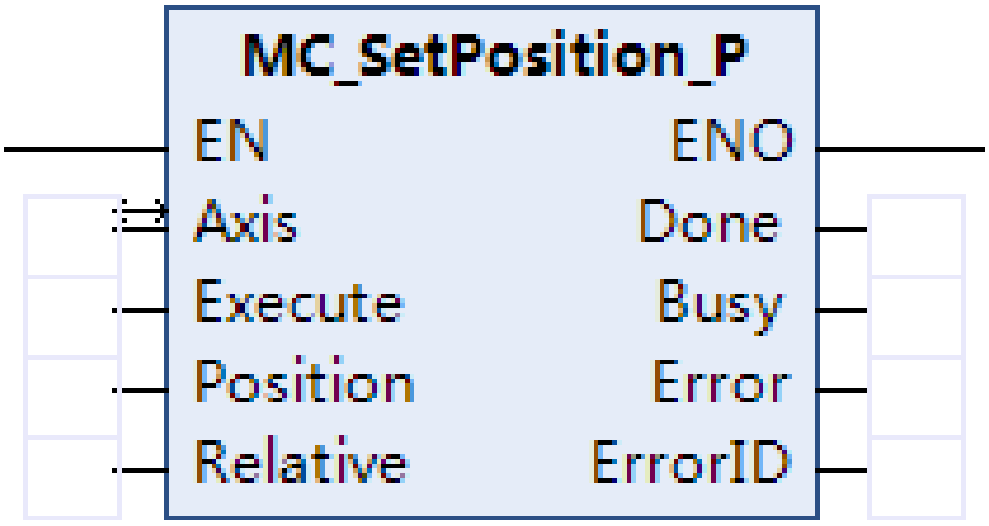
When both the set input signal SET and the reset input signal Reset are active, the set input signal SET takes priority as the output Q.

SET1	Reset	Q
TRUE	TRUE	TRUE
FALSE	TRUE	FALSE
TRUE	FALSE	TRUE
FALSE	FALSE	Not changed

ST



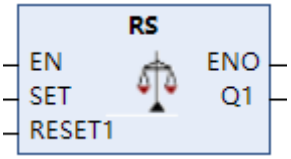
LD



3.5. 13 RS

This instruction gives priority to the Reset input if both the Set input and Reset input are TRUE.

■ Instruction description

Instruction	Name	LD Expression	ST Expression
RS	Reset-priority keep		RS(SET:= , RESET1:= , Q1=>);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
SET1	Set signal	BOOL	[TRUE, FALSE]	FALSE	-
Reset	Reset signal	BOOL	[TRUE, FALSE]	FALSE	-

Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
Q1	Output signal	BOOL	[TRUE, FALSE]	FALSE	Set priority output

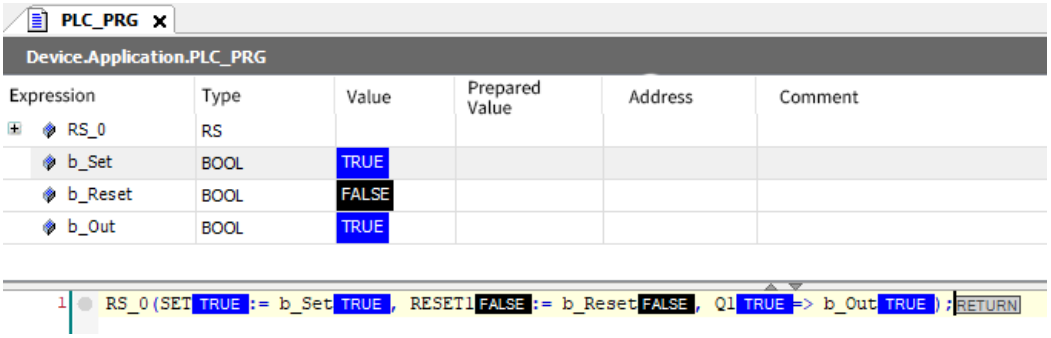
	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
SET1	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Reset	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Q1	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

■ Program example

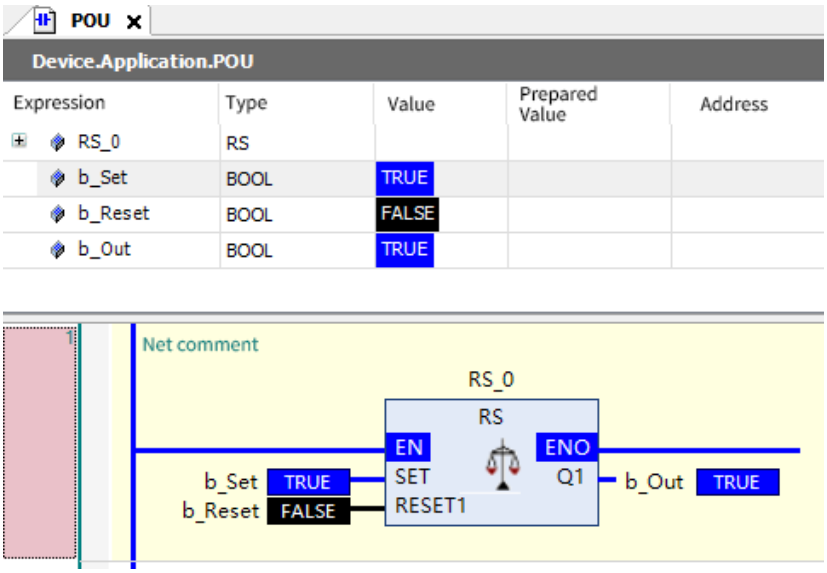
When both the set input signal SET and the reset input signal Reset are active, the reset input signal Reset takes priority as the output Q.

SET	RESET1	Q
TRUE	TRUE	TRUE
FALSE	TRUE	FALSE
TRUE	FALSE	TRUE
FALSE	FALSE	Not changed

ST



LD

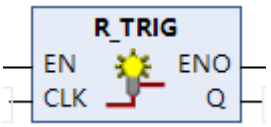


3.5. 14 R_TRIG

This instruction outputs a TRUE signal for only one task period when an input signal transitions from FALSE to TRUE (rising edge).

■ Instruction description

This instruction outputs a TRUE signal for only one task period when an input signal transitions from FALSE to TRUE (rising edge).

Instruction	Name	LD Expression	ST Expression
R_TRIG	Up trigger		R_TRIG(CLK:= , Q=>);

■ Variables

Input variables

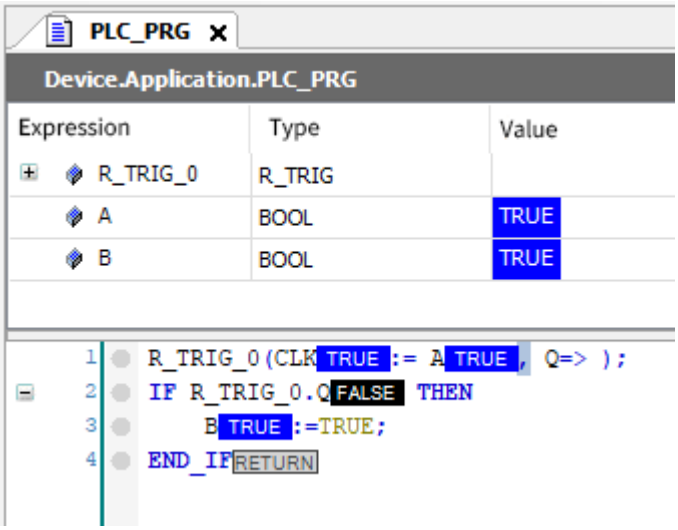
Input Variable	Name	Data Type	Value Range	Initial Value	Description
CLK	Input signal	BOOL	[TRUE, FALSE]	FALSE	Valid when FALSE is changed to TRUE

Output variables

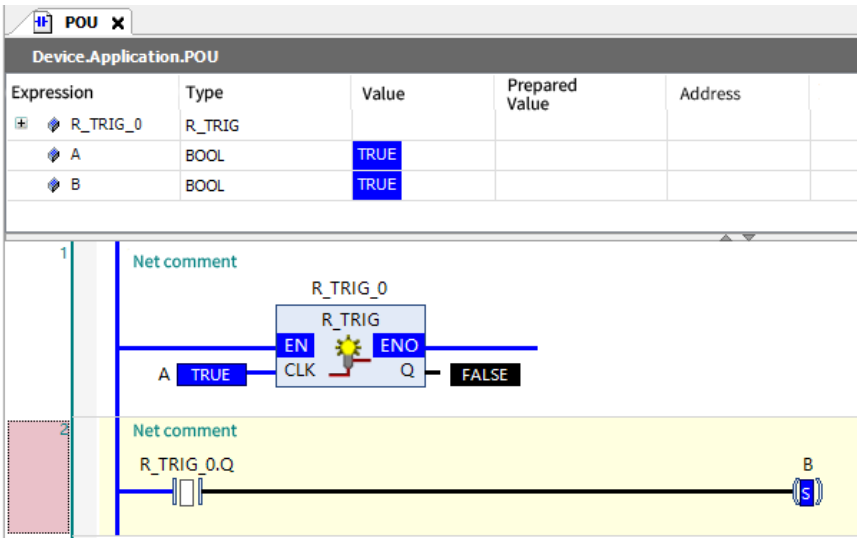
Output Variable	Name	Data Type	Value Range	Initial Value	Description
Q	Output signal	BOOL	[TRUE, FALSE]	FALSE	TRUE detected

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
CLK	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Q	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

ST



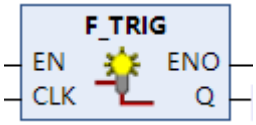
LD



3.5. 15 F_TRIG

This instruction outputs a TRUE signal for only one task period when an input signal transitions from TRUE to FALSE (falling edge).

■ Instruction description

Instruction	Name	LD Expression	ST Expression
F_TRIG	Down trigger		F_TRIG(CLK:= , Q=>);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
CLK	Input signal	BOOL	[TRUE, FALSE]	FALSE	Valid when TRUE is changed to FALSE

Output variables

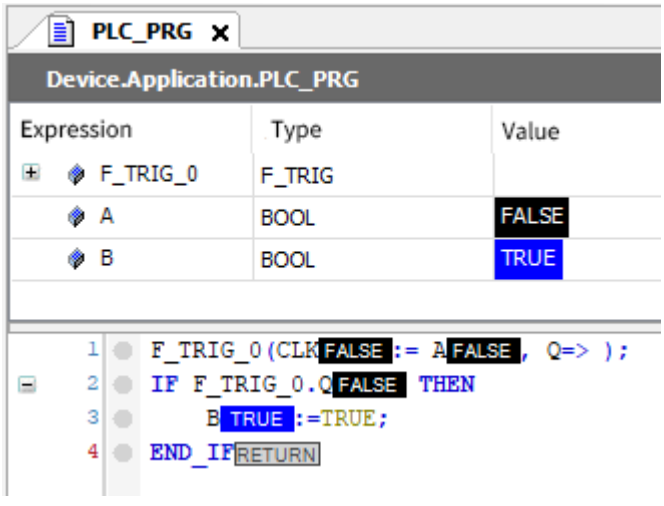
Output Variable	Name	Data Type	Value Range	Initial Value	Description
Q	Output signal	BOOL	[TRUE, FALSE]	FALSE	FALSE detected

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
CLK	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Q	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

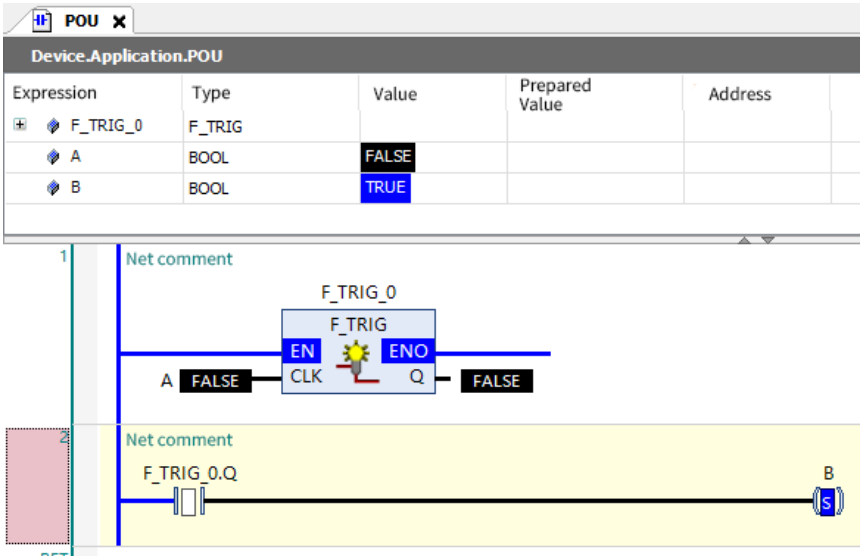
■ Program example

When the value of the input signal CLK is changed from TRUE to FALSE, the output signal Q is set to TRUE in only one task period and set to FALSE in other conditions.

ST



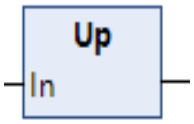
LD



3.5. 16 Up

This instruction outputs a TRUE signal for only one task period when an input signal transitions from FALSE to TRUE (rising edge).

■ Description

Instruction	Name	LD Expression	ST Expression
Up	Up trigger		None

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
In	Input signal	BOOL	[TRUE, FALSE]	FALSE	Valid when TRUE is changed to FALSE

Output variables

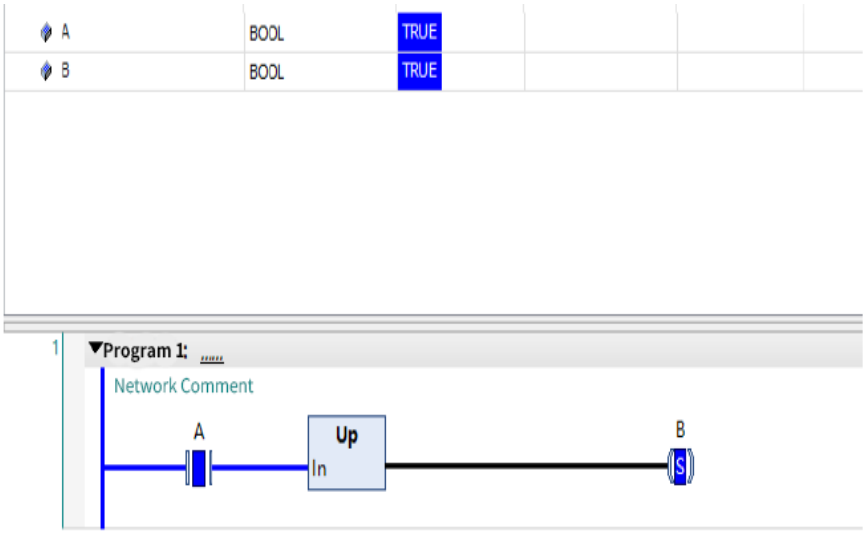
Output Variable	Name	Data Type	Value Range	Initial Value	Description
Out	Output signal	BOOL	[TRUE, FALSE]	FALSE	TRUE detected

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Out	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

■ Program example

When the value of the input signal In is changed from FALSE to TRUE, the output pin signal is set to TRUE in only one task period and set to FALSE in other conditions.

LD



3.5. 17 Down

This instruction outputs a TRUE signal for only one task cycle when an input signal transitions from TRUE to FALSE (falling edge).

■ Description

Instruction	Name	LD Expression	ST Expression
Down	Down trigger		None

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
In	Input signal	BOOL	[TRUE, FALSE]	FALSE	Valid when TRUE is changed to FALSE

Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
Out	Output signal	BOOL	[TRUE, FALSE]	FALSE	FALSE detected

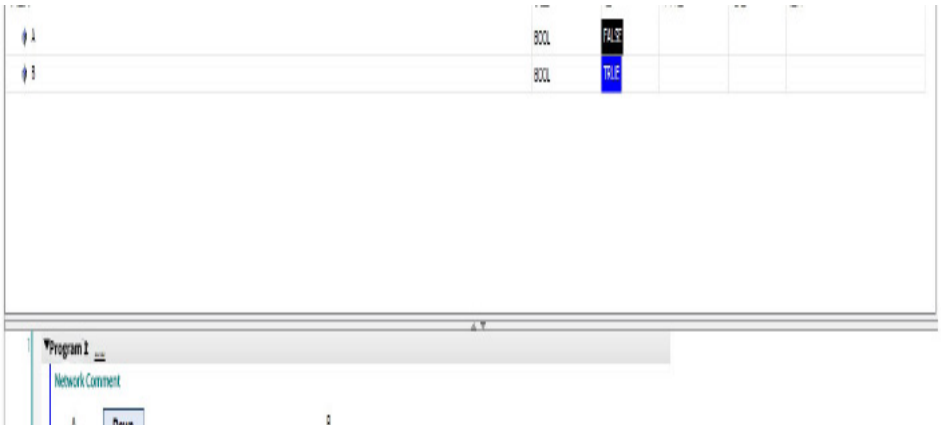
	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Out	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

■ Program example

When the value of the input signal In is changed from TRUE to FALSE, the output pin signal is set to TRUE

in only one task period and set to FALSE in other conditions.

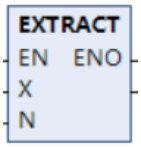
LD



3.5. 18 EXTRACT

This instruction reads the status of bit N in data source X and provides output from the right side.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
EXTRACT	Bit extraction	FC		c:=EXTRACT(X:= , N:=);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
X	Data source	DWORD	-	0	Data to operate
N	Read bit	BYTE	-	0	Bit to read

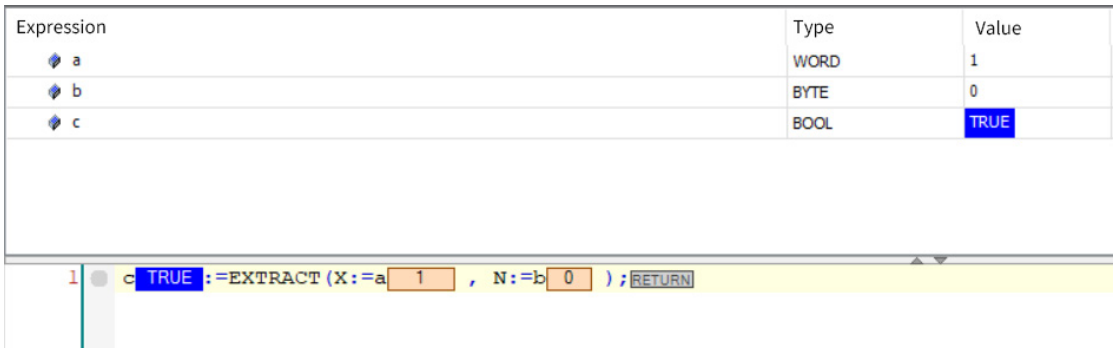
Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
OUT	Read bit status	B00L	-	0	Bit status that is read

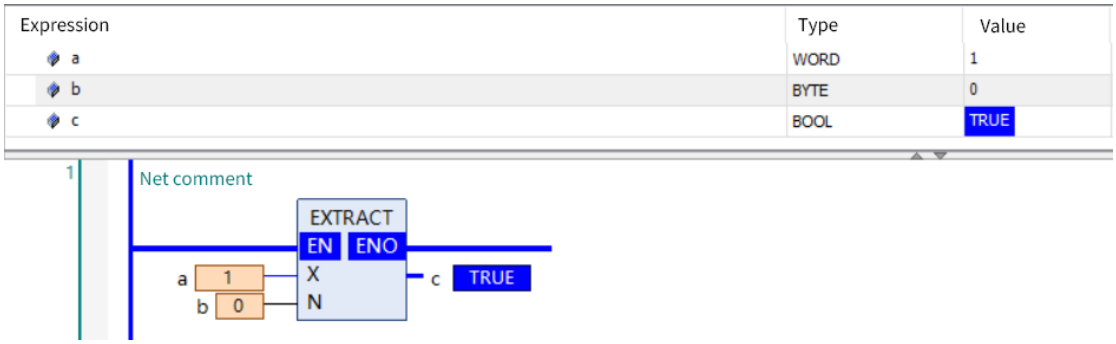
	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
X	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
N	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
OUT	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

■ Program example

LD:



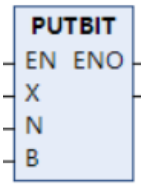
ST:



3.5. 19 PUTBIT

This instruction reads data from X, converts it to binary mode, and selects bit N. When B is set to ON, this instruction sets bit N of X to ON and outputs the processed data.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
PUTBIT	Bit assignment	FC		d:=PUTBIT(X:= , N:= , B:=);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
X	Data source	DWORD	-	0	Operated data
N	Operation bit	BYTE	-	0	Bit to operate
B	Operating switch	BOOL	-	0	Operating switch of bit

Output variables

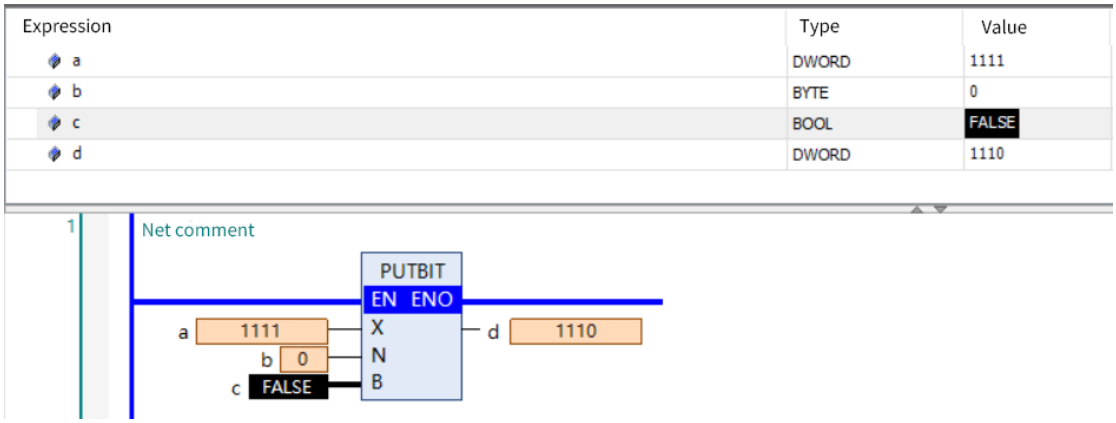
Output Variable	Name	Data Type	Value Range	Initial Value	Description
OUT	Target position	DWORD	-	0	Data after operation

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
X	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
N	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
B	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
OUT	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

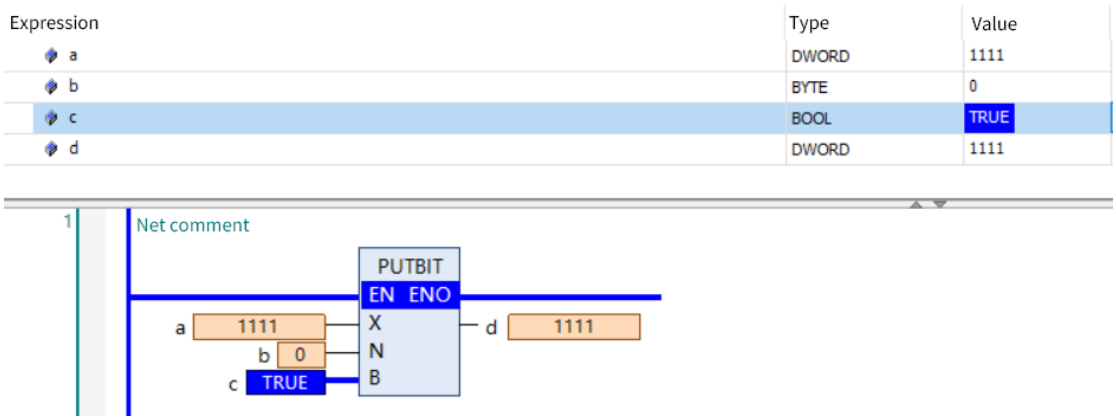
■ Program example

LD:

(1) Switch c is not closed.

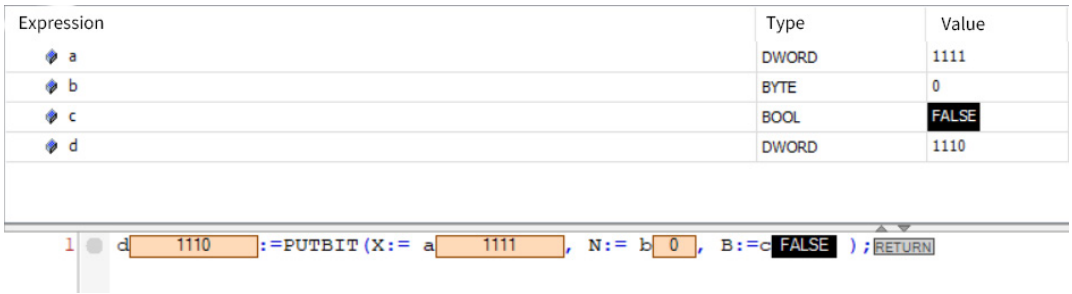


(2) Switch c is closed.

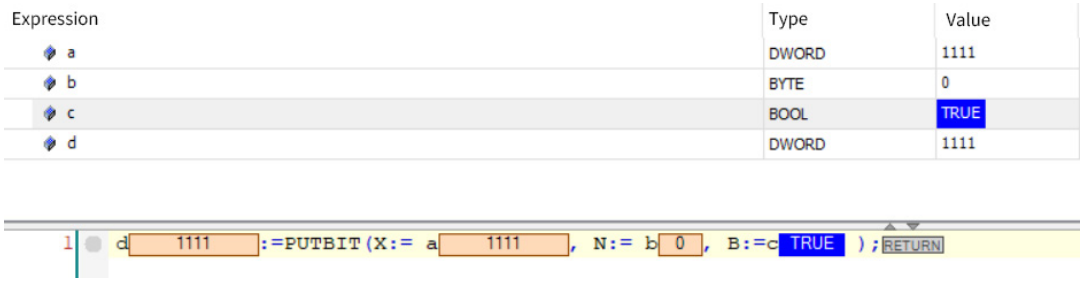


ST:

(1) Switch c is not closed.



(2) Switch c is closed.



3.5. 20 PACK

This instruction packs eight bits of the input data into a byte and then outputs it from the right side.

■ Instruction format

This instruction packs eight bits of the input data into a byte and then outputs it from the right side.

Instruction	Name	FB/FC	LD Expression	ST Expression
PACK	Bit packing	FC	PACK EN ENO B0 B1 B2 B3 B4 B5 B6 B7	a:=PACK(B0:= , B1:= , B2:= , B3:= , B4:= , B5:= , B6:= , B7:=);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
B0-B7	Data source	BOOL	-	0	Bits to pack

Output variables

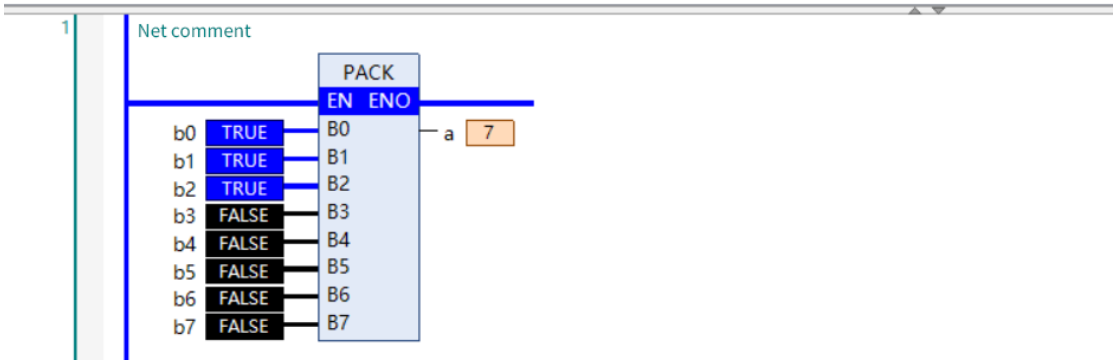
Output Variable	Name	Data Type	Value Range	Initial Value	Description
OUT	Output target	BYTE	-	0	Data after packing

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
B0-B7	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
OUT	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

■ Program example

LD:

Expression	Type	Value
a	BYTE	7
b0	BOOL	TRUE
b1	BOOL	TRUE
b2	BOOL	TRUE
b3	BOOL	FALSE
b4	BOOL	FALSE
b5	BOOL	FALSE
b6	BOOL	FALSE
b7	BOOL	FALSE



ST:

Expression	Type	Value
a	BYTE	7
b0	BOOL	TRUE
b1	BOOL	TRUE
b2	BOOL	TRUE
b3	BOOL	FALSE
b4	BOOL	FALSE
b5	BOOL	FALSE
b6	BOOL	FALSE
b7	BOOL	FALSE

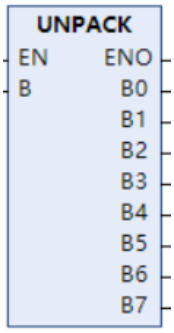


3.5. 21 UNPACK

This instruction unpacks a byte in B into bits and outputs them from the right side.

■ Instruction format

This instruction unpacks a byte in B into bits and outputs them from the right side.

Instruction	Name	FB/FC	LD Expression	ST Expression
UNPACK	Bit unpacking	FC		UNPACK(B:= , B0=> , B1=> , B2=> , B3=> , B4=> , B5=> , B6=> ,B7=>);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
B	Data source	BYTE	-	0	Operated data

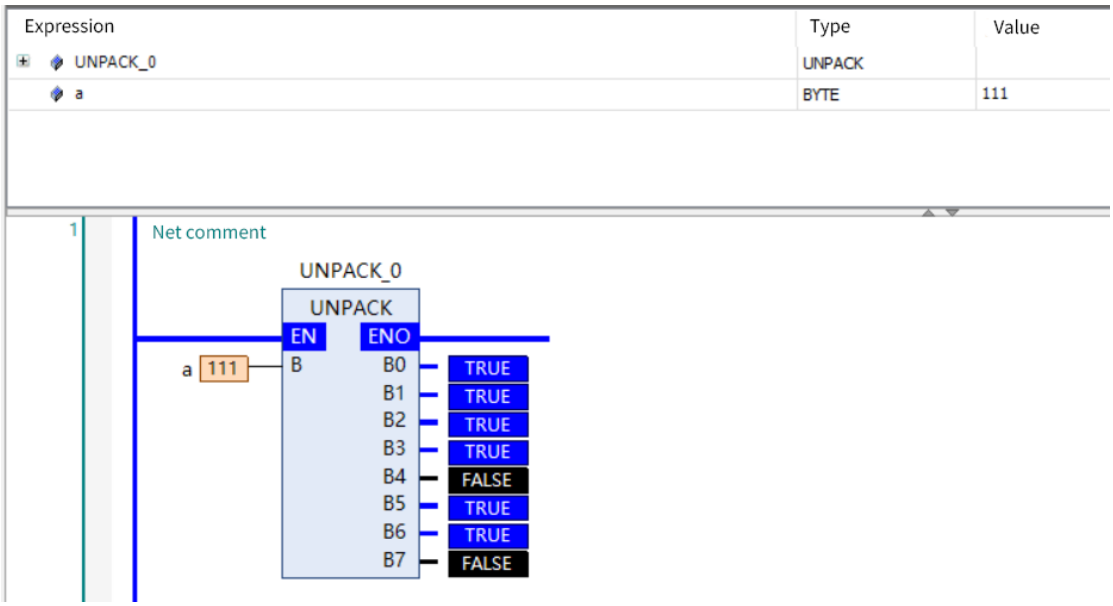
Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
B0-B7	Output target	BOOL	-	0	Bit status after data unpacking

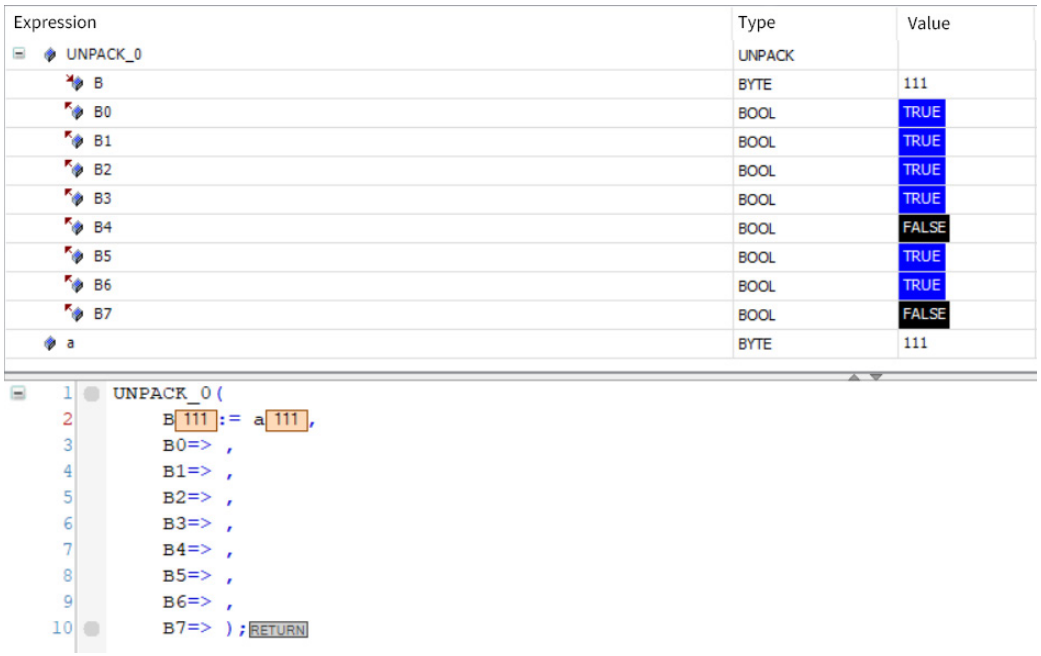
	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
B	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
B0-B7	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

■ Program example

LD:



ST:



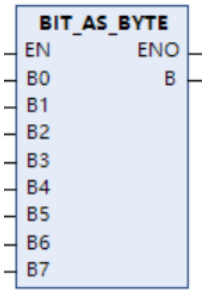
Expression	Type	Value
UNPACK_0	UNPACK	
B	BYTE	111
B0	BOOL	TRUE
B1	BOOL	TRUE
B2	BOOL	TRUE
B3	BOOL	TRUE
B4	BOOL	FALSE
B5	BOOL	TRUE
B6	BOOL	TRUE
B7	BOOL	FALSE
a	BYTE	111

```
1  UNPACK_0(
2      B 111 := a 111,
3      B0=> ,
4      B1=> ,
5      B2=> ,
6      B3=> ,
7      B4=> ,
8      B5=> ,
9      B6=> ,
10     B7=> );RETURN
```

3.5. 22 BIT_AS_BYTE

This instruction converts eight input values of the Bool type into values of the Byte type, and outputs them from the right side.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
BIT_AS_BYTE	8-bit packing into byte	FC		BIT_AS_BYTE(B0:= , B1:= , B2:= , B3:= , B4:= , B5:= , B6:= , B7:= , B=>);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
B0	Bit 0	BOOL	[TRUE, FALSE]	FALSE	-
B1	Bit 1	BOOL	[TRUE, FALSE]	FALSE	-
B2	Bit 2	BOOL	[TRUE, FALSE]	FALSE	-
B3	Bit 3	BOOL	[TRUE, FALSE]	FALSE	-
B4	Bit 4	BOOL	[TRUE, FALSE]	FALSE	-
B5	Bit 5	BOOL	[TRUE, FALSE]	FALSE	-
B6	Bit 6	BOOL	[TRUE, FALSE]	FALSE	-
B7	Bit 7	BOOL	[TRUE, FALSE]	FALSE	-

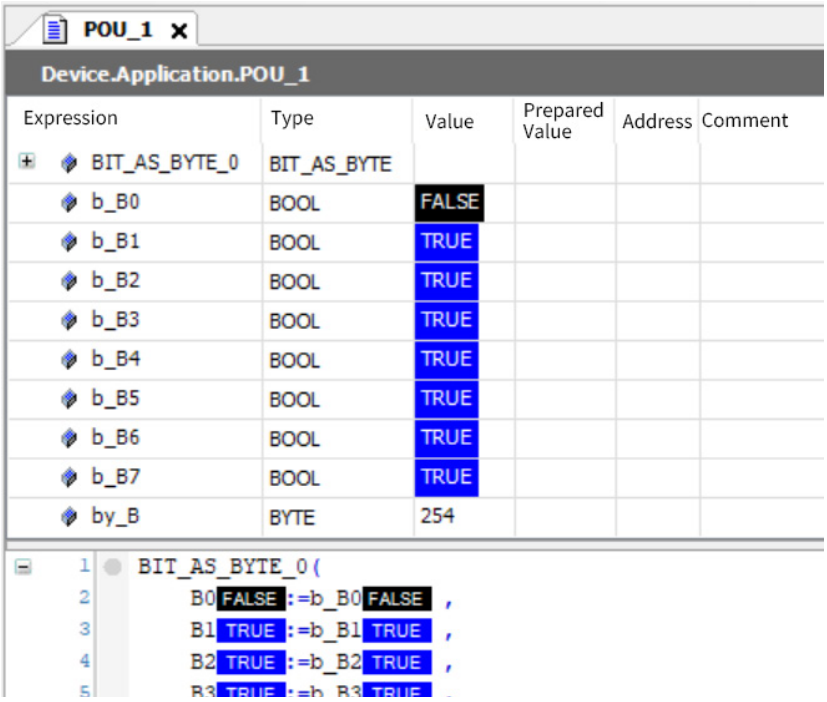
Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
B	Return value	BYTE	0 to 255	0	Conversion result

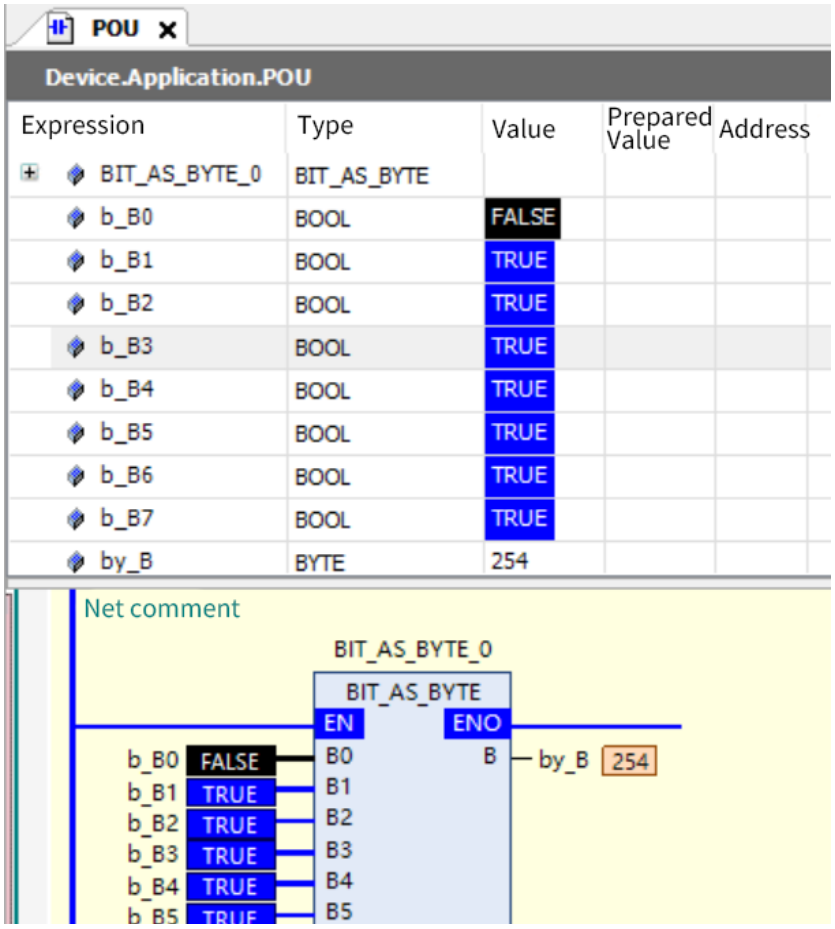
	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
B0-B7	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
B	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

■ Program example

ST



LD



■ Precautions

1. An error is reported when the input type is not BOOL.

2. The output range is 0 to 255.

3.5. 23 BYTE_AS_BIT

This instruction converts eight input values of the Byte type into values of the Bool type, and outputs them.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
BYTE_AS_BIT	Byte unpacking to bits	FC	BYTE_AS_BIT EN ENO B B0 B1 B2 B3 B4 B5 B6 B7	BYTE_AS_BIT(B:= , B0=> , B1=> , B2=> , B3=> , B4=> , B5=> , B6=> , B7=>);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
B	Return value	BYTE	0 to 255	0	Conversion result

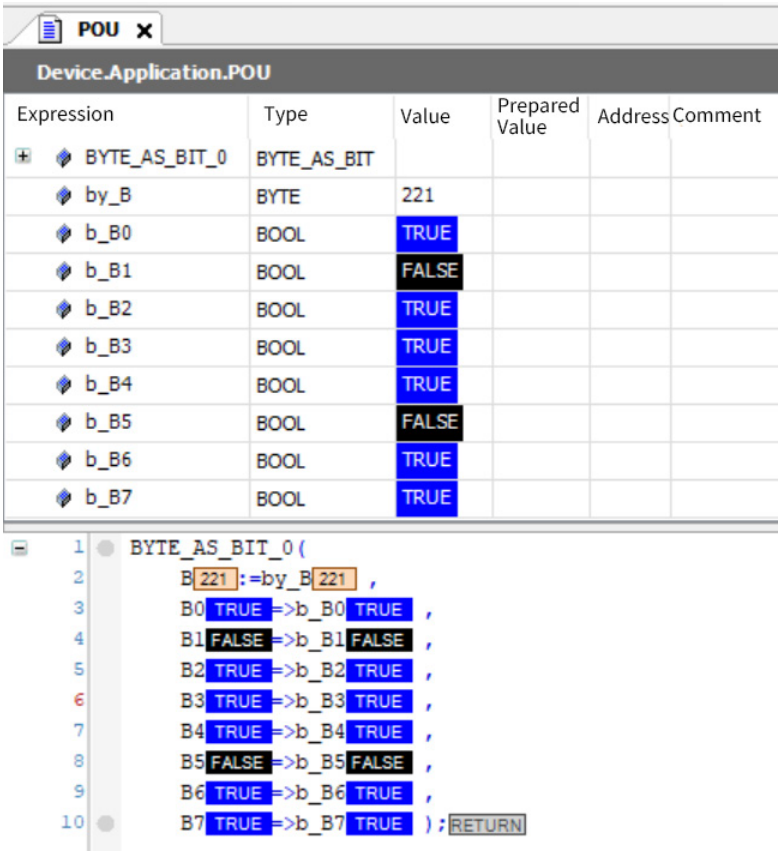
Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
B0	Bit 0	BOOL	[TRUE, FALSE]	FALSE	-
B1	Bit 1	BOOL	[TRUE, FALSE]	FALSE	-
B2	Bit 2	BOOL	[TRUE, FALSE]	FALSE	-
B3	Bit 3	BOOL	[TRUE, FALSE]	FALSE	-
B4	Bit 4	BOOL	[TRUE, FALSE]	FALSE	-
B5	Bit 5	BOOL	[TRUE, FALSE]	FALSE	-
B6	Bit 6	BOOL	[TRUE, FALSE]	FALSE	-
B7	Bit 7	BOOL	[TRUE, FALSE]	FALSE	-

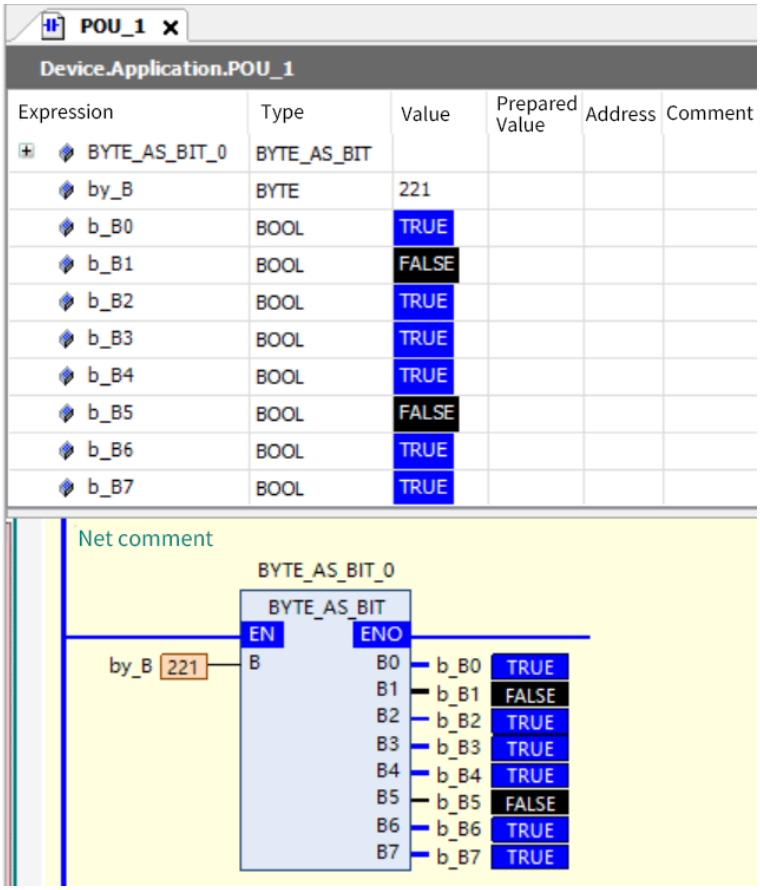
	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
B	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
B0-B7	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

■ Program example

ST



LD



■ Precautions

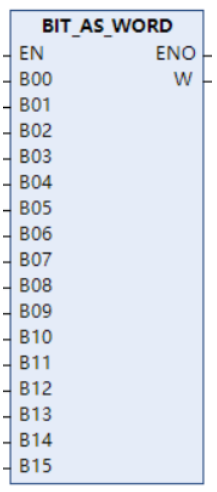
1. An error is reported when the input type is not BOOL.

2. The output range is 0 to 255.

3.5. 24 BIT_AS_WORD

This instruction packs 16 bits of the input data into a word and then outputs it from the output side.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
BIT_AS_WORD	16-bit packing into word	FC		POU.BIT_AS_WORD(B00:= , B01:= , B02:= , B03:= , B04:= , B05:= , B06:= , B07:= , B08:= , B09:= , B10:= , B11:= , B12:= , B13:= , B14:= , B15:= , W=> a);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
B00-B15	Data source	BOOL	-	0	Each bit to pack

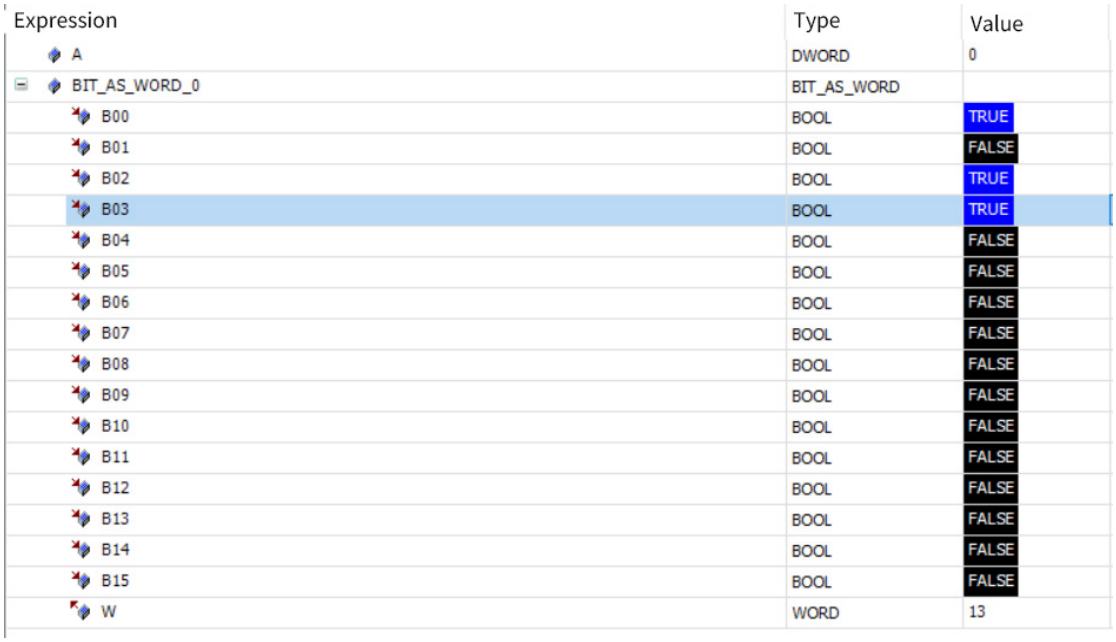
Output variables

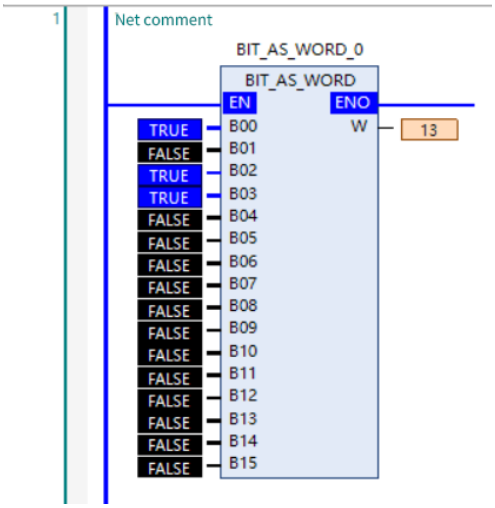
Output Variable	Name	Data Type	Value Range	Initial Value	Description
W	Output target	WORD	-	0	Data after unpacking

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
B00-B15	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
W	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

■ Program example

LD:





ST:

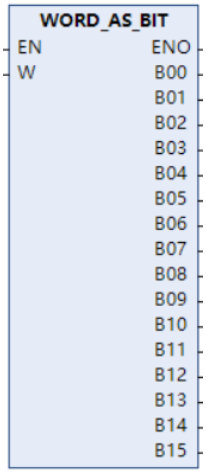
Expression	Type	Value
B00	BOOL	FALSE
B01	BOOL	FALSE
B02	BOOL	FALSE
B03	BOOL	FALSE
B04	BOOL	FALSE
B05	BOOL	FALSE
B06	BOOL	TRUE
B07	BOOL	TRUE
B08	BOOL	TRUE
B09	BOOL	FALSE
B10	BOOL	FALSE
B11	BOOL	TRUE
B12	BOOL	FALSE
B13	BOOL	FALSE
B14	BOOL	FALSE
B15	BOOL	FALSE
a	WORD	51

```
POU.BIT_AS_WORD_0(
    B00:= ,
    B01:= ,
    B02:= ,
    B03:= ,
    B04:= ,
    B05:= ,
    B06:= ,
    B07:= ,
    B08:= ,
    B09:= ,
    B10:= ,
    B11:= ,
    B12:= ,
    B13:= ,
    B14:= ,
    B15:= ,
    W 51 => a 51 );RETURN
```

3.5. 25 WORD_AS_BIT

This instruction unpacks a word in W into bits and outputs them from the right side.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
WORD_AS_BIT	Word unpacking to bits	FC		WORD_AS_BIT(W:= a, B00=> , B01=>, B02=> , B03=> , B04=> , B05=> , B06=> , B07=> , B08=>, B09=> , B10=>, B11=> , B12=> , B13=> , B14=>, B15=>);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
W	Output target	WORD	-	0	Data to unpack

Output variables

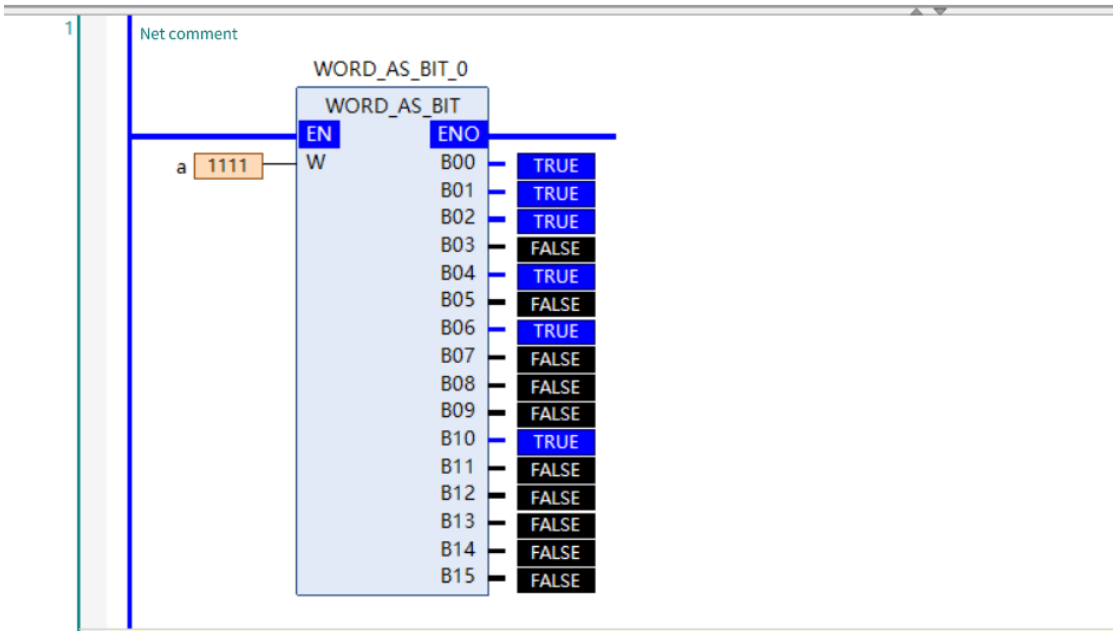
Output Variable	Name	Data Type	Value Range	Initial Value	Description
B00-B15	Status display	BOOL	-	0	Status of each bit after unpacking

	Bool-ean	Bit String				Integer								Real Num-ber		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
W	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
B00-B15	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

■ Program example

LD:

Expression	Type	Value
WORD_AS_BIT_0	WORD_AS_BIT	
a	WORD	1111



ST:

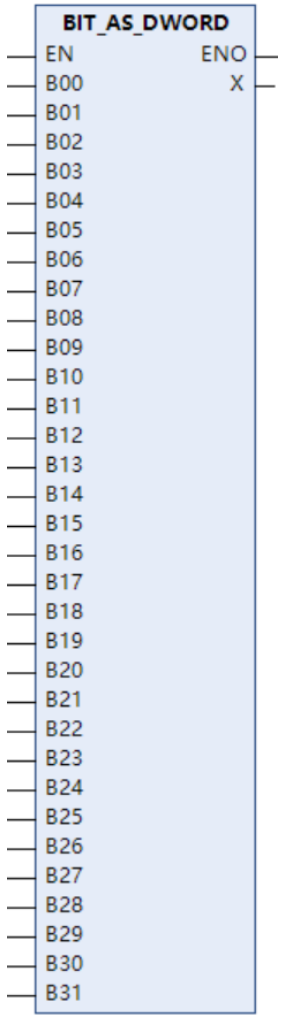
Expression	Type	Value
WORD_AS_BIT_0	WORD_AS_BIT	
W	WORD	1111
B00	BOOL	TRUE
B01	BOOL	TRUE
B02	BOOL	TRUE
B03	BOOL	FALSE
B04	BOOL	TRUE
B05	BOOL	FALSE
B06	BOOL	TRUE
B07	BOOL	FALSE
B08	BOOL	FALSE
B09	BOOL	FALSE
B10	BOOL	TRUE
B11	BOOL	FALSE
B12	BOOL	FALSE
B13	BOOL	FALSE
B14	BOOL	FALSE
B15	BOOL	FALSE
a	WORD	1111

```
WORD_AS_BIT_0(
    W 1111 := a 1111 ,
    B00=> ,
    B01=>,
    B02=> ,
    B03=> ,
    B04=> ,
    B05=> ,
    B06=> ,
    B07=> ,
    B08=>,
    B09=> ,
    B10=>,
    B11=> ,
    B12=> ,
    B13=> ,
    B14=>,
    B15=> );RETURN
```

3.5. 26 BIT_AS_DWORD

This instruction converts 32 input values of the Bool type into values of the Dword type, and outputs them from the right side.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
BIT_AS_DWORD	32-bit packing into doubleword	FC		BIT_AS_DWORD(B00:= , B01:= , B02:= , B03:= , B04:= , B05:= , B06:= , B07:= , B08:= , B09:= , B10:= , B11:= , B12:= , B13:= , B14:= , B15:= , B16:= , B17:= , B18:= , B19:= , B20:= , B21:= , B22:= , B23:= , B24:= , B25:= , B26:= , B27:= , B28:= , B29:= , B30:= , B31:= , X=>);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
B00-B31	BOOL value to convert	BOOL	[TRUE, FALSE]	FALSE	Status of each bit in X (binary mode)

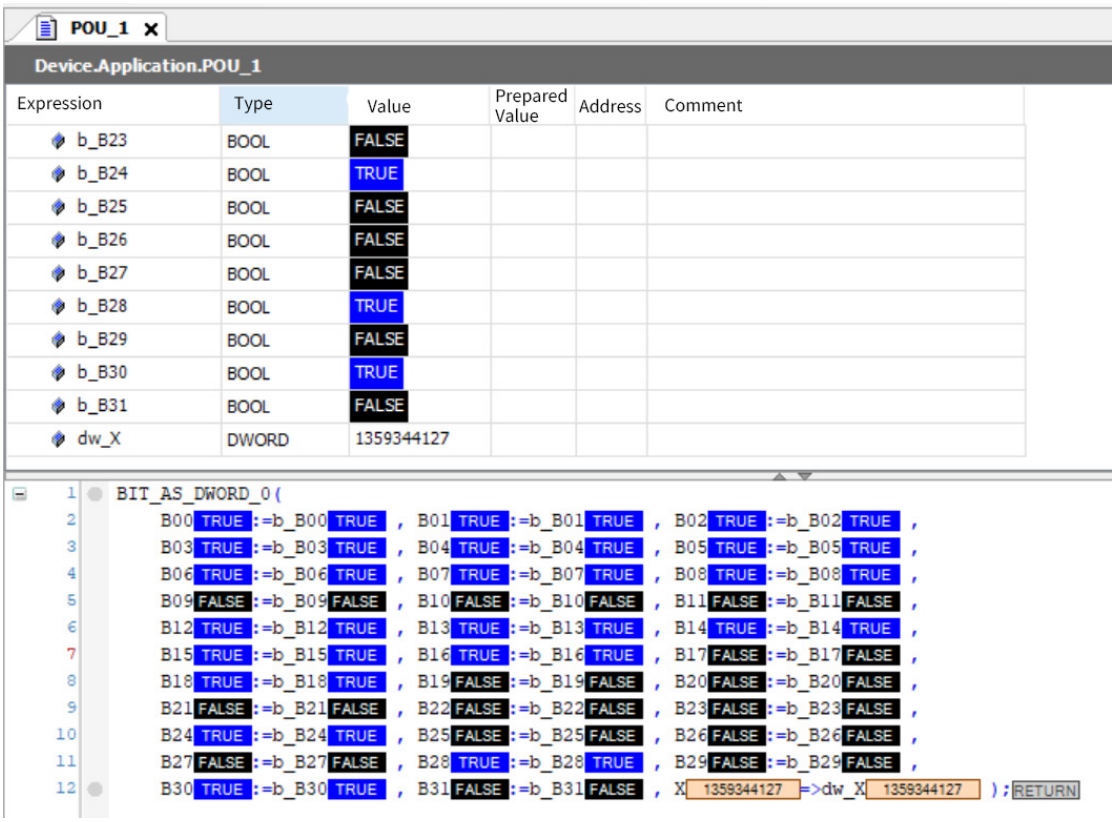
Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
X	Return value	DWORD	0-FFFFFFFF	0	Conversion result

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
B00-B31	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
W	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

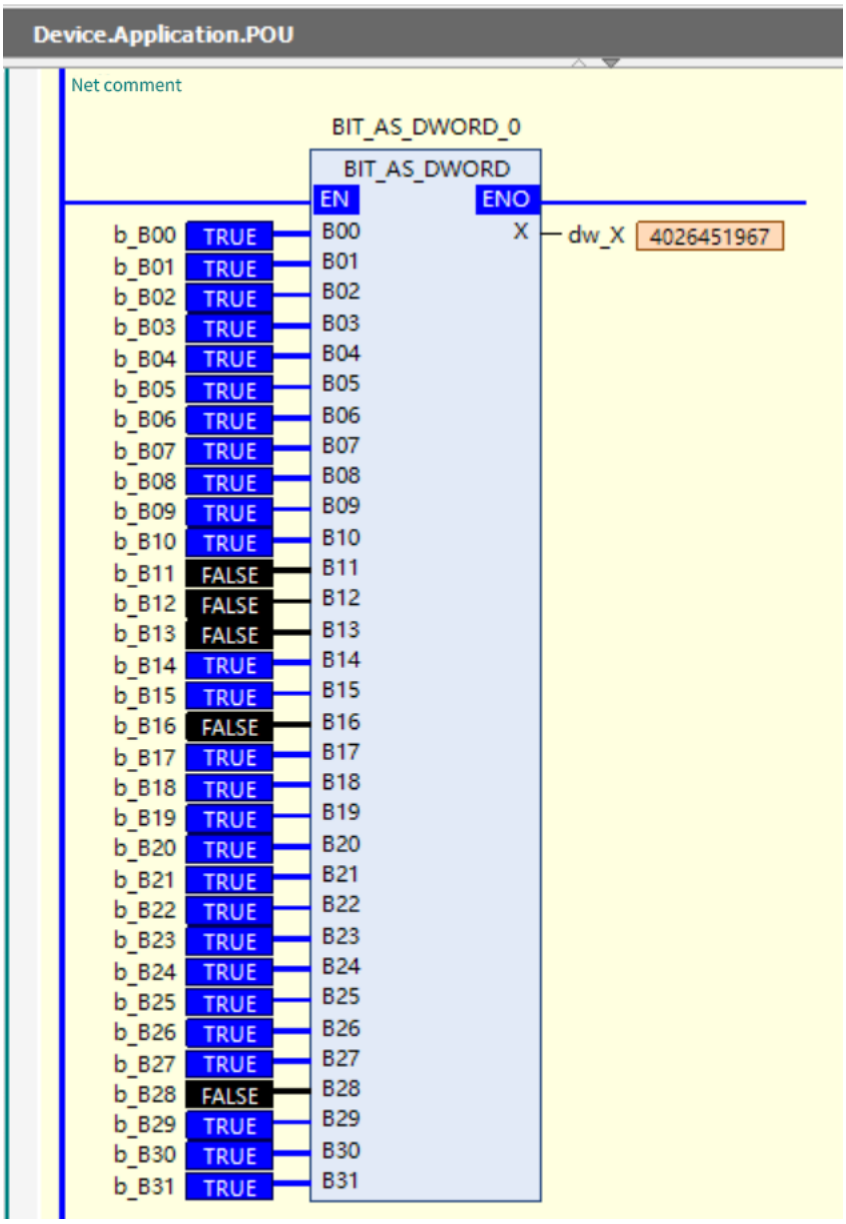
■ Program example

ST



LD

POU



■ Precautions

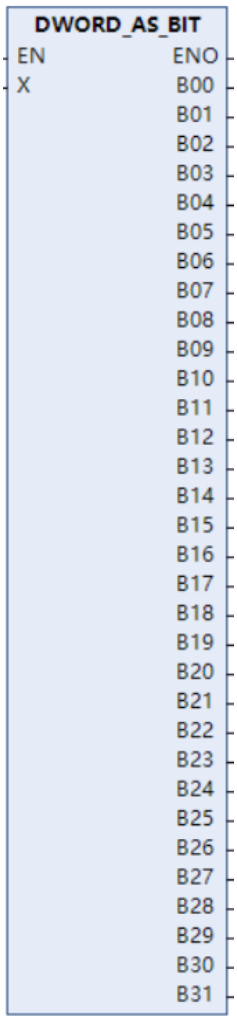
· An error is reported when the input type is not BOOL.

· The output range is 0 to 4294967295.

3.5. 27 DWORD_AS_BIT

This instruction unpacks a specified doubleword into bits and outputs them from the right side.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
DWORD_AS_BIT	Doubleword unpacking to bits	FC		DWORD_AS_BIT(X:=A , B00=> , B01=> , B02=> , B03=> , B04=> , B05=> , B06=> , B07=> , B08=> , B09=> , B10=> , B11=> , B12=> , B13=> , B14=> , B15=> , B16=> , B17=> , B18=> , B19=> , B20=> , B21=> , B22=> , B23=> , B24=> , B25=> , B26=> , B27=> , B28=> , B29=> , B30=> , B31=>);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
X	Return value	DWORD	0-FFFFFFFF	0	Data to unpack

Output variables

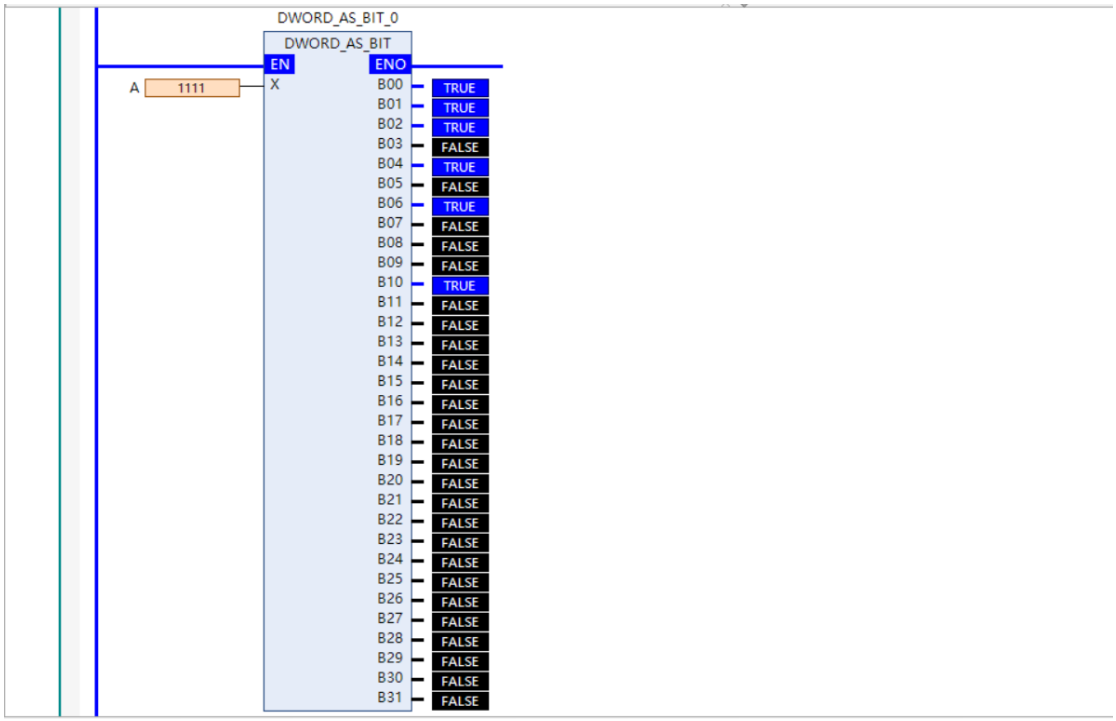
Output Variable	Name	Data Type	Value Range	Initial Value	Description
B00-B31	Output pin	BOOL	[TRUE, FALSE]	FALSE	Status of each bit in X (binary mode)

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
X	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
B00-B15	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

■ Program example

LD:

Expression	Type	Value
A	DWORD	1111
DWORD_AS_BIT_0	DWORD_AS_BIT	



ST:

Expression	Type	Value
A	DWORD	1111
DWORD_AS_BIT_0	DWORD_AS_BIT	

```
1  DWORD_AS_BIT_0(
2      X 1111 :=A 1111 ,
3      B00=> ,
4      B01=> ,
5      B02=> ,
6      B03=> ,
7      B04=> ,
8      B05=> ,
9      B06=> ,
10     B07=> ,
11     B08=> ,
12     B09=> ,
13     B10=> ,
14     B11=> ,
15     B12=> ,
16     B13=> ,
17     B14=> ,
18     B15=> ,
19     B16=> ,
20     B17=> ,
21     B18=> ,
22     B19=> ,
23     B20=> ,
24     B21=> ,
25     B22=> ,
```

3.5. 28 AryAnd, AryOr, AryXor, and AryXorN

These instructions calculate each bit in Boolean variables or individual bits of each element between arrays.

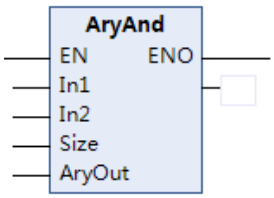
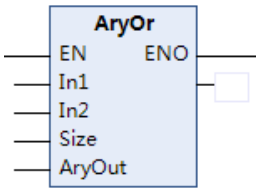
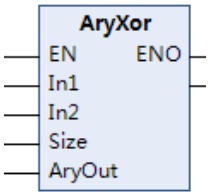
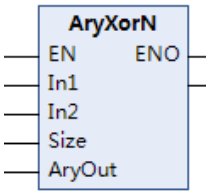
AryAnd: Logical AND

AryOr: Logical OR

AryXor: Exclusive OR

AryXorN: Exclusive NOR

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
AryAnd	Array logical AND	FC		AryAnd (In1:=,In2:=,Size:=, AryOut:=);
AryOr	Array logical OR	FC		AryOr (In1:=,In2:=, Size:=,AryOut:=);
AryXor	Array logical exclusive OR	FC		AryXor (In1:=,In2:=, Size:=,AryOut:=);
AryXorN	Array logical exclusive NOR	FC		AryXorN (In1:=,In2:=, Size:=,AryOut:=);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
In1[], In2[] (arrays)	Array to process	-	Depends on data types	-	Array to process
Size	Number of elements	UINT	Depends on data types	1	Number of elements to process

In-out variables

In-Out Variable	Name	Data Type	Value Range	Initial Value	Description

AryOut[] (array)	Processing result array	-	Depends on data types	-	Processing result array

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In1[]	√	√	√	√	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
In2[]	Array with the same data type as In1[]																			
Size	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-
AryOut	Array with the same data type as In1[]																			

■ Function

These instructions process corresponding bits of the first Size elements in In1[] and In2[] arrays. The processing results are stored in corresponding elements of AryOut[].

Therefore, the data types are the same for In1[], In2[], and AryOut[].

The relationships between input and output variables are given in the following tables.

AryAnd

When both bits are TRUE, the processing result is TRUE. Otherwise, the processing result is FALSE.

Bit of Element in In1[]	Bit of Element in In2[]	Bit of AryOut[]
FALSE	FALSE	FALSE
FALSE	TRUE	FALSE
TRUE	FALSE	FALSE
TRUE	TRUE	TRUE

AryOr

When both bits are FALSE, the processing result is FALSE. Otherwise, the processing result is TRUE.

Bit of Element in In1[]	Bit of Element in In2[]	Bit of AryOut[]
FALSE	FALSE	FALSE
FALSE	TRUE	TRUE
TRUE	FALSE	TRUE
TRUE	TRUE	TRUE

AryXor

When both bits are the same, the processing result is FALSE. When one bit is TRUE and the other is FALSE, then the processing result is TRUE.

Bit of Element in In1[]	Bit of Element in In2[]	Bit of AryOut[]
FALSE	FALSE	FALSE
FALSE	TRUE	TRUE
TRUE	FALSE	TRUE
TRUE	TRUE	FALSE

AryXorN

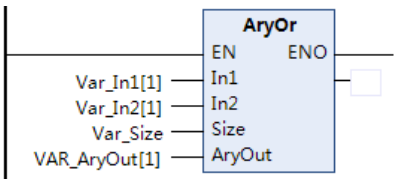
When both bits are the same, the processing result is TRUE. When one bit is TRUE and the other is FALSE, then the processing result is FALSE.

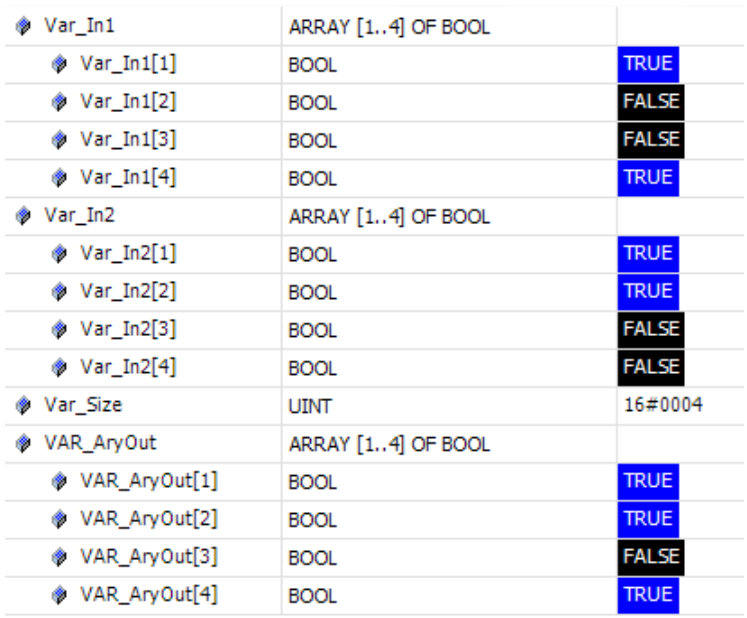
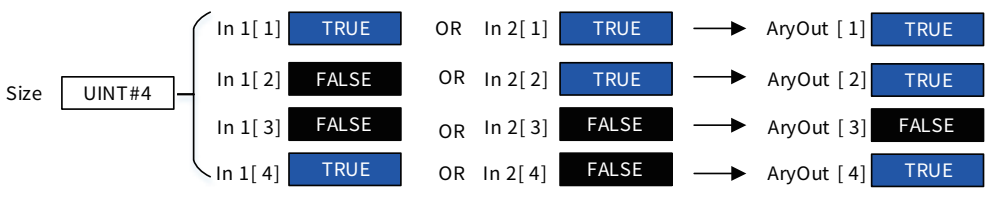
Bit of Element in In1[]	Bit of Element in In2[]	Bit of AryOut[]
FALSE	FALSE	TRUE
FALSE	TRUE	FALSE
TRUE	FALSE	FALSE
TRUE	TRUE	TRUE

■ Program example

The following example shows the AryOr instruction when Size is UINT#4.

	LD	ST
Defined variable		

Program		`AryOr(`
		`  In1:=Var_In1[1],`
		`  In2:=Var_In2[1],`
		`  Size:=Var_Size,`
		`  AryOut:=VAR_AryOut[1],`
		`);`

Running result		
Work principle		

■ Precautions

· The data types of In1[], In2[], and AryOut[] must be the same. · If they are different, an error occurs

during compilation.

· When the value of Size exceeds the In1[], In2[], or AryOut[] array, the return value is FALSE and AryOut does not change.

· When the value of Size is 0, AryOut[] does not change and the return value is TRUE.

3. 6 Data Shift Instructions

3.6. 1 Instruction List

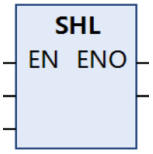
Instruction Category	Name	FB/FC	Function
Shift instructions	SHL	FC	Left shift
	SHR	FC	Right shift
	ArySHL	FC	Array N-element left shift
	ArySHR	FC	Array N-element right shift
	AryShiftReg	FC	Left shift register
	AryShiftRegLR	FC	Reversible shift register
	ROL	FC	Rotate left
	ROR	FC	Rotate right
	RCL	FC	Rotate left with carry
	RCR	FC	Rotate right with carry
	SFTL	FC	Bit shift left
	SFTR	FC	Bit shift right
	WSFL	FC	Word shift left
	WSFR	FC	Word shift right
	SFRD	FC	Shift read in FIFO mode
	SFWR	FC	Shift write in FIFO mode
	NSHLC	FC	Shift N-bits left with carry
	NSHRC	FC	Shift N-bits right with carry

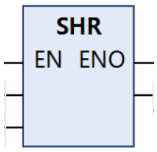
3.6. 2 SHL and SHR

SHL: Shifts an input value by one or more bits to the left, with zeros automatically inserted into the right side.

SHR: Shifts an input value by one or more bits to the right, with zeros automatically inserted into the left side.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
SHL	Left shift	FC		:=SHL(,);

Instruction	Name	FB/FC	LD Expression	ST Expression
SHR	Right shift	FC		:=SHR(,);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
In1	Data source	Integer	Depends on data types	0	Data source
In2	Number of bits to shift	Integer	Depends on data types	0	Number of bits to shift

Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
In1	Shift result	Integer	Depends on data types	0	Shift result

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In1	-	-	-	-	-	√	√	√	√	√	√	√	√	-	-	-	-	-	-	-
In2	-	-	-	-	-	√	√	√	√	√	√	√	√	-	-	-	-	-	-	-
Out	-	-	-	-	-	√	√	√	√	√	√	√	√	-	-	-	-	-	-	-

■ Program example

1. SHL

When the input signal "Source data" is 2#0000000000000001 and "Number of bits to shift" is 2, the output signal "Result data" is 2#0000000000000100, obtained by 2-bit shift to the left.

ST:

Expression	Type	Value	Prepared Value	Address
Result data	INT	2#0000000000000100		
Source data	INT	2#0000000000000001		
Number of bits to shift	INT	2#0000000000000010		

LD:

Expression	Type	Value	Prepared Value	Address
Result data	INT	2#0000000000000001		
Source data	INT	2#0000000000000100		
Number of bits to shift	INT	2#0000000000000010		

2. SHR

When the input signal "Source data" is 8 (2#0000000000001000) and "Number of bits to shift" is 2, the output signal "Result data" is 2#0000000000000010, obtained by 2-bit shift to the right.

ST:

Expression	Type	Value	Prepared Value	Address
Result data	INT	2#0000000000000010		
Source data	INT	2#0000000000001000		
Number of bits to shift	INT	2#0000000000000010		

LD:

Expression	Type	Value	Prepared Value	Address
Result data	INT	2#0000000000001000		
Source data	INT	2#0000000000000010		
Number of bits to shift	INT	2#0000000000000010		

■ Precautions

· The data can only be of the integer type. If the data is of the floating-point type, an error occurs.

3.6. 3 ArySHL and ArySHR

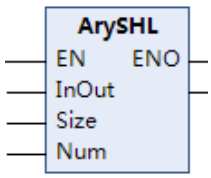
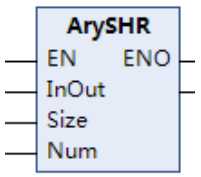
These instructions shift multiple array elements.

ArySHL: Shifts multiple array elements to the left (toward the higher elements).

ArySHR: Shifts multiple array elements to the right (toward the lower elements).

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression

ArySHL	Array N-element left shift	FC		ArySHL(InOut :=, Size := , Num:=);
ArySHR	Array N-element right shift	FC		ArySHR(InOut :=, Size :=, Num :=);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
Size	Number of elements in the shift register array	UINT	Depends on data types	1	Number of elements in the shift register array
Num	Number of bits to shift	UINT	Depends on data types	1	Number of bits to shift

In-out variables

In-Out Variable	Name	Data Type	Value Range	Initial Value	Description
InOut[] Array	Shift register array	-	Depends on data types	-	Shift register array

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
InOut[]	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√
	Arrays of structures can also be specified.																			
Size	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-
Num	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-

■ Function

These instructions shift the lower/upper Size elements in shift register array InOut[] by Num elements.

Elements that are shifted out of the array are discarded.

The default initial value for the data type of InOut[] is stored in the empty elements.

The default values for the data types are listed in the following table.

Data Type	Initial Value
BOOL	FALSE
BYTE, WORD, DWORD, or LWORD	16#0

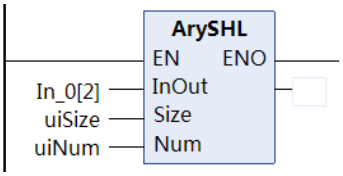
Data Type	Initial Value
USINT, UINT, UDINT, ULINT, SINT, INT, DINT, LINT, REAL, or LREA	0
TIME	T#0ms
DATE	D#1970-1-1
TOD	TOD#0:0:0
DT	DT#1970-1-1-0:0:0
STRING	''

■ Program example

<table>
<tr><th></th><th>LD</th><th>ST</th></tr>
<tr><td>Defined variable</td><td colspan="2">

```
VAR
    xStartButton: BOOL;
    In_0        : ARRAY [0..9] OF INT := [0,1,2,3,4,5,6,7,8,9];
    uiSize      : UINT := 7;
    uiNum       : UINT := 2;
END_VAR
```

</td></tr>
<tr><td>Program</td><td>



</td><td>

```
ArySHL(InOut := In_0[2],
    Size := uiSize,
    Num := uiNum,
    );
```

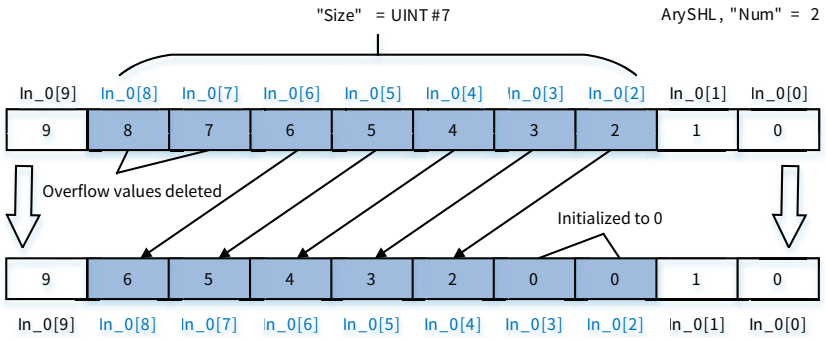
</td></tr>
<tr><td>Running result</td><td>

In_0	ARRAY [0..9] OF INT	
In_0[0]	INT	0
In_0[1]	INT	1
In_0[2]	INT	0
In_0[3]	INT	0
In_0[4]	INT	2
In_0[5]	INT	3
In_0[6]	INT	4
In_0[7]	INT	5
In_0[8]	INT	6
In_0[9]	INT	9

</td><td>

In_0	ARRAY [0..9] OF INT	
In_0[0]	INT	0
In_0[1]	INT	1
In_0[2]	INT	0
In_0[3]	INT	0
In_0[4]	INT	2
In_0[5]	INT	3
In_0[6]	INT	4
In_0[7]	INT	5
In_0[8]	INT	6
In_0[9]	INT	9

</td></tr>
<tr><td>Work principle</td><td colspan="2">



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

■ Precautions

· When the value of Size exceeds the InOut[] array, the return value is FALSE and InOut[] does not change.

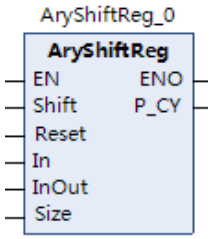
· When the value of Num or Size is 0, the shift operation is not performed, the return value is TRUE, and InOut[] does not change.

· When the value of Num is greater than Size, all values from InOut[0] to InOut[Size − 1] are initialized to 0.

3.6. 4 AryShiftReg

This instruction shifts a shift register one bit to the left and inserts the input value to the least-significant bit.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
AryShiftReg	Left shift register	FC		AryShiftReg_0(Shift :=, Reset :=, In :=, InOut :=, Size :=, P_CY =>,);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
Shift	Shift	BOOL	[FALSE, TRUE]	FALSE	Shift when the signal changes to TRUE
Reset	Reset	BOOL	[FALSE, TRUE]	FALSE	Reset when the signal changes to TRUE
In	Input value	BOOL	[FALSE, TRUE]	FALSE	Value to insert to the least-significant bit of InOut[]
Size	Number of elements in the array of bit strings	UINT	Depends on data types	0	Number of elements to use as a shift register in InOut[]

■ In-out variables

In-Out Variable	Name	Data Type	Value Range	Initial Value	Description
InOut[] (array)	Array of bit strings	-	Depends on data types	-	Array of bit strings

■ Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
P_CY	Carry flag	BOOL	Depends on data types	-	Value stored in the carry flag

	Bool-ean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Shift	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Reset	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
In	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
InOut[]	√	√	√	√	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Size	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-
P_CY	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

■ Function

This instruction shifts Size array elements in the array of bit strings InOut[] to the left by one bit (toward the most-significant bit) when Shift changes to TRUE. The shift operation starts from InOut[0].

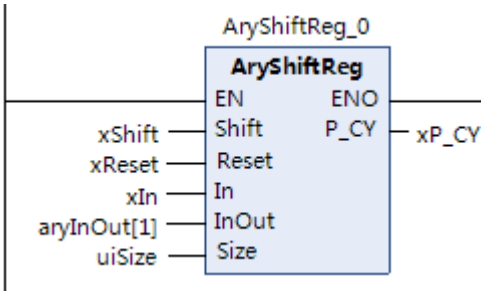
Input value In is inserted to the least-significant bit. The most-significant bit of the array of bit strings is output to the Carry (CY) Flag (P_CY).

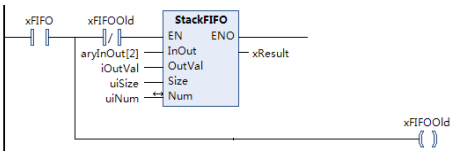
When Reset is TRUE, CY and all bits and carry bits in Size elements starting from InOut[0] are set to FALSE.

■ Program example

	LD	ST
Defined variable		
Program		
Running result		

```
VAR
    AryShiftReg_0   : AryShiftReg;
    xShift          : BOOL;
    xReset          : BOOL;
    xIn             : BOOL := FALSE;
    aryInOut        : ARRAY [1..4] OF BYTE := [2#10011001,2#10111001,2#11101101,2#10101101];
    uiSize          : UINT := 3;
    xP_CY           : BOOL;
END_VAR
```

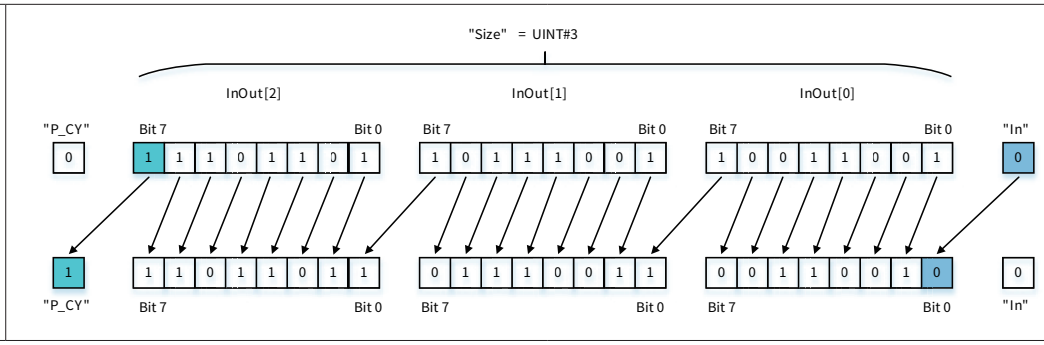





```
IF xFIFO AND NOT xFIFOOld  THEN

    xResult := StackFIFO(InOut := aryInOut[2],
            OutVal := iOutVal,
            Size := uiSize,
            Num := uiNum
            );
END_IF
```

Work principle



■ Precautions

· While Reset is TRUE, the register is not shifted even if Shift changes to TRUE.

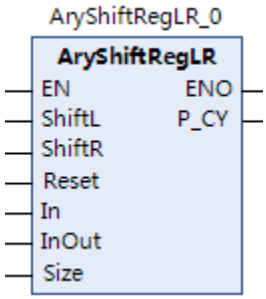
· When the value of Size is 0, InOut[] does not change.

· When the value of Size exceeds the InOut[] array, InOut[] does not change.

3.6. 5 AryShiftRegLR

This instruction shifts a shift register one bit to the left or right and inserts the input value to the least-significant bit or most-significant bit.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
AryShiftRegLR	Reversible shift register	FB		AryShiftRegLR_0(ShiftL :=,ShiftR :=, Reset :=, In :=, InOut :=,Size :=,P_CY =>,);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
ShiftL	Left shift	BOOL	[FALSE, TRUE]	FALSE	Left shift when the signal changes to TRUE
ShiftR	Right shift	BOOL	[FALSE, TRUE]	FALSE	Right shift when the signal changes to TRUE
Reset	Reset	BOOL	[FALSE, TRUE]	FALSE	Reset when the signal changes to TRUE
In	Input value	BOOL	[FALSE, TRUE]	FALSE	Value to insert to the least-significant bit or most-significant bit of InOut[]

Input Variable	Name	Data Type	Value Range	Initial Value	Description
Size	Number of elements in the array of bit strings	UINT	Depends on data types	0	Number of elements to use as a shift register in InOut[]

■ In-out variables

In-Out Variable	Name	Data Type	Value Range	Initial Value	Description
InOut[] (array)	Array of bit strings	-	Depends on data types	-	Array of bit strings

■ Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
P_CY	Carry flag	BOOL	Depends on data types	-	Value stored in the carry flag

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
ShiftL	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
ShiftR	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Reset	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
In	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
InOut[]	√	√	√	√	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Size	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-
P_CY	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

■ Function

1. This instruction shifts Size array elements in the array of bit strings InOut[] to the left by one bit (toward the most-significant bit) when ShiftL changes to TRUE. The shift operation starts from InOut[0]. Input value In is inserted to the least-significant bit. The most-significant bit of the array of bit strings is output to the Carry (CY) Flag (P_CY).

2. This instruction shifts Size array elements in the array of bit strings InOut[] to the right by one bit (toward the least-significant bit) when ShiftR changes to TRUE. The shift operation starts from InOut[0]. Input value In is inserted to the most-significant bit. The least-significant bit of the array of bit strings is output to the Carry (CY) Flag (P_CY).

3. When Reset is TRUE, CY and all bits and carry bits in Size elements starting from InOut[0] are set to FALSE.

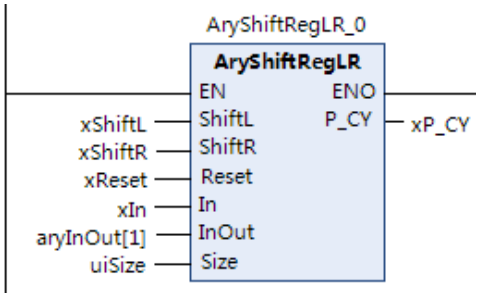
■ Program example

	LD	ST

Defined variable

```
VAR
    AryShiftRegLR_0 : AryShiftRegLR;
    xShiftL         : BOOL;
    xShiftR         : BOOL;
    xReset          : BOOL;
    xIn             : BOOL := FALSE;
    aryInOut        : ARRAY [1..4] OF BYTE := [2#10011001,2#10111001,2#11101101,2#10101101];
    uiSize          : UINT := 3;
    xP_CY           : BOOL;
END_VAR
```

Program




```
AryShiftRegLR_0(ShiftL := xShiftL,
        ShiftR := xShiftR,
        Reset := xReset,
        In := xIn,
        InOut := aryInOut[1],
        Size := uiSize,
        P_CY => xP_CY
        );
```

Running result

AryShiftRegLR_0	AryShiftRegLR	
xShiftL	BOOL	FALSE
xShiftR	BOOL	TRUE
xReset	BOOL	FALSE
xIn	BOOL	FALSE
aryInOut	ARRAY [1..4] OF BYTE	
aryInOut[1]	BYTE	2#11001100
aryInOut[2]	BYTE	2#11011100
aryInOut[3]	BYTE	2#01110110
aryInOut[4]	BYTE	2#10101101
uiSize	UINT	2#0000000000000011
xP_CY	BOOL	TRUE

■ Precautions

· While Reset is TRUE, the register is not shifted even if ShiftL or ShiftR changes to TRUE.

· When ShiftL and ShiftR change to TRUE concurrently, the register is not shifted.

· When the value of Size is 0, InOut[] does not change.

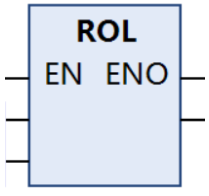
· When the value of Size exceeds the InOut[] array, InOut[] does not change.

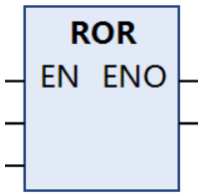
3.6. 6 ROL and ROR

■ Instruction description

ROL: Rotates the bit string of the input value to the left (toward the least-significant bit).

ROR: Rotates the bit string of the input value to the right (toward the most-significant bit).

Instruction	Name	FB/FC	LD Expression	ST Expression
ROL	Rotate left	FC		:= ROL(,);

Instruction	Name	FB/FC	LD Expression	ST Expression
ROR	Rotate right	FC		:= ROR(,);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
Int1	Data source	Integer	-	-	Data source
Int2	Number of bits to shift	Integer	-	-	Number of bits to shift

Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
OUT	Shift result	Integer	-	-	Shift result

	Bool-ean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In1	-	√	√	√	√	√	√	√	√	√	√	√	√	-	-	-	-	-	-	-
In2	-	√	√	√	√	√	√	√	√	√	√	√	√	-	-	-	-	-	-	-
Out	-	√	√	√	√	√	√	√	√	√	√	√	√	-	-	-	-	-	-	-

■ Program example

1. ROL

When the input signal "Source data" is 2#1100000000000000and "Number of bits to shift" is 1, the output signal "Result data" is 2#1000000000000001, obtained by 1-bit shift to the left.

ST:

Expression	Type	Value	Prepared Value	Address	Comment
Result data	INT	2#1000000000000001			
Number of rotation bits	INT	2#0000000000000001			
Source data	INT	2#1100000000000000			

LD:

Expression	Type	Value	Prepared Value	Address
Source data	INT	2#1100000000000000		
Result data	INT	2#1000000000000001		
Number of rotation bits	INT	2#0000000000000001		

2. ROR

When the input signal "Source data" is 2#0000000000000011 and "Number of bits to shift" is 1, the output signal "Result data" is 2#1000000000000001, obtained by 1-bit shift to the right.

ST:

Expression	Type	Value	Prepared Value	Address	Comment
Source data	INT	2#0000000000000011			
Result data	INT	2#1000000000000001			
Number of rotation bits	INT	2#0000000000000001			

LD:

Expression	Type	Value	Prepared Value	Address
Source data	INT	2#0000000000000011		
Result data	INT	2#1000000000000001		
Number of rotation bits	INT	2#0000000000000001		

■ Precautions

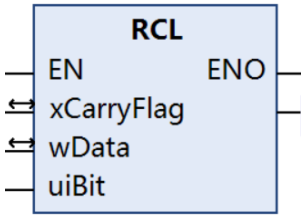
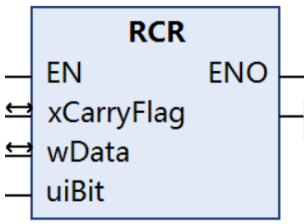
· This instruction supports the integer data type.

3.6. 7 RCL and RCR

■ Instruction description

RCL: Rotates the bit string of the operand to the left (toward the least-significant bit). When a carry occurs, the carry flag is set to ON.

RCR: Rotates the bit string of the operand to the right (toward the most-significant bit). When a carry occurs, the carry flag is set to ON.

Instruction	Name	FB/FC	LD Expression	ST Expression
RCL	Rotate left with carry	FC		RCL(xCarryFlag:=, wData:= , uiBit:=);
RCR	Rotate right with carry	FC		RCR(xCarryFlag:=, wData:= , uiBit:= cyclic);

■ Variable pin definition

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
XCarryFlag	Data carry flag	BOOL	-	-	Data carry flag
wData	Data source	WORD	-	-	Data source
uiBit	Number of rotation bits	UINT	1 to 1024	-	Number of rotation bits

	Bool-ean	Bit String				Integer								Real Num-ber		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
XCarryFlag	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
wData	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
uiBit	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-

■ Program example

1. RCL

When the input signal "Source data" is 2#1110000000000000 and "Number of bits to shift" is 1, shift is triggered once, the output signal "Result data" is 2#1100000000000001, obtained by 1-bit shift to the left, and XCarryFlag is ON.

ST

Expression	Type	Value	Prepared Value	Address	Comment
Source data	WORD	2#1100000000000000			
Number of rotation bits	UINT	2#0000000000000001			
Carry flag	BOOL	TRUE			
Execute	BOOL	FALSE			
R_TRIG_0	R_TRIG				

LD

Expression	Type	Value	Prepared Value
Source data	WORD	2#0000000000000110	
Number of rotation bits	UINT	2#0000000000000001	
Carry flag	BOOL	FALSE	
Execute	BOOL	TRUE	

2. ROR

When the input signal wData is 2#0000000000000011 and uiBit is 1, shift is triggered once, the output signal wData is 2#0000000000000001, one bit of the valid data is shifted to the right, and XCarryFlag is ON.

ST

Expression	Type		Value	Prepared Value	Address	Comment
R_TRIG_0		R_TRIG				
Execute		BOOL	FALSE			
Carry flag		BOOL	TRUE			
Source data		WORD	2#0000000000000001			
Number of rotation bits		UINT	2#0000000000000001			

LD

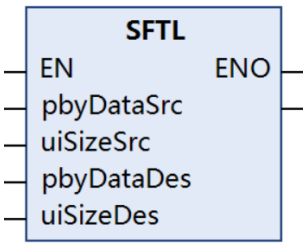
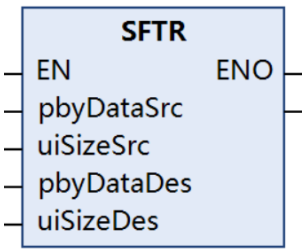
Expression	Type		Value	Prepared Value
Source data		WORD	2#0000000000000001	
Number of rotation bits		UINT	2#0000000000000001	
Carry flag		BOOL	TRUE	
Execute		BOOL	TRUE	

3.6. 8 SFTL and SFTR

■ Instruction description

SFTL: Copies a specified length of bit data from the source array to the destination array in a continuous leftward shift.

SFTR: Copies a specified length of bit data from the source array to the destination array in a continuous rightward shift.

Instruction	Name	FB/FC	LD Expression	ST Expression
SFTL	Bit shift left	FC		SFTL(pbyDataSrc:=ADR() , uiSizeSrc:= , pbyDataDes:= , uiSizeDes:=);
SFTR	Bit shift right	FC		SFTR(pbyDataSrc:= , uiSizeSrc:= , pbyDataDes:= , uiSizeDes:=);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
pbyDataSrc	Source data	POINTER_TO_BYTE	-	-	Source data
pbyDataDes	Target data	POINTER_TO_BYTE	-	-	Target data
uiSizeSrc	Number of data sources	BYTE	-	-	Number of data sources
uiSizeDes	Number of target data entries	BYTE	-	-	Number of target data entries

	Bool-ean	Bit String				Integer								Real Num-ber		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
pbyDataSrc	POINTER_TO_BYTE																			
pbyDataDes	POINTER_TO_BYTE																			
uiSizeSrc	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
uiSizeDes	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

■ Program example

SFTL

This instruction copies uiSizeSrc (number of array elements copied once) bits of the head address of pbyDataSrc (source array) to the low-bit region with the head address of pbyDataDes (destination array). Before the copy operation, uiSizeDes bits of pbyDataDes are shifted leftwards by uiSizeSrc bits.

The following figures show the results after two triggers.

ST

Expression	Type	Value	Prepared Value	Address	Comment
uiSizeSrc	BYTE	2			
uiSizeDes	BYTE	4			
Source data	ARRAY [0..1] OF BOOL				
Source data [0]	BOOL	TRUE			
Source data [1]	BOOL	TRUE			
Target data	ARRAY [0..10] OF BOOL				
Target data [0]	BOOL	TRUE			
Target data [1]	BOOL	TRUE			
Target data [2]	BOOL	TRUE			
Target data [3]	BOOL	TRUE			
Target data [4]	BOOL	FALSE			
Target data [5]	BOOL	FALSE			
Target data [6]	BOOL	FALSE			
Target data [7]	BOOL	FALSE			
Target data [8]	BOOL	FALSE			
Target data [9]	BOOL	FALSE			
Target data [10]	BOOL	FALSE			
Execute	BOOL	FALSE			
R_TRIG_0	R_TRIG				

LD

Expression	Type	Value
uiSizeSrc	UINT	2
uiSizeDes	UINT	4
Source data	ARRAY [0..1] OF BOOL	
Source data [0]	BOOL	TRUE
Source data [1]	BOOL	TRUE
Target data	ARRAY [0..10] OF BOOL	
Target data [0]	BOOL	TRUE
Target data [1]	BOOL	TRUE
Target data [2]	BOOL	TRUE
Target data [3]	BOOL	TRUE
Target data [4]	BOOL	FALSE
Target data [5]	BOOL	FALSE
Target data [6]	BOOL	FALSE
Target data [7]	BOOL	FALSE
Target data [8]	BOOL	FALSE
Target data [9]	BOOL	FALSE
Target data [10]	BOOL	FALSE
Execute	BOOL	FALSE

2. SFTR

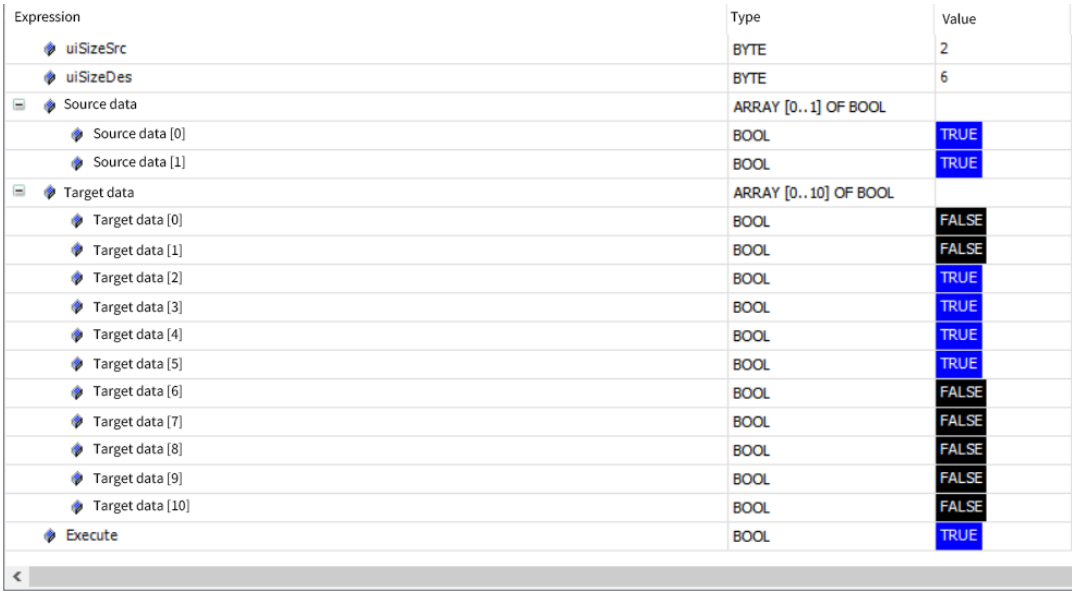
This instruction copies uiSizeSrc (number of array elements copied once) bits of the head address of pbyDataSrc (source array) to the high-bit region with the head address of pbyDataDes (destination array). Before the copy operation, uiSizeDes bits of pbyDataDes are shifted rightwards by uiSizeSrc bits.

The following figures show the results after two triggers.

ST

Expression	Type	Value	Prepared Value	Address	Comment
uiSizeSrc	BYTE	2			
uiSizeDes	BYTE	6			
Source data	ARRAY [0..1] OF BOOL				
Source data [0]	BOOL	TRUE			
Source data [1]	BOOL	TRUE			
Target data	ARRAY [0..10] OF BOOL				
Target data [0]	BOOL	FALSE			
Target data [1]	BOOL	FALSE			
Target data [2]	BOOL	FALSE			
Target data [3]	BOOL	FALSE			
Target data [4]	BOOL	TRUE			
Target data [5]	BOOL	TRUE			
Target data [6]	BOOL	FALSE			
Target data [7]	BOOL	FALSE			
Target data [8]	BOOL	FALSE			
Target data [9]	BOOL	FALSE			
Target data [10]	BOOL	FALSE			
Execute	BOOL	FALSE			
R_TRIG_0	R_TRIG				

LD



Expression	Type	Value
uiSizeSrc	BYTE	2
uiSizeDes	BYTE	6
Source data	ARRAY [0..1] OF BOOL	
Source data [0]	BOOL	TRUE
Source data [1]	BOOL	TRUE
Target data	ARRAY [0..10] OF BOOL	
Target data [0]	BOOL	FALSE
Target data [1]	BOOL	FALSE
Target data [2]	BOOL	TRUE
Target data [3]	BOOL	TRUE
Target data [4]	BOOL	TRUE
Target data [5]	BOOL	TRUE
Target data [6]	BOOL	FALSE
Target data [7]	BOOL	FALSE
Target data [8]	BOOL	FALSE
Target data [9]	BOOL	FALSE
Target data [10]	BOOL	FALSE
Execute	BOOL	TRUE

■ Precautions

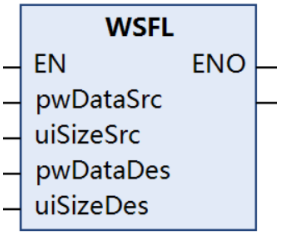
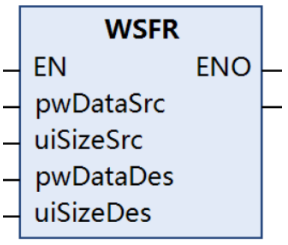
uiSizeSrc should not exceed uiSizeDes. Otherwise, the PLC stops unexpectedly.

3.6. 9 WSFL and WSFR

■ Instruction description

WSFL: Copies a specified length of word data from the source array to the destination array in a continuous leftward shift.

WSFR: Copies a specified length of word data from the source array to the destination array in a continuous rightward shift.

Instruction	Name	FB/FC	LD Expression	ST Expression
WSFL	Word shift left	FC		WSFL(pwDataSrc:= , uiSizeSrc:= , pwDataDes:= , uiSizeDes:=);
WSFR	Word shift right	FC		WSFR(pwDataSrc:= , uiSizeSrc:= , pwDataDes:= , uiSizeDes:=);

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
pbyDataSrc	Source data	POINTER_TO_INT	-	-	Source data
pbyDataDes	Target data	POINTER_TO_INT	-	-	Target data
uiSizeSrc	Number of source data entries	UINT	1 to 1024	-	Number of source data entries
uiSizeDes	Number of target data entries	UINT	1 to 1024	-	Number of target data entries

	Bool-ean	Bit String				Integer								Real Num-ber		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
pbyDataSrc	POINTER_TO_BYTE																			
pbyDataDes	POINTER_TO_BYTE																			
uiSizeSrc	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
uiSizeDes	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

■ Program example

1. SFL

Shift the first iSizeSrc (number of array elements copied) bits of pwDataSrc (source array) and the first uiSizeDes (number of target data entries) bits of pwDataDes (destination array) leftwards by uiSizeDes bits, and save the result to pwDataDes. In general, the execution is triggered by pulses.

The following figures show the results after three triggers.

ST

```
IF xDel AND NOT xDelOld  THEN

    xResult := StackDel(InOut := aryInOut[2],
            Size := uiSize,
            Num := uiNum,
            Offset := uiOffset,
            );
END_IF
```

LD

Expression	Type	Value
uiSizeSrc	UINT	2
uiSizeDes	UINT	6
Source data	ARRAY [0..1] OF INT	
Source data [0]	INT	11
Source data [1]	INT	22
Target data	ARRAY [0..10] OF INT	
Target data [0]	INT	11
Target data [1]	INT	22
Target data [2]	INT	11
Target data [3]	INT	22
Target data [4]	INT	11
Target data [5]	INT	22
Target data [6]	INT	0
Target data [7]	INT	0
Target data [8]	INT	0
Target data [9]	INT	0
Target data [10]	INT	0
Execute	BOOL	TRUE

2. WSFR

Shift the first iSizeSrc (number of array elements copied) bits of pwDataSrc (source array) and the first uiSizeDes (number of target data entries) bits of pwDataDes (destination array) rightwards by uiSizeDes bits, and save the result to pwDataDes. In general, the execution is triggered by pulses.

The following figures show the results after two triggers.

ST

Expression	Type	Value	Prepared Value	Address	Comment
uiSizeSrc	UINT	2			
uiSizeDes	UINT	6			
Source data	ARRAY [0..1] OF INT				
Source data [0]	INT	11			
Source data [1]	INT	22			
Target data	ARRAY [0..10] OF INT				
Target data [0]	INT	0			
Target data [1]	INT	0			
Target data [2]	INT	11			
Target data [3]	INT	22			
Target data [4]	INT	11			
Target data [5]	INT	22			
Target data [6]	INT	0			
Target data [7]	INT	0			
Target data [8]	INT	0			
Target data [9]	INT	0			
Target data [10]	INT	0			
Execute	BOOL	FALSE			

LD

Expression	Type	Value
uiSizeSrc	UINT	2
uiSizeDes	UINT	6
Source data	ARRAY [0..1] OF INT	
Source data [0]	INT	11
Source data [1]	INT	22
Target data	ARRAY [0..10] OF INT	
Target data [0]	INT	0
Target data [1]	INT	0
Target data [2]	INT	11
Target data [3]	INT	22
Target data [4]	INT	11
Target data [5]	INT	22
Target data [6]	INT	0
Target data [7]	INT	0
Target data [8]	INT	0
Target data [9]	INT	0
Target data [10]	INT	0
Execute	BOOL	TRUE

■ Precautions

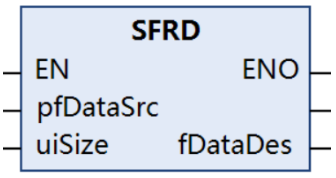
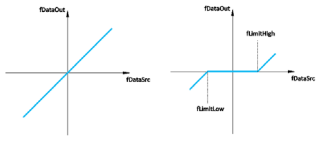
· uiSizeSrc should not exceed uiSizeDes. Otherwise, the PLC stops unexpectedly.

3.6. 10 SFRD and SFWR

■ Instruction description

SFRD: Reads data of a specified length after execution.

SFWR: Writes data of a specified length after execution.

Instruc-tion	Name	FB/FC	LD Expression	ST Expression
SFRD	Shift read in FIFO mode	FC		SFRD(pfDataSrc:=, uiSize:=, fDataDes=>);
SFWR	Shift write in FIFO mode	FC		SFWR(fDataSrc:=, pfDataDes:=, uiSize:=);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
PfDataSrc	Source data	POINTER_TO_REAL	-	-	Source data

Input Variable	Name	Data Type	Value Range	Initial Value	Description
uiSize	Number of elements in the source data queue or the floating-point array	UINT	-	-	Number of elements in the source data queue or the floating-point array
fDataDes	Read data	REAL	-	-	Read data

Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
fDataSrc	Source data	REAL	-	-	Source data
PfDataDes	Target data	POINTER_TO_REAL	-	-	Target data
uiSize	Number of elements in the source data queue or the floating-point array	UINT	-	-	Number of elements in the source data queue or the floating-point array

	Bool-ean	Bit String				Integer								Real Num-ber		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
PfDataSrc	POINTER_TO_REAL																			
uiSize	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-
fDataDes	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-
fDataSrc	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-
PfDataDes	POINTER_TO_REAL																			
uiSize	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-

■ Program example

1. SFRD

This instruction reads the first data entry in the FIFO queue pfDataSrc to fDataDes, and shifts the data within the queue rightwards by one word. Then the queue pointer decreases by 1. The first device stores a pointer. When the instruction is executed, the pointer value decreases by 1.

Then the content of the device specified by pfDataSrc is written to the position specified by the pointer in the FIFO queue fDataDes. If the pointer is 0, the instruction is not executed.

ST

Initial status

Expression	Type	Value	Prepared Value
Source data	ARRAY [0..9] OF REAL		
Source data [0]	REAL	1	
Source data [1]	REAL	1	
Source data [2]	REAL	2	
Source data [3]	REAL	3	
Source data [4]	REAL	4	
Source data [5]	REAL	5	
Source data [6]	REAL	6	
Source data [7]	REAL	7	
Source data [8]	REAL	8	
Source data [9]	REAL	9	
Data length	UINT	10	
Result data	REAL	0	
Execute	BOOL	FALSE	

Status after five triggers

Expression	Type	Value	Prepar Value
Source data	ARRAY [0..9] OF REAL		
Source data [0]	REAL	1	
Source data [1]	REAL	6	
Source data [2]	REAL	7	
Source data [3]	REAL	8	
Source data [4]	REAL	9	
Source data [5]	REAL	9	
Source data [6]	REAL	9	
Source data [7]	REAL	9	
Source data [8]	REAL	9	
Source data [9]	REAL	9	
Data length	UINT	10	
Result data	REAL	5	
Execute	BOOL	FALSE	
R_TRIG_0	R_TRIG		

LD

Initial status

Expression	Type	Value	Prepar Value
Source data	ARRAY [0..9] OF REAL		
Source data [0]	REAL	1	
Source data [1]	REAL	1	
Source data [2]	REAL	2	
Source data [3]	REAL	3	
Source data [4]	REAL	4	
Source data [5]	REAL	5	
Source data [6]	REAL	6	
Source data [7]	REAL	7	
Source data [8]	REAL	8	
Source data [9]	REAL	9	
Data length	UINT	10	
Result data	REAL	0	
Execute	BOOL	FALSE	

Status after five triggers

2. SFWR

This instruction writes the value of fDataSrc to the FIFO queue starting from pfDataDes (with the length of uiSize). The first device stores a pointer. When the instruction is executed, the pointer value increases by 1.

Then the content of the device specified by fDataSrc is written to the position specified by the pointer in the FIFO queue pfDataDes.

ST

Initial status

Expression	Type	Value	Prepared Value
R_TRIG_0	R_TRIG		
Execute	BOOL	FALSE	
Source data	REAL	1	
Data length	UINT	10	
Result data	ARRAY [0..9] OF REAL		
Result data [0]	REAL	0	
Result data [1]	REAL	0	
Result data [2]	REAL	0	
Result data [3]	REAL	0	
Result data [4]	REAL	0	
Result data [5]	REAL	0	
Result data [6]	REAL	0	
Result data [7]	REAL	0	
Result data [8]	REAL	0	
Result data [9]	REAL	0	

Status after five triggers

Expression	Type	Value	Prepared Value
R_TRIG_0	R_TRIG		
CLK	BOOL	TRUE	
Q	BOOL	FALSE	
Execute	BOOL	FALSE	
Source data	REAL	1	
Data length	UINT	10	
Result data	ARRAY [0..9] OF REAL		
Result data [0]	REAL	5	
Result data [1]	REAL	1	
Result data [2]	REAL	1	
Result data [3]	REAL	1	
Result data [4]	REAL	1	
Result data [5]	REAL	1	
Result data [6]	REAL	0	
Result data [7]	REAL	0	
Result data [8]	REAL	0	
Result data [9]	REAL	0	

LD

Initial status

Expression	Type	Value	Prepared Value
Execute	BOOL	FALSE	
Source data	REAL	1	
Data length	UINT	10	
Result data	ARRAY [0..9] OF REAL		
Result data [0]	REAL	0	
Result data [1]	REAL	0	
Result data [2]	REAL	0	
Result data [3]	REAL	0	
Result data [4]	REAL	0	
Result data [5]	REAL	0	
Result data [6]	REAL	0	
Result data [7]	REAL	0	
Result data [8]	REAL	0	
Result data [9]	REAL	0	

Status after five triggers

Expression	Type	Value	Prepared Value
Execute	BOOL	TRUE	
Source data	REAL	1	
Data length	UINT	10	
Result data	ARRAY [0..9] OF REAL		
Result data [0]	REAL	5	
Result data [1]	REAL	1	
Result data [2]	REAL	1	
Result data [3]	REAL	1	
Result data [4]	REAL	1	
Result data [5]	REAL	1	
Result data [6]	REAL	0	
Result data [7]	REAL	0	
Result data [8]	REAL	0	
Result data [9]	REAL	0	

■ Precautions

· The first element of the source data needs to be assigned in advance. Otherwise, the execution fails.

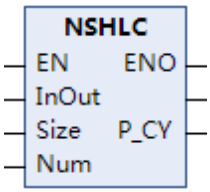
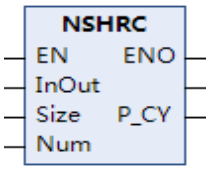
3.6. 11 NSHLC and NSHRC

These instructions shift bit strings by more bits. The Carry (CY) Flag is included.

NSHLC: Shifts the bit strings to the left (toward higher bits).

NSHRC: Shifts the bit strings to the right (toward lower bits).

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
NSHLC	Shift N-bits left with carry	FC		NSHLC (InOut :=, Size := , Num := , P_CY =>);
NSHRC	Shift N-bits right with carry	FC		NSHRC (InOut :=, Size :=, Num :=, P_CY =>);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
Size	Number of bits in the shift register	USINT	Depends on data types	1	Number of bits in the shift register
Num	Number of bits to shift	USINT	Depends on data types	1	Number of bits to shift

■ In-out variables

In-Out Variable	Name	Data Type	Value Range	Initial Value	Description

InOut[] (array)	Shift register array	-	Depends on data types	-	Shift register array

■ Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
P_CY	Carry flag	BOOL	Depends on data types	-	Carry flag

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
InOut[] (array)	√	√	√	√	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Size	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Num	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-
P_CY	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

■ Function

These instructions shift Size array elements starting from InOut[0] in shift register array InOut[] by Num bits. The last bit that is shifted out of the register is output to the Carry (CY) Flag. Zeros are inserted for the bits at the other end.

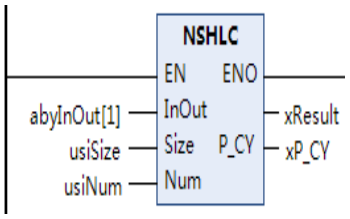
NSHLC: Shifts bits leftwards from the lower elements in the array to the higher elements and from the least-significant bits to the most-significant bits.

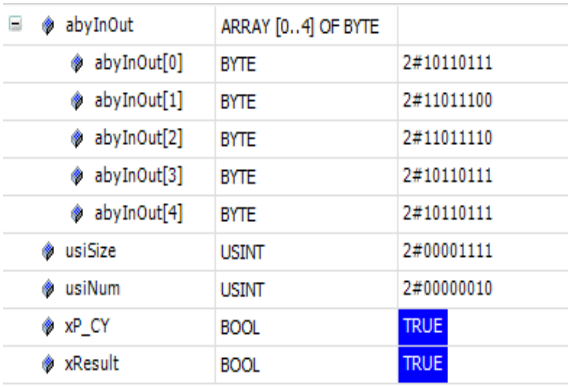
NSHRC: Shifts bits rightwards from the higher elements in the array to the lower elements and from the most-significant bits to the least-significant bits.

■ Program example

	LD	ST

Defined variable	VAR
	abyInOut : ARRAY [0..4] OF BYTE := [5(2#10110111)];
	usiSize : USINT := 15;
	usiNum : USINT := 2;
	xP_CY : BOOL;
	xResult : BOOL;
	END_VAR	

Program		xResult := NSHLC(InOut:= abyInOut[1], Size:= usiSize, Num:= usiNum, P_CY=> xP_CY);

Running result	

■ Precautions

When the value of Num is 0, the shift operation is not performed, the return value is TRUE, InOut[] does not change, and the value of P_CY is FALSE.

When the value of Num is greater than Size, Size bit values from bit 0 of InOut[0] are FALSE. The value of the Carry Flag (CY) changes to FALSE.

When the value of Size exceeds the InOut[] array, the return value is FALSE, InOut[] does not change, and the value of P_CY is FALSE.

3. 7 Data Type Conversion Instructions

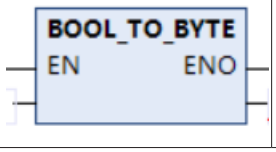
3.7. 1 Instruction List

Instruction Category	Name	FB/FC	Function
Data type conversion instructions	BOOL_TO_<TYPE>	FC	Conversion from bool to a type
	BYTE_TO_<TYPE>	FC	Conversion from byte to a type
	WORD_TO_<TYPE>	FC	Conversion from word to a type
	DWORD_TO_<TYPE>	FC	Conversion from doubleword to a type
	INT_TO_<TYPE>	FC	Conversion from integer to a type
	SINT_TO_<TYPE>	FC	Conversion from short integer to a type
	DINT_TO_<TYPE>	FC	Conversion from long integer to a type
	UDINT_TO_<TYPE>	FC	Conversion from unsigned long integer to a type
	REAL_TO_<TYPE>	FC	Conversion from real number to a type
	STRING_TO_<TYPE>	FC	Conversion from text string to a type
	TIME_TO_<TYPE>	FC	Conversion from time to a type
	TOD_TO_<TYPE>	FC	Conversion from time of day to a type
	DATE_TO_<TYPE>	FC	Conversion from date to a type
	DT_TO_<TYPE>	FC	Conversion from date and time to a type
	DtToString	FC	DT-to-text string conversion
	DateToString	FC	Date-to-text string conversion
	TodToString	FC	Time of day-to-text string conversion
	EnumToNum	FC	Enumeration-to-integer conversion
	NumToEnum	FC	Integer-to-enumeration conversion

3.7. 2 BOOL_TO_<TYPE>

■ Instruction description

This instruction converts a variable from the Boolean type to other data types.

Instruction	Name	FB/FC	LD Expression	ST Expression
BOOL_TO_<TYPE>	Conversion from bool to a type	FC		:=BOOL_TO_<TYPE>();

■ Variables

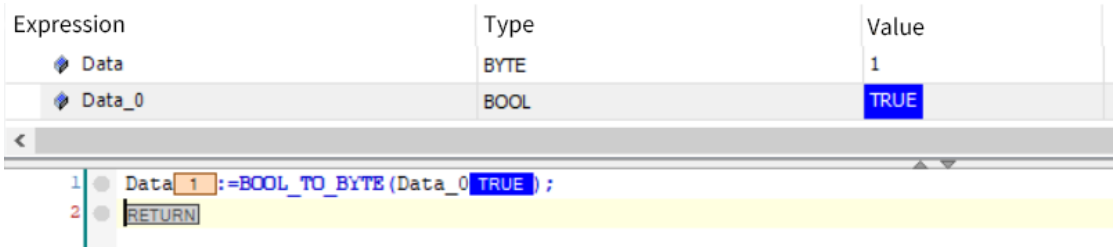
Variable	Name	Data Type	Value Range	Initial Value	Description
IN	Data source	BOOL	-	-	Data source
OUT	Output data	<TYPE>	-	-	Data after conversion

	Bool-ean	Bit String				Integer								Real Num-ber		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
IN	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
OUT	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√

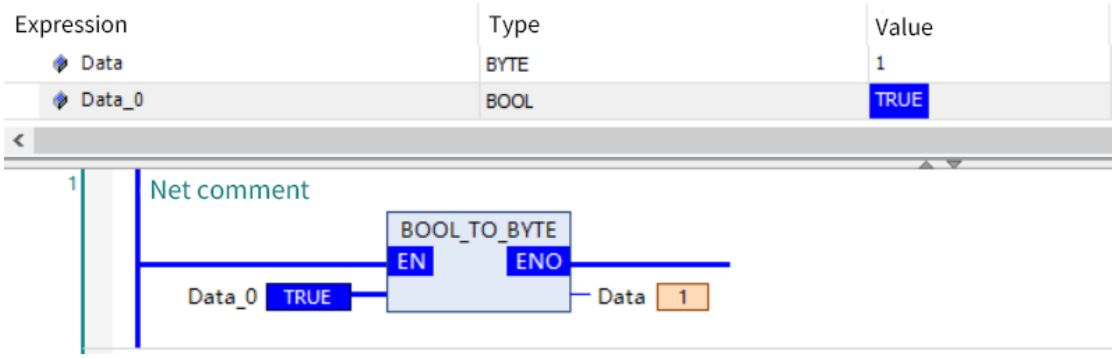
■ Program example

BOOL_TO_BYTE is used as an example.

ST



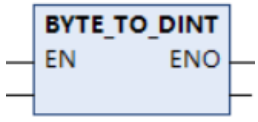
LD



3.7. 3 BYTE_TO_<TYPE>

■ Instruction description

This instruction converts a variable from the Byte type to other data types.

Instruction	Name	FB/FC	LD Expression	ST Expression
BYTE_TO_<TYPE>	Conversion from byte to a type	FC		:=BYTE_TO_<TYPE>();

■ Variables

Variable	Name	Data Type	Value Range	Initial Value	Description
IN	Data source	BYTE	-	-	Data source

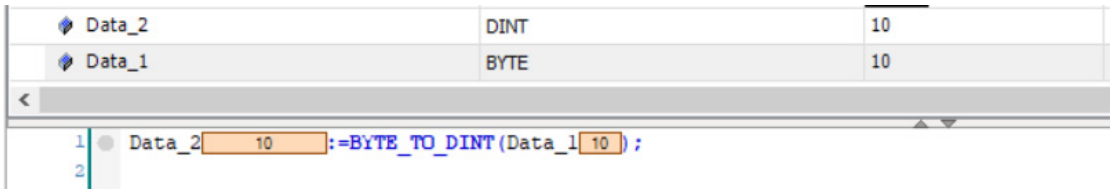
Variable	Name	Data Type	Value Range	Initial Value	Description
OUT	Output data	<TYPE>	-	-	Data after conversion

	Bool-ean	Bit String				Integer								Real Num-ber		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
IN	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
OUT	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√

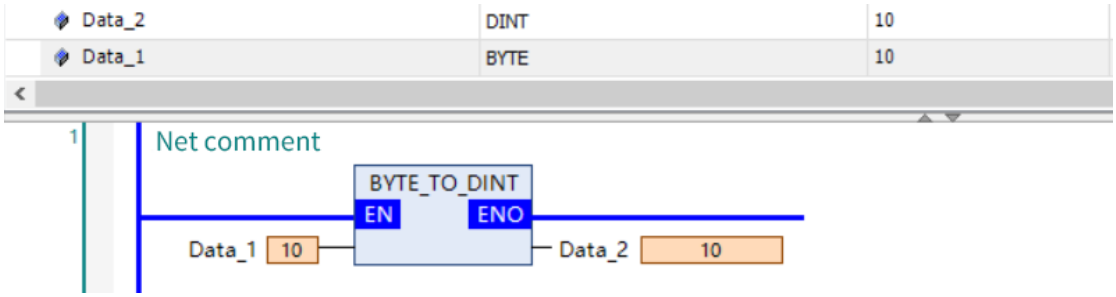
■ Program example

BYTE_TO_DINT is used as an example.

ST



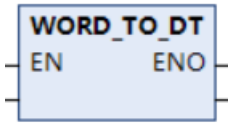
LD



3.7. 4 WORD_TO_<TYPE>

■ Instruction description

This instruction converts a variable from the Word type to other data types.

Instruction	Name	FB/FC	LD Expression	ST Expression
WORD_TO_<TYPE>	Conversion from word to a type	FC		:=WORD_TO_<TYPE>();

■ Variables

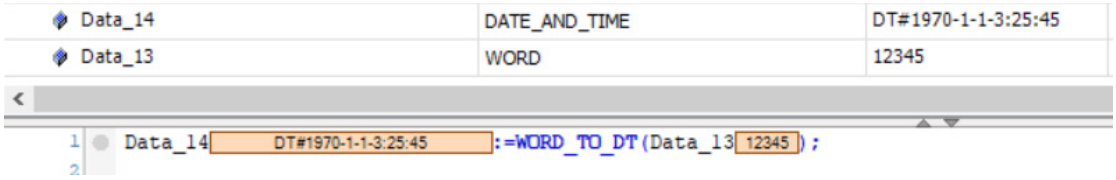
Variable	Name	Data Type	Value Range	Initial Value	Description
IN	Data source	WORD	-	-	Data source
OUT	Output data	<TYPE>	-	-	Data after conversion

	Bool-ean	Bit String				Integer								Real Num-ber		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
IN	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
OUT	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√

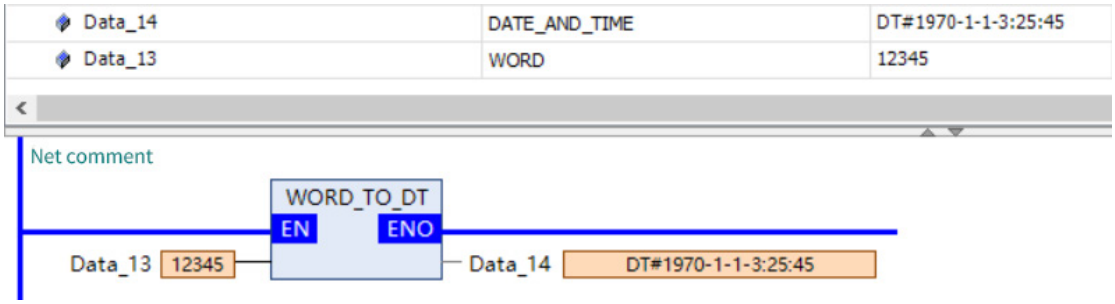
■ Program example

WORD_TO_DT is used as an example.

ST



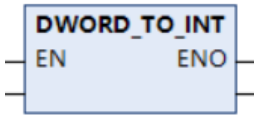
LD



3.7. 5 DWORD_TO_<TYPE>

■ Instruction description

This instruction converts a variable from the DWORD type to other data types.

Instruction	Name	FB/FC	LD Expression	ST Expression
DWORD_TO_<TYPE>	Conversion from doubleword to a type	FC		:=DWORD_TO_<TYPE>();

■ Variables

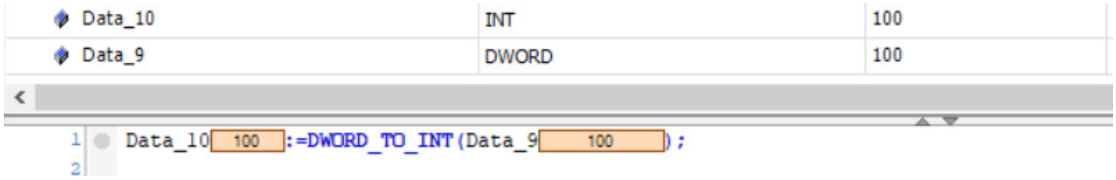
Variable	Name	Data Type	Value Range	Initial Value	Description
IN	Data source	DWORD	-	-	Data source
OUT	Output data	<TYPE>	-	-	Data after conversion

	Bool-ean	Bit String				Integer								Real Num-ber		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
IN	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
OUT	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√

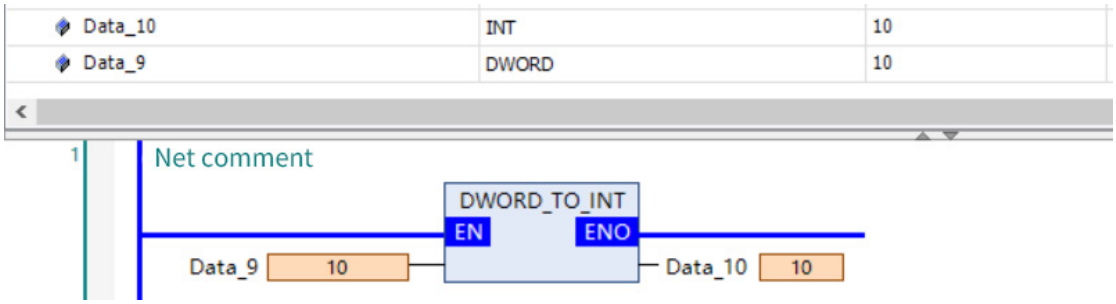
■ Program example

DWORD_TO_DT is used as an example.

ST



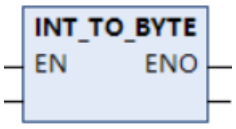
LD



3.7. 6 INT_TO_<TYPE>

■ Instruction description

This instruction converts a variable from the Integer type to other data types.

Instruction	Name	FB/FC	LD Expression	ST Expression
INT_TO_<TYPE>	Conversion from integer to a type	FC		:=INT_TO_<TYPE>();

■ Variables

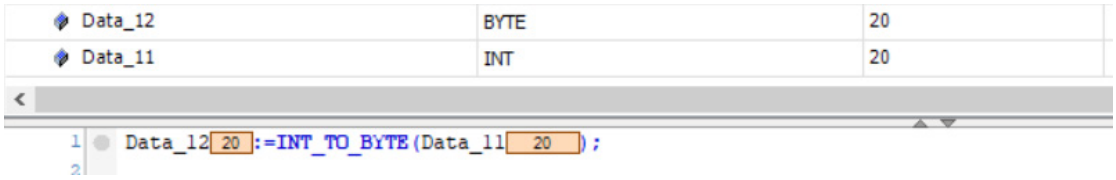
Variable	Name	Data Type	Value Range	Initial Value	Description
IN	Data source	INT	-	-	Data source
OUT	Output data	<TYPE>	-	-	Data after conversion

	Bool-ean	Bit String				Integer								Real Num-ber		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
IN	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-
OUT	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√

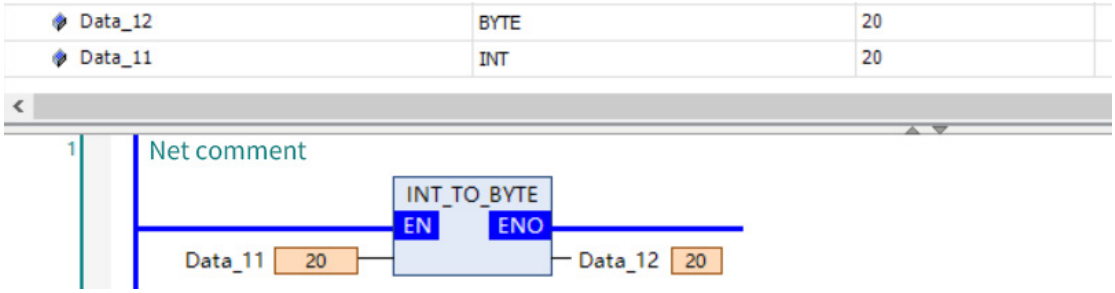
■ Program example

INT_TO_BYTE is used as an example.

ST



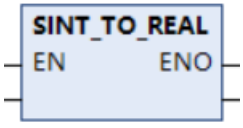
LD



3.7. 7 SINT_TO_<TYPE>

■ Instruction description

This instruction converts a variable from the short integer type to other data types.

Instruction	Name	FB/FC	LD Expression	ST Expression
SINT_TO_<TYPE>	Conversion from short integer to a type	FC		:=SINT_TO_<TYPE>();

■ Variables

In-out variables

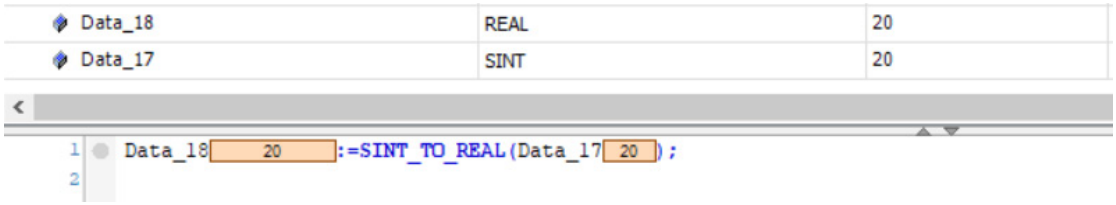
Variable	Name	Data Type	Value Range	Initial Value	Description
IN	Data source	INT	-	-	Data source
OUT	Output data	<TYPE>	-	-	Data after conversion

	Bool-ean	Bit String				Integer								Real Num-ber		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
IN	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-
OUT	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√

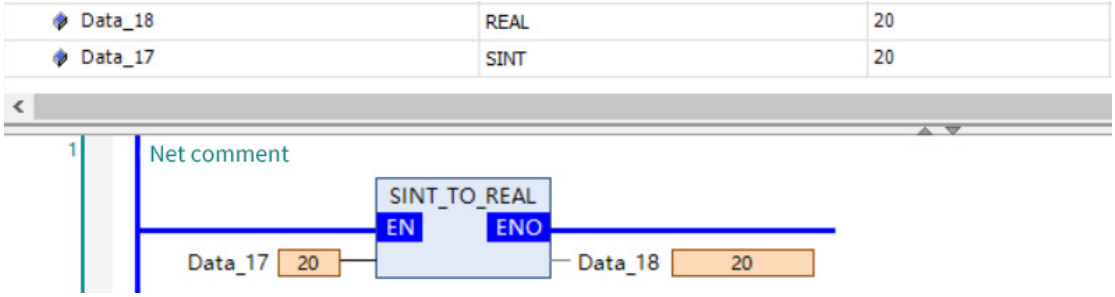
■ Program example

SINT_TO_REAL is used as an example.

ST



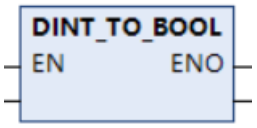
LD



3.7. 8 DINT_TO_<TYPE>

■ Instruction description

This instruction converts a variable from the long integer type to other data types.

Instruction	Name	FB/FC	LD Expression	ST Expression
DINT_TO_<TYPE>	Conversion from long integer to a type	FC		:=DINT_TO_<TYPE>();

■ Variables

In-out variables

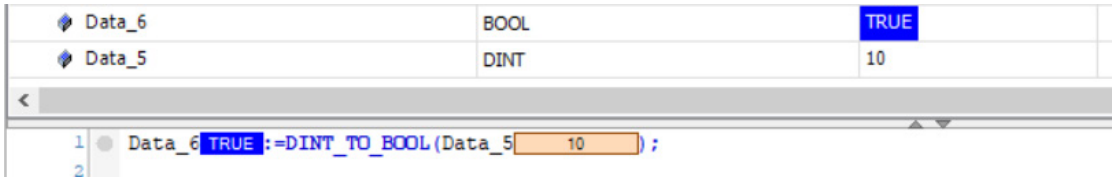
Variable	Name	Data Type	Value Range	Initial Value	Description
IN	Data source	INT	-	-	Data source
OUT	Output data	<TYPE>	-	-	Data after conversion

	Bool-ean	Bit String				Integer								Real Num-ber		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
IN	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-
OUT	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√

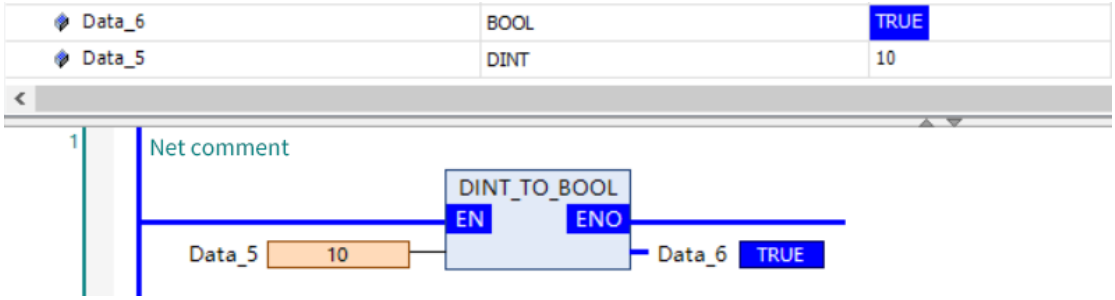
■ Program example

DINT_TO_BOOL is used as an example.

ST



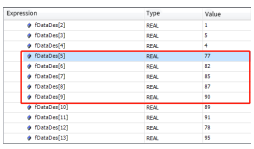
LD



3.7. 9 UDINT_TO_<TYPE>

■ Instruction description

This instruction converts a variable from the unsigned long integer type to other data types.

Instruction	Name	FB/FC	LD Expression	ST Expression
UDINT_TO_<TYPE>	Conversion from unsigned long integer to a type	FC		:=UDINT_TO_<TYPE>();

■ Variables

In-out variables

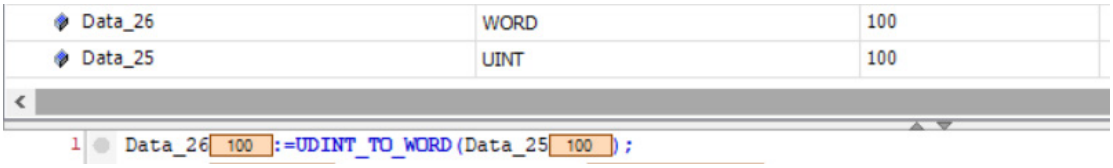
Variable	Name	Data Type	Value Range	Initial Value	Description
IN	Data source	INT	-	-	Data source
OUT	Output data	<TYPE>	-	-	Data after conversion

	Bool-ean	Bit String				Integer								Real Num-ber		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
IN	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-
OUT	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√

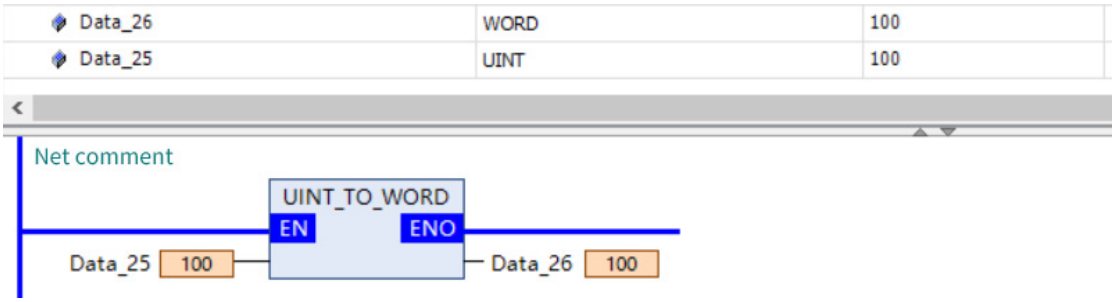
■ Program example

UDINT_TO_WORD is used as an example.

ST



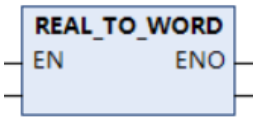
LD



3.7. 10 REAL_TO_<TYPE>

■ Instruction description

This instruction converts a variable from the real number type to other data types.

Instruction	Name	FB/FC	LD Expression	ST Expression
REAL_TO_<TYPE>	Conversion from real number to a type	FC		:=REAL_TO_<TYPE>();

■ Variables

In-out variables

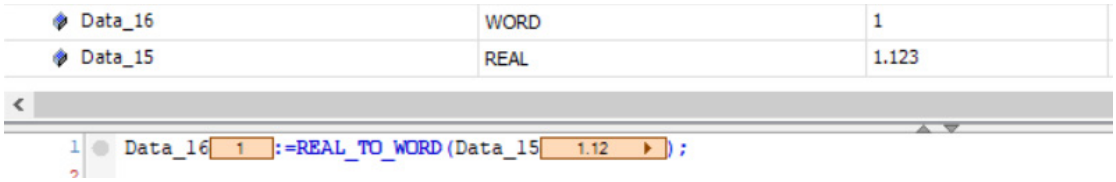
Variable	Name	Data Type	Value Range	Initial Value	Description
IN	Data source	INT	-	-	Data source
OUT	Output data	<TYPE>	-	-	Data after conversion

	Bool-ean	Bit String				Integer								Real Num-ber		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
IN	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-
OUT	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√

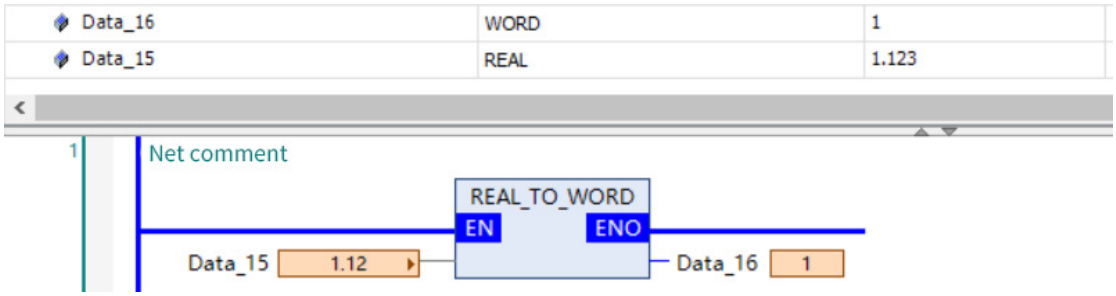
■ Program example

REAL_TO_WORD is used as an example.

ST



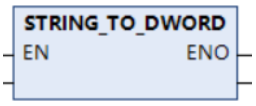
LD



3.7. 11 STRING_TO_<TYPE>

■ Instruction description

This instruction converts a variable from the String type to other data types.

Instruction	Name	FB/FC	LD Expression	ST Expression
STRING_TO_<TYPE>	Conversion from text string to a type	FC		:=STRING_TO_<TYPE>();

■ Variables

In-out variables

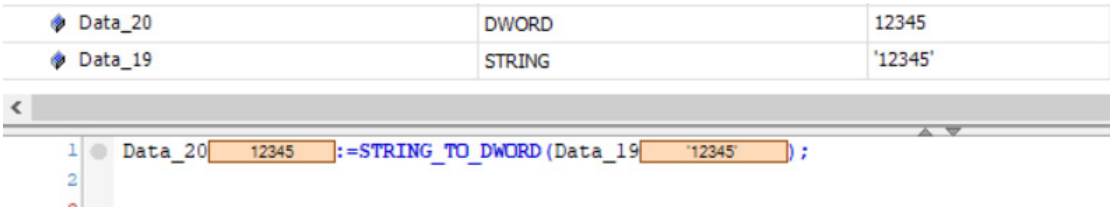
Variable	Name	Data Type	Value Range	Initial Value	Description
IN	Data source	STRING	-	-	Data source
OUT	Output data	<TYPE>	-	-	Data after conversion

	Bool-ean	Bit String				Integer								Real Num-ber		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
IN	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√
OUT	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√

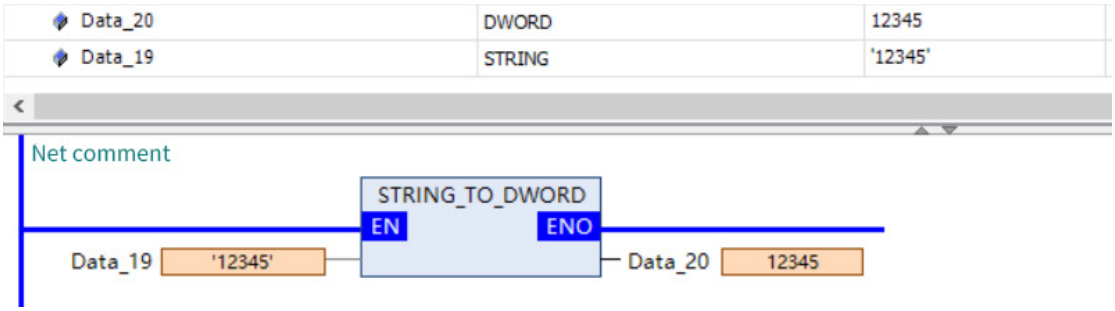
■ Program example

STRING_TO_DWORD is used as an example.

ST

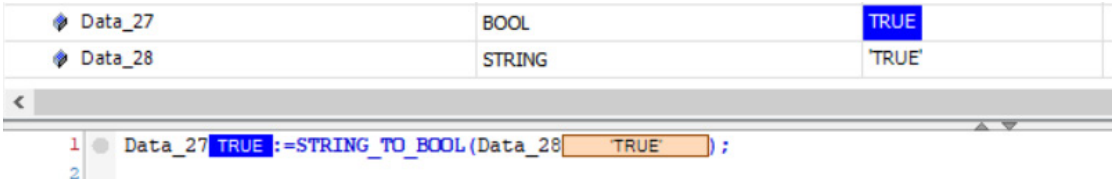


LD



■ Special example

ST

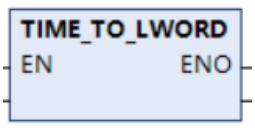


When the input for the String data type is TRUE, the corresponding Bool type variable is set to TRUE.

3.7. 12 TIME_TO_<TYPE>

■ Instruction description

This instruction converts a variable from the Time type to other data types.

Instruction	Name	FB/FC	LD Expression	ST Expression
TIME_TO_<TYPE>	Conversion from time to a type	FC		:=TIME_TO_<TYPE>();

■ Variables

In-out variables

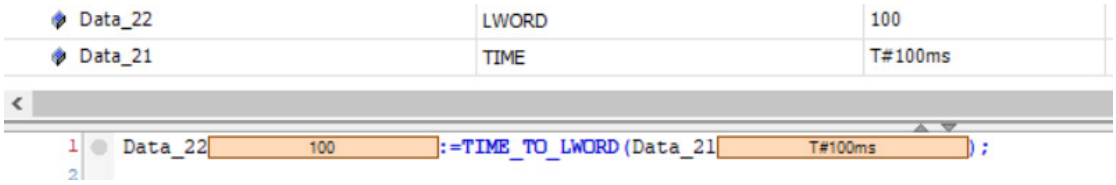
Variable	Name	Data Type	Value Range	Initial Value	Description
IN	Data source	INT	-	-	Data source
OUT	Output data	<TYPE>	-	-	Data after conversion

	Bool-ean	Bit String				Integer								Real Num-ber		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
IN	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-
OUT	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√

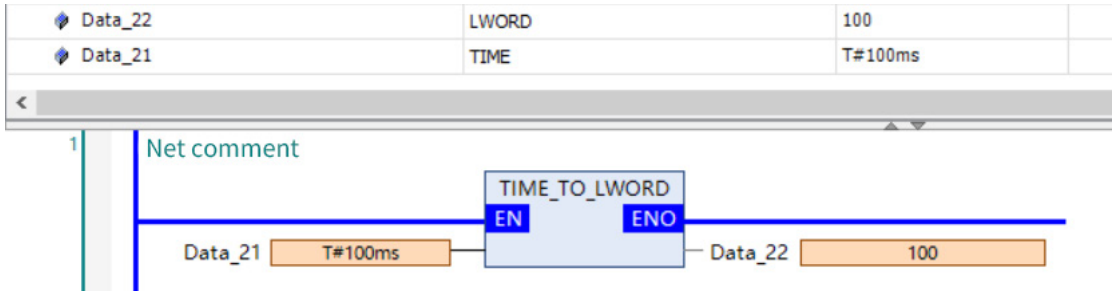
■ Program example

TIME_TO_LWORD is used as an example.

ST



LD



3.7. 13 TOD_TO_<TYPE>

■ Instruction description

This instruction converts a variable from the TOD type to other data types.

Instruction	Name	FB/FC	LD Expression	ST Expression
TOD_TO_<TYPE>	Conversion from time of day to a type	FC	TOD_TO_LWORD EN ENO	:=TIME_TO_<TYPE>();

■ Variables

In-out variables

Variable	Name	Data Type	Value Range	Initial Value	Description
IN	Data source	INT	-	-	Data source

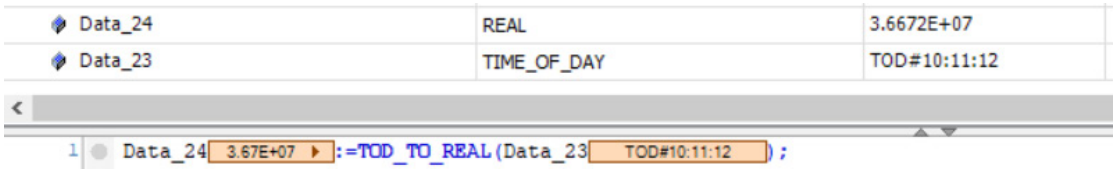
Variable	Name	Data Type	Value Range	Initial Value	Description
OUT	Output data	<TYPE>	-	-	Data after conversion

	Bool-ean	Bit String				Integer								Real Num-ber		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
IN	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-
OUT	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√

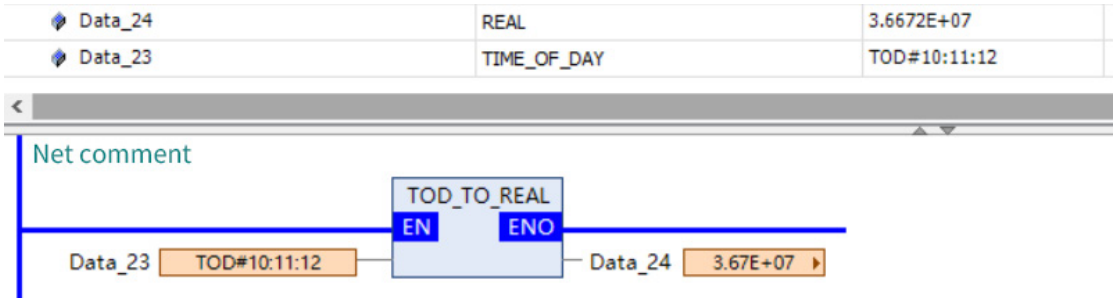
■ Program example

TOD_TO_REAL is used as an example.

ST



LD



3.7. 14 DATE_TO_<TYPE>

■ Instruction description

This instruction converts a variable from the Date type to other data types.

Instruction	Name	FB/FC	LD Expression	ST Expression
DATE_TO_<TYPE>	Conversion from date to a type	FC	DATE_TO_DWORD EN ENO	:=DATE_TO_<TYPE>();

■ Variables

In-out variables

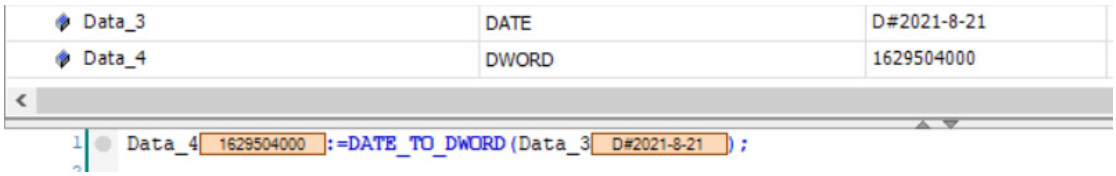
Variable	Name	Data Type	Value Range	Initial Value	Description
IN	Data source	INT	-	-	Data source
OUT	Output data	<TYPE>	-	-	Data after conversion

	Bool-ean	Bit String				Integer								Real Num-ber		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
IN	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-
OUT	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√

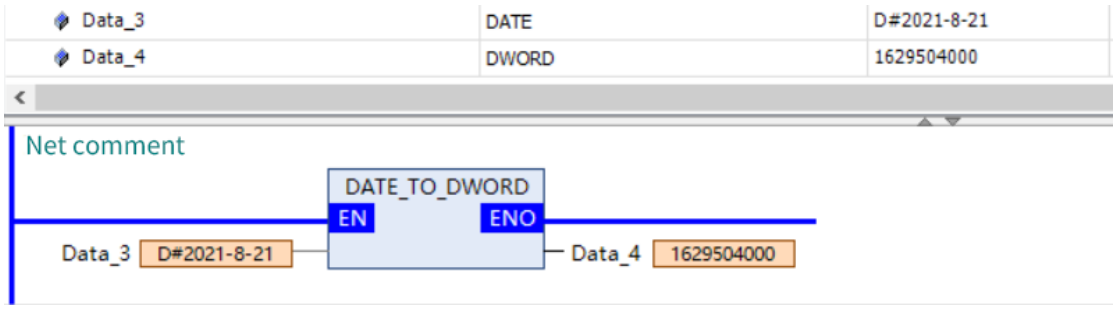
■ Program example

DATE_TO_DWORD is used as an example.

ST



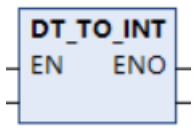
LD



3.7. 15 DT_TO_<TYPE>

■ Instruction description

This instruction converts a variable from the DT type to other data types.

Instruction	Name	FB/FC	LD Expression	ST Expression
DT_TO_<TYPE>	Conversion from date and time to a type	FC		:=DT_TO_<TYPE>();

■ Variables

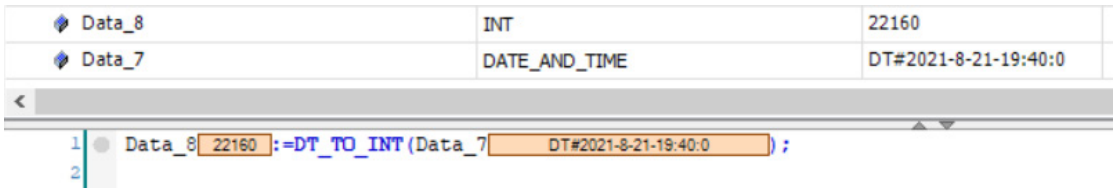
Variable	Name	Data Type	Value Range	Initial Value	Description
IN	Data source	INT	-	-	Data source
OUT	Output data	<TYPE>	-	-	Data after conversion

	Bool-ean	Bit String				Integer								Real Num-ber		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
IN	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-
OUT	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√

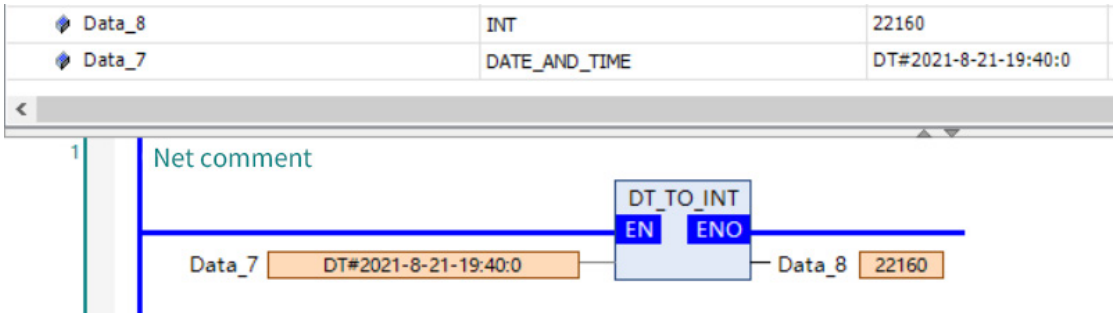
■ Program example

DT_TO_INT is used as an example.

ST



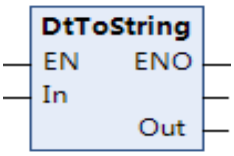
LD



3.7. 16 DtToString

This instruction converts date and time to a text string.

■ Instruction description

Instruction	Name	FB/FC	LD Expression	ST Expression
DtToString	DT-to-text string conversion	FC		DtToString (In :=, Out =>);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
In	Date and time	DT	Depends on data types	DT#1970-1-1-0:0:0	Date and time

Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
Out	Conversion result	STRING[19]	19 byte	-	Result text string

	Bool-ean	Bit String				Integer								Real Num-ber		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-
Out	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√

■ Function

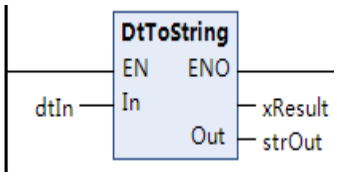
This instruction converts date and time In to a text string.

NULL is added to the end of Out.

■ Program example

	LD	ST
Defined variable		
Program		
Running result		

```
VAR
    dtIn    : DT := DT#2022-1-1-12:34:56;
    strOut  : STRING[19];
    xResult : BOOL;
END_VAR
```



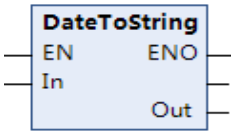

```
xResult := DtToString(
        In:= dtIn,
        Out=> strOut);
```

dtIn	DATE_AND_TIME	DT#2022-1-1-12:34:56
strOut	STRING(19)	'2022-01-01-12:34:56'
xResult	BOOL	TRUE

3.7. 17 DateToString

This instruction converts date to a text string.

■ Instruction description

Instruction	Name	FB/FC	LD Expression	ST Expression
DateToString	Date-to-text string conversion	FC		DateToString (In :=, Out =>);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
In	Date	DATE	Depends on data types	D#1970-1-1	Date

Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
Out	Conversion result	STRING[10]	10 byte	-	Result text string

	Bool-ean	Bit String				Integer								Real Num-ber		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-
Out	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√

■ Function

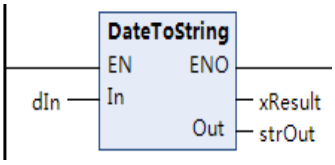
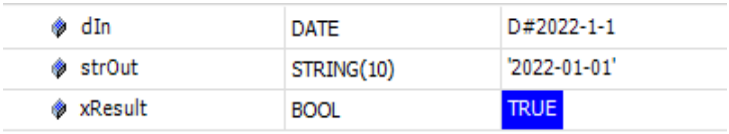
This instruction converts date In to a text string.

NULL is added to the end of Out.

■ Program example

	LD	ST
Defined variable		

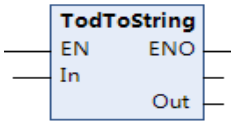
```
VAR
    dIn      : DATE := D#2022-1-1;
    strOut   : STRING[10];
    xResult : BOOL;
END_VAR
```

	LD	ST
Program		xResult := DateToString(In:= dIn, Out=> strOut);
Running result		

3.7. 18 TodToString

This instruction converts TOD to a text string.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
TodToString	Time of day-to-text string conversion	FC		TodToString(In:= , Out=>);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
In	TOD	TOD	Depends on data types	TOD#:0:0:0	Hour, minute, second

Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
Out	String	Output	13 bytes (12 single-byte alphanumeric characters and the final NULL characters)	-	String

	Bool-ean	Bit String				Integer								Real Num-ber		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-

	Bool-ean	Bit String				Integer								Real Num-ber		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Out	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√

■ Function

This instruction converts TOD In to a text string. NULL is added to the end of Out.

■ Program example

	LD	ST
Defined variable	VAR IN_VAR : TOD; Out_VAR : STRING(13); Return_VAR : BOOL; END_VAR	
Program	TodToString EN ENO IN_VAR — In Return_VAR — Out Out_VAR	Return_VAR:=TodToString(In:=IN_VAR , Out=>Out_VAR);
Running result	IN_VAR TIME_OF_DAY TOD#23:59:59.999 Out_VAR STRING(13) '23:59:59.999' Return_VAR BOOL TRUE	

■ Precautions

The unit of Out is ms.

3.7. 19 EnumToNum

This instruction converts enumeration data to DINT data.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
EnumToNum	Enumeration-to-integer conversion	FC	EnumToNum EN ENO In Out	EnumToNum (In :=, Out =>);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
In	Data to convert	-	Depends on data types	-	Data to convert

Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
Out	Conversion result	DINT	Depends on data types	-	Conversion result

	Bool-ean	Bit String				Integer								Real Num-ber		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In	Enumeration																			
Out	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-

■ Function

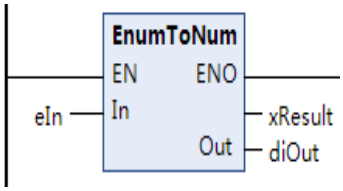
This instruction converts the value of data to convert In, which is an enumeration, to a DINT value, and outputs the value to conversion result Out.

Use this instruction, for example, to monitor the value of an enumerated variable on a display or other devices that do not handle enumerated variables.

■ Program example

	LD	ST
Defined variable		
Program		
Running result		

```
VAR
    eIn     : Weekday := Weekday.Tue;
    diOut   : DINT;
    xResult : BOOL;
END_VAR
```




```
xResult := EnumToNum(In:= eIn,
    Out=> diOut);
```

eIn	WEEKDAY	Tue
diOut	DINT	2
xResult	BOOL	TRUE

■ Precautions

When the value after conversion of the data to convert In exceeds the DINT data range, an error occurs,

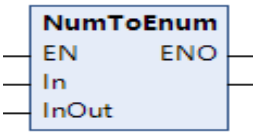
the return value is FALSE, and Out is initialized to 0.

Set the basic data type for the data to convert In to BYTE, WORD, DWORD, LWORD, USINT, UINT, UDINT, ULINT, SINT, INT, DINT, or LINT. Otherwise, the return value is FALSE and Out is initialized to 0.

3.7. 20 NumToEnum

This instruction converts DINT data to enumeration data.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
NumToEnum	Integer-to-enumeration conversion	FC		NumToEnum (In :=, InOut :=);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
In	Data to convert	DINT	Depends on data types	0	Data to convert

In-out variables

In-Out Variable	Name	Data Type	Value Range	Initial Value	Description
InOut	Conversion result	-	Depends on data types	-	Conversion result

	Bool-ean	Bit String				Integer								Real Num-ber		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-
Out	Enumeration																			

■ Function

This instruction converts the value of data to convert In, which is DINT data, to an enumeration value, and outputs the value to conversion result Out.

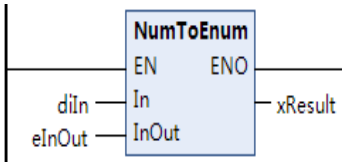
Use this instruction, for example, to change the value of an enumerated variable on a display or other devices that do not handle enumerated variables.

■ Program example

	LD	ST

<table>
<tr><td>Defined variable</td><td colspan="2">

```
VAR
    diIn    : DINT := 2;
    eInOut  : Weekday;
    xResult : BOOL;
END_VAR
```

</td></tr>
<tr><td>Program</td><td> 
</td><td>

```
xResult := NumToEnum(In:= diIn,
    InOut:= eInOut);
```

</td></tr>
<tr><td>Running result</td><td colspan="2">

diIn	DINT	2
eInOut	WEEKDAY	Tue
xResult	BOOL	TRUE

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

■ Precautions

When the value after conversion of the data to convert In exceeds the range of the enumerated variable InOut, an error occurs, the return value is FALSE, and Out does not change.

Ensure that the value of In after conversion is within the enumeration values of InOut. Otherwise, the program obtains an unexpected result, the return value is TRUE, and the value of Out is the converted value of In.

Set the basic data type for the enumeration value of InOut to BYTE, WORD, DWORD, LWORD, USINT, UINT, UDINT, ULINT, SINT, INT, DINT, or LINT. Otherwise, the return value is FALSE and Out does not change.

3. 8 Data Processing Instructions

3.8. 1 Instruction List

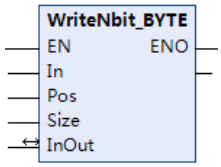
Instruction Category	Name	FB/FC	Function
Data processing instructions	WriteNbit_**	FC	N-bit write group
	ReadNbit_**	FC	N-bit read group
	TransBits	FC	Move bits
	Encoder	FC	Bit encoder
	Decoder	FC	Bit decoder
	NumToDecString	FC	Integer-to-fixed-length decimal text string conversion
	NumToHexString	FC	Integer-to-fixed-length hexadecimal text string conversion
	HexStringToNum_**	FC	Hexadecimal text string-to-number conversion group
	AryToString	FC	Array-to-text string conversion
	StringToAry	FC	Text string-to-array conversion
	Clear	FC	Initialize
	SetBlock	FC	Block set
	MOVE	FC	Move
	BMOV	FC	Batch move
	FMOV	FC	Multi-point move
	BON	FC	Bit state check
	SUM	FC	Sum of ON bits
	BTOW	FC	Byte to word
	SWAP	FC	Swap bytes
	XCH	FC	Data exchange
	HEXinASCII_TO_BYTE	FC	ASCII-to-byte conversion
	BYTE_TO_HEXinASCII	FC	Byte-to-ASCII conversion
	WORD_AS_STRING	FC	Word-to-string conversion
	Byte_To_HexString	FC	Byte-to-hexadecimal text string conversion
	Word_To_HexString	FC	Word-to-hexadecimal text string conversion
	Dword_To_HexString	FC	Doubleword-to-hexadecimal text string conversion
	CopyRealToNum	FC	Bit pattern copy (real number to signed integer)
	CopyByteToNum	FC	Bit pattern copy (bit string to signed integer)
	CopyDwordToReal	FC	Bit pattern copy (bit string to real number)
	CopyNumToByte	FC	Bit pattern copy (signed integer to bit string)
	CopyNumToReal	FC	Bit pattern copy (signed integer to real number)
	CopyRealToDword	FC	Bit pattern copy (real number to bit string)
	BitCnt	FC	Bit counter
	AryByteTo	FC	Conversion from byte array
	DispartReal	FC	Separate mantissa and exponent
	UniteReal	FC	Combine real number mantissa and exponent
	Gray	FC	Gray code conversion
	ColmToLine_**	FC	Column-to-line conversion group
	LineToColm	FC	Line-to-column conversion
	FixNumToString	FC	Fixed-decimal number-to-text string conversion
	StringToFixNum	FC	Text string-to-fixed-decimal number conversion

Instruction Category	Name	FB/FC	Function
Data processing instructions	DispartDigit	FC	Four-bit separation
	UniteDigit_**	FC	Four-bit join group
	Dispart8Bit	FC	Byte data separation
	Unite8Bit_**	FC	Byte data join group
	ToAryByte	FC	Conversion to byte array
	AryToBCD	FC	Array-to-BCD conversion
	AryToBin	FC	Array-to-bit string conversion
	MoveDigit	FC	Move digit
	Exchange	FC	Data exchange
	AryExchange	FC	Array data exchange
	AryMove	FC	Array move
	SizeOfAry	FC	Get number of array elements
	GrayToBin_**	FC	Gray code-to-binary code conversion group
	BinToGray_**	FC	Binary code-to-gray code conversion

3.8. 2 WriteNbit_**

This instruction writes multiple bits to a bit string.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
WriteNbit_**	N-bit write group	FC		WriteNbit _INT(In :=, InOut :=, Pos :=, Size :=);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
In	Read source	-	Depends on data types	-	Bit string from which to read bits to write to InOut
Pos	Write position	USINT	Depends on data types	0	Bit position to which to write
Size	Write size	USITN	Depends on data types	0	Number of bits to write

In-out variables

In-Out Variable	Name	Data Type	Value Range	Initial Value	Description
InOut	Write target	BOOL	[FALSE, TRUE]	FALSE	Read result

	Bool-ean	Bit String				Integer								Real Num-ber		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In	-	√	√	√	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Pos	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Size	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-
InOut	Data with the same data type as In																			

■ Function

This instruction first reads the lower Size bits from read source In. Then it writes the read values to write position Pos in write target InOut.

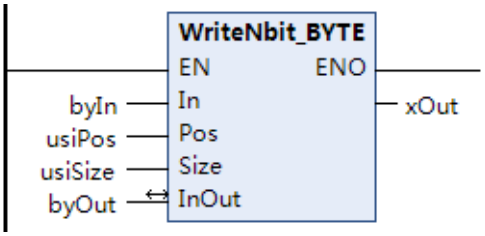
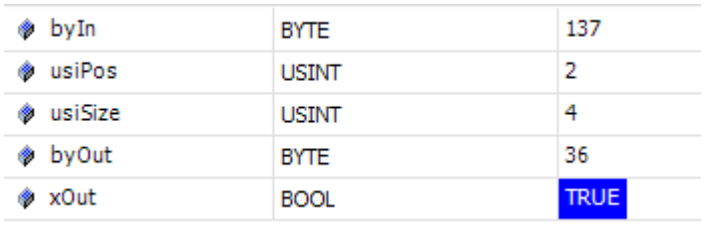
The instruction name is determined by the data types of In and InOut. For example, if In and InOut are of the Byte data type, the instruction is WriteNbit_BYTE.

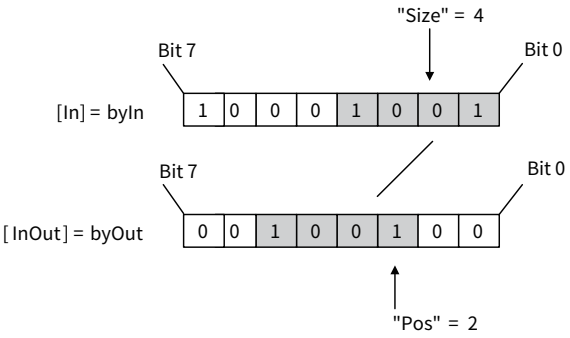
■ Program example

The following example shows the WriteNbit_BYTE instruction when In is BYTE#16#89, Pos is USINT#2, and Size is USINT#4.

	LD	ST

Defined variable		VAR
		byIn : BYTE := 16#89;
		usiPos : USINT := 2;
		usiSize : USINT := 4;
		byOut : BYTE;
		xOut : BOOL;
		END_VAR

Program		WriteNbit_BYTE(In := byIn, Pos := usiPos, Size := usiSize, InOut := byOut);
Running result		

	LD	ST
Work principle		

■ Precautions

· The data types of In and InOut must be the same.

· When the value of Size is 0, InOut does not change.

· When the instruction is used in ST, return value Out is not used.

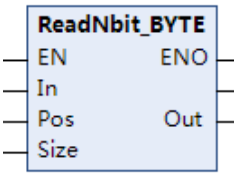
· When the value of Size or Pos exceeds the valid range, an error occurs, the return value is FALSE, and InOut does not change.

· When the bit string in InOut does not have enough bits for the number of bits specified by Size from the position specified by Pos, an error occurs, the return value is FALSE, and InOut does not change.

3.8. 3 ReadNbit_**

This instruction reads multiple bits from a bit string.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
ReadNbit_**	N-bit read group	FC		ReadNbit_** (In :=, Pos :=, Size :=, Out =>);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
In	Read source	-	Depends on data types	-	Bit string from which to read bits to write to InOut
Pos	Read position	USINT	Depends on data types	0	Bit position to read
Size	Read size	USITN	Depends on data types	0	Number of bits to read

Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
Out	Read result	-	Depends on data types	-	Read result

	Bool-ean	Bit String				Integer								Real Num-ber		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In	-	√	√	√	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Pos	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Size	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Out	Data with the same data type as In																			

■ Function

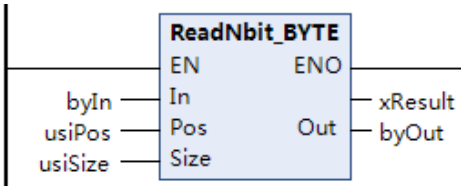
This instruction reads the values of the upper Size bits from read position Pos in source bit string In, and assigns the values to read result Out.

The instruction name is determined by the data types of In and Out. For example, if In and Out are of the Byte data type, the instruction is ReadNbit_BYTE.

■ Program example

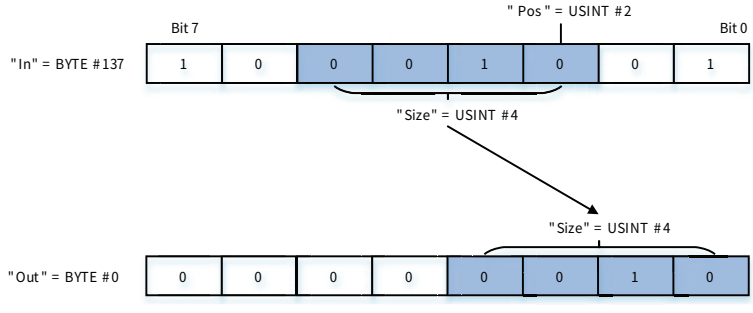
	LD	ST
Defined variable		
Program		
Running result		
Work principle		

```
VAR
    byIn        : BYTE  := 137;
    usiPos      : USINT := 2;
    usiSize     : USINT := 4;
    byOut       : BYTE;
    xResult     : BOOL;
END_VAR
```




```
xResult := ReadNbit_BYTE(In := byIn,
        Pos := usiPos,
        Size := usiSize,
        Out => byOut
        );
```

byIn	BYTE	16#89
usiPos	USINT	16#02
usiSize	USINT	16#04
byOut	BYTE	16#02
xResult	BOOL	TRUE



■ Precautions

Use the same data type for In and Out. Otherwise, the return value is FALSE.

When the value of Size is 0, the value of Out is 16#0.

In the following cases, an error occurs, the return value is FALSE, and the value of Out is 16#0:

· The value of Pos exceeds the valid range.

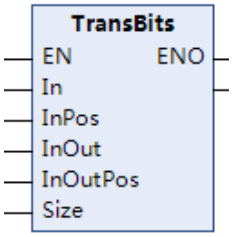
· The value of Size exceeds the valid range.

· The bit string in In does not have enough bits for the number of bits specified by Size from the position specified by Pos.

3.8. 4 TransBits

This instruction moves multiple bits in a bit string.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
TransBits	Move bits	FC		TransBits (In :=, InPos :=, InOut :=, InOutPos :=, Size :=);

■ Variables

Input variables

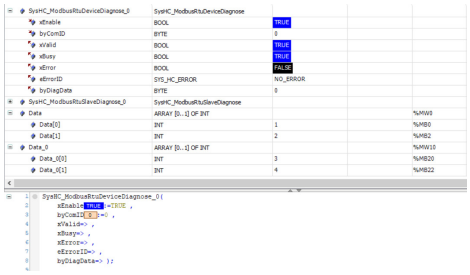
Input Variable	Name	Data Type	Value Range	Initial Value	Description
In	Move source	-	Depends on data types	-	Move source
InPos	Move source bit	USINT	0 to No. of bits in In − 1	-	Position of bit in In to move bits
InOut	Move destination	-	Depends on data types	-	Move destination
InOutPos	Move destination bit	USINT	0 to No. of bits in InOut − 1	-	Position of bit in InOut to receive bits
Size	Bits to move	USINT	0 to minimum number of bits of In and InOut	-	Number of bits to move

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In	-	√	√	√	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
InPos	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-
InOut	-	√	√	√	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
InOutPos	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Size	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-

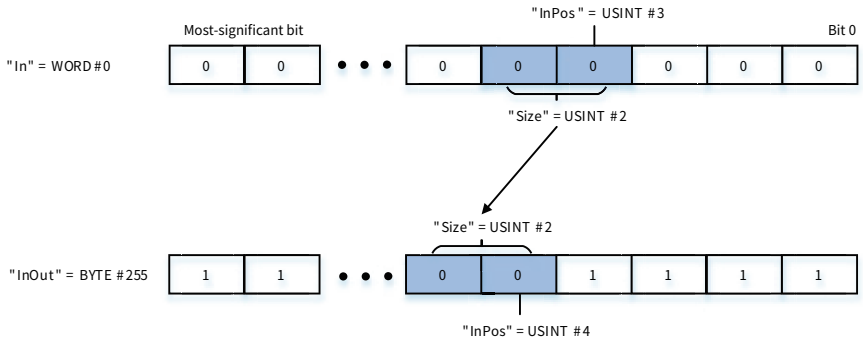
■ Function

This instruction moves Size bits from the bit position InPos in move source In to the bit position InOutPos in move destination InOut.

■ Program example

<table>
<tr><th></th><th>LD</th><th>ST</th></tr>
<tr><td>Defined variable</td><td colspan="2">  </td></tr>
<tr><td>Program</td><td>
<table>
<tr><td>D0</td><td>INT</td><td>Decimal</td><td>1</td></tr>
<tr><td>D1</td><td>INT</td><td>Decimal</td><td>2</td></tr>
<tr><td>D10</td><td>INT</td><td>Decimal</td><td>3</td></tr>
<tr><td>D11</td><td>INT</td><td>Decimal</td><td>4</td></tr>
</table>
</td><td>

```
xResult := TransBits(In := wIn,
        InPos := usiInPos,
        InOut := byInOut,
        InOutPos := usiInOutPos,
        Size := usiSize
        );
```

</td></tr>
<tr><td>Running result</td><td colspan="2">
<table>
<tr><td>wIn</td><td>WORD</td><td>16#0000</td></tr>
<tr><td>usiInPos</td><td>USINT</td><td>16#03</td></tr>
<tr><td>byInOut</td><td>BYTE</td><td>16#CF</td></tr>
<tr><td>usiInOutP...</td><td>USINT</td><td>16#04</td></tr>
<tr><td>usiSize</td><td>USINT</td><td>16#02</td></tr>
<tr><td>xResult</td><td>BOOL</td><td>TRUE</td></tr>
</table>
</td></tr>
<tr><td>Work principle</td><td colspan="2"> 
</td></tr>
</table>

■ Precautions

When the value of Size is 0, no bit is moved.

Bits that are not moved in InOut do not change.

In the following cases, an error occurs, the return value is FALSE, and the output variables do not change:

· The value of InPos exceeds the valid range.

· The value of InOutPos exceeds the valid range.

· The value of Size exceeds the valid range.

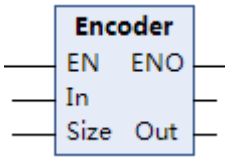
· The bit string in In does not have enough bits for the number of bits specified by Size from the position specified by InPos.

· The bit string in InOut does not have enough bits for the number of bits specified by Size from the position specified by InOutPos.

3.8. 5 Encoder

This instruction finds the position of the TRUE bit in array elements that consist of a maximum of 256 bits.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
Encoder	Bit encoder	FC		Encoder(In :=, Size :=, Out :=);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
In[] (array)	Array to convert	-	Depends on data types	-	Array to convert
Size	Bits to convert	USITN	0 to 8	1	Number of bits to convert

Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
Out	Return value	BYTE	Depends on data types	0	Conversion result

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In	√	√	√	√	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Size	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Out	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

■ Function

This instruction finds the position of a TRUE bit in a range of bits in array to convert In[].

The instruction looks for a TRUE bit in 2^{Size} bits from In[0]. The position of the TRUE bit in this range is expressed in binary and stored in the Size bits in the lower bits of conversion result Out. FALSE is stored in the remaining bits of Out. If there are multiple TRUE bits in the specified range, the position of the highest TRUE bit is found.

Always attach the element number to the input parameter that is passed to In[], such as Array[3].

Consider an example for when Size is USINT#4 and In[] is a Byte array.

Size is USINT#4, so the range in which to find a TRUE bit is 2^4, or 16 bits, from In[0]. In the following diagram, the ninth bit in the range is TRUE.

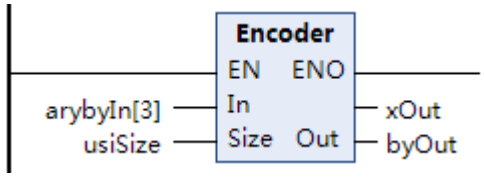
■ Program example

Size is USINT#4, so 2#1001 (9) is stored in the lower 4 bits of Out. FALSE is stored in the upper four bits of Out.

<table>
<tr><th></th><th>LD</th><th>ST</th></tr>
<tr><td>Defined variable</td><td colspan="2">

```
VAR
    arybyIn : ARRAY[1..4] OF BYTE := [7,9,0,2];
    usiSize : USINT := 4;
    byOut   : BYTE;
    xOut    : BOOL;
END_VAR
```

</td></tr>
<tr><td>Program</td><td>



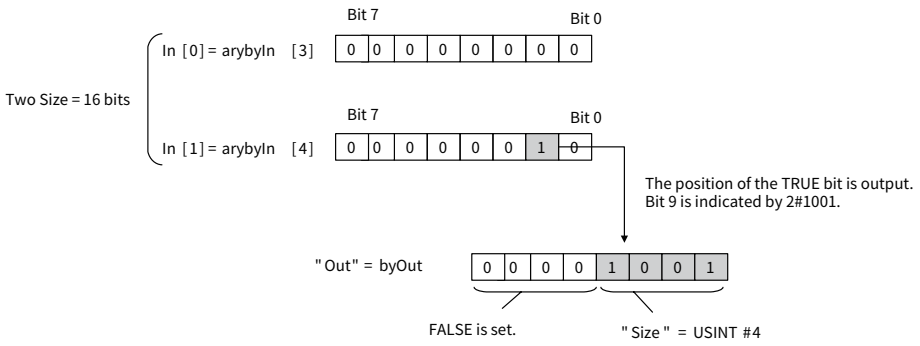
</td><td>

```
Encoder(
    In := arybyIn[3],
    Size := usiSize,
    Out => byOut,
);
```

</td></tr>
<tr><td>Running result</td><td colspan="2">

arybyIn	ARRAY [1..4] OF BYTE	
arybyIn[1]	BYTE	7
arybyIn[2]	BYTE	9
arybyIn[3]	BYTE	0
arybyIn[4]	BYTE	2
usiSize	USINT	4
byOut	BYTE	9
xOut	BOOL	TRUE

</td></tr>
<tr><td>Work principle</td><td colspan="2">



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

■ Precautions

· When the value of Size is 0, all bits in Out change to FALSE.

· When the value of Size exceeds the valid range, an error occurs, the return value is FALSE, and the value of Out does not change.

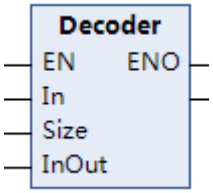
· When the value of Size exceeds the number of bits in the array elements of In[], an incorrect value may be obtained.

· When the value of all bits in In[] that are specified by Size changes to FALSE, an error occurs, the return value is FALSE, and the value of Out does not change.

3.8. 6 Decoder

This instruction sets the specified bit to TRUE and the other bits to FALSE in array elements that consist of a maximum of 256 bits.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
Decoder	Bit decoder	FC		Decoder (In :=, Size :=, InOut :=,);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
In	Conversion bit position	BYTE	Depends on data types	0	Bit position to convert
Size	Bits to convert	USITN	0 to 8	1	Number of bits to convert

In-out variables

In-Out Variable	Name	Data Type	Value Range	Initial Value	Description
InOut[]	Array to convert	-	Depends on data types	-	Array to convert

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Size	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-
InOut[]	√	√	√	√	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

■ Function

This instruction converts array elements with 2^{Size} bits that start from InOut[0] in array to convert InOut[]. It sets the specified bit to TRUE and other bits to FALSE.

The bit to make TRUE is specified by the lower Size bits of conversion bit position In.

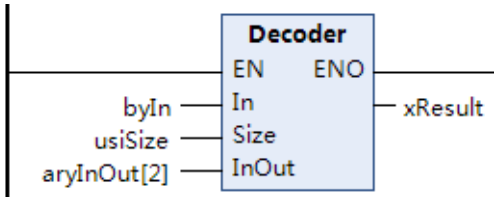
Always attach the element number to the input parameter that is passed to InOut[], such as Array[3].

■ Program example

<table>
<tr><th></th><th>LD</th><th>ST</th></tr>
<tr><td>Defined variable</td><td colspan="2">

```
VAR
    byIn        : BYTE := 9;
    usiSize     : USINT := 5;
    aryInOut    : ARRAY [1..10] OF WORD := [10(65535)];
    xResult     : BOOL := FALSE;
END_VAR
```

</td></tr>
<tr><td>Program</td><td>



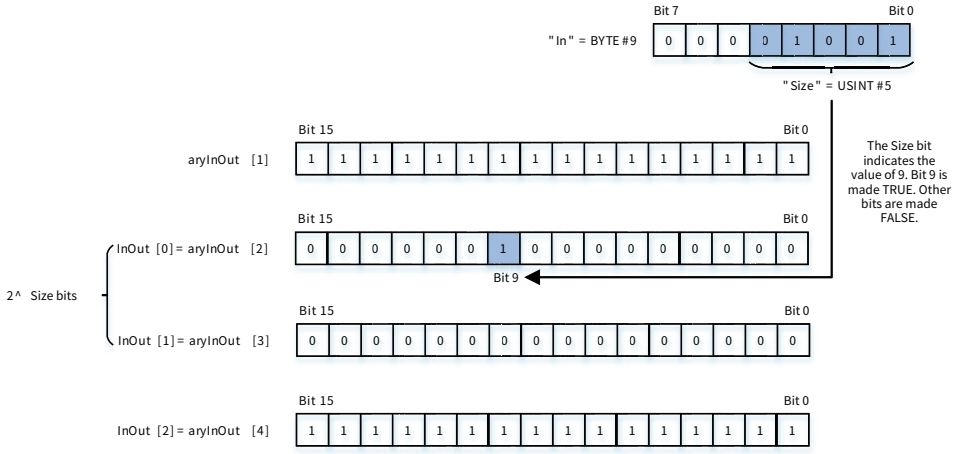
</td><td>

```
xResult := Decoder(In := byIn,
           Size := usiSize,
           InOut := aryInOut[2]
           );
```

</td></tr>
<tr><td>Running result</td><td colspan="2">

byIn	BYTE	9
usiSize	USINT	5
aryInOut	ARRAY [1..10] OF WORD	
aryInOut[1]	WORD	65535
aryInOut[2]	WORD	512
aryInOut[3]	WORD	0
aryInOut[4]	WORD	65535
aryInOut[5]	WORD	65535
aryInOut[6]	WORD	65535
aryInOut[7]	WORD	65535
aryInOut[8]	WORD	65535
aryInOut[9]	WORD	65535
aryInOut[10]	WORD	65535
xResult	BOOL	TRUE

</td></tr>
<tr><td>Work principle</td><td colspan="2">



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

■ Precautions

· When the value of Size exceeds the number of bits in the array elements of InOut, the return value is FALSE and InOut[] does not change.

· When the value of Size exceeds the valid range, the return value is FALSE and InOut[] does not change.

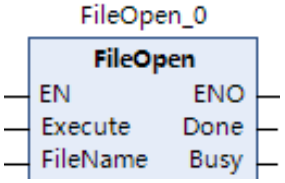
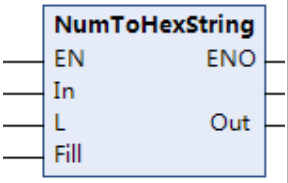
· When the value of Size is 0, all bits in InOut[] change to FALSE.

3.8. 7 NumToDecString and NumToHexString

NumToDecString: Converts an integer to a fixed-length decimal text string.

NumToHexStirng: Converts an integer to a fixed-length hexadecimal text string.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
NumToDecString	Integer-to-fixed-length decimal text string conversion	FC		NumToDecString (In := , UiL := , Fill := , Out:=);
NumToHexStirng	Integer-to-fixed-length hexadecimal text string conversion	FC		NumToHexString (In := , UiL := , Fill := , Out:=);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
In	Integer	-	Depends on data types	-	Integer
L	Number of characters	USITN	0 to 1985	1	String
Fill	Fill character	_eFill_CHR	1 (_BLANK), 2 (_ZERO)	1 (BLANK)	Fill character

Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
Out	String	-	Depends on data types	-	String

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In	-	-	-	-	-	√	√	√	√	√	√	√	√	-	-	-	-	-	-	-
L	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-
Fill	See "Function" for the enumerators for the enumerated type _eFill_CHR.																			
Out	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√

■ Function

NumToDecString

This instruction converts integer In to a decimal text string of UTF-8 alphanumeric characters. If In contains a negative value, a minus sign (－) is added to the front of the text string.

NumToHexString

This instruction converts integer In to a hexadecimal text string of UTF-8 alphanumeric characters. If In is

negative, it is expressed in its two's complement (bits inverted and then 1 added).

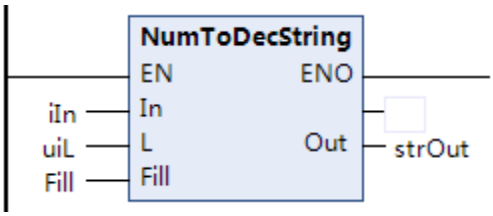
For either instruction, the number of characters in text string Out is adjusted to number of characters L. If there are not enough characters, the upper digits are filled with fill character Fill. If the number of characters in the conversion result exceeds L, L characters from the lower digits of the conversion result are assigned to Out.

The data type of Fill is enumerated type _eFill_CHR. The following table lists the meaning of the enumerators.

Enumerator	Meaning
1 (_BLANK)	Blank character
2 (_ZERO)	0

■ Program example

LD



ST

```
NumToDecString(
    In:= iIn,
    L:= uiL,
    Fill:= Fill,
    Out=> strOut);
```

[In] = sin = INT#128, [L] = uiL = UINT#8, [Fill] = Fill = 1

[Out] = strOut = | | | | | | 1 | 2 | 8 |

[In] = sin = INT#-128, [L] = uiL = UINT#8, [Fill] = Fill = 1

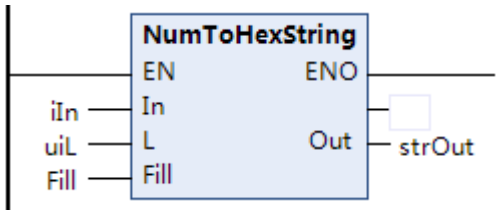
[Out] = strOut = | | | | | - | 1 | 2 | 8 |

[In] = sin = INT#-128, [L] = uiL = UINT#8, [Fill] = Fill = 2

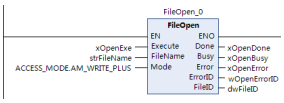
[Out] = strOut = | - | 0 | 0 | 0 | 0 | 1 | 2 | 8 |

The following examples are for the NumToHexString instruction.

LD



ST



[In] = sin = INT#128, [L] = uiL = UINT#8, [Fill] = Fill = 1

[Out] = strOut = | | | | | | | 8 | 0 |

[In] = sin = INT#128, [L] = uiL = UINT#8, [Fill] = Fill = 2

[Out] = strOut = | 0 | 0 | 0 | 0 | 0 | 0 | 8 | 0 |

[In] = sin = INT#-128, [L] = uiL = UINT#8, [Fill] = Fill = 1

[Out] = strOut = | F | F | F | F | F | F | 8 | 0 |

■ Precautions

When the value of L is 0, the value of Out does not change.

When the number of characters in the conversion result exceeds the value of L, L characters from the lower characters of the conversion result are stored in Out. The following is an example.

Instruction	Value of In	Value of L	Value of Out
NumToDecString	128	2	28
NumToHexString			80

· When the value of L exceeds the valid range, an error occurs, the return value is FALSE, and the value of Out does not change.

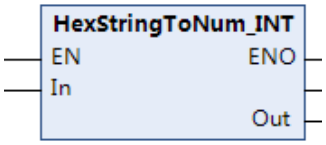
· When the value of Fill exceeds the valid range, the program connection fails.

· When the conversion result exceeds the range of Out, the output value is incorrect.

3.8. 8 HexStringToNum_**

This instruction converts a hexadecimal text string to an integer.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
HexStringToNum_**	Hexadecimal text string-to-number conversion group	FC		HexStringToNum_INT(In := , Out=>);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
In	Hexadecimal text string	STRING	Depends on data types	-	Hexadecimal text string

Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
Out	Integer	-	Depends on data types	-	Integer

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√
Out	-	-	-	-	-	√	√	√	√	√	√	√	√	-	-	-	-	-	-	-

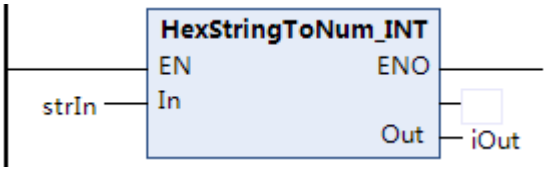
■ Function

This instruction converts hexadecimal text string In to an integer. Any spaces (16#20) or 0 (16#30) in the upper digits are ignored. Underlines (16#5F) in the text string are ignored.

The instruction name is determined by the data types of Out. For example, if Out is of the INT data type, the instruction name is HexStringToNum_INT.

■ Program example

LD



ST

```
HexStringToNum_INT(
    In := strIn,
    Out => iOut);
```

[In] = strOut = | | | | | | | 8 | 0 |

[Out] = sin = INT#128

[In] = strOut = | | | | | | - | 8 | 0 |

[Out] = sin = INT#-128

[In] = strOut = | - | 0 | 0 | 0 | 0 | 0 | 0 | F |

[Out] = sin = INT#-15

■ Precautions

· Even if the conversion result exceeds the valid range of Out, no error occurs but the value of Out is an illegal value.

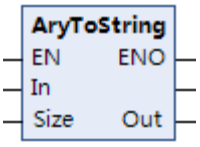
· When the content of In does not end with NULL, the conversion may fail.

· When the content of In includes characters that cannot be converted to numbers, an error occurs, the return value is FALSE, and the value of Out does not change.

3.8. 9 AryToString

This instruction converts a Byte array to a text string.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
AryToString	Array-to-text string conversion	FC		AryToString (In :=, Size :=, Out =>);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
In[] (array)	Byte array	-	Depends on data types	-	Byte array, with a maximum of 1985 elements

Input Variable	Name	Data Type	Value Range	Initial Value	Description
Size	Number of elements to convert	UINT	Depends on data types	1	Number of elements in the In[] array to convert

Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
Out	String	STRING	Depends on data types	-	String

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Size	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-
Out	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√

■ Function

· This instruction takes elements of a Byte array, from In[0] as character codes and stores them in text string Out. NULL is added to the end of Out.

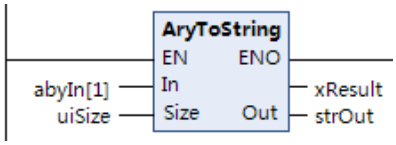
· Size specifies the number of elements of In[] to convert.

· When there is a NULL character between In[0] and In[Size − 1], only the content before NULL is stored in Out.

■ Program example

	LD	ST

Defined variable	VAR
	abyIn : ARRAY [1..5] OF BYTE := [48,49,65,66,67];
	uiSize : UINT := 4;
	strOut : STRING;
	xResult : BOOL;
	END_VAR	

Program		xResult := AryToString(In := abyIn[1], Size := uiSize, Out => strOut);

Running result	abyIn	ARRAY [1..5] OF BYTE	
	abyIn[1]	BYTE	48
	abyIn[2]	BYTE	49
	abyIn[3]	BYTE	65
	abyIn[4]	BYTE	66
	abyIn[5]	BYTE	67
	uiSize	UINT	4
	strOut	STRING	'01AB'
	xResult	BOOL	TRUE

■ Precautions

· When the value of Size exceeds the In[] array, the return value is FALSE and Out does not change.

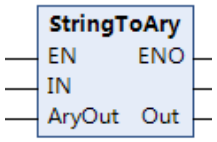
· In[] must be of the Byte data type. Otherwise, an error occurs during compilation.

· When the value of Size is 0, Out is a text string containing only the NULL character.

3.8. 10 StringToAry

This instruction converts a text string to a Byte array.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
StringToAry	Text string-to-array conversion	FC		StringToAry(In :=, AryOut :=, Out :=);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
In	String	STRING	A maximum of 1985 characters	-	String

In-out variables

In-Out Variable	Name	Data Type	Value Range	Initial Value	Description
AryOut[]	Byte array	BYTE	Depends on data types	-	Byte array

Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
Out	Number of bytes after conversion	UINT	0 to 1985	-	Number of bytes after conversion

	Bool-ean	Bit String				Integer								Real Num-ber		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√
AryOut[]	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Out	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-

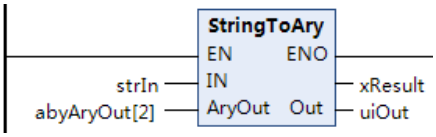
■ Function

This instruction takes the character codes in text string In as numbers and stores them individually in a Byte array, AryOut[]. The number of bytes after conversion is stored in Out.

■ Program example

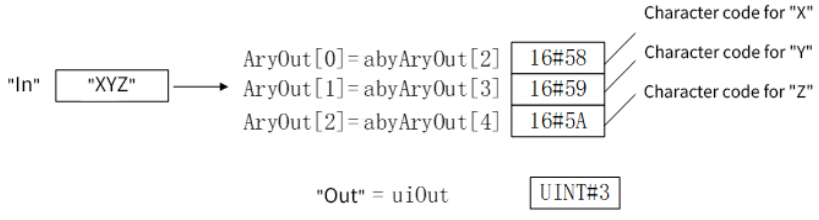
	LD	ST
Defined variable		
Program		
Running result		
Work principle		

```
VAR
    strIn       : STRING := 'XYZ';
    abyAryOut   : ARRAY [0..5] OF BYTE;
    uiOut       : UINT;
    xResult     : BOOL;
END_VAR
```




```
xResult := StringToAry(IN:= strIn,
    AryOut:= abyAryOut[2],
    Out=> uiOut);
```

strIn	STRING	'XYZ'
abyAryOut	ARRAY [0..5] OF BYTE	
abyAryOut[0]	BYTE	16#00
abyAryOut[1]	BYTE	16#00
abyAryOut[2]	BYTE	16#58
abyAryOut[3]	BYTE	16#59
abyAryOut[4]	BYTE	16#5A
abyAryOut[5]	BYTE	16#00
uiOut	UINT	16#0003
xResult	BOOL	TRUE



■ Precautions

· NULL at the end of In is not stored in AryOut[].

· When the In text string contains only the NULL character, the value of Out is 0 and AryOut[] does not change.

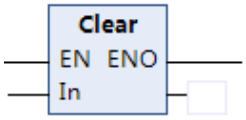
· When the number of bytes in In is greater than the number of elements in AryOut[], the return value is FALSE and RoundUpAryOut[] does not change.

· When the number of bytes in In exceeds the upper limit 1985, excessive characters are truncated and only 1985 characters are converted.

3.8. 11 Clear

This instruction initializes variables.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
Clear	Initialization	FC		Clear(In:=);

■ Variables

In-out variables

In-Out Variable	Name	Data Type	Value Range	Initial Value	Description
In	Data to initialize	-	Depends on data types	-	Data to initialize

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√
	An array or array element can also be specified.																			

■ Function

· This instruction initializes the value of the data to initialize In to the default initial value of each data type.

· The default values for the data types are listed in the following table.

Data Type	Initial Value
BOOL	FALSE
BYTE, WORD, DWORD, or LWORD	16#0
USINT, UINT, UDINT, ULINT, SINT, INT, DINT, LINT, REAL, or LREA	0
TIME	T#0ms
DATE	D#1970-1-1
TOD	TOD#0:0:0
DT	DT#1970-1-1-0:0:0
STRING	''

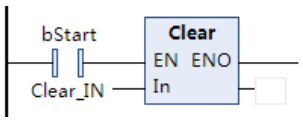
When In is an array, an array element, or a variable, the following processing is performed.

In	Processing
Array	All elements in the array are initialized.
Array element	Only the specified element is initialized.
Structure	All members in the structure are initialized.
Structure member	Only the specified member is initialized.

■ Program example

<table>
<tr><th></th><th>LD</th><th>ST</th></tr>
<tr><td>Defined variable</td><td colspan="2">

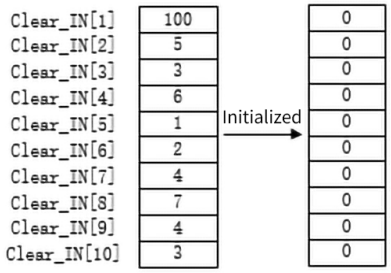
```
PROGRAM POU
VAR
    bStart   : BOOL;
    Clear_IN: ARRAY[1..10] OF UINT:=[100,5,3,6,1,2,4,7,4,3];
END_VAR
```

</td></tr>
<tr><td>Program</td><td> 
</td><td>

```
Clear(
    In:=Clear_IN,
);
```

</td></tr>
<tr><td>Running result</td><td colspan="2">

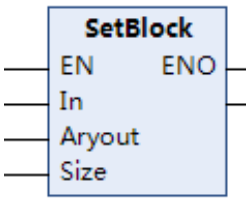
bStart	BOOL	TRUE
Clear_IN	ARRAY [1..10] OF UINT	
Clear_IN[1]	UINT	0
Clear_IN[2]	UINT	0
Clear_IN[3]	UINT	0
Clear_IN[4]	UINT	0
Clear_IN[5]	UINT	0
Clear_IN[6]	UINT	0
Clear_IN[7]	UINT	0
Clear_IN[8]	UINT	0
Clear_IN[9]	UINT	0
Clear_IN[10]	UINT	0

</td></tr>
<tr><td>Work principle</td><td colspan="2"> 
</td></tr>
</table>

3.8. 12 SetBlock

This instruction moves values of variables or constants to multiple array elements.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
SetBlock	Block set	FC		SetBlock(In := , AryOut := , Size :=);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
In	Move source	-	Depends on data types	-	Move source
Size	Number of elements	UINT	0 to 65535	1	Number of array elements to move

In-out variables

In-Out Variable	Name	Data Type	Value Range	Initial Value	Description
AryOut[] (array)	Move destination array	-	Depends on data types	-	Move destination array

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√
	An enumeration, a structure, or a structure member can also be specified.																			
Size	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-
AryOut[]	Array with elements that have the same data type as In																			

■ Function

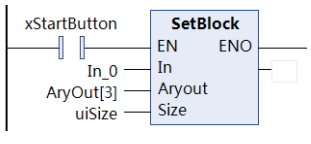
This instruction moves the value of move source In to Size positions in move destination array AryOut[] starting from AryOut[0].

■ Program example

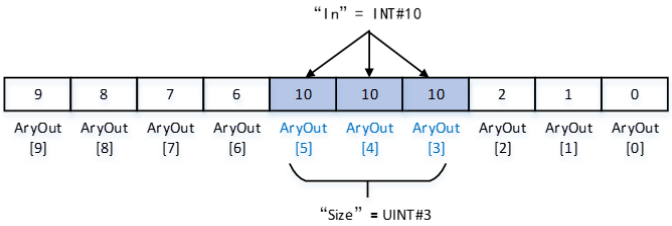
Defined variable

```
VAR
    xStartButton: BOOL;
    AryOut      : ARRAY [0..9] OF INT := [0,1,2,3,4,5,6,7,8,9];
    In_0        : INT := 10;
    uiSize      : UINT := 3;
END_VAR
```

Program




```
IF xStartButton = TRUE THEN
    SetBlock(In := In_0,
        Aryout := AryOut[3],
        Size := uiSize,
        );
END_IF
```

Running result	AryOut / ARRAY [0..9] OF INT AryOut[0] INT 0 AryOut[1] INT 1 AryOut[2] INT 2 AryOut[3] INT 10 AryOut[4] INT 10 AryOut[5] INT 10 AryOut[6] INT 6 AryOut[7] INT 7 AryOut[8] INT 8 AryOut[9] INT 9	AryOut / ARRAY [0..9] OF INT AryOut[0] INT 0 AryOut[1] INT 1 AryOut[2] INT 2 AryOut[3] INT 10 AryOut[4] INT 10 AryOut[5] INT 10 AryOut[6] INT 6 AryOut[7] INT 7 AryOut[8] INT 8 AryOut[9] INT 9
Work principle		

■ Precautions

· When the data types of In and AryOut[] are different, an error occurs during compilation.

· When the data types of In and AryOut[] are String, their sizes must be the same.

· When the value of Size is 0, the return value is TRUE and AryOut[] does not change.

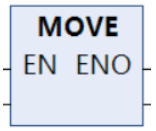
· When the value of Size exceeds the AryOut[] array, the return value is FALSE and AryOut[] does not change.

· When the data types of In and AryOut[] are not supported, an error occurs during compilation.

3.8. 13 MOVE

This instruction moves the value of a variable to another variable of the same type.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
MOVE	Move	FC		a2:= MOVE(a1);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
In	Move source	-	Depends on data types	-	Input data address

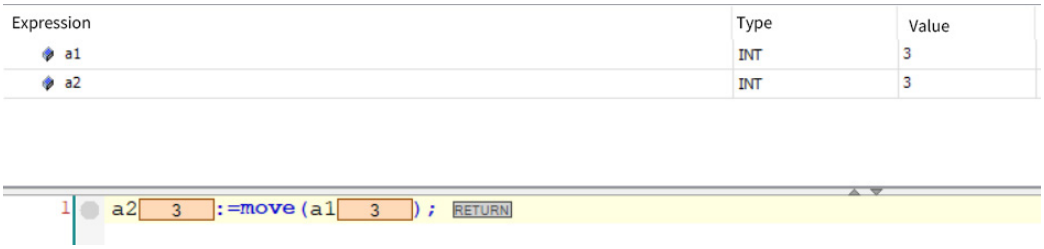
Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
Out	Move destination	-	Depends on data types	-	Output data address

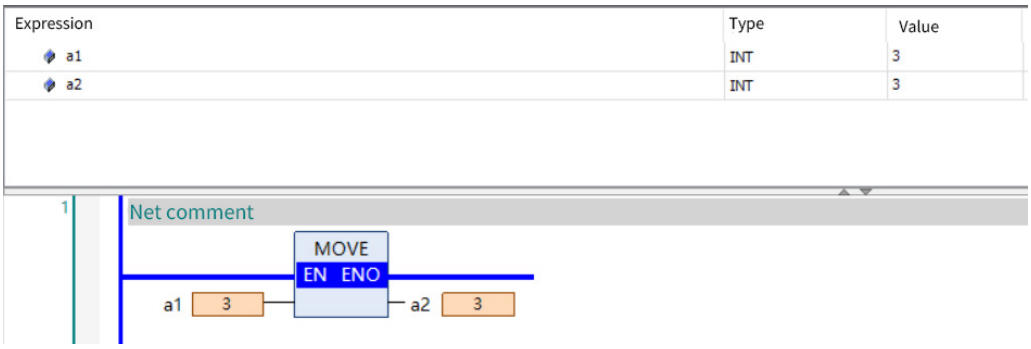
	Bool-ean	Bit String				Integer								Real Num-ber		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√
Out	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√

■ Program example

ST:



LD:



3.8. 14 BMOV

This instruction copies the source data of a specified length to the target data.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
BMOV	Batch move	FC	BMOV EN ENO pbyDataSrc uiSize pbyDataDes	BMOV(pbyDataSrc:= , uiSize:= , pbyDataDes:=);

■ Variables

Input variables

Input Vari-able	Name	Data Type	Value Range	Initial Value	Description
pbyDataSrc	Data copy head address	POINTER TO BYTE	-	-	Data copy head address
uiSize	Variable quantity	UINT	Depends on data types	0	Variable quantity

Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
PbyDataDes	Data storage head address	POINTER TO BYTE	-	-	Data storage head address

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
pbyDataSrc	POINTER TO BYTE																			
uiSize	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-
PbyDataDes	POINTER TO BYTE																			

■ Function

This instruction is contact-driven. It copies values of uiSize variables from the head address specified by pbyDataSrc to uiSize units of the head address specified by pbyDataDes. The valid range for uiSize is 1 to 1024. If the return value is TRUE, the execution is successful. If the return value is FALSE, the execution fails.

■ Program example

ST: Specifies PT1 for array A1 and PT2 for array A2, and uses the BMOV instruction to write the uiSize x 4 values from the head address of array A1 to A2.

```
PROGRAM PLC_PRG
VAR
    A1 : ARRAY [0..3] OF REAL;
    PT1 : POINTER TO REAL := ADR(A1[0]);
    A2 : ARRAY [0..3] OF REAL;
    PT2 : POINTER TO REAL := ADR(A2[0]);
END_VAR
```

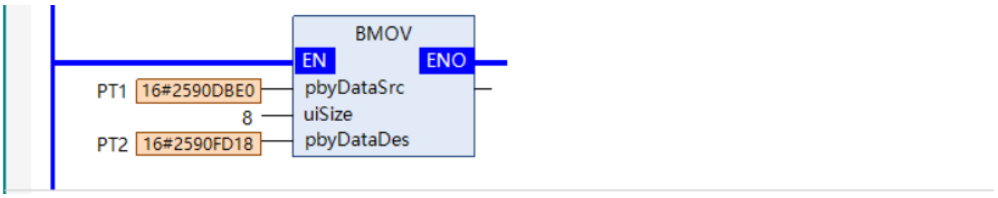
Expression	Type	Value
A1	ARRAY [0..3] OF REAL	
A1[0]	REAL	1
A1[1]	REAL	2
A1[2]	REAL	0
A1[3]	REAL	0
PT1	POINTER TO REAL	16#2590FD28
A2	ARRAY [0..3] OF REAL	
A2[0]	REAL	1
A2[1]	REAL	2
A2[2]	REAL	0
A2[3]	REAL	0
PT2	POINTER TO REAL	16#2590FD38

```
1  PT1 16#2590FD28 :=ADR(A1);
2  PT2 16#2590FD38 :=ADR(A2);
3  BMOV(pbyDataSrc:=PT1 16#2590FD28 , uiSize:=8 , pbyDataDes:=PT2 16#2590FD38 );RETURN
```

LD: Specifies PT1 for array A1 and PT2 for array A2, and uses the BMOV instruction to write the uiSize x 4 values from the head address of array A1 to A2.

```
PROGRAM PLC_PRG
VAR
    A1:ARRAY[0..3] OF REAL;
    PT1:POINTER TO REAL:=ADR(A1[1]);
    A2:ARRAY[0..3] OF REAL;
    PT2:POINTER TO REAL:=ADR(A2[0]);
END_VAR
```

Expression	Type	Value
A1	ARRAY [0..3] OF REAL	
A1[0]	REAL	1
A1[1]	REAL	2
A1[2]	REAL	0
A1[3]	REAL	0
PT1	POINTER TO REAL	16#2590FD28
A2	ARRAY [0..3] OF REAL	
A2[0]	REAL	1
A2[1]	REAL	2
A2[2]	REAL	0
A2[3]	REAL	0
PT2	POINTER TO REAL	16#2590FD38



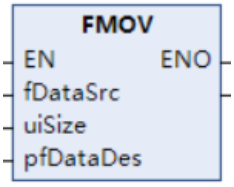
■ Precautions

Write uiSize x 4 values.

3.8. 15 FMOV

This instruction copies the specified bits of the source array to multiple bits of the destination array. For example, it moves the value of A1[1] to A2[0,1,2].

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
FMOV	Multi-point move	FC		FMOV(fDataSrc:=, uiSize:= , pfDataDes:=);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
fDateScr	Data copy head address	REAL	Depends on data types	0	Data copy head address
uiSize	Variable quantity	UINT	Depends on data types	0	Variable quantity

Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
PbyDataDes	Data storage head address	POINTER TO BYTE	-	-	Data storage head address

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
fDateScr	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-
uiSize	-	-	-	-	-	-	√	-	-	-	-	-	-	√	-	-	-	-	-	-
PbyDataDes	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

■ Program example

ST: Specifies PT2 to array A2. The FMOV instruction continuously assigns the second bit (A1[1]) of array A1 to the first three bits (A2[0,1,2]) of A2.

```
PROGRAM PLC_PRG
VAR
    A1:ARRAY[0..3] OF REAL;
    PT1:POINTER TO REAL:=ADR(A1[1]);
    A2:ARRAY[0..3] OF REAL;
    PT2:POINTER TO REAL:=ADR(A2[0]);
END_VAR
```

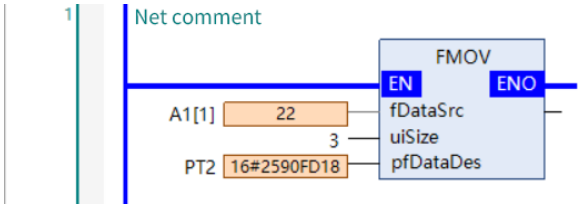
Expression	Type	Value
A1	ARRAY [0..3] OF REAL	
A1[0]	REAL	0
A1[1]	REAL	22
A1[2]	REAL	0
A1[3]	REAL	0
PT1	POINTER TO REAL	16#2590DBE4
A2	ARRAY [0..3] OF REAL	
A2[0]	REAL	22
A2[1]	REAL	22
A2[2]	REAL	22
A2[3]	REAL	0
PT2	POINTER TO REAL	16#2590FD18

```
FMOV(fDataSrc:=A1[0] 0 , uiSize:=3 , pfDataDes:=PT2 16#2590FD38 );RETURN
```

LD: Specifies PT2 to array A2. The FMOV instruction continuously assigns the second bit (A1[1]) of array A1 to the first three bits (A2[0,1,2]) of A2.

```
PROGRAM PLC_PRG
VAR
    A1:ARRAY[0..3] OF REAL;
    PT1:POINTER TO REAL:=ADR(A1[1]);
    A2:ARRAY[0..3] OF REAL;
    PT2:POINTER TO REAL:=ADR(A2[0]);
END_VAR
```

Expression	Type	Value
A1	ARRAY [0..3] OF REAL	
A1[0]	REAL	0
A1[1]	REAL	22
A1[2]	REAL	0
A1[3]	REAL	0
PT1	POINTER TO REAL	16#2590DBE4
A2	ARRAY [0..3] OF REAL	
A2[0]	REAL	22
A2[1]	REAL	22
A2[2]	REAL	22
A2[3]	REAL	0
PT2	POINTER TO REAL	16#2590FD18



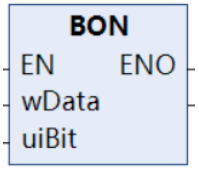
■ Precautions

The valid range for uiSize is 1 to 1024.

3.8. 16 BON

This instruction judges the status of the uiBit bit in wData, saves the result to BON, and output the result from the right side.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
BON	Bit state check	FC		M1:=BON(wData:= , uiBit:=);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
wData	Data source	WORD	Depends on data types	0	Decimal data to judge
uiBit	Bit to judge	UINT	Depends on data types	0	Number of binary bits to judge

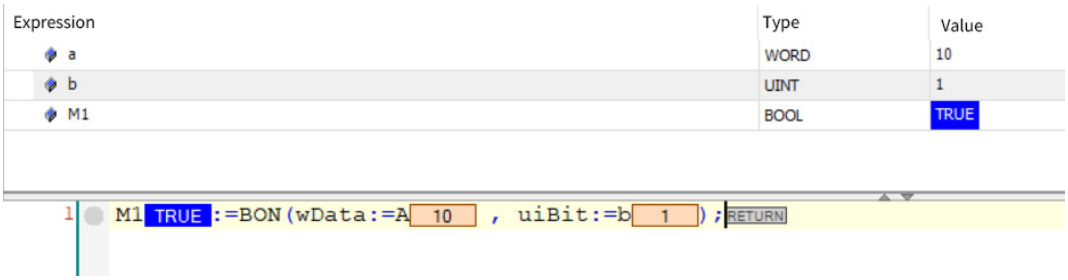
Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
Out	Judgment result	BOOL	[FALSE, TRUE]	FALSE	Status of the uiBit bit in wData

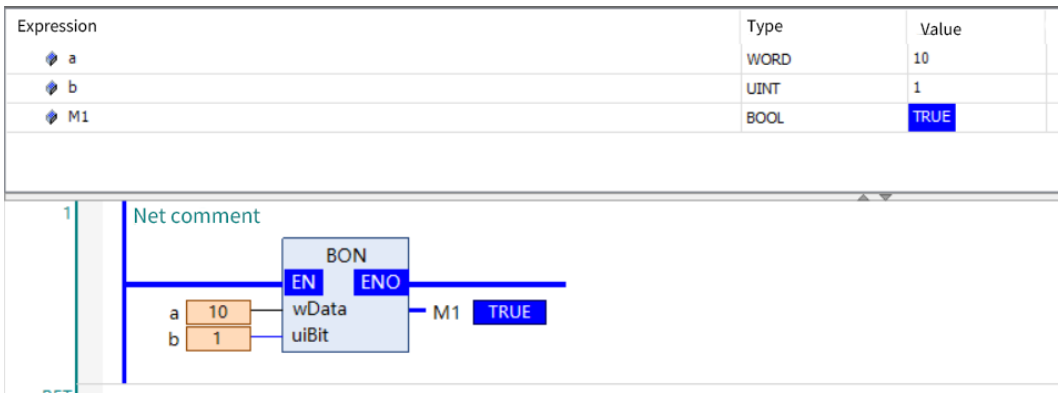
	Bool-ean	Bit String				Integer								Real Num-ber		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
wData	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
uiBit	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-
Out	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

■ Program example

ST: Inputs data a and b to judge, and outputs the corresponding bit status M1.



LD: Inputs data a and b to judge, and outputs the corresponding bit status M1.



3.8. 17 SUM

Collects statistics on the number of 1 in binary data, saves the result to SUM, and outputs the result from the right side.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
SUM	Sum of ON bits	FC	SUM EN ENO wData	a:=SUM(wData:=);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
wData	Data source	WORD	Depends on data types	0	Decimal data to judge

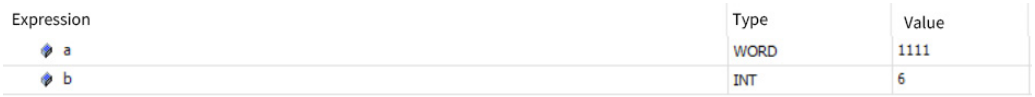
Output variables

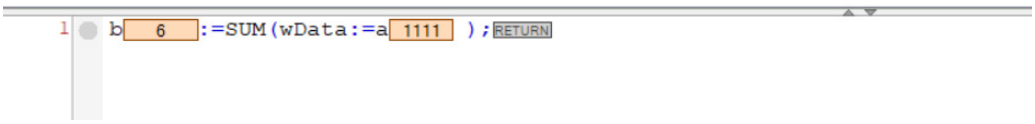
Output Variable	Name	Data Type	Value Range	Initial Value	Description
Out	Judgment result	BOOL	[FALSE, TRUE]	FALSE	Total number of binary bits 1 in wData

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
wData	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Out	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

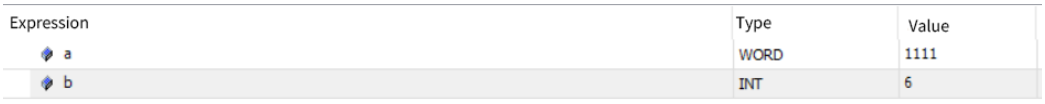
■ Program example

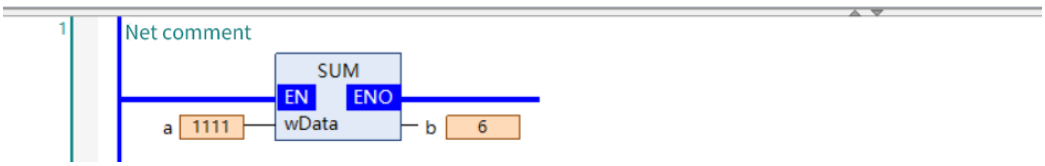
ST: Inputs data a, collects statistics on the number of binary bits 1, and outputs the result b.





LD: Inputs data a, collects statistics on the number of binary bits 1, and outputs the result b.

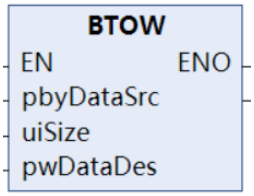




3.8. 18 BTOW

This instruction combines the low-order eight bits (low-order byte) of consecutive 16-bit data.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
BTOW	Byte to word	FC		BTOW(pbyDataSrc:= , uiSize:= , pwDataDes:=);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
PbyDataSrc	Source data	POINTER TO BYTE	-	-	Head address of combined data
uiSize	Number of combined data entries	UINT	Depends on data types	0	Number of combined data entries

Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
PwDataDes	Target data	POINTER TO WORD	-	-	Target address for storing combined data

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
PbyDataSrc	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
uiSize	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-
PbyDataDes	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

■ Function

byDataSrc indicates the source data and uiSize indicates the number of combined data entries. This instruction combines the low-order eight bits of 16-bit values into a new 16-bit value, and saves the new value to the head address specified by pwDataDes. High-order eight bits in pbyDataSrc are ignored.

■ Program example

ST:

```
PROGRAM aa
VAR
    A1:ARRAY[0..3] OF WORD;
    PT1:POINTER TO BYTE:=ADR(A1[0]);
    A2:ARRAY[0..3] OF WORD;
    PT2:POINTER TO REAL:=ADR(A2[0]);
END_VAR
```

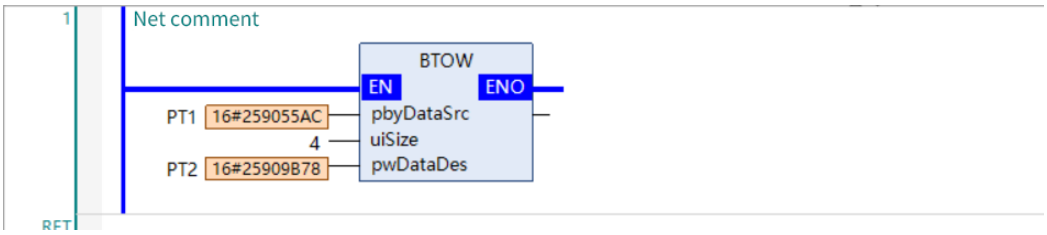
Expression	Type	Value
A1	ARRAY [0..3] OF W...	
A1[0]	WORD	1000
A1[1]	WORD	1000
A1[2]	WORD	1000
A1[3]	WORD	1000
PT1	POINTER TO BYTE	16#259055AC
A2	ARRAY [0..3] OF W...	
A2[0]	WORD	59624
A2[1]	WORD	59624
A2[2]	WORD	0
A2[3]	WORD	0
PT2	POINTER TO REAL	16#25909B78

```
1  BTOW(pbyDataSrc:= PT1 16#25909B80, uiSize:= 4, pwDataDes:= PT2 16#25909B88);
2  RETURN
```

LD:

```
1  PROGRAM aa
2  VAR
3      A1:ARRAY[0..3] OF WORD;
4      PT1:POINTER TO BYTE:=ADR(A1[0]);
5      A2:ARRAY[0..3] OF WORD;
6      PT2:POINTER TO REAL:=ADR(A2[0]);
7  END_VAR
8
```

Expression	Type	Value
A1	ARRAY [0..3] OF W...	
A1[0]	WORD	1000
A1[1]	WORD	1000
A1[2]	WORD	1000
A1[3]	WORD	1000
PT1	POINTER TO BYTE	16#259055AC
A2	ARRAY [0..3] OF W...	
A2[0]	WORD	59624
A2[1]	WORD	59624
A2[2]	WORD	0
A2[3]	WORD	0
PT2	POINTER TO REAL	16#25909B78



3.8. 19 SWAP

This instruction exchanges the high-order eight bits and low-order eight bits in a 16-bit value.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
SWAP	Swap bytes	FC	SWAP EN ENO wData	SWAP(wData:=);

■ Variables

Input variables

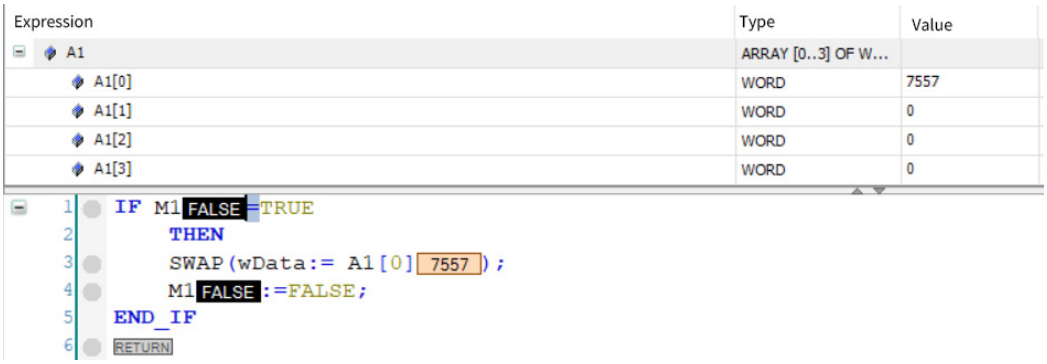
Input Variable	Name	Data Type	Value Range	Initial Value	Description
wData	Operation data	WORD	Depends on data types	0	Data to operate

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
wData	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

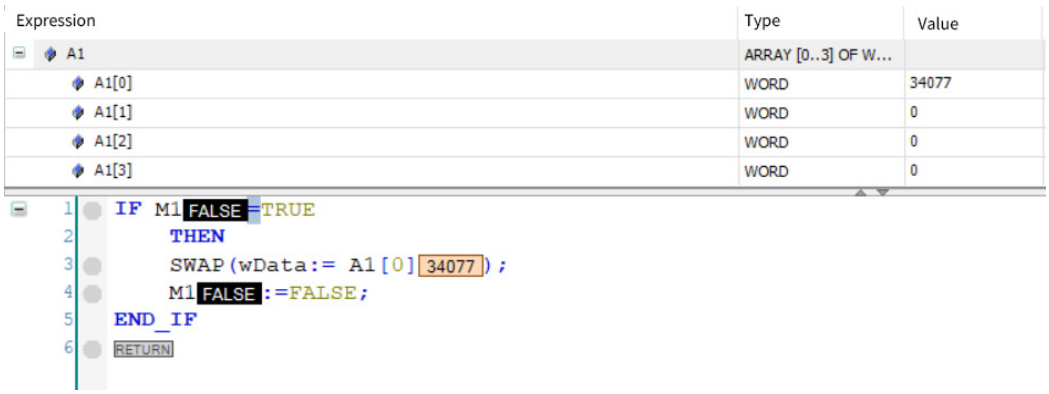
■ Program example

ST:

(1) Switch M1 is not closed.



(2) Switch M1 is closed.

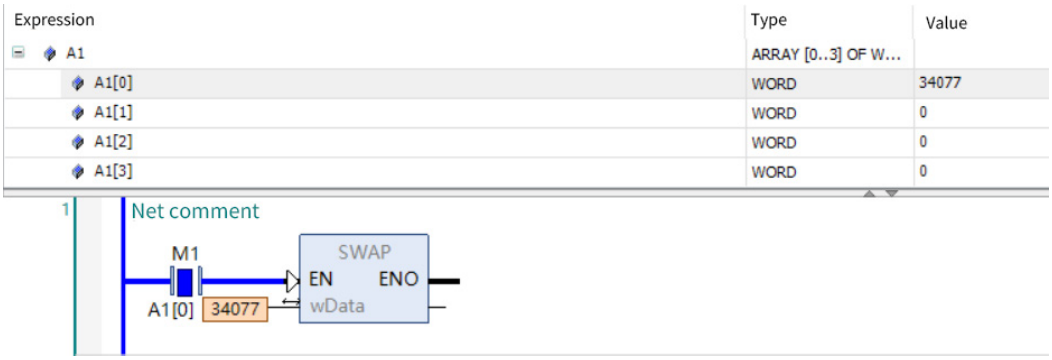


LD:

(1) Switch M1 is not closed.



(2) Switch M1 is closed.



3.8. 20 SWAP16

This instruction exchanges the high-order eight bits and low-order eight bits in a 16-bit value.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
SWAP16	Byte swap	FC	SWAP16 EN ENO wData	wOut := SWAP16(wData:=);

■ Variables

Input variables

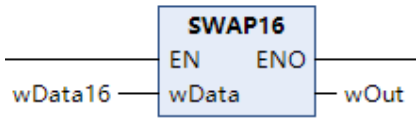
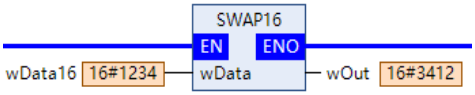
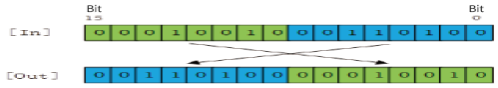
Input Variable	Name	Data Type	Value Range	Initial Value	Description
wData	Operation data	WORD	Depends on data types	0	Data to operate

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
wData	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

■ Function

This instruction exchanges the high-order eight bits and low-order eight bits in a 16-bit value. The return value is the exchange result.

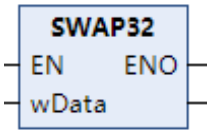
■ Program example

	LD	ST
Defined variable	wData16 :WORD := 16#1234; wOut :WORD;	
Program		wOut := SWAP16(wData:= wData16);
Running result		wOut 16#3412 := SWAP16(wData:= wData16 16#1234);
Work principle		

3.8. 21 SWAP32

This instruction exchanges the high-order 16 bits and low-order 16 bits in a 32-bit value. The high-order eight bits and low-order eight bits in the high-order 16 bits and low-order 16 bits are also exchanged.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
SWAP32	Byte swap	FC		dwOut := SWAP32(wData:=);

■ Variables

Input variables

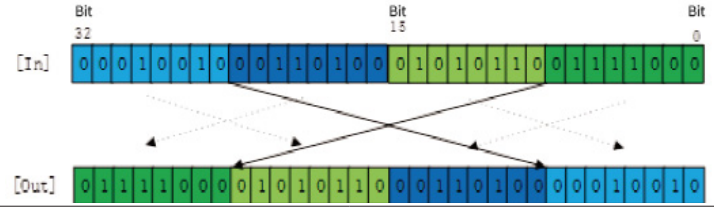
Input Variable	Name	Data Type	Value Range	Initial Value	Description
wData	Operation data	DWORD	Depends on data types	0	Data to operate

	Bool-ean	Bit String				Integer								Real Num-ber		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
wData	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

■ Function

This instruction exchanges the high-order 16 bits and low-order 16 bits in a 32-bit value. The high-order eight bits and low-order eight bits in the high-order 16 bits and low-order 16 bits are also exchanged. The return value is the exchange result.

■ Program example

	LD	ST
Defined variable	dwData32 :DWORD := 16#12345678; dwOut :DWORD;	
Program	SWAP32 (EN, ENO, wData); dwData32 → wData, dwOut ←	dwOut := SWAP32(wData:= dwData32);
Running result	SWAP32 (EN, ENO, wData); dwData32 16#12345678 → wData, dwOut ← 16#78563412	dwOut 16#78563412 := SWAP32(wData:= dwData32 16#12345678);
Work principle		

3.8. 22 XCH

This instruction exchanges two floating-point values.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
XCH	Data exchange	FC	XCH (EN, ENO, fDataSrc1, fDataSrc2)	XCH(fDataSrc1:= , fDataSrc2:=);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
FDataSrc1	Switch data 1	REAL	Depends on data types	0	Data to exchange
FDataSrc2	Switch data 2	REAL	Depends on data types	0	Data to exchange

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
FDataSrc1	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-
FDataSrc2	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-

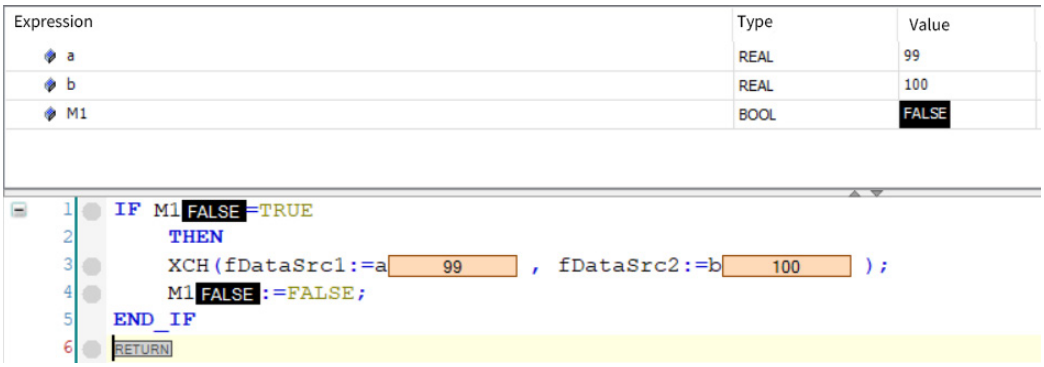
■ Function

This instruction exchanges two floating-point values.

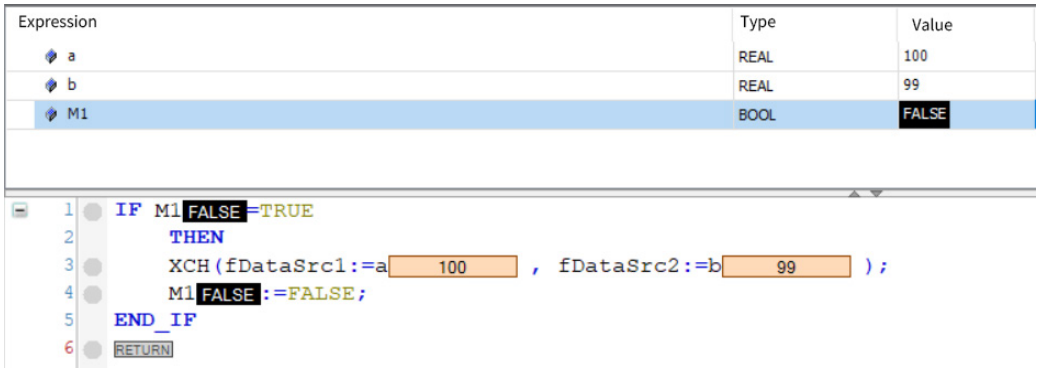
■ Program example

ST:

(1) Switch M1 is not closed.

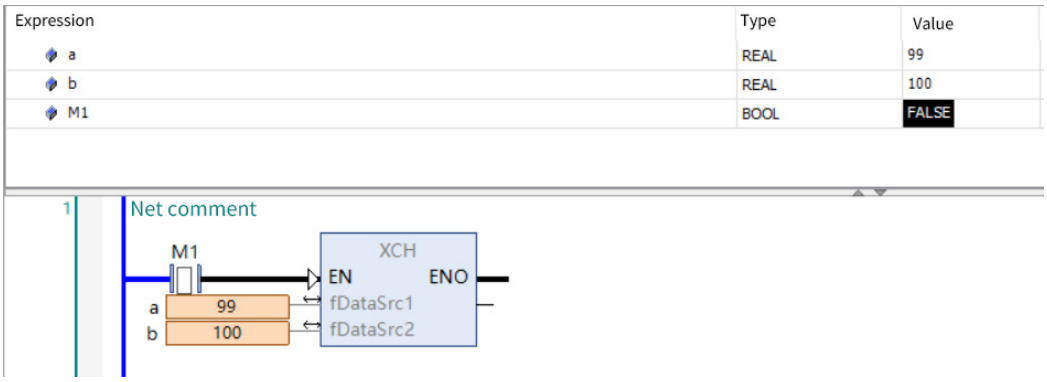


(2) Switch M1 is closed.

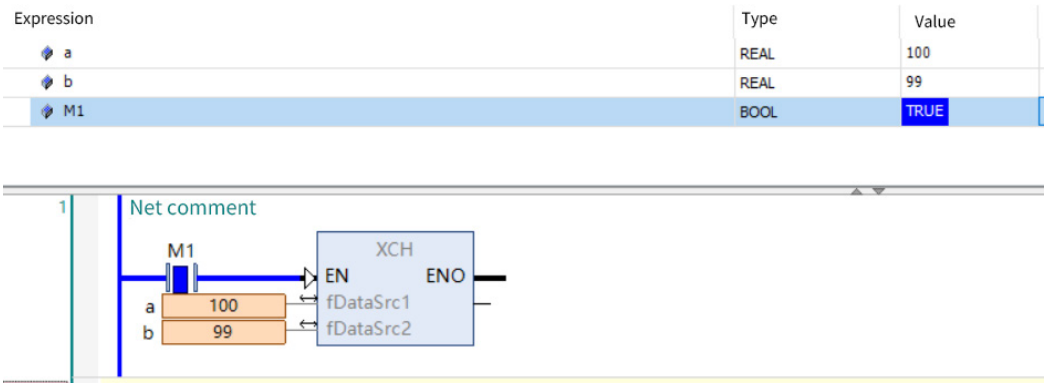


LD:

(1) Switch M1 is not closed.



(2) Switch M1 is closed.



3.8. 23 HEXinASCII_TO_BYTE

■ Instruction description

This instruction converts the source data from the HEXinASCII format to bytes.

Instruction	Name	FB/FC	LD Expression	ST Expression
HEXinASCII_TO_BYTE	ASCII-to-byte conversion	FC	HEXINASCII_TO_BYTE EN ENO W	HEXinASCII_TO_BYTE(W:=);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
W	Input data	WORD	0-FFFF	0	-

Output variables

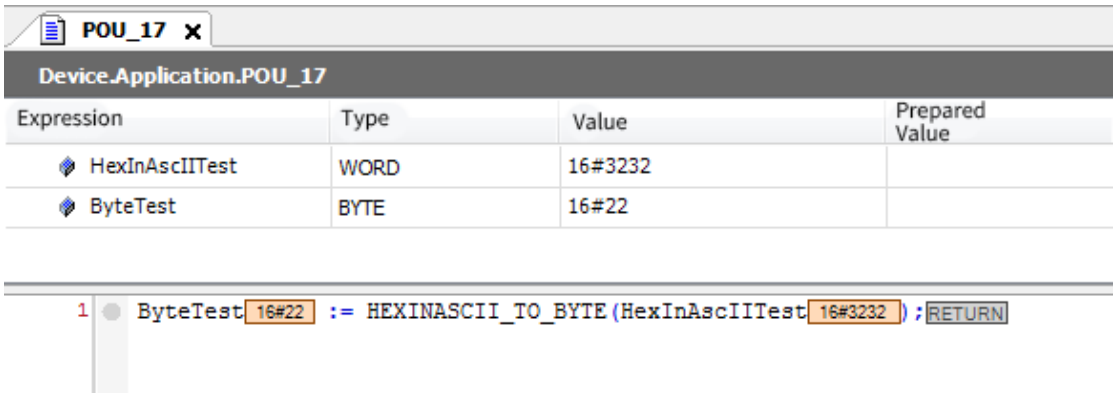
Output Variable	Name	Data Type	Value Range	Initial Value	Description
OUT	Output data	BYTE	0 to 255	0	-

	Bool-ean	Bit String				Integer								Real Num-ber		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
FDataSrc1	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-
FDataSrc2	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-

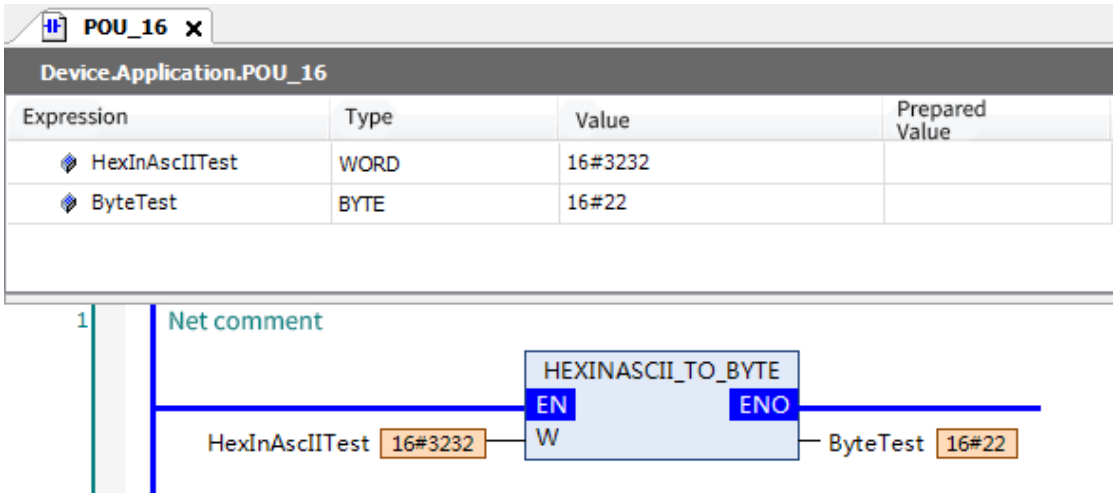
■ Program example

This instruction converts the source data from the HEXinASCII format to bytes.

ST



LD



3.8. 24 BYTE_TO_HEXinASCII

■ Instruction description

This instruction converts the source data from bytes to the HEXinASCII format.

Instruction	Name	FB/FC	LD Expression	ST Expression
BYTE_TO_HEXinASCII	Byte-to-ASCII conversion	FC	BYTE_TO_HEXINASCII EN ENO B	BYTE_TO_HEXinASCII(B:=);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
B	Input data	BYTE	0 to 255	0	-

Output variables

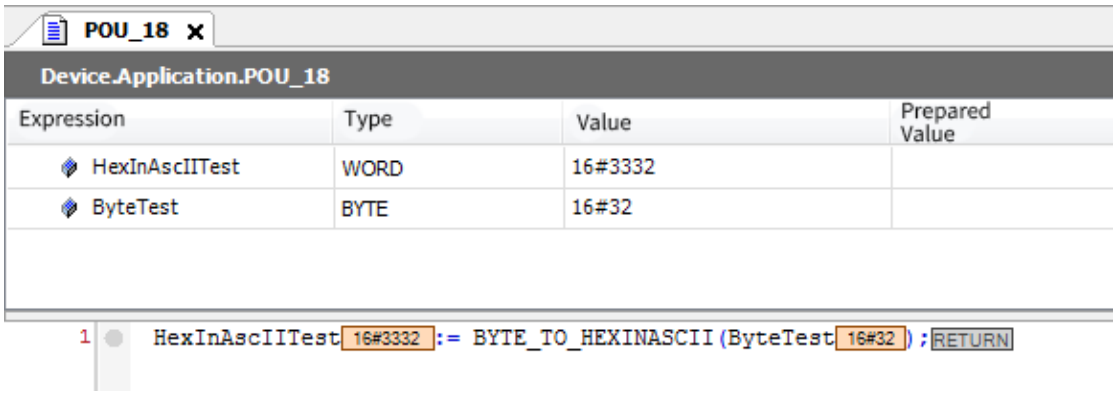
Output Variable	Name	Data Type	Value Range	Initial Value	Description
OUT	Output data	WORD	0-FFFF	0	-

	Bool-ean	Bit String				Integer								Real Num-ber		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
B	-	√	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-
OUT	-	-	√	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-

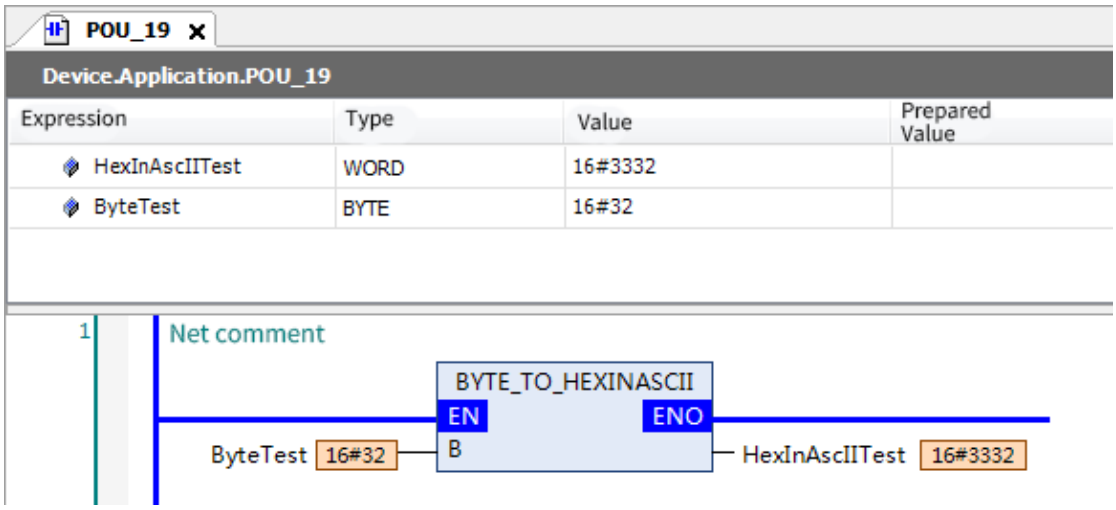
■ Program example

This instruction converts the source data from bytes to the HEXinASCII format.

ST



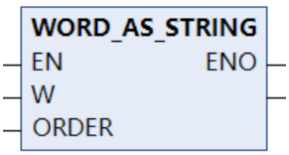
LD



3.8. 25 WORD_AS_STRING

■ Instruction description

This instruction converts the source data from words to strings.

Instruction	Name	FB/FC	LD Expression	ST Expression
WORD_AS_STRING	Word-to-string conversion	FC		WORD_AS_STRING(W:= , ORDER:=);

■ Variables

Input Variable

Input variables	Name	Data Type	Value Range	Initial Value	Description
W	Input data	WORD	0 to 255	0	Input data
ORDER	High/Low-order bit conversion	BOOL	[FALSE, TRUE]	FALSE	Conversion of high/low-order bits for variables

Output variables

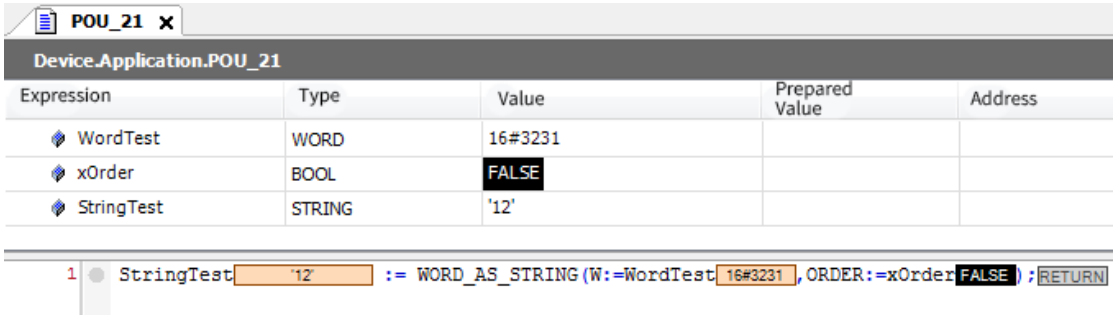
Output Variable	Name	Data Type	Value Range	Initial Value	Description
OUT	Conversion result	STRING	-	''	-

	Bool-ean	Bit String				Integer								Real Num-ber		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
W	-	-	√	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-
ORDER	√	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-
OUT	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

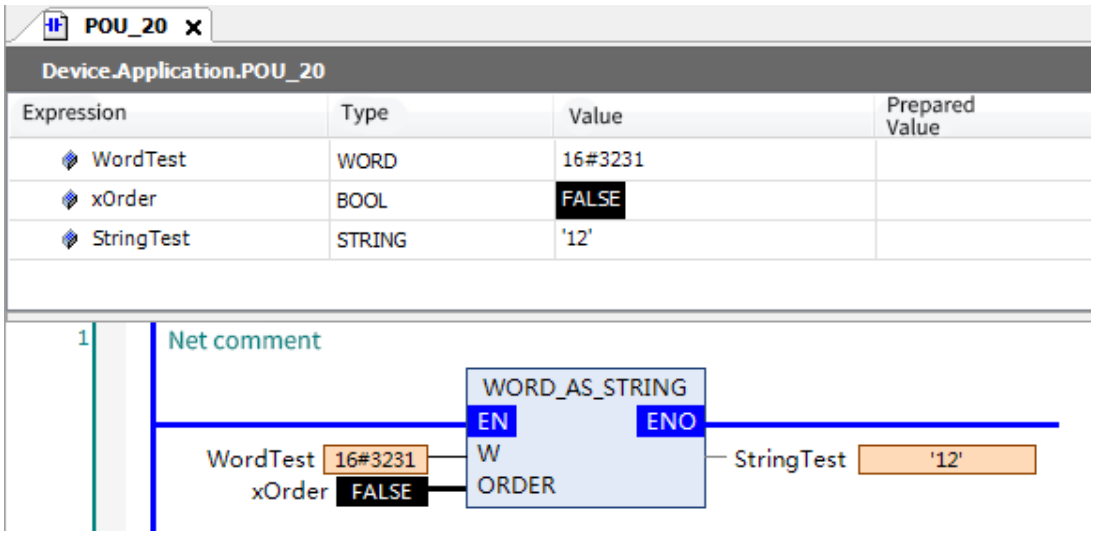
■ Program example

This instruction converts the source data from words to strings.

ST

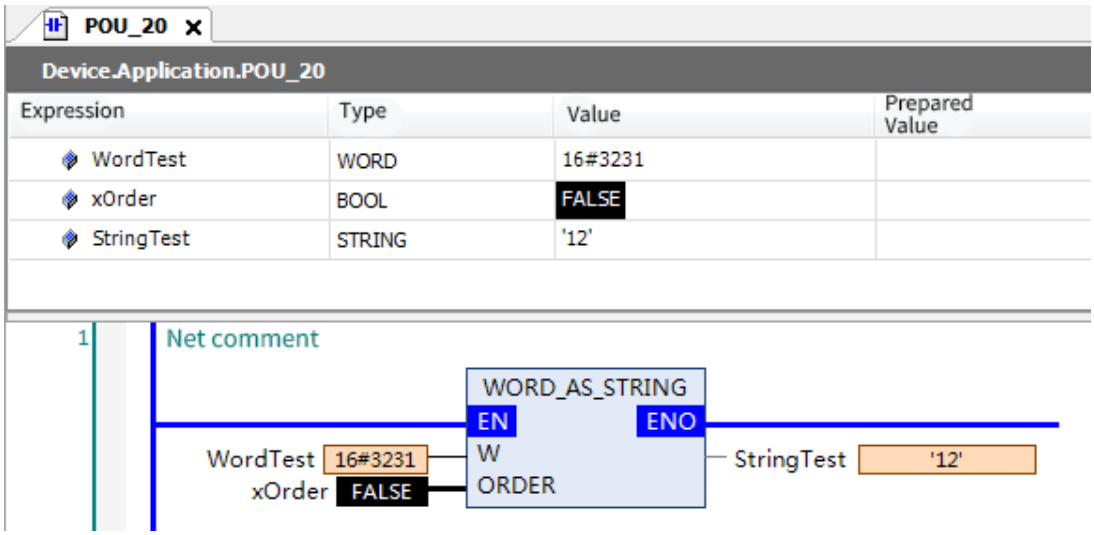


LD

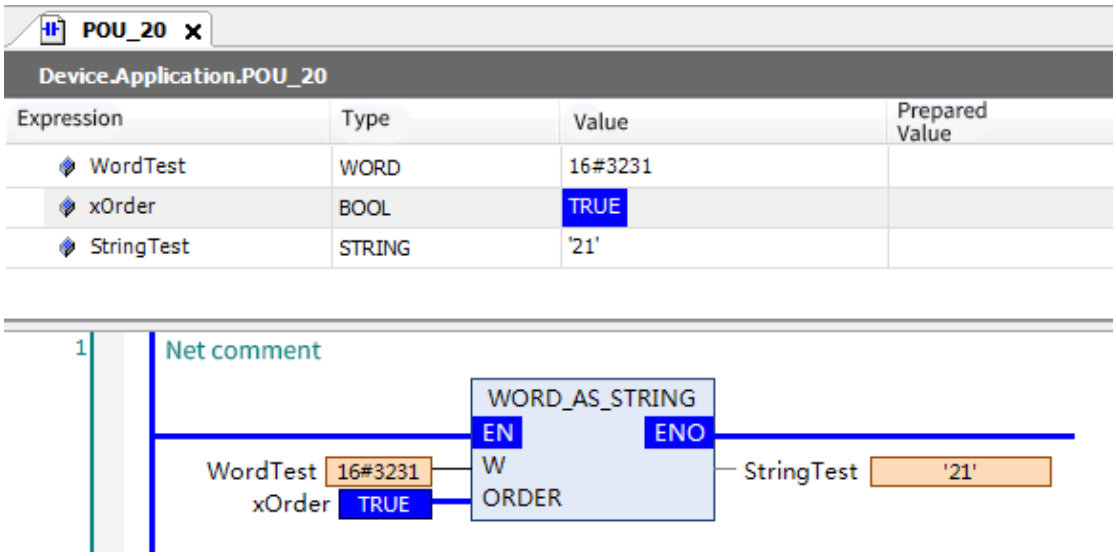


The ORDER pin converts high/low-order bits for variables.

ORDER: = False:



ORDER: = True:



■ Precautions

· When the ORDER pin is not used to switch high/low-order bits, this instruction only converts ASCII code of the Word type to text strings, without changing the values. When the ORDER pin is TRUE, the high/low-order bits are switched.

3.8. 26 Byte_To_HexString

■ Instruction description

This instruction converts bytes to hexadecimal text strings.

Instruction	Name	FB/FC	LD Expression	ST Expression
Byte_To_HexString	Byte-to-hexadecimal text string conversion	FC	Byte_To_HexString EN ENO B	Byte_To_HexString(B:=);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
B	Byte to convert	BYTE	0 to 255	0	Input data address

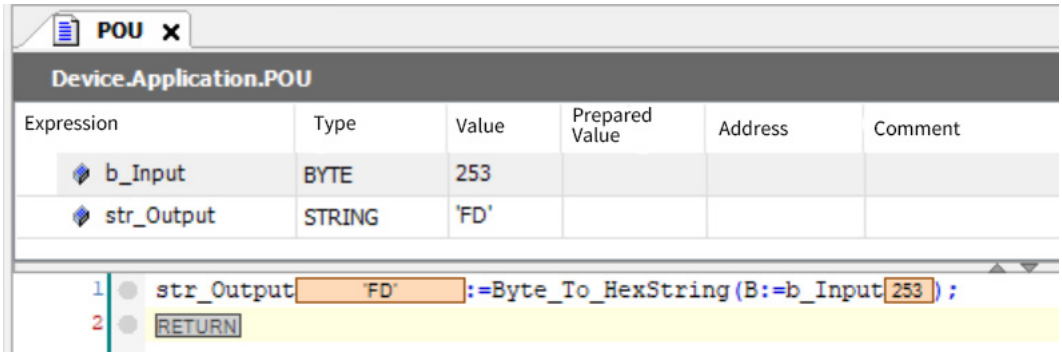
Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
OUT	Return value	STRING	-	' 00'	Hexadecimal text string

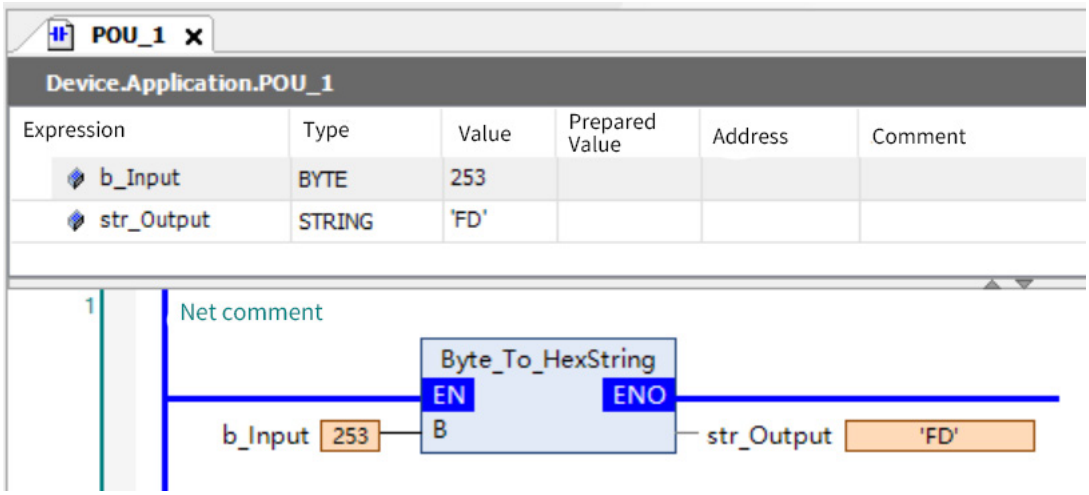
	Bool-ean	Bit String				Integer								Real Num-ber		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
W	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
ORDER	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√

■ Program example

ST



LD



■ Precautions

· The value range of B is 0 to 255. If the value is not within the range, an error occurs.

· The output range is '00' to 'FF', that is, 00000000 to 11111111.

3.8. 27 Word_To_HexString

■ Instruction description

This instruction converts words to hexadecimal text strings.

Instruction	Name	FB/FC	LD Expression	ST Expression
Word_To_HexString	Word-to-hexadecimal text string conversion	FC		Word_To_HexString(w:=);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
w	Word to convert	WORD	0-FFFF	0	Input data address

Output variables

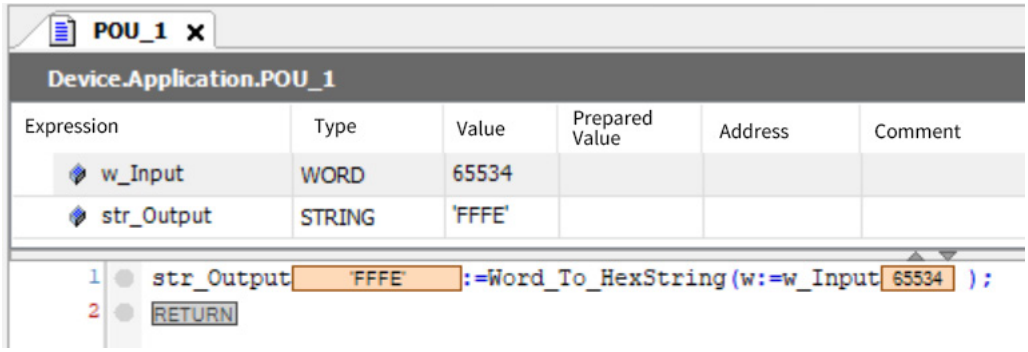
Output Variable	Name	Data Type	Value Range	Initial Value	Description

OUT	Return value	STRING	-	0000	Hexadecimal text string

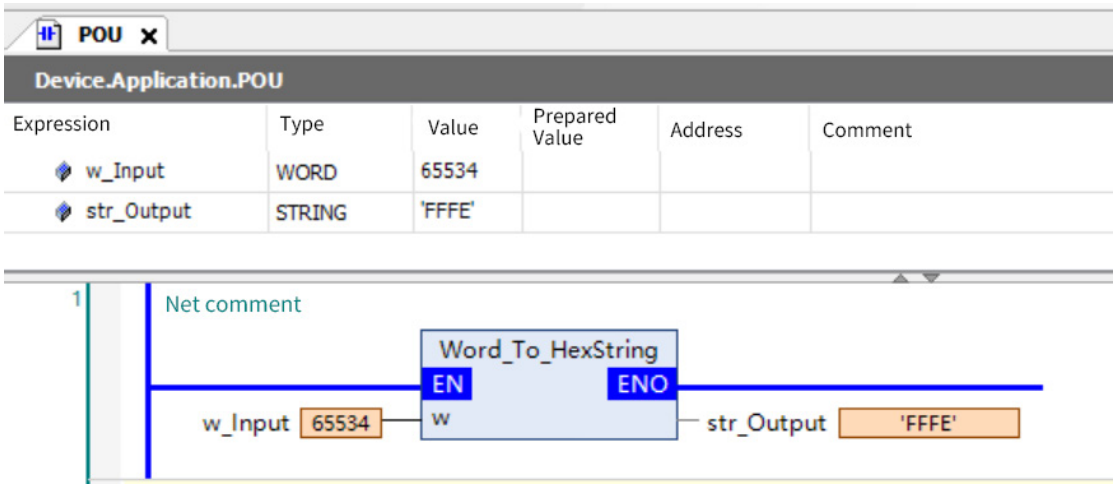
	Bool-ean	Bit String				Integer								Real Num-ber		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
W	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
OUT	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√

■ Program example

ST



LD



■ Precautions

· The value range of B is 0 to 65535. If the value is not within the range, an error occurs.

· The output range is '0000' to 'FFFF', that is, 0000 0000 0000 0000 to 1111 1111 1111 1111.

3.8. 28 Dword_To_HexString

■ Instruction description

This instruction converts doublewords (DWORD) to hexadecimal text strings.

Instruction	Name	FB/FC	LD Expression	ST Expression
Dword_To_HexString	Doubleword-to-hexadecimal text string conversion	FC	DWord_To_HexString EN ENO dw	DWord_To_HexString(dw:=);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
dw	Doubleword to convert	DWORD	0 to FFFF FFFF	0	Input data address

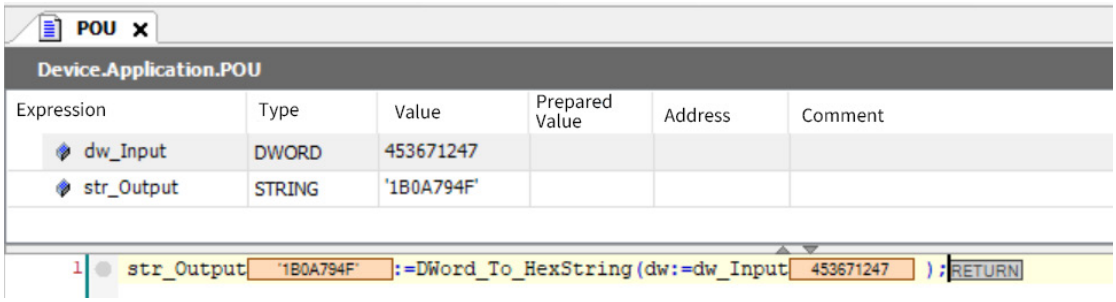
Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
OUT	Return value	STRING	-	'0000 0000'	Hexadecimal text string

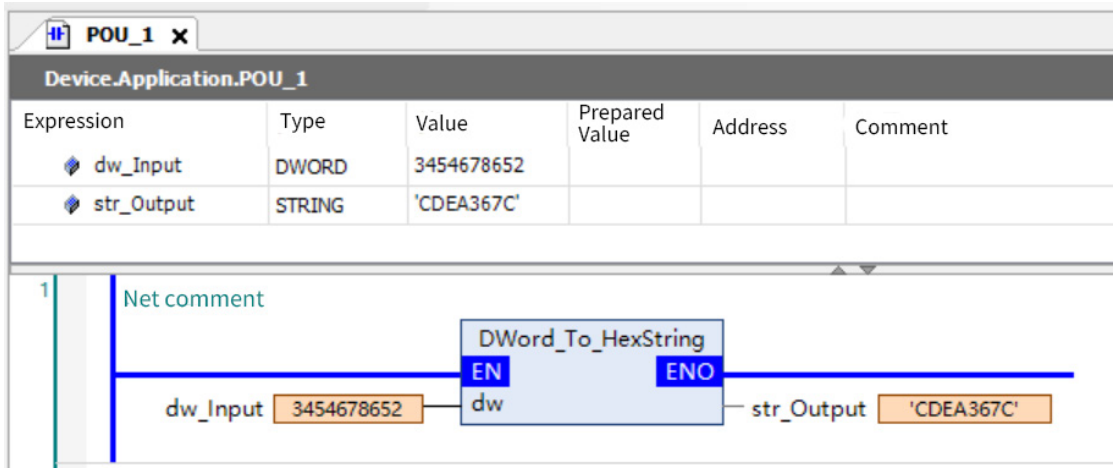
	Bool-ean	Bit String				Integer								Real Num-ber		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
dw	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
OUT	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√

■ Program example

ST



LD



■ Precautions

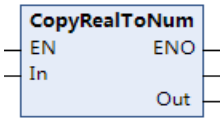
· The value range of dw is 0 to 4294967295. If the value is not within the range, an error occurs.

· The output range is '0000 0000' to 'FFFF_FFFF', that is, 0000 0000 0000 0000 0000 0000 0000 0000 to 1111 1111 1111 1111 1111 1111 1111 1111.

3.8. 29 CopyRealToNum

This instruction converts a text string to a Byte array.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
CopyRealToNum	Bit pattern copy (real number to signed integer)	FC		CopyRealToNum(In :=, Out =>);
CopyLrealToNum	Bit pattern copy (long real number to signed integer)	FC		CopyLrealToNum(In :=, Out =>);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
In	Copy source	-	Depends on data types	0	Copy source

Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
Out	Paste target	-	Depends on data types	(*)	Paste target

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In	-	-	-	-	-	-	-	-	-	-	-	-	-	√	√	-	-	-	-	-
Out	If the data type of In is REAL, the output is DINT. If the data type is LREAL, the output is LINT.																			

■ Function

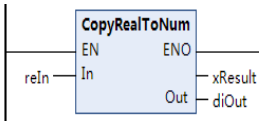
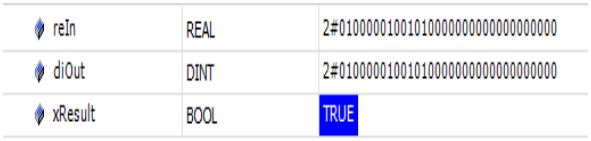
The content of the copy source is directly copied to the target Out, without processing.

The instruction name is determined by two combinations of data types of In and Out.

In	Out	Directive Name
REAL	DINT	CopyRealToNum

LREAL	LINT	CopyLrealToNum

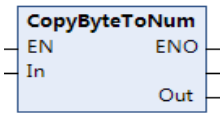
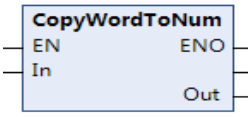
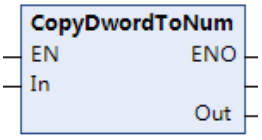
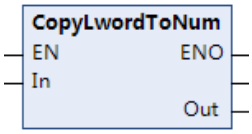
■ Program example

	LD	ST
Defined variable	VAR reIn : REAL := 10.5; diOut : DINT; xResult : BOOL; END_VAR	
Program		xResult := CopyRealToNum(In:= reIn, Out=> diOut);
Running result		

3.8. 30 CopyByteToNum

This instruction directly copies the content of a bit string to a signed integer.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
CopyByteToNum	Bit pattern copy (bit string to signed integer)	FC		CopyByteToNum(In :=, Out =>);
CopyWordToNum	Bit pattern copy (word to signed integer)	FC		CopyWordToNum(In :=, Out =>);
CopyDwordToNum	Bit pattern copy (bit string to signed integer)	FC		CopyDwordToNum(In :=, Out =>);
CopyDwordToNum	Bit pattern copy (bit string to signed integer)	FC		CopyLwordToNum(In :=, Out =>);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
In	Copy source	-	Depends on data types	0	Copy source

Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
Out	Paste target	-	Depends on data types	(*)	Paste target

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In	-	√	√	√	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Out	Data type of a signed integer with the same value as In																			

■ Function

The content of the copy source is directly copied to the target Out, without processing.

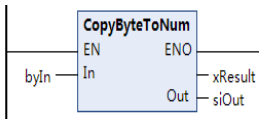
The instruction name is determined by four combinations of data types of In and Out.

In	Out	Directive Name
BYTE	SINT	CopyByteToNum
WORD	INT	CopyWordToNum
DWORD	DINT	CopyDwordToNum
LWORD	LINT	CopyLwordToNum

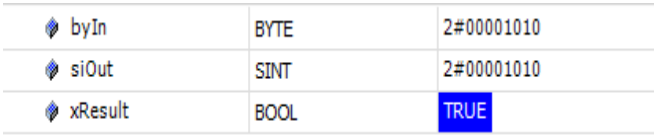
■ Program example

	LD	ST
Defined variable		
Program		

```
VAR
    byIn    : BYTE := 10;
    siOut   : SINT;
    xResult : BOOL;
END_VAR
```



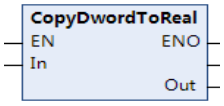

```
xResult := CopyByteToNum(
        In:= byIn,
        Out=> siOut);
```

Running result	

3.8. 31 CopyDwordToReal

This instruction directly copies the content of a bit string to a real number.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
CopyDwordToReal	Bit pattern copy (bit string to real number)	FC		CopyDwordToReal(In :=, Out =>);
CopylwordToLreal	Bit pattern copy (long word to long real number)	FC		CopyLwordToLreal(In :=, Out =>);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
In	Copy source	-	Depends on data types	0	Copy source

Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
Out	Paste target	-	Depends on data types	(*)	Paste target

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In	-	-	-	-	√	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Out	If the data type of In is DWORD, the output is REAL. If the data type is LWORD, the output is LREAL.																			

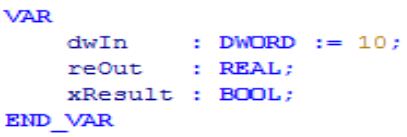
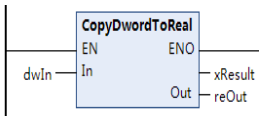
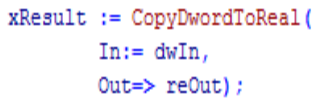
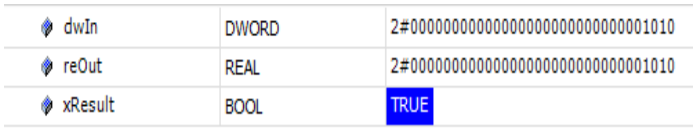
■ Function

The content of the copy source is directly copied to the target Out, without processing.

The instruction name is determined by four combinations of data types of In and Out.

In	Out	Directive Name
DWORD	REAL	CopyDwordToReal
LWORD	LREAL	CopyLwordToLreal

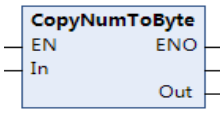
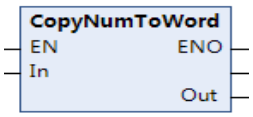
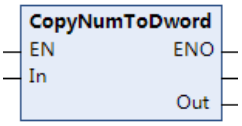
■ Program example

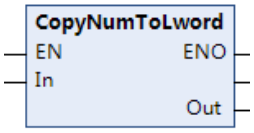
	LD	ST
Defined variable		
Program		
Running result		

3.8. 32 CopyNumToByte

This instruction directly copies the content of a bit string to bytes.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
CopyNumToByte	Bit pattern copy (signed integer to bit string)	FC		CopyNumToByte (In :=, Out =>);
CopyNumToWord	Bit pattern copy (signed integer to word)	FC		CopyNumToWord (In :=, Out =>);
CopyNumToDword	Bit pattern copy (signed integer to bit string)	FC		CopyNumToDword(In :=, Out =>);

Instruction	Name	FB/FC	LD Expression	ST Expression
CopyNumToLword	Bit pattern copy (signed integer to long word)	FC		CopyNumToLword(In :=, Out =>);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
In	Copy source	-	Depends on data types	0	Copy source

Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
Out	Paste target	-	Depends on data types	(*)	Paste target

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In	-	-	-	-	-	-	-	-	-	√	√	√	√	-	-	-	-	-	-	-
Out	Data type of a bit string with the same value as In																			

■ Function

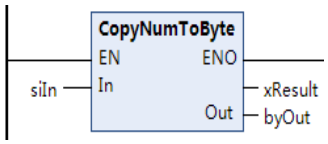
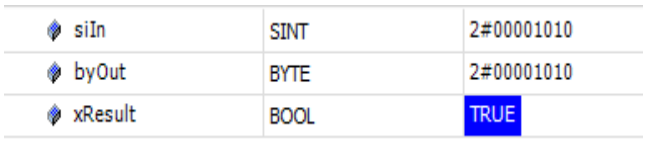
The content of the copy source is directly copied to the target Out, without processing.

The instruction name is determined by four combinations of data types of In and Out.

In	Out	Directive Name
SINT	BYTE	CopyNumToByte
INT	WORD	CopyNumToWord
DINT	DWORD	CopyNumToDword
LINT	LWORD	CopyNumToLword

■ Program example

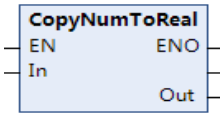
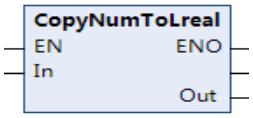
	LD	ST

Defined variable	VAR siIn : SINT := 10; byOut : BYTE; xResult : BOOL; END_VAR
Program	 xResult := CopyNumToByte(In:= siIn, Out=> byOut);
Running result	

3.8. 33 CopyNumToReal

This instruction directly copies the content of a signed integer to a real number.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
CopyNumToReal	Bit pattern copy (signed integer to real number)	FC		CopyNumToReal (In :=, Out =>);
CopyNumToLreal	Bit pattern copy (signed integer to long real number)	FC		CopyNumToLreal (In :=, Out =>);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
In	Copy source	-	Depends on data types	0	Copy source

Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
Out	Paste target	-	Depends on data types	(*)	Paste target

	Bool-ean	Bit String				Integer								Real Num-ber		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In	-	-	-	-	-	-	-	-	-	-	-	√	√	-	-	-	-	-	-	-
Out	If the data type of In is DINT, the output is REAL. If the data type is LINT, the output is LREAL.																			

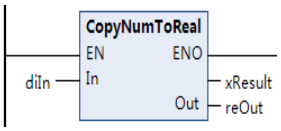
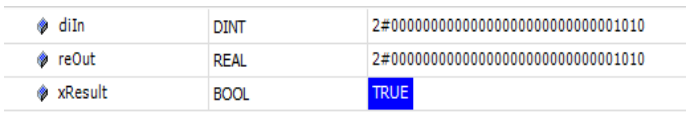
■ Function

The content of the copy source is directly copied to the target Out, without processing.

The instruction name is determined by two combinations of data types of In and Out.

In	Out	Directive Name
DINT	REAL	CopyNumToReal
LINT	LREAL	CopyNumToLreal

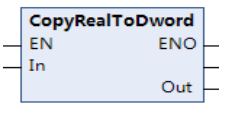
■ Program example

	LD	ST
Defined variable	VAR diIn : DINT := 10; reOut : REAL; xResult : BOOL; END_VAR	
Program		xResult := CopyNumToReal(In:= diIn, Out=> reOut);
Running result		

3.8. 34 CopyRealToDword

This instruction directly copies the content of a real number to a bit string.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
CopyRealToDword	Bit pattern copy (real number to bit string)	FC		CopyRealToDword (In :=, Out =>);

Instruction	Name	FB/FC	LD Expression	ST Expression
CopyLrealToLword	Bit pattern copy (long real number to long word)	FC		CopyLrealToLword (In :=, Out =>);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
In	Copy source	-	Depends on data types	0	Copy source

Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
Out	Paste target	-	Depends on data types	(*)	Paste target

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In	-	-	-	-	-	-	-	-	-	-	-	-	-	√	√	-	-	-	-	-
Out	If the data type of In is REAL, the output is DWORD. If the data type is LREAL, the output is LWORD.																			

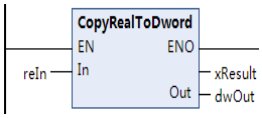
■ Function

The content of the copy source is directly copied to the target Out, without processing.

The instruction name is determined by two combinations of data types of In and Out.

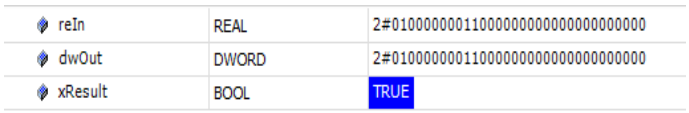
In	Out	Directive Name
REAL	DWORD	CopyRealToDword
LREAL	LWORD	CopyLrealToLword

■ Program example

	LD	ST
Defined variable		
Program		

```
VAR
    reIn    : REAL := 3.5;
    dwOut   : DWORD;
    xResult : BOOL;
END_VAR
```

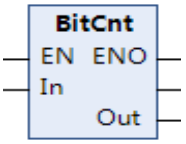
```
xResult := CopyRealToDword(
        In:= reIn,
        Out=> dwOut);
```

Running result	

3.8. 35 BitCnt

This instruction counts TRUE bits in a bit string.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
BitCnt	Bit counter	FC		BitCnt (In :=, Out =>);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
In	Count string	-	Depends on data types	0	Count string

Output variables

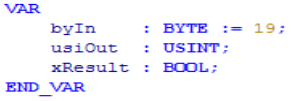
Output Variable	Name	Data Type	Value Range	Initial Value	Description
Out	Count result	USINT	Depends on data types	0	Number of TRUE bits in a count string

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In	-	√	√	√	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Out	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-

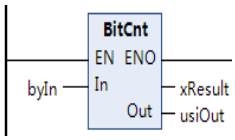
■ Function

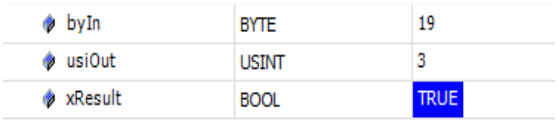
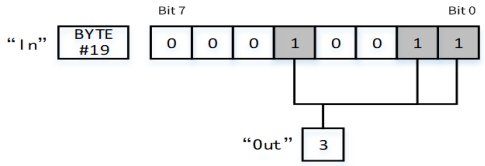
This instruction counts TRUE bits in count string In.

■ Program example

	LD	ST
Defined variable		

	LD	ST

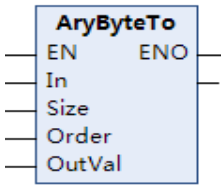
Program		`xResult := BitCnt(`
		`In:= byIn,`
		`Out=> usiOut);`

Running result		
Work principle		

3.8. 36 AryByteTo

This instruction joins byte array elements and saves the result in a variable.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
AryByteTo	Conversion from byte array	FC		AryByteTo (In :=, Size :=, Order :=, OutVal :=,);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
In[] (array)	Array to convert	-	Depends on data types	-	Array to convert
Size	Number of elements to convert	UINT	Depends on data types	-	Number of elements in In[] to convert
Order	Conversion order	-	Depends on data types	-	Conversion order
OutVal	Conversion result	-	Depends on data types	-	Conversion result

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In[] (array)	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Size	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-
Order	See "Function" for the enumerators for the enumerated type _eBYTE_ORDER.																			
OutVal	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√
	An enumeration, an array, an array element, a structure, or a structure member can also be specified.																			

■ Function

This instruction takes the first Size elements in array to convert In[], joins them to match the size of the data type of conversion result OutVal, and then saves the result to OutVal.

Order specifies the order to join the elements of In[]. The data type of Order is enumerated type _eBYTE_ORDER. The meanings of the enumerators are as follows.

Enumerator	Meaning
_LOW_HIGH	Lower byte first, higher byte last
_HIGH_LOW	Higher byte first, lower byte last

The following describes the functions of AryByteTo in two situations.

(1) When the data type of OutVal is one byte

If the data type of OutVal is one byte, In[] is stored in OutVal by one byte.

The following data types have one byte.

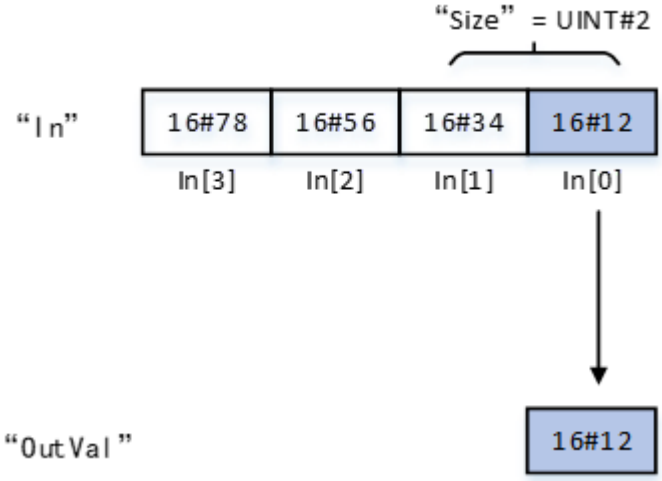
Category	Data Type
Bit string	BYTE
Integer	USINT and SINT
Real number	None
Time, duration, date, and text string	None
Others	An enumerator of one byte An array for which the total for all elements is one byte An array element of one byte A structure for which the total for all members is one byte A member of one byte

The following storage method is used.

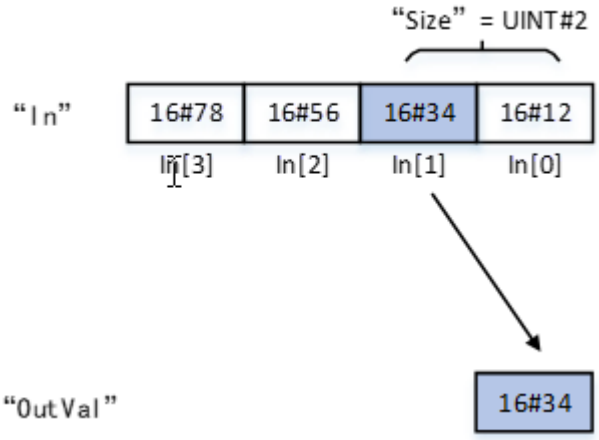
Value of Order	Data Type of OutVal	Storage Method in OutVal

LOW_HIGH	BOOL	Bit 0 of In[0] is stored in OutVal.
	Not BOOL	The value of In[0] is stored in OutVal.
HIGH_LOW	BOOL	Bit 1 of In[0] is stored in OutVal.
	Not BOOL	The value of In[1] is stored in OutVal.

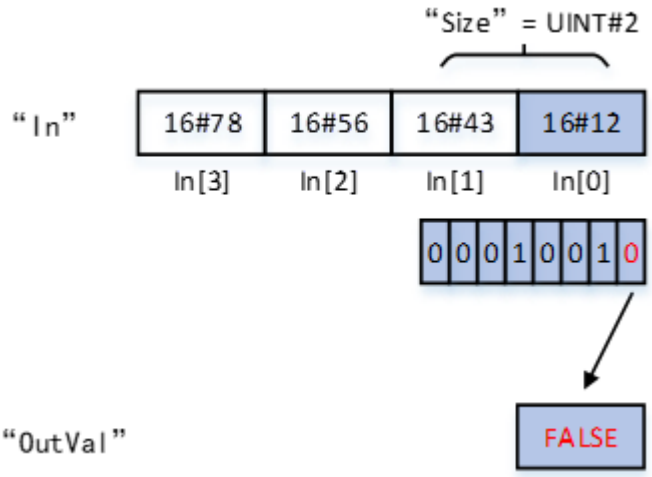
The following example is for when OutVal is a SINT variable, Size is UINT#2, and Order is _LOW_HIGH.



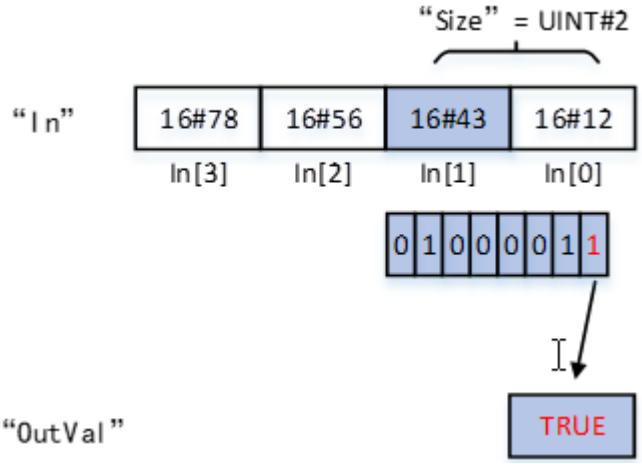
The following example is for when OutVal is a SINT variable, Size is UINT#2, and Order is _HIGH_LOW.



The following example is for when OutVal is a BOOL variable, Size is UINT#2, and Order is _LOW_HIGH.



The following example is for when OutVal is a BOOL variable, Size is UINT#2, and Order is _HIGH_LOW.



(2) When the data type of OutVal is two bytes or more

If the data type of OutVal is two bytes or more, elements from In[] are joined so that the result is just large enough for the size of the data type of OutVal. The result is stored in OutVal.

The following data types have two bytes or more.

Category	Data Type
Bit string	WORD, DWORD, and LWORD
Integer	UINT, UDINT, ULINT, INT, DINT, and LINT
Real number	REAL and LREAL
Time, duration, date, and text string	TIME, DATE, TOD, DT, and STRING types of two bytes or more
Others	An enumerator of two bytes or more An array for which the total for all elements is two bytes or more An array element of two bytes or more A structure for which the total for all members is two bytes or more A member of two bytes or more

The following storage method is used.

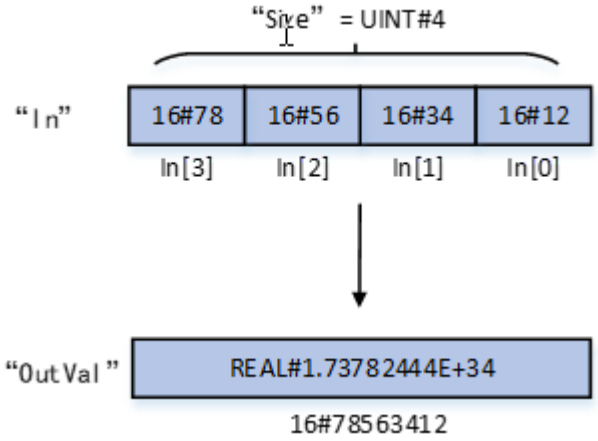
In[0] and In[1] are joined according to the value of Order to create one word (two bytes) of data. If Order is _LOW_HIGH, the higher byte is stored in In[1] and the lower byte is stored in In[0]. If Order is _HIGH_LOW, the higher byte is stored in In[0] and the lower byte is stored in In[1].

In the same way elements that start from In[2] and In[3] are joined to make more words of data.

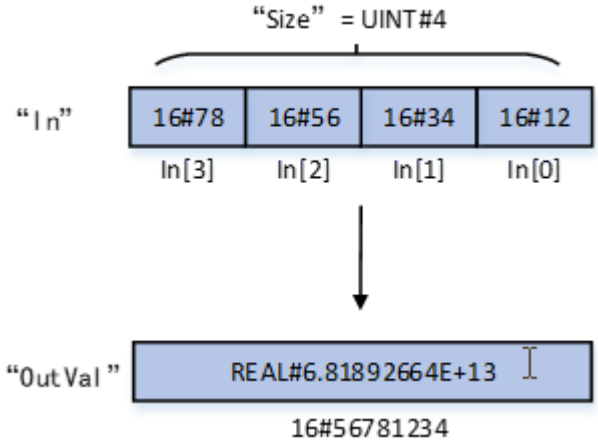
The words of data are joined to match the size of the data type of OutVal. For example, if OutVal is DWORD data, four individual words of data are joined. If OutVal is an array, In[i] (i as an even number) is stored in OutVal[i+1] and In[i] (i as an odd number) is stored in OutVal[i-1]. If the value of Size is an odd number, data is stored till OutVal[Size], and 16#00 is stored in OutVal[Size-1].

The obtained data is stored in OutVal.

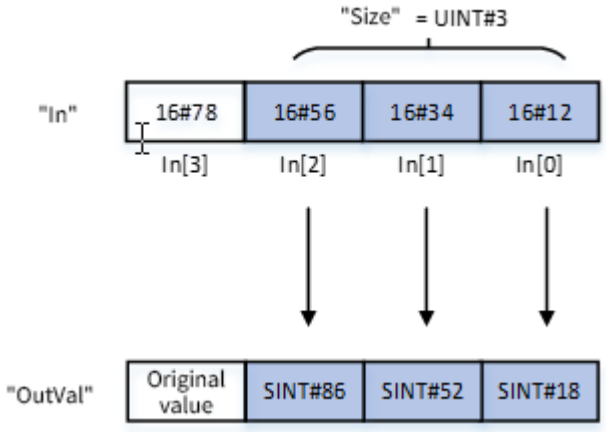
The following example is for when OutVal is a REAL variable, Size is UINT#4, and Order is _LOW_HIGH.



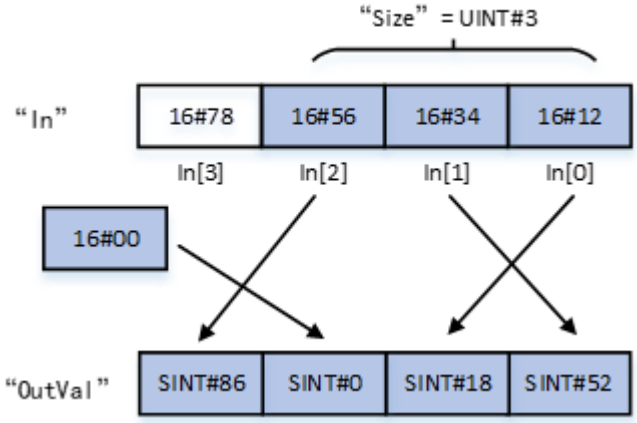
The following example is for when OutVal is a REAL variable, Size is UINT#4, and Order is _HIGH_LOW.



The following example is for when OutVal is a SINT array with four elements, Size is UINT#3, and Order is _LOW_HIGH.



The following example is for when OutVal is a SINT array with four elements, Size is UINT#3, and Order is _HIGH_LOW.

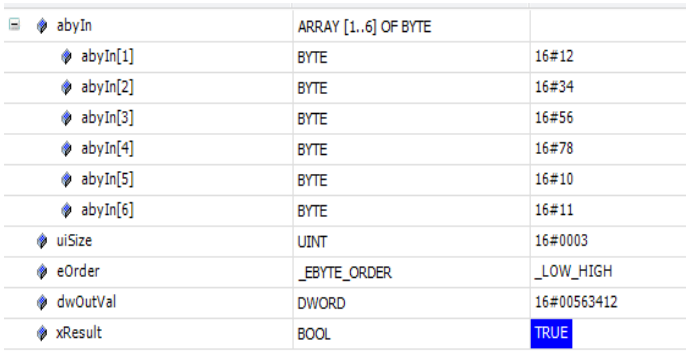


■ Program example

	LD	ST

Defined variable	VAR
	abyIn : ARRAY [1..6] OF BYTE := [16#12,16#34,16#56,16#78,16#10,16#11];
	uiSize : UINT := 3;
	eOrder : _eBYTE_ORDER := _eBYTE_ORDER._LOW_HIGH;
	dwOutVal : DWORD;
	xResult : BOOL;
	END_VAR	

Program	AryByteTo EN ENO abyIn[1] — In uiSize — Size eOrder — Order dwOutVal — OutVal xResult	xResult := AryByteTo(In:= abyIn[1], Size:= uiSize, Order:= eOrder, OutVal:= dwOutVal);

Running result	

■ Precautions

When the value of Size exceeds the In[] array, the return value is FALSE and OutVal does not change.

OutVal cannot be a BOOL array. Otherwise, the program run fails.

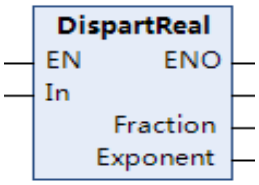
When OutVal is a structure, some of the values of In[] may be inserted in adjustment areas between members depending on the composition.

· When the value of Size is 0, the value of Out is TRUE and OutVal does not change.

3.8. 37 DispartReal

This instruction separates a real number into the signed mantissa and the exponent.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
DispartReal	Separate mantissa and exponent	FC		DispartReal (In :=, Fraction =>, Exponent =>);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
In	Real number	-	Depends on data types	-	Real number to separate

Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
Fraction	Signed mantissa	LINT	Depends on data types	0	Signed mantissa
Exponent	Exponent	INT	Depends on data types	0	Exponent

	Bool-ean	Bit String				Integer								Real num-ber		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In	-	-	-	-	-	-	-	-	-	-	-	-	-	√	√	-	-	-	-	-
Fraction	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-
Exponent	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-

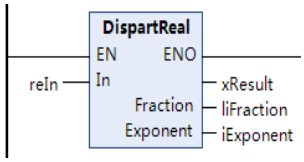
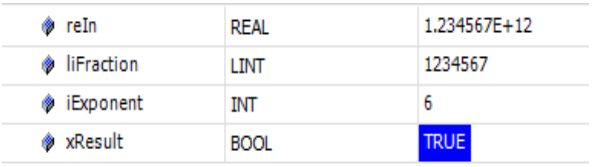
■ Function

This instruction separates a real number into the signed mantissa Fraction and the exponent Exponent.

If In is REAL data, Fraction is a 7-digit integer. If In is LREAL data, Fraction is a 15-digit integer. The following table lists the valid ranges of Fraction according to the data types of In.

Data Type of In	Valid Range of Fraction
REAL	–9999999 to +9999999
LREAL	–999999999999999 to +999999999999999

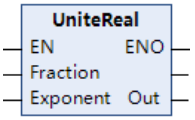
■ Program example

	LD	ST
Defined variable	VAR reIn : REAL := 123.4567E+10; liFraction : LINT; iExponent : INT; xResult : BOOL; END_VAR	
Program		xResult := DispartReal(In:= reIn, Fraction=> liFraction, Exponent=> iExponent);
Running result		

3.8. 38 UniteReal

This instruction combines a signed mantissa and an exponent to make a real number.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
UniteReal	Combine real number mantissa and exponent	FC		UniteReal (Fraction :=, Exponent :=, Out =>,);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
Fraction	Signed mantissa	-	Depends on data types	-	Signed mantissa
Exponent	Exponent	INT	Depends on data types	0	Exponent

Output variables

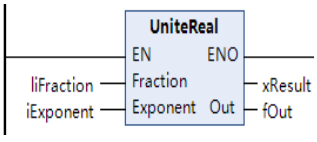
Output Variable	Name	Data Type	Value Range	Initial Value	Description
Out	Real number	LREAL	Depends on data types	0	Real number

	Boolean	Bit String				Integer								Real number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Fraction	-	-	-	-	-	-	-	-	-	-	-	√	√	-	-	-	-	-	-	-
Exponent	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-
Out	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-

■ Function

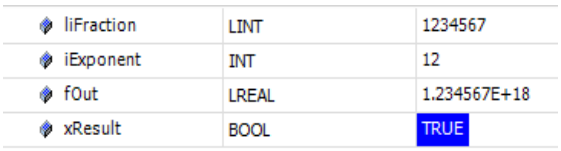
This instruction combines the signed mantissa Fraction and the exponent Exponent to make the real number Out.

■ Program example

	LD	ST
Defined variable		
Program		

```
VAR
    liFraction   : LINT := 1234567;
    iExponent    : INT := 12;
    fOut         : LREAL;
    xResult      : BOOL;
END_VAR
```

```
xResult := UniteReal(Fraction:= liFraction,
    Exponent:= iExponent,
    Out=> fOut);
```

Running result	

■ Precautions

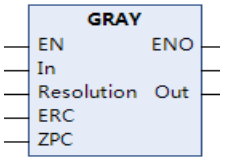
Depending on the values of Fraction and Exponent, an error may occur in the conversion from an integer to a real number.

If the combined result exceeds the valid range of Out and Exponent is positive, the value of Out is infinity with the same sign as Fraction. If Exponent is negative, the value of Out is 0.

3.8. 39 GRAY

This instruction converts a Gray code into an angle.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
Gray	Gray code conversion	FC		Gray (In :=, Resolution :=, ERC :=, ZPC :=, Out =>,);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
In	Data to convert	WORD	Depends on data types	0	Gray code to convert
Resolution	Resolution	-	Depends on data types	_R256	Resolution
ERC	Encoder remainder correction	UINT	Depends on data types	0	Encoder remainder correction
ZPC	Zero point correction	UINT	Depends on data types	0	Zero point correction

Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
Out	Conversion result	LREAL	Depends on data types	-	Conversion result

	Boolean	Bit String				Integer								Real number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Resolution	See "Function" for the enumerators of the enumerated type _eGRY_RESOLUTION.																			
ERC	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-
ZPC	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-
Out	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-

■ Function

This instruction converts the Gray code in In (the output value from a rotary encoder) to an angle. The conversion result Out is in degrees.

The data type of Resolution is the enumerated type _eGRY_RESOLUTION. The meanings of the enumerators are as follows.

Enumerator	Meaning
_R256	256
_R1B	1-bit (2)
_R2B	2-bit (4)
_R3B	3-bit (8)
_R4B	4-bit (16)
_R5B	5-bit (32)
_R6B	6-bit (64)
_R7B	7-bit (128)
_R8B	8-bit (256)
_R9B	9-bit (512)
_R10B	10-bit (1024)
_R11B	11-bit (2048)
_R12B	12-bit (4096)
_R13B	13-bit (8192)
_R14B	14-bit (16384)
_R15B	15-bit (32768)
_R360	360
_R720	720
_R1024	1024

The Gray code is a reflected binary code. Two successive values, such as 0 and 1 and 1 and 2, differ in only one bit. Gray codes are used for the output from absolute encoders.

The following table lists the 4-bit binary code and Gray code.

Decimal Number	Binary Code				Gray Code			
	2^3	2^2	2^1	2^0	d	c	b	a
0	0	0	0	0	0	0	0	0

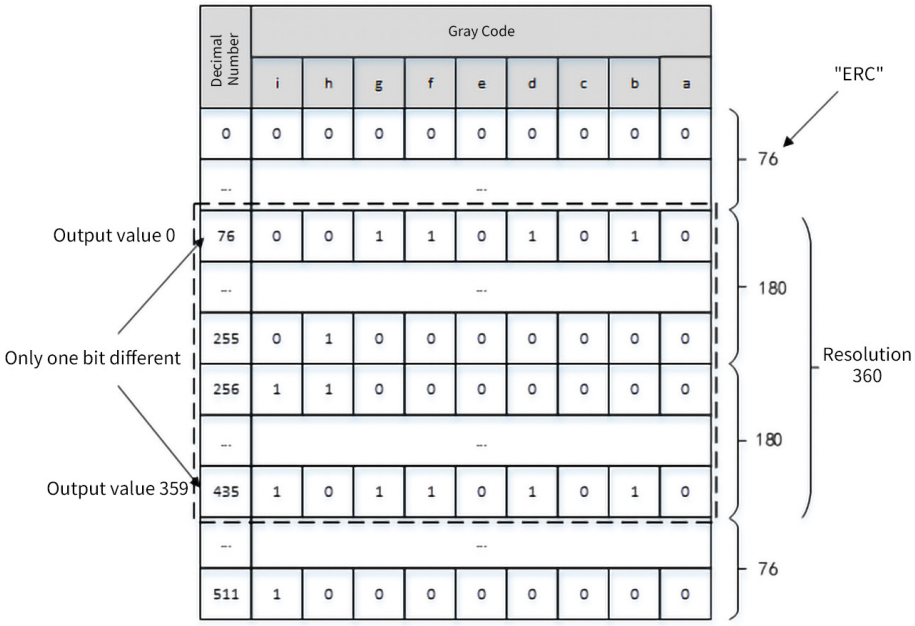
Decimal Number	Binary Code				Gray Code			
	2^3	2^2	2^1	2^0	d	c	b	a
1	0	0	0	1	0	0	0	1
2	0	0	1	0	0	0	1	1
3	0	0	1	1	0	0	1	0
4	0	1	0	0	0	1	1	0
5	0	1	0	1	0	1	1	1
6	0	1	1	0	0	1	0	1
7	0	1	1	1	0	1	0	0
8	1	0	0	0	1	1	0	0
9	1	0	0	1	1	1	0	1
10	1	0	1	0	1	1	1	1
11	1	0	1	1	1	1	1	0
12	1	1	0	0	1	0	1	0
13	1	1	0	1	1	0	1	1
14	1	1	1	0	1	0	0	1
15	1	1	1	1	1	0	0	0

Using the Gray code enables prevention of incorrect output values because only one bit in the Gray code changes when the output value of the encoder is incremented or decremented by 1.

The ERC variable is used to specify the Gray code range when the encoder resolution is not a power of 2. The range is specified so that only one bit is different between the maximum and minimum encoder output values.

For example, consider the use of an absolute encoder with a resolution of 360. Nine bits are used for the Gray code. The range that can be expressed with nine bits is 0 to 511. In this case, a range of 180 from the center of 0 to 511 is used for the Gray code, that is, 76 to 435. Therefore, a Gray code of 001101010 (76 decimal) is output for an output value of 0, and a Gray code of 101101010 (435 decimal) is output for an output value of 359. Only one bit is different between these values.

In this case, the value of the encoder remainder correction ERC is 76.



The ZPC variable is set to offset the zero position of the rotary encoder. For example, to offset the zero position from 90° for a rotary encoder with a resolution of 256, the value of ZPC is calculated as follows: 256 x (90/360) = 64.

The following example is for when In is WORD#16#123, Resolution is _R9B, ERC is UINT#0, and ZPC is UINT#88.

First, the resolution is 9 bits, so one increment in the Gray code is calculated as follows: 360° /29 = 0.703125° .

A decimal value of 450 corresponds to a Gray code of 16#123. Therefore, the angle before compensation is calculated as follows: 0.703125° x 450 = 316.40625° .

The value of ERC is 0 and the value of ZPC is 88. Therefore, the angle after compensation is calculated as follows: 316.40625° - (0 + 88) x 0.703125° = 254.53125° .

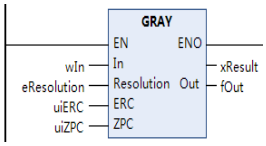
The value of Out is LREAL#254.53125.

■ Program example

<table>
<tr><th></th><th>LD</th><th>ST</th></tr>
<tr><td>Defined variable</td><td colspan="2">

```
VAR
    wIn          : WORD := 16#123;
    eResolution  : _eGRY_RESOLUTION := HC_OmronUtils._eGRY_RESOLUTION._R9B;
    uiERC        : UINT := 0;
    uiZPC        : UINT := 88;
    fOut         : LREAL;
    xResult      : BOOL;
END_VAR
```

</td></tr>
<tr><td>Program</td><td>



</td><td>

```
xResult := GRAY(In:= wIn,
    Resolution:= eResolution,
    ERC:= uiERC,
    ZPC:= uiZPC,
    Out=> fOut);
```

</td></tr>
<tr><td>Running result</td><td colspan="2">

wIn	WORD	291
eResolution	_EGRY_RESOLUTION	_R9B
uiERC	UINT	0
uiZPC	UINT	88
fOut	LREAL	254.53125
xResult	BOOL	TRUE

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

■ Precautions

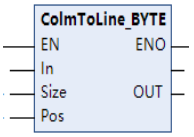
An error occurs in the following cases. The return value changes to FALSE and Out is initialized to 0.

- The value of ERC exceeds the resolution that is specified in Resolution.
- The value of ZPC exceeds the resolution that is specified in Resolution.
- In, when converted to a bit string, is less than the value of ERC.
- The value of the bit string after correction for ERC exceeds the resolution that is specified in Resolution.

3.8. 40 ColmToLine_**

This instruction extracts bit values from the specified position of array elements and outputs them as a bit string.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
ColmToLine_ **	Column-to-line conversion group	FC		ColmToLine_**(In:= , Size:= , Pos:= , OUT=>);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
In[] (array)	Array to convert	-	Depends on data types	-	Array to convert
Size	Number of elements to convert	USINT	Depends on data types	1	Number of elements in In[] to convert
Pos	Conversion bit position	USINT	Depends on data types	0	Bit position to convert

Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
Out	Conversion result	-	Depends on data types	-	Conversion result

	Boolean	Bit String				Integer								Real number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In[] (array)	-	√	√	√	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Size	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Pos	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Out	-	√	√	√	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

■ Function

This instruction extracts bit values from the specified position of array elements and outputs them in order as a bit string.

First, Size elements are extracted from array to convert In[] starting from In[0]. Then, only the values of bits in Pos are extracted. They are placed in order in a bit string of Size bits and stored in the conversion result Out from the least-significant bit. FALSE is stored in the remaining bits of Out.

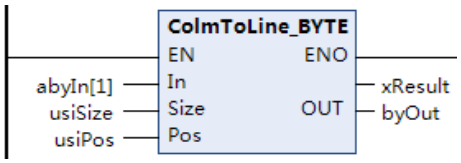
The instruction name is determined by the data types of Out. For example, if Out is of the Byte data type, the instruction is ColmToLine_BYTE.

■ Program example

<table>
<tr><th></th><th>LD</th><th>ST</th></tr>
<tr><td>Defined variable</td><td colspan="2">

```
VAR
    abyIn   : ARRAY [0..4] OF BYTE := [1,23,45,67,89];
    usiSize : USINT := 4;
    usiPos  : USINT := 2;
    byOut   : BYTE;
    xResult : BOOL;
END_VAR
```

</td></tr>
<tr><td>Program</td><td>



</td><td>

```
xResult := ColmToLine_BYTE(In:= abyIn[1],
    Size:= usiSize,
    Pos:= usiPos,
    OUT=> byOut);
```

</td></tr>
<tr><td>Running result</td><td colspan="2">

abyIn	ARRAY [0..4] OF BYTE	
abyIn[0]	BYTE	2#00000001
abyIn[1]	BYTE	2#00010111
abyIn[2]	BYTE	2#00101101
abyIn[3]	BYTE	2#01000011
abyIn[4]	BYTE	2#01011001
usiSize	USINT	2#00000100
usiPos	USINT	2#00000010
byOut	BYTE	2#00000011
xResult	BOOL	TRUE

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

■ Precautions

When the value of Size is 0, all bits in Out change to FALSE.

When the value of Size exceeds the valid range, the return value is FALSE and Out does not change.

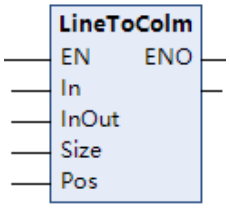
When the value of Pos exceeds the valid range, the return value is FALSE and Out does not change.

When the value of Size exceeds the In[] array, the return value is FALSE and Out does not change.

3.8. 41 LineToColm

This instruction takes the bits from a bit string and outputs them to the specified bit position in array elements.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
LineToColm	Line-to-column conversion	FC		LineToColm(In:= , InOut:= , Size:= , Pos:=);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
In	Data to convert	-	Depends on data types	-	Data to convert
Size	Conversion bit position	UINT	Depends on data types	1	Number of elements in the conversion result
Pos	Number of elements in the conversion result	UINT	Depends on data types	0	Number of elements in the conversion result

In-out variables

In-Out Variable	Name	Data Type	Value Range	Initial Value	Description
InOut[] (array)	Conversion result array	-	Depends on data types	-	Conversion result array

	Boolean	Bit String				Integer								Real number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In	-	√	√	√	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
InOut[] (array)	-	√	√	√	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Size	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-
Pos	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-

■ Function

This instruction takes the bits from a bit string and outputs them to the specified bit position in array elements.

First, Size bits are extracted from the least-significant bit of data to convert In. These bits are treated individually.

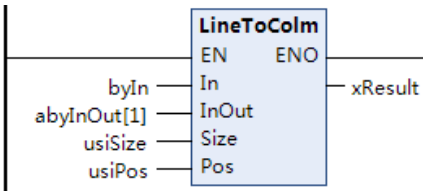
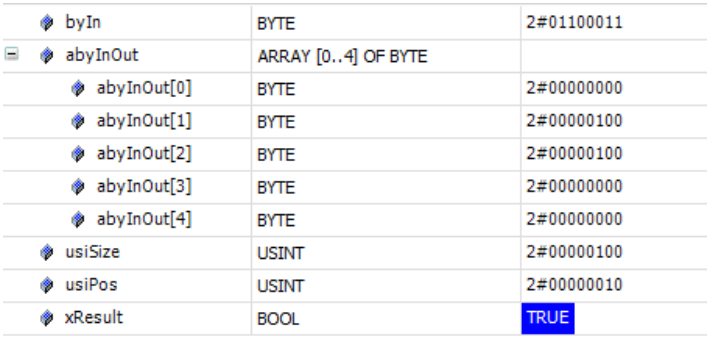
Then, the bits are stored in the conversion result array InOut[] in the Pos bit of the elements starting from InOut[0]. Size specifies the number of array elements to receive bits.

The values of all bits for which values are not stored are retained.

■ Program example

	LD	ST

Defined variable	VAR
	byIn : BYTE := 99;
	abyInOut : ARRAY [0..4] OF BYTE;
	usiSize : USINT := 4;
	usiPos : USINT := 2;
	xResult : BOOL;
	END_VAR	

Program	 `xResult := LineToColm(In:= byIn, InOut:= abyInOut[1], Size:= usiSize, Pos:= usiPos);`
Running result	

■ Precautions

When the value of Size is 0, InOut[] does not change and the return value is TRUE.

When the value of Size exceeds the valid range, InOut[] does not change and the return value is FALSE.

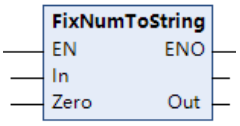
When the value of Pos exceeds the valid range, InOut[] does not change and the return value is FALSE.

When the value of Size exceeds the InOut[] array, InOut[] does not change and the return value is FALSE.

3.8. 42 FixNumToString

This instruction converts a signed fixed-decimal number to a decimal text string.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
FixNumToString	Fixed-decimal number-to-text string conversion	FC		FixNumToString(In:= , Zero:= , Out=>);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
In	Fixed-decimal number	DWORD	Depends on data types	0	Signed fixed-decimal number
Zero	Zero augmentation	BOOL	Depends on data types	TRUE	Augmentation of zeros if there are less than 3 decimal digits TRUE: Add 0 FALSE: Do not add 0

Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
Out	Decimal text string	STRING	Depends on data types	-	Decimal text string

	Boolean	Bit String				Integer								Real number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Zero	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Out	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√

■ Function

This instruction converts the signed fixed-decimal number In to a decimal text string. The conversion steps are as follows:

1. The hexadecimal number In is converted to a decimal number.

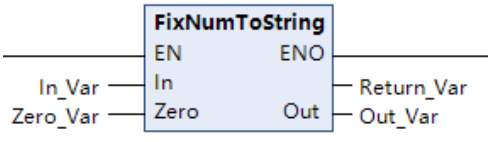
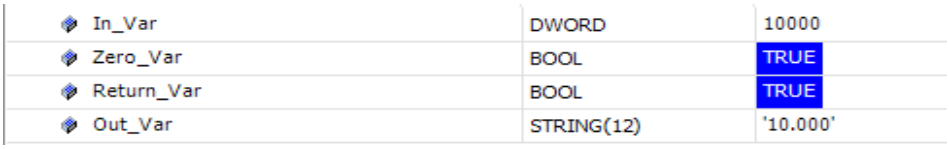
2. The result is divided by 1,000.

Zero specifies whether to add 0 to the third decimal place of Out when there are less than three decimal digits in In.

If the value of Zero is TRUE, 0 is added.

A NULL character is placed at the end of Out.

■ Program example

	LD	ST
Defined variable	In_Var: DWORD; Zero_Var: BOOL; Return_Var: BOOL; Out_Var: STRING(12);	
Program		Return_Var:=FixNumToString(In:=In_Var , Zero:=Zero_Var , Out=>Out_Var);
Running result		

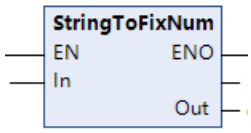
■ Precautions

When the conversion result exceeds the valid range of Out, the return value is FALSE and Out does not change.

3.8. 43 StringToFixNum

This instruction converts a decimal text string to a signed fixed-decimal number.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
StringToFix Num	Text string-to-fixed-decimal number conversion	FC		StringToFixNum(In:= , Out=>);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
In	Decimal text string	STRING	Depends on data types	-	Decimal text string

Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
Out	Fixed-decimal number	DWORD	Depends on data types	-	Fixed-decimal number

	Boolean	Bit String				Integer								Real number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√
Out	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

■ Function

This instruction converts the decimal text string In to a fixed-decimal number. The conversion steps are as follows:

1. The number in In is multiplied by 1,000.

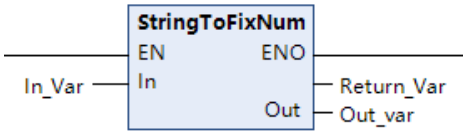
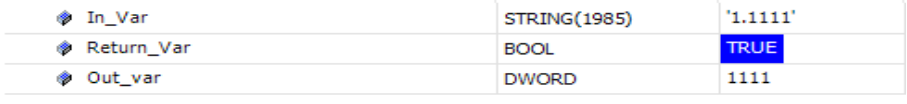
2. The fractional part is truncated.

3. The result is given as a 32-bit hexadecimal number (DWORD).

Name	Format
Sign	· Any consecutive blank characters (16#20) at the beginning of the text string are ignored. Any single plus or minus sign that follows is treated as the sign. · The sign can be omitted. · Any consecutive blank characters after the sign are ignored.

Integer part	· Consecutive numbers (0 to 9) after the sign and up to the decimal point are taken as the integer part. The sign may be omitted sometimes. There may be consecutive blank characters between the sign and the integer part. · If the decimal point and fractional part are omitted, the characters up to the exponent are taken as the integer part. · If the decimal point, fractional part, and exponent are omitted, the characters up to the end of the text string are taken as the integer part. · The integer part cannot be omitted. · The maximum number of digits in the integer part is the maximum text string length of 1986 minus the total number of bytes in the following: the sign, decimal point, fractional part, exponent, and blank characters before and after the sign.
Decimal point	· A single dot (.) following the integer part is taken as the decimal point. · Omit the decimal point if there is no fractional part.
Fractional part	· Consecutive numbers (0 to 9) after the decimal point and up to the exponent are taken as the fractional part. · If the exponent is omitted, the characters up to the end of the text string are taken as the fractional part. · The fractional part can be omitted. If there is no decimal point, there is no fractional part. · The fractional part can consist of a maximum of 15 digits.
Exponent	· The exponent consists of a single e or E after the fractional part, a following single plus or minus sign, and the remaining continuous numbers (0 to 9) to the end of the text string. · If there is no fractional part, the text string after the decimal point is taken as the exponent. · If there is no decimal point or fractional part, the above text string after the integer part is taken as the exponent. · The exponent can be omitted. · The numeric part of the exponent can consist of a maximum of three digits.

■ Program example

	LD	ST
Defined variable	In_Var: STRING(1985); Return_Var: BOOL; Out_var: DWORD;	
Program		Return_Var := StringToFixNum(In:=In_Var , Out=>Out_var);
Running result		

■ Precautions

The digits after the fourth decimal in In digit are truncated.

Underlines (16#5F) in the text string In are ignored.

When the value of In does not end with NULL, the return value is FALSE and Out does not change.

When the content of In includes characters that cannot be converted to numbers, the return value is FALSE and Out does not change.

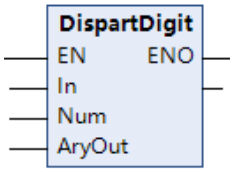
When content of In has a decimal point but not a fractional part, the return value is FALSE and Out does not change.

When the conversion result exceeds the valid range of Out, the return value is FALSE and Out does not change.

3.8. 44 DispartDigit

This instruction separates a bit string into 4-bit units.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
DispartDigit	Four-bit separation	FC		DispartDigit(In:= , Num:= , AryOut:=);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
In	Data to separate	-	Depends on data types	-	Bit string to separate
Num	Number of bits to separate	USINT	Depends on data types	1	Number of bits to separate

In-out variables

In-Out Variable	Name	Data Type	Value Range	Initial Value	Description
AryOut[] (array)	Separation result array	-	16#00 to 16#0F	-	Separation result array

	Boolean	Bit String				Integer								Real number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In	-	√	√	√	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Num	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-
AryOut[]	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

■ Function

This instruction separates the data to separate In into 4-bit units (digits) and stores them in the separation result array AryOut[].

First, In is separated into 4-bit units. Then, the four least-significant bits are stored in AryOut[0]. AryOut[0]

is Byte data, so 16#0 is stored in bits 4 to 7.

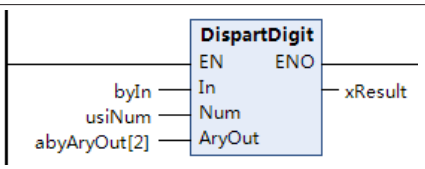
This process is repeated for the number of digits to separate Num.

■ Program example

<table>
<tr><th></th><th>LD</th><th>ST</th></tr>
<tr><td>Defined variable</td><td colspan="2">

```
VAR
    byIn : BYTE := 16#FF;
    usiNum : USINT := 2;
    abyAryOut : ARRAY [0..4] OF BYTE := [0,0,0,0,0];
    xResult : BOOL;
END_VAR
```

</td></tr>
<tr><td>Program</td><td>



</td><td>

```
xResult := DispartDigit(In:= byIn,
    Num:= usiNum,
    AryOut:= abyAryOut[2]);
```

</td></tr>
<tr><td>Running result</td><td colspan="2">

byIn	BYTE	16#FF
usiNum	USINT	16#02
abyAryOut	ARRAY [0..4] OF BYTE	
abyAryOut[0]	BYTE	16#00
abyAryOut[1]	BYTE	16#00
abyAryOut[2]	BYTE	16#0F
abyAryOut[3]	BYTE	16#0F
abyAryOut[4]	BYTE	16#00
xResult	BOOL	TRUE

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

■ Precautions

When the value of Num is 0, the return value is TRUE and AryOut[] does not change.

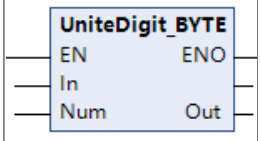
When the value of Num exceeds the valid range, the return value is FALSE and AryOut[] does not change.

When the value of Num exceeds the AryOut[] array, the return value is FALSE and AryOut[] does not change.

3.8. 45 UniteDigit_**

This instruction joins 4-bit units of data into a bit string.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
UniteDigit_**	Four-bit join group	FC		UniteDigit_BYTE(In:= , Num:= , Out=>);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description

In[] (array)	Array to join	-	Depends on data types	-	Array to join
Num	Number of digits to join	USINT	Depends on data types	1	Number of digits to join

Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
Out	Joined result	-	Depends on data types	-	Bit string of the joined result

	Boolean	Bit String				Integer								Real number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In[] (array)	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Num	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Out	-	√	√	√	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

■ Function

This instruction joins 4-bit units (four bits = one digit) from each element of the array to join In[] and creates a bit string in the joined result Out.

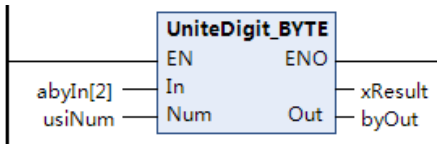
The number of digits to join Num specifies the number of array elements to join. First, the lower four bits from each element from In[0] to In[Num − 1] are joined to create a bit string with Num digits. To this, 16#0 is added to the upper digits for the number of digits in Out minus the value of Num.

The instruction name is determined by the data types of Out. For example, if Out is of the Word data type, the instruction is UniteDigit_WORD.

■ Program example

	LD	ST

Defined variable	VAR
	abyIn : ARRAY [1..10] OF BYTE := [10(1)];
	usiNum : USINT := 2;
	byOut : BYTE;
	xResult : BOOL;
	END_VAR	

Program		xResult := UniteDigit_BYTE(In:= abyIn[2], Num:= usiNum, Out=> byOut);

Running result

abyIn	ARRAY [1..10] OF BYTE	
abyIn[1]	BYTE	16#01
abyIn[2]	BYTE	16#01
abyIn[3]	BYTE	16#01
abyIn[4]	BYTE	16#01
abyIn[5]	BYTE	16#01
abyIn[6]	BYTE	16#01
abyIn[7]	BYTE	16#01
abyIn[8]	BYTE	16#01
abyIn[9]	BYTE	16#01
abyIn[10]	BYTE	16#01
usiNum	USINT	16#02
byOut	BYTE	16#11
xResult	BOOL	TRUE

■ Precautions

When the value of Num is 0, the value of Out is 0 and the return value is TRUE.

When the value of Num exceeds the valid range, the return value is FALSE and Out does not change.

When the value of Num exceeds the In[] array, the return value is FALSE and Out does not change.

3.8. 46 Dispart8Bit

This instruction separates a bit string into individual bytes.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
Dispart8Bit	Byte data separation	FC	Dispart8Bit EN ENO In Num AryOut	Dispart8Bit(In:= , Num:= , AryOut:=);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
In	Data to separate	-	Depends on data types	-	Bit string to separate
Num	Number of bytes to separate	USINT	Depends on data types	1	Number of bytes to separate

In-out variables

In-Out Variable	Name	Data Type	Value Range	Initial Value	Description
AryOut[] (array)	Separation result array	-	Depends on data types	-	Separation result array

	Bool-ean	Bit String				Integer								Real num-ber		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In	-	√	√	√	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Num	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-
AryOut[] (array)	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

■ Function

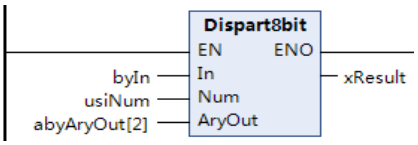
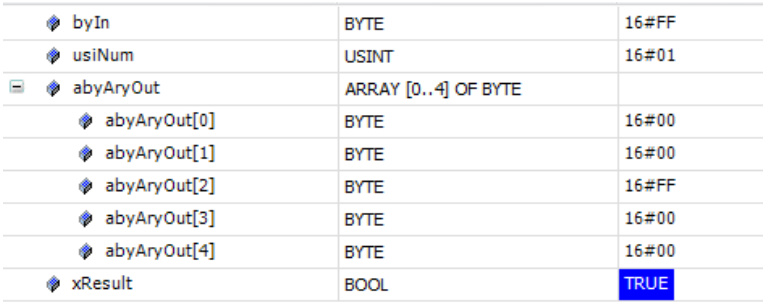
This instruction separates the data to separate In into individual bytes and stores them in the separation result array AryOut[].

First, In is separated into individual bytes. Then, the lowest byte is stored in AryOut[0]. Then, the next byte is stored in AryOut[1]. This process is repeated for the number of bytes to separate Num.

■ Program example

	LD	ST

Defined variable	VAR
	byIn : BYTE := 16#FF;
	usiNum : USINT := 1;
	abyAryOut : ARRAY [0..4] OF BYTE := [0,0,0,0,0];
	xResult : BOOL;
	END_VAR	

Program		xResult := Dispart8bit(In:= byIn, Num:= usiNum, AryOut:= abyAryOut[2]);
Running result		

■ Precautions

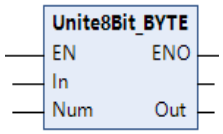
When the value of Num exceeds the valid range, the return value is FALSE and AryOut[] does not change.

When the value of Num exceeds the AryOut[] array, the return value is FALSE and AryOut[] does not change.

3.8. 47 Unite8Bit_**

This instruction joins bytes of data into a bit string.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
Unite8Bit_**	Byte data join group	FC		Unite8Bit_BYTE(In:= , Num:= , Out=>);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
In[] (array)	Array to join	-	Depends on data types	-	Array to join
Num	Number of bytes to join	USINT	Depends on data types	1	Number of bytes to join

Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
Out	Joined result	-	Depends on data types	-	Bit string of the joined result

	Boolean	Bit String				Integer								Real number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In[] (array)	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Num	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Out	-	√	√	√	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

■ Function

This instruction joins elements of the array to join In[] to create a bit string in the joined result Out.

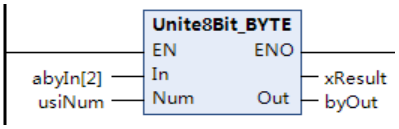
The number of bytes to join Num specifies the number of array elements to join. First, In[0] to In[Num − 1] are joined to create a bit string with Num bytes. To this, 16#00 is added to the upper bytes for the number of bytes in Out minus the value of Num.

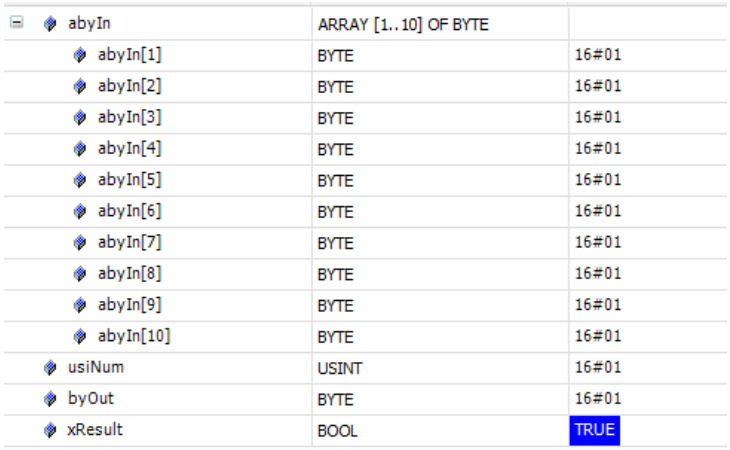
The instruction name is determined by the data types of Out. For example, if Out is of the Dword data type, the instruction is Unite8Bit_DWORD.

■ Program example

	LD	ST

Defined variable	VAR
	abyIn : ARRAY [1..10] OF BYTE := [10(1)];
	usiNum : USINT := 1;
	byOut : BYTE;
	xResult : BOOL;
	END_VAR	

Program		`xResult := Unite8Bit_BYTE(In:= abyIn[2],`
		`Num:= usiNum,`
		`Out=> byOut);`

Running result		

abyIn	ARRAY [1..10] OF BYTE	
abyIn[1]	BYTE	16#01
abyIn[2]	BYTE	16#01
abyIn[3]	BYTE	16#01
abyIn[4]	BYTE	16#01
abyIn[5]	BYTE	16#01
abyIn[6]	BYTE	16#01
abyIn[7]	BYTE	16#01
abyIn[8]	BYTE	16#01
abyIn[9]	BYTE	16#01
abyIn[10]	BYTE	16#01
usiNum	USINT	16#01
byOut	BYTE	16#01
xResult	BOOL	TRUE

■ Precautions

When the value of Num is 0, the value of Out is initialized to 0 and the return value is TRUE.

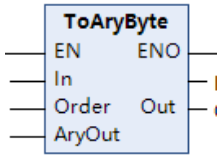
When the value of Num exceeds the valid range, the return value is FALSE and the value of Out is initialized to 0.

When the value of Num exceeds the In[] array, the return value is FALSE and the value of Out is initialized to 0.

3.8. 48 ToAryByte

This instruction separates a variable into bytes and stores the bytes in a Byte array.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
ToAryByte	Conversion to byte array	FC		ToAryByte(In:= , Order:= , AryOut:= , Out=>);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
In	Data to convert	-	Depends on data types	-	Data to convert
Order	Conversion order	-	Depends on data types	_LOW_HIGH	Conversion order

In-out variables

In-Out Variable	Name	Data Type	Value Range	Initial Value	Description
AryOut[] (array)	Conversion result array	-	Depends on data types	-	Conversion result array

Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
Out	Number of elements in the conversion result	UINT	Depends on data types	-	Number of elements in the conversion result

	Boolean	Bit String				Integer								Real number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√
	An enumeration, an array, an array element, a structure, or a structure member can also be specified.																			
Order	See "Function" for the enumerators for the enumerated type _eBYTE_ORDER.																			
AryOut[] (array)	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Out	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-

■ Function

This instruction separates the data to convert In into individual bytes and stores them in order in the conversion result array AryOut[] starting from AryOut[0]. The number of elements in the result Out contains the number of elements stored in AryOut[].

The conversion order Order specifies the order in which to convert the value of In to bytes. The data type of Order is enumerated type _eBYTE_ORDER.

The meanings of the enumerators are as follows.

Enumerator	Meaning
_LOW_HIGH	Lower byte first, higher byte last
_HIGH_LOW	Higher byte first, lower byte last

When the data type of In is two bytes or more:

In is separated into bytes and stored in AryOut[]. The following data types have two bytes or more.

Category	Data Type
Bit string	WORD, DWORD, and LWORD
Integer	UINT, UDINT, ULINT, INT, DINT, and LINT
Real number	REAL and LREAL

Time, duration, date, and text string	TIME, DATE, TOD, DT, and STRING types of two bytes or more
Others	An enumeration An array for which the total for all elements is two bytes or more An array element of two bytes or more A structure for which the total for all members is two bytes or more A structure member of two bytes or more

The processing steps are as follows:

1. First, the value in In is separated into words (two bytes).

2. The lowest word (two bytes) is separated into bytes.

3. If the value of Order is _LOW_HIGH, the lower byte is stored in AryOut[0] and the higher byte is stored in AryOut[1]. If the value of Order is _HIGH_LOW, the higher byte is stored in AryOut[0] and the lower byte is stored in AryOut[1].

4. The next word is separated into bytes and stored in AryOut[2] and AryOut[3] in the same way.

5. This process is repeated to the end of the value of In. If In is an array, the same process is repeated to the last element in In.

When the data type of In is one byte:

In is stored in AryOut[] as one byte. The following data types have one byte.

Category	Data Type
Bit String	BYTE
Integer	USINT and SINT
Real number	None
Time, duration, date, and text string	One-byte STRING
Others	An array for which the total for all elements is one byte An array element of one byte A structure for which the total for all members is one byte A structure member of one byte

The following storage method is used.

Value of Order	In (Number or Not)	Storage Method in AryOut[]
_LOW_HIGH	Not an integer	The value of In is stored in AryOut[0].
	Integer	The value of In[i] is stored in AryOut[i].

_HIGH_LOW	Not an array	The value of In is stored in AryOut[1]. 16#00 is stored in AryOut[0].
	Array	In[i] (where i is even) is stored in AryOut[i+1]. In[i] (where i is odd) is stored in AryOut[i-1]. If the number of elements in In[] is odd, 16#00 is stored last in AryOut[n − 1].

When the data type of In is Bool (one bit):

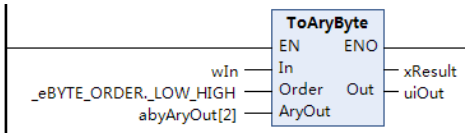
Data is stored in AryOut[] as described below.

Value of Order	In (Array or Not)	Storage Method in AryOut[]
_LOW_HIGH	Not an array	The logical AND of the value of In and 16#00 is stored in AryOut[0].
	Array	The value of In[0] is stored in AryOut[0]. The value of In[1] is stored in AryOut[1]. The same process is repeated to store the rest of the data. The value of Out is always even.
_HIGH_LOW	Not an array	The value of In[0] is stored in AryOut[0]. The value of In[1] is stored in AryOut[1]. The same process is repeated to store the rest of the data. The value of Out is always even.
	Array	The value of In[0] is stored in AryOut[1]. The value of In[1] is stored in AryOut[0]. The value of Out is always even.

■ Program example

	LD	ST
Defined variable		
Program		
Running result		

```
VAR
    wIn          : WORD := 16#FF00;
    abyAryOut    : ARRAY [0..4] OF BYTE;
    uiOut        : UINT;
    xResult      : BOOL;
END_VAR
```




```
xResult := ToAryByte(In:= wIn,
    Order:= _eBYTE_ORDER._LOW_HIGH,
    AryOut:= abyAryOut[2],
    Out=> uiOut);
```

wIn	WORD	16#FF00
abyAryOut	ARRAY [0..4] OF BYTE	
abyAryOut[0]	BYTE	16#00
abyAryOut[1]	BYTE	16#00
abyAryOut[2]	BYTE	16#00
abyAryOut[3]	BYTE	16#FF
abyAryOut[4]	BYTE	16#00
uiOut	UINT	16#0002
xResult	BOOL	TRUE

■ Precautions

Always use a variable for the input parameter to move to In. If a constant is moved, an error occurs during compilation.

When In is an enumeration, you cannot directly move an enumerator to it. If an enumerator is moved to it

directly, an error occurs during compilation.

When In is STRING data, the text string is not converted to numbers. The content of the variable is taken as a bit string and converted to a byte array.

When In is a structure, adjustment areas between members may be inserted into AryOut[].

When the value of Order is _HIGH_LOW and the total number of bytes in In is an odd number, 16#00 is added to the end of In to make an even number of bytes before the conversion is started.

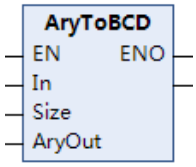
When the value of Order exceeds the valid range, the return value is FALSE and Out and AryOut[] do not change.

When the conversion result exceeds the AryOut[] array, the return value is FALSE and Out and AryOut[] do not change.

3.8. 49 AryToBCD

This instruction converts the elements of an unsigned integer array to BCD bit strings.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
AryToBCD	Array-to-BCD conversion	FC		AryToBCD(In :=, Size :=, AryOut :=);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
In[] (array)	Unsigned integer array	-	Depends on data types	-	Unsigned integer array
Size	Number of elements to convert	UINT	Depends on data types	1	Number of elements in In[] to convert

In-out variables

In-Out Variable	Name	Data Type	Value Range	Initial Value	Description
AryOut[] (array)	BCD array	-	Depends on data types	-	BCD array

	Boolean	Bit String				Integer								Real number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In[] (array)	-	-	-	-	-	√	√	√	√	-	-	-	-	-	-	-	-	-	-	-
Size	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-

	Bool-ean	Bit String				Integer								Real num-ber		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
AryOut[] (array)	Array with the same data type as the elements of In[]																			

■ Function

This instruction converts Size elements of the unsigned integer array In[] starting from In[0] to a BCD bit string, and outputs the BCD bit string to the BCD array AryOut[].

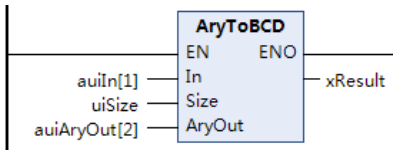
The following table lists the valid ranges for In[] and AryOut[] according to the data types of their elements.

Data Type of In[] Element	Valid Range of In[]	Valid Range of AryOut[]
USINT	0 to 99	16#00 to 16#99 (BCD)
UINT	0 to 9999	16#0000 to 16#9999 (BCD)
UDINT	0 to 99999999	16#0000_0000 to 16#9999_9999 (BCD)
ULINT	0 to 9999999999999999	16#0000_0000_0000_0000 to 16#9999_9999_9999_9999 (BCD)

■ Program example

	LD	ST
Defined variable		
Program		
Running result		

```
VAR
    auiIn       : ARRAY [0..4] OF UINT := [1,23,456,7,89];
    uiSize      : UINT := 3;
    auiAryOut   : ARRAY [0..4] OF UINT;
    xResult     : BOOL;
END_VAR
```




```
xResult := AryToBCD(In:= auiIn[1],
    Size:= uiSize,
    AryOut:= auiAryOut[2]);
```

auiIn	ARRAY [0..4] OF UINT	
auiIn[0]	UINT	16#0001
auiIn[1]	UINT	16#0017
auiIn[2]	UINT	16#01C8
auiIn[3]	UINT	16#0007
auiIn[4]	UINT	16#0059
uiSize	UINT	16#0003
auiAryOut	ARRAY [0..4] OF UINT	
auiAryOut[0]	UINT	16#0000
auiAryOut[1]	UINT	16#0000
auiAryOut[2]	UINT	16#0023
auiAryOut[3]	UINT	16#0456
auiAryOut[4]	UINT	16#0007
xResult	BOOL	TRUE

■ Precautions

This instruction does not convert signed BCD. Therefore, use an unsigned integer (USINT, UINT, UDINT, or ULINT) as the data type of In[].

When the value of Size is 0, the return value is TRUE and AryOut[] does not change.

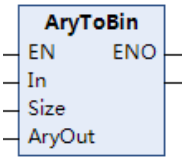
When the value of In[] exceeds the valid range, the return value is FALSE and AryOut[] does not change.

When the value of Size exceeds the In[] or AryOut[] array, the return value is FALSE and AryOut[] does not change.

3.8. 50 AryToBin

This instruction converts the elements of a BCD array into bit strings.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
AryToBin	Array-to-bit string conversion	FC		AryToBin (In :=, Size :=, AryOut :=);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
In[] (array)	BCD array	-	Depends on data types	-	BCD array
Size	Number of elements to convert	UINT	Depends on data types	1	Number of elements in In[] to convert

In-out variables

In-Out Variable	Name	Data Type	Value Range	Initial Value	Description
AryOut[] (array)	Bit string array	-	Depends on data types	-	Bit string after conversion

	Boolean	Bit String				Integer								Real number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In[] (array)	-	√	√	√	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Size	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-
AryOut[] (array)	Array with the same data type as the elements of In[]																			

■ Function

This instruction converts Size elements of the BCD array In[] starting from In[0] to bit strings, and outputs the bit strings to the BCD array AryOut[].

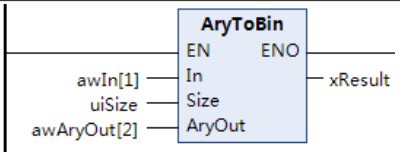
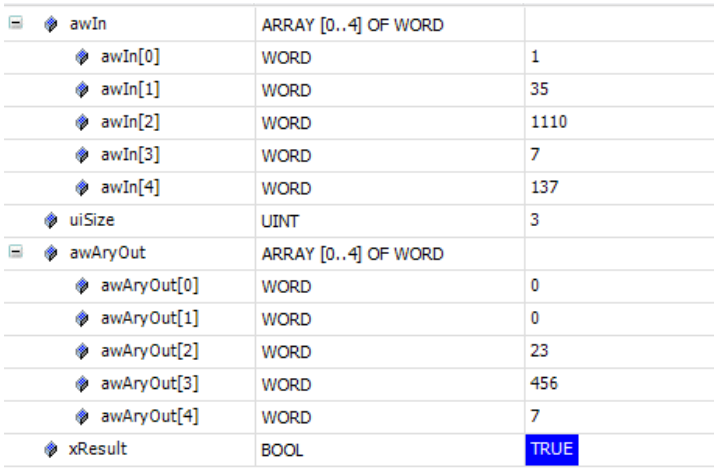
The following table lists the valid ranges for In[] and AryOut[] according to the data types of their elements.

Data Type of In[] Element	Valid Range of In[]	Valid Range of AryOut[]
BYTE	16#00 to 16#99 (BCD)	0 to 99
WORD	16#0000 to 16#9999 (BCD)	0 to 9999
DWORD	16#0000_0000 to 16#9999_9999 (BCD)	0 to 99999999
LWORD	16#0000_0000_0000_0000 to 16#9999_9999_9999_9999 (BCD)	0 to 9999999999999999

■ Program example

	LD	ST

Defined variable	VAR
	awIn : ARRAY [0..4] OF WORD := [16#1,16#23,16#456,16#7,16#89];
	uiSize : UINT := 3;
	awAryOut : ARRAY [0..4] OF WORD;
	xResult : BOOL;
	END_VAR	

Program		xResult := AryToBin(In:= awIn[1], Size:= uiSize, AryOut:= awAryOut[2]);
Running result		

awIn	ARRAY [0..4] OF WORD	
awIn[0]	WORD	1
awIn[1]	WORD	35
awIn[2]	WORD	1110
awIn[3]	WORD	7
awIn[4]	WORD	137
uiSize	UINT	3
awAryOut	ARRAY [0..4] OF WORD	
awAryOut[0]	WORD	0
awAryOut[1]	WORD	0
awAryOut[2]	WORD	23
awAryOut[3]	WORD	456
awAryOut[4]	WORD	7
xResult	BOOL	TRUE

■ Precautions

This instruction does not convert signed BCD. Therefore, use a bit string (BYTE, WORD, DWORD, or LWORD) as the data type of AryOut[].

When the value of Size is 0, the return value is TRUE and AryOut[] does not change.

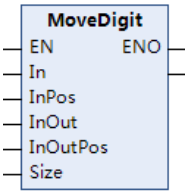
When the value of In[] exceeds the valid range, the return value is FALSE and AryOut[] does not change.

When the value of Size exceeds the In[] or AryOut[] array, the return value is FALSE and AryOut[] does not change.

3.8. 51 MoveDigit

This instruction moves digits (4 bits for each digit) in a bit string.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
MoveDigit	Move digit	FC		MoveDigit (In :=, InPos :=, InOut :=, InOutPos :=, Size :=);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
In	Move source	-	Depends on data types	-	Move source
InPos	Move source digit	USINT	Depends on data types	0	Position of digit in In to move
InOut	Move destination	-	Depends on data types	-	Move destination
InOutPos	Move destination digit	USINT	Depends on data types	0	Position of digit in Out to receive the digit
Size	Number of digits	USINT	Depends on data types	1	Number of digits to move

	Boolean	Bit String				Integer								Real number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In	-	√	√	√	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
InPos	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-
InOut	-	√	√	√	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
InOutPos	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Size	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-

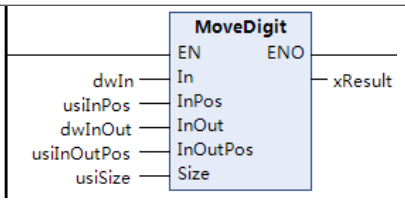
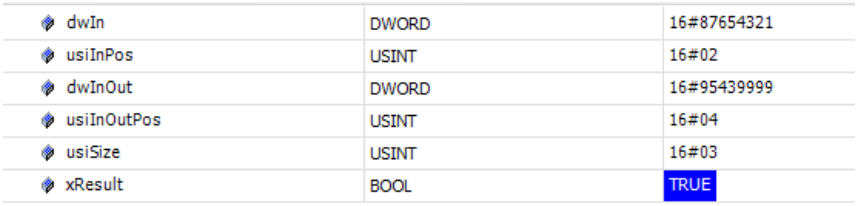
■ Function

This instruction moves Size digits (four bits for each digit) from the InPos digit in the move source In to the InOutPos digit in the move destination InOut.

■ Program example

	LD	ST

Defined variable	VAR
	dwIn : DWORD := 16#87654321;
	usiInPos : USINT := 2;
	dwInOut : DWORD := 16#99999999;
	usiInOutPos : USINT := 4;
	usiSize : USINT := 3;
	xResult : BOOL;
	END_VAR

Program	 xResult := MoveDigit(In:= dwIn, InPos:= usiInPos, InOut:= dwInOut, InOutPos:= usiInOutPos, Size:= usiSize);
Running result	

■ Precautions

If the position of the digit at the destination exceeds the most-significant digit of InOut, the remaining digits are stored in the least-significant digits of InOut.

If the position of the digit at the source exceeds the most-significant digit of In, the remaining digits are moved to the least-significant digits of In.

When the value of Size is 0, the return value is TRUE and InOut does not change.

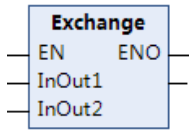
An error occurs in the following cases. The return value changes to FALSE and InOut does not change.

- The value of InPos exceeds the valid range.
- The value of InOutPos exceeds the valid range.
- The value of Size exceeds the valid range.

3.8. 52 Exchange

This instruction exchanges the values of two variables.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
Exchange	Data exchange	FC		Exchange (InOut1 :=, InOut2 :=);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description

InOut1	Data to exchange	-	Depends on data types	-	Data to exchange
InOut2	Data to exchange	-	Depends on data types	-	Data to exchange

	Bool-ean	Bit String				Integer								Real num-ber		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
InOut1	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√
	An enumeration, a structure, or a structure member can also be specified.																			
InOut2	Array with the same data type as the elements of InOut1																			

■ Function

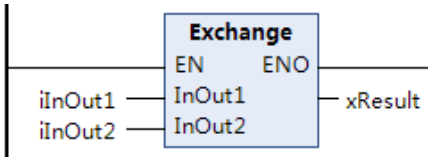
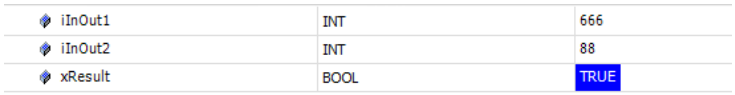
This instruction exchanges values of InOut1 and InOut2.

An enumeration, a structure, or a structure member can also be specified for InOut1 and InOut2.

■ Program example

	LD	ST

Defined variable	VAR
	iInOut1 : INT := 88;
	iInOut2 : INT := 666;
	xResult : BOOL;
	END_VAR	

Program		xResult := Exchange(InOut1:= iInOut1, InOut2:= iInOut2);
Running result		

■ Precautions

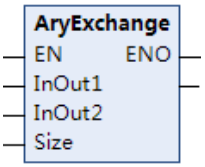
Use the same data type for InOut1 and InOut2. If they are different, an error occurs during compilation.

When both InOut1 and InOut2 are STRING data and the length of the text string in one of them does not fit into the other, the return value is FALSE and InOut1 and InOut2 do not change.

3.8. 53 AryExchange

This instruction exchanges the elements of two arrays.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
AryExchange	Array data exchange	FC		AryExchange (InOut1 :=, InOut2 :=, Size :=);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
Size	Number of elements to exchange	UINT	Depends on data types	1	Number of elements to exchange

In-out variables

In-Out Variable	Name	Data Type	Value Range	Initial Value	Description
InOut1[]	Arrays to exchange	-	Depends on data types	-	Arrays to exchange
InOut2[]	Arrays to exchange	-	Depends on data types	-	Arrays to exchange

	Boolean	Bit String				Integer								Real number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Size	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-
InOut1[]	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√
	Arrays of enumerations or structures can also be specified.																			
InOut2[]	Array with the same data type as the elements of InOut1[]																			

■ Function

This instruction exchanges values of InOut1 and InOut2.

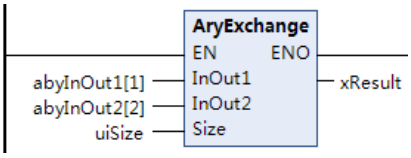
It exchanges Size elements of the array to exchange InOut1[] from InOut1[0] with Size elements of the array to exchange InOut2[] from InOut2[0].

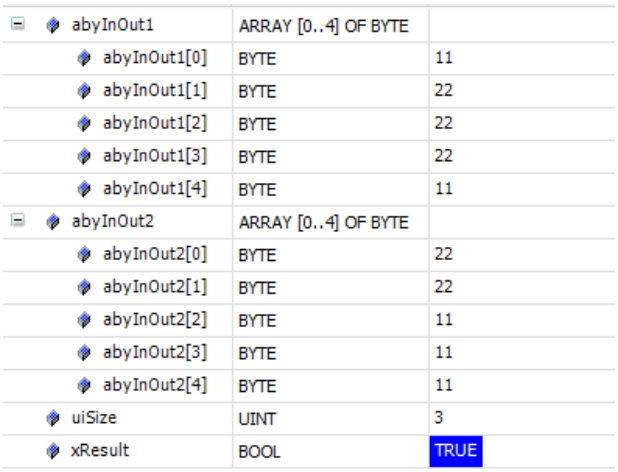
■ Program example

	LD	ST
Defined variable		

```
VAR
    abyInOut1   : ARRAY [0..4] OF BYTE := [5(11)];
    abyInOut2   : ARRAY [0..4] OF BYTE := [5(22)];
    uiSize      : UINT := 3;
    xResult     : BOOL;
END_VAR
```

	LD	ST

Program		`xResult := AryExchange(InOut1:= abyInOut1[1],`
		`InOut2:= abyInOut2[2],`
		`Size:= uiSize);`

Running result		

abyInOut1	ARRAY [0..4] OF BYTE	
abyInOut1[0]	BYTE	11
abyInOut1[1]	BYTE	22
abyInOut1[2]	BYTE	22
abyInOut1[3]	BYTE	22
abyInOut1[4]	BYTE	11
abyInOut2	ARRAY [0..4] OF BYTE	
abyInOut2[0]	BYTE	22
abyInOut2[1]	BYTE	22
abyInOut2[2]	BYTE	11
abyInOut2[3]	BYTE	11
abyInOut2[4]	BYTE	11
uiSize	UINT	3
xResult	BOOL	TRUE

■ Precautions

Use the same data type for InOut1 and InOut2. If they are different, an error occurs during compilation.

When the value of Size is 0, the return value is TRUE and InOut1[] and InOut2[] do not change.

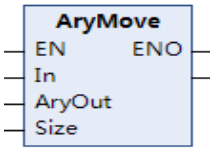
When both InOut1 and InOut2 are STRING data and the length of the text string in one of them does not fit into the other, the return value is FALSE and InOut1[] and InOut2[] do not change.

When the value of Size exceeds the InOut1[] or InOut2[] array, the return value is FALSE and InOut1[] and InOut2[] do not change.

3.8. 54 AryMove

This instruction moves multiple array elements.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
AryMove	Array move	FC		AryMove (In :=, AryOut :=, Size :=);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
In[]	Move source array	-	Depends on data types	-	Array to move

In-out variables

In-Out Variable	Name	Data Type	Value Range	Initial Value	Description
AryOut[]	Move result array	-	Depends on data types	-	Move result array
Size	Number of elements to move	UINT	Depends on data types	1	Number of elements to move

	Boolean	Bit String				Integer								Real number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In[] (array)	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√
	Arrays of enumerations or structures can also be specified.																			
Size	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-
AryOut[] (array)	Array with the same data type as the elements of In[]																			

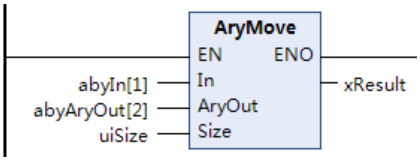
■ Function

This instruction moves Size elements of the move source array In[] starting from In[0] to the move result array AryOut[] starting from AryOut[0].

■ Program example

	LD	ST
Defined variable		
Program		
Running result		

```
VAR
    abyIn       : ARRAY [0..4] OF BYTE := [5(11)];
    abyAryOut   : ARRAY [0..4] OF BYTE := [5(22)];
    uiSize      : UINT := 2;
    xResult     : BOOL;
END_VAR
```




```
xResult := AryMove(In:= abyIn[1],
    AryOut:= abyAryOut[2],
    Size:= uisize);
```

abyIn	ARRAY [0..4] OF BYTE	
abyIn[0]	BYTE	11
abyIn[1]	BYTE	11
abyIn[2]	BYTE	11
abyIn[3]	BYTE	11
abyIn[4]	BYTE	11
abyAryOut	ARRAY [0..4] OF BYTE	
abyAryOut[0]	BYTE	22
abyAryOut[1]	BYTE	22
abyAryOut[2]	BYTE	11
abyAryOut[3]	BYTE	11
abyAryOut[4]	BYTE	22
uiSize	UINT	2
xResult	BOOL	TRUE

■ Precautions

Use the same data type for In[] and AryOut[]. If they are different, an error occurs during compilation.

When the value of Size is 0, the return value is TRUE and AryOut[] does not change.

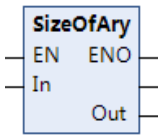
When both In[] and AryOut[] are STRING data and the length of the text string in In[] is greater than the memory size of AryOut[], the return value is FALSE and AryOut[] does not change.

When the value of Size exceeds the In[] and AryOut[] array, the return value is FALSE and AryOut[] does not change.

3.8. 55 SizeOfAry

This instruction gets the number of elements in an array.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
SizeOfAry	Get number of array elements	FC		SizeOfAry (In :=, Out =>);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
In[] (array)	Array	-	Depends on data types	-	Array

Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
Out	Number of elements	UINT	Depends on data types	-	Number of elements

	Boolean	Bit String				Integer								Real number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In[] (array)	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√
	Arrays of enumerations or structures can also be specified.																			
Out	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-

■ Function

This instruction gets the number of elements in array In[].

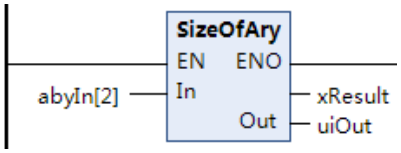
For the input parameter, use an array name, such as array, and not an array element name, such as array[0].

■ Program example

<table>
<tr><th></th><th>LD</th><th>ST</th></tr>
<tr><td>Defined variable</td><td colspan="2">

```
VAR
    abyIn       : ARRAY [0..4] OF BYTE := [1,2,3,4,5];
    uiOut       : UINT;
    xResult     : BOOL;
END_VAR
```

</td></tr>
<tr><td>Program</td><td>



</td><td>

```
xResult := SizeOfAry(In:= abyIn[2],
    Out=> uiOut);
```

</td></tr>
<tr><td>Running result</td><td colspan="2">

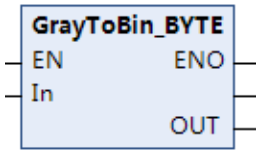
abyIn	ARRAY [0..4] OF BYTE	
abyIn[0]	BYTE	1
abyIn[1]	BYTE	2
abyIn[2]	BYTE	3
abyIn[3]	BYTE	4
abyIn[4]	BYTE	5
uiOut	UINT	5
xResult	BOOL	TRUE

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

3.8. 56 GrayToBin_**

This instruction converts a gray code to a bit string.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
GrayToBin_**	Gray code-to-binary code conversion group	FC		GrayToBin_BYTE(In :=, Out =>);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
In	Data to convert	-	Depends on data types	0	Data to convert

Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
Out	Conversion result	-	Depends on data types	-	Conversion result

	Bool-ean	Bit String				Integer								Real number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In	-	√	√	√	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Out	Data with the same data type as In																			

■ Function

This instruction converts a gray code in the data to convert In to a bit string.

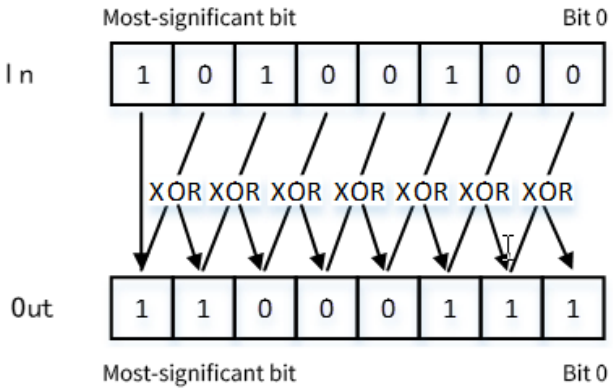
The instruction name is determined by the data types of In and Out. For example, if In and Out are of the Word data type, the instruction is GrayToBin_WORD.

When In and Out are Byte data, the conversion steps are as follows:

1. The most-significant bit (bit 7) of In is used as that of Out.

2. A logical exclusive OR is taken of the value of bit 6 in In and the value of bit 7 in Out. The result is used as bit 6 of Out.

3. This process is repeated through the least-significant bit (bit 0) of Out.



■ Program example

	LD	ST
Defined variable	VAR byIn : BYTE := 2#10100100; byOut : BYTE; xResult : BOOL; END_VAR	
Program	GrayToBin_BYTE EN ENO byIn — In OUT — xResult, byOut	xResult := GrayToBin_BYTE(In:= byIn, OUT=> byOut);
Running result	byIn BYTE 2#10100100 byOut BYTE 2#11000111 xResult BOOL TRUE	

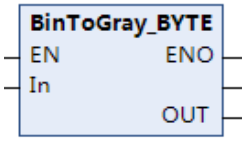
■ Precautions

Use the same data type for In and Out. If they are different, an error occurs during compilation.

3.8. 57 BinToGray_**

This instruction converts a bit string to a gray code.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
BinToGray_**	Binary code-to-gray code conversion	FC		BinToGray_BYTE(In :=, Out =>);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
In	Data to convert	-	Depends on data types	0	Data to convert

Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
Out	Conversion result	-	Depends on data types	-	Conversion result

	Boolean	Bit String				Integer								Real number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In	-	√	√	√	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Out	Data with the same data type as In																			

■ Function

This instruction converts the bit string in the data to convert In to a gray code.

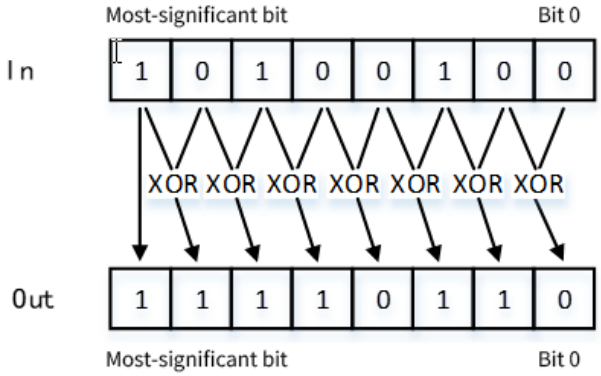
The instruction name is determined by the data types of In and Out. For example, if In and Out are of the Word data type, the instruction is BinToGray_WORD.

When In and Out are Byte data, the conversion steps are as follows:

1. The most-significant bit (bit 7) of In is used as that of Out.

2. An logical exclusive OR is taken of the value of bit 7 in In and the value of bit 6 in In. The result is used as bit 6 of Out.

3. This process is repeated through the least-significant bit (bit 0) of Out.



■ Program example

	LD	ST
Defined variable	VAR byIn : BYTE := 2#10100100; byOut : BYTE; xResult : BOOL; END_VAR	
Program	BinToGray_BYTE EN ENO byIn — In OUT — byOut xResult	xResult := BinToGray_BYTE(In:= byIn, OUT=> byOut);
Running result	byIn BYTE 2#10100100 byOut BYTE 2#11110110 xResult BOOL TRUE	

■ Precautions

Use the same data type for In and Out. If they are different, an error occurs during compilation.

3. 9 Math Instructions

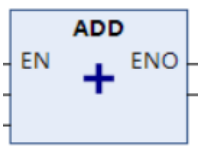
3.9. 1 Instruction List

Instruction Category	Name	FB/FC	Function
Math instructions	ADD	FC	Addition
	SUB	FC	Subtraction
	MUL	FC	Multiplication
	DIV	FC	Division
	MOD	FC	Modulo-division
	ABS	FC	Absolute value
	SQRT	FC	Square root
	LN	FC	Natural logarithm
	LOG	FC	Logarithm base 10
	EXP	FC	Natural exponential operation
	EXPT	FC	Exponentiation
	SIN	FC	Sine
	COS	FC	Cosine
	TAN	FC	Tangent
	ASIN	FC	Arcsine
	ACOS	FC	Arc cosine
	ATAN	FC	Arc tangent
	RAD	FC	Degrees to radians
	DEG	FC	Radians to degrees
	ArySD	FC	Array element standard deviation
	RoundUp	FC	Round up real number (Rounds up a real number at the first decimal digit to make an integer)
	MovingAverage	FC	Moving average
	Inc	FC	Increment
	Dec	FC	Decrement
	AryAddV	FC	Array value addition
	ArySubV	FC	Array value subtraction
	CheckReal	FC	Real number check
	AryMean	FC	Array mean
	ModReal	FC	Real number modulo-division
	ModReal_LR	FC	Long real number modulo-division
	Rand	FB	Random number

3.9. 2 ADD

This instruction adds two input values on the left side and outputs the addition result from the right side.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
ADD	Addition	FC		a1:=a2+a3;

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
In1	Value to add	-	Depends on data types	-	Input
In2	Value to add	-	Depends on data types	-	Input

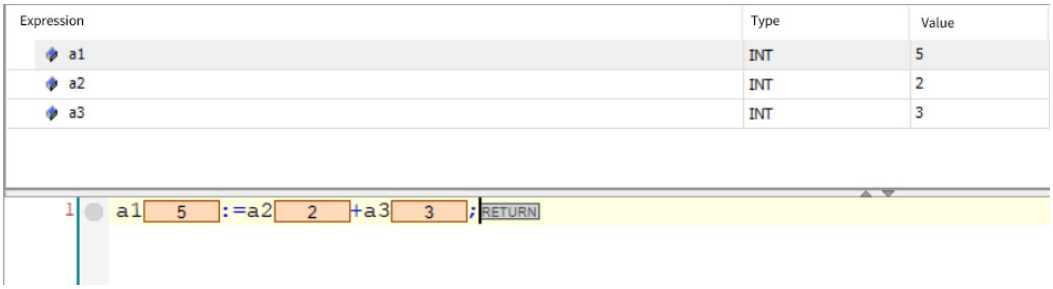
Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
OUT	Output value	-	Depends on data types	-	Output

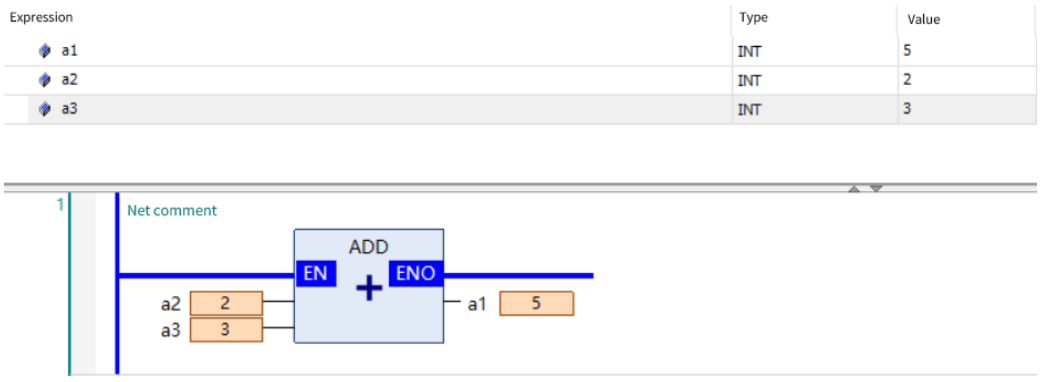
	Bool-ean	Bit String				Integer								Real number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In1	-	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	-	-	-	-
In2	-	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	-	-	-	-
Out	-	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	-	-	-	-

■ Program example

ST:



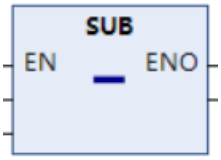
LD:



3.9. 3 SUB

This instruction subtracts two input values on the left side and outputs the subtraction result from the right side.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
SUB	Subtraction	FC		a1:=a2-a3;

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
In1	Subtrahend	-	Depends on data types	-	Input
In2	Minuend	-	Depends on data types	-	Input

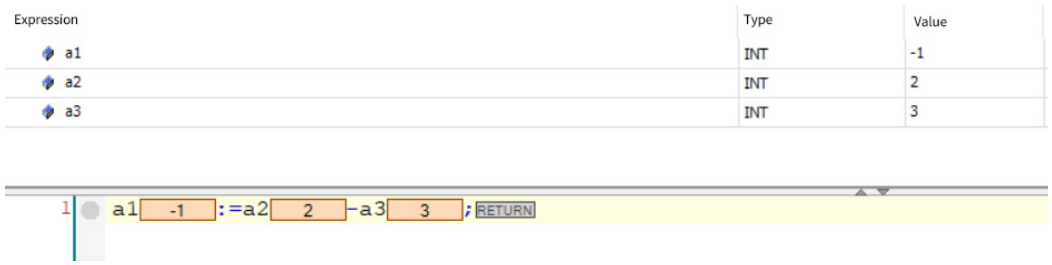
Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
OUT	Output value	-	Depends on data types	-	Output

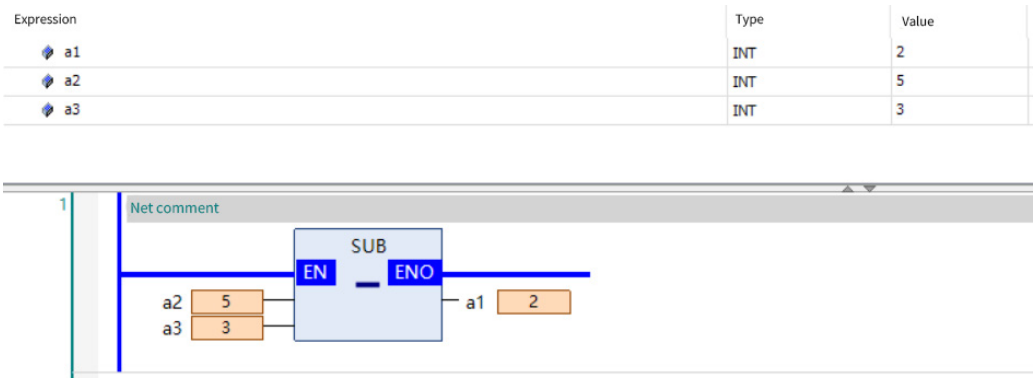
	Bool-ean	Bit String				Integer								Real number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In1	-	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	-	-	-	-
In2	-	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	-	-	-	-
Out	-	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	-	-	-	-

■ Program example

ST:



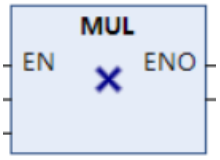
LD:



3.9. 4 MUL

This instruction multiplies two input values on the left side and outputs the multiplication result from the right side.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
MUL	Multiplication	FC		a1:=a2*a3;

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
In1	Value to multiply	-	Depends on data types	-	Input
In2	Value to multiply	-	Depends on data types	-	Input

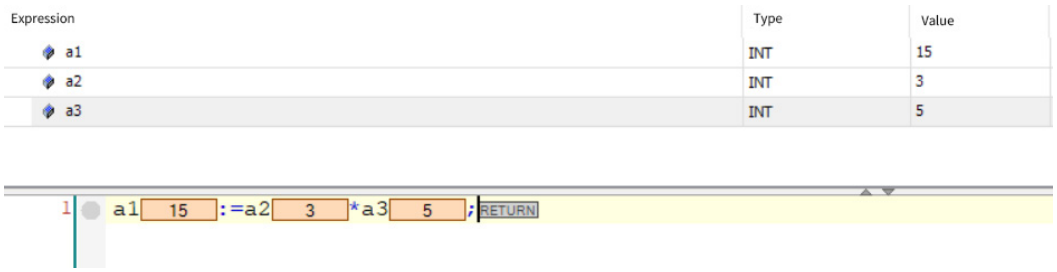
Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
OUT	Output value	-	Depends on data types	-	Output

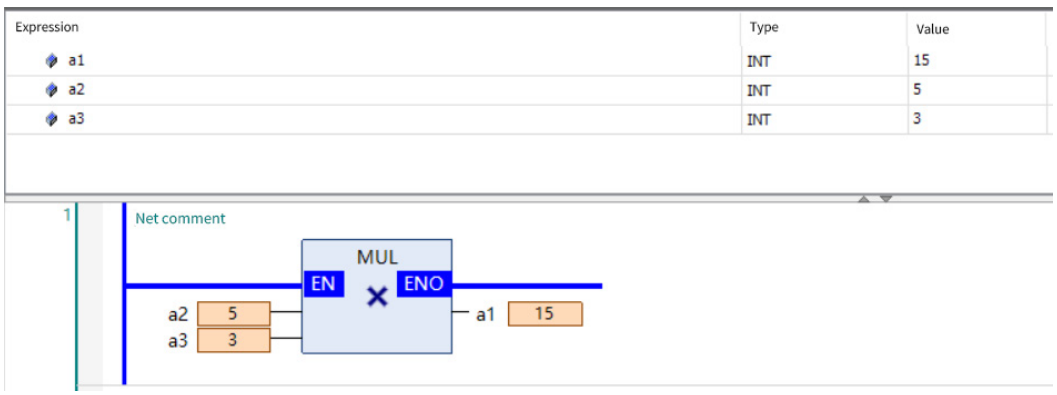
	Bool-ean	Bit String				Integer								Real number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In1	-	√	√	√	√	√	√	√	√	√	√	√	√	√	√	-	-	-	-	-
In2	-	√	√	√	√	√	√	√	√	√	√	√	√	√	√	-	-	-	-	-
Out	-	√	√	√	√	√	√	√	√	√	√	√	√	√	√	-	-	-	-	-

■ Program example

ST:



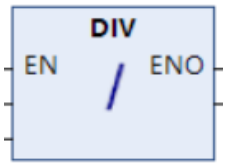
LD:



3.9. 5 DIV

This instruction divides two input values on the left side and outputs the division result from the right side.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
DIV	Division	FC		a1:=a2/a3;

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
In1	Dividend	-	Depends on data types	-	Input
In2	Divisor	-	Depends on data types	-	Input

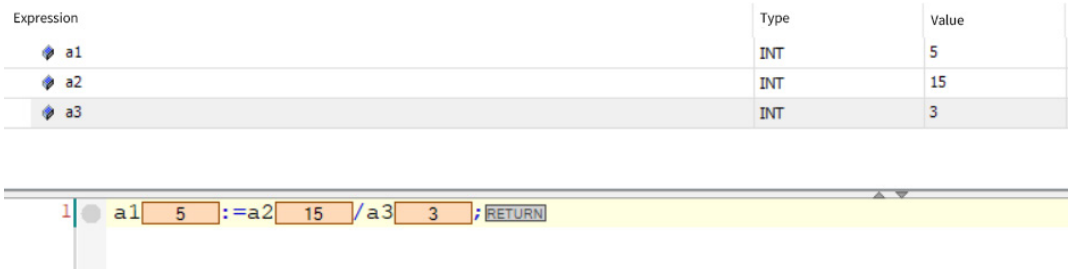
Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
OUT	Output value	-	Depends on data types	-	Output

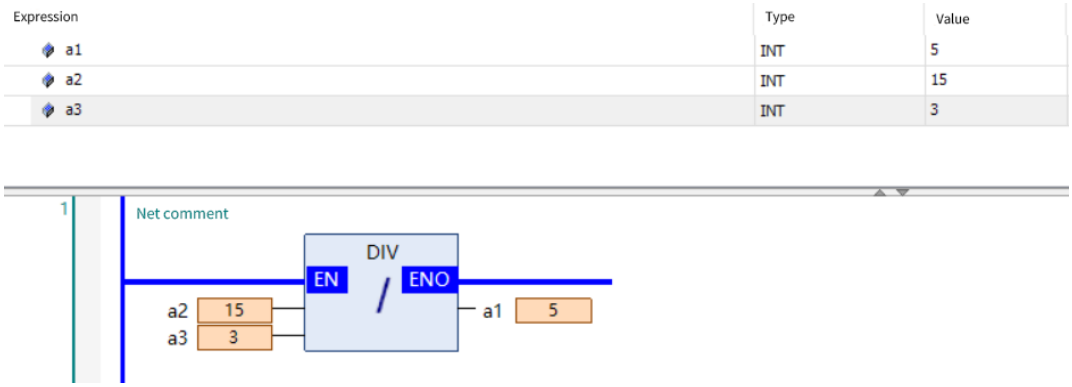
	Bool-ean	Bit String				Integer								Real number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In1	-	√	√	√	√	√	√	√	√	√	√	√	√	√	√	-	-	-	-	-
In2	-	√	√	√	√	√	√	√	√	√	√	√	√	√	√	-	-	-	-	-
Out	-	√	√	√	√	√	√	√	√	√	√	√	√	√	√	-	-	-	-	-

■ Program example

ST:



LD:



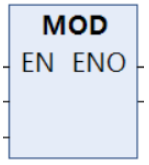
■ Precautions

Using 0 as the divisor or dividend may lead to different results according to the target system.

3.9. 6 MOD

This instruction divides two input values on the left side and outputs the quotient and integer remainder from the right side.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
MOD	Modulo-division	FC		a1:=a2 MOD a3;

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
In1	Dividend	-	Depends on data types	-	Input
In2	Divisor	-	Depends on data types	-	Input

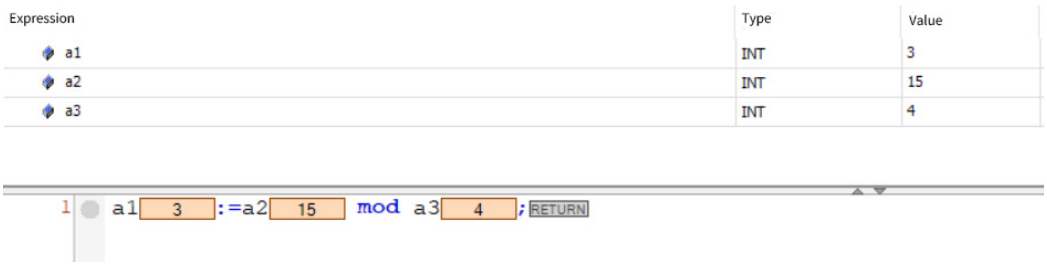
Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
OUT	Remainder	-	Depends on data types	-	Output

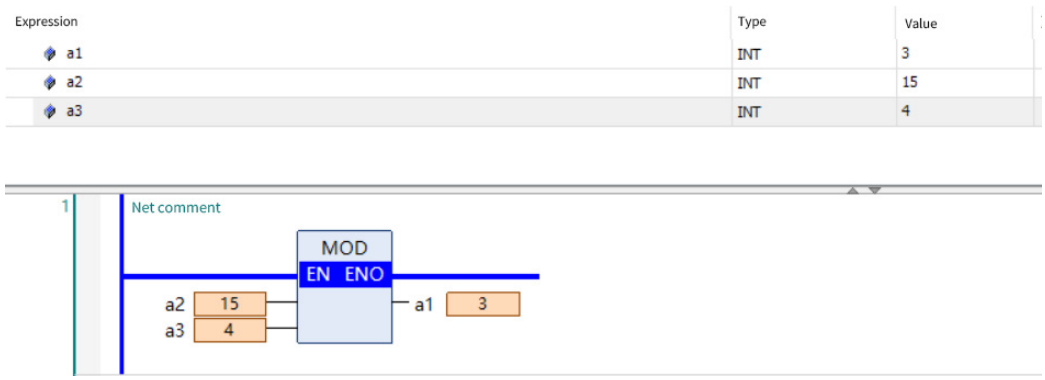
	Bool-ean	Bit String				Integer								Real number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In1	-	√	√	√	√	√	√	√	√	√	√	√	√	-	-	-	-	-	-	-
In2	-	√	√	√	√	√	√	√	√	√	√	√	√	-	-	-	-	-	-	-
Out	-	√	√	√	√	√	√	√	√	√	√	√	√	-	-	-	-	-	-	-

■ Program example

ST:



LD:



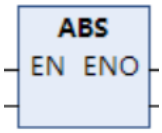
■ Precautions

Using 0 as the divisor or dividend may lead to different results according to the target system.

3.9. 7 ABS

This instruction gets the absolute value of the input data and then assigns the value to the output variable.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
ABS	Absolute value	FC		q:=ABS( );

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
In1	Input variables	-	Depends on data types	-	Variable to process

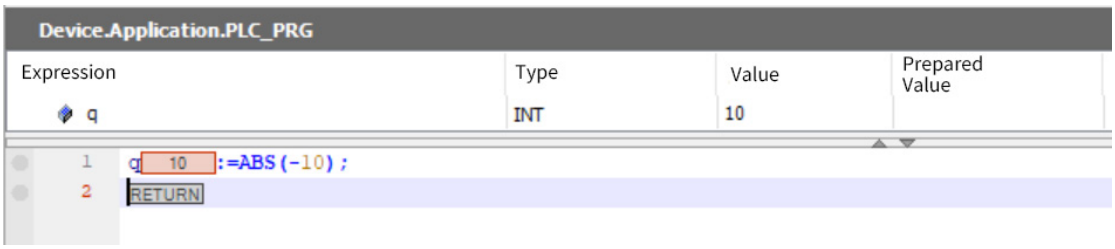
Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
OUT	Output value	-	Depends on data types	-	Output value

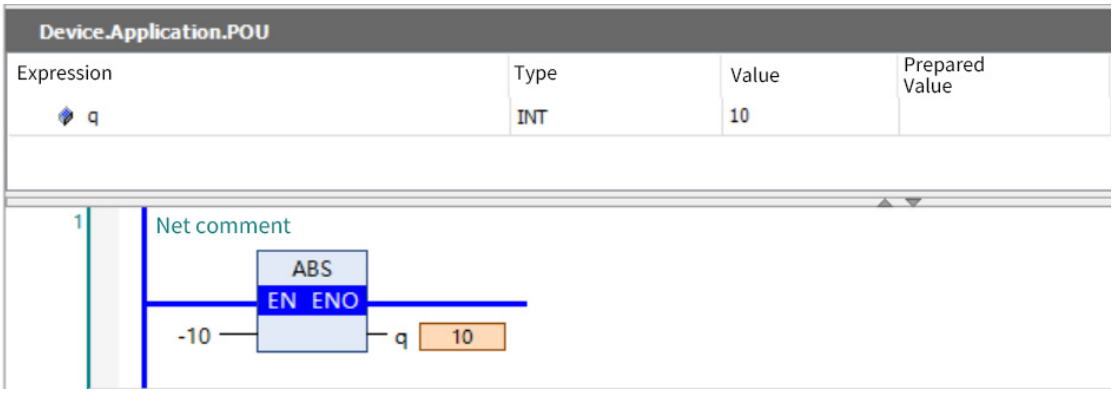
	Bool-ean	Bit String				Integer								Real number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In1	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	-
Out	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	-

■ Program example

ST:



LD:



3.9. 8 SQRT

This instruction gets the square root of an input value and then outputs the result.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
SQRT	Square root	FC		q:= SQRT ( );

■ Variables

Input Variable

Input variables	Name	Data Type	Value Range	Initial Value	Description
In1	Input Variable	-	Depends on data types	-	Variable to process

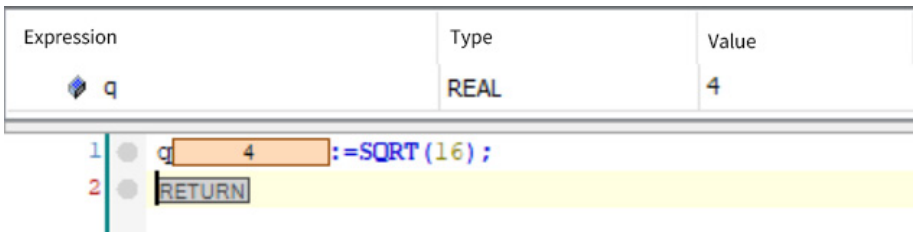
Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
OUT	Output value	-	Depends on data types	0	Output value

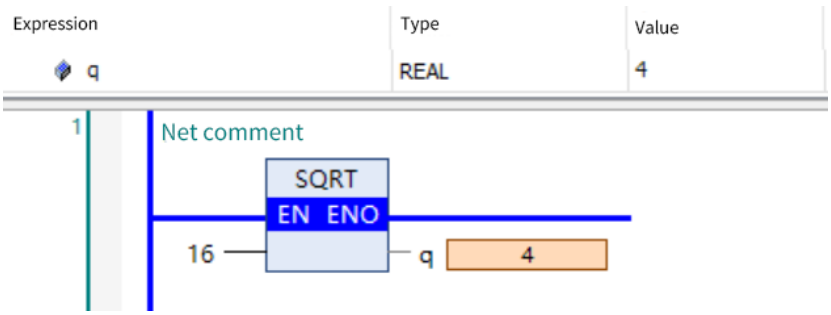
	Bool-ean	Bit String				Integer								Real number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In	-	√	√	√	√	√	√	√	√	√	√	√	√	√	-	-	-	-	-	-
Out	-	-	-	-	-	-	-	-	-	-	-	-	-	√	√	-	-	-	-	-

■ Program example

ST:



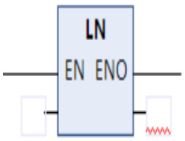
LD:



3.9. 9 LN

This instruction gets the logarithm of an input value and then outputs the result.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
LN	Natural logarithm	FC		q:= LN ( );

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
In1	Input variables	-	Depends on data types	-	Variable to process

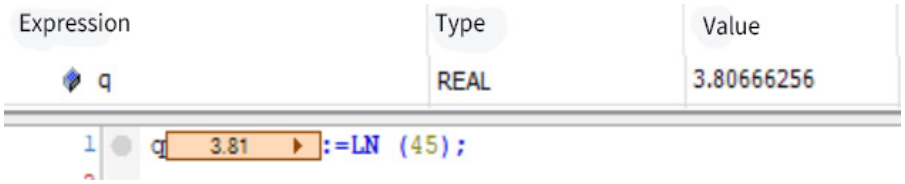
Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
OUT	Output value	-	Depends on data types	0	Output value

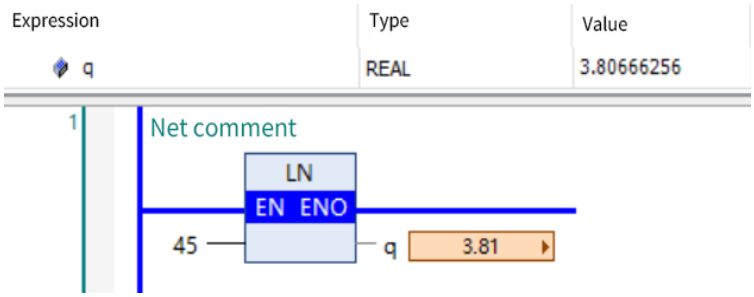
	Bool-ean	Bit String				Integer								Real number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In	-	√	√	√	√	√	√	√	√	√	√	√	√	√	-	-	-	-	-	-
Out	-	-	-	-	-	-	-	-	-	-	-	-	-	√	√	-	-	-	-	-

■ Program example

ST:



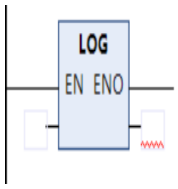
LD:



3.9. 10 LOG

This instruction gets the base-10 logarithm of an input value and then outputs the result.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
LOG	Logarithm base 10	FC		q:= LOG ( );

■ Variables

Input Variable

Input variables	Name	Data Type	Value Range	Initial Value	Description
In1	Input Variable	-	Depends on data types	-	Variable to process

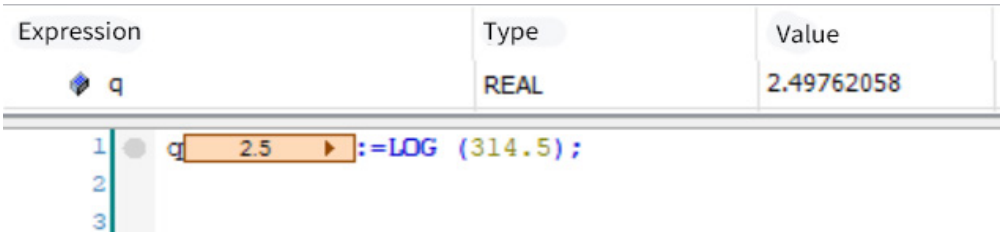
Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
OUT	Output value	-	Depends on data types	0	Output value

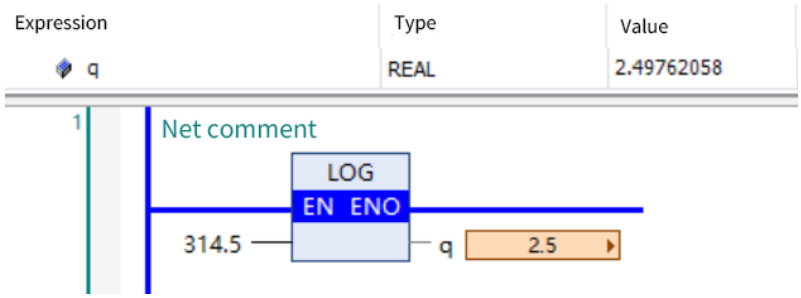
	Bool-ean	Bit String				Integer								Real number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In	-	√	√	√	√	√	√	√	√	√	√	√	√	√	-	-	-	-	-	-
Out	-	-	-	-	-	-	-	-	-	-	-	-	-	√	√	-	-	-	-	-

■ Program example

ST:



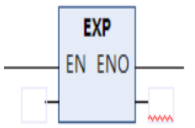
LD:



3.9. 11 EXP

This instruction performs exponential function calculation for an input value and then outputs the result.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
EXP	Natural exponential operation	FC		q:= EXP ( );

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
In1	Input variables	-	Depends on data types	-	Variable to process

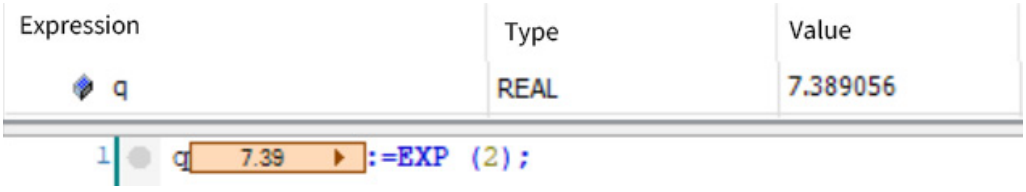
Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
OUT	Output value	-	Depends on data types	0	Output value

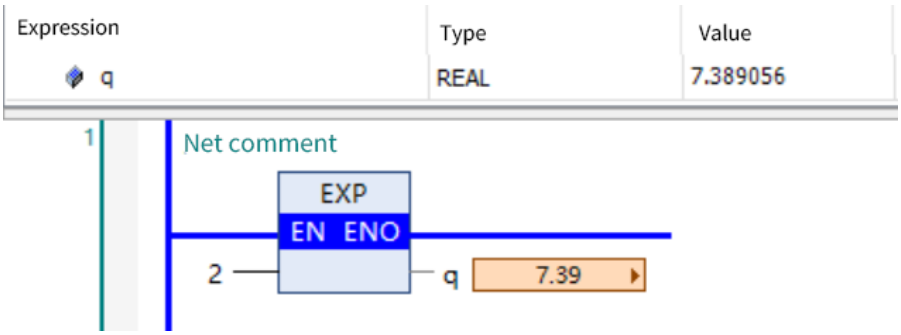
	Bool-ean	Bit String				Integer								Real number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In	-	√	√	√	√	√	√	√	√	√	√	√	√	√	-	-	-	-	-	-
Out	-	-	-	-	-	-	-	-	-	-	-	-	-	√	√	-	-	-	-	-

■ Program example

ST:



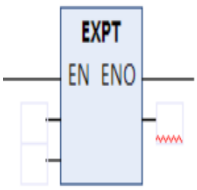
LD:



3.9. 12 EXPT

This instruction uses input variable 2 as the power of input variable 1 and then outputs the power operation result.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
EXPT	Exponentiation	FC		q:=EXPT(x1,x2);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
In1	Input base number	-	Depends on data types	-	Input base number
In2	Input exponent	-	Depends on data types	-	Input exponent

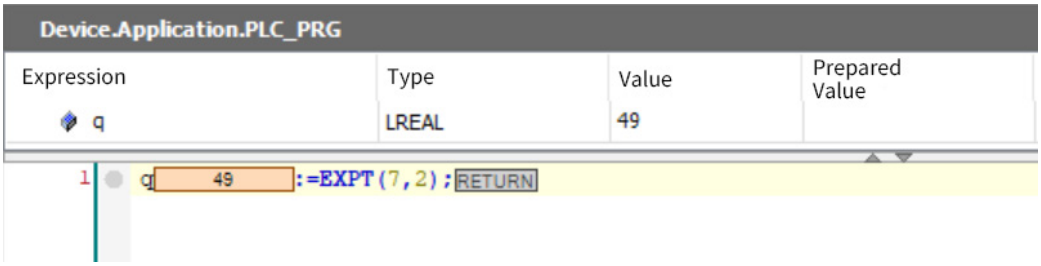
Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
OUT	Output value	-	Depends on data types	0	Output value

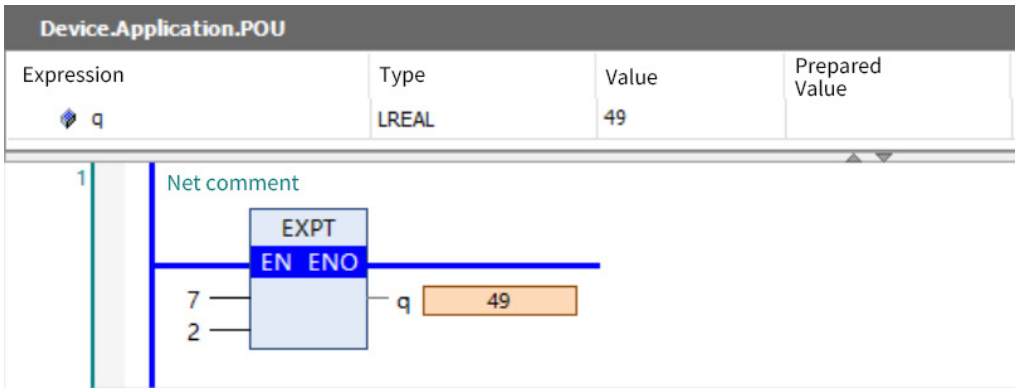
	Boolean	Bit String				Integer								Real number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In1	-	√	√	√	√	√	√	√	√	√	√	√	√	√	-	-	-	-	-	-
In2	-	√	√	√	√	√	√	√	√	√	√	√	√	√	-	-	-	-	-	-
Out	-	-	-	-	-	-	-	-	-	-	-	-	-	√	√	-	-	-	-	-

■ Program example

ST:



LD:



■ Precautions

The function result is not defined in the following cases:

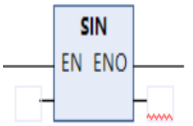
1. The base value is negative.

2. The base value is 0 and the exponent is less than or equal to 0.

3.9. 13 SIN

This instruction performs sine operation for an input value and then outputs the result.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
SIN	Sine	FC		q:=SIN( );

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
In1	Input variables	-	Depends on data types	-	Variable to process

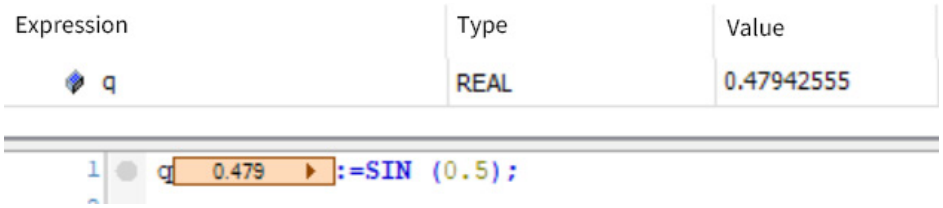
Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
OUT	Output value	-	Depends on data types	0	Output value

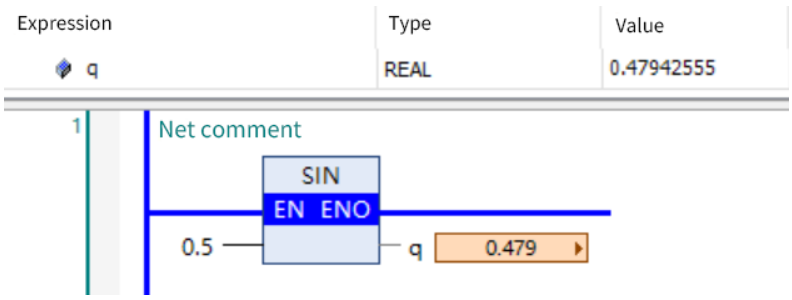
	Bool-ean	Bit String				Integer								Real number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In	-	√	√	√	√	√	√	√	√	√	√	√	√	√	-	-	-	-	-	-
Out	-	-	-	-	-	-	-	-	-	-	-	-	-	√	√	-	-	-	-	-

■ Program example

ST:



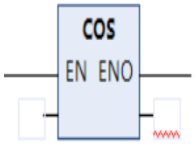
LD:



3.9. 14 COS

This instruction performs cosine operation for an input value and then outputs the result.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
COS	Cosine	FC		q:=COS( );

■ Variables

Input Variable

Input variables	Name	Data Type	Value Range	Initial Value	Description
In1	Input Variable	-	Depends on data types	-	Variable to process

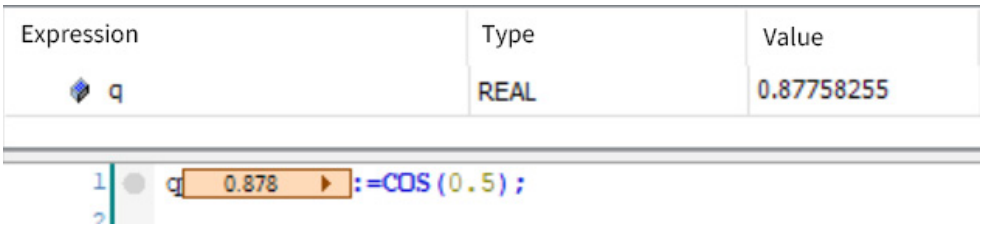
Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
OUT	Output value	-	Depends on data types	0	Output value

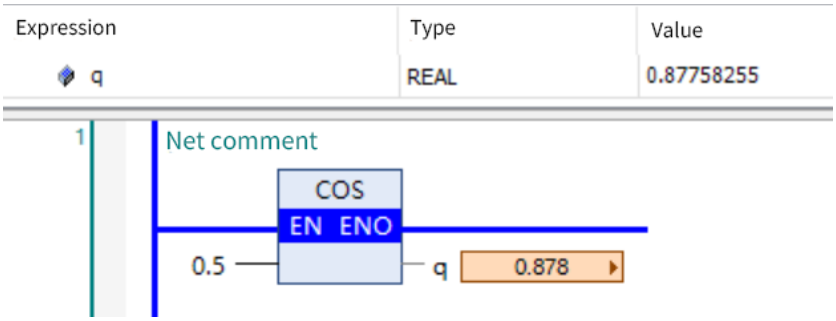
	Bool-ean	Bit String				Integer								Real number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In	-	√	√	√	√	√	√	√	√	√	√	√	√	√	-	-	-	-	-	-
Out	-	-	-	-	-	-	-	-	-	-	-	-	-	√	√	-	-	-	-	-

■ Program example

ST:



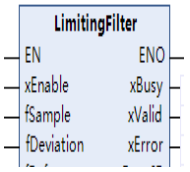
LD:



3.9. 15 TAN

This instruction performs tangent operation for an input value and then outputs the result.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
TAN	Tangent	FC		q:=TAN( );

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
In1	Input variables	-	Depends on data types	-	Variable to process

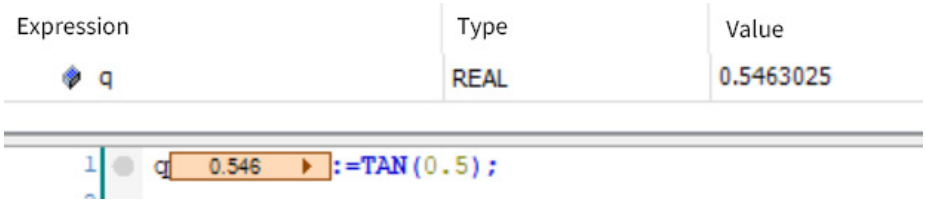
Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
OUT	Output value	-	Depends on data types	0	Output value

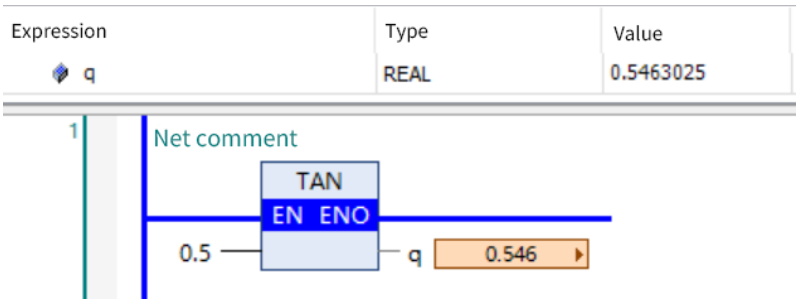
	Bool-ean	Bit String				Integer								Real number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In	-	√	√	√	√	√	√	√	√	√	√	√	√	√	-	-	-	-	-	-
Out	-	-	-	-	-	-	-	-	-	-	-	-	-	√	√	-	-	-	-	-

■ Program example

ST:



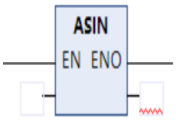
LD:



3.9. 16 ASIN

This instruction performs arcsine operation for an input value and then outputs the result.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
ASIN	Arcsine	FC		q:=ASIN( );

■ Variables

Input Variable

Input variables	Name	Data Type	Value Range	Initial Value	Description
In1	Input Variable	-	Depends on data types	-	Variable to process

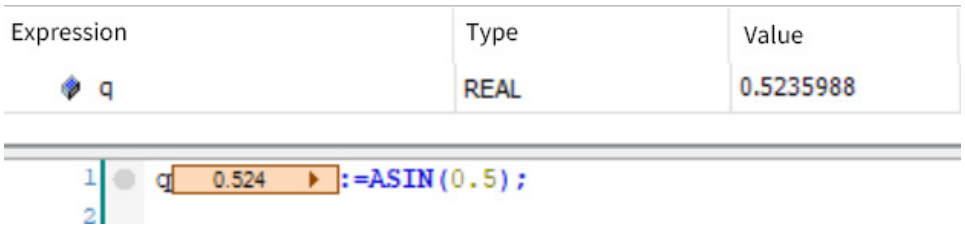
Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
OUT	Output value	-	Depends on data types	0	Output value

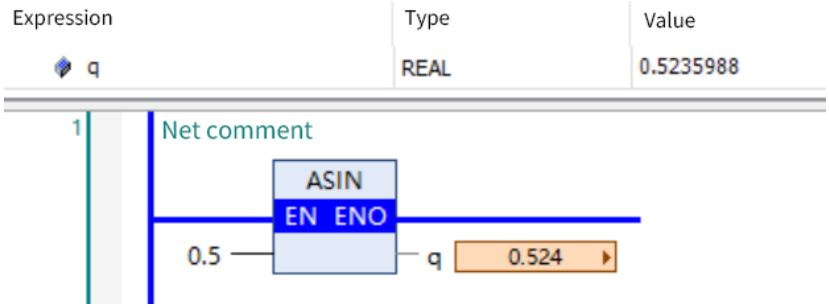
	Bool-ean	Bit String				Integer								Real number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In	-	√	√	√	√	√	√	√	√	√	√	√	√	√	-	-	-	-	-	-
Out	-	-	-	-	-	-	-	-	-	-	-	-	-	√	√	-	-	-	-	-

■ Program example

ST:



LD:



3.9. 17 ACOS

This instruction performs arc cosine operation for an input value and then outputs the result as a radian.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
ACOS	Arc cosine	FC		q:=ACOS( );

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
In1	Input variables	-	Depends on data types	-	Variable to process

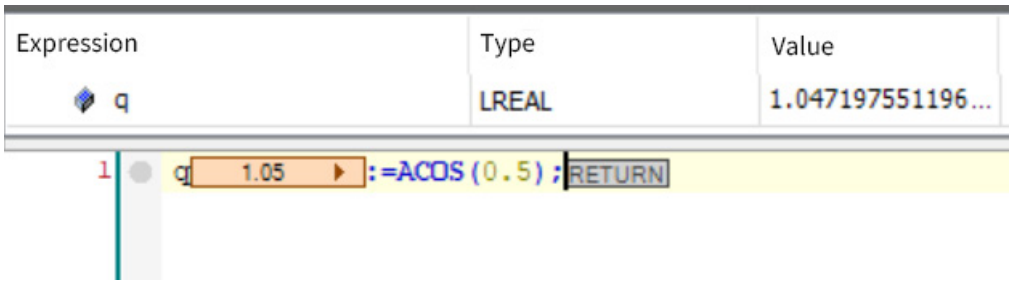
Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
OUT	Output value	-	Depends on data types	0	Output value

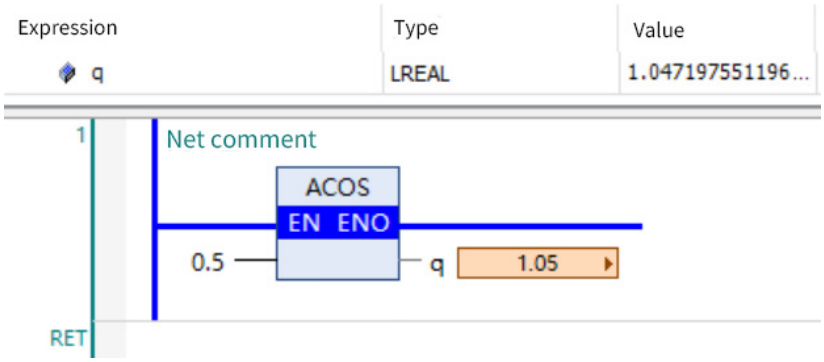
	Bool-ean	Bit String				Integer								Real number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In	-	√	√	√	√	√	√	√	√	√	√	√	√	√	-	-	-	-	-	-
Out	-	-	-	-	-	-	-	-	-	-	-	-	-	√	√	-	-	-	-	-

■ Program example

ST:



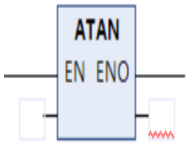
LD:



3.9. 18 ATAN

This instruction performs arc tangent operation for an input value and then outputs the result as a radian.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
ATAN	Arc tangent	FC		q:=ATAN( );

■ Variables

Input Variable

Input variables	Name	Data Type	Value Range	Initial Value	Description
In1	Input Variable	-	Depends on data types	-	Variable to process

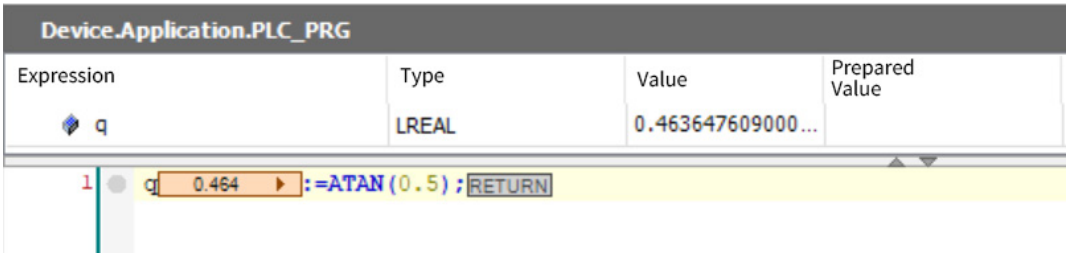
Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
OUT	Output value	-	Depends on data types	0	Output value

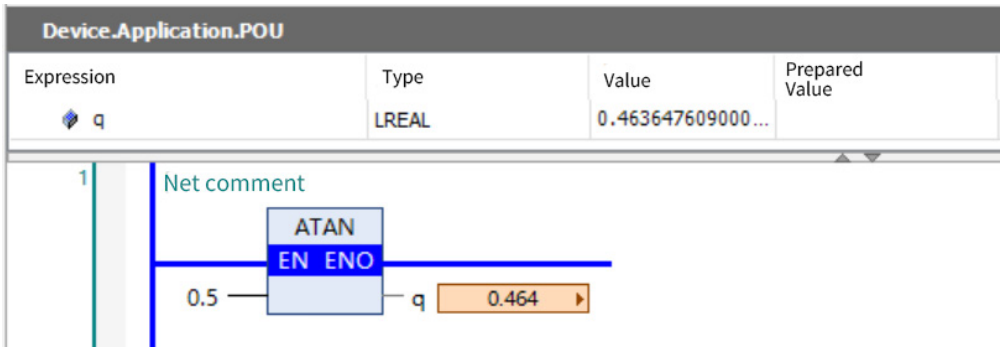
	Bool-ean	Bit String				Integer								Real number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In	-	√	√	√	√	√	√	√	√	√	√	√	√	√	-	-	-	-	-	-
Out	-	-	-	-	-	-	-	-	-	-	-	-	-	√	√	-	-	-	-	-

■ Program example

ST:



LD:

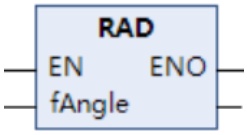
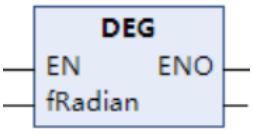


3.9. 19 RAD and DEG

RAD: Converts a binary floating-point value in degrees into a value in radians. The formula is as follows: Value in radians = Value in degrees x л/180.

DEG: Converts a binary floating-point value in radians into a value in degrees. The formula is as follows: Value in degrees = Value in radians x л/180.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
RAD	Degrees to radians	FC		q:=RAD( );
DEG	Radians to degrees	FC		q:=DEG( );

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description

fAngle	Input degree	LREAL	Depends on data types	0	Input degree
	Output radian	LREAL	Depends on data types	0	Output radian
fRadian	Input radian	LREAL	Depends on data types	0	Input radian
	Output degree	LREAL	Depends on data types	0	Output degree

Output variables

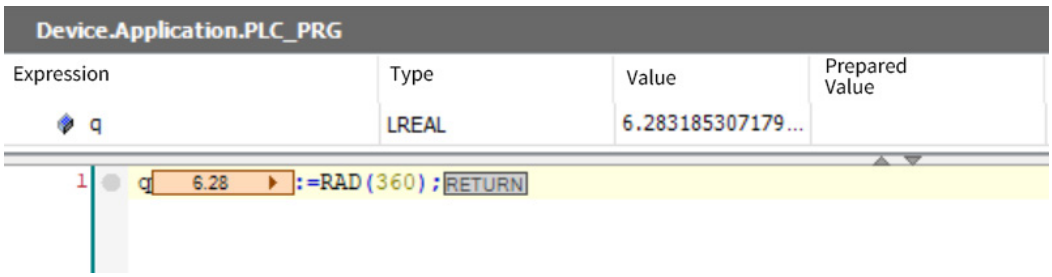
Output Variable	Name	Data Type	Value Range	Initial Value	Description
OUT	Output value	-	Depends on data types	0	Output value

	Bool-ean	Bit String				Integer								Real number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
fAngle	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-
fRadian	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-

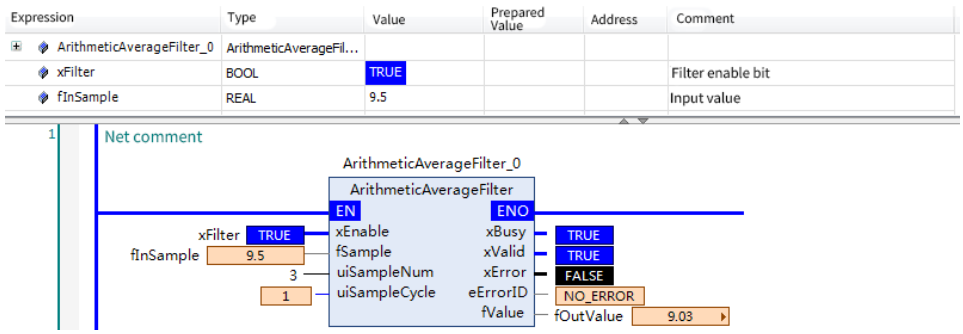
■ Program example

1. RAD

ST:

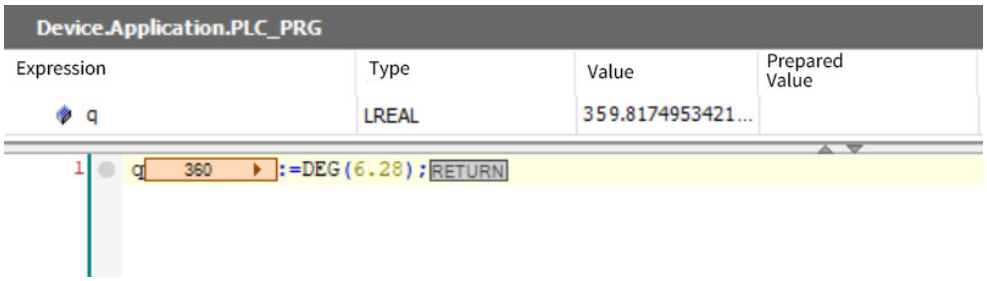


LD:

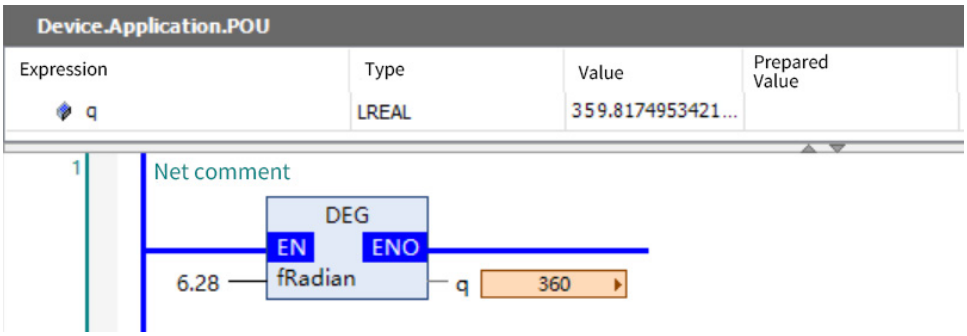


2. DEG

ST:



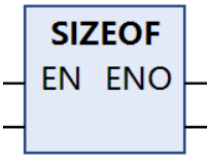
LD:



3.9. 20 SIZEOF

■ Instruction description

This instruction defines the number of bytes required by a variable. The SIZEOF operator symbol always generates an unsigned value. The type of the return variable adapts to the variable length check.

Instruction	Name	FB/FC	LD Expression	ST Expression
SIZEOF	Data size	FC		SIZEOF( );

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
IN	Target data	<TYPE>	-	-	Target data

Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
OUT	Data length	UINT	-	-	Data length

	Bool-ean	Bit String				Integer								Real number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
IN	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√
OUT	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

<TYPE> in the table indicates an unspecified data type.

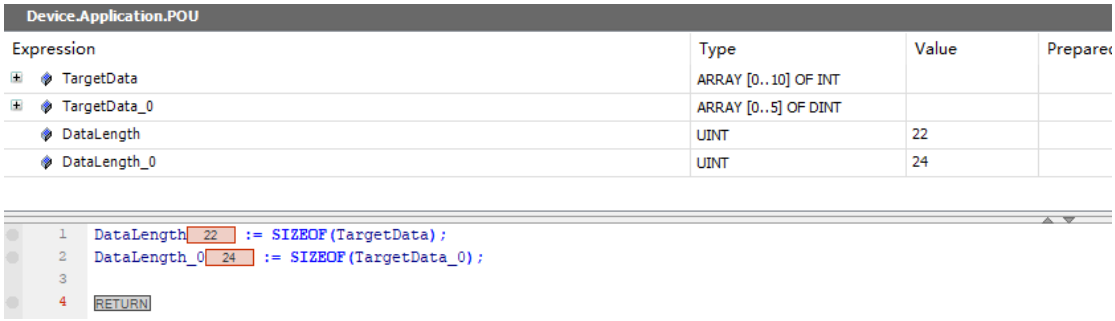
■ Program example

1. SIZEOF

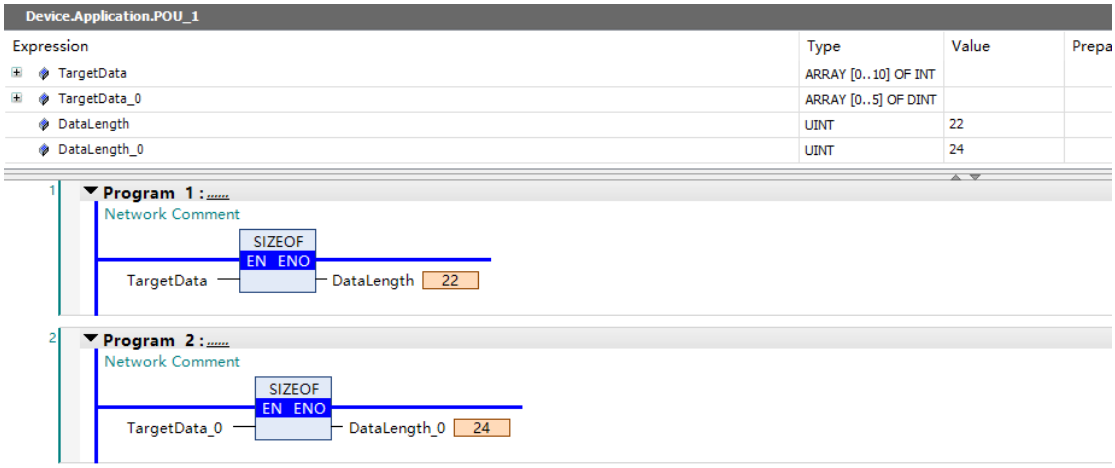
Example 1 defines an INT array with 11 elements. Then the size is 22.

Example 2 defines a DINT array with 6 elements. Then the size is 24.

ST



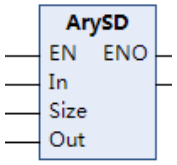
LD



3.9. 21 ArySD

This instruction calculates standard deviation of array elements.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
ArySD	Array element standard deviation	FC		ArySD(In :=, Size :=, Out :=);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
In[] (array)	Array to process	-	Depends on data types	-	Array to process

Size	Number of elements	UINT	Depends on data types	0	Number of elements in In[] to convert

Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
Out	Standard deviation	-	Depends on data types	-	Standard deviation

	Bool-ean	Bit String				Integer								Real number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In[]	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-
Size	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-
Out	-	-	-	-	-	-	-	-	-	-	-	-	-	√	√	-	-	-	-	-

■ Function

This instruction calculates the standard deviation of Size elements of the array to process In[] starting from In[0].

i: Subscript of In[], 0 to Size − 1

InM: Average value of In[0] to In[Size - 1]

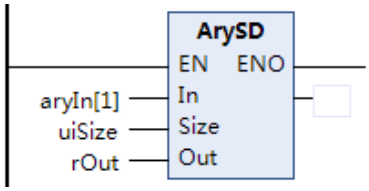
■ Program example

The following example is for when Size is UINT#5.

	LD	ST
Defined variable		

```
VAR
    aryIn : ARRAY[1..5] OF REAL := [123.4,234.5,345.6,456.7,567.8];
    uiSize : UINT := 5;
    rOut : REAL;
END_VAR
```

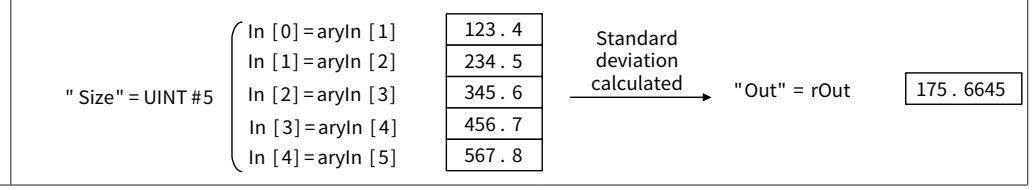
Program




```
ArySD(
    In := aryIn[1],
    Size := uiSize,
    Out := rOut
);
```

Running result

aryIn	ARRAY [1..5] OF REAL	
aryIn[1]	REAL	123.4
aryIn[2]	REAL	234.5
aryIn[3]	REAL	345.6
aryIn[4]	REAL	456.7
aryIn[5]	REAL	567.8
uiSize	UINT	5
rOut	REAL	175.66452

	LD	ST
Work principle		

■ Precautions

· When the value of Size is 0 or 1, the value of Out is 0.

· When an intermediate value in the calculation process exceeds the valid range of In[], the value of Out is an illegal value. No error occurs in this case.

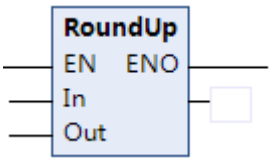
· When the value of Size exceeds the In[] array, the return value is FALSE and Out does not change.

3.9. 22 RoundUp

This instruction changes a real number to a positive number.

RoundUp: Rounds up the number at the first decimal digit

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
RoundUp	Round up real number (Rounds up a real number at the first decimal digit to make an integer)	FC		RoundUp (In :=, Out :=);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
In	Data to convert	-	Depends on data types	-	Data to convert
Out	Conversion result	-	Depends on data types	-	Conversion result

	Bool-ean	Bit String				Integer								Real number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In	-	-	-	-	-	-	-	-	-	-	-	-	-	√	√	-	-	-	-	-
Out	-	-	-	-	-	-	-	-	-	-	-	√	√	-	-	-	-	-	-	-

■ Function

This instruction changes the real number in In to an integer by rounding up the first decimal digit.

Input value	Output value
REAL#1.6	DINT#2
REAL#1.5	DINT#2

Input value	Output value
REAL#1.4	DINT#2
REAL#2.5	DINT#3
REAL#-1.6	DINT#-2
REAL#-1.5	DINT#-2
REAL#-1.4	DINT#-2
REAL#-2.5	DINT#-3

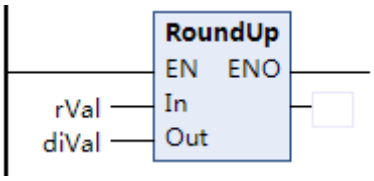
■ Program example

The following example for the RoundUp instruction is for when In is REAL#-3.55. The value of Out is DINT#-4.

<table>
<tr><th></th><th>LD</th><th>ST</th></tr>
<tr><td>Defined variable</td><td colspan="2">

```
VAR
    rVal : REAL := -3.55;
    diVal : DINT;
END_VAR
```

</td></tr>
<tr><td>Program</td><td>



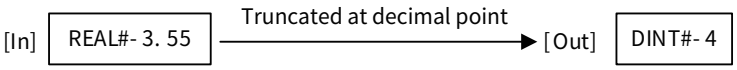
</td><td>

```
RoundUp(In := rVal,Out := diVal);
```

</td></tr>
<tr><td>Running result</td><td colspan="2">

rVal	REAL	-3.55
diVal	DINT	-4

</td></tr>
<tr><td>Work principle</td><td colspan="2">



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

■ Precautions

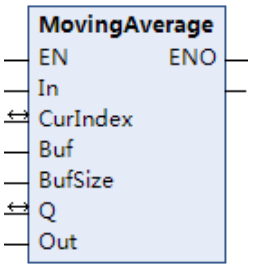
When the conversion result exceeds the valid range of Out, the value of Out is incorrect.

Make sure that the data types of In and Out are the same. Otherwise, a program error occurs and the return value is FALSE.

3.9. 23 MovingAverage

This instruction calculates a moving average.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
MovingAverage	Moving average	FC		MovingAverage(In :=, CurIndex :=, Buf :=, BufSize :=, Q := Out :=,);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
In	Input value	-	Depends on data types	-	Number to include in average
BufSize	Maximum number stored	UINT	Depends on data types	1	Maximum number of elements to include in average
Out	Operation result	-	Depends on data types	-	Operation result

In-out variables

In-Out Variable	Name	Data Type	Value Range	Initial Value	Description
CurInDex	Input value storage position	UINT	Depends on data types	0	Position in Buf[] in which to store In
Buf[] (array)	Input value storage array	UINT	Depends on data types	0	Array to store In values
Q	Calculation completed flag	BOOL	[FALSE, TRUE]	FALSE	TRUE: BufSize elements or more have been stored in Buf[] FALSE: BufSize elements are not yet stored in Buf[]

Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
Out	Operation result	-	Depends on data types	-	Operation result

	Bool-ean	Bit String				Integer								Real number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In	-	-	-	-	-	√	√	√	√	√	√	√	√	√	√	-	-	-	-	-
BufSize	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-
CurInDex	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-
Buf[]	Array with the same data type as elements of In[]																			
Q	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Out	-	-	-	-	-	√	√	√	√	√	√	√	√	√	√	-	-	-	-	-

■ Function

This instruction stores the value of In in the input value storage array Buf[] each time the instruction is executed. It stores the average of the stored values in the calculation result Out. The maximum number of elements to include in the average is specified by BufSize.

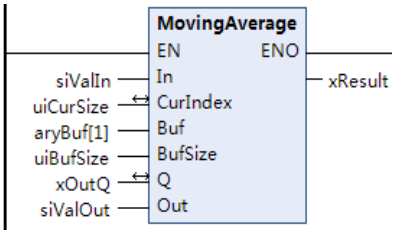
■ Program example

The processing procedure when BufSize is UINT#3 is described below as an example. The instruction and statement are written as follows:

<table>
<tr><th></th><th>LD</th><th>ST</th></tr>
<tr><td>Defined variable</td><td colspan="2">

```
VAR
    siValIn     : SINT;
    uiCurSize   : UINT;
    aryBuf      : ARRAY [1..4] OF SINT;
    uiBufSize   : UINT := 3;
    xOutQ       : BOOL;
    siValOut    : SINT;
    xResult     : BOOL;
END_VAR
```

</td></tr>
<tr><td>Program</td><td>



</td><td>

```
xResult := MovingAverage(In:= siValIn
    CurIndex:= uiCurSize,
    Buf:= aryBuf[1],
    BufSize:= uiBufSize,
    Q:= xOutQ,
    Out:= siValOut);
```

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

First input value

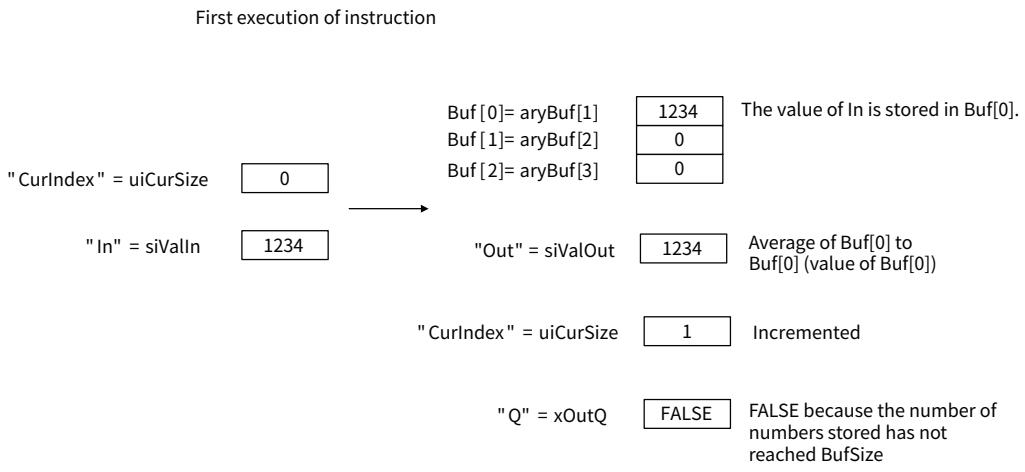
The input value storage position CurIndex is set to 0 and then the instruction is executed.

Buf[0] to Buf[BufSize − 1] of the input value storage array Buf[] are cleared to zeros and then the first input value In is stored in Buf[0].

The value of calculation completed flag Q changes to FALSE. This indicates that the number of values that are stored in Buf[] has not reached BufSize yet.

While the value of Q is FALSE, the average value is calculated for the CurIndex + 1 numbers that start from Buf[0]. The calculation result is stored in Out.

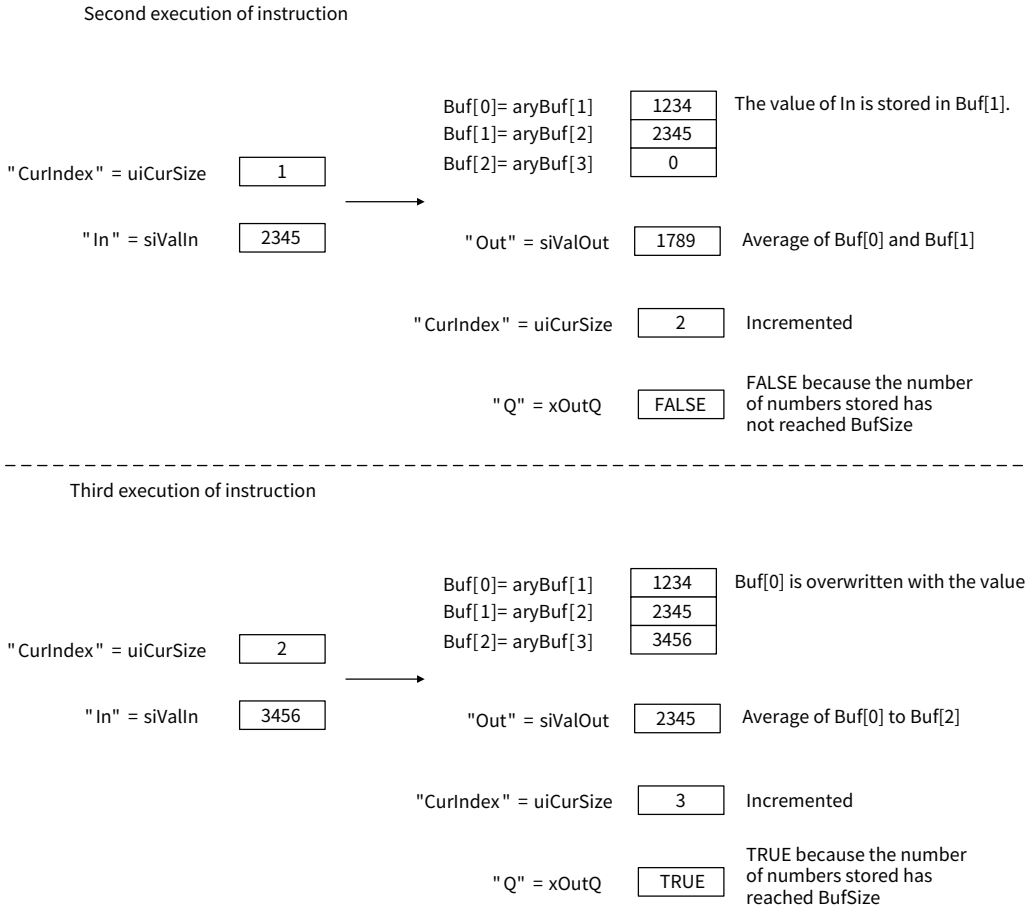
Finally, the value of CurIndex is incremented.



Value input up to BufSize

Each time the instruction is executed, the value of In is stored in Buf[CurIndex]. The average of CurIndex + 1 numbers that start from Buf[0] is calculated and stored in Out.

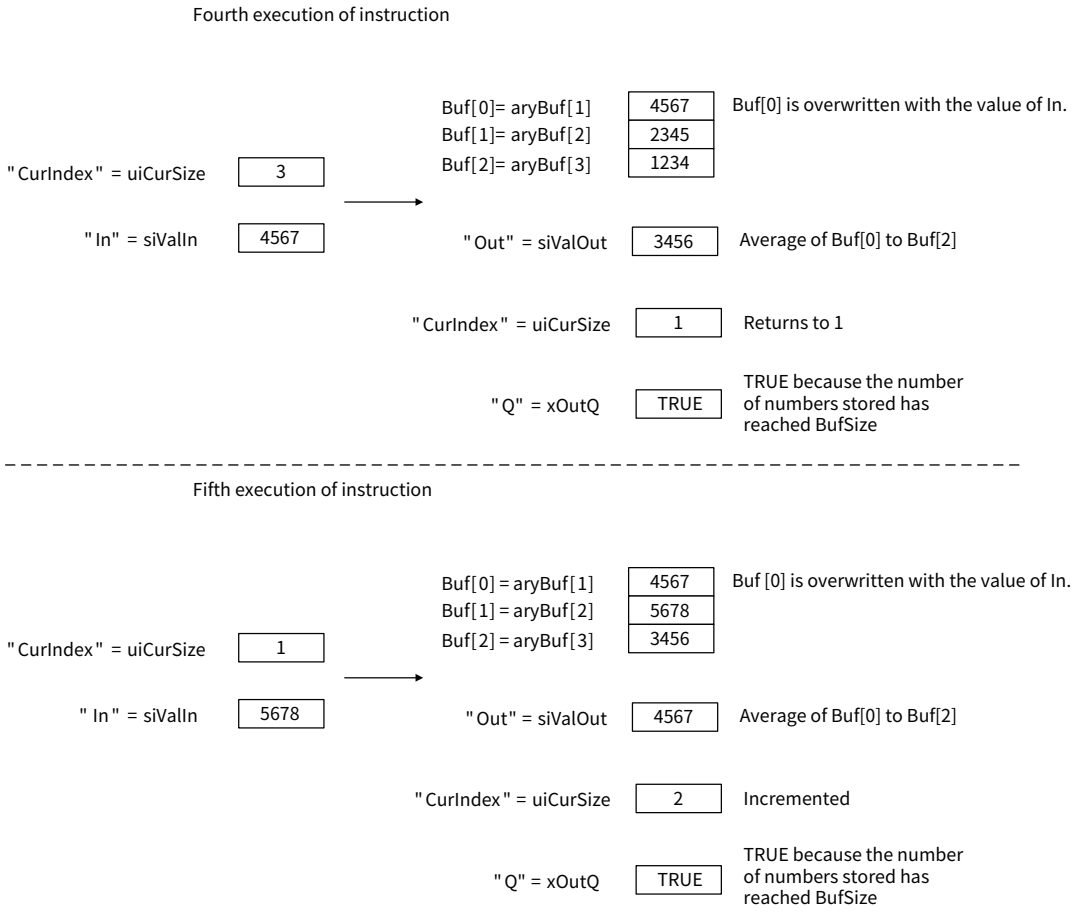
When the number of instruction executions reaches BufSize, the value of Q changes to TRUE.



Value input after BufSize times

Each time the instruction is executed, Buf[0] to Buf[BufSize − 1] are overwritten with the value of In in cyclic fashion. The average of Buf[0] to Buf[BufSize − 1] is calculated and stored in Out.

The value of CurIndex returns to 1 after it reaches BufSize and it is then incremented again. The value of Q remains TRUE.



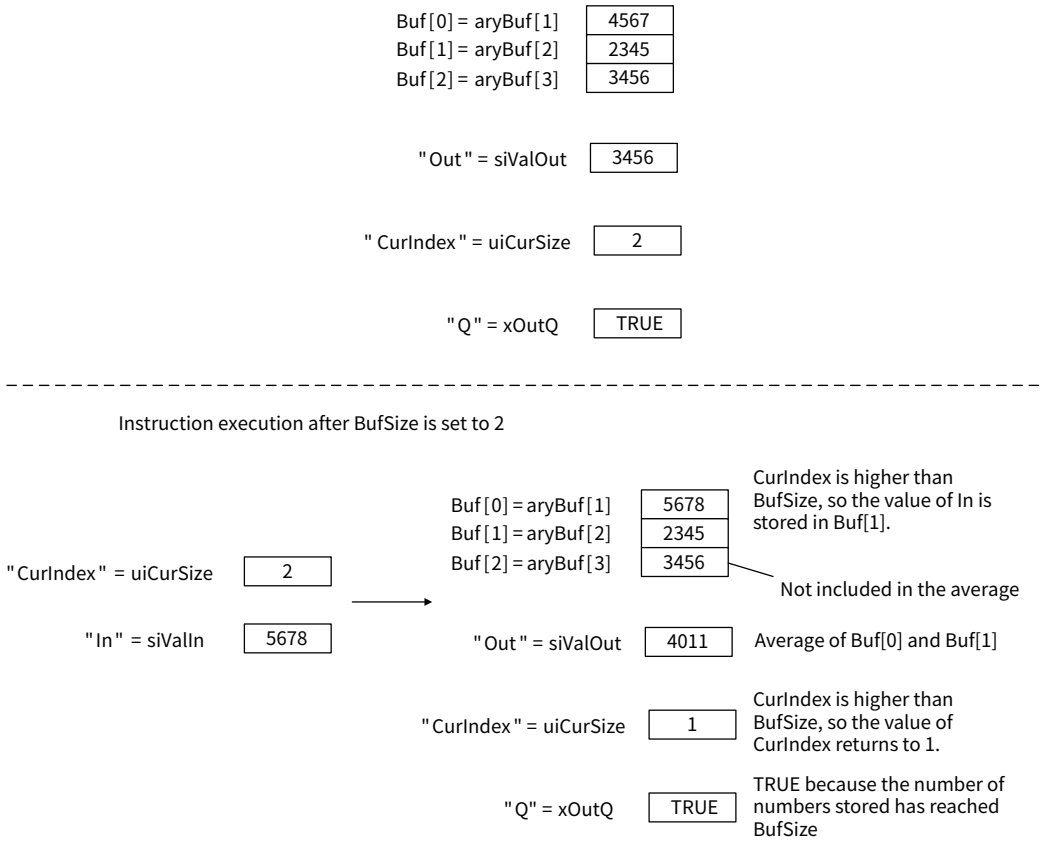
Initialization of stored values

If the value of CurSize is set to 0 before the instruction is executed, values of Buf[0] to Buf[BufSize − 1] change to 0 and the current value of In is stored again in Buf[0].

The value of CurIndex changes to 1 and the value of Q changes to FALSE.

Change of the BufSize value

If the value of BufSize changes to 0 before the instruction is executed, the operation is performed with the new value of BufSize and the current value of CurIndex.



■ Precautions

· Use the same data type for In, Out, and elements of Buf[]. If they are different, an error occurs during compilation.

· Even if the operation result exceeds the valid range of Out, no error occurs but the value of Out is an illegal value.

· When the value of BufSize is 0, the values of Out and CurIndex change to 0, and the value of Q changes to TRUE.

· After BufSize changes, the current value of CurIndex remains.

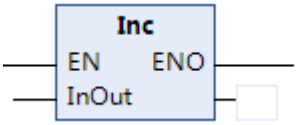
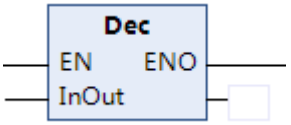
· When the value of BufSize exceeds the Buf[] array, the return value is FALSE and Out does not change.

3.9. 24 Inc and Dec

Inc: Increments an integer value.

Dec: Decrements an integer value.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
Inc	Increment	FC		Inc (InOut :=);
Dec	Decrement	FC		Dec (InOut :=);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
InOut	Target data	-	Depends on data types	-	Target data

	Bool-ean	Bit String				Integer								Real number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
InOut	-	-	-	-	-	√	√	√	√	√	√	√	√	-	-	-	-	-	-	-

■ Function

Inc

This instruction increments target data InOut. When the result exceeds the maximum value of InOut, InOut returns to the minimum value.

Dec

This instruction decrements target data InOut. When the result exceeds the minimum value of InOut, InOut returns to the maximum value.

■ Program example

The following example for the Inc instruction is for when the input value is moved to InOut. When the input value is INT#4, the value of InOut is INT#5 after the instruction is executed.

	LD	ST
Defined variable	VAR iVal : INT := 4; END_VAR	
Program	bStart, iVal, Inc (EN, ENO, InOut)	
Running result	bStart BOOL FALSE iVal INT 5	

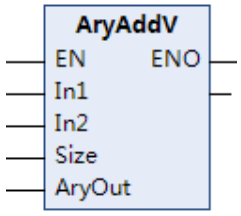
■ Precautions

When the instruction is used in ST, return value Out is not used.

3.9. 25 AryAddV

This instruction adds the same value to specified elements of an array.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
AryAddV	Array value addition	FC		AryAddV(In1 :=, In2 :=, Size := , AryOut :=);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
In1[] (array)	Addition array	-	Depends on data types	-	Addition array
In2	Value to add	-	Depends on data types	-	Value to add
Size	Number of elements	UINT	Depends on data types	1	Number of elements of In1[] for addition

Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
AryOut[] (array)	Addition result array	-	Depends on data types	-	Addition result array

	Bool-ean	Bit String				Integer								Real number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In1[]	-	-	-	-	-	√	√	√	√	√	√	√	√	√	√	-	-	-	-	-
In2	Array with the same data type as the elements of In1[]																			
Size	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-
AryOut[]	Array with the same data type as the elements of In1[]																			

■ Function

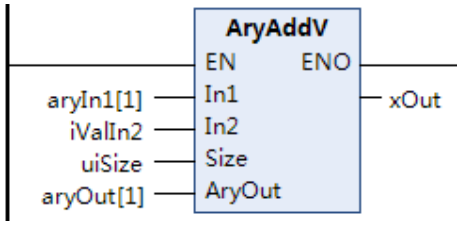
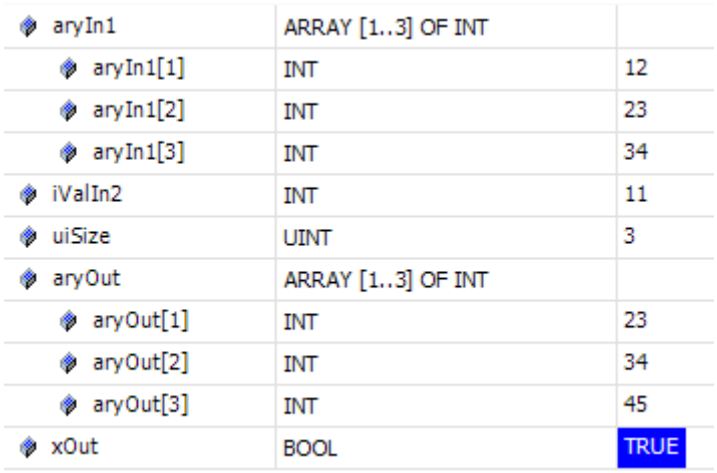
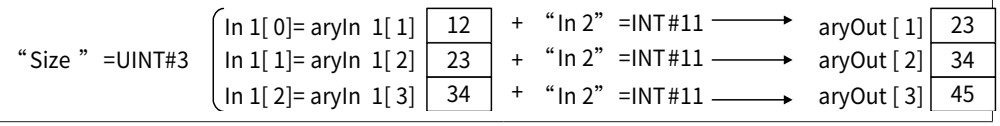
This instruction adds the value to add In2 to Size elements of the addition array In1[] starting from In1[0], and outputs the results to addition results array AryOut[].

■ Program example

The following example is for when In2 is INT#11 and Size is UINT#3.

	LD	ST

Defined variable	VAR
	aryIn1 : ARRAY[1..3] OF INT := [12,23,34];
	iValIn2 : INT := 11;
	uiSize : UINT := 3;
	aryOut : ARRAY[1..3] OF INT;
	xOut : BOOL;
	END_VAR	

	LD	ST
Program		AryAddV(In1:= aryIn1[1], In2:= iValIn2, Size:= uiSize, AryOut:= aryOut[1]);
Running result		
Work principle		

■ Precautions

· When the value of Size exceeds the In1[] or AryOut[] array, the return value is FALSE and AryOut[] does not change.

· The data types of In[], In2, and AryOut must be the same. If they are different, an error occurs during compilation.

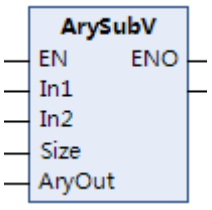
· If the addition results exceed the valid range of AryOut[], the elements of AryOut[] contain illegal values. No error occurs in this case. Corruption does not occur in the data in the memory area adjacent to those elements.

· When the value of Size is 0, AryOut[] does not change and the return value is TRUE.

3.9. 26 ArySubV

This instruction subtracts the same value from specified elements of an array.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
ArySubV	Array value subtraction	FC		ArySubV(In1 :=, In2 :=, Size := , AryOut :=);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
In1[] (array)	Minuend array	-	Depends on data types	-	Minuend array
In2	Subtrahend	-	Depends on data types	-	Subtrahend
Size	Number of elements	UINT	Depends on data types	1	Number of elements of In1[] for addition

In-out variables

In-Out Variable	Name	Data Type	Value Range	Initial Value	Description
AryOut[] (array)	Subtraction result array	-	Depends on data types	-	Subtraction result array

	Bool-ean	Bit String				Integer								Real number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In1[]	-	-	-	-	-	√	√	√	√	√	√	√	√	-	-	-	-	-	-	-
In2	Array with the same data type as the elements of In1[]																			
Size	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-
AryOut[]	Array with the same data type as the elements of In1[]																			

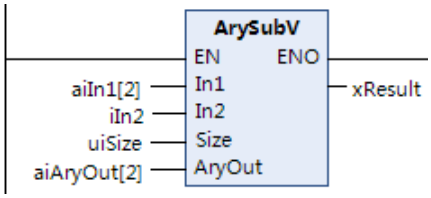
■ Function

This instruction subtracts the subtrahend In2 from Size elements of the minuend array In1[] starting from In1[0], and outputs the results to subtraction result array AryOut[].

■ Program example

	LD	ST
Defined variable		
Program		

```
VAR
    aiIn1       : ARRAY [1..5] OF INT := [-10,0,10,20,30];
    iIn2        : INT   := 10;
    uiSize      : UINT  := 3;
    aiAryOut    : ARRAY [1..5] OF INT;
    xResult     : BOOL;
END_VAR
```




```
xResult := ArySubV(In1 := aiIn1[2],
    In2 := iIn2,
    Size := uiSize,
    AryOut := aiAryOut[2]
    );
```

Running result

aiIn1	ARRAY [1..5] OF INT	
aiIn1[1]	INT	-10
aiIn1[2]	INT	0
aiIn1[3]	INT	10
aiIn1[4]	INT	20
aiIn1[5]	INT	30
iIn2	INT	10
uiSize	UINT	3
aiAryOut	ARRAY [1..5] OF INT	
aiAryOut[1]	INT	0
aiAryOut[2]	INT	-10
aiAryOut[3]	INT	0
aiAryOut[4]	INT	10
aiAryOut[5]	INT	0
xResult	BOOL	TRUE

■ Precautions

· When the value of Size exceeds the In1[] or AryOut[] array, the return value is FALSE and AryOut[] does not change.

· The data types of In[], In2, and AryOut must be the same. If they are different, an error occurs during compilation.

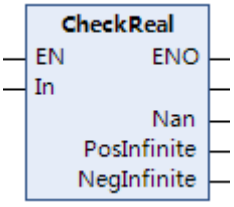
· If the subtraction results exceed the valid range of AryOut[], the elements of AryOut[] are not expected values. No error occurs in this case. Corruption does not occur in the data in the memory area adjacent to those elements.

· When the value of Size is 0, AryOut[] does not change and the return value is TRUE.

3.9. 27 CheckReal

This instruction checks a real number to see whether it is infinity or nonnumeric data.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
CheckReal	Real number check	FC		CheckReal (In :=, Nan =>, PosInfinite =>, NegInfinite =>,);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
In	Real number	-	Depends on data types	-	Real number

Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
Nan	Nonnumeric data check result	BOOL	[FALSE, TRUE]	FALSE	TRUE: Nonnumeric data FALSE: Not nonnumeric data
PosInfinite	Positive infinity check result	BOOL	[FALSE, TRUE]	FALSE	TRUE: Positive infinity FALSE: Not positive infinity
NegInfinite	Negative infinity check result	BOOL	[FALSE, TRUE]	FALSE	TRUE: Negative infinity FALSE: Not negative infinity

	Boolean	Bit String				Integer								Real number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In	-	-	-	-	-	-	-	-	-	-	-	-	-	√	√	-	-	-	-	-
Nan	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
PosInfinite	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
NegInfinite	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

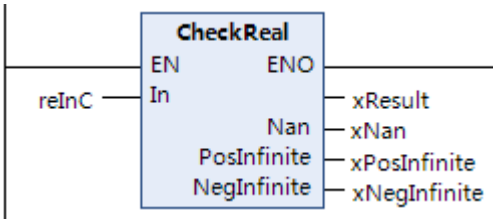
■ Function

This instruction checks the real number In to see if it is nonnumeric data, positive infinity, or negative infinity. It outputs the results to Nan, PosInfinite, and NegInfinite.

■ Program example

	LD	ST
Defined variable		
Program		

```
VAR
    reInA           : REAL := 3.4E+38;
    reInB           : REAL := -2;
    reInC           : REAL;
    xNan            : BOOL;
    xPosInfinite    : BOOL;
    xNegInfinite    : BOOL;
    xResult         : BOOL;
END_VAR
```




```
reInC := reInA * reInB;
xResult := CheckReal(In := reInC,
    Nan => xNan,
    PosInfinite => xPosInfinite,
    NegInfinite => xNegInfinite
    );
```

	LD		ST
Running result	reInA	REAL	3.4E+38
	reInB	REAL	-2
	reInC	REAL	-Infinity
	xNan	BOOL	FALSE
	xPosInfinite	BOOL	FALSE
	xNegInfinite	BOOL	TRUE
	xResult	BOOL	TRUE

■ Precautions

· If the parameter passed to In is not an integer or a real number, an error occurs during compilation.

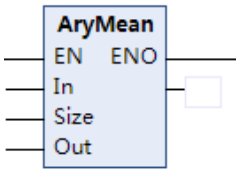
· If the parameter passed to In is an integer, the data type is converted as follows.

Data Type of Parameter Passed to In	Data Type of In
USINT, UINT, SINT, or INT	REAL
UDINT or DINT	LREAL
ULINT or LINT	An error occurs during compilation.

3.9. 28 AryMean

This instruction calculates the average of elements of an array.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
AryMean	Array mean	FC		AryMean(In:=, Size:=, Out:=);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
In[] (array)	Array to process	-	Depends on data types	-	Array to process
Size	Number of elements to process	-	Depends on data types	1	Number of In[] elements
Out	Operation result	-	Depends on data types	-	Operation result

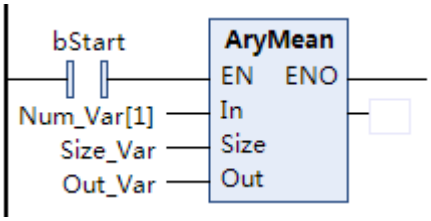
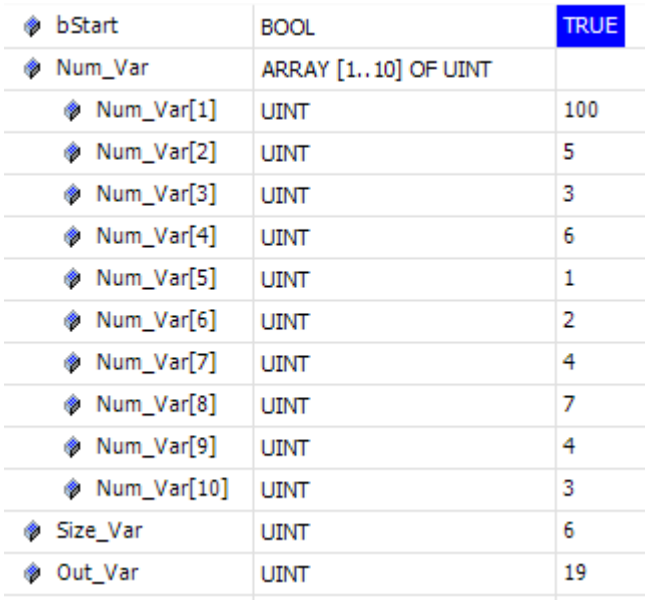
	Bool-ean	Bit String				Integer								Real number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In[]	-	-	-	-	-	√	√	√	√	√	√	√	√	√	√	√	-	-	-	-

	Bool-ean	Bit String				Integer								Real number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Size	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-
Out	Array with the same data type as the elements of In[]																			

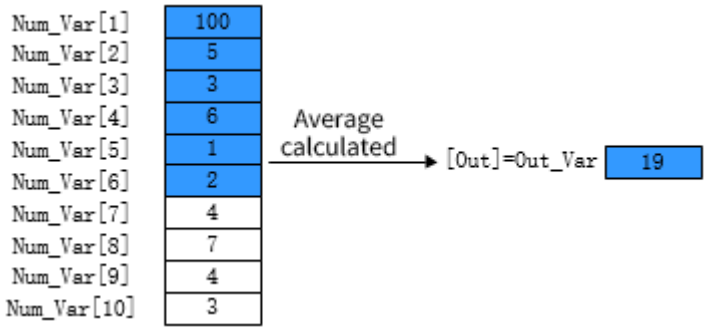
■ Function

This instruction calculates the average of elements of an array.

■ Program example

	LD	ST
Defined variable	PROGRAM POU VAR Num_Var:ARRAY[1..10] OF UINT:=[100,5,3,6,1,2,4,7,4,3]; Size_Var:UINT:=6; Out_Var:UINT; END_VAR	
Program		AryMean(In:=Num_Var[1], Size:=Size_Var, Out:=Out_Var1,);
Running result		

bStart	BOOL	TRUE
Num_Var	ARRAY [1..10] OF UINT	
Num_Var[1]	UINT	100
Num_Var[2]	UINT	5
Num_Var[3]	UINT	3
Num_Var[4]	UINT	6
Num_Var[5]	UINT	1
Num_Var[6]	UINT	2
Num_Var[7]	UINT	4
Num_Var[8]	UINT	7
Num_Var[9]	UINT	4
Num_Var[10]	UINT	3
Size_Var	UINT	6
Out_Var	UINT	19

	LD	ST
Work principle		

■ Precautions

· Arithmetic operations are not supported when the value of In[] is positive infinity, negative infinity, or nonnumeric data.

· When the data types of In[] and Out are different, an error occurs during compilation.

· When the value of Size exceeds the In[] array, the return value is FALSE and Out does not change.

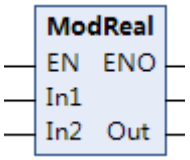
· When In[] or Out is an integer, the decimal portion of the average is truncated.

· When the value of Size is 0, the value of Out is 0 and the return value is TRUE.

3.9. 29 ModReal

This instruction calculates the remainder of real number division.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
ModReal	Real number modulo-division	FC		ModReal (In1 :=,In2 :=, Out =>,);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
In1	Dividend	REAL	Depends on data types	(*)	Dividend
In2	Divisor	REAL	Depends on data types	(*)	Divisor

Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
Out	Remainder	REAL	Depends on data types	(*)	Remainder

	Bool-ean	Bit String				Integer								Real number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In1	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-
In2	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-
Out	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-

■ Function

This instruction divides the dividend In1 by the divisor In2 for the remainder.

The operation for this instruction is performed in the following expression:

Out = In1 - (In1/In2) x In2
The result of the division operation in parentheses is the integer after rounding down.

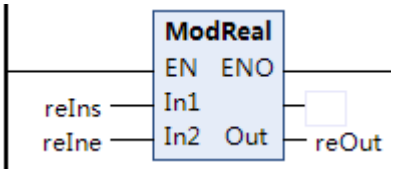
The following table lists examples of the In1, In2, and Out values.

Value of In1	Value of In2	Value of Out
9.9	3.14	0.48
9.9	–3.14	–2.66
–9.9	3.14	2.66
–9.9	–3.14	–0.48

■ Program example

	LD	ST

Defined variable	VAR
	reIns : REAL := 9.9;
	reIne : REAL := 3.14;
	reOut : REAL ;
	END_VAR	

Program		ModReal(In1 := reIns, In2 := reIne, Out => reOut);
Running result	reIns REAL 9.9 reIne REAL 3.14 reOut REAL 0.4799993	

■ Precautions

If the parameter passed to In1 or In2 is an integer, the data type is converted as follows.

Data Type of Parameter Passed to In1 or In2	Data Type of In1 or In2
USIN, UINT, SINT, or INT	REAL
UDINT, DINT, ULINT, or LINT	Data may be lost.

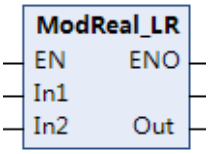
The following table lists the values of Out for different combinations of In1 and In2 values.

0		In1				
		Numerical Value	+ ∞	– ∞	Nonnumeric Data	
In2	0	Nonnumeric data	Nonnumeric data	Nonnumeric data	Nonnumeric data	Nonnumeric data
	Numerical value	0	Remainder of In1/In2	Nonnumeric data	Nonnumeric data	Nonnumeric data
	+ ∞	0	Value of In1	Nonnumeric data	Nonnumeric data	Nonnumeric data
	– ∞	0	Value of In1	Nonnumeric data	Nonnumeric data	Nonnumeric data
	Nonnumeric data	Nonnumeric data	Nonnumeric data	Nonnumeric data	Nonnumeric data	Nonnumeric data

3.9. 30 ModReal_LR

This instruction calculates the remainder of long real number division.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
ModReal_LR	Long real number modulo-division	FC		ModReal_LR (In1 :=,In2 :=, Out =>,);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
In1	Dividend	REAL	Depends on data types	(*)	Dividend
In2	Divisor	REAL	Depends on data types	(*)	Divisor

Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
Out	Remainder	REAL	Depends on data types	(*)	Remainder

	Bool-ean	Bit String				Integer								Real number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In1	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-
In2	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-
Out	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-

■ Function

This instruction divides the dividend In1 by the divisor In2 for the remainder.

The operation for this instruction is performed in the following expression:

Out = In1 - (In1/In2) x In2
The result of the division operation in parentheses is the integer after rounding down.

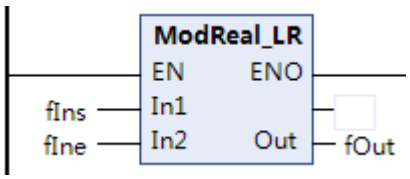
The following table lists examples of the In1, In2, and Out values.

Value of In1	Value of In2	Value of Out
9.9	3.14	0.48
9.9	–3.14	–2.66
–9.9	3.14	2.66
–9.9	–3.14	–0.48

■ Program example

	LD	ST

Defined variable	VAR
	fIns : LREAL := 9.9;
	fIne : LREAL := 3.14;
	fOut : LREAL;
	END_VAR	

Program		ModReal_LR(In1 := fIns, In2 := fIne, Out => fout,);
Running result	fIne LREAL 3.14 fOut LREAL 0.48000000000000043 fout1 LREAL 0	

■ Precautions

If the parameter passed to In1 or In2 is an integer, the data type is converted as follows.

Data Type of Parameter Passed to In1 or In2	Data Type of In1 or In2
USIN, UINT, SINT, INT, UDINT, or DINT	LREAL
ULINT or LINT	Data may be lost.

The following table lists the values of Out for different combinations of In1 and In2 values.

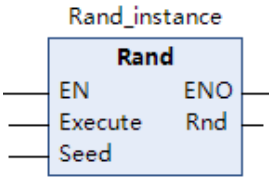
0	In1				
	Numerical Value	+ ∞	– ∞	Nonnumeric Data	

In2	0	Nonnumeric data	Nonnumeric data	Nonnumeric data	Nonnumeric data	Nonnumeric data
	Numerical value	0	Remainder of In1/In2	Nonnumeric data	Nonnumeric data	Nonnumeric data
	+ ∞	0	Value of In1	Nonnumeric data	Nonnumeric data	Nonnumeric data
	– ∞	0	Value of In1	Nonnumeric data	Nonnumeric data	Nonnumeric data
	Nonnumeric data	Nonnumeric data	Nonnumeric data	Nonnumeric data	Nonnumeric data	Nonnumeric data

3.9. 31 Rand

This instruction specifies pseudo random numbers.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
Rand	Random number	FC		Rand_instance(Execute:= , Seed:= , Rnd=>);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
Execute	Trigger	BOOL	Depends on data types	FALSE	Trigger of random numbers
Seed	Random number pattern	UINT	Depends on data types	0	Random number pattern 0: Not specified

Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
Rnd	Random number	LREAL	0.00000000000000e+0 to 1.00000000000000e+0	-	Random number

	Boolean	Bit String				Integer								Real number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Execute	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Seed	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-
Rnd	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-

■ Function

This instruction specifies the random number Rnd. The value of Rnd is different each time the instruction

is executed.

The random number pattern Seed specifies the random number pattern. If the values of Seed are the same, the same random number pattern is generated each time the power supply is turned on. This allows you to generate a repeatable random number.

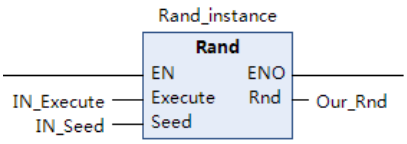
If the value of Seed is 0, random numbers that cannot be repeated are generated. If you do not want to generate the same series of random numbers each time the power supply is turned on, set Seed to 0.

■ Program example

<table>
<tr><th></th><th>LD</th><th>ST</th></tr>
<tr><td>Defined variable</td><td colspan="2">

```
VAR
    Rand_instance: Rand;
    IN_Execute    : BOOL;
    IN_Seed       : UINT;
    Our_Rnd       : LREAL;
END_VAR
```

</td></tr>
<tr><td>Program</td><td>



</td><td>

```
Rand_instance(
    Execute:= IN_Execute,
    Seed:= IN_Seed,
    Rnd=> Our_Rnd
    );
```

</td></tr>
<tr><td>Running result</td><td colspan="2">

Rand_instance	Rand	
IN_Execute	BOOL	TRUE
IN_Seed	UINT	0

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

■ Precautions

The value of Rnd is a real number between 0 and 1.

Chapter 4 Extended Instructions

4. 1 Time and Time of Day Instructions

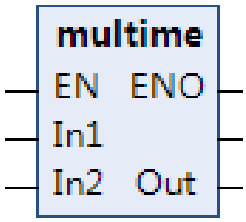
4.1. 1 Instruction List

Instruction Category	Name	FB/FC	Function
Time and time of day instructions	MULTIME	FC	Multiply time
	DIVTIME	FC	Divide time
	TruncTime	FC	Truncate time
	TruncDt	FC	Truncate date and time
	TruncTod	FC	Truncate time of day
	ChkLeapYear	FC	Check for leap year
	GetDaysOfMonth	FC	Get days in month
	GetDayOfWeek	FC	Get day of week
	GetWeekOfYear	FC	Get week number
	ChkLeapYear	FC	Check for leap year
	GetDaysOfMonth	FC	Get days in month
	GetDayOfWeek	FC	Get day of week
	GetWeekOfYear	FC	Get week number
	ADD_TOD_TIME	FC	Add time to time of day
	ADD_DT_TIME	FC	Add time to date and time
	SUB_TOD_TIME	FC	Subtract time from time of day
	SUB_TOD_TOD	FC	Subtract time of day
	SUB_DATE_DATE	FC	Subtract date
	DtToDateStruct	FC	Break down date and time
	DateStructToDt	FC	Join time
	GetSystemDate_sDT	FC	Get system date struct
	GetSystemTime	FB	Get system time
	SysHC_SetSystemDate	FB	Set system date and timezone
	SysHC_GetSystemDate	FB	Get system date and timezone
	DtToSec	FC	Convert date and time to seconds
	DateToSec	FC	Convert date to seconds
	TodToSec	FC	Convert time of day to seconds
	SecToDt	FC	Convert seconds to date and time
	SecToDate	FC	Convert seconds to date
	SecToTod	FC	Convert seconds to time of day
	TimeToNanoSec	FC	Convert time to nanoseconds
	TimeToSec	FC	Convert time to seconds
	NanoSecToTime	FC	Convert nanoseconds to time
Time and time of day instructions	SecToTime	FC	Convert seconds to time
	DaysToMonth	FC	Convert days to month

4.1. 2 MULTIME

This instruction multiplies a time by a specified number.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
MULTIME	Multiply time	FC		MULTIME (In1:=, In2 :=, Out =>);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
In1	Original time	TIME	Depends on data types	-	Original time
In2	Multiplier	-	Depends on data types	-	Multiplier

Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
Out	Resulting time	TIME	Depends on data types	-	Resulting time

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-
In2	-	-	-	-	-	√	√	√	√	√	√	√	√	√	√	-	-	-	-	-
Out	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-

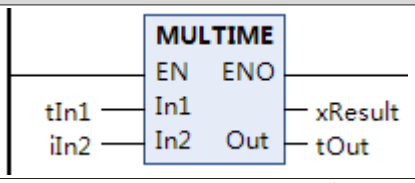
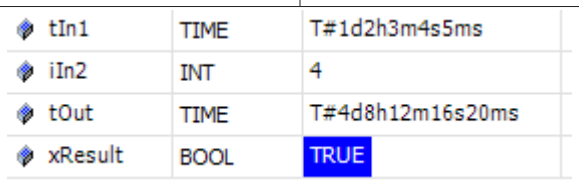
■ Function

This instruction multiplies a time In1 by multiplier In2. The result of multiplication in Out is a time.

■ Program example

Item	LD	ST

Defined variable	VAR
	tIn1 : TIME := T#1D2H3M4S5MS;
	iIn2 : INT := 4;
	tOut : TIME;
	xResult : BOOL;
	END_VAR	

Item	LD	ST
Program		xResult := MULTIME(In1 := tIn1, In2 := iIn2, Out => tOut);
Running result		

■ Precautions

When In2 is negative, a program error occurs, the value of Out is T#0ms, and the return value is FALSE.

When the multiplication result exceeds the valid range of Out, the value of Out is T#0ms and the return value is FALSE.

When the input value of In2 makes the value of Out exceed the upper limit of the TIME type, the value of Out is T#0ms and the return value is FALSE.

When In2 is a real number, the multiplication result less than 1 ms is rounded.

When In2 is a real number, the multiplication result containing more valid digits may cause an error of several milliseconds.

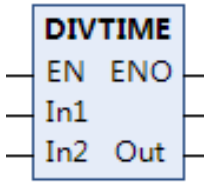
When the value of In2 is positive infinity, negative infinity, or nonnumeric data, the corresponding value of Out is listed in the following table.

Value of In2	Value of Out
+ ∞	T#0ms
– ∞	T#0ms
Nonnumeric data	T#0ms

4.1. 3 DIVTIME

This instruction divides a time by a specified number.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
DIVTIME	Divide time	FC		DIVTIME (In1:=, In2 :=, Out =>);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
In1	Original time	TIME	Depends on data types	-	Original time

Input Variable	Name	Data Type	Value Range	Initial Value	Description
In2	Divisor	-	Depends on data types	-	Divisor

■ Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
Out	Resulting time	TIME	Depends on data types	-	Resulting time

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-
In2	-	-	-	-	-	√	√	√	√	√	√	√	√	√	√	-	-	-	-	-
Out	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-

■ Function

This instruction divides a time In1 by a number In2. The result of division in Out is a time.

■ Program example

Item	LD	ST

Defined variable	VAR
	tIn1 : TIME := T#1D2H3M4S5MS;
	iIn2 : INT := 4;
	tOut : TIME;
	xResult : BOOL;
	END_VAR	

Program	DIVTIME EN ENO tIn1 — In1 iIn2 — In2 Out — tOut xResult	xResult := DIVTIME(In1 := tIn1, In2 := iIn2, Out => tOut);
Running result	tIn1 TIME T#1d2h3m4s5ms iIn2 INT 4 tOut TIME T#6h30m46s1ms xResult BOOL TRUE	

■ Precautions

When In2 is negative, a program error occurs, the value of Out is T#0ms, and the return value is FALSE.

When the division result exceeds the valid range of Out, the value of Out is T#0ms and the return value is FALSE.

When the input value of In2 makes the value of Out exceed the upper limit of the TIME type, the value of Out is T#0ms and the return value is FALSE.

When In2 is an integer with a value of 0, the return value is FALSE and the value of Out is 0.

When In2 is a real number, the division result less than 1 ms is rounded.

When In2 is a real number, the division result containing more valid digits may cause an error of several

milliseconds.

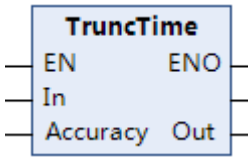
When the value of In2 is 0, positive infinity, negative infinity, or nonnumeric data, the corresponding value of Out is listed in the following table.

Value of In2	Value of Out
0	T#49d17h2m47s295ms
+ ∞	T#0ms
– ∞	T#0ms
Nonnumeric data	T#0ms

4.1. 4 TruncTime

This instruction truncates all digits below the specified unit in a Time variable.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
TruncTime	Truncate time	FC		TruncTime (In :=, Accuracy :=, Out =>);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
In	Time to truncate	TIME	Depends on data types	-	Time to truncate
Accuracy	Smallest unit after truncation	_eSUBSEC	-	_SEC	Smallest time unit to leave after truncation

Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
Out	Time after truncation	TIME	Depends on data types	-	Time after truncation

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-
Accuracy	See "Function" for the enumerators of enumeration type _eSUBSEC.																			
Out	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-

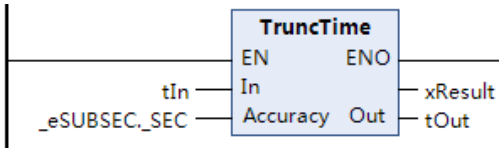
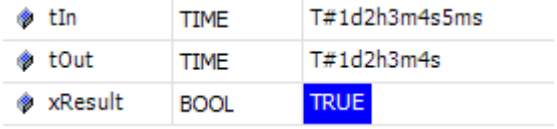
■ Function

This instruction truncates all digits below the smallest unit after truncation that is specified in Accuracy from the time to truncate in In. The resulting time after truncation is stored in time after truncation Out.

The data type of Accuracy is enumerated type _eSUBSEC. The meanings of the enumerators are as follows.

Enumerator	Meaning
_MILLISEC	Millisecond
_SEC	Second
_MINUTE	Minute
_HOUR	Hour

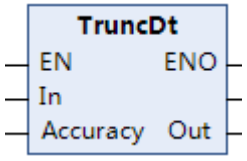
■ Program example

Item	LD	ST
Defined variable		VAR tIn : TIME := T#1D2H3M4S5MS; tOut : TIME; xResult : BOOL; END_VAR
Program		xResult := TruncTime(In := tIn, Accuracy := _eSUBSEC._SEC, Out => tOut);
Running result		

4.1. 5 TruncDt

This instruction truncates all digits below the specified unit in a DT variable.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
TruncDt	Truncate date and time	FC		TruncDt (In :=, Accuracy :=, Out =>);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
In	DT to truncate	DT	Depends on data types	-	DT to truncate
Accuracy	Smallest unit after truncation	_eSUBSEC	-	_SEC	Smallest time unit to leave after truncation

Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
Out	DT after truncation	DT	Depends on data types	-	DT after truncation

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-
Accuracy	See "Function" for the enumerators of enumeration type _eSUBSEC.																			
Out	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-

■ Function

This instruction truncates all digits below the smallest unit after truncation that is specified in Accuracy from the DT to truncate in In. The resulting time after truncation is stored in the DT after truncation Out.

The data type of Accuracy is enumerated type _eSUBSEC. The meanings of the enumerators are as follows. _MILLISEC cannot be used for enumerators. Otherwise, a program error occurs and the return value is FALSE.

Enumerator	Meaning
_MILLISEC	Millisecond
_SEC	Second
_MINUTE	Minute
_HOUR	Hour

■ Program example

Item	LD	ST
Defined variable		VAR tIn : DT := DT#2021-1-1-12:34:56; tOut : DT; xResult : BOOL; END_VAR
Program	TruncDt EN ENO In tIn _eSUBSEC._MINUTE Accuracy Out xResult tOut	xResult := TruncDt(In := tIn, Accuracy := _eSUBSEC._MINUTE, Out => tOut);
Running result	tIn DATE_A... DT#2021-1-1-12:34:56 tOut DATE_A... DT#2021-1-1-12:34:0 xResult BOOL TRUE	

4.1.6 TruncTod

This instruction truncates all digits below the specified unit in a TOD variable.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
TruncTod	Truncate time of day	FC	TruncTod EN ENO In Accuracy Out	TruncTod (In :=, Accuracy :=, Out =>);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
In	TOD to truncate	TOD	Depends on data types	-	TOD to truncate
Accuracy	Smallest unit after truncation	_eSUBSEC	-	_SEC	Smallest time unit to leave after truncation

Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
Out	TOD after truncation	TOD	Depends on data types	-	TOD after truncation

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-
Accuracy	See "Function" for the enumerators of enumeration type _eSUBSEC.																			
Out	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-

■ Function

This instruction truncates all digits below the smallest unit after truncation that is specified in Accuracy from the TOD to truncate in In. The resulting time after truncation is stored in the TOD after truncation Out.

The data type of Accuracy is enumerated type _eSUBSEC. The meanings of the enumerators are as follows.

Enumerator	Meaning
_MILLISEC	Millisecond
_SEC	Second
_MINUTE	Minute
_HOUR	Hour

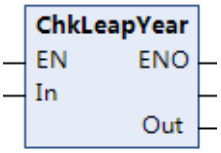
■ Program example

Item	LD	ST
Defined variable	VAR tIn : TOD := TOD#12:34:56.789; tOut : TOD; xResult : BOOL; END_VAR	
Program	TruncTod EN ENO tIn — In _eSUBSEC._MINUTE — Accuracy Out — xResult, tOut	xResult := TruncTod(In := tIn, Accuracy := _eSUBSEC._MINUTE, Out => tOut);
Running result	tIn TIME_OF... TOD#12:34:56.789 tOut TIME_OF... TOD#12:34:0 xResult BOOL TRUE	

4.1. 7 ChkLeapYear

This instruction checks for a leap year.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
ChkLeapYear	Check for leap year	FC		ChkLeapYear (In :=, Out =>);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
In	Year	UINT	1970 to 2106	-	Year

Output variables

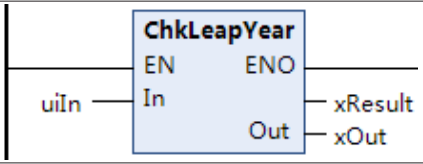
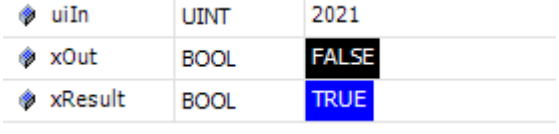
Output Variable	Name	Data Type	Value Range	Initial Value	Description
Out	Result	BOOL	Depends on data types	-	TRUE: Leap year FALSE: Not leap year

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-
Out	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

■ Function

This instruction checks whether the year In is a leap year. If it is a leap year, the value of result Out is TRUE. If it is not a leap year, Out is FALSE.

■ Program example

Item	LD	ST
Defined variable	VAR uiIn : UINT := 2021; xOut : BOOL; xResult : BOOL; END_VAR	
Program		xResult := ChkLeapYear(In := uiIn, Out => xOut);
Running result		

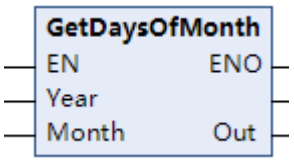
■ Precautions

When the value of In exceeds the valid range, a program error occurs, the value of Out is FALSE, and the return value is FALSE.

4.1. 8 GetDaysOfMonth

This instruction gets the number of days in the specified month.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
GetDaysOfMonth	Get days in month	FC		GetDaysOfMonth(Year :=, Month :=, Out =>);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
In	Year	UINT	1970 to 2106	-	Year
Month	Month	USINT	1 to 12	-	Month

Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
Out	Days	USINT	28 to 31	-	Days

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Year	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-
Month	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Out	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-

■ Function

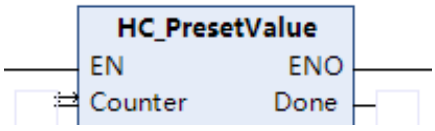
This instruction gets the number of days in month Month of year Year.

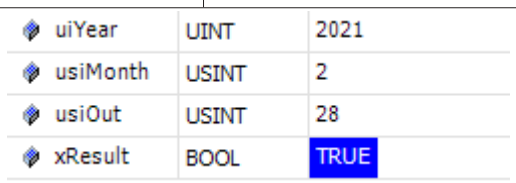
■ Program example

Item	LD	ST

Defined variable	VAR
	uiYear : UINT := 2021;
	usiMonth : USINT := 2;
	usiOut : USINT;
	xResult : BOOL;
	END_VAR	

Item	LD	ST

Program		`xResult := GetDaysOfMonth(Year := uiYear,`
		`Month := usiMonth,`
		`Out => usiOut`
		`);`

Running result		

■ Precautions

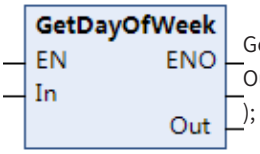
When the value of Year exceeds the valid range, a program error occurs, the value of Out is 0, and the return value is FALSE.

When the value of Month exceeds the valid range, a program error occurs, the value of Out is 0, and the return value is FALSE.

4.1. 9 GetDayOfWeek

This instruction gets the day of the week for the specified year, month, and day of month.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
GetDayOfWeek	Get day of week	FC		GetDayOfWeek(In :=, Out =>);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
In	Year, month, day	DATE/DT	DATE#1970-1-1 to DATE#2106-2-7 DT#1970-1-1-0:0:0 to DT#2106-2-7-6:28:15	-	Year, month, day

Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
Out	Day of the week	_eDAYOFWEEK	Depends on data types	_MON	Day of the week

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	√	-
Out	See "Function" for the enumerators of enumeration type _eDAYOFWEEK.																			

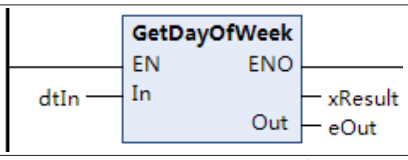
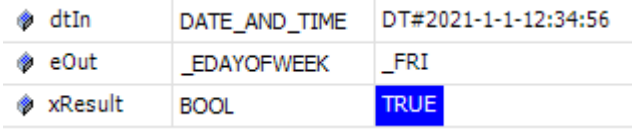
■ Function

This instruction gets the day of the week for the year, month, and day of month specified in In.

The data type of Out is enumerated type _eDAYOFWEEK. The meanings of the enumerators are as follows.

Enumerator	Meaning
_MON	Monday
_TUE	Tuesday
_WED	Wednesday
_THU	Thursday
_FRI	Friday
_SAT	Saturday
_SUN	Sunday

■ Program example

Item	LD	ST
Defined variable	VAR dtIn : DT := DT#2021-1-1-12:34:56; eOut : _eDAYOFWEEK; xResult : BOOL; END_VAR	
Program		xResult := GetDayOfWeek(In := dtIn, Out => eOut);
Running result		

■ Precautions

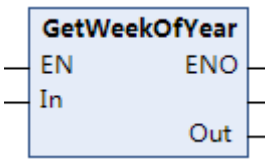
When the value of In exceeds the valid range, a program error occurs, the value of Out is _MON, and the return value is FALSE.

The input parameter transmitted to In must be a variable. If a constant is transmitted to In, a compilation error occurs.

4.1. 10 GetWeekOfYear

This instruction gets the week number for the specified year, month, and day of month.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
GetWeekOfYear	Get week number	FC		GetWeekOfYear (In :=, Out =>);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
In	Year, month, day	DATE/DT	DATE#1970-1-1 to DATE#2106-2-7 DT#1970-1-1-0:0:0 to DT#2106-2-7-6:28:15	-	Year, month, day

Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
Out	Week	USINT	1 to 54	-	Number of week in the year

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	√	-
Out	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-

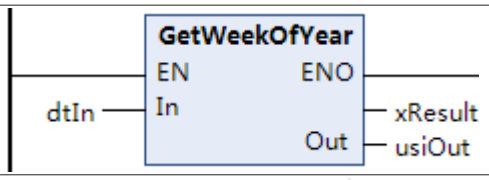
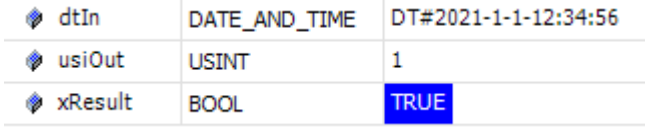
■ Function

This instruction gets the week number for the year, month, and day of month specified in In.

Weeks are counted from Monday to Sunday. The count is incremented when changing from Sunday to Monday.

January 1 is always in week 1.

■ Program example

Item	LD	ST
Defined variable	VAR dtIn : DT := DT#2021-1-1-12:34:56; usiOut : USINT; xResult : BOOL; END_VAR	
Program		xResult := GetWeekOfYear(In := dtIn, Out => usiOut);
Running result		

■ Precautions

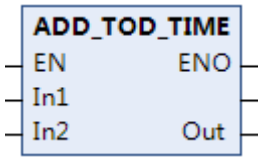
When the value of In exceeds the valid range, a program error occurs, the value of Out is 0, and the return value is FALSE.

The input parameter transmitted to In must be a variable. If a constant is transmitted to In, a compilation error occurs.

4.1. 11 ADD_TOD_TIME

This instruction adds time to time of day.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
ADD_TOD_TIME	Add time to time of day	FC		ADD_TOD_TIME (In1 :=, In2 :=, Out =>);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
In1	TOD to add	TOD	Depends on data types	-	TOD to add
In2	Time to add	TIME	Depends on data types	-	Time to add

Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
Out	Resulting TOD	TOD	Depends on data types	-	Resulting TOD

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-
In2	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-
Out	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-

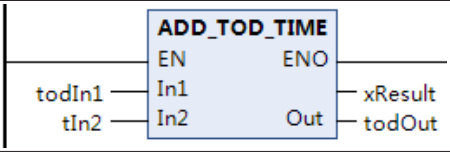
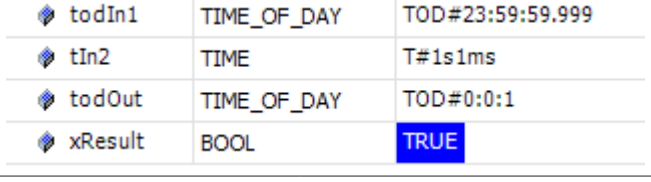
■ Function

This instruction adds a TOD In1 to a time In2. The result of addition in Out is a TOD.

■ Program example

Item	LD	ST

Defined variable	VAR
	todIn1 : TOD := TOD#23:59:59.999;
	tIn2 : TIME := T#1S1MS;
	todOut : TOD;
	xResult : BOOL;
	END_VAR	

Program		xResult := ADD_TOD_TIME(In1 := todIn1, In2 := tIn2, Out => todOut);
Running result		

■ Precautions

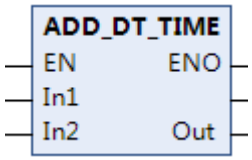
When the result of addition exceeds the valid range of Out, no program error occurs and the following operation is executed.

TOD#23:59:59.999 + T#1s1ms → TOD#0:0:1

4.1. 12 ADD_DT_TIME

This instruction adds time to date and time.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
ADD_DT_TIME	Add time to date and time	FC		ADD_DT_TIME (In1 :=, In2 :=, Out =>);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
In1	DT to add	DT	Depends on data types	-	DT to add
In2	Time to add	TIME	Depends on data types	-	Time to add

Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
Out	Addition result DT	DT	Depends on data types	-	Addition result DT

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-
In2	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-
Out	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-

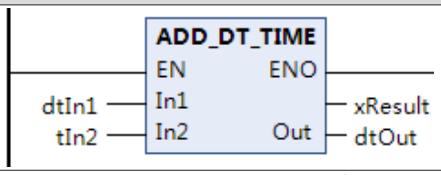
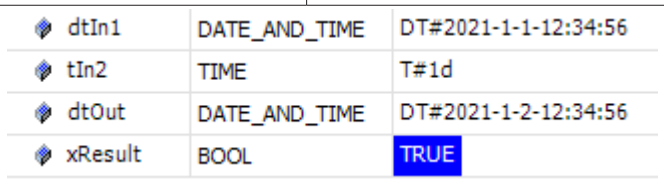
■ Function

This instruction adds a DT In1 to a time In2. The result of addition in Out is a DT.

■ Program example

Item	LD	ST

Defined variable	VAR
	dtIn1 : DT := DT#2021-1-1-12:34:56;
	tIn2 : TIME := T#1D;
	dtOut : DT;
	xResult : BOOL;
	END_VAR	

Item	LD	ST
Program		xResult := ADD_DT_TIME(In1 := dtIn1, In2 := tIn2, Out => dtOut);
Running result		

■ Precautions

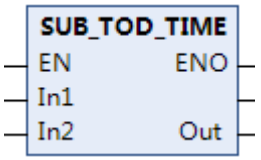
When the result of addition exceeds the valid range of Out, no program error occurs and the following operation is executed.

DT#2106-2-7-6:28:15 + T#1s → DT#1970-1-1-0:0:0

4.1. 13 SUB_TOD_TIME

This instruction subtracts a time from time of day.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
SUB_TOD_TIME	Subtract time from time of day	FC		SUB_TOD_TIME (In1 :=, In2 :=, Out =>);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
In1	Original TOD	TOD	Depends on data types	-	Original TOD
In2	Time to subtract	TIME	Depends on data types	-	Time to subtract

Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
Out	Resulting TOD	TOD	Depends on data types	-	Resulting TOD

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-
In2	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-
Out	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-

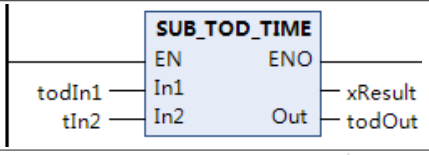
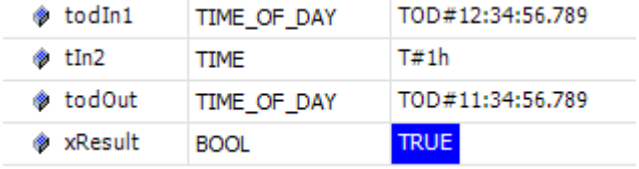
■ Function

This instruction subtracts a time In2 from a TOD In1. The result of subtraction in Out is a TOD.

■ Program example

Item	LD	ST

Defined variable	VAR
	todIn1 : TOD := TOD#12:34:56.789;
	tIn2 : TIME := T#1H;
	todOut : TOD;
	xResult : BOOL;
	END_VAR	

Program		xResult := SUB_TOD_TIME(In1 := todIn1, In2 := tIn2, Out => todOut);
Running result		

■ Precautions

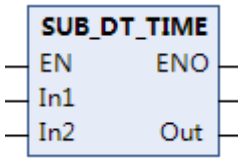
When the result of subtraction exceeds the valid range of Out, no program error occurs and the following operation is executed.

TOD#0:0:0 - T#1ms → TOD#23:59:59:999.

4.1. 14 SUB_DT_TIME

This instruction subtracts a time from date and time.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
SUB_DT_TIME	Subtract time from date and time	FC		SUB_DT_TIME (In1 :=, In2 :=, Out =>);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
In1	Original DT	DT	Depends on data types	-	Original DT
In2	Time to subtract	TIME	Depends on data types	-	Time to subtract

Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
Out	Subtraction result DT	DT	Depends on data types	-	Subtraction result DT

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-
In2	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-
Out	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-

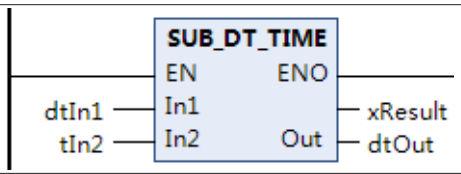
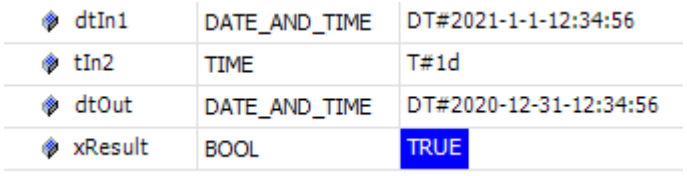
■ Function

This instruction subtracts a time In2 from a DT In1. The result of subtraction in Out is a DT.

■ Program example

Item	LD	ST

Defined variable		VAR
		dtIn1 : DT := DT#2021-1-1-12:34:56;
		tIn2 : TIME := T#1D;
		dtOut : DT;
		xResult : BOOL;
		END_VAR

Program		xResult := SUB_DT_TIME(In1 := dtIn1, In2 := tIn2, Out => dtOut);
Running result		

■ Precautions

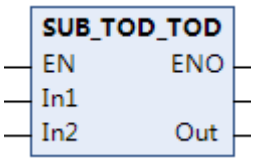
When the result of subtraction exceeds the valid range of Out, no program error occurs and the following operation is executed.

DT#1970-1-1-0:0:0 - T#1d → DT#2106-2-6-6:28:15

4.1. 15 SUB_TOD_TOD

This instruction subtracts time of day from time of day.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
SUB_TOD_TOD	Subtract time of day	FC		SUB_TOD_TOD (In1 :=, In2 :=, Out =>);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
In1	Original TOD	TOD	Depends on data types	-	Original TOD
In2	TOD to subtract	TOD	Depends on data types	-	TOD to subtract

Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
Out	Resulting time	TIME	Depends on data types	-	Resulting time

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-
In2	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-
Out	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-

■ Function

This instruction subtracts a TOD In2 from a TOD In1. The result of subtraction in Out is a time.

■ Program example

Item	LD	ST

Defined variable	VAR
	todIn1 : TOD := TOD#12:34:56.789;
	todIn2 : TOD := TOD#11:11:11.111;
	tOut : TIME;
	xResult : BOOL;
	END_VAR	

Program	SUB_TOD_TOD EN ENO todIn1 In1 todIn2 In2 Out xResult tOut	xResult := SUB_TOD_TOD(In1 := todIn1, In2 := todIn2, Out => tOut);
Running result	todIn1 TIME_OF_DAY TOD#12:34:56.789 todIn2 TIME_OF_DAY TOD#11:11:11.111 tOut TIME T#1h23m45s678ms xResult BOOL TRUE	

■ Precautions

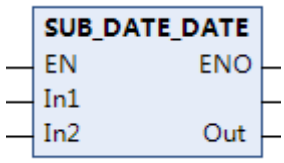
When the value of original TOD In1 is less than the value of TOD to subtract In2, no program error occurs and the following operation is executed.

TOD#13:34:56.789 - TOD#13:35:56.789 → T#23h59m

4.1. 16 SUB_DATE_DATE

This instruction subtracts a date from another date.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
SUB_DATE_DATE	Subtract date	FC		SUB_DATE_DATE (In1 :=, In2 :=, Out =>);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
In1	Original date	DATE	Depends on data types	-	Original date
In2	Date to subtract	DATE	Depends on data types	-	Date to subtract

Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
Out	Resulting time	TIME	Depends on data types	-	Resulting time

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-
In2	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-
Out	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-

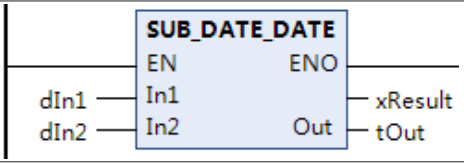
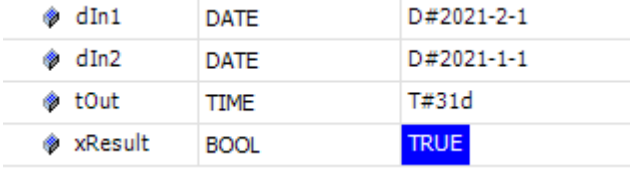
■ Function

This instruction subtracts a date In2 from a date In1. The result of subtraction in Out is a time.

■ Program example

Item	LD	ST

Defined variable	VAR
	dIn1 : DATE := D#2021-2-1;
	dIn2 : DATE := D#2021-1-1;
	tOut : TIME;
	xResult : BOOL;
	END_VAR	

Program		xResult := SUB_DATE_DATE(In1 := dIn1, In2 := dIn2, Out => tOut);
Running result		

■ Precautions

When the result of subtraction exceeds the valid range of Out, a program error occurs, the value of Out is T#0ms, and the return value is FALSE.

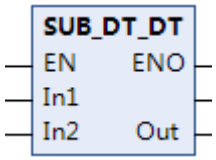
When the value of original date In1 is less than the value of date to subtract In2, no program error occurs and the following operation is executed.

DATE#1970-1-1 - DATE#2106-2-7 → T#1d

4.1. 17 SUB_DT_DT

This instruction subtracts date and time from date and time.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
SUB_DT_DT	Subtract date and time	FC		SUB_DT_DT (In1 :=, In2 :=, Out =>);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
In1	Original DT	DT	Depends on data types	-	Original DT
In2	DT to subtract	DT	Depends on data types	-	DT to subtract

Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
Out	Resulting time	TIME	Depends on data types	-	Resulting time

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-
In2	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-
Out	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-

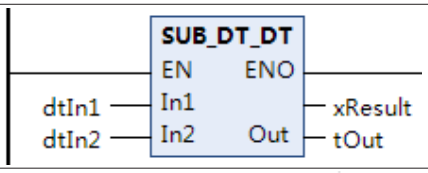
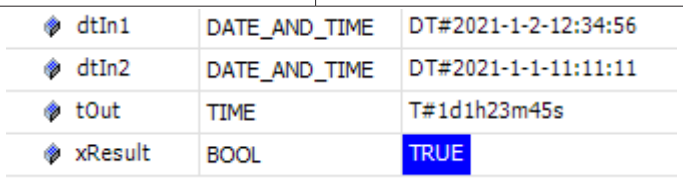
■ Function

This instruction subtracts a DT In2 from a DT In1. The result of subtraction in Out is a time.

■ Program example

Item	LD	ST

Defined variable	VAR
	dtIn1 : DT := DT#2021-1-2-12:34:56;
	dtIn2 : DT := DT#2021-1-1-11:11:11;
	tOut : TIME;
	xResult : BOOL;
	END_VAR	

Program		xResult := SUB_DT_DT(In1 := dtIn1, In2 := dtIn2, Out => tOut);
Running result		

■ Precautions

When the result of subtraction exceeds the valid range of Out, a program error occurs, the value of Out is T#0ms, and the return value is FALSE.

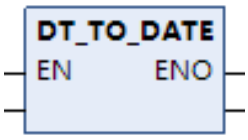
When the value of original date In1 is less than the value of date to subtract In2, no program error occurs and the following operation is executed.

DT#1970-1-1-0:0:0 - DT#2106-2-6-6:28:15 → T#1d1s

4.1. 18 DT_TO_DATE

This instruction extracts a date from date and time.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
DT_TO_DATE	Extract date from date and time	FC		Out := DT_TO_DATE (In);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
In	DT	DT	Depends on data types	-	DT

Output variables

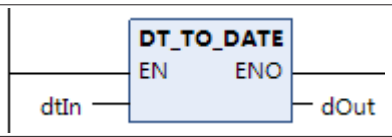
Output Variable	Name	Data Type	Value Range	Initial Value	Description
Out	Date	DATE	Depends on data types	-	Date

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-
Out	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-

■ Function

This instruction extracts the date from date and time In to Out.

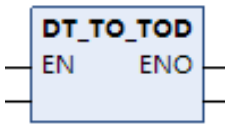
■ Program example

Item	LD	ST
Defined variable	VAR dtIn : DT := DT#2000-1-1-12:34:56; dOut : DATE; END_VAR	
Program		dOut := DT_TO_DATE(dtIn);
Running result	dtIn DATE_AND_TIME DT#2000-1-1-12:34:56 dOut DATE D#2000-1-1	

4.1. 19 DT_TO_TOD

This instruction extracts time of day from date and time.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
DT_TO_TOD	Extract time of day from date and time	FC		Out := DT_TO_TOD (In);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
In	DT	DT	Depends on data types	-	DT

Output variables

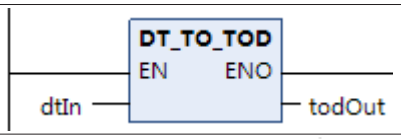
Output Variable	Name	Data Type	Value Range	Initial Value	Description
Out	Tod	TOD	Depends on data types	-	Tod

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-
Out	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-

■ Function

This instruction extracts time of day from date and time In to Out.

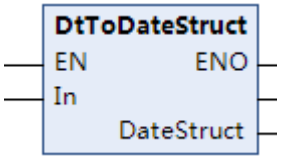
■ Program example

Item	LD	ST
Defined variable	VAR dtIn : DT := DT#2000-1-1-12:34:56; todOut : TOD; END_VAR	
Program		todOut := DT_TO_TOD(dtIn);
Running result	dtIn: DATE_AND_TIME = DT#2000-1-1-12:34:56 todOut: TIME_OF_DAY = TOD#12:34:56	

4.1. 20 DtToDateStruct

This instruction converts date and time to the year, month, day, hour, minutes, and seconds.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
DtToDateStruct	Break down date and time	FC		DtToDateStruct (In :=, DateStruct =>);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
In	DT	DT	DT#1970-1-1-0:0:0 to DT#2106-2-7-6:28:15	DT#1970-1-1-0:0:0	DT to break down

Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
DateStruct	DT	_sDT	-	-	DT after breakdown

_sDT structure

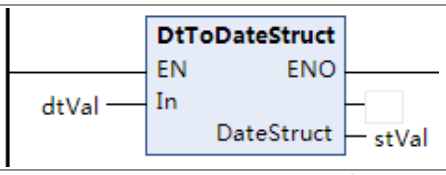
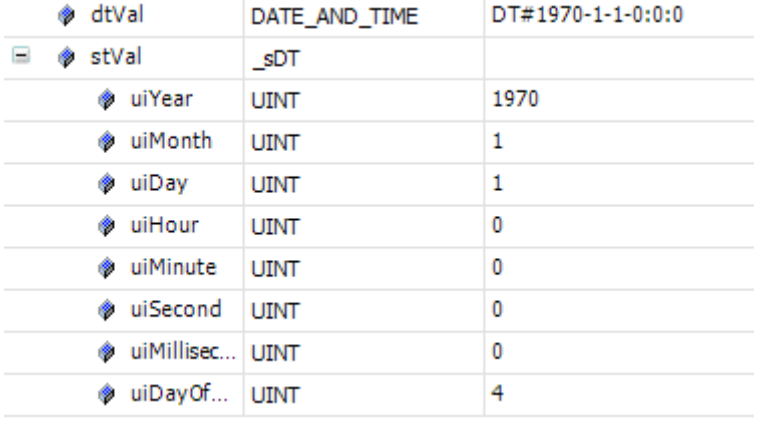
Variable	Name	Description	Data Type	Value Range	Initial Value	Unit
uiYear	Year	Year	UINT	1970 to 2106	1970	Year
uiMonth	Month	Month	UINT	1 to 12	1	Month
uiDay	Day	Day	UINT	1 to 31	1	Day
uiHour	Hour	Hour	UINT	0 to 23	0	Hour
uiMinute	Minute	Minute	UINT	0 to 59	0	Minute
uiSecond	Second	Second	UINT	0 to 59	0	Sec-ond
uiMillisecond	Millisecond	Millisecond	UINT	0 to 999	0	Milli-second
uiDayOfWeek	Day of the week	Day of the week	UINT	1 to 7	1	-

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-
Date-Struct	_sDT structure																			

■ Function

This instruction converts a DT In to the year, month, day, hour, minutes, and seconds in the _sDT structure format.

■ Program example

Item	LD	ST
Defined variable		VAR dtVal : DT := DT#1970-1-1-0:0:0; stVal: _sDT; END_VAR
Program		DtToDateStruct(In := dtVal,DateStruct => stVal);
Running result		

dtVal	DATE_AND_TIME	DT#1970-1-1-0:0:0
stVal	_sDT	
uiYear	UINT	1970
uiMonth	UINT	1
uiDay	UINT	1
uiHour	UINT	0
uiMinute	UINT	0
uiSecond	UINT	0
uiMillisec...	UINT	0
uiDayOf...	UINT	4

■ Precautions

The valid range of the input DT In is DT#1970-1-1-0:0:0 to DT#2106-2-7-6:28:15.

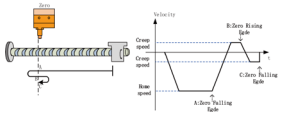
If the value of In exceeds the valid range, the return value is FALSE and the operation result is the breakdown result of DT#1970-1-1-0:0:0.

The initial value of each element in the _sDT structure is the breakdown value of DT#1970-1-1-0:0:0.

4.1. 21 DateStructToDt

This instruction joins a year, month, day, hour, minutes, and seconds into date and time.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
DateStructToDt	Join time	FC		DateStructToDt (In :=, Out =>);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
In	DT	_sDT	-	-	DT as a year, month, day, hour, minutes, and seconds

Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
Out	DT	DT	DT#1970-1-1-0:0:0 to DT#2106-2-7-6:28:15	DT#1970-1-1-0:0:0	DT

_sDT structure

Variable	Name	Description	Data Type	Value Range	Initial Value	Unit
uiYear	Year	Year	UINT	1970 to 2106	1970	Year
uiMonth	Month	Month	UINT	1 to 12	1	Month
uiDay	Day	Day	UINT	1 to 31	1	Day
uiHour	Hour	Hour	UINT	0 to 23	0	Hour
uiMinute	Minute	Minute	UINT	0 to 59	0	Minute
uiSecond	Second	Second	UINT	0 to 59	0	Second
uiMillisecond	Millisecond	Millisecond	UINT	0 to 999	0	Millisecond
uiDayOfWeek	Day of the week	Day of the week	UINT	1 to 7	1	-

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In	_sDT structure																			
Out	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-

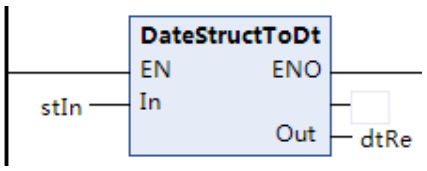
■ Function

This instruction joins a year, month, day, hour, minutes, and seconds into a DT In.

■ Program example

<table>
<tr><th>Item</th><th>LD</th><th>ST</th></tr>
<tr><td>Defined variable</td><td colspan="2">

```
VAR
    stIn : _sDT;
    dtRe : DT;
END_VAR
```

</td></tr>
<tr><td>Program</td><td>


</td><td>

```
stIn.uiYear := 2020;
stIn.uiMonth := 12;
stIn.uiDay := 31;
stIn.uiHour := 23;
stIn.uiMinute := 59;
stIn.uiSecond := 59;
DateStructToDt(In := stIn,Out => dtRe);
```

</td></tr>
<tr><td>Running result</td><td colspan="2">

stIn	_sDT	
uiYear	UINT	2020
uiMonth	UINT	12
uiDay	UINT	31
uiHour	UINT	23
uiMinute	UINT	59
uiSecond	UINT	59
uiMillisec...	UINT	0
uiDayOf...	UINT	1
dtRe	DATE_AND_TIME	DT#2020-12-31-23:59:59

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

■ Precautions

· The valid range of the output DT Out is DT#1970-1-1-0:0:0 to DT#2106-2-7-6:28:15.

· If the value of Out exceeds the valid range, the return value is FALSE and the operation result is DT#1970-1-1-0:0:0.

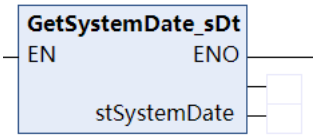
· If the value of In exceeds the valid range of each element in the _sDT structure, the return value is FALSE and the operation result is DT#1970-1-1-0:0:0.

· The initial value of each element in the _sDT structure is the breakdown value of DT#1970-1-1-0:0:0.

4.1. 22 GetSystemDate_sDt

This instruction gets the structure of the current system time.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
GetSystemDate_sDt	Get system date struct	FC		GetSystemDate_sDt(stSystemDate =>);

■ Variables

Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
stSystemDate	System Time	_sDT	-	-	Structure of system time

_sDT structure

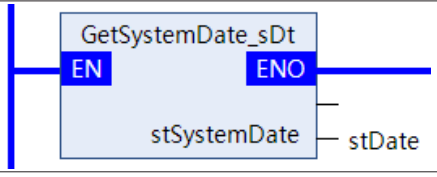
Variable	Name	Description	Data Type	Value Range	Initial Value	Unit
uiYear	Year	Year	UINT	1970 to 2106	0	Year
uiMonth	Month	Month	UINT	1 to 12	0	Month
uiDay	Day	Day	UINT	1 to 31	0	Day
uiHour	Hour	Hour	UINT	0 to 23	0	Hour
uiMinute	Minute	Minute	UINT	0 to 59	0	Min-ute
uiSecond	Second	Second	UINT	0 to 59	0	Sec-ond
uiMillisecond	Millisecond	Millisecond	UINT	0 to 999	0	Milli-sec-ond
uiDayOfWeek	Day of the week	Day of the week	UINT	1 to 7	0	-

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
stSystem-Date	_sDT structure																			

■ Function

This instruction reads the current system time and outputs the time in the _sDT structure format.

■ Program example

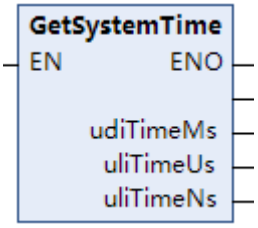
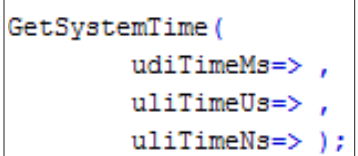
Item	LD	ST
Defined variable	VAR stDate : _sDT; END_VAR	
Program		GetSystemDate_sDt(stSystemDate => stDate);

Item	LD			ST
Running result	stDate	_sDT		
	uiYear	UINT	2021	
	uiMonth	UINT	8	
	uiDay	UINT	31	
	uiHour	UINT	11	
	uiMinute	UINT	28	
	uiSecond	UINT	43	
	uiMillisecond	UINT	523	
	uiDayOfWeek	UINT	2	

4.1. 23 GetSystemTime

This instruction gets the system time after startup.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
GetSystemTime	Get system time	FB		


```
GetSystemTime(
        udiTimeMs=> ,
        uliTimeUs=> ,
        uliTimeNs=> );
```

■ Variables

Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
udiTimeMs	Run time (ms)	UDINT	-	-	Run time (ms)
uliTimeUs	Run time (us)	ULINT	-	-	Run time (us)
uliTimeNs	Run time (ns)	ULINT	-	-	Run time (ns)

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
udiTimeMs	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-
uliTimeUs	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-
uliTimeNs	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-

■ Program example

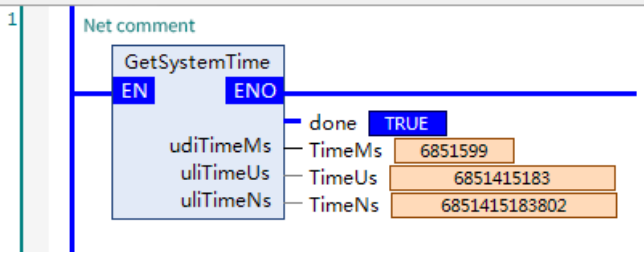
ST

Expression	Type	Value
TimeMs	UDINT	6819814
TimeUs	ULINT	6819633057
TimeNs	ULINT	6819633057657

```
1  GetSystemTime(
2  udiTimeMs=>TimeMs 6819814 ,
3  uliTimeUs=>TimeUs 6819633057 ,
4  uliTimeNs=> TimeNs 6819633057657 );
5  RETURN
```

LD

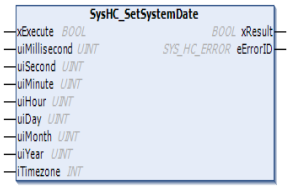
Expression	Type	Value
TimeMs	UDINT	6851599
TimeUs	ULINT	6851415183
TimeNs	ULINT	6851415183802
done	BOOL	TRUE



4.1. 24 SysHC_SetSystemDate

This instruction sets the system date.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
SysHC_SetSystem-Date	Set system date and timezone	FB		SysHC_SetSystemDate(xExecute:= , uiMillisecond:= , uiSecond:= , uiMinute:= , uiHour:= , uiDay:= , uiMonth:= , uiYear:= , iTimezone:= , xResult=> , eErrorID=>);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
xExecute	Execution of trigger	BOOL	[FALSE, TRUE]	FALSE	Execution of trigger at the rising edge
uiMilliseconds	Millisecond	UINT	0 to 999	-	Millisecond (ms)
uiSecond	Second	UINT	0 to 59	-	Second (s)
uiMinute	Minute	UINT	0 to 59	-	Minute (min)
uiHour	Hour	UINT	0 to 23	-	Hour (h)
uiDay	Day	UINT	1 to 31	-	Day
uiMonth	Month	UINT	1 to 12	-	Month
uiYear	Year	UINT	1970 to 2038	-	Year
iTimezone	Time zone	INT	–12 to +12	-	Time zone (reserved and cannot be set)

Output variables

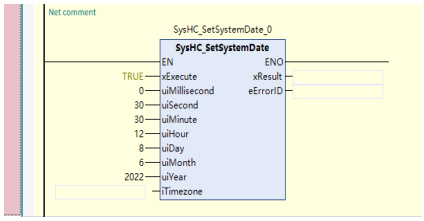
Output Variable	Name	Data Type	Value Range	Initial Value	Description
XResult	Execution Result	BOOL	[FALSE, TRUE]	-	Execution Result
eErrorID	Error ID	SYS_HC_ERROR	Depends on data types	-	Error ID

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
xExecute	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
uiMillisec-ond	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-
uiSecond	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-
uiMinute	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-
uiHour	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-
uiDay	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-
uiMonth	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-
uiYear	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-
iTimezone	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-
XResult	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
eErrorID	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

■ Function

This instruction sets the system date.

■ Program example

Item	LD	ST
Defined variable	VAR SysHC_SetSystemDate_0: SysHC_SetSystemDate; END_VAR	
Program		1 SysHC_SetSystemDate_0(2 xExecute:= TRUE, 3 uiMillisecond:= 0, 4 uiSecond:= 30, 5 uiMinute:= 30, 6 uiHour:= 12, 7 uiDay:= 8, 8 uiMonth:= 6, 9 uiYear:= 2022, 10 iTimezone:= , 11 xResult=> , 12 eErrorID=>);

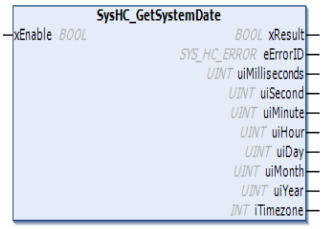
■ Precautions

Frequent triggering of the SetSystemDate instruction may cause task jitters. Replace SetSystemDate with this instruction.

4.1. 25 SysHC_GetSystemDate

This instruction gets the system date.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
SysHC_GetSys-temDate	Get system date and timezone	FB		SysHC_GetSystemDate(xEnable:= , xResult=> , eErrorID=> , uiMilliseconds=> , uiSecond=> , uiMinute=> , uiHour=> , uiDay=> , uiMonth=> , uiYear=> , iTimezone=>);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
xEnable	Function block enable	BOOL	[FALSE, TRUE]	FALSE	Function enable, triggered by level

Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
XResult	Execution Result	BOOL	[FALSE, TRUE]		Execution Result
eErrorID	Error ID	SYS_HC_ERROR	Depends on data types	-	Error ID
uiMilliseconds	Millisecond	UINT	0 to 999	-	Millisecond (ms)

Output Variable	Name	Data Type	Value Range	Initial Value	Description
uiSecond	Second	UINT	0 to 59	-	Second (s)
uiMinute	Minute	UINT	0 to 59	-	Minute (min)
uiHour	Hour	UINT	0 to 23	-	Hour (h)
uiDay	Day	UINT	1 to 31	-	Day
uiMonth	Month	UINT	1 to 12	-	Month
uiYear	Year	UINT	1970 to 2038	-	Year
iTimezone	Time zone	INT	–11 to +11	-	Time zone

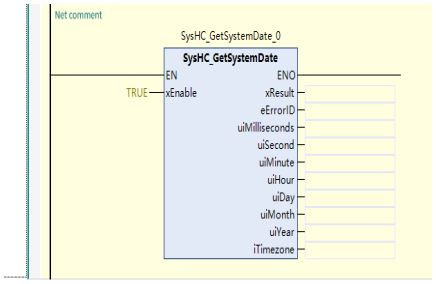
	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
xEnable	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
XResult	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
eErrorID	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
uiMillisec-onds	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-
uiSecond	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-
uiMinute	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-
uiHour	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-
uiDay	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-
uiMonth	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-
uiYear	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-
iTime-zone	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-

■ Function

This instruction gets the system date.

■ Program example

Item	LD	ST
Defined variable		VAR SysHC_GetSystemDate_0: SysHC_GetSystemDate; END_VAR

Item	LD	ST
Program		1 SysHC_GetSystemDate_0(2 xEnable:= TRUE, 3 xResult=> , 4 eErrorID=> , 5 uiMilliseconds=> , 6 uiSecond=> , 7 uiMinute=> , 8 uiHour=> , 9 uiDay=> , 10 uiMonth=> , 11 uiYear=> , 12 iTimezone=>);

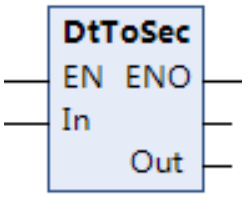
■ Precautions

Frequent triggering of the GetSystemDate instruction may cause task jitters. Replace GetSystemDate with this instruction.

4.1. 26 DtToSec

This instruction converts date and time to the number of seconds from 00:00:00 on January 1, 1970.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
DtToSec	Convert date and time to seconds	FC		DtToSec(In:= , Out=>);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
In	DT	DT	Depends on data types	DT#1970-1-1-0:0:0	DT

Output variables

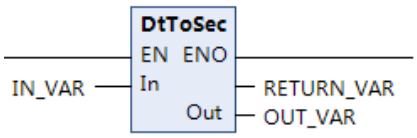
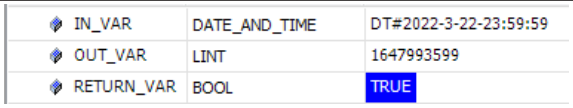
Output Variable	Name	Data Type	Value Range	Initial Value	Description
Out	Conversion result	LINT	0 to 4294967295	-	Number of seconds from 00:00:00 on January 1, 1970

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
IN	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-
Out	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-

■ Function

This instruction converts a DT In to the number of seconds from 00:00:00 on January 1, 1970. The converted value is in seconds. The value below 1 second is truncated.

■ Program example

Item	LD	ST
Defined variable	VAR IN_VAR: DATE_AND_TIME; OUT_VAR: LINT; RETURN_VAR: BOOL; END VAR	
Program		RETURN_VAR:=DtToSec(In:= IN_VAR, Out=>OUT_VAR);
Running result		

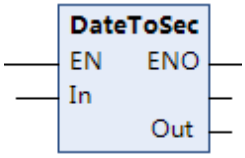
■ Precautions

To convert the number of seconds from 00:00:00 on January 1, 1970 to a date, use the SecToDate instruction.

4.1. 27 DateToSec

This instruction converts a date to the number of seconds from 00:00:00 on January 1, 1970.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
DateToSec	Convert date to seconds	FC		DateToSec(In:= , Out=>);

■ Variables

In-out variables

In-Out Variable	Name	Input/ Output	Description	Value Range	Unit	Initial Value
In	Date	Input	Date	Depends on data types	Year, month, day	D#197
Out	Second	Output	Number of seconds from 00:00:00 on January 1, 1970	0 to 4294967295	Second	-

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description

In	Date	DATE	Depends on data types	D#1970-1-1	Date

Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
Out	Conversion result	0 to 4294967295	Depends on data types	-	Number of seconds from 00:00:00 on January 1, 1970

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-
Out	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-

■ Function

This instruction converts 00:00:00 in date In to the number of seconds from 00:00:00 on January 1, 1970. The converted value is in seconds.

■ Program example

Item	LD	ST
Defined variable	VAR IN_VAR: DATE; OUT_VAR: LINT; RETURN_VAR: BOOL; END VAR	
Program	DateToSec EN ENO IN_VAR — In Out — RETURN_VAR OUT_VAR	RETURN_VAR:=DateToSec(In:=IN_VAR , Out=> OUT_VAR);
Running result	IN_VAR DATE D#2022-3-22 OUT_VAR LINT 1647907200 RETURN_VAR BOOL TRUE	

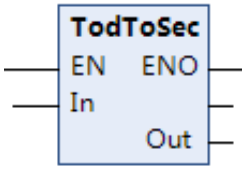
■ Precautions

To convert the number of seconds from 00:00:00 on January 1, 1970 to a date, use the SecToDate instruction.

4.1. 28 TodToSec

This instruction converts time of day to the number of seconds from 00:00:00.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
TodToSec	Convert time of day to seconds	FC		TodToSec(In:= , Out=>);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
In	Tod	TOD	Depends on data types	TOD#0:0:0	Tod

Output variables

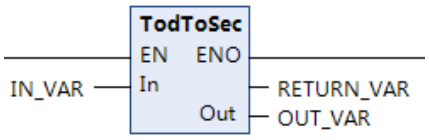
Output Variable	Name	Data Type	Value Range	Initial Value	Description
Out	Conversion result	LINT	0 to 86399	-	Number of seconds from 00:00:00

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-
Out	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-

■ Function

This instruction converts a TOD In to the number of seconds from 00:00:00. The converted value is in seconds. The value below 1 second is truncated.

■ Program example

Item	LD	ST
Defined variable	VAR IN_VAR: TOD; OUT_VAR: LINT; RETURN_VAR: BOOL; END_VAR	
Program		RETURN_VAR:=TODToSec(In:=IN_VAR , Out=> OUT_VAR);
Running result		

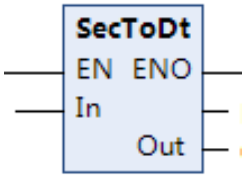
■ Precautions

To convert the number of seconds from 00:00:00 to a TOD, use the SecToTod instruction.

4.1. 29 SecToDt

This instruction converts the number of seconds from 00:00:00 on January 1, 1970 to date and time.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
SecToDt	Convert seconds to date and time	FC		SecToDt(In:= , Out=>);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
In	Second	LINT	0 to 4294967295	0	Number of seconds from 00:00:00 on January 1, 1970

Output variables

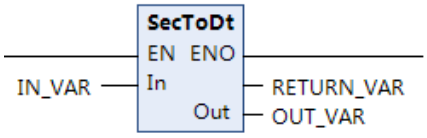
Output Variable	Name	Data Type	Value Range	Initial Value	Description
Out	Conversion result	DT	Depends on data types	-	DT

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-
Out	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-

■ Function

This instruction converts the number of seconds from 00:00:00 on January 1, 1970 in In to a DT.

■ Program example

Item	LD	ST
Defined variable	VAR IN_VAR: LINT; OUT_VAR: DATE_AND_TIME; RETURN_VAR: BOOL; END_VAR	
Program		RETURN_VAR:=SecToDt(In:=IN_VAR , Out=> OUT_VAR);

Item	LD	ST
Running result		

■ Precautions

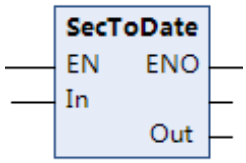
To convert a DT to the number of seconds from 00:00:00 on January 1, 1970, use the DtToSec instruction.

If the value of In exceeds the valid range, the return value changes to FALSE and the value of Out changes to DT#1970-1-1-0:0:0.

4.1. 30 SecToDate

This instruction multiplies a time by a specified number.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
SecToDate	Convert seconds to date	FC		SecToDate(In:= , Out=>);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
In	Second	LINT	0 to 4294967295	0	Number of seconds from 00:00:00 on January 1, 1970

Output variables

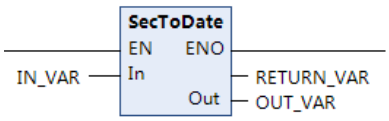
Output Variable	Name	Data Type	Value Range	Initial Value	Description
Out	Conversion result	DATE	Depends on data types	-	Date

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-
Out	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-

■ Function

This instruction converts the number of seconds from 00:00:00 on January 1, 1970 in In to a date. The value below one day is truncated.

■ Program example

Item	LD	ST
Defined variable	VAR IN_VAR: LINT; OUT_VAR: DATE; RETURN_VAR: BOOL; END_VAR	
Program		RETURN_VAR:=SecToDate(In:= IN_VAR, Out=> OUT_VAR);
Running result		

■ Precautions

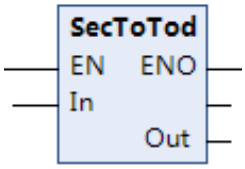
To convert a date to the number of seconds from 00:00:00 on January 1, 1970, use the DateToSec instruction.

If the value of In exceeds the valid range, the return value changes to FALSE and the value of Out changes to D#1970-1-1.

4.1. 31 SecToTod

This instruction converts the number of seconds from 00:00:00 to time of day.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
SecToTod	Convert seconds to time of day	FC		SecToTod(In:= , Out=>);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
In	Second	LINT	0 to 4294967295	0	Number of seconds from 00:00:00 on January 1, 1970

Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
Out	Conversion result	TOD	Depends on data types	-	Tod

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-
Out	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-

■ Function

This instruction converts the number of seconds from 00:00:00 in In to a TOD. When the value of In is greater than 24 hours, divide the value of In by 24 hours and then convert the remainder into a TOD.

■ Program example

Item	LD	ST
Defined variable	VAR IN_VAR: LINT; OUT_VAR: TIME_OF_DAY; RETURN_VAR: BOOL; END_VAR	
Program	SecToTod EN ENO IN_VAR — In Out — OUT_VAR RETURN_VAR	RETURN_VAR:=SecToTod(In:= IN_VAR, Out=> OUT_VAR);
Running result	IN_VAR LINT 666666666 OUT_VAR TIME_OF_DAY TOD#1:11:6 RETURN_VAR BOOL TRUE	

■ Precautions

To convert a TOD to the number of seconds from 00:00:00, use the TodToSec instruction.

If the value of In exceeds the valid range, the return value changes to FALSE and the value of Out changes to TOD#0:0:0.

4.1. 32 TimeToNanoSec

This instruction converts time to the number of nanoseconds.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
TimeToNanoSec	Convert time to nano-seconds	FC	TimeToNanoSec EN ENO In Out	TimeToNanoSec(In:= , Out=>);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description

In	Time	TIME	Depends on data types	T#0s	Time

Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
Out	Conversion result	LINT	Depends on data types	-	Number of nanoseconds

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-
Out	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-

■ Function

This instruction converts time In to the number of nanoseconds.

■ Program example

Item	LD	ST
Defined variable	VAR IN_VAR: TIME; OUT_VAR: LINT; RETURN_VAR: BOOL; END_VAR	
Program	TimeToNanoSec EN ENO IN_VAR — In Out — OUT_VAR RETURN_VAR	RETURN_VAR:=TimeToNanoSec(In:= IN_VAR, Out=> OUT_VAR);
Running result	IN_VAR TIME T#1m40s OUT_VAR LINT 100000000000 RETURN_VAR BOOL TRUE	

■ Precautions

To convert the number of nanoseconds to a time, use the NanoSecToTime instruction.

4.1. 33 TimeToSec

This instruction converts a time to the number of seconds.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
TimeToSec	Convert time to seconds	FC	TimeToSec EN ENO In Out	S TimeToSec(In:= , Out=>);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
In	Time	TIME	Depends on data types	T#0s	Time

Output variables

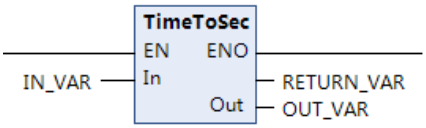
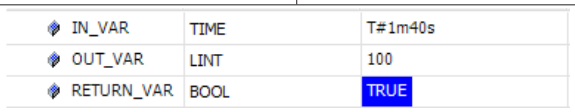
Output Variable	Name	Data Type	Value Range	Initial Value	Description
Out	Conversion result	LINT	Depends on data types	-	Number of seconds

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-
Out	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-

■ Function

This instruction converts time In to the number of seconds. The value below 1 second is truncated.

■ Program example

Item	LD	ST
Defined variable	VAR IN_VAR: TIME; OUT_VAR: LINT; RETURN_VAR: BOOL; END_VAR	
Program		RETURN_VAR:=TimeToSec(In:= IN_VAR, Out=> OUT_VAR);
Running result		

■ Precautions

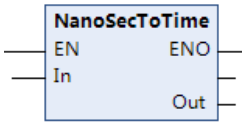
To convert the number of seconds to a time, use the SecToTime instruction.

The unit of In is millisecond, and the unit of Out is second.

4.1. 34 NanoSecToTime

This instruction converts the number of nanoseconds to a time.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
NanoSecToTime	Convert nanoseconds to time	FC		NanoSecToTime(In:= Out=>);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
In	Number of seconds	LINT	0 to 4294967295000000	0	Number of seconds

Output variables

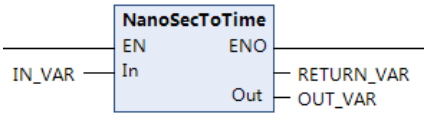
Output Variable	Name	Data Type	Value Range	Initial Value	Description
Out	Conversion result	TIME	Depends on data types	-	Time

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-
Out	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-

■ Function

This instruction converts the number of nanoseconds In to a time.

■ Program example

Item	LD	ST
Defined variable	VAR IN_VAR: LINT; OUT_VAR: TIME; RETURN_VAR: BOOL; END_VAR	
Program		RETURN_VAR:=NanoSecToTime(In:= IN_VAR, Out=> OUT_VAR);
Running result		

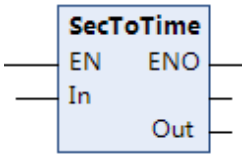
■ Precautions

To convert a time to the number of nanosecond, use the TimeToNanoSec instruction.

4.1. 35 SecToTime

This instruction converts the number of seconds to a time.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
SecToTime	Convert seconds to time	FC		SecToTime(In:= , Out=>);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
In	Number of seconds	LINT	0 to 4294967	0	Number of seconds

Output variables

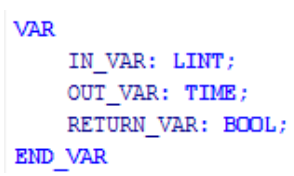
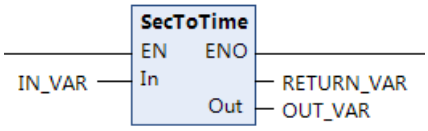
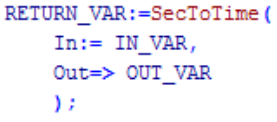
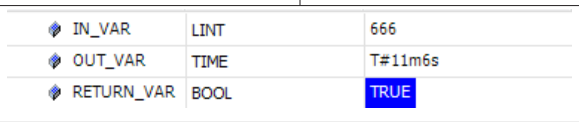
Output Variable	Name	Data Type	Value Range	Initial Value	Description
Out	Conversion result	TIME	Depends on data types	-	Time

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-
Out	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-

■ Function

This instruction converts the number of seconds In to a time.

■ Program example

Item	LD	ST
Defined variable		
Program		
Running result		

■ Precautions

To convert a time to the number of seconds, use the TimeToSec instruction.

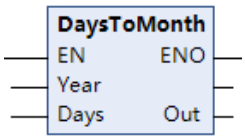
The unit of In is second, and the unit of Out is millisecond.

When the value of In exceeds the valid range, the return value is FALSE and Out does not change.

4.1. 36 DaysToMonth

This instruction calculates the month based on the number of days from January 1.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
DaysToMonth	Convert days to month	FC		DaysToMonth(Year:= , Days:= , Out=>);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
Year	Year	UINT	1970 to 2106	1970	Year
Days	Days	UINT	1 to 365 1 to 366 for a leap year	1	Number of days from January 1

Output variables

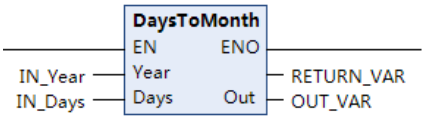
Output Variable	Name	Data Type	Value Range	Initial Value	Description
Out	Month	USINT	1 to 12	-	Month

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Year	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-
Days	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-
Out	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-

■ Function

This instruction calculates the month based on the number of days Days from January 1 in year Year.

■ Program example

<table>
<tr><th>Item</th><th>LD</th><th>ST</th></tr>
<tr><td>Defined variable</td><td colspan="2">
<pre>
VAR
    IN_Year: UINT;
    IN_Days: UINT;
    OUT_VAR: USINT;
    RETURN_VAR: BOOL;
END_VAR
</pre>
</td></tr>
<tr><td>Program</td><td>


</td><td>
<pre>
RETURN_VAR:=DaysToMonth(
    Year:= IN_Year,
    Days:= IN_Days,
    Out=> OUT_VAR
    );
</pre>
</td></tr>
<tr><td>Running result</td><td colspan="2">


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

■ Precautions

If the value of Year exceeds the valid range, no error occurs and the value of Out is an illegal value.

When the value of Days exceeds the valid range, the return value is FALSE and Out does not change.

4. 2 Text String Instructions

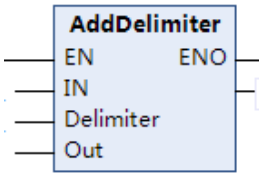
4.2. 1 Instruction List

Instruction Category	Name	FB/FC	Function
Text string instructions	AddDelimiter	FC	Put REAL to text strings with delimiters
	AddDelimiter_LR	FC	Put LREAL to text strings with delimiters
	SubDelimiter	FC	Get text strings to LREAL minus delimiters
	SubDelimiter_LR	FC	Get text strings to LREAL minus delimiters
	ToUCase	FC	Convert to uppercase
	ToLCase	FC	Convert to lowercase
	StringSum	FC	Checksum calculation
	LEN	FC	Get characters count of string
	LEFT	FC	Intercept string, start from left
	RIGHT	FC	Intercept string, start from right
	MID	FC	Intercept string, start from specification
	CONCAT	FC	Contact two strings
	INSERT	FC	Insert string
	DELETE	FC	Delete chars form string
	FIND	FC	Get index of string in searched string
	REPLACE	FC	Replace some characters in a string
	GetByteLen	FC	Get byte length
	ClearString	FC	Clear string
	TrimL	FC	Trim string left
	TrimR	FC	Trim string right

4.2. 2 AddDelimiter

This instruction converts an array element of the REAL type to a text string with a delimiter.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
AddDelimiter	Put REAL to text strings with delimiters	FC		AddDelimiter (In:=, Delimiter:=, Out:=);

■ Variables

Input variables

Input Variable	Name	Description	Value Range	Unit	Initial Value
Delimiter	Delimiter	Processing result array	1, 2, 3, and 4	-	1
Out	Return value	Text string with a delimiter	A maximum of 1985 characters	-	-

In-out variables

In-Out Variable	Name	Description	Value Range	Unit	Initial Value
In (array)	Input array	Array to convert as a text string	Depends on data types	-	-

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In[]	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-
Delimiter	See "Function" for the enumerators for the enumerated type _eDELIMITER.																			
Out	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√

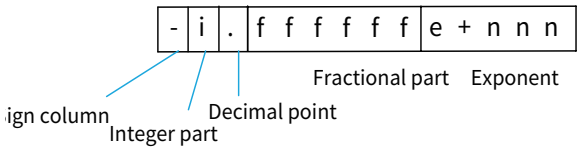
■ Function

This instruction converts the Size array elements starting from In[0] in the In input array to text strings in sequence, which are separated by delimiters Delimiter and then concatenated. The concatenated text string is output to return value Out. A NULL text string is placed at the end of Out.

The data type of Delimiter is enumerated type _Edelimiter. The meanings of the enumerators are as follows.

Enumerator	Meaning
1 (_COMMA)	Comma (,)
2 (_TAB)	$T (tab)
3 (_SEMICOLON)	Semicolon (;)
4 (_SPACE)	Blank character ( )

The structure of the text string to which the value of the array element is converted is shown below.



Item	Description
Sign column	If the value of the array element is negative, a minus sign (−) is added. If the value of the array element is positive, a plus sign (+) is not added.
Integer part	The integer part is always one digit.
Decimal point	The decimal point is always given even if the value of the element is not a decimal number.
Fractional part	The fractional part is six digits.
Exponent	The exponent is always given. e indicates the exponent. nn is 3 digits. The sign of nn is positive (+) if the absolute value of the array element is 1.0 or higher and negative (−) if it is less than 1.0. If the value of the array element is 0, the symbol is the plus sign (+).

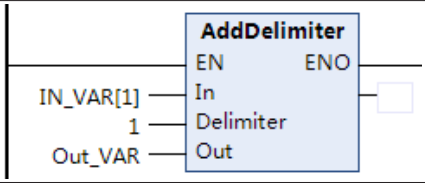
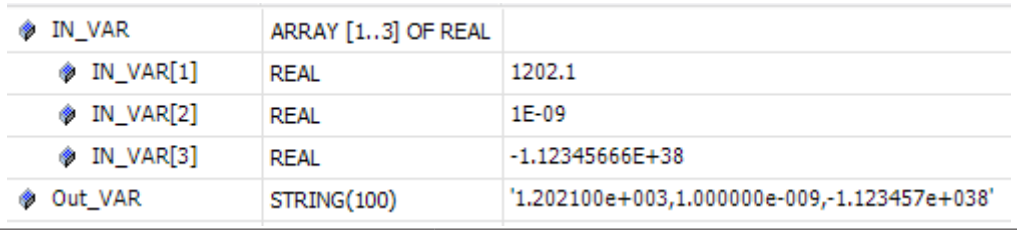
If the value of the array element is infinity or nonnumeric data, the text string is as shown below.

Value of Array Element	String
+ ∞	'inf'
– ∞	'–inf'
Nonnumeric data	'nan'

Examples are given below.

Value of Array Element	Text String After Conversion
REAL#3.14e1	'3.140000e+001'
REAL#-123.4567	'–1.234567e+002'
REAL#0	'0.000000e+000'

■ Program example

Item	LD	ST
Defined variable	VAR IN_VAR: ARRAY[1..3] OF REAL := [1202.10000011,1E-9,-1.1234567E+38]; Out_VAR: STRING[100]; END_VAR	
Program		AddDelimiter(In:= IN_VAR[1], Delimiter:= 1, Out:= Out_VAR);
Running result		

Item	LD	ST
Work principle	IN_Var [1] 1202. 10000011 IN_Var [2] 1E- 9 IN_Var [3] - 1. 12345666E+38 Converted to a character string → [Out] = Out_Var ' 1. 202100e+003, 1. 000000e- 009, - 1. 123457e+038 '	

■ Precautions

When the length (in byte) required to convert the input In is greater than the length (in byte) defined for the output Out, the length of the actual output text string is shorter than the defined length (maximum number of bytes that can be output).

When the length (in byte) required to convert the input In is greater than the maximum length (1985 bytes) defined for the output Out, the length of the actual output text string is within 1985 bytes (maximum number of bytes that can be output).

The number of bytes for the output Out cannot be less than 14. Otherwise, the return value is FALSE and the value of Out does not change.

4.2. 3 AddDelimiter_LR

This instruction converts an array element of the LREAL type to a text string with a delimiter.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
AddDelimiter_LR	Put LREAL to text strings with delimiters	FC	AddDelimiter_LR EN ENO IN Delimiter Out	AddDelimiter_LR (In:=, Delimiter:=, Out:=);

■ Variables

Input variables

Input Variable	Name	Description	Value Range	Unit	Initial Value
Delimiter	Delimiter	Processing result array	1, 2, 3, and 4	-	1
Out	Return value	Text string with a delimiter	A maximum of 198 characters	-	-

In-out variables

In-Out Variable	Name	Description	Value Range	Unit	Initial Value
In (array)	Input array	Array to convert as a text string	Depends on data types	-	-

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In[]	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-
Delimiter	See "Function" for the enumerators for the enumerated type _eDELIMITER.																			

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Out	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√

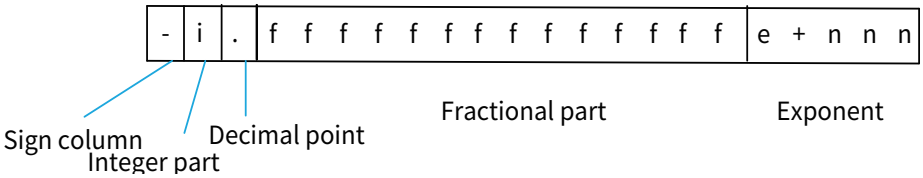
■ Function

This instruction converts the Size array elements starting from In[0] in the In input array to text strings in sequence, which are separated by delimiters Delimiter and then concatenated. The concatenated text string is output to return value Out. A NULL text string is placed at the end of Out.

The data type of Delimiter is enumerated type _Edelimiter. The meanings of the enumerators are as follows.

Enumerator	Meaning
1 (_COMMA)	Comma (,)
2 (_TAB)	$T (tab)
3 (_SEMICOLON)	Semicolon (;)
4 (_SPACE)	Blank character ( )

The structure of the text string to which the value of the array element is converted is shown below.



Item	Description
Sign column	If the value of the array element is negative, a minus sign (−) is added. If the value of the array element is positive, a plus sign (+) is not added.
Integer part	The integer part is always one digit.
Decimal point	The decimal point is always given even if the value of the element is not a decimal number.
Fractional part	The fractional part is 14 digits.
Exponent	The exponent is always given. e indicates the exponent. nn is 3 digits. The sign of nn is positive (+) if the absolute value of the array element is 1.0 or higher and negative (−) if it is less than 1.0. If the value of the array element is 0, the symbol is the plus sign (+).

If the value of the array element is infinity or nonnumeric data, the text string is as shown below.

Value of Array Element	String
+ ∞	'inf'
– ∞	'–inf'
Nonnumeric data	'nan'

Examples are given below.

Value of Array Element	Text String After Conversion
LREAL#3.14e1	'3.14000000000000e+001'

LREAL#-123.4567	-1.23456700000000e+002
LREAL#0	'0.00000000000000e+000'

■ Program example

Item	LD	ST
Defined variable	VAR IN_VAR: ARRAY[1..3] OF LREAL := [1202.10000011,1E-17,-1.12345671234567E+308]; Out_VAR: STRING[100]; END_VAR	
Program	AddDelimiter_LR EN ENO IN_VAR[1] In 1 Delimiter Out_VAR Out	AddDelimiter_LR(In:= IN_VAR[1], Delimiter:= 1, Out:= Out_VAR);
Running result	IN_VAR ARRAY [1..3] OF LR... IN_VAR[1] LREAL 1202.10000011 IN_VAR[2] LREAL 1E-17 IN_VAR[3] LREAL -1.12345671234567E+308 Out_VAR STRING(100) '1.20210000011000e+003,1.00000000000000e-017,-1.12345671234567e+308'	
Work principle	IN_Var[1] 1202.10000011 IN_Var[2] 1E-17 IN_Var[3] -1.12345671234567E+308 Converted to a character string → [Out] = Out_Var '1.202100e+003,1.000000e-017,-1.12345671234567e+308'	

■ Precautions

When the length (in byte) required to convert the input In is greater than the maximum length (1985 bytes) defined for the output Out, the length of the actual output text string is within 1985 bytes (maximum number of bytes that can be output).

The number of bytes for the output Out cannot be less than 22. Otherwise, the return value is FALSE and the value of Out does not change.

4.2. 4 SubDelimiter

This instruction reads delimited data from a text string and stores the results as a real number array.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
SubDelimiter	Get text strings to LREAL minus delimiters	FC	SubDelimiter EN ENO In Delimiter Out	SubDelimiter (In :=, Delimiter:=, Out :=,);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
In	Input text string	STRING	Maximum 1985 bytes	-	Delimited text string to read
Delimiter	Delimiter	_eDELIMITER	Depends on data types	-	Delimiter

In-out variables

In-Out Variable	Name	Data Type	Value Range	Initial Value	Description
Out[]	Real number array of storage position	-	Depends on data types	-	Real number array for storing data after conversion

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√
Delimiter	See "Function" for the enumerators for the enumerated type _eDELIMITER.																			
Out[]	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-

■ Function

This instruction converts the text strings delimited by Delimiter in the input text string In to the real number array Out of the storage position, and saves them as values of array elements in sequence.

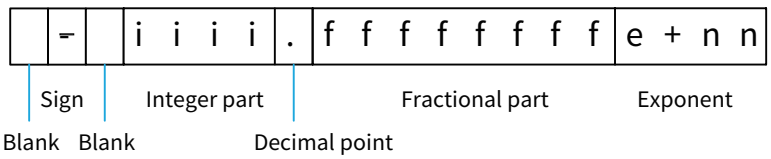
The data type of Delimiter is enumerated type _eDELIMITER. The meanings of the enumerators are as follows.

Enumerator	Meaning
_COMMA	Comma (,)
_TAB	$T (tab)
_SEMICOLON	Semicolon (;)
_SPACE	Blank character ( )

When the number of text strings delimited in In is greater than the number of members in Out, the remaining data is ignored.

When the number of text strings delimited in In is less than the number of members in Out, values of remaining members do not change.

Data delimited in In is of the String type. The format of a real-number text string is given below.



Name	Format
Sign	Any consecutive blank characters at the beginning of the text string are ignored. Any following single plus sign (+) or minus sign (–) is treated as the sign. The plus sign (+) can be omitted. Any consecutive blank characters after the sign are ignored.

Name	Format
Integer part	The characters after the sign and up to the decimal point are taken as the integer part. Any consecutive blank characters after the sign are not included in the integer part. The sign may sometimes be omitted. If the decimal point and fractional part are omitted, the characters up to the exponent are taken as the integer part. If the decimal point, fractional part, and exponent are omitted, the characters up to the end of the text string are taken as the integer part. The integer part consists of 0 to 9. The integer part cannot be omitted. The maximum number of digits in the integer part is the maximum text string length of 1985 minus the total number of bytes in the following: the sign, decimal point, fractional part, exponent, and blank characters before and after the sign.
Decimal point	A single period (.) following the integer part is taken as the decimal point. Omit the decimal point if there is no fractional part.
Fractional part	The characters after the decimal point and up to the exponent are taken as the fractional part. If the exponent is omitted, the characters up to the end of the decimal point are taken as the fractional part. The fractional part consists of 0 to 9. The fractional part can be omitted. The fractional part can consist of a maximum of 15 digits. If there is no decimal point, there is no fractional part.
Exponent	The exponent consists of a single e or E after the fractional part, a following single plus sign (+) or minus sign (–), and the remaining characters to the end of the text string. If there is no fractional part, the above text string after the decimal point is taken as the exponent. If there is no decimal point or fractional part, the above text string after the integer part is taken as the exponent. The numeric part of the exponent consists of 0 to 9. The exponent can be omitted. The numeric part of the exponent can consist of a maximum of three digits.

An example of conversion rules is as follows.

Format	STRING	REAL
With the sign, decimal point, and fractional part, without an exponent	'–123.4567'	–123.4567
With the sign, decimal point, fractional part, and exponent	'+123.4567e+02'	12345.67
Without a sign, with the decimal point, fractional part, and exponent	'123.4567e-02'	1.234567
Without a sign, a decimal point, a fractional part, or an exponent	'1'	1

Precautions for conversion:

1) When the value of the input text string exceeds the valid range of REAL data, if the value is positive, the conversion result is Infinity; if the value is negative, the conversion result is –Infinity.

2) When the input text string does not specify a real number value, the conversion fails and the return value is FALSE.

3) If the input text string contains two or more consecutive underlines (_) between any characters, the conversion is normal.

4) When there is an underline (_) between the negative sign (–) or positive sign (+) at the beginning of the input text string and digits, the conversion is normal.

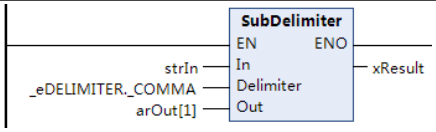
5) When the input text string starts or ends with an underline (_), an error occurs.

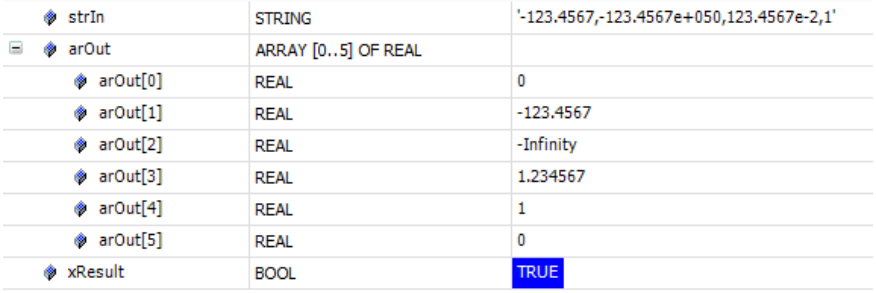
6) When the input text string contains a single underline (_), it is ignored.

7) When the content of the input text string exceeds the accuracy of the REAL data type, the content is rounded off.

8) When the content of the input text string is closer to 0 than the REAL data type, the conversion result is 0.

■ Program example

<table>
<tr><th>Item</th><th>LD</th><th>ST</th></tr>
<tr><td>Defined variable</td><td colspan="2">
<pre>VAR
    strIn   : STRING := '-123.4567,-123.4567e+050,123.4567e-2,1';
    arOut   : ARRAY [0..5] OF REAL;
    xResult : BOOL;</pre>
</td></tr>
<tr><td>Program</td><td>


</td><td>
<pre>xResult := SubDelimiter(In:= strIn,
    Delimiter:= _eDELIMITER._COMMA,
    Out:= arOut[1]);</pre>
</td></tr>
<tr><td>Running result</td><td colspan="2">


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

■ Precautions

When In contains a delimiter between characters, no delimited data exists. If no data exists, the value of the corresponding member in Out[] is REAL#0.

When the text string of In contains non-REAL type data, such data and data following is not converted, the return value is FALSE, and the converted values remain unchanged.

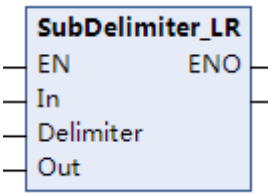
Delimiters can be used only for delimitation in In. If a delimiter is used for other purposes, it is identified only as a delimiter for this instruction.

When the conversion result exceeds the Out[] array, the return value is FALSE and the data within the Out[] array is saved.

4.2. 5 SubDelimiter_LR

This instruction reads delimited data from a text string and stores the results as a long real number array.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
SubDelimiter_LR	Get text strings to LREAL minus delimiters	FC		SubDelimiter_LR (In :=, Delimiter:=, Out :=,);

■ Variables

Input variables

Input Variable	Name	Description	Value Range	Unit	Initial Value
In	Input text string	STRING	Maximum 1985 bytes	-	Delimited text string to read
Delimiter	Delimiter	_eDELIMITER	Depends on data types	-	Delimiter

In-out variables

In-Out Variable	Name	Data Type	Value Range	Initial Value	Description
Out[]	Long real number array of storage position	-	-	-	Long real number array for storing data after conversion

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√
Delimiter	See "Function" for the enumerators for the enumerated type _eDELIMITER.																			
Out	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-

■ Function

This instruction converts the text strings delimited by Delimiter in the input text string In to the long real number array Out of the storage position, and saves them as values of array elements in sequence.

The data type of Delimiter is enumerated type _eDELIMITER. The meanings of the enumerators are as follows.

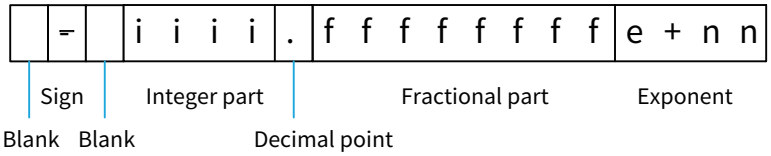
Enumerator	Meaning
_COMMA	Comma (,)
_TAB	$T (tab)
_SEMICOLON	Semicolon (;)
_SPACE	Blank character ( )

When the number of text strings delimited in In is greater than the number of members in Out, the remaining data is ignored.

When the number of text strings delimited in In is less than the number of members in Out, values of remaining members do not change.

Data delimited in In is of the String type. The conversion rules for text string data are as follows.

The format of a long real-number text string is given below.



Name	Format
Sign	Any consecutive blank characters at the beginning of the text string are ignored. Any following single plus sign (+) or minus sign (–) is treated as the sign. The plus sign (+) can be omitted. Any consecutive blank characters after the sign are ignored.
Integer part	The characters after the sign and up to the decimal point are taken as the integer part. Any consecutive blank characters after the sign are not included in the integer part. The sign may sometimes be omitted. If the decimal point and fractional part are omitted, the characters up to the exponent are taken as the integer part. If the decimal point, fractional part, and exponent are omitted, the characters up to the end of the text string are taken as the integer part. The integer part consists of 0 to 9. The integer part cannot be omitted. The maximum number of digits in the integer part is the maximum text string length of 1985 minus the total number of bytes in the following: the sign, decimal point, fractional part, exponent, and blank characters before and after the sign.
Decimal point	A single period (.) following the integer part is taken as the decimal point. Omit the decimal point if there is no fractional part.
Fractional part	The characters after the decimal point and up to the exponent are taken as the fractional part. If the exponent is omitted, the characters up to the end of the decimal point are taken as the fractional part. The fractional part consists of 0 to 9. The fractional part can be omitted. The fractional part can consist of a maximum of 15 digits. If there is no decimal point, there is no fractional part.
Exponent	The exponent consists of a single e or E after the fractional part, a following single plus sign (+) or minus sign (–), and the remaining characters to the end of the text string. If there is no fractional part, the above text string after the decimal point is taken as the exponent. If there is no decimal point or fractional part, the above text string after the integer part is taken as the exponent. The numeric part of the exponent consists of 0 to 9. The exponent can be omitted. The numeric part of the exponent can consist of a maximum of three digits.

An example of conversion rules is as follows.

Format	STRING	LREAL
With the sign, decimal point, and fractional part, without an exponent	'–123.4567'	–123.45670318603516
With the sign, decimal point, fractional part, and exponent	'+123.4567e+02'	12345.669921875
Without a sign, with the decimal point, fractional part, and exponent	'123.4567e-02'	1.2345670461654663
Without a sign, a decimal point, a fractional part, or an exponent	'1'	1

Precautions for conversion:

1) When the value of the input text string exceeds the valid range of LREAL data, if the value is positive, the conversion result is Infinity; if the value is negative, the conversion result is –Infinity.

2) When the input text string does not specify a long real number value, the conversion result is 0 and the return value is FALSE.

3) If the input text string contains two or more consecutive underlines (_) between any characters, the conversion is normal.

4) When there is an underline (_) between the negative sign (–) or positive sign (+) at the beginning of the input text string and digits, the conversion is normal.

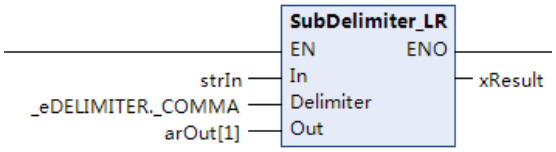
5) When the input text string starts or ends with an underline (_), an error occurs.

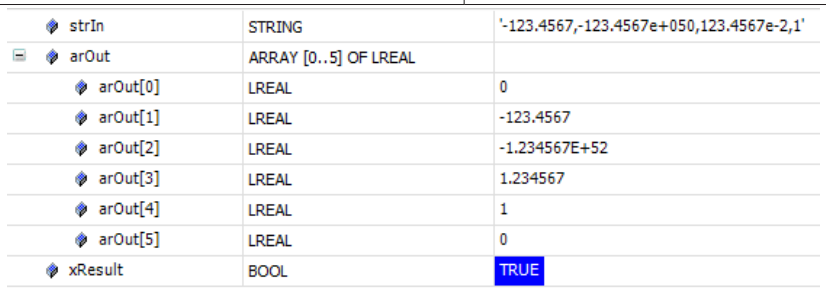
6) When the input text string contains a single underline (_), it is ignored.

7) When the content of the input text string exceeds the accuracy of the LREAL data type, the content is rounded off.

8) When the content of the input text string is closer to 0 than the LREAL data type, the conversion result is 0.

■ Program example

<table>
<tr><th>Item</th><th>LD</th><th>ST</th></tr>
<tr><td>Defined variable</td><td colspan="2">
<pre>VAR
    strIn   : STRING := '-123.4567,-123.4567e+050,123.4567e-2,1';
    arOut   : ARRAY [0..5] OF LREAL;
    xResult : BOOL;</pre>
</td></tr>
<tr><td>Program</td><td>


</td><td>
<pre>xResult := SubDelimiter_LR(In:= strIn,
    Delimiter:= _eDELIMITER._COMMA,
    Out:= arOut[1]);</pre>
</td></tr>
<tr><td>Running result</td><td colspan="2">


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

■ Precautions

When In contains a delimiter between characters, no delimited data exists. If no data exists, the value of the corresponding member in Out[] is LREAL#0.

When the text string of In contains non-REAL type data, such data and data following is not converted, the return value is FALSE, and the converted values remain unchanged.

Delimiters can be used only for delimitation in In. If a delimiter is used for other purposes, it is identified only as a delimiter for this instruction.

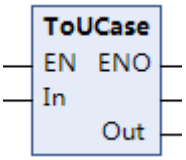
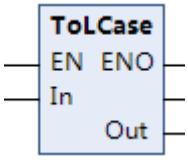
When the conversion result exceeds the Out[] array, the return value is FALSE and the data within the Out[] array is saved.

4.2. 6 ToUCase and ToLCase

ToUCase: Converts all single-byte letters in a text string to uppercase letters.

ToLCase: Converts all single-byte letters in a text string to lowercase letters.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
ToUCase	String Convert to uppercase	FC		ToUCase (In :=,Out =>);
ToLCase	String Convert to lowercase	FC		ToLCase (In :=,Out =>);

■ Variables

Input variables

Input Variable	Name	Description	Value Range	Unit	Initial Value
In	Data to convert	STRING	Depends on data types	0	Text string to convert

Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
Out	Conversion result	STRING	Depends on data types	-	Text String After Conversion

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√
Out	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√
Out	Real number array ARRAR [*] OF LREAL																			

■ Function

ToUCase

This instruction converts all single-byte letters in the text string to convert In to uppercase letters.

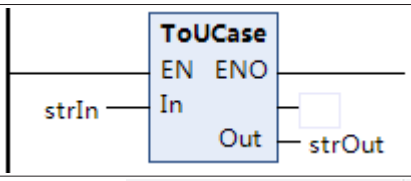
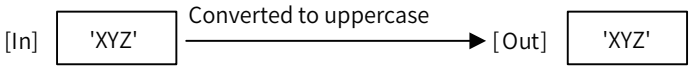
ToLCase

This instruction converts all single-byte letters in the text string to convert In to lowercase letters.

Both instructions output a NULL text string at the end of the text string. Only single-byte characters are changed.

■ Program example

The following example for the ToUCase instruction is for when In is xyz. The output value is XYZ.

Item	LD	ST
Defined variable	VAR strIn : STRING := 'xyz'; strOut : STRING; END_VAR	
Program		ToUCase(In := strIn, Out => strOut);
Running result	strIn STRING 'xyz' strOut STRING 'XYZ'	
Work principle		

■ Precautions

· Two-byte letters are not converted.

· When In results in a character code error, an error may occur and the value of Out is incorrect.

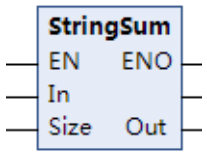
· When In does not end with NULL, the conversion may fail.

· When the conversion result exceeds the value of Out, the conversion value only of the Out length is displayed.

4.2. 7 StringSum

This instruction calculates the checksum for a text string.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
StringSum	Checksum calculation	FC		StringSum(In :=, Size :=, Out :=);

■ Variables

Input variables

Input Variable	Name	Description	Value Range	Unit	Initial Value
In	Text string to convert	STRING	Depends on data types	-	Text string to convert
Size	Byte size	USINT	1 or 2	1	Byte size of checksum

Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
Out	Checksum	STRING	Number of bytes specified by Size	-	Text String After Conversion

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√
Size	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Out	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√

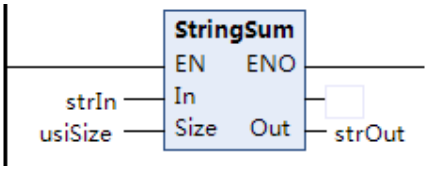
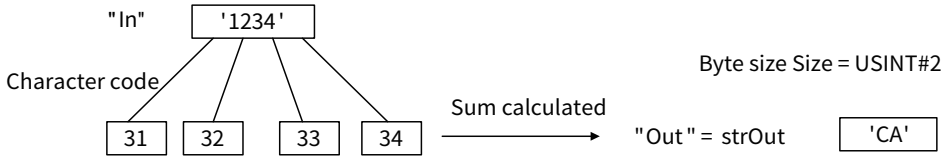
■ Function

This instruction calculates the checksum of the text string to process In. The checksum Out is the number of bytes specified with the byte size Size.

Out is given as a hexadecimal text string with a NULL text string stored at the end.

■ Program example

The following example is for when In is '1234' and Size is USINT#2.

Item	LD	ST
Defined variable	VAR strIn : STRING := '1234'; usiSize : USINT := 2; strOut : STRING; END_VAR	
Program		StringSum(In := strIn, Size := usiSize, Out => strOut);
Running result	strIn STRING '1234' usiSize USINT 2 strOut STRING 'CA'	
Work principle		

■ Precautions

· When the sum of the character codes in In exceeds the number of digits of Size, the upper digits are discarded and the return value is FALSE.

· When the value of Size exceeds the valid range, an error occurs and the return value is FALSE.

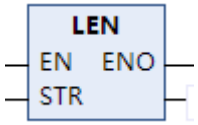
· When the number of bytes in In is 0 (NULL), an error occurs and the return value is FALSE.

When In does not end with NULL, an error may occur.

4.2. 8 LEN

This instruction gets the number of characters in a target text string.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
LEN	Get characters count of string	FC		LEN(STR :=)

■ Variables

Input variables

Input Variable	Name	Description	Value Range	Unit	Initial Value
STR	Source data	STRING	-	''	

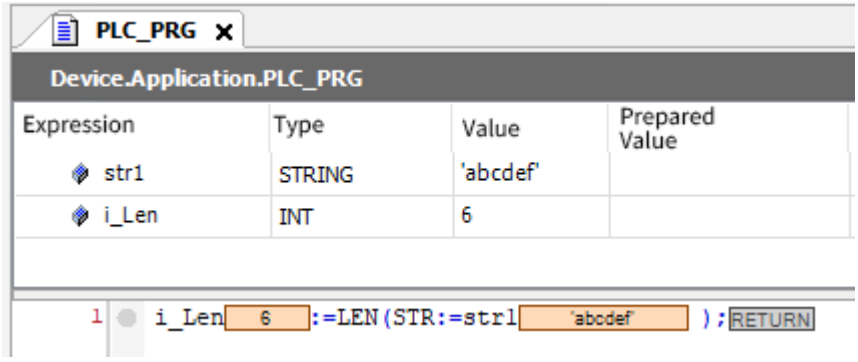
Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
LEN	Return value	INT	0 to 255	0	Number of target text strings

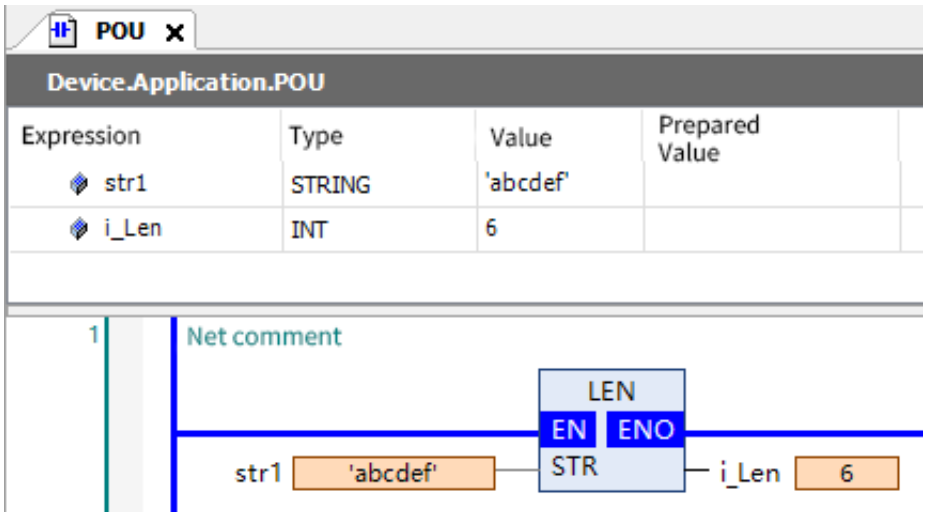
	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
STR	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√
LEN	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-

■ Program example

ST



LD



■ Precautions

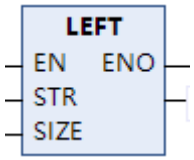
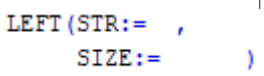
· If the text string size is not defined, the maximum text string length is 80.

· A blank character in a text string also occupies a character.

4.2. 9 LEFT

This instruction extracts and returns characters of a specified length from the left (beginning) of the text string.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
LEFT	Intercept string, start from left	FC		

■ Variables

Input variables

Input Variable	Name	Description	Value Range	Unit	Initial Value
STR	Source data	STRING	-	''	
Size	Data length	INT	0 to 255	0	Data length

Output variables

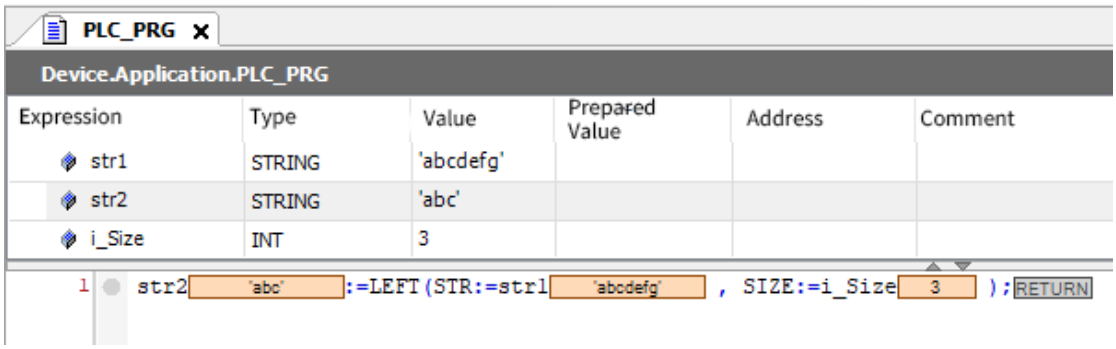
Output Variable	Name	Data Type	Value Range	Initial Value	Description
OUT	Return value	STRING	-	''	Target text string

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
STR	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√
Size	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-
OUT	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√

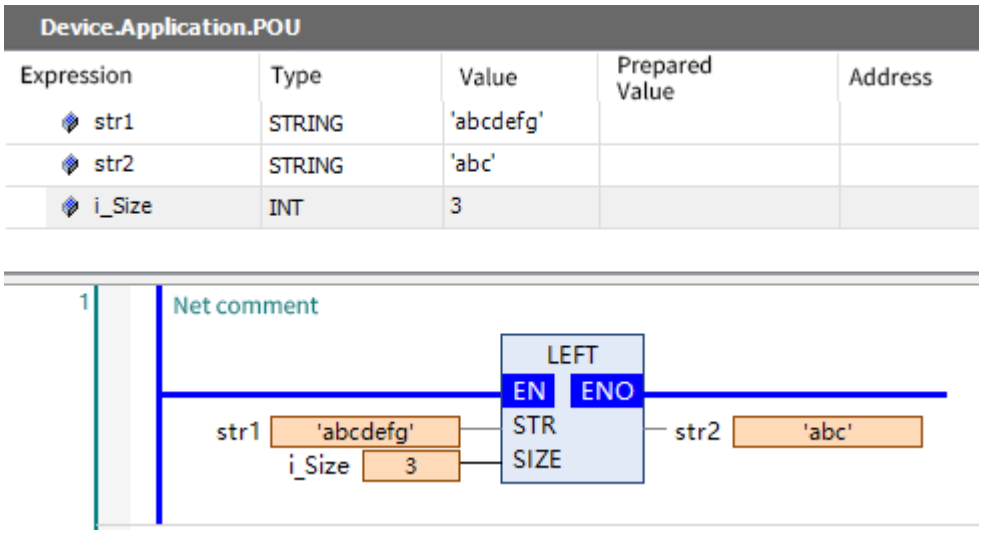
■ Program example

This instruction extracts and returns SIZE characters from the left of STR.

ST



LD



■ Precautions

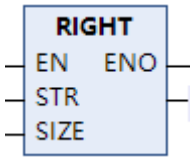
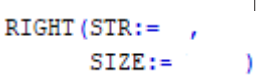
· If the value of SIZE is 0, the output text string is null.

· If the value of SIZE is greater than the length of the input text string, the output text string is the original input text string.

4.2. 10 RIGHT

This instruction extracts and returns characters of a specified length from the right (end) of the text string.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
RIGHT	Intercept string, start from right	FC		

■ Variables

Input variables

Input Variable	Name	Description	Value Range	Unit	Initial Value
STR	Source data	STRING	-	''	
Size	Data length	INT	0 to 255	0	Data length

Output variables

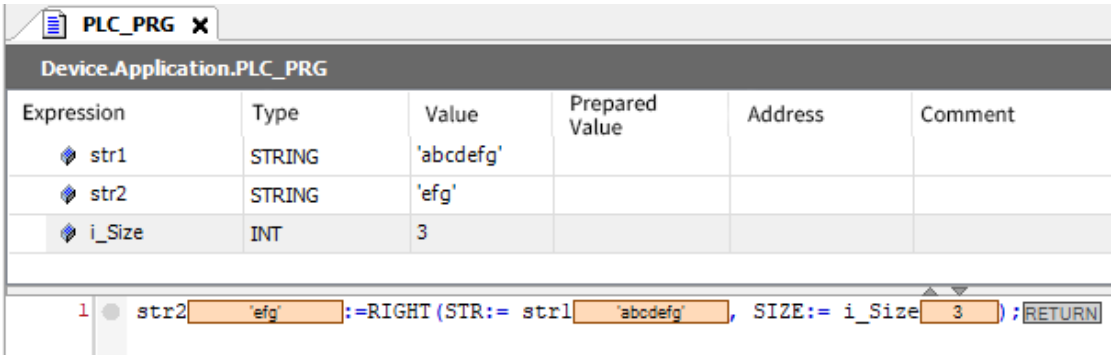
Output Variable	Name	Data Type	Value Range	Initial Value	Description
OUT	Return value	STRING	-	''	Target text string

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
STR	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√
Size	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-
OUT	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√

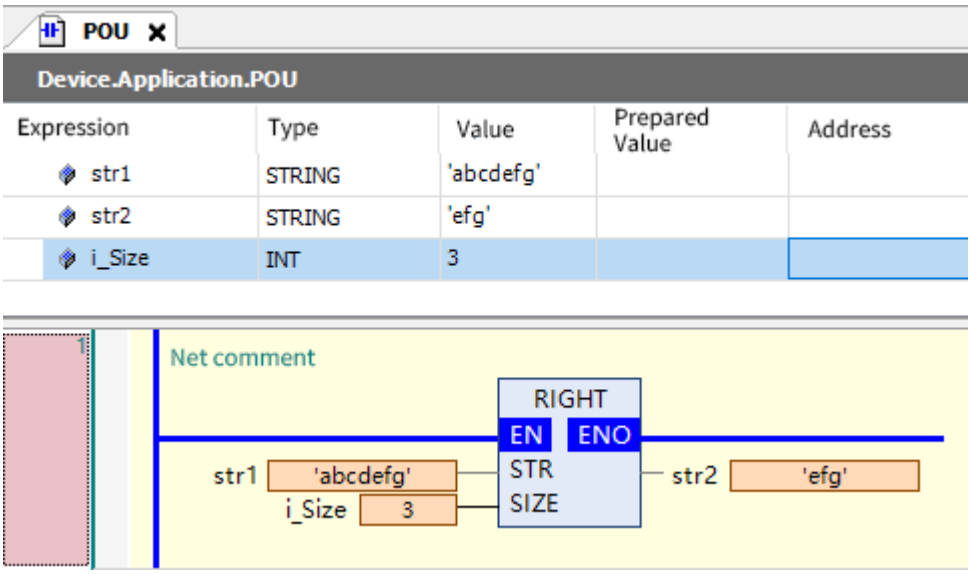
■ Program example

This instruction extracts and returns SIZE characters from the right of STR.

ST



LD



■ Precautions

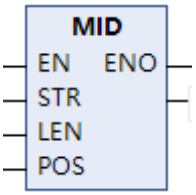
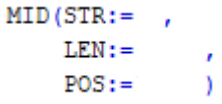
· If the value of SIZE is 0, the output text string is null.

· If the value of SIZE is greater than the length of the input text string, the output text string is the original input text string.

4.2. 11 MID

This instruction extracts and returns characters of a specified length from the specified position of the text string.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
MID	Intercept string, start from specification			

■ Variables

Input variables

Input Variable	Name	Description	Value Range	Unit	Initial Value
STR	Source data	STRING	-	''	
LEN	Data length	INT	0 to 255	0	Data length
POS	Target position	INT	0 to 255	0	Position from which a text string is extracted

Output variables

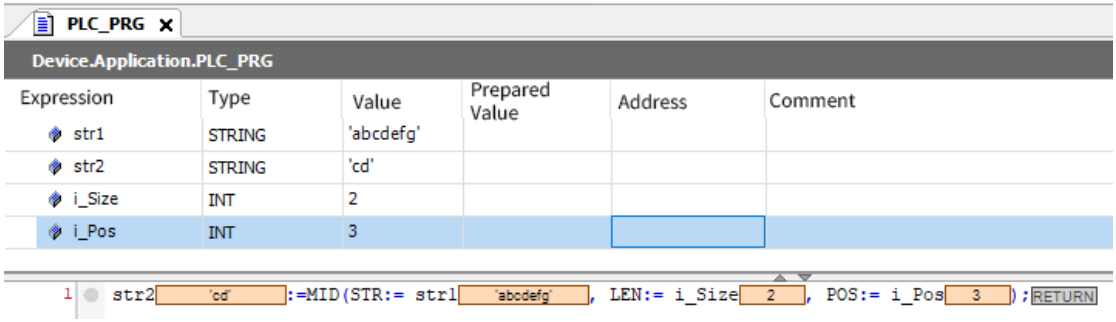
Output Variable	Name	Data Type	Value Range	Initial Value	Description
OUT	Return value	STRING	-	''	Target text string

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
STR	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√
LEN	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-
POS	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-
OUT	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-		√

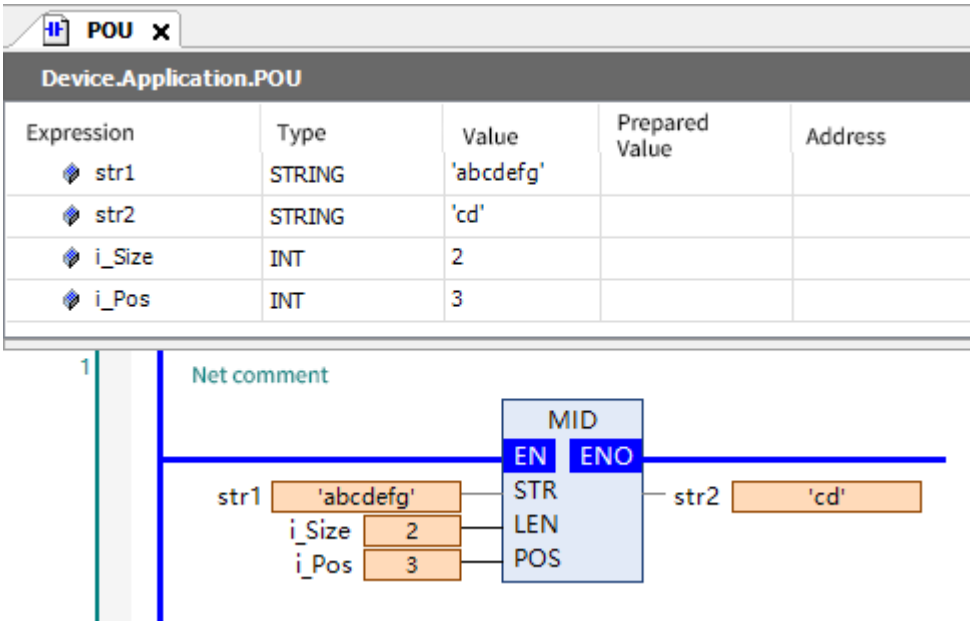
■ Program example

This instruction extracts a text string of set length LEN from the set position POS in the input text string STR as the output text string.

ST



LD



■ Precautions

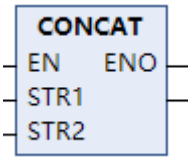
· If the set position POS exceeds the length of the text string, no output is provided.

· If the text string of set length LEN extracted from the set position POS exceeds the remaining length of the input text string, the function effect equals RIGHT.

4.2. 12 CONCAT

This instruction joins two input text strings and provides the output on the right side.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
CONCAT	Contact two strings			CONCAT(STR1:= , STR2:=)

■ Variables

Input variables

Input Variable	Name	Description	Value Range	Unit	Initial Value
STR1	Source data 1	STRING	-	''	
STR2	Source data 2	STRING	-	''	

Output variables

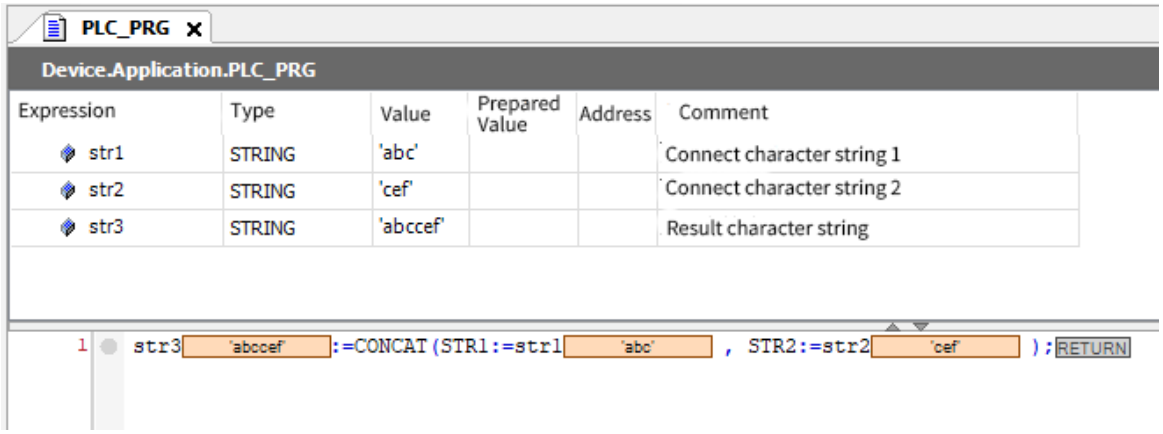
Output Variable	Name	Data Type	Value Range	Initial Value	Description
OUT	Return value	STRING	-	''	Text string resulted from joining

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
STR1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√
STR2	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√
OUT	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√

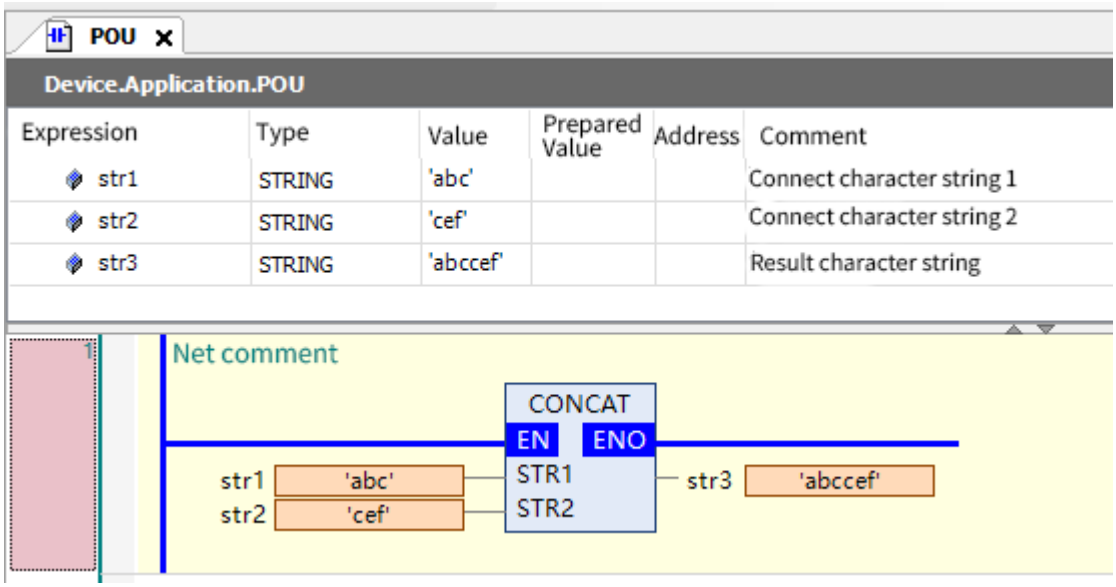
■ Program example

This instruction joins the input text strings STR1 and STR2 and provides the result.

ST



LD



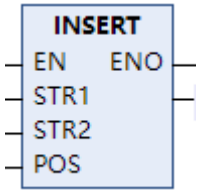
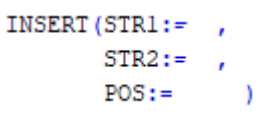
■ Precautions

· If the length of the joined text string exceeds 255, no output is provided.

4.2. 13 INSERT

This instruction inserts a text string within the input source text string on the left, and outputs the resulted text string.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
INSERT	Insert string	FC		

■ Variables

Input variables

Input Variable	Name	Description	Value Range	Unit	Initial Value
STR1	Source data	STRING	-	''	-
STR2	Source data	STRING	-	''	-
POS	Data insertion start position	INT	0 to 255	0	-

Output variables

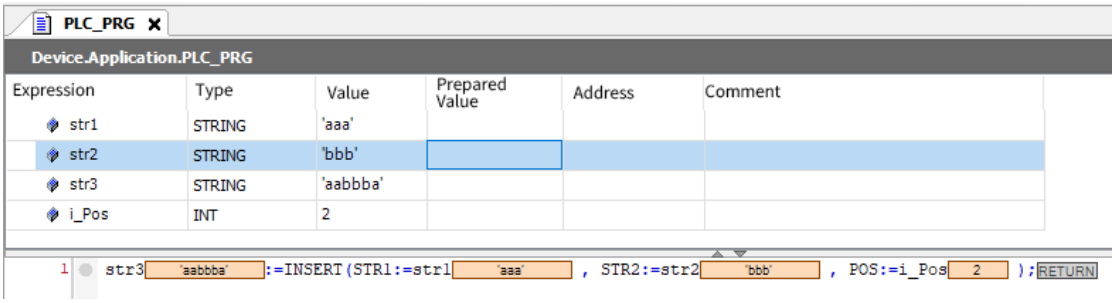
Output Variable	Name	Data Type	Value Range	Initial Value	Description
OUT	Return value	STRING	1	''	Text string after insertion

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
STR1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√
STR2	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√
POS	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-
OUT	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√

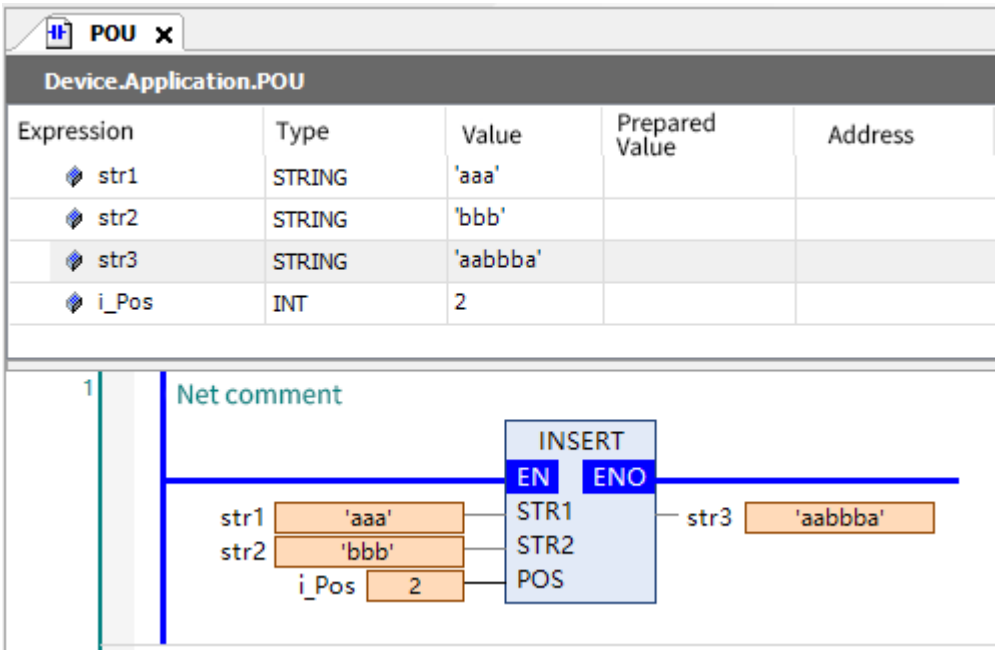
■ Program example

This instruction inserts STR2 next to the POS bit in the input text string STR1 to make a new text string as the STR3 output.

ST



LD



■ Precautions

· If POS is 0, STR1 is inserted at the end of STR2.

· If the value of POS is not less than the text string length, the output is the data of STR1.

4.2. 14 DELETE

This instruction deletes a specified-length text string at a specified position from the input text string, outputting the processed text string.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
DELETE	Delete chars form string	FC	DELETE EN ENO STR LEN POS	DELETE(STR:= , LEN:= , POS:=)

■ Variables

Input variables

Input Variable	Name	Description	Value Range	Unit	Initial Value
STR	Source data	STRING	-	''	-
LEN	Data length	INT	0 to 255	0	Data length
POS	Data position	INT	0 to 255	0	Data position

Output variables

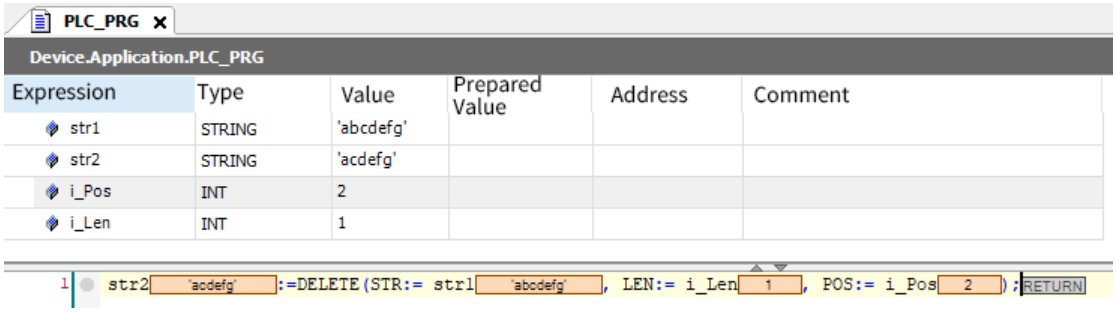
Output Variable	Name	Data Type	Value Range	Initial Value	Description
OUT	Return value	STRING	1	''	Target text string

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
STR	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√
LEN	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-
POS	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-
OUT	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√

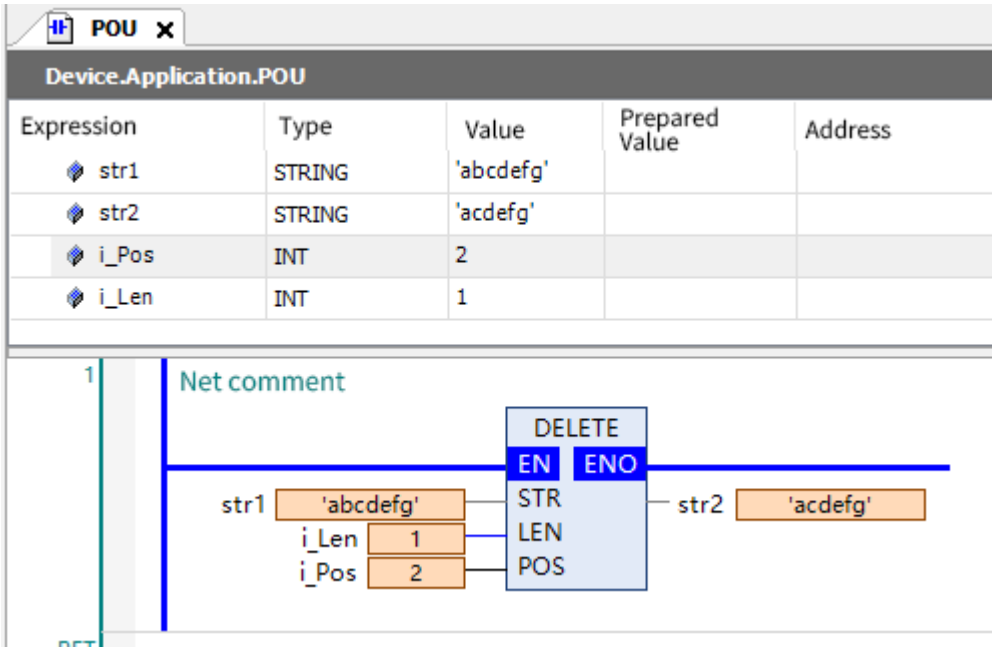
■ Program example

This instruction deletes a text string of LEN length next to the POS position from the input text string, obtaining a new text string as the return value.

ST



LD



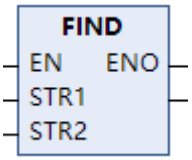
■ Precautions

· If POS is 0, this instruction deletes a text string from the first character.

4.2. 15 FIND

This instruction searches the input source text string for the position of a specified text string.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
FIND	Get index of string in searched string	FC		FIND(STR1:= , STR2:=)

■ Variables

Input variables

Input Variable	Name	Description	Value Range	Unit	Initial Value
STR1	Source data 1	STRING	-	''	Source data
STR2	Source data 2	STRING	-	''	Data to search

Output variables

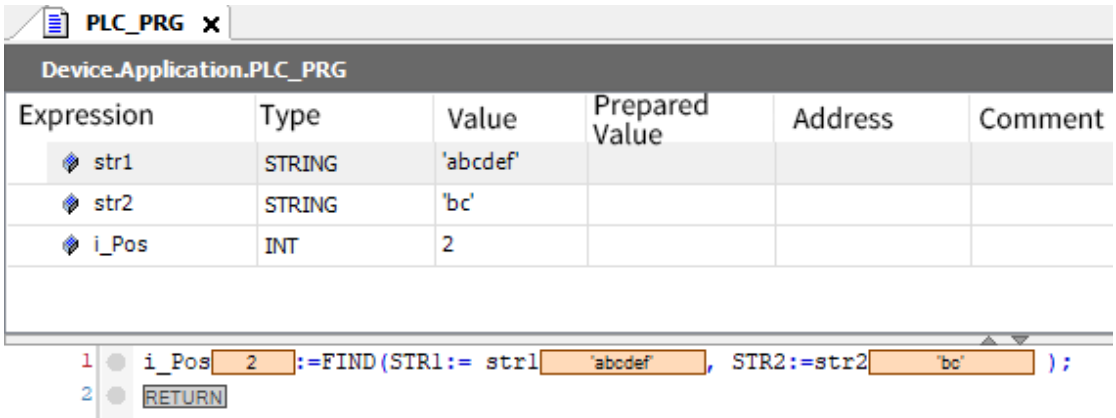
Output Variable	Name	Data Type	Value Range	Initial Value	Description
OUT	Return value	INT	0 to 255	0	Position of a specified text string in the source data

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
STR1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√
STR2	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√
OUT	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-

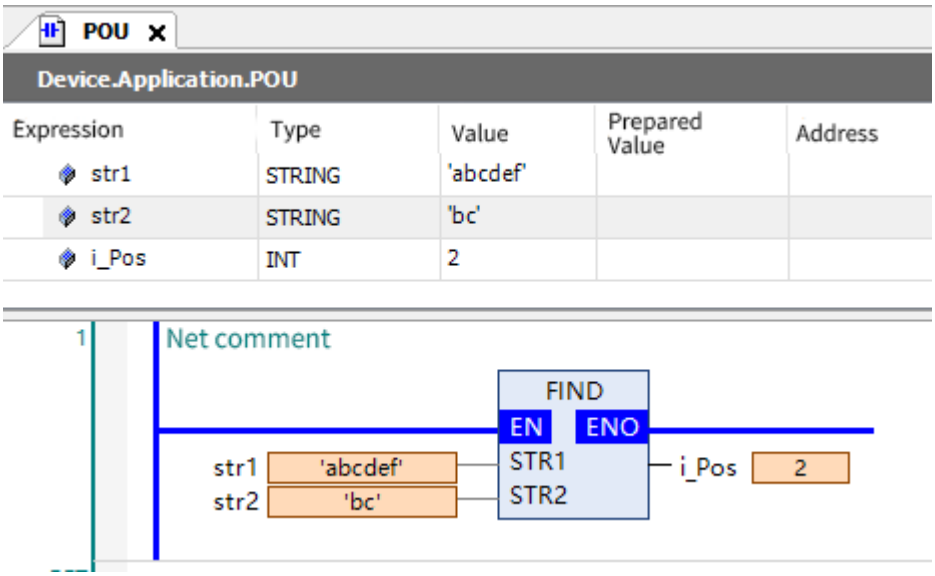
■ Program example

This instruction searches the input text string STR1 for the position of the first character of text string STR2.

ST



LD



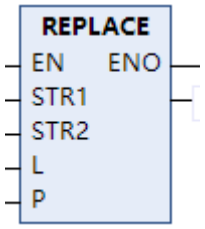
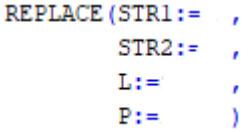
■ Precautions

· If the text string to search for does not exist, the return value is 0.

4.2. 16 REPLACE

This instruction replaces a part of the source text string with a specified text string, obtaining a new text string as the return value.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
REPLACE	Replace some characters in a string	FC		

■ Variables

Input variables

Input Variable	Name	Description	Value Range	Unit	Initial Value
STR1	Source data	STRING	-	''	-
STR2	Source data	STRING	-	''	-
L	Data length	INT	0 to 255	0	Data length
P	Data position	INT	0 to 255	0	Data position

Output variables

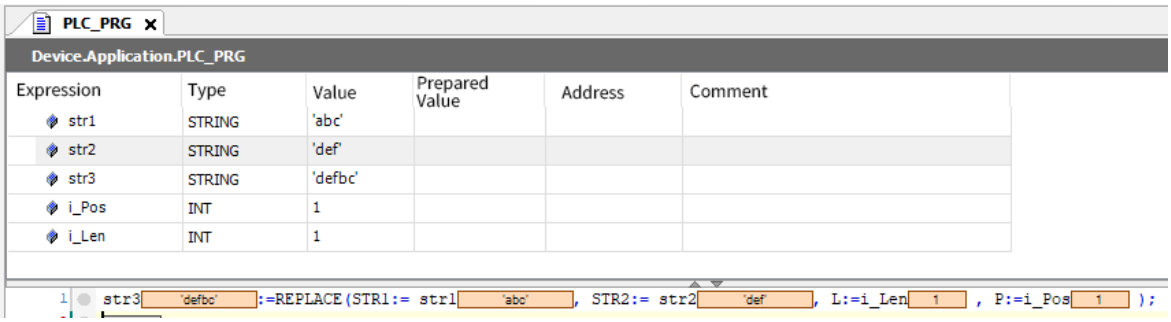
Output Variable	Name	Data Type	Value Range	Initial Value	Description
OUT	Return value	STRING	1	''	Target text string

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
STR1	-	-	-	-	-	-	-	-	-	-	-	-	-	-		-	-	-	-	√
STR2	-	-	-	-	-	-	-	-	-	-	-	-	-	-		-	-	-	-	√
L	-	-	-	-	-	-	-	-	-	-	√	-	-	-		-	-	-	-	-
P	-	-	-	-	-	-	-	-	-	-	√	-	-	-		-	-	-	-	-
OUT	-	-	-	-	-	-	-	-	-	-	-	-	-	-		-	-	-	-	√

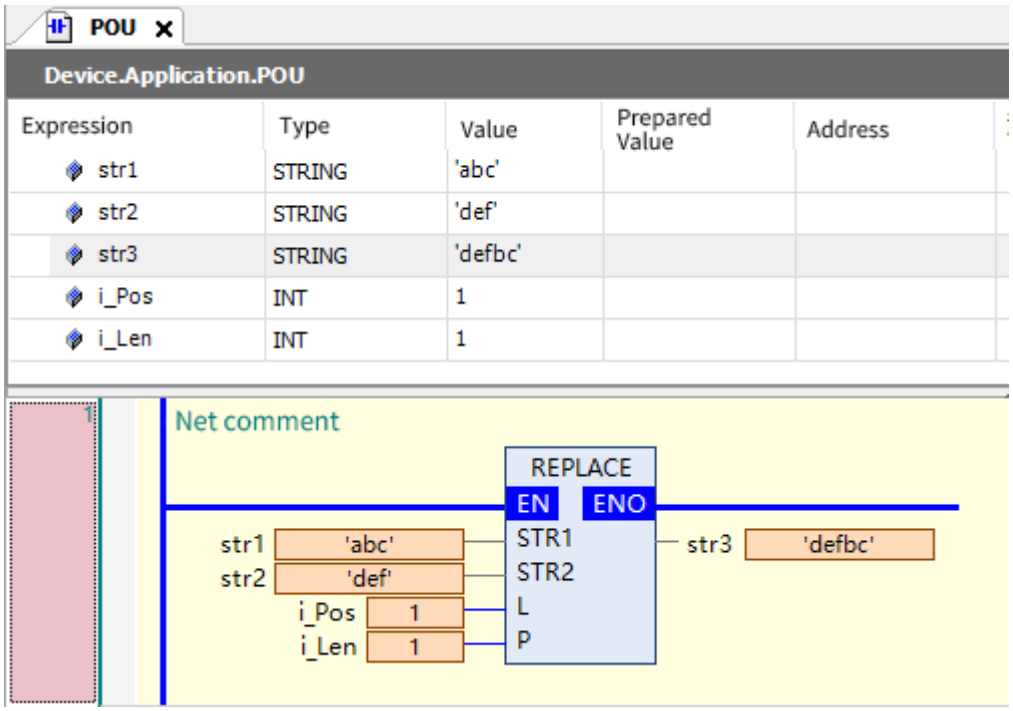
■ Program example

This instruction replaces the text string of L length at P position in the STR1 text string with STR2, outputting a new text string.

ST



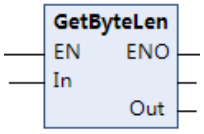
LD



4.2. 17 GetByteLen

This instruction counts the number of bytes in a text string.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
GetByteLen	Get byte length	FC		GetByteLen(In:= , Out=>);

■ Variables

Input variables

Input Variable	Name	Description	Value Range	Unit	Initial Value
In	Count string	STRING	Depends on data types	-	Text string to count number of bytes

Output variables

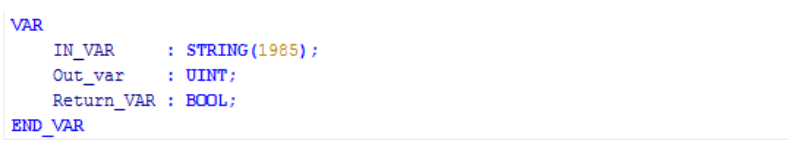
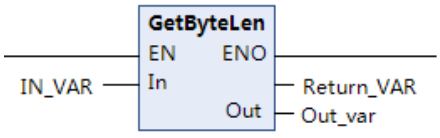
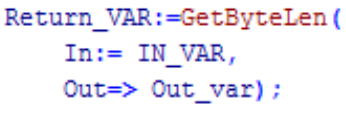
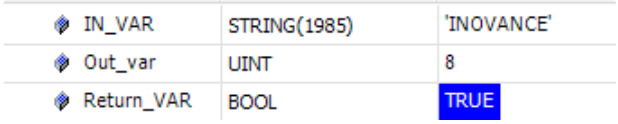
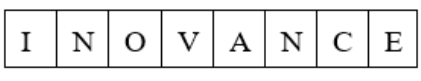
Output Variable	Name	Data Type	Value Range	Initial Value	Description
Out	Number of bytes	UINT	0 to 1985	-	Number of bytes

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
IN	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√
Out	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-

■ Function

This instruction counts the number of bytes in the text string In. A NULL text string at the end of the text string is not counted.

■ Program example

Item	LD	ST
Defined variable		
Program		
Running result		
Work principle	 Number of bytes varies based on memory usage	

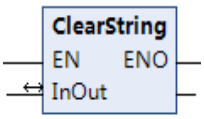
■ Precautions

The length of the input text string is restricted by the data type STRING.

4.2. 18 ClearString

This instruction clears text strings.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
ClearString	Clear string	FC		ClearString(InOut:=);

■ Variables

In-out variables

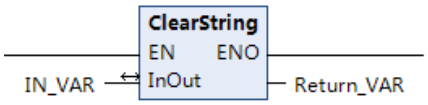
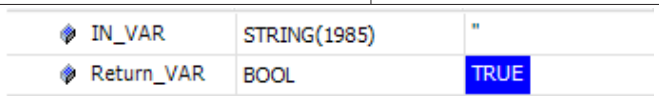
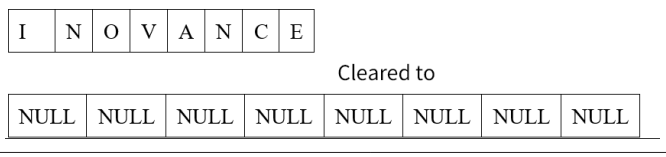
In-Out Variable	Name	Data Type	Value Range	Initial Value	Description
InOut	Clear string	STRING	Depends on data types	-	Text string to clear

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
InOut	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√

■ Function

This instruction clears the text string InOut. The NULL text string is stored in the entire range of InOut.

■ Program example

Item	LD	ST
Defined variable	VAR IN_VAR : STRING(1985); Return_VAR : BOOL; END VAR	
Program		Return_VAR:=ClearString(InOut:= IN_VAR);
Running result		
Work principle		

■ Precautions

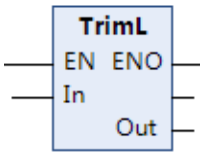
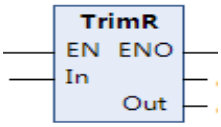
The return value is always TRUE.

4.2. 19 TrimL and TrimR

TrimL: Removes blank space from the start (left) of a text string.

TrimR: Removes blank space from the end (right) of a text string.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
TrimL	Trim string left	FC		TrimL(In:= , Out=>)
TrimR	Trim string right			Trimr(In:= , Out=>)

■ Variables

In-out variables

In-Out Variable	Name	Input/ Output	Description	Value Range	Unit	Initial Value
In	String to trim	Input	Text string to trim	Depends on data types	-	-
Out	Trimming result	Output	Text string after trimming	Depends on data types	-	-

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
In	String to trim	STRING	Depends on data types	-	Text string to trim

Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
Out	Trimming result	STRING	Depends on data types	-	Text string after trim-ming

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√
Out	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√

■ Function

TrimL

This instruction deletes the blank characters from the beginning of the text string to trim In. If there are no

blank characters at the beginning of the text string, no operation is required.

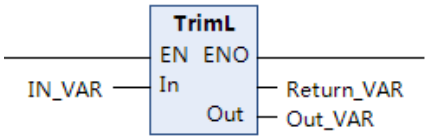
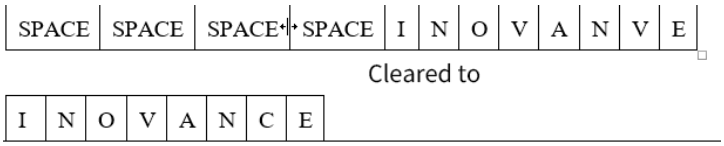
TrimR

This instruction deletes the blank characters from the end of the text string to trim In. If there are no blank characters at the end of the text string, no operation is required.

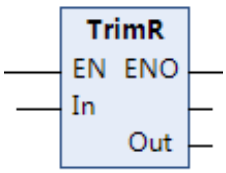
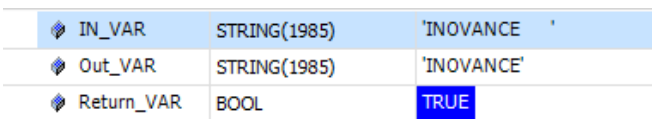
Both instructions output a NULL text string at the end of the text string.

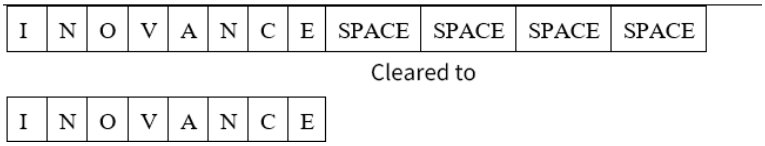
■ Program example

TrimL

Item	LD	ST
Defined variable	VAR IN_VAR : STRING(1985); Out_VAR : STRING(1985); Return_VAR : BOOL; END VAR	
Program		Return_VAR:=TrimL(In:=IN_VAR , Out=>Out_VAR);
Running result		
Work principle		

TrimR

Item	LD	ST
Defined variable	VAR IN_VAR : STRING(1985); Out_VAR : STRING(1985); Return_VAR : BOOL; END VAR	
Program		Return_VAR:=TrimR(In:=IN_VAR , Out=>Out_VAR);
Running result		

Item	LD	ST
Work principle		

■ Precautions

Both instructions output a NULL text string at the end of the text string.

4. 3 Address Operation Instructions

4.3. 1 Instruction List

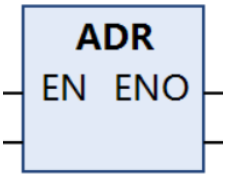
Instruction Category	Name	FB/FC	Function
Address operation instructions	ADR	FC	Get address
	^	FC	Get address content
	BITADR	FC	Get bit address

4.3. 2 ADR and ^

ADR: Obtains the memory address of the input variable and assigns the result to the output variable. This instruction is an operator symbol as an extension of the IEC 61131-3 standard.

^: Obtains the address content of the input variable and assigns the result to the output variable.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
ADR	Get address	FC		ADR( );
^	Get address content	FC		^

■ Variables

ADR

In-Out Variable	Name	Data Type	Value Range	Initial Value	Description
IN	Target data	<TYPE>	-	-	Target data
OUT	Address of target data	POINTER_TO_<-TYPE>	-	-	Address of target data

^

Output Variable	Name	Data Type	Value Range	Initial Value	Description
OUT	Address content output	<TYPE>	-	-	Address content output

<TYPE> in the table indicates the same data type.

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	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>																			
OUT	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

■ Program example

ADR and ^

The ADR instruction returns a referenced address in DWORD format. This address can be used to modify the operation pointer of a function or used for pointer operations in a project.

ST

After program running, the pointer automatically generates the parameter address. Adding ^ next to the pointer indicates the address content pointed to by the pointer. In the following sample address, the value pointed to by the example address is 100.

Expression	Type	Value
Target data	INT	100
Target address output	POINTER TO INT	16#3D640628
Target address output ^	INT	100
Target data content	INT	100

LD

Expression	Type	Value	Prepared Value
Target data	INT	100	
Target address output	POINTER TO INT	16#3D64061C	
Target address output ^	INT	100	
Target data content	INT	100	

■ Precautions

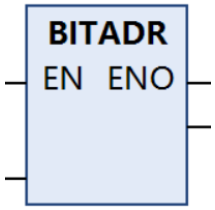
· During online modification, the address content changes, pointing the POINTER variable to an unavailable memory area. Ensure that CODESYS updates the pointer value in each period.

· Do not return the Pointer-TO variable and method of the function to the called function or assign them to global variables.

4.3. 3 BITADR

This instruction obtains the memory address of a Bool-type variable, assigning the result to the output variable.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
BITADR	Get bit address	FC		BITADR( );

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
INT	Target data	BOOL	-	-	Target data

Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
OUT	Target address output	DWORD	-	-	Target address output

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
IN	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
OUT	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

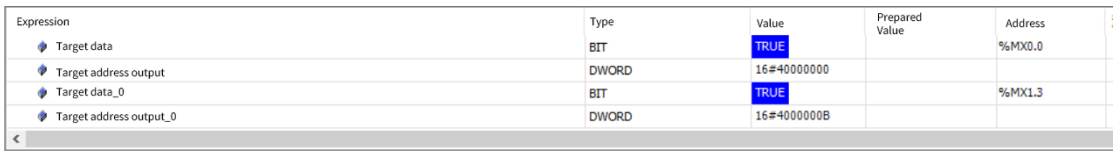
■ Program example

This instruction causes a bit shift in a segment of DWORD.

Example 1: The address returned from DWORD is 16x40000000=16x40000000.

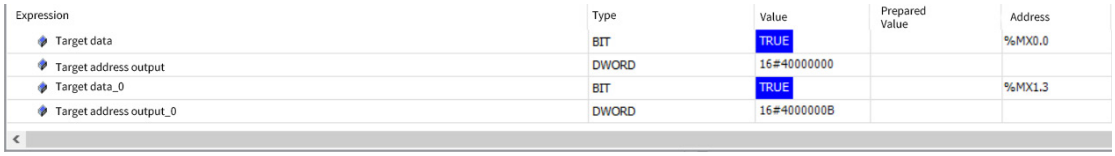
Example 2: The address returned from DWORD is 16x40000000 + 1 x 8 + 3 = 16x4000000B.

ST



Expression	Type	Value	Prepared Value	Address
Target data	BIT	TRUE		%MX0.0
Target address output	DWORD	16#40000000		
Target data_0	BIT	TRUE		%MX1.3
Target address output_0	DWORD	16#4000000B		

LD



Expression	Type	Value	Prepared Value	Address
Target data	BIT	TRUE		%MX0.0
Target address output	DWORD	16#40000000		
Target data_0	BIT	TRUE		%MX1.3
Target address output_0	DWORD	16#4000000B		

■ Precautions

· Apply for an online modification to move the address content when using the pointer address.

4. 4 Queue Instructions

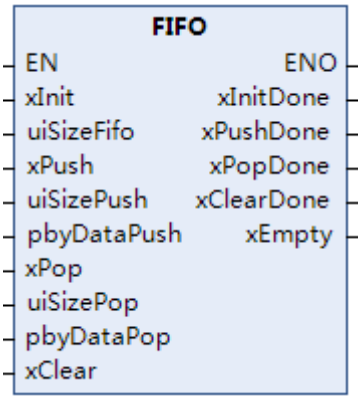
4.4. 1 Instruction List

Instruction Category	Name	FB/FC	Function
Queue instructions	StackPush	FC	Push onto stack
	StackFIFO	FC	First in first out
	StackLIFO	FC	Last in first out
	StackIns	FC	Insert into stack
	StackDel	FC	Delete from stack
	FIFO	FB	First in first out ring queue

4.4. 2 FIFO

This instruction provides the queue function block, including the queue initialization, entering, exiting, and clearing functions.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
FIFO	First in first out ring queue	FB		FIFO(xInit:= , uiSizeFifo:= , xPush:= , uiSizePush:= , pbyDataPush:= , xPop:= , uiSizePop:= , pbyDataPop:= , xClear:= , xInitDone=> , xPushDone=> , xPopDone=> , xClearDone=> , xEmpty=>);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
xInit	Initialization	BOOL	TRUE/FALSE	FALSE	Initialization for memory opening upon trigger at the rising edge, requiring only once initialization after power-on
uiSizeFifo	Initialization for memory allocation	UINT	0 kB to 512 kB	0	Initialization of queue size
xPush	Data to write to queue	BOOL	TRUE/FALSE	FALSE	Data to write to a queue upon trigger at the rising edge

Input Variable	Name	Data Type	Value Range	Initial Value	Description
uiSizePush	Size of data to write to queue	UINT	0 kB to 512 kB	0	Amount of data to write to a queue, in bytes, which should not exceed the initialization value
pbyDataPush	Head address to write data to queue	POINTER TO BYTE	-	-	Head address of the pointer to write data to a queue
xPop	Queue to exit	BOOL	TRUE/FALSE	FALSE	Data to write to a queue upon trigger at the rising edge
uiSizePop	Size of data to read from queue	UINT	0 kB to 512 kB	0	Amount of data to read from a queue, in bytes, which should not exceed the initialization value
pbyDataPop	Head address to read data from a queue	POINTER TO BYTE	-	-	Head address of the pointer to read data from a queue
xClear	Queue clearing	BOOL	TRUE/FALSE	FALSE	Data clearing from a queue upon trigger at the rising edge

Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
xInitDone	Initialization completion flag	BOOL	TRUE/FALSE	FALSE	Initialization completion flag
xPushDone	Queue entered flag	BOOL	TRUE/FALSE	FALSE	Queue entered flag
xPopDone	Queue exited flag	BOOL	TRUE/FALSE	FALSE	Queue exited flag
xClearDone	Queue cleared flag	BOOL	TRUE/FALSE	FALSE	Queue cleared flag
xEmpty	Queue empty flag	BOOL	TRUE/FALSE	FALSE	Queue empty flag

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
xInit	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
uiSizeFifo	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-
xPush	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
uiSizePush	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-
pbyDataPush	POINTER TO BYTE																			
xPop	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
uiSizePop	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-
pbyDataPop	POINTER TO BYTE																			
xClear	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
xInitDone	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
xPushDone	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
xPopDone	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
xClearDone	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
xEmpty	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

NOTE

◆ Create an FIFO queue, with the queue initialization, entering, exiting, and clearing functions. Complete execution of trigger at the rising edge within one period, and output the corresponding flag. If the output fails, check the input parameters.

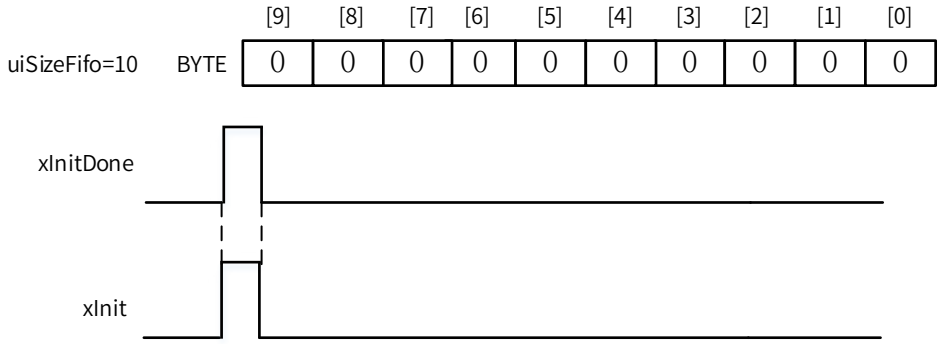
■ Program example

This instruction copies data from array ary1 to array ary2.

The procedure for using the FIFO function block is as follows:

1) Initialize a queue to apply for queue cache:

Trigger xInit to initialize a FIFO queue. This creates a cache space for the FIFO queue. If successful, xInit-Done will be set to true within one period. In this example, a 10-byte cache space is created successfully in the PLC memory. (Initialization is required only once after power-on.)

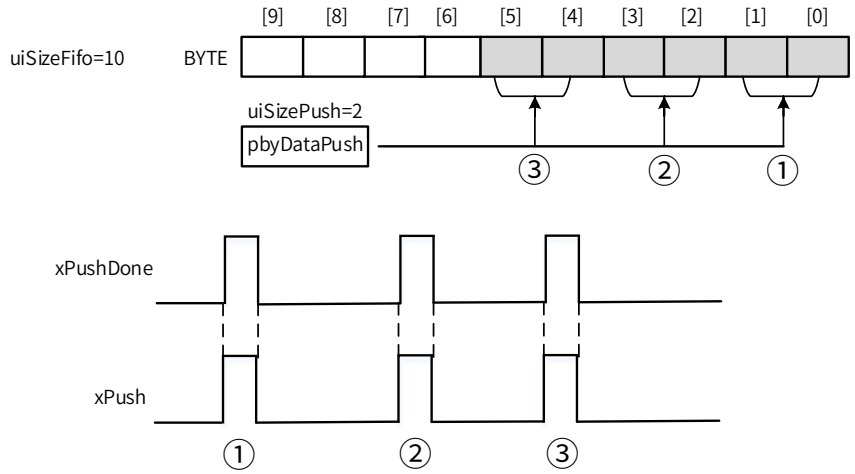


NOTE

◆ A queue needs to be initialized only once. Initializing a queue multiple times may cause memory fragments, which affects follow-up application for memory. The size of the memory to apply for should be less than 1024 bytes.

2) Push data onto the stack:

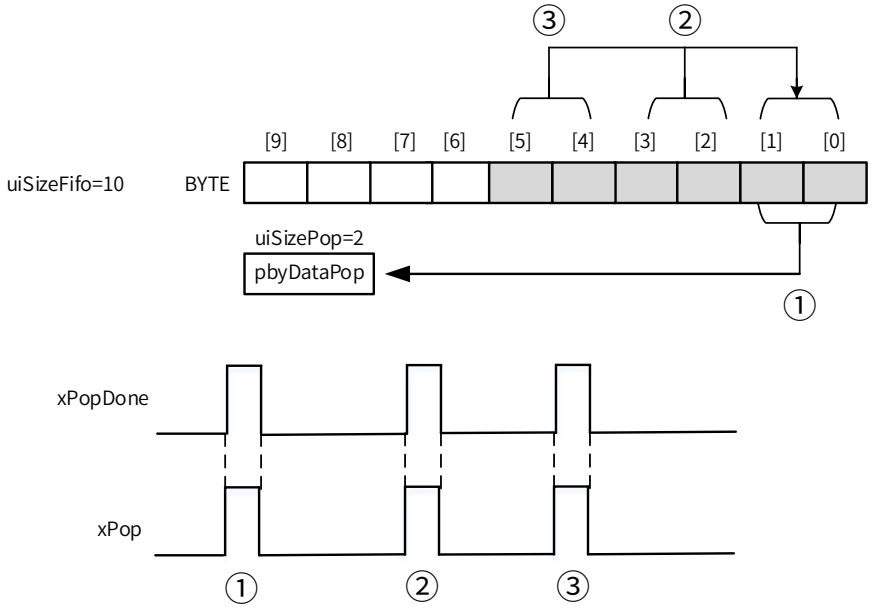
Trigger xPush to push the data pointed to by pbyDataPush into the queue, with a size of uiSizePush bytes. If successful, xPushDone will be set to true within one period.



In this example, uiSizePush is set to 2, so every time xPush is triggered, it writes two bytes of data into the queue.

3) Pop data from the stack:

Trigger xPop to retrieve uiSizePop bytes of data from the FIFO cache queue and store the bytes in the data pointed to by pbyDataPop. If successful, xPopDone will be set to true within one period.



In this example, uiSizePop is set to 2, so every time xPop is triggered, it reads two bytes of data from the beginning of the data queue into the address specified by pbyDataPop, and two bytes in the queue move to the right.

4) Clear the queue:

Trigger xClear to clear the queue as needed. If successful, xClearDone will be set to true within one period.

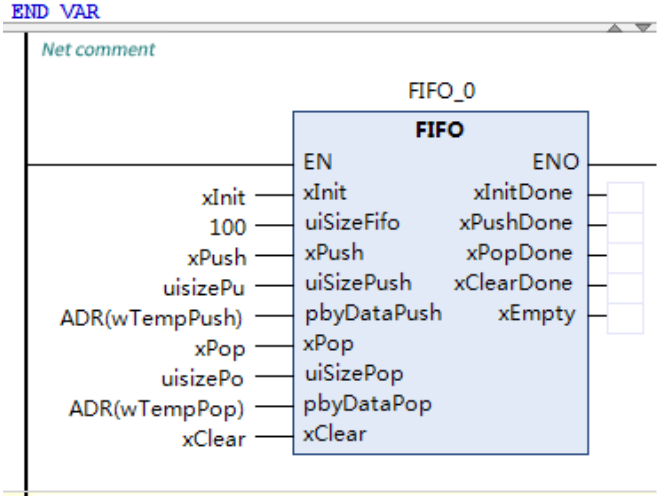
NOTE

◆ uiSizePush must be equal to the size specified by uiSizePop. When xEmpty is true, the queue has no data, and xPop cannot be performed.

■ Program example

LD

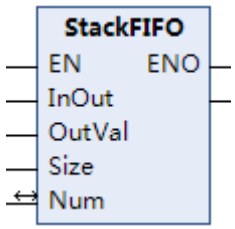
```
PROGRAM MC_Axis
VAR
    FIFO_0: FIFO;
    xInit: BOOL;// Initialize buffer allocation.
    uiSiaeFifo:UINT;//Buffer size
    xPush:BOOL;// Enter a queue each time when data is pushed onto the stack.
    xPop:BOOL;// Exit a queue each time when data is popped from the stack.
    xClear:BOOL;//Clear a queue.
    wTempPush:WORD;// Value for each data pushing onto the stack
    wTempPop:WORD;// Value for each data popping from the stack
    uisizePu: UINT := 2;// Two bytes for each data pushed onto the stack
    uisizePo: UINT := 2;// Two bytes for each data popped from the stack
END VAR
```



4.4. 3 StackFIFO

This instruction remove the bottom value from a stack.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
StackFIFO	First in first out	FC		StackFIFO (InOut :=, OutVal :=, Size :=, Num :=,);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
OutVal	Output value	-	Depends on data types	-	Value output from stack
Size	Number of stack elements	UINT	Depends on data types	1	Number of stack array elements

In-out variables

In-Out Variable	Name	Data Type	Value Range	Initial Value	Description
InOut[] (array)	Stack array	-	Depends on data types	-	Array that functions as stack

In-Out Variable	Name	Data Type	Value Range	Initial Value	Description
Num	Number of elements stored in stack	UINT	Depends on data types	-	Number of elements stored in stack

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
InOut[]	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√
	An enumerator, a structure, or a structure member can also be specified.																			
OutVal	Array with the same data type as the elements of InOut[]																			
Size	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-
Num	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-

■ Function

This instruction assumes that the InOut[] array containing Num elements is the stack array. The instruction removes the value of the least-significant bit InOut[0] from the stack and assigns it to OutVal.

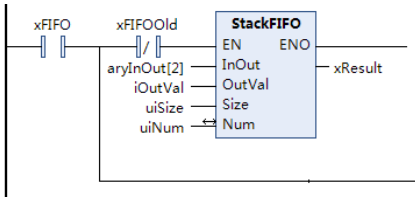
Then, all Num − 1 elements from InOut[1] are shifted to the next lower element in the stack array.

Size specifies the number of elements to be used as a stack in InOut[].

■ Program example

Item	LD	ST
Defined variable		
Program		

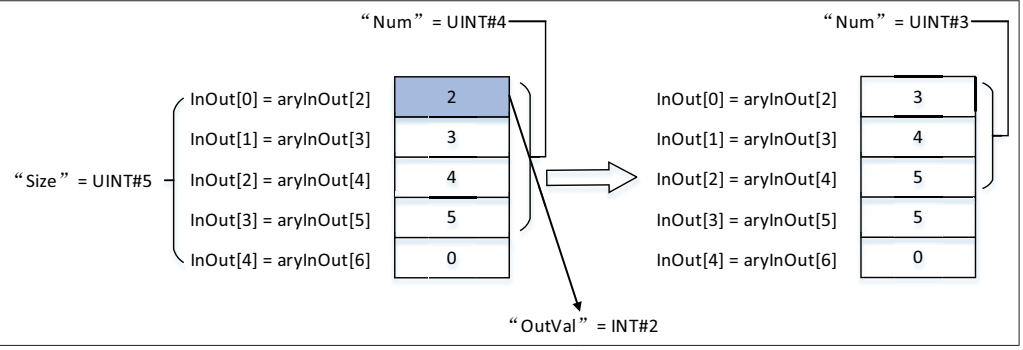
```
VAR
    aryInOut    : ARRAY [1..10] OF INT := [1,2,3,4,5,5(0)];
    iOutVal     : INT;
    uiSize      : UINT  := 5;
    uiNum       : UINT  := 4;
    xFIFO       : BOOL;
    xFIFOOld    : BOOL;
    xResult     : BOOL;
END_VAR
```




```
IF xFIFO AND NOT xFIFOOld  THEN

    xResult := StackFIFO(InOut := aryInOut[2],
            OutVal := iOutVal,
            Size := uiSize,
            Num := uiNum
            );
END_IF

xFIFOOld := xFIFO;
```

Item	LD	ST	
Running result	aryInOut	ARRAY [1..10] OF INT	
	aryInOut[1]	INT	1
	aryInOut[2]	INT	3
	aryInOut[3]	INT	4
	aryInOut[4]	INT	5
	aryInOut[5]	INT	5
	aryInOut[6]	INT	0
	aryInOut[7]	INT	0
	aryInOut[8]	INT	0
	aryInOut[9]	INT	0
	aryInOut[10]	INT	0
	iOutVal	INT	2
	uiSize	UINT	5
	uiNum	UINT	3
	xFIFO	BOOL	TRUE
	xFIFOOld	BOOL	TRUE
	xResult	BOOL	TRUE
Work principle			

■ Precautions

When the value of Size exceeds the InOut[] array, the return value is FALSE, and InOut[], Num, and OutVal do not change.

When OutVal and InOut[] are of the String type and the number of bytes in InOut[] exceeds the value of OutVal, the return value is FALSE, and InOut[], Num, and OutVal do not change.

The data types of the OutVal and InOut[] elements must be the same. Otherwise, the program cannot run, the return value is FALSE, and InOut[], Num, and OutVal do not change.

When the value of Size is 0, the return value is TRUE, and InOut[], Num, and OutVal do not change.

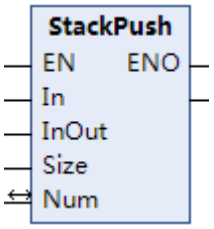
When the value of Size or Num is not 0 and Num is greater than Size, the return value is FALSE, and InOut[], Num, and OutVal do not change.

When an element in the array is transmitted to InOut[], all elements following the transmitted element are processed.

4.4. 4 StackPush

This instruction stores a value in a stack.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
StackPush	Push onto stack	FC		StackPush (In :=, InOut :=, Size :=, Num :=,);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
In	Input value	-	Depends on data types	-	Value to input to the stack
Size	Number of stack elements	UINT	Depends on data types	1	Number of stack array elements

In-out variables

In-Out Variable	Name	Data Type	Value Range	Initial Value	Description
InOut[] (array)	Stack array	-	Depends on data types	-	Array that functions as stack
Num	Number of elements stored in stack	UINT	Depends on data types	0	Number of elements stored in stack

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√
	An enumerator, a structure, or a structure member can also be specified.																			
InOut[]	Array with the same data type as elements of In																			
Size	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-
Num	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-

■ Function

1) This instruction assumes that the InOut[] array containing Num elements is the stack array. Input value In is written to the next element, InOut[Num]. Then, Num is incremented by 1.

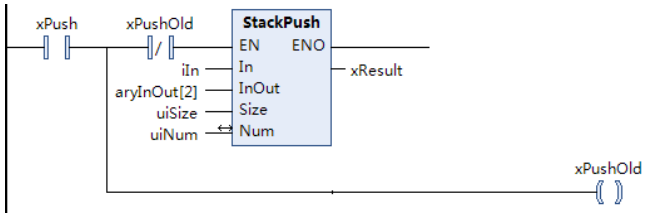
2) Size specifies the number of elements to be used as a stack in InOut[].

■ Program example

<table>
<tr><th>Item</th><th>LD</th><th>ST</th></tr>
<tr><td>Defined variable</td><td colspan="2">

```
VAR
    iIn         : INT   := 10;
    aryInOut    : ARRAY [1..10] OF INT;
    uiSize      : UINT  := 5;
    uiNum       : UINT  := 1;
    xPush       : BOOL;
    xPushOld    : BOOL;
    xResult     : BOOL;

END_VAR
```

</td></tr>
<tr><td>Program</td><td> 
</td><td>

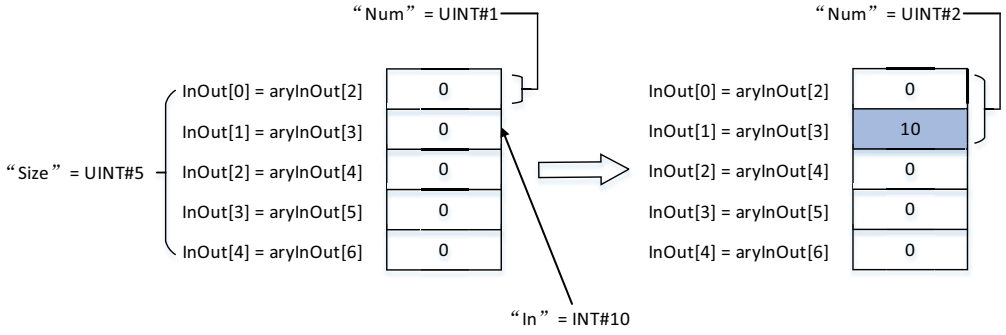
```
IF xPush AND NOT xPushOld THEN

    xResult := StackPush(In := iIn,
               InOut := aryInOut[2],
               Size := uiSize,
               Num := uiNum
               );
END_IF

xPushOld := xPush;
```

</td></tr>
<tr><td>Running result</td><td colspan="2">

iIn	INT	10
aryInOut	ARRAY [1..10] OF INT	
aryInOut[1]	INT	0
aryInOut[2]	INT	0
aryInOut[3]	INT	10
aryInOut[4]	INT	0
aryInOut[5]	INT	0
aryInOut[6]	INT	0
aryInOut[7]	INT	0
aryInOut[8]	INT	0
aryInOut[9]	INT	0
aryInOut[10]	INT	0
uiSize	UINT	5
uiNum	UINT	2
xPush	BOOL	TRUE
xPushOld	BOOL	TRUE
xResult	BOOL	TRUE

</td></tr>
<tr><td>Work principle</td><td colspan="2"> 
</td></tr>
</table>

■ Precautions

· When the value of Size exceeds the InOut[] array, the return value is FALSE and InOut[] does not change.

· When In and InOut[] are of the String type and the number of bytes in In exceeds the value of InOut[], the return value is FALSE and InOut[] does not change.

· When the data types of In and InOut[] are different, an error occurs during compilation.

· When the value of Size is 0, InOut[] and Num do not change and the return value is TRUE.

· When the value of Size is 0 and Num is greater than or equal to Size, the program cannot run and the return value is FALSE.

· When an element in the array is transmitted to InOut[], all elements following the transmitted element are processed.

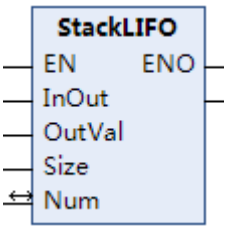
• The input parameter transmitted to In must be a variable. If a constant is transmitted, an error occurs during compilation.

• When In is an enumeration, no enumerator can be directly transmitted. If an enumerator is directly transmitted, an error occurs during compilation.

4.4. 5 StackLIFO

This instruction removes the top value from a stack.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
StackLIFO	Last in first out	FC		StackLIFO (InOut :=, OutVal :=, Size :=, Num :=,);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
OutVal	Output value	-	Depends on data types	-	Value output from stack
Size	Number of stack elements	UINT	Depends on data types	1	Number of stack array elements

In-out variables

In-Out Variable	Name	Data Type	Value Range	Initial Value	Description
InOut[] (array)	Stack array	-	Depends on data types	-	Array that functions as stack
Num	Number of elements stored in stack	UINT	Depends on data types	-	Number of elements stored in stack

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
InOut[]	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√
	An enumerator, a structure, or a structure member can also be specified.																			
OutVal	Array with the same data type as the elements of InOut[]																			
Size	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-
Num	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-

■ Function

1) This instruction assumes that the InOut[] array containing Num elements is the stack array. The instruction removes the value of the most-significant bit InOut[Num-1] from the stack and assigns it to OutVal.

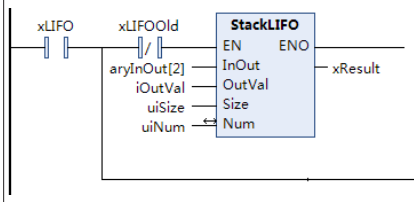
2) Num is decremented by 1.

3) Size specifies the number of elements to be used as a stack in InOut[].

■ Program example

Item	LD	ST
Defined variable		
Program		

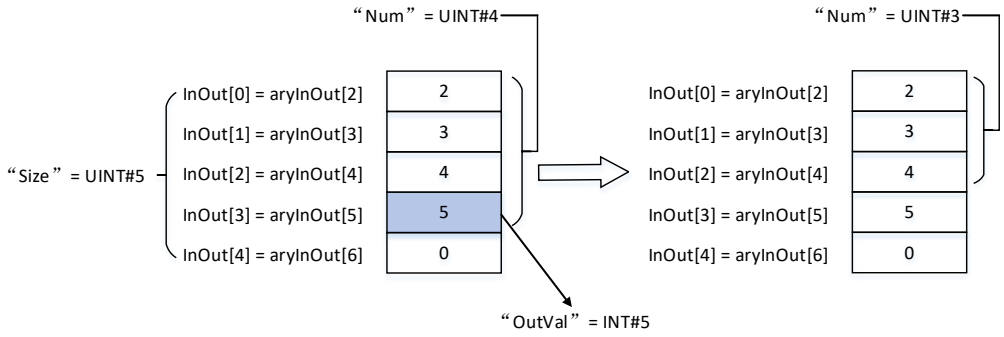
```
VAR
    aryInOut    : ARRAY [1..10] OF INT := [1,2,3,4,5,5(0)];
    iOutVal     : INT;
    uiSize      : UINT  := 5;
    uiNum       : UINT  := 4;
    xLIFO       : BOOL;
    xLIFOOld    : BOOL;
    xResult     : BOOL;
END_VAR
```




```
IF xLIFO AND NOT xLIFOOld  THEN

    xResult := StackLIFO(InOut := aryInOut[2],
            OutVal := iOutVal,
            Size := uiSize,
            Num := uiNum
            );
END_IF

xLIFOOld := xLIFO;
```

Item	LD	ST	
Running result	aryInOut	ARRAY [1..10] OF INT	
	aryInOut[1]	INT	1
	aryInOut[2]	INT	2
	aryInOut[3]	INT	3
	aryInOut[4]	INT	4
	aryInOut[5]	INT	5
	aryInOut[6]	INT	0
	aryInOut[7]	INT	0
	aryInOut[8]	INT	0
	aryInOut[9]	INT	0
	aryInOut[10]	INT	0
	iOutVal	INT	5
	uiSize	UINT	5
	uiNum	UINT	3
	xLIFO	BOOL	TRUE
	xLIFOOld	BOOL	TRUE
	xResult	BOOL	TRUE
Work principle			

■ Precautions

· When the value of Size exceeds the InOut[] array, the return value is FALSE, and InOut[], Num, and OutVal do not change.

· When OutVal and InOut[] are of the String type and the number of bytes in InOut[] exceeds the value of OutVal, the return value is FALSE, and InOut[], Num, and OutVal do not change.

· The data types of the OutVal and InOut[] elements must be the same. Otherwise, the program cannot run, the return value is FALSE, and InOut[], Num, and OutVal do not change.

· When the value of Size is 0, the return value is TRUE, and InOut[], Num, and OutVal do not change.

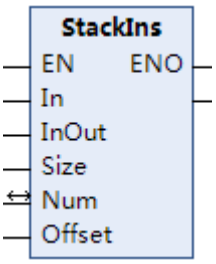
· When the value of Size or Num is not 0 and Num is greater than Size, the return value is FALSE, and InOut[], Num, and OutVal do not change.

· When an element in the array is transmitted to InOut[], all elements following the transmitted element are processed.

4.4. 6 StackIns

This instruction inserts a value at any position in a stack.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
StackIns	Insert into stack	FC		StackIns (In :=, InOut :=, Size :=, Num :=, Offset :=);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
In	Input value	-	Depends on data types	-	Value to input to the stack
Size	Number of stack elements	UINT	Depends on data types	1	Number of stack array elements
Offset	Offset	UINT	Depends on data types	0	Position in stack at which to insert In

In-out variables

In-Out Variable	Name	Data Type	Value Range	Initial Value	Description
InOut[] (array)	Stack array	-	Depends on data types	-	Array that functions as stack
Num	Number of elements stored in stack	UINT	Depends on data types	-	Number of elements stored in stack

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√
	An enumerator, a structure, or a structure member can also be specified.																			
InOut[]	Array with the same data type as the elements of InOut[]																			
Size	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-
Num	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-
Offset	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-

■ Function

1) This instruction assumes that the InOut[] array containing Num elements is the stack array. The input value In is inserted at the position specified by InOut[Offset] in the stack, with Offset specified.

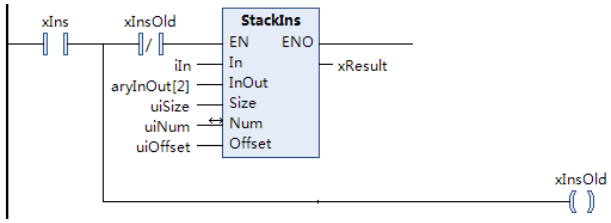
2) All higher elements, that is, InOut[Offset] to InOut[Num − 1], are moved to the next higher element in the stack array. Then, Num is incremented by 1.

3) Size specifies the number of elements to be used as a stack in InOut[].

■ Program example

<table>
<tr><th>Item</th><th>LD</th><th>ST</th></tr>
<tr><td>Defined variable</td><td colspan="2">

```
VAR
    iIn         : INT   := 10;
    aryInOut    : ARRAY [1..10] OF INT := [1,2,3,4,5,5(0)];
    uiSize      : UINT  := 5;
    uiNum       : UINT  := 4;
    uiOffset    : UINT  := 2;
    xIns        : BOOL;
    xInsOld     : BOOL;
    xResult     : BOOL;
END_VAR
```

</td></tr>
<tr><td>Program</td><td> 
</td><td>

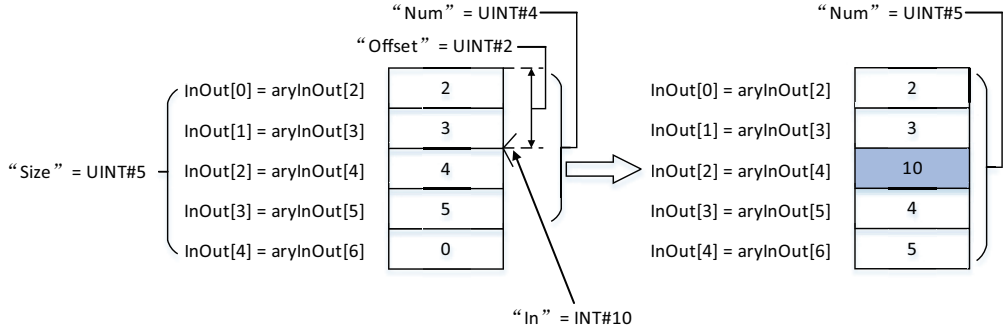
```
IF xIns AND NOT xInsOld  THEN

    xResult := StackIns(In := iIn,
               InOut := aryInOut[2],
               Size := uiSize,
               Num := uiNum,
               Offset := uiOffset
               );
END_IF

xInsOld := xIns;
```

</td></tr>
<tr><td>Running result</td><td colspan="2">

iIn	INT	10
aryInOut	ARRAY [1..10] OF INT	
aryInOut[1]	INT	1
aryInOut[2]	INT	2
aryInOut[3]	INT	3
aryInOut[4]	INT	10
aryInOut[5]	INT	4
aryInOut[6]	INT	5
aryInOut[7]	INT	0
aryInOut[8]	INT	0
aryInOut[9]	INT	0
aryInOut[10]	INT	0
uiSize	UINT	5
uiNum	UINT	5
uiOffset	UINT	2
xIns	BOOL	TRUE
xInsOld	BOOL	TRUE
xResult	BOOL	TRUE

</td></tr>
<tr><td>Work principle</td><td colspan="2"> 
</td></tr>
</table>

■ Precautions

· When the value of Size exceeds the InOut[] array, the return value is FALSE, and InOut[] and Num do not change.

· When In and InOut[] are of the String type and the number of bytes in In exceeds the value of InOut[], the return value is FALSE and InOut[] and Num do not change.

· When the data types of In and InOut[] are different, an error occurs during compilation.

· When the value of Size is 0, the return value is TRUE and InOut[] and Num do not change.

· When the value of Size is 0 and Num is not greater than or equal to Offset and not less than Size, the program cannot run, the return value is FALSE, and InOut[] and Num do not change.

· When an element in the array is transmitted to InOut[], all elements following the transmitted element are processed.

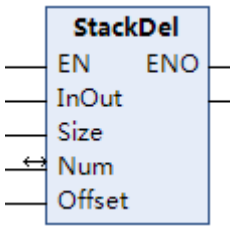
· The input parameter transmitted to In must be a variable. If a constant is transmitted, an error occurs during compilation.

· When In is an enumeration, no enumerator can be directly transmitted. If an enumerator is directly transmitted, an error occurs during compilation.

4.4. 7 StackDel

This instruction deletes a value from any position in a stack.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
StackDel	Delete from stack	FC		StackDel (InOut :=, Size :=, Num :=, Offset :=,);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
Size	Number of stack elements	UINT	Depends on data types	1	Number of stack array elements
Offset	Offset	UINT	Depends on data types	0	Offset of an element to delete from a stack

In-out variables

In-Out Variable	Name	Data Type	Value Range	Initial Value	Description
InOut[] (array)	Stack array	-	Depends on data types	-	Array that functions as stack
Num	Number of elements stored in stack	UINT	Depends on data types	-	Number of elements stored in stack

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
InOut[]	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√
	An enumerator, a structure, or a structure member can also be specified.																			
Size	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-
Num	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-
Offset	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-

■ Function

1) This instruction assumes that the InOut[] array containing Num elements is the stack array. The value is deleted from the position specified by InOut[Offset] in the stack, with Offset specified.

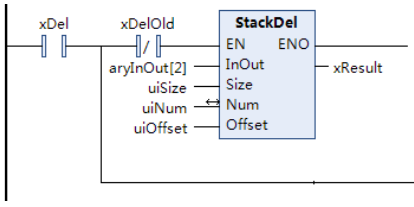
2) All higher elements, that is, InOut[Offset+1] to InOut[Num − 1], are moved to the next lower element in the stack array. Num is decremented by 1.

3) Size specifies the number of elements to be used as a stack in InOut[].

■ Program example

Item	LD	ST
Defined variable		
Program		

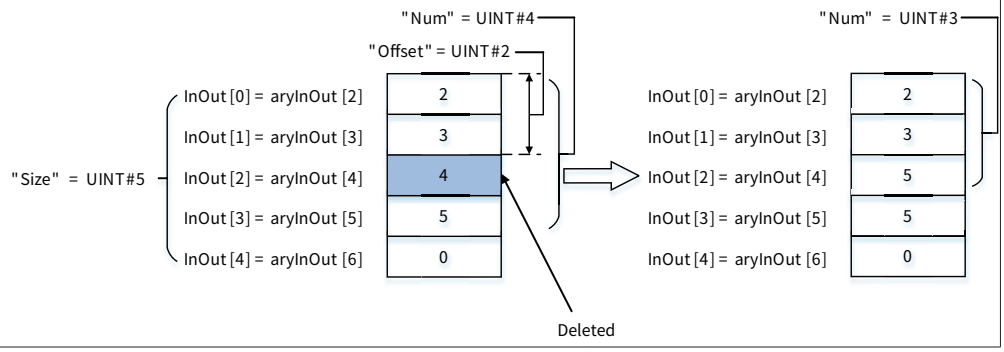
```
VAR
    aryInOut    : ARRAY [1..10] OF INT := [1,2,3,4,5,5(0)];
    uiSize      : UINT  := 5;
    uiNum       : UINT  := 4;
    uiOffset    : UINT  := 2;
    xDel        : BOOL;
    xDelOld     : BOOL;
    xResult     : BOOL;
END_VAR
```




```
IF xDel AND NOT xDelOld  THEN

    xResult := StackDel(InOut := aryInOut[2],
            Size := uiSize,
            Num := uiNum,
            Offset := uiOffset,
            );
END_IF

xDelOld := xDel;
```

Item	LD	ST	
Running result	aryInOut	ARRAY [1..10] OF INT	
	aryInOut[1]	INT	1
	aryInOut[2]	INT	2
	aryInOut[3]	INT	3
	aryInOut[4]	INT	5
	aryInOut[5]	INT	5
	aryInOut[6]	INT	0
	aryInOut[7]	INT	0
	aryInOut[8]	INT	0
	aryInOut[9]	INT	0
	aryInOut[10]	INT	0
	uiSize	UINT	5
	uiNum	UINT	3
	uiOffset	UINT	2
	xDel	BOOL	TRUE
	xDelOld	BOOL	TRUE
	xResult	BOOL	TRUE
Work principle			

■ Precautions

· When the value of Size exceeds the InOut[] array, the return value is FALSE, and InOut[] and Num do not change.

· When the value of Size is 0, the return value is TRUE and InOut[] and Num do not change.

· When the value of Size or Num is 0 and Num is not greater than Offset and not less than or equal to Size, the program cannot run, the return value is FALSE, and InOut[] and Num do not change.

· When an element in the array is transmitted to InOut[], all elements following the transmitted element are processed.

4. 5 Table and Range Instructions

4.5. 1 Instruction List

Instruction Category	Name	FB/FC	Function

Table and range instructions	BZAND_TAB	FC	Dead zone control
	MEAN_TAB	FC	Average calculation
	ZONE_TAB	FC	Regional control
	ZRST_TAB	FC	Reset all
	SCL_TAB	FC	Fixed coordinates (different point coordinate)
	SORT_TAB	FC	Data sorting
	RAMP_TAB	FB	Ramp instruction
	WSUM_TAB	FC	Calculate the total value
	RecSearch	FC	Record search
	RecRangeSearch	FC	Range record search
	RecSort	FB	Record sort
	RecNum	FC	Get number of records
	RecMax	FC	Maximum record search
	RecMin	FC	Minimum record search

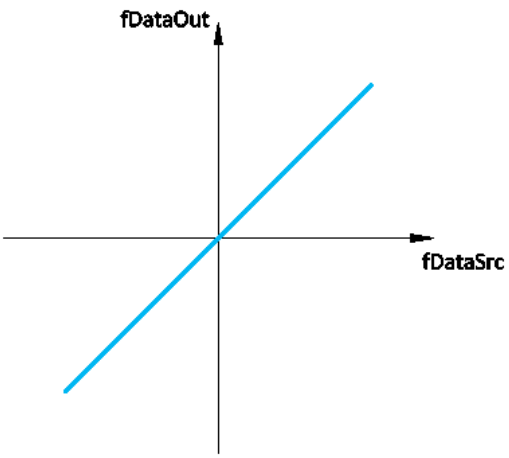
4.5. 2 BZAND_TAB

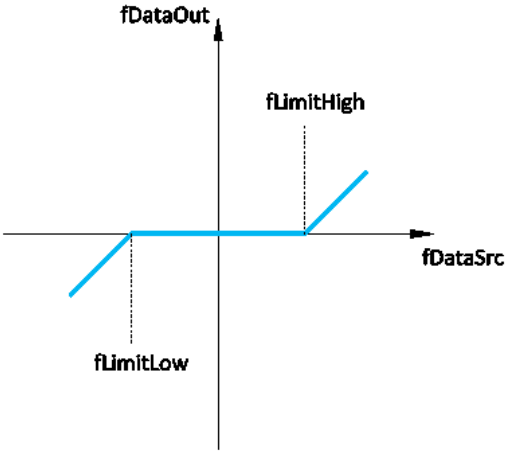
This instruction controls an output value based on whether the input value is within the specified dead zone range (defined by upper and lower limits). The dead zone range is set in fLimitLow and fLimitHigh so that the input value fDataSrc is output to fDataOut outside the dead zone range.

When fLimitLow is greater than fDataSrc, fDataSrc minus fLimitLow is equal to fDataOut.

When fLimitHigh is less than fDataSrc, fDataSrc minus fLimitHigh is equal to fDataOut.

When fDataSrc is greater than or equal to fLimitLow and less than or equal to fLimitHigh, fDataOut is less than or equal to 0.





■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
BZAND_TAB	Dead zone control	FC	BZAND_TAB EN ENO fLimitHigh fLimitLow fDataOut fDataSrc	BZAND_TAB(fLimitHigh:= , fLimitLow:= , fDataSrc:= , fDataOut=>)

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
fLimitHigh	Dead zone upper limit	REAL	-	0	Dead zone upper limit
fLimitLow	Dead zone lower limit	REAL	-	0	Dead zone lower limit
fDataSrc	Source data	REAL	-	0	Input value subject to dead zone control

Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
fDataOut	Target data	REAL	-	-	Output value from the dead zone

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
fLim-itHigh	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-
fLimitLow	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-
fDataSrc	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-
fDataOut	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-

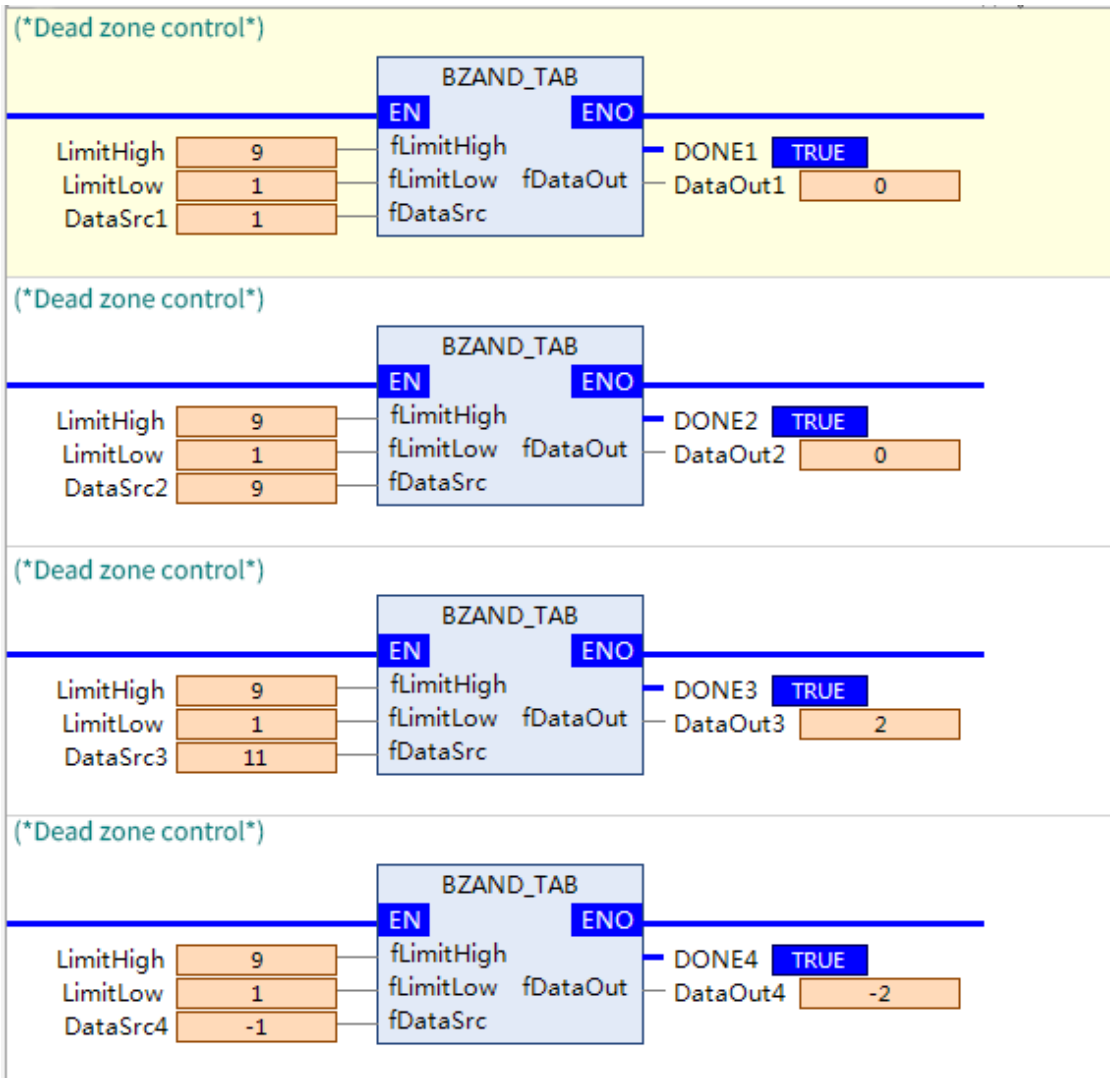
■ Program example

ST

Expression	Type	Value
LimitHigh	REAL	9
LimitLow	REAL	1
DataSrc1	REAL	1
DataSrc2	REAL	9
DataSrc3	REAL	11
DataSrc4	REAL	-1

```
1  BZAND_TAB(
2  fLimitHigh:=LimitHigh 9 ,
3  fLimitLow:= LimitLow 1,
4  fDataSrc:= DataSrc1 1,
5  fDataOut=> DataOut1 0 );
6  BZAND_TAB(
7  fLimitHigh:=LimitHigh 9 ,
8  fLimitLow:= LimitLow 1,
9  fDataSrc:= DataSrc2 9,
10 fDataOut=> DataOut2 0 );
11 BZAND_TAB(
12 fLimitHigh:=LimitHigh 9 ,
13 fLimitLow:= LimitLow 1,
14 fDataSrc:= DataSrc3 11,
15 fDataOut=> DataOut3 2 );
16 BZAND_TAB(
17 fLimitHigh:=LimitHigh 9 ,
18 fLimitLow:= LimitLow 1,
19 fDataSrc:= DataSrc4 -1,
20 fDataOut=> DataOut4 -2 );
21 RETURN
```

LD



4.5. 3 MEAN_TAB

This instruction calculates the average value of specified data (uiSize numbers starting from pfDataSrc), storing the result in fDataOut.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
MEAN_TAB	Average calculation	FC	MEAN_TAB EN ENO pfDataSrc uiSize fDataOut	MEAN_TAB(pfDataSrc:= , uiSize:= , fDataOut=>);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
pfDataSrc	Source data	POINTER TO REAL	-	-	Input data address
uiSize	Data length	UINT	1 to 1024	0	Data length

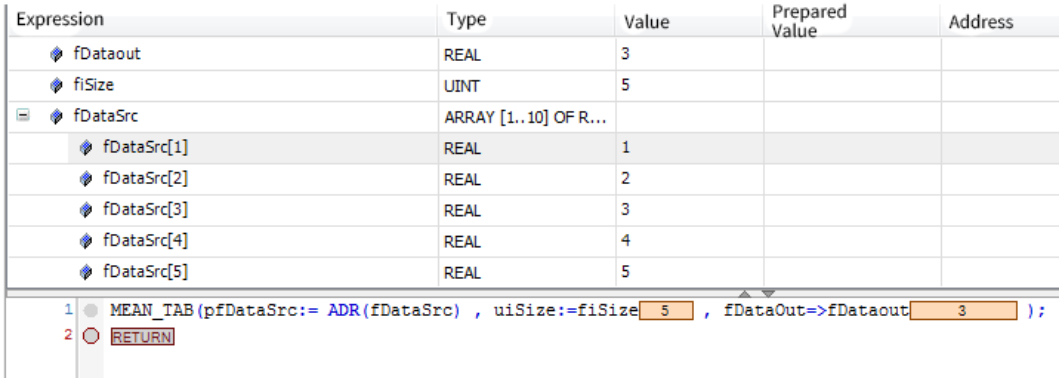
Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
fDataOut	Target data	REAL	-	-	Average value output

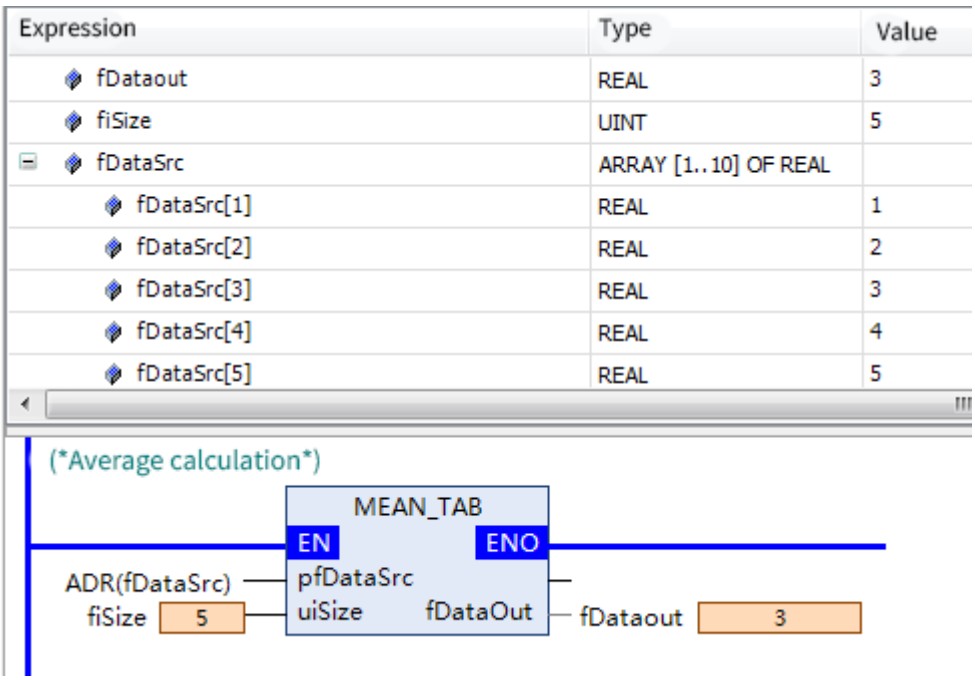
	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
pf-DataSrc	POINTER TO REAL																			
uiSize	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-
fDataOut	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-

■ Program example

ST



LD



4.5.4 ZONE_TAB

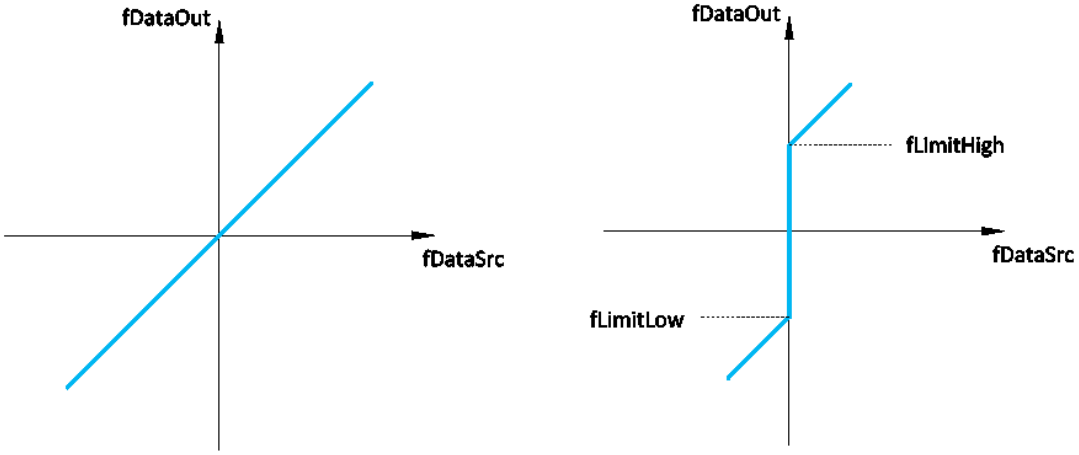
This instruction controls an output value (adding the deviation to the input value) by using the specified deviation based on whether the input value is positive or negative.

The execution result is saved to fDataOut based on the sign of the input value fDataSrc as well as fLimitHigh or fLimitLow.

When fDataSrc is less than 0, fDataSrc plus fLimitLow is greater than or equal to fDataOut.

When fDataSrc is greater than 0, fDataSrc plus fLimitHigh is greater than or equal to fDataOut.

When fDataSrc is 0, fDataOut is less than or equal to 0.



■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
ZONE_TAB	Regional control	FC	ZONE_TAB EN ENO fLimitHigh fLimitLow fDataOut fDataSrc	ZONE_TAB(fLimitHigh:= , fLimitLow:= , fDataSrc:= , fDataOut=>)

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
fLimitHigh	Positive deviation	REAL	-	0	Positive deviation, with the deviation value of the input value
fLimitLow	Negative deviation	REAL	-	0	Negative deviation, with the deviation value of the input value
fDataSrc	Source data	REAL	-	0	Input value

Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
fDataOut	Target data	REAL	-	0	Output value

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
fLimitHigh	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-
fLimitLow	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-
fDataSrc	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-
fDataOut	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-

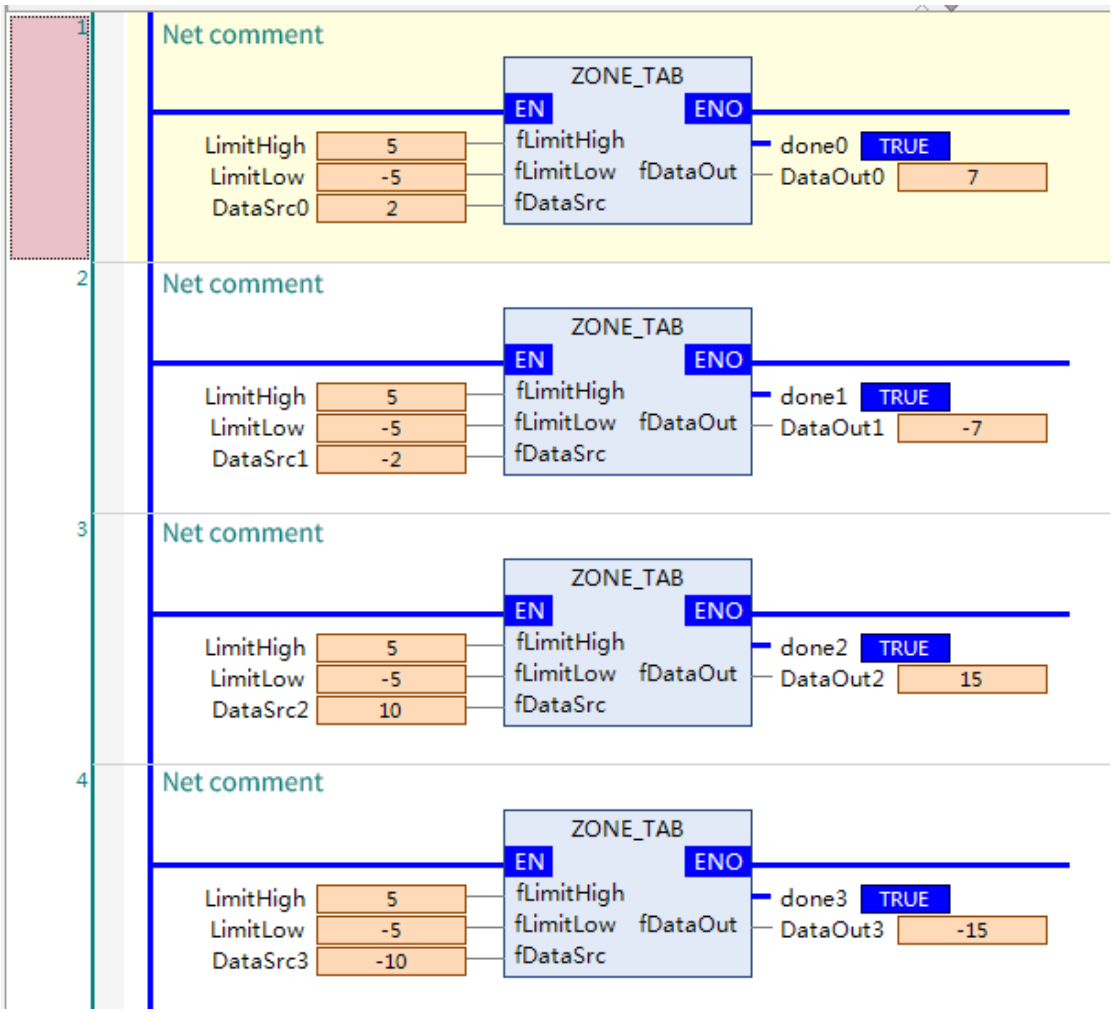
■ Program example

ST

Expression	Type	Value
LimitHigh	REAL	5
LimitLow	REAL	-5
DataSrc0	REAL	2
DataSrc1	REAL	-2
DataSrc2	REAL	10
DataSrc3	REAL	-10
DataOut0	REAL	7
DataOut1	REAL	-7
DataOut2	REAL	15

```
1  ZONE_TAB(
2  fLimitHigh:=LimitHigh 5 ,
3  fLimitLow:=LimitLow -5 ,
4  fDataSrc:=DataSrc0 2 ,
5  fDataOut=> DataOut0 7 );
6  ZONE_TAB(
7  fLimitHigh:=LimitHigh 5 ,
8  fLimitLow:=LimitLow -5 ,
9  fDataSrc:=DataSrc1 -2 ,
10 fDataOut=> DataOut1 -7 );
11 ZONE_TAB(
12 fLimitHigh:=LimitHigh 5 ,
13 fLimitLow:=LimitLow -5 ,
14 fDataSrc:=DataSrc2 10 ,
15 fDataOut=> DataOut2 15 );
16 ZONE_TAB(
17 fLimitHigh:=LimitHigh 5 ,
18 fLimitLow:=LimitLow -5 ,
19 fDataSrc:=DataSrc3 -10 ,
20 fDataOut=> DataOut3 -15 );RETURN
```

LD



4.5. 5 ZRST_TAB

This instruction resets data in a table in batches.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
ZRST_TAB	Reset all	FC	ZRST_TAB EN ENO pbyData uiSize	ZRST_TAB(pbyData:=), uiSize:=);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
pbyData	Source data	POINTER TO REAL	-	-	Head address of data
uiSize	Data length	UINT	1 to 1024	0	Data length

Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
ZRST_TAB	Return value	BOOL	[TRUE, FALSE]	FALSE	TRUE: Function execution is successful. FALSE: Function execution fails.

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
pbyData	POINTER TO REAL																			
uiSize	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-
ZRST_TAB	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

■ Program example

ST

uiSize = 10 indicates that 10 elements of the floating-point number array are reset in a batch.

```
1  ZRST_TAB(
2  pbyData:=ADR(byData[0]) ,
3  uiSize:=Size[10]*4 );
4  RETURN
```

16-bit WORD addresses %MW0 to %MW9 are reset in a batch.

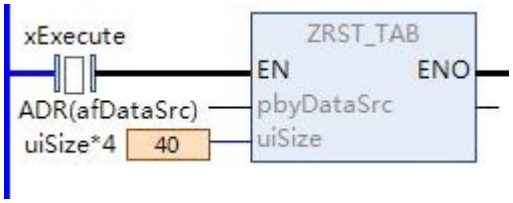
```
ZRST_TAB(
pbyData:=ADR(%MW0[0]),
uiSize:=Size[10]*2);
```

Bit elements %QX0.0 to %QX9.7 are reset in a batch. The bit element address must start from 0, and the number of bits to reset is iSize x 8.

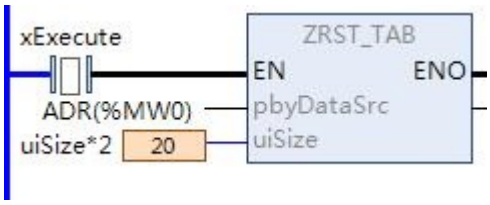
```
ZRST_TAB(
pbyData:=ADR(%QX0.0[FALSE]),
uiSize:=Size[10]);RETURN
```

LD

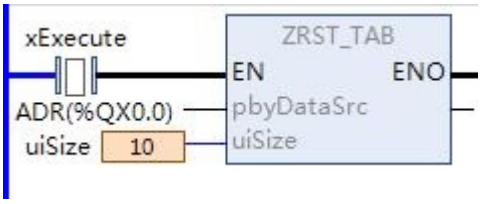
uiSize = 10 indicates that 10 elements of the floating-point number array are reset in a batch.



16-bit WORD addresses %MW0 to %MW9 are reset in a batch.



Bit elements %QX0.0 to %QX9.7 are reset in a batch. The bit element address must start from 0, and the number of bits to reset is iSize x 8.



4.5. 6 SCL_TAB

This instruction controls an output value (adding the deviation to the input value) by using the specified deviation based on whether the input value is positive or negative.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
SCL_TAB	Fixed coordinates (different point coordinate)	FC	SCL_TAB EN ENO fDataIn pfDataSrc fDataOut byMode	SCL_TAB(fDataIn:= , pfDataSrc:= , byMode:= , fDataOut=>)

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
fDataIn	Source data	REAL	-	0	Input data
uiSize	Data length	UINT	-	0	Data length
pfDataSrc	Source data	POINTER TO REAL	-	-	Table data
byMode	Mode	BYTE	1 to 2	0	1\2 (x, y) mode

1\2 (x, y) mode: 1: x1,y1;x2,y2;….xn,yn 2: x1,x2,…xn;y1,y2…yn;

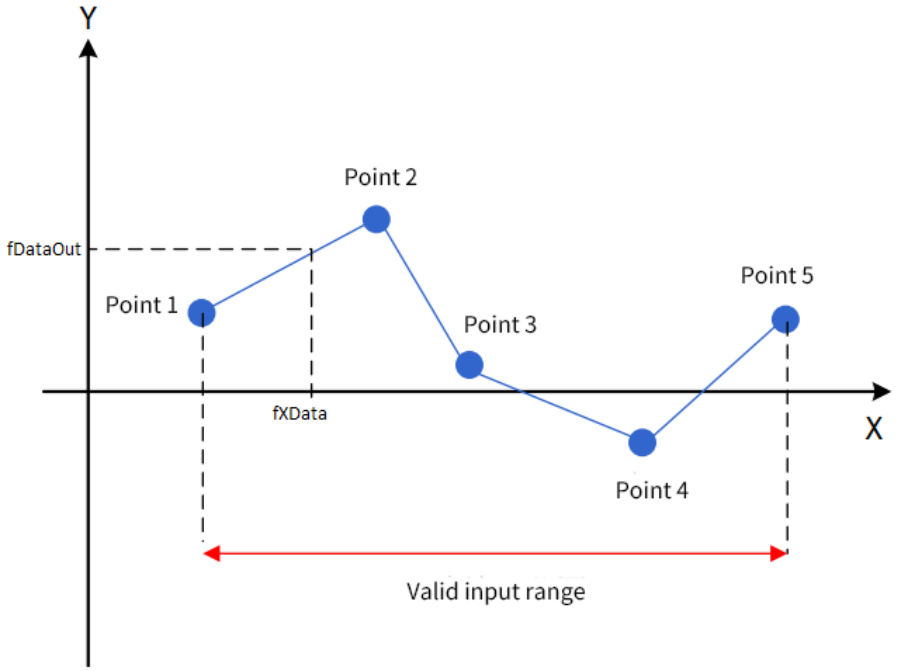
Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
fDataOut	Target data	REAL	-	0	Output value

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
fDataIn	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-
uiSize	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-
pfDataSrc	POINTER TO REAL																			
byMode	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
fDataOut	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-

■ Program example

This instruction finds the output value fDataOut corresponding to the input value fData based on the diagram determined by the table in pfDataSrc.



See the data in the following table.

Setting Item		Data Table Address
Point and Coordinates		pfDataSrc^
Point 1	X coordinate	(pfDataSrc+1)^
	Y coordinate	(pfDataSrc+2)^
Point 2	X coordinate	(pfDataSrc+3)^
	Y coordinate	(pfDataSrc+4)^
Point 3	X coordinate	(pfDataSrc+5)^
	Y coordinate	(pfDataSrc+6)^
Point 4	X coordinate	(pfDataSrc+7)^
	Y coordinate	(pfDataSrc+8)^
Point 5	X coordinate	(pfDataSrc+9)^
	Y coordinate	(pfDataSrc+10)^

The initial values for this instruction are as follows:

Variable pin definition

Variable	Name	Data Type	Initial Value
fDataIn	Input value	REAL	0
pfDataSrc	Table data	ARRAY OF REAL	0
byMode	1\2 (x, y) mode	BYTE	0
fDataOut	Output value	REAL	0

1\2 (x, y) mode: 1: x1,y1;x2,y2;….xn,yn 2: x1,x2,…xn;y1,y2…yn;

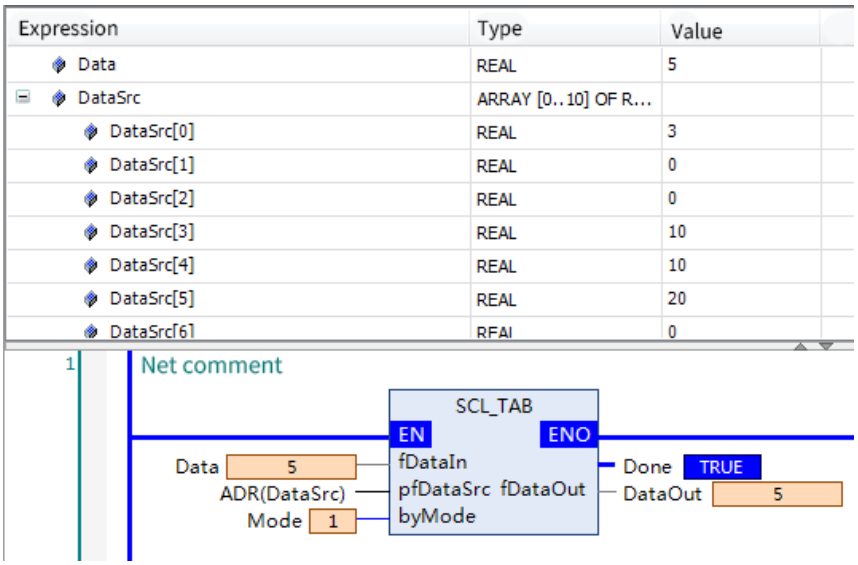
■ Program example

ST

Expression	Type	Value
Data	REAL	5
DataSrc	ARRAY [0..10] OF R...	
DataSrc[0]	REAL	3
DataSrc[1]	REAL	0
DataSrc[2]	REAL	0
DataSrc[3]	REAL	10
DataSrc[4]	REAL	10
DataSrc[5]	REAL	20
DataSrc[6]	REAL	0

```
1  SCL_TAB(
2  fDataIn:= Data 5 ,
3  pfDataSrc:=ADR(DataSrc) ,
4  byMode:= Mode 1 ,
5  fDataOut=>DataOu 5 );
6  RETURN
```

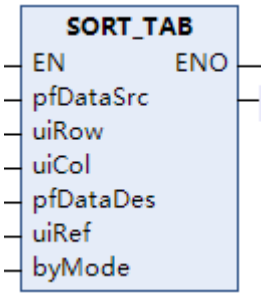
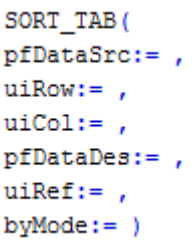
LD



4.5. 7 SORT_TAB

This instruction sorts arrays of the iRow row and iCol column based on the parameters in the uiRef column, storing the sorting result in the variable area starting from the pfDataDes unit.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
SORT_TAB	Data sorting	FC		

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
pfDataSrc	Source data	POINTER TO REAL	-	-	Data sorting
uiRow	Row	UINT	1 to 32	0	Data row
uiCol	Column	UINT	1 to 32	0	Data column
pfDataDes	Target data	POINTER TO REAL	-	-	Data output
uiRef	Row/Column	UINT	1 to 32	0	By row or column
byMode	Sorting mode	BYTE	1 to 4	0	Sorting mode

Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
SORT_TAB	Return value	BOOL	[TRUE, FALSE]	FALSE	TRUE: Function execution is successful. FALSE: Function execution fails.

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
pfDataSrc	POINTER TO REAL																			
uiRow	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-
uiCol	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-
pfDataDes	POINTER TO REAL																			
uiRef	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-
byMode	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
SORT_TAB	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

pfDataSrc is the start unit for the first variable in the first row (or first record).

uiRow is the number of rows (or records) in the array.

uiCol is the number of columns in the array, or the number of entries in each record.

pfDataDes is the start unit for storing the sorted data. The number of subsequent variable units occupied is the same as the number of array variables before the sorting.

uiRef is the number of the array row or column that serves as the reference for sorting.

byMode is the sorting mode. 1: By column in ascending order; 2: By column in descending order; 3: By row in ascending order; 4: By row in descending order

■ Program example

ST

When uiRef is 2, data sorting is shown in the following figure.

Expression	Type	Value
fDataDes[2]	REAL	1
fDataDes[3]	REAL	5
fDataDes[4]	REAL	4
fDataDes[5]	REAL	77
fDataDes[6]	REAL	82
fDataDes[7]	REAL	85
fDataDes[8]	REAL	87
fDataDes[9]	REAL	90
fDataDes[10]	REAL	89
fDataDes[11]	REAL	91
fDataDes[12]	REAL	78
fDataDes[13]	REAL	95

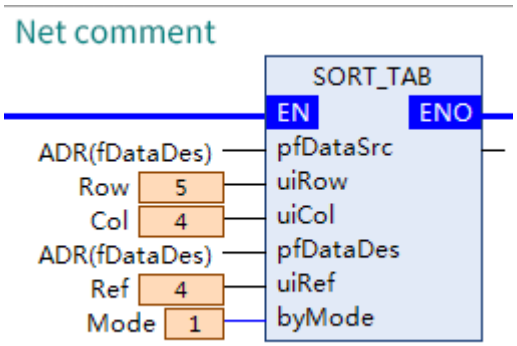
```
1  SORT_TAB(
2      pfDataSrc:= ADR(fDataSrc),
3      uiRow:=Row 5 ,
4      uiCol:= Col 4 ,
5      pfDataDes:=ADR(fDataDes) ,
6      uiRef:=Ref 2 ,
7      byMode:= Mode 1 );
8      RETURN
```

When uiRef is 4, data sorting is shown in the following figure.

Expression	Type	Value
fDataDes[12]	REAL	91
fDataDes[13]	REAL	78
fDataDes[14]	REAL	89
fDataDes[15]	REAL	75
fDataDes[16]	REAL	77
fDataDes[17]	REAL	81
fDataDes[18]	REAL	83
fDataDes[19]	REAL	88
fDataDes[20]	REAL	0
Ref	UINT	4
Mode	BYTE	1

```
1  SORT_TAB(
2      pfDataSrc:= ADR(fDataSrc),
3      uiRow:=Row 5 ,
4      uiCol:= Col 4 ,
5      pfDataDes:=ADR(fDataDes) ,
6      uiRef:=Ref 4 ,
7      byMode:= Mode 1 );
8      RETURN
```

LD



The initial values of afDataSrc are as follows.

afDataSrc	ARRAY [0..100] OF REAL	
afDataSrc[0]	REAL	1
afDataSrc[1]	REAL	2
afDataSrc[2]	REAL	3
afDataSrc[3]	REAL	4
afDataSrc[4]	REAL	5
afDataSrc[5]	REAL	85
afDataSrc[6]	REAL	82
afDataSrc[7]	REAL	77
afDataSrc[8]	REAL	90
afDataSrc[9]	REAL	87
afDataSrc[10]	REAL	78
afDataSrc[11]	REAL	91
afDataSrc[12]	REAL	89
afDataSrc[13]	REAL	81
afDataSrc[14]	REAL	95
afDataSrc[15]	REAL	83
afDataSrc[16]	REAL	81
afDataSrc[17]	REAL	88
afDataSrc[18]	REAL	75
afDataSrc[19]	REAL	77
afDataSrc[20]	REAL	0

The initial values of afDataDes are as follows.

afDataDes	ARRAY [0..100] OF REAL	
afDataDes[0]	REAL	0
afDataDes[1]	REAL	0
afDataDes[2]	REAL	0
afDataDes[3]	REAL	0
afDataDes[4]	REAL	0
afDataDes[5]	REAL	0
afDataDes[6]	REAL	0
afDataDes[7]	REAL	0
afDataDes[8]	REAL	0
afDataDes[9]	REAL	0
afDataDes[10]	REAL	0
afDataDes[11]	REAL	0
afDataDes[12]	REAL	0
afDataDes[13]	REAL	0
afDataDes[14]	REAL	0
afDataDes[15]	REAL	0
afDataDes[16]	REAL	0
afDataDes[17]	REAL	0

The equivalent table for preceding values of afDataSrc is as follows.

Line No. / Line No.	1	2	3	4
	Student ID	Chinese	Mathematics	Physics
1	fDataSrc[0] 1	fDataSrc[5] 85	fDataSrc[10] 78	fDataSrc[15] 83
2	fDataSrc[1] 2	fDataSrc[6] 82	fDataSrc[11] 91	fDataSrc[16] 81
3	fDataSrc[2] 3	fDataSrc[7] 77	fDataSrc[12] 89	fDataSrc[17] 88
4	fDataSrc[3] 4	fDataSrc[8] 90	fDataSrc[13] 81	fDataSrc[18] 75
5	fDataSrc[4] 5	fDataSrc[9] 87	fDataSrc[14] 95	fDataSrc[19] 77

When uiColRef is 2 and xExecute is TRUE, the sorting result is as follows.

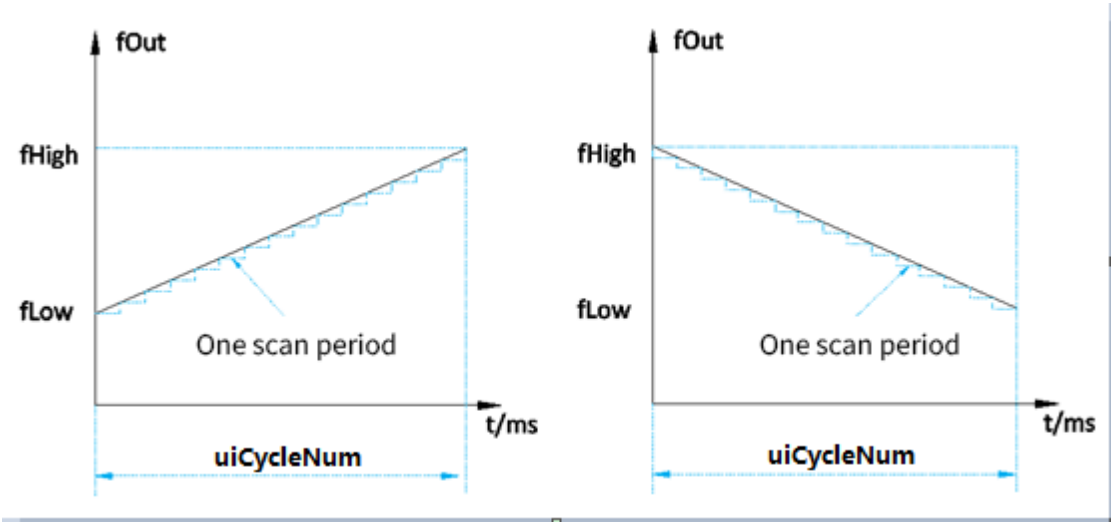
Line No. / Line No.	1	2	3	4
	Student ID	Chinese	Mathematics	Physics
1	fDataDes[0] 3	fDataDes[5] 77	fDataDes[10] 89	fDataDes[15] 88
2	fDataDes[1] 2	fDataDes[6] 82	fDataDes[11] 91	fDataDes[16] 81
3	fDataDes[2] 1	fDataDes[7] 85	fDataDes[12] 78	fDataDes[17] 83
4	fDataDes[3] 5	fDataDes[8] 87	fDataDes[13] 95	fDataDes[18] 77
5	fDataDes[4] 4	fDataDes[9] 90	fDataDes[14] 81	fDataDes[19] 75

When uiColRef is 4 and xExecute is TRUE, the sorting result is as follows.

Line No. / Line No.	1	2	3	4
	Student ID	Chinese	Mathematics	Physics
1	fDataDes[0] 4	fDataDes[5] 90	fDataDes[10] 81	fDataDes[15] 75
2	fDataDes[1] 5	fDataDes[6] 87	fDataDes[11] 95	fDataDes[16] 77
3	fDataDes[2] 2	fDataDes[7] 82	fDataDes[12] 91	fDataDes[17] 81
4	fDataDes[3] 1	fDataDes[8] 85	fDataDes[13] 78	fDataDes[18] 83
5	fDataDes[4] 3	fDataDes[9] 77	fDataDes[14] 89	fDataDes[19] 88

4.5. 8 RAMP_TAB

This instruction changes the output value from the minimum value to the maximum value at a specified interval, and vice versa. This instruction aims to perform linear interpolation between two given data entries at a specified time interval, so as to output intermediate values in a sequential manner based on the scanning execution time, until the end value is reached.



■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
RAMP_TAB	Ramp instruction	FB	RAMP_TAB xEnable xValid fStart xBusy fEnd xError xPeriodic dwErrorID uiCycleNum fValue	RAMP_TAB(xEnable:= , fStart:= , fEnd:= , xPeriodic:= , uiCycleNum:= , xValid=> , xBusy=> , xError=> , dwErrorID=> , fValue=>);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
xEnable	Executed	BOOL	[TRUE, FALSE]	FALSE	Input high level to enable the function block
fStart	Start value	REAL	-	0	Start value
fEnd	End value	REAL	-	0	End value
xPeriodic	Cyclic output	BOOL	[TRUE, FALSE]	FALSE	FALSE: Executing once TRUE: Cyclic execution
uiCycleNum	Number of completed periods	UINT	-	0	Number of completed periods

Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
xValid	Compliant	BOOL	[TRUE, FALSE]	FALSE	TRUE: Function execution is successful. FALSE: Function execution fails.

Output Variable	Name	Data Type	Value Range	Initial Value	Description
xBusy	Executing	BOOL	[TRUE, FALSE]	FALSE	TRUE: Function execution is successful. FALSE: Function execution fails.
xError	Error	BOOL	[TRUE, FALSE]	FALSE	TRUE: Function execution is successful. FALSE: Function execution fails.
dwErrorID	Error code	DWORD	-	0	Error ID
fValue	Current value	REAL	-	0	Current value

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
xEnable	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
fStart	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-
fEnd	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-
xPeriodic	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
uiCycleNum	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-
xValid	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
xBusy	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
xError	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
dwErrorID	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
fValue	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-

■ Program example

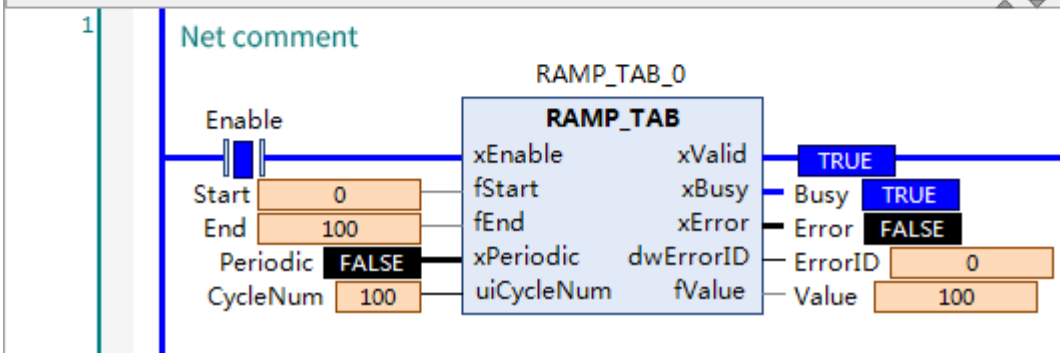
ST

Expression	Type	Value
Enable	BOOL	TRUE
Start	REAL	0
End	REAL	100
Periodic	BOOL	FALSE
CycleNum	UINT	100
Valid	BOOL	TRUE

```
1  RAMP_TAB_1(
2      xEnable TRUE :=Enable TRUE,
3      fStart 0 :=Start 0,
4      fEnd 100 := End 100,
5      xPeriodic FALSE :=Periodic FALSE ,
6      uiCycleNum 100 :=CycleNum 100 ,
7      xValid TRUE =>Valid TRUE ,
8      xBusy TRUE =>Busy TRUE ,
9      xError FALSE =>Error FALSE ,
10     dwErrorID 0 => ErrorID 0,
11     fValue 100 => Value 100 );
12     RETURN
```

LD

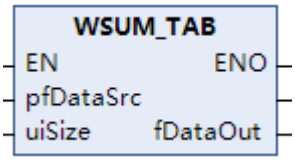
Expression	Type	Value
RAMP_TAB_0	RAMP_TAB	
Enable	BOOL	TRUE
Start	REAL	0
End	REAL	100
Periodic	BOOL	FALSE
CycleNum	UINT	100
Valid	BOOL	FALSE
Busy	BOOL	TRUE
Error	BOOL	FALSE
ErrorID	DWORD	0
Value	REAL	100



4.5. 9 WSUM_TAB

This instruction calculates the total sum for a specified amount of data in a table.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
WSUM_TAB	Calculate the total value	FC		WSUM_TAB(pfDataSrc:= , uiSize:= , fDataOut=>)

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
pfDataSrc	Source data	POINTER TO REAL	-	-	Head address of data
uiSize	Data length	UINT	1 to 1024	0	Data length

Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
fDataOut	Target data	REAL	-	-	Total sum

Output Vari-able	Name	Data Type	Value Range	Initial Value	Description
fDataOut	Target data	REAL	-	-	Total sum

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
pfDataSrc	POINTER TO REAL																			
uiSize	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-
fDataOut	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-

■ Program example

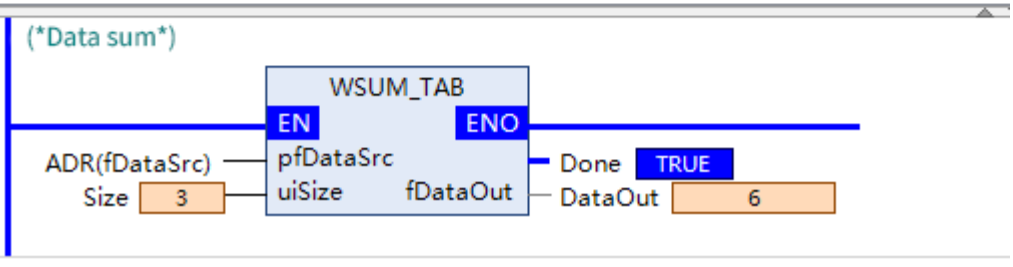
ST

Expression	Type	Value
fDataSrc	ARRAY [0..10] OF R...	
fDataSrc[0]	REAL	1
fDataSrc[1]	REAL	2
fDataSrc[2]	REAL	3
fDataSrc[3]	REAL	0
fDataSrc[4]	REAL	0
fDataSrc[5]	REAL	0
fDataSrc[6]	REAL	0
fDataSrc[7]	REAL	0
fDataSrc[8]	REAL	0
fDataSrc[9]	REAL	0
fDataSrc[10]	REAL	0
Size	UINT	3
DataOut	REAL	6

```
1  WSUM_TAB(
2  pfDataSrc:= ADR(fDataSrc),
3  uiSize:= Size 3 ,
4  fDataOut=>DataOut 6 );
5  RETURN
```

LD

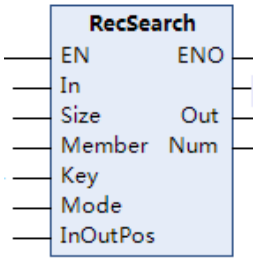
Expression	Type	Value
fDataSrc	ARRAY [0..10] OF R...	
fDataSrc[0]	REAL	1
fDataSrc[1]	REAL	2
fDataSrc[2]	REAL	3
fDataSrc[3]	REAL	0
fDataSrc[4]	REAL	0
fDataSrc[5]	REAL	0
fDataSrc[6]	REAL	0
fDataSrc[7]	REAL	0
fDataSrc[8]	REAL	0
fDataSrc[9]	REAL	0
fDataSrc[10]	REAL	0
Size	UINT	3
DataOut	REAL	6
Done	BOOL	TRUE



4.5. 10 RecSearch

This instruction searches an array of structures for elements that match the search key with the specified method.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
RecSearch	Record search	FC		RecSearch(In :=, Size := , Member := , Key := , Mode := , InOutPos := , Out =>, Num =>);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
In[] (array)	Array to search	-	Depends on data types	-	Array of structures to search
Size	Number of elements to search	UINT	Depends on data types	1	Number of array elements to search
Member	Member to search	-	Depends on data types	-	Member of In[] structure to search
Key	Search keyword	-	Depends on data types	-	Search value
Mode	Search method	_eSEARCH_MODE	1 (_LINEAR), 2 (_BIN_ASC), 3 (_BIN_DESC)	1 (_LINEAR)	Search method

In-out variables

In-Out Variable	Name	Data Type	Value Range	Initial Value	Description
InOutPos[] (array)	Element numbers of matching elements	UINT	Depends on data types	0	Element numbers of matching elements

Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
Out	Search result	BOOL	[FALSE, TRUE]	FALSE	TRUE: There are elements that match conditions. FALSE: There are no elements that match conditions.
Num	Number of matches	UINT	Depends on data types	0	Number of matches

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In[] (array)	Specify an array of structures.																			
Size	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-
MemBer	-	-	-	-	-	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√
	Data with the same data type as the search member of In[]																			
Key	Array with the same data type as the elements of Member																			
Mode	Refer to Function for the enumerators of the enumerated type _eSEARCH_MODE.																			
InOut-Pos[] (array)	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-
Out	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Num	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-

■ Function

This instruction searches an array of structures In[] for Size elements ranging from In[0] to In[Size − 1] where the value of Member matches Key.

One of the members to search in the elements of In[] is transmitted to Member as a parameter.

If any matching elements are found, the value of search result Out changes to TRUE. The element number of the matching element is assigned to InOutPos[0] and the number of matching elements is assigned to Num. If there are two or more matching elements, the element number of the lowest matching element in In[] is assigned to InOutPos[0].

If there are no matching elements, the value of Out is FALSE and InOutPos[0] and Num are 0.

Always attach the element number to the input parameter that is transmitted to In[], such as Array[3].

The data type of search method Mode is enumerated type _eSEARCH_MODE. The meanings of the enumerators are as follows.

Enumerator	Meaning
1 (_LINEAR)	Linear search
2 (_BIN_ASC)	Ascending binary search
3 (_BIN_DESC)	Descending binary search

■ Program example

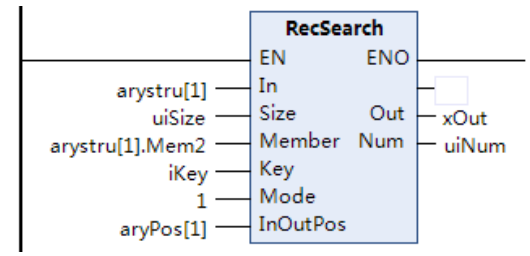
For a linear search, the search is performed in order from the first element of In[].

The following example is for when Size is UINT#5, Key is INT#1234 and Mode is _LINEAR.

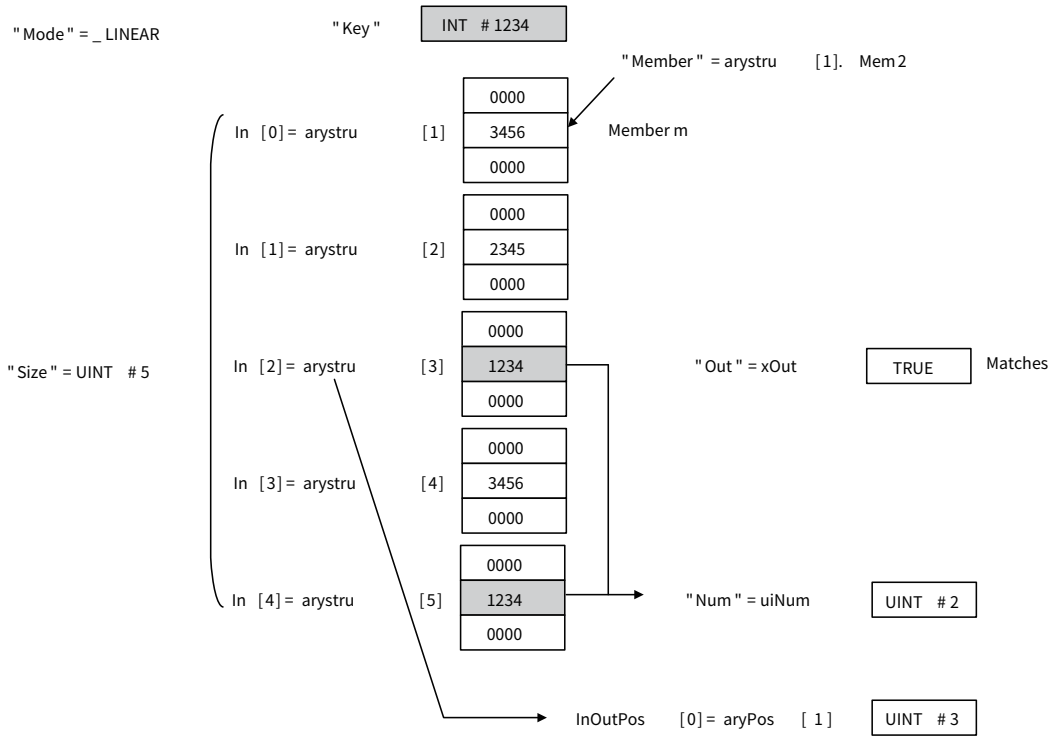
Item	LD	ST

Defined variable	VAR
	arystru : ARRAY[1..5] OF STRUCT_TEST;
	uiSize : UINT := 5;
	iKey : INT := 1234;
	xOut : BOOL;
	uiNum : UINT;
	aryPos : ARRAY[1..5] OF UINT;
	END_VAR	

Program

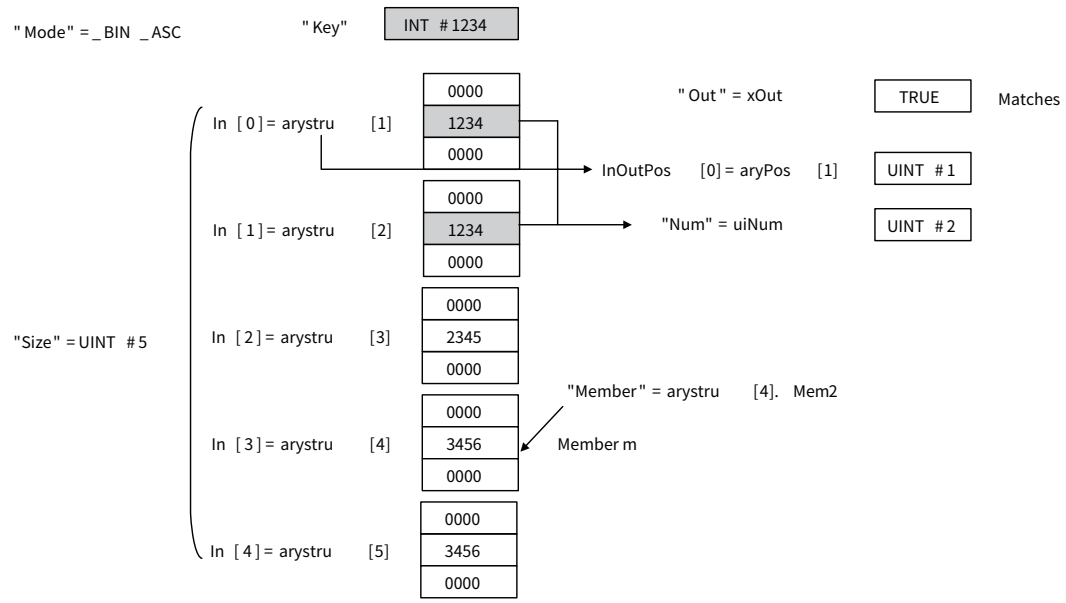



```
RecSearch(
    In := arystru[1],
    Size := uiSize,
    Member := arystru[1].Mem2,
    Key := iKey,
    Mode := 1,
    InOutPos := aryPos[1],
    Out => xOut,
    Num => uiNum
);
```



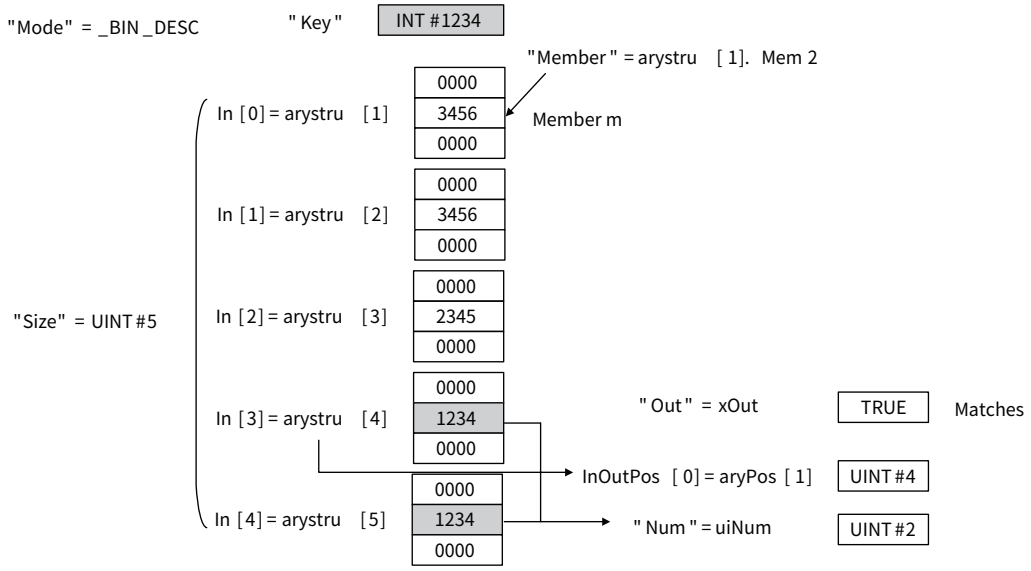
For an ascending binary search, the array elements in the input parameter that is transmitted to In[] must be in ascending order before this instruction is executed. Then a binary search is performed by executing this instruction.

Using the same example as before, the order of the array elements and the processing results will be as shown below for an ascending binary search.



For a descending binary search, the array elements in the input parameter that is transmitted to In[] must be in descending order before this instruction is executed. Then a binary search is performed by executing this instruction.

Using the same example as before, the order of the array elements and the processing results will be as shown below for a descending binary search.



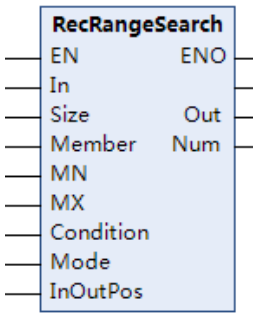
■ Precautions

1) The overall structure of In[] can also be a higher-level member.
2) (Example) In[0] = str0.str1[0].
3) When Member is a real number, different values may introduce errors and may not produce the expected results.
4) When the value of Size is 0, the value of Out is FALSE and the value of Num is 0. InOutPos[] does not change.
5) When the value of Mode is _BIN_ASC or _BIN_DESC and the elements of In[] are not in ascending or descending order, the correct result is not obtained. Place the elements in ascending or descending order before executing this instruction.
6) When Key is a real number, do not specify nonnumeric data for Key. Otherwise, the program may fail.
7) In[] cannot be a multi-dimensional array.
8) In[] must be an array of structure members. Otherwise, an error occurs during compilation.
9) Use the same data type for Key and Member. If they are different, an error occurs during compilation.
10) When the value of Mode exceeds the valid range, an error occurs during compilation.
11) When the value of Size exceeds the In[] array, the return value is FALSE and In[] does not change.
12) When Member is not a member of In[], the return value is FALSE and In[] does not change.
13) Both InOutPos[] and In[] must be one-dimensional arrays.
14) When Member is of the String data type and is not terminated by a NULL text string, the program may fail.

4.5. 11 RecRangeSearch

This instruction searches an array of structures for elements that match the search condition range with the specified method.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
RecRangeSearch	Range record search	FC		RecSearch(In := , Size := , Member := , MN := , MX := , Condition:= , Mode :=, InOutPos:=, Out=>, Num=>);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
In[] (array)	Array to search	-	Depends on data types	-	Array of structures to search
Size	Number of elements to search	UINT	Depends on data types	1	Number of array elements to search
Member	Member to search	-	Depends on data types	-	Member of In[] structure to search
MN	Search condition lower limit	-	Depends on data types	-	Search condition lower limit
MX	Search condition upper limit	-	Depends on data types	-	Search condition upper limit
Condition	Search condition	_eSEARCH_CONDITION	1 (_EQ_BOTH), 2 (_EQ_MIN), 3 (_EQ_MAX), 4 (_NE_BOTH)	1 (_EQ_BOTH)	Search condition
Mode	Search method	_eSEARCH_MODE	1 (_LINEAR), 2 (_BIN_ASC), 3 (_BIN_DESC)	1 (_LINEAR)	Search method

In-out variables

In-Out Variable	Name	Data Type	Value Range	Initial Value	Description
InOutPos[] (array)	Element numbers of matching elements	UINT	Depends on data types	0	Element numbers of matching elements

Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
Out	Search result	BOOL	[FALSE, TRUE]	FALSE	TRUE: There are elements that match conditions. FALSE: There are no elements that match conditions.

Num	Number of matches	UINT	Depends on data types	0	Number of matches

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In[] (array)	Specify an array of structures.																			
Size	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-
MemBer	-	-	-	-	-	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√
	Data with the same data type as the search member of In[]																			
MN	Array with the same data type as the elements of Member																			
MX	Array with the same data type as the elements of Member																			
Condition	Refer to Function for the enumerators of the enumerated type _eSEARCH_CONDITION.																			
Mode	Refer to Function for the enumerators of the enumerated type _eSEARCH_MODE.																			
InOutPos[] (array)	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-
Out	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Num	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-

■ Function

This instruction searches an array of structures In[] for Size elements ranging from In[0] to In[Size − 1] where the value of Member matches the search conditions.

Condition specifies the search condition. Mode specifies the search method. Details are provided below.

One of the members to search in the elements of In[] is transmitted to Member as a parameter.

If any elements matching the conditions are found, the value of search result Out changes to TRUE. The element number of the matching element is assigned to InOutPos[0] and the number of matching elements is assigned to Num. If there are two or more matching elements, the element number of the lowest matching element in In[] is assigned to InOutPos[0].

If there are no matching elements, the value of Out is FALSE and InOutPos[0] and Num are 0.

Always attach the element number to the input parameter that is transmitted to In[], such as Array[3].

The data type of search condition Condition is enumerated type _eSEARCH_CONDITION. The meanings of the enumerators are as follows.

Enumerator	Meaning
1 (EQ_BOTH)	MN ⩽ Member ⩽ MX
2 (_EQ_MIN)	MN ⩽ Member < MX
3 (_EQ_MAX)	MN < Member ⩽ MX
4 (_NE_BOTH)	MN < Member < MX

The data type of search method Mode is enumerated type _eSEARCH_MODE. The meanings of the enumerators are as follows.

Enumerator	Meaning
1 (_LINEAR)	Linear search
2 (_BIN_ASC)	Ascending binary search

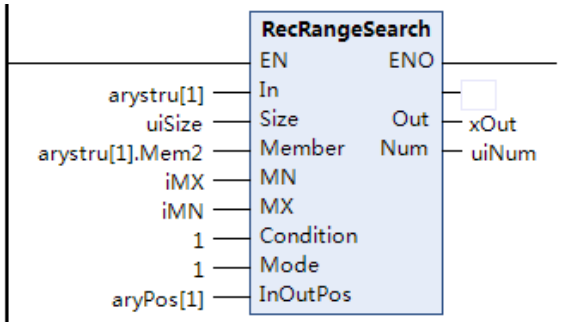
Enumerator	Meaning
3 (_BIN_DESC)	Descending binary search

■ Program example

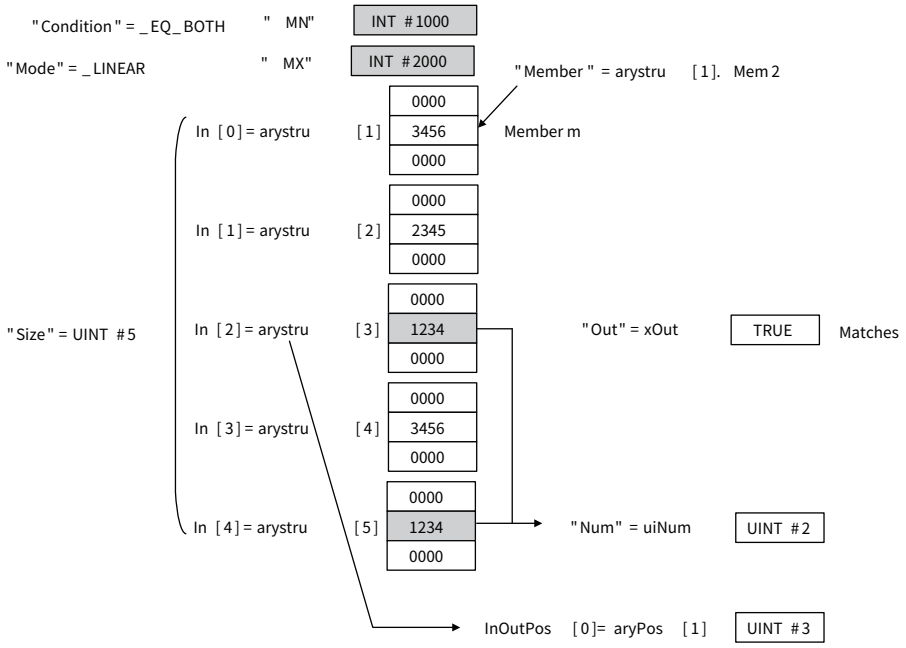
The following example is for when Size is UINT#5, MN is IN#1000, MX is IN#2000, Condition is _EQ_BOTH, and Mode is _LINEAR.

Item	LD	ST
Defined variable		
Program		

```
VAR
    arystru : ARRAY[1..5] OF STRUCT_TEST;
    uisize : UINT :=5;
    iMX    : INT :=2000;
    iMN    : INT :=1000;
    xOUT   :BOOL;
    uiNum  :UINT;
    aryPos :ARRAY[1..5] OF UINT;
END_VAR
```

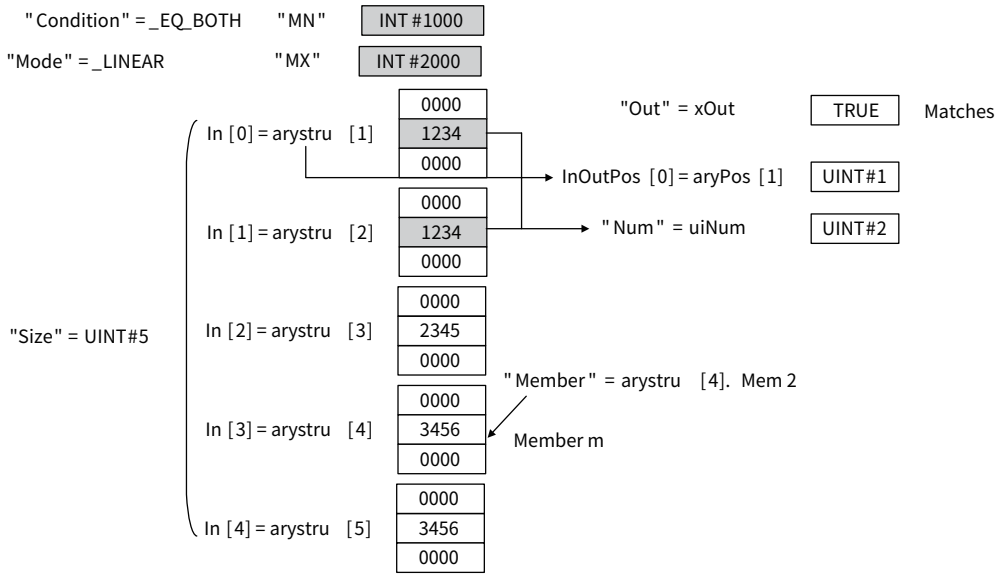



```
RecRangeSearch(
    In := arystru[1],
    Size := uiSize,
    Member := arystru[1].Mem2,
    MN := iMN,
    MX := iMX,
    Condition := 1,
    Mode := 1,
    InOutPos := aryPos[1],
    Out => xOut,
    Num => uiNum
);
```



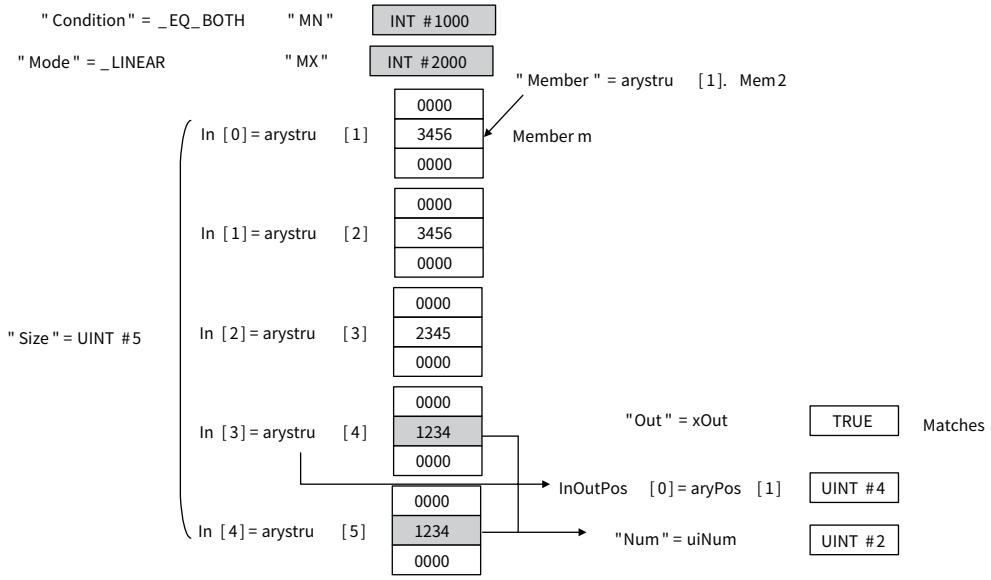
For an ascending binary search, the array elements in the input parameter that is transmitted to In[] must be in ascending order before this instruction is executed. Then a binary search is performed by executing this instruction.

Using the same example as before, the order of the array elements and the processing results will be as shown below for an ascending binary search.



For a descending binary search, the array elements in the input parameter that is transmitted to In[] must be in descending order before this instruction is executed. Then a binary search is performed by executing this instruction.

Using the same example as before, the order of the array elements and the processing results will be as shown below for a descending binary search.



■ Precautions

· The overall structure of In[] can also be a higher-level member.

(Example) In[0] = str0.str1[0].

· In[] must be an array of structure members. Otherwise, an error occurs during compilation.

· Make the data types of Member, MN, and MX the same as that of the members searched in In[]. If they are different, an error occurs during compilation.

· When an element in the array is transmitted to In[], all elements following the transmitted element are processed.

· When Member is a real number, different values may introduce errors and may not produce the expected results.

· When MN or MX is a real number, do not specify nonnumeric data for them.

· When the value of Size is 0, the value of Out is FALSE and the value of Num is 0. InOutPos[] does not change.

· When the value of Mode is _BIN_ASC or _BIN_DESC and the elements of In[] are not in ascending or descending order, the correct result is not obtained. Place the elements in ascending or descending order before executing this instruction.

· When the value of MN is greater than the value of MX, an error occurs and the return value is FALSE.

· When the value of Condition exceeds the valid range, an error occurs during compilation.

· When the value of Mode exceeds the valid range, an error occurs during compilation.

· When the value of Size exceeds the In[] array, the return value is FALSE and In[] does not change.

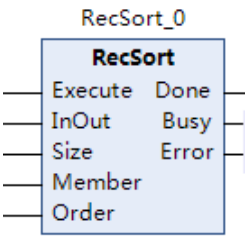
· When Member is not a member of In[], the return value is FALSE and In[] does not change.

· Both InOutPos[] and In[] must be one-dimensional arrays.

4.5. 12 RecSort

This instruction sorts the elements of an array of structures.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
RecSort	Record sort	FB		RecSort(Execute :=, InOut := , Size := , Member := , Order := , Done =>, Busy =>, Error =>);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
Size	Number of elements to sort	UINT	Depends on data types	1	Number of array elements to sort
Member	Member to sort	-	Depends on data types	-	Member of InOut[] structure to sort
Order	Sort order	_eSORT_ORDER	1 (_ASC), 2 (_DESC)	1 (_ASC)	Sort order

In-out variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
InOut[] (array)	Array to sort	-	Depends on data types	-	Array of structures to sort

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Size	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-
MemBer	-	-	-	-	-	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√
	Data with the same data type as the search member of InOut[]																			
Order	Refer to Function for the enumerators of the enumerated type _eSORT_ORDER.																			
InOut[] (array)	Specify an array of structures.																			

■ Function

When the value of Execute is TRUE, Size elements of InOut[] (a structure array) are sorted. Specifically, the elements from InOut[0] to InOut[Size − 1] are sorted based on the value of Member for the structure. Order specifies the sort order. Details are provided below.

One of the members to sort in the elements of InOut[] is transmitted to Member as a parameter.

Always attach the element number to the in-out parameter that is transmitted to InOut[], such as Array[3].

The data type of sort order Order is enumerated type _eSORT_ORDER. The meanings of the enumerators are as follows.

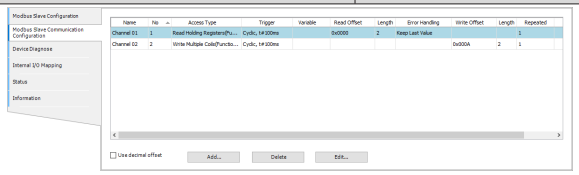
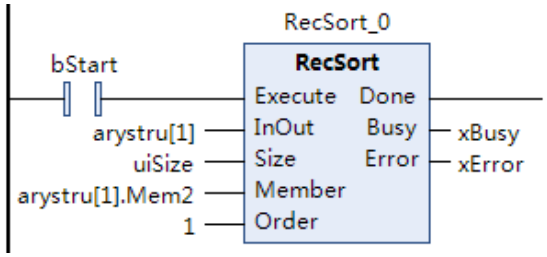
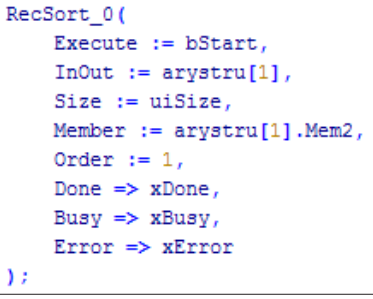
Enumerator	Meaning
1 (_ASC)	Ascending
2 (_DESC)	Descending

The relationships between values with data types that are not integers or real numbers are determined as given in the following table.

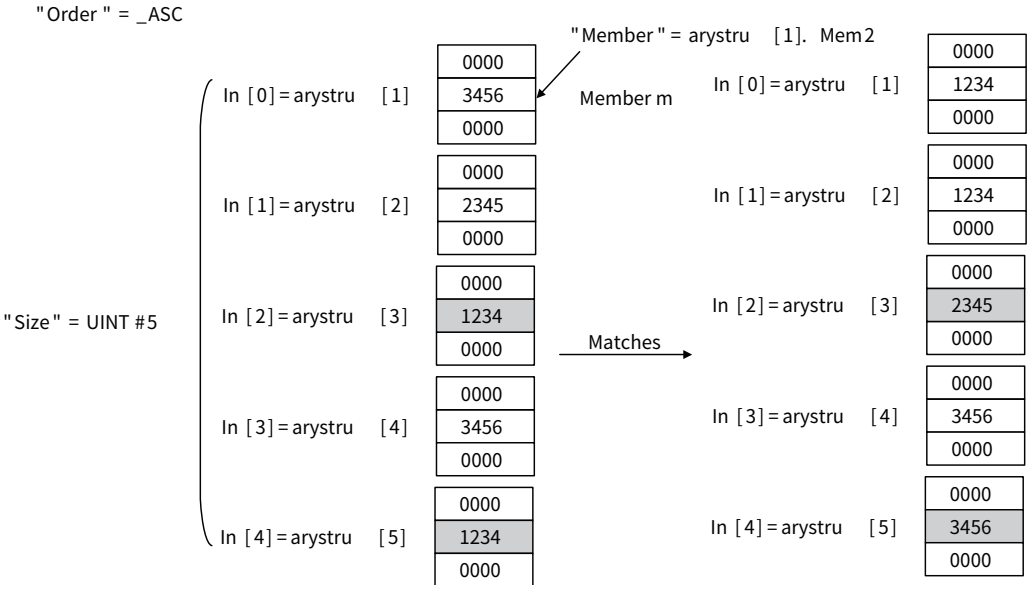
Data Type	Relationship
TIME	The numerically larger value is considered to be larger.
DATE, TOD, or DT	Later dates or TOD are considered to be larger.
STRING	The character code is used for size comparison. The comparison steps are as follows: First, the first character codes in all of the text strings are compared. If the first character codes are different, the size of each character code is the size of the corresponding text string. If the first character codes are the same, comparison continues in order to the other characters until a different character code is found. If the lengths of the text strings are different, NULL characters (16#00) are added to the shorter text string to complete the comparison.

■ Program example

The following example is for when Size is UINT#5 and Order is _ASC.

Item	LD	ST
Defined variable		
Program		


```
RecSort_0(
    Execute := bStart,
    InOut := arystru[1],
    Size := uiSize,
    Member := arystru[1].Mem2,
    Order := 1,
    Done => xDone,
    Busy => xBusy,
    Error => xError
);
```



■ Precautions

· InOut[] must be an array of structure members. Otherwise, an error occurs during compilation.

· Execution of this instruction is continued until processing is completed even if the value of Execute changes to FALSE or the execution time exceeds the task period. The value of Done changes to TRUE when processing is completed. Use this to confirm normal completion of processing.

· When Member is a real number, different values may introduce errors and may not produce the expected results.

· When an element in the array is transmitted to InOut[], all elements following the transmitted element are processed.

· InOut[] must be a one-dimensional array. Otherwise, an error occurs during compilation.

· When the value of Size is 0, the value of Done is TRUE and InOut[] does not change.

· When the value of Order exceeds the valid range, an error occurs during compilation.

· When the value of Size exceeds the InOut[] array, the return value is FALSE and InOut[] does not change.

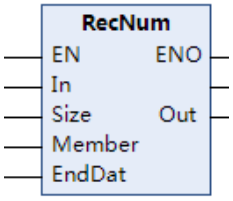
· When Member is not a member of InOut[], the return value is FALSE and InOut[] does not change.

· When Member is of the String data type and is not terminated by a NULL text string, an error may occur in the result.

4.5. 13 RecNum

This instruction counts records in an array of structures to the end data.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
RecNum	Get number of records	FC		RecNum(In, := Size := , Member := , EndDat := , Out =>);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
In[] (array)	Array to process	-	Depends on data types	-	Array of structures to process
Size	Number of elements to process	UINT	Depends on data types	1	Number of array elements to process
Member	Member to process	-	Depends on data types	-	Member of In[] structure to process
EndDat	End data	-	Depends on data types	-	Technical data

Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
Out	Number of records	UINT	Depends on data types	-	Number of records

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In[] (array)	Specify an array of structures.																			
Size	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-
MemBer	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√
	Data with the same data type as the search member of In[]																			
EndDat	Array with the same data type as the elements of Member																			
InOut[] (array)	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-

■ Function

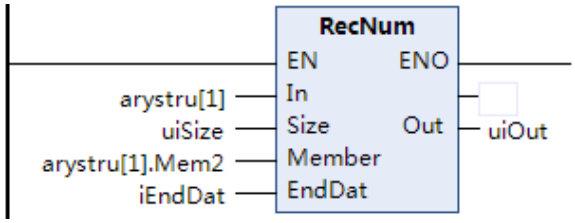
This instruction searches an array of structures In[] for Size elements ranging from In[0] to In[Size − 1] where the value of Member matches the end data EndDat. Then the number of elements (records) matching EndDat is assigned to Out.

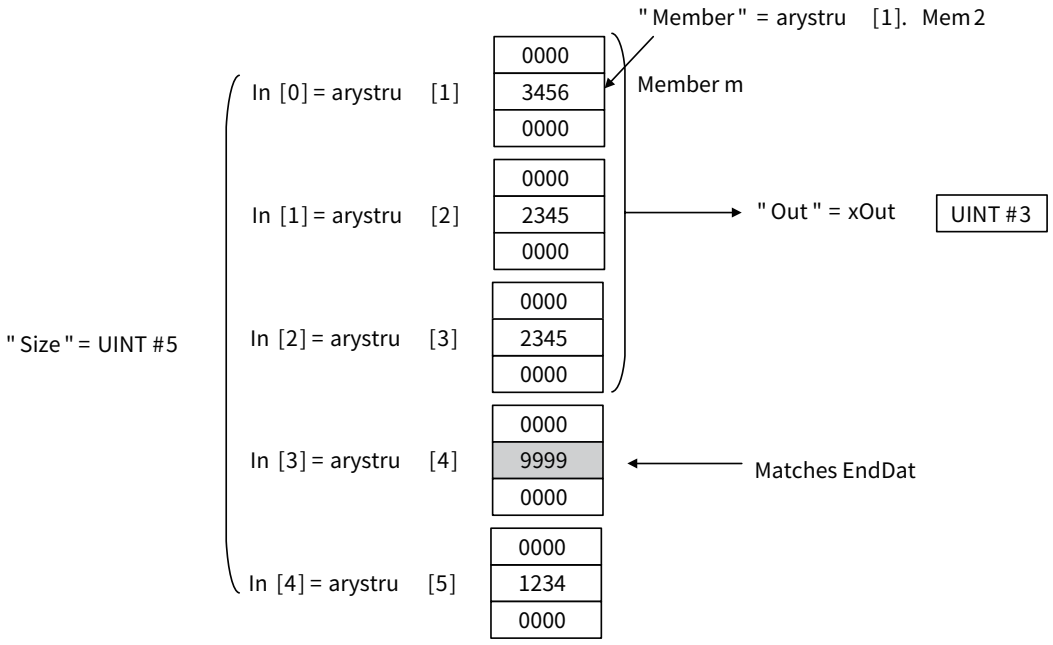
One of the members in the elements of In[] is transmitted to Member as a parameter.

Always attach the element number to the input parameter that is transmitted to In[], such as Array[3].

■ Program example

The following example is for when EndDat is INT#9999.

Item	LD	ST
Defined variable	VAR arystru : ARRAY[1..5] OF STRUCT_TEST; uiSize : UINT := 5; iEndDat : INT; uiOut : UINT; END_VAR	
Program		RecNum(In:= arystru[1], Size:= uiSize, Member:= arystru[1].Mem2, EndDat:= iEndDat, Out=> uiOut);



■ Precautions

· In[] must be an array of structure members. Otherwise, an error occurs during compilation.

· Use the same data type for Member and EndDat. If they are different, an error occurs during compilation.

· If no element in Size members of In[] matches EndDat, the value of Out is Size.

· When Member is a real number, different values may introduce errors and may not produce the expected results.

· When EndDat is a real number, do not specify nonnumeric data for EndDat.

· When an element in the array is transmitted to In[], all elements following the transmitted element are

processed.

· In[] must be a one-dimensional array. Otherwise, an error occurs during compilation.

· When the value of Size is 0, the value of Out is initialized to 0.

· When the value of Size exceeds the In[] array, the return value is FALSE and the value of Out is initialized to 0.

· When the value of Member is not a member of In[], the return value is FALSE and the value of Out is initialized to 0.

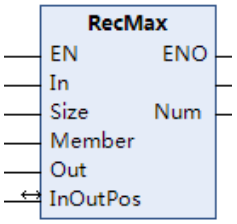
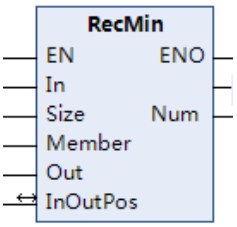
· When Member is of the String data type and is not terminated by a NULL text string, an error may occur in the result.

4.5. 14 RecMax and RecMin

RecMax: Searches the array of structures for the maximum value of a specified member.

RecMin: Searches the array of structures for the minimum value of a specified member.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
RecMax	Maximum record search	FC		RecMax(In := , Size := , Member := , Out := , InOutPos := , Num =>);
RecMin	Minimum record search	FC		RecMax(In := , Size := , Member := , Out := , InOutPos := , Num =>);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
In[] (array)	Array to search	-	Depends on data types	-	Array of structures to search
Size	Number of elements to search	UINT	Depends on data types	1	Number of array elements to search
Member	Member to search	-	Depends on data types	-	Member of In[] structure to search

In-out variables

In-Out Variable	Name	Data Type	Value Range	Initial Value	Description
InOutPos[] (array)	Found element number	UINT	Depends on data types	-	Found element number

In-Out Variable	Name	Data Type	Value Range	Initial Value	Description
Num	Number found	UINT	Depends on data types	-	Number found

Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
Out	Search result	-	Depends on data types	-	Search result
Num	Number found	UINT	Depends on data types	-	Number found

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In[] (array)	Specify an array of structures.																			
Size	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-
MemBer	-	-	-	-	-	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√
	Data with the same data type as the search member of In[]																			
InOut[] (array)	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-
Out	-	-	-	-	-	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√
Num	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-

■ Function

This instruction searches an array of structures In[] for Size elements ranging from In[0] to In[Size − 1] to retrieve the value of Member.

One of the members to search in the elements of In[] is transmitted to Member as a parameter.

The element number of the search result is assigned to InOutPos[0] and the number of retrieved elements is assigned to Num. If there are more than two search results, the element number of the lowest search result in In[] is assigned to InOutPos[0].

Always attach the element number to the input parameter that is transmitted to In[], such as Array[3].

The relationships between values with data types that are not integers or real numbers are determined as given in the following table.

Data Type	Relationship
TIME	The numerically larger value is considered to be larger.
DATE, TOD, or DT	Later dates or TOD are considered to be larger.
STRING	The character code is used for size comparison. The comparison steps are as follows: First, the first character codes in all of the text strings are compared. If the first character codes are different, the size of each character code is the size of the corresponding text string. If the first character codes are the same, comparison continues in order to the other characters until a different character code is found. If the lengths of the text strings are different, NULL characters (16#00) are added to the shorter text string to complete the comparison.

RecMax

This instruction searches for the maximum value. The maximum value of the member to search is assigned to search result Out.

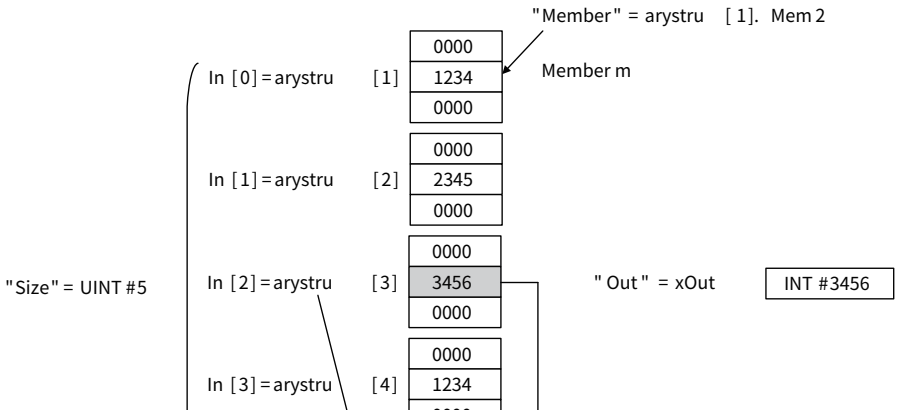
RecMin

This instruction searches for the minimum value. The minimum value of the member to search is assigned to search result Out.

■ Program example

The following example shows the RecMax instruction when Size is UINT#5.

Item	LD	ST
Defined variable	VAR arystru : ARRAY[1..5] OF STRUCT_TEST; uiSize : UINT := 5; iOut : INT; uiNum : UINT; aryPos : ARRAY[1..5] OF UINT; END_VAR	
Program	RecMax EN ENO arystru[1] In uiSize Size Num uiNum arystru[1].Mem2 Member iOut Out aryPos[1] InOutPos	RecMax(In := arystru[1], Size := uiSize, Member := arystru[1].Mem2, Out := iOut, InOutPos := aryPos[1], Num => uiNum);



■ Precautions

· The overall structure of In[] can also be a higher-level member.

· (Example) In[0] = str0.str1[0]

Use the same data type for Member and Out.

· When Member is a real number, different values may introduce errors and may not produce the expected results.

· When an element in the array is transmitted to In[], all elements following the transmitted element are

processed.

· When In is an enumeration, the input parameter transmitted to In must be a variable. If a constant is transmitted to In, a compilation error occurs.

· When the value of Size is 0, the values of Out and Num are 0 and InOutPos[] does not change.

· InOutPos[] and In[] must be one-dimensional arrays. Otherwise, an error occurs during compilation.

· When the value of Size exceeds the In[] array, the return value is FALSE, and Out, Num, and InOutPos[] do not change.

· When the value of Member is not a member of In[], the return value is FALSE, and Out, Num, and InOut-Pos[] do not change.

· When Member is of the String data type and is not terminated by a NULL text string, an error may occur in the result.

4. 6 File Operation Instructions

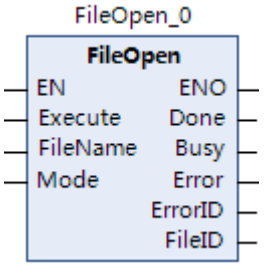
4.6. 1 Instruction List

Instruction Category	Name	FB/FC	Function
File operation instructions	FileOpen	FB	Open file
	FileClose	FB	Close file
	FileRead	FB	Read file
	FileWrite	FB	Write file
	FileCopy	FB	Copy file
	FileRemove	FB	Delete file
	FileRename	FB	Change file name
	FileWriteVar	FB	Write variable to file
	FileReadVar	FB	Read variable from file
	FileSeek	FB	Seek file
	FileGets	FB	Get text string
	FilePuts	FB	Put text string
	DirCreate	FB	Create directory
	DirRemove	FB	Delete directory

4.6. 2 FileOpen

This instruction opens the specified file in the SD memory card.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
FileOpen	Open file	FB		FileOpen_0(Execute :=, FileName :=, Mode :=, Done =>, Busy =>, Error =>, ErrorID =>, FileID =>,);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
Execute	Enable	BOOL	Depends on data types	-	(Triggered by edges) Execution of the function block switch
FileName	Specified file name	STRING	A maximum of 1985 characters	-	Name of the file to open
Mode	Open mode	ACCESS_MODE	Depends on data types	-	Mode in which to open file

Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
Done	Function block completed state	BOOL	Depends on data types	-	Function block completed state
Busy	Function block execution flag	BOOL	Depends on data types	-	Function block execution flag
Error	Function block error flag	BOOL	Depends on data types	-	Function block error flag
ErrorID	Function block error ID	WORD	Depends on data types	-	Function block error ID
FileID	File ID	DWORD	Depends on data types	-	ID of the opened file

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Execute	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
FileName	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√
Mode	Refer to Function for the enumerators for the enumerated type ACCESS_MODE.																			
Done	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Busy	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Error	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
ErrorID	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
FileID	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

■ Function

1. This instruction opens the file specified by FileName in the SD memory card in the mode specified by Mode.

2. The result outputs the file ID specified by FileID. FileID is used to specify the file in other instructions, such as FileRead and FileWrite.

3. The data type of Mode is enumerated type ACCESS_MODE. The meanings of the enumerators are as follows.

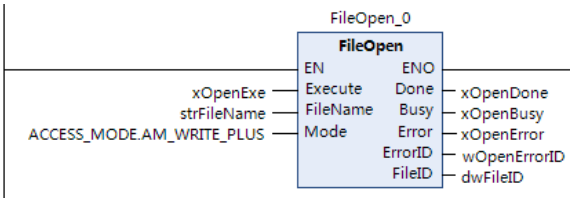
Enumerator	Meaning
AM_READ	If the file exists in the target path, the file is opened in read-only mode. If the file does not exist, an error value is returned.
AM_WRITE	If the file exists in the target path, the file is opened in re-write mode. If the file does not exist, a new file with the write permission is automatically created.
AM_APPEND	If the file exists in the target path, open the file in write-only mode and the content is appended to the end of the file. If the file does not exist, an error value is returned.
AM_READ_PLUS	If the file exists in the target path, open the file in read-write mode. If the file does not exist, an error value is returned.
AM_WRITE_PLUS	If the file exists in the target path, the file is opened in re-read-write mode. If the file does not exist, a new file with the read/write permission is automatically created.
AM_APPEND_PLUS	If the file exists in the target path, the file is opened in read-write mode and the content is appended to the end of the file. If the file does not exist, a new file with the read/write permission is automatically created.

■ Program example

<table>
<tr><th>Item</th><th>LD</th><th>ST</th></tr>
<tr><td>Defined variable</td><td colspan="2">

```
VAR
    FileOpen_0      : FileOpen;
    xOpenExe        : BOOL;
    strFileName     : STRING := 'FileTest/File.txt';
    xOpenDone       : BOOL;
    xOpenBusy       : BOOL;
    xOpenError      : BOOL;
    wOpenErrorID    : WORD;
    dwFileID        : DWORD;
END_VAR
```

</td></tr>
<tr><td>Program</td><td>



</td><td>

```
FileOpen_0(Execute := xOpenExe,
    FileName := strFileName,
    Mode := ACCESS_MODE.AM_WRITE_PLUS,
    Done => xOpenDone,
    Busy => xOpenBusy,
    Error => xOpenError,
    ErrorID => wOpenErrorID,
    FileID => dwFileID,
    );
```

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

Item	LD		ST
Running result	FileOpen_0	FileOpen	
	xOpenExe	BOOL	TRUE
	strFileName	STRING	'FileTest/File.txt'
	xOpenDone	BOOL	TRUE
	xOpenBusy	BOOL	FALSE
	xOpenError	BOOL	FALSE
	wOpenErrorID	WORD	0
	dwFileID	DWORD	4459552

■ Precautions

· Execution of this instruction is continued until processing is completed even if the value of Execute changes to FALSE or the execution time exceeds the task period. The value of Done changes to TRUE when processing is completed. Use this to confirm normal completion of processing.

· This instruction must be executed before the FileSeek, FileRead, FileWrite, FileGets, and FilePuts instructions.

· After using the file opened by this instruction, execute the FileClose instruction to close the file.

· After this instruction is completed, save the value to FileID. The value of Done changes to TRUE.

· When the file is opened, remove the SD memory card. Then the file remains the open state. When you insert the SD memory card again, you cannot read or write the file. To read and write the file, re-open the file.

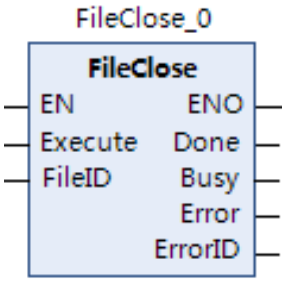
An error occurs in the following cases. The value of Error is TRUE.

1) The SD memory card is unavailable.
2) The SD memory card is write-protected.
3) The value of Mode is AM_READ, AM_APPEND, or AM_READ_PLUS, and the file specified by FileName does not exist.
4) The value of FileName is not a valid file name.
5) The maximum number of files or directories is exceeded.
6) The file specified by FileName is being accessed.
7) The file specified by FileName is read-only.
8) An error that prevents access occurs during SD memory card access.

4.6. 3 FileClose

This instruction closes the specified file in the SD memory card.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
FileClose	Close file	FB		FileClose_0(Execute :=, FileID :=, Done =>, Busy =>, Error =>, ErrorID =>,);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
Execute	Enable	BOOL	Depends on data types	-	(Triggered by edges) Execution of the function block switch
FileID	File ID	DWORD	Depends on data types	-	ID of the file to close

Output variables

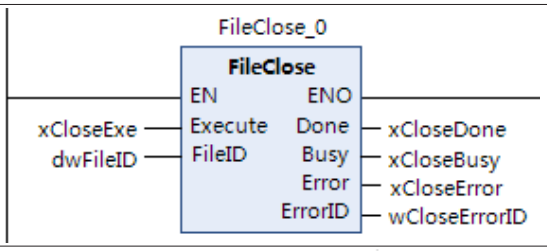
Output Variable	Name	Data Type	Value Range	Initial Value	Description
Done	Function block completed state	BOOL	Depends on data types	-	Function block completed state
Busy	Function block execution flag	BOOL	Depends on data types	-	Function block execution flag
Error	Function block error flag	BOOL	Depends on data types	-	Function block error flag
ErrorID	Function block error ID	WORD	Depends on data types	-	Function block error ID

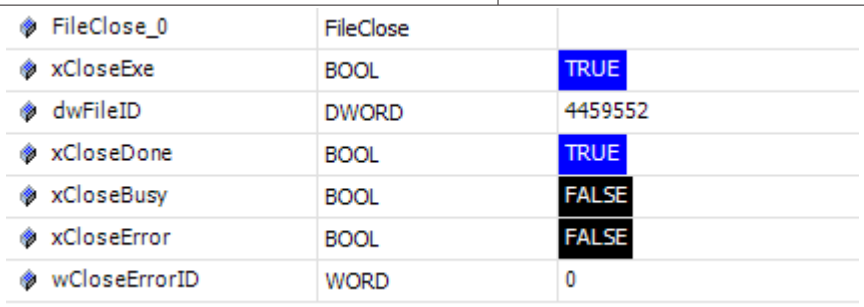
	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Execute	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
FileID	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Done	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Busy	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Error	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
ErrorID	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

■ Function

This instruction closes the file specified by FileID in the SD memory card.

■ Program example

<table>
<tr><th>Item</th><th>LD</th><th>ST</th></tr>
<tr><td>Defined variable</td><td colspan="2">
<pre>
VAR
    FileClose_0   : FileClose;
    xCloseExe   : BOOL;
    dwFileID   : DWORD := 4459552;
    xCloseDone   : BOOL;
    xCloseBusy   : BOOL;
    xCloseError   : BOOL;
    wCloseErrorID   : WORD;
END_VAR
</pre>
</td></tr>
<tr><td>Program</td><td>


</td><td>
<pre>
FileClose_0(Execute := xCloseExe,
    FileID := dwFileID,
    Done => xCloseDone,
    Busy => xCloseBusy,
    Error => xCloseError,
    ErrorID => wCloseErrorID,
    );
</pre>
</td></tr>
<tr><td>Running result</td><td colspan="2">


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

■ Precautions

Do not execute this instruction on a file for multiple consecutive times. Otherwise, the PLC fails to run and even breaks down.

Execution of this instruction is continued until processing is completed even if the value of Execute changes to FALSE or the execution time exceeds the task period. The value of Done changes to TRUE when processing is completed. Use this to confirm normal completion of processing.

Execute the FileOpen instruction in advance to obtain the value for FileID.

After using the file opened by the FileOpen instruction, execute this instruction to close the file. Otherwise, the result will be unexpected.

When the file is opened, remove the SD memory card. Then the file remains the open state. When you insert the SD memory card again, you cannot read or write the file. To read and write the file, re-open the file.

An error occurs in the following cases. The value of Error is TRUE.

1) The SD memory card is unavailable.

2) The file specified by FileID is being accessed.

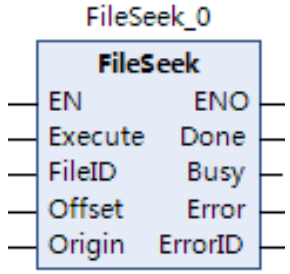
3) The file specified by FileID does not exist.

4) An error that prevents access occurs during SD memory card access.

4.6. 4 FileSeek

This instruction sets a file position indicator in the specified file in the SD memory card.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
FileSeek	Seek file	FB		FileSeek_0(Execute :=, FileID :=, Offset :=, Origin :=, Done =>, Busy =>, Error =>, ErrorID =>,);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
Execute	Enable	BOOL	Depends on data types	-	(Triggered by edges) Execution of the function block switch
FileID	File ID	DWORD	Depends on data types	-	ID of the file in which to set the file position indicator
Offset	Offset	DINT	Depends on data types	-	Offset from Origin
Origin	Reference position	_eFSEEK_ORIGIN	Depends on data types	-	Reference position for file position indicator

Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
Done	Function block completed state	BOOL	Depends on data types	-	Function block completed state
Busy	Function block execution flag	BOOL	Depends on data types	-	Function block execution flag
Error	Function block error flag	BOOL	Depends on data types	-	Function block error flag
ErrorID	Function block error ID	WORD	Depends on data types	-	Function block error ID

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Execute	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
FileID	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Offset	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-
Origin	Refer to Function for the enumerators for the enumerated type _eFSEEK_ORIGIN.																			
Done	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Busy	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Error	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
ErrorID	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

■ Function

This instruction sets a file position indicator in the file specified by file ID FileID in the SD memory card. A file position indicator is the position in a file at which to start reading or writing when an instruction such as the FileRead or FileWrite instruction is executed. For example, to read from the beginning of a file, set a file position indicator at the beginning of the file with the FileSeek instruction, and then execute the FileRead instruction.

The file position indicator is set at offset Offset from reference position Origin.

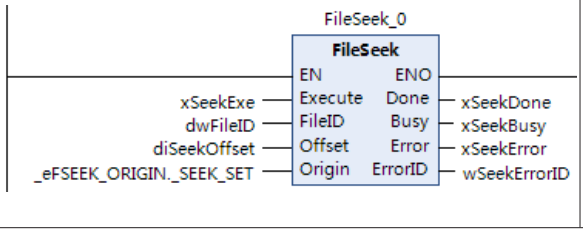
The data type of Origin is enumerated type _eFSEEK_ORIGIN. The meanings of the enumerators are as follows.

Enumerator	Meaning
_SEEK_SET	Beginning of the file
_SEEK_CUR	Location of the current file position indicator
_SEEK_END	End of the file

■ Program example

Item	LD	ST
Defined variable		
Program		

```
VAR
    FileSeek_0      : FileSeek;
    xSeekExe        : BOOL;
    dwFileID        : DWORD := 4459552;
    diSeekOffset    : DINT  := 2;
    xSeekDone       : BOOL;
    xSeekBusy       : BOOL;
    xSeekError      : BOOL;
    wSeekErrorID    : WORD;
END_VAR
```




```
FileSeek_0(Execute := xSeekExe,
    FileID := dwFileID,
    Offset := diSeekOffset,
    Origin := _eFSEEK_ORIGIN._SEEK_SET,
    Done => xSeekDone,
    Busy => xSeekBusy,
    Error => xSeekError,
    ErrorID => wSeekErrorID,
    );
```

Item	LD		ST
Running result	FileSeek_0	FileSeek	
	xSeekExe	BOOL	TRUE
	dwFileID	DWORD	4459552
	diSeekOffset	DINT	2
	xSeekDone	BOOL	TRUE
	xSeekBusy	BOOL	FALSE
	xSeekError	BOOL	FALSE
	wSeekErrorID	WORD	0

■ Precautions

· Execution of this instruction is continued until processing is completed even if the value of Execute changes to FALSE or the execution time exceeds the task period. The value of Done changes to TRUE when processing is completed. Use this to confirm normal completion of processing.

· Execute the FileOpen instruction in advance to obtain the value for FileID.

· An error occurs in the following cases. The value of Error is TRUE.

1) The SD memory card is unavailable.

2) The file specified by FileID is being accessed.

3) The file specified by FileID does not exist.

4) An error that prevents access occurs during SD memory card access.

4.6. 5 FileRead

This instruction reads data from the specified file in the SD memory card.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
FileRead	Read file	FB	FileRead_0 FileRead EN ENO Execute Done FileID Busy ReadBuf Error Size ErrorID ReadSize EOF	FileRead_0(Execute :=, FileID :=, ReadBuf :=, Size :=, Done =>, Busy =>, Error =>, ErrorID =>, ReadSize =>, EOF =>,);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
Execute	Enable	BOOL	Depends on data types	-	(Triggered by edges) Execution of the function block switch

Input Variable	Name	Data Type	Value Range	Initial Value	Description
FileID	File ID	DWORD	Depends on data types	0	ID of the file to read
Size	Number of elements to read	UINT	Depends on data types	1	Number of elements to read

In-out variables

In-out variables	Name	Data Type	Value Range	Initial Value	Description
ReadBuf[]	Read buffer	-	Depends on data types	-	Buffer in which to write data that was read

Output Variable

Output Variable	Name	Data Type	Value Range	Initial Value	Description
Done	Function block completed state	BOOL	Depends on data types	-	Function block completed state
Busy	Function block execution flag	BOOL	Depends on data types	-	Function block execution flag
Error	Function block error flag	BOOL	Depends on data types	-	Function block error flag
ErrorID	Function block error ID	WORD	Depends on data types	-	Function block error ID
ReadSize	Number of elements actually read	UINT	Depends on data types	-	Number of elements actually read
EOF	End of file	BOOL	Depends on data types	-	Whether the end of file is reached TRUE: Reached FALSE: Not reached

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Execute	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
FileID	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
ReadBuf[] (array)	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√
	Arrays enumerations or structures can also be specified.																			
Size	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-
Done	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Busy	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Error	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
ErrorID	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
ReadSize	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-
EOF	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

■ Function

1. This instruction reads data from position of the file position indicator in the file specified by FileID in the SD memory card, and then stores the data in read buffer ReadBuf[].

2. The file position indicator is set at the desired location in advance with the FileSeek instruction.

3. The amount of data that is read is the size of the data type of ReadBuf[] times Size.

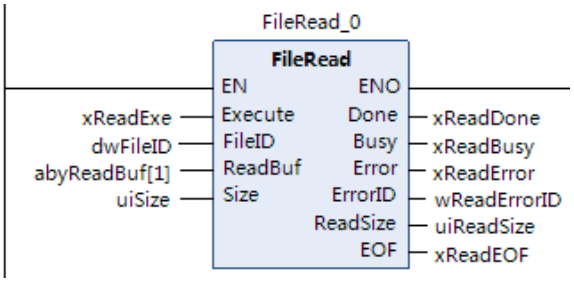
4. The number of elements actually read is stored in ReadSize. In general, values of Size and ReadSize are the same. If the amount of data from the file position indicator to the end of the file is less than the value of Size, no error occurs and the data to the end of the file is stored in ReadBuf[]. If that occurs, the value of ReadSize is less than the value of Size.

5. If data is read to the end of the file, EOF (end of file) changes to TRUE. Otherwise, the value of EOF is FALSE.

■ Program example

<table>
<tr><th>Item</th><th>LD</th><th>ST</th></tr>
<tr><td>Defined variable</td><td colspan="2">

```
VAR
    FileRead_0      : FileRead;
    xReadExe        : BOOL;
    dwFileID        : DWORD := 4459552;
    abyReadBuf      : ARRAY [1..5] OF BYTE;
    uiSize          : UINT  := 3;
    xReadDone       : BOOL;
    xReadBusy       : BOOL;
    xReadError      : BOOL;
    wReadErrorID    : WORD;
    uiReadSize      : UINT;
    xReadEOF        : BOOL;
END_VAR
```

</td></tr>
<tr><td>Program</td><td>


</td><td>

```
FileRead_0(Execute := xReadExe,
    FileID := dwFileID,
    ReadBuf := abyReadBuf[1],
    Size := uiSize,
    Done => xReadDone,
    Busy => xReadBusy,
    Error => xReadError,
    ErrorID => wReadErrorID,
    ReadSize => uiReadSize,
    EOF => xReadEOF,
    );
```

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

Item	LD		ST
Running result	FileRead_0	FileRead	
	xReadExe	BOOL	TRUE
	dwFileID	DWORD	4459552
	abyReadBuf	ARRAY [1..5] OF BYTE	
	abyReadBuf[1]	BYTE	12
	abyReadBuf[2]	BYTE	34
	abyReadBuf[3]	BYTE	5
	abyReadBuf[4]	BYTE	0
	abyReadBuf[5]	BYTE	0
	uiSize	UINT	3
	xReadDone	BOOL	TRUE
	xReadBusy	BOOL	FALSE
	xReadError	BOOL	FALSE
	wReadErrorID	WORD	0
	uiReadSize	UINT	3
	xReadEOF	BOOL	FALSE

■ Precautions

· Execution of this instruction is continued until processing is completed even if the value of Execute changes to FALSE or the execution time exceeds the task period. The value of Done changes to TRUE when processing is completed. Use this to confirm normal completion of processing.

· If the data is read to the end of the file and the size of the data is not evenly divisible by the size of the data type of ReadBuf[], the data that is insufficient for the data size of ReadBuf[] is discarded. The file position indicator advances to the end of the file, and the value of EOF changes to TRUE.

· When the value of Size exceeds the ReadBuf[] array, an program error occurs, Error is TRUE, and ErrorID is 16#406. · Elements beyond Size in ReadBuf[] (that is, the elements not overwritten when data is read) retain the values from before execution of this instruction.

· Execute the FileOpen instruction in advance to obtain the value for FileID.

· After this instruction is completed, save the value to EOF. The value of Done changes to TRUE.

· If ReadBuf[] is an array of structures, adjustment areas between members may be inserted depending on the composition.

· An error occurs in the following cases. The value of Error is TRUE.

1) The SD memory card is unavailable.

2) The file specified by FileID is being accessed.

3) The file specified by FileID does not exist.

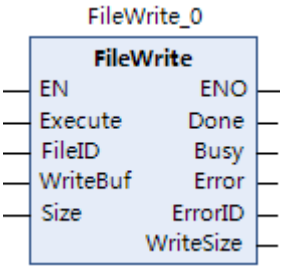
4) An error that prevents access occurs during SD memory card access.

5) The file specified by FileID is not opened in a reading mode.

4.6. 6 FileWrite

This instruction writes data to the specified file in the SD memory card.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
FileWrite	Write file	FB		FileWrite_0(Execute :=, FileID :=, WriteBuf :=, Size :=, Done =>, Busy =>, Error =>, ErrorID =>, WriteSize =>,);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
Execute	Enable	BOOL	Depends on data types	-	(Triggered by edges) Execution of the function block switch
FileID	File ID	DWORD	Depends on data types	-	ID of the file to write
WriteBuf[] Array	Write buffer	-	Depends on data types	-	Data to write
Size	Number of elements to write	UINT	Depends on data types	-	Number of elements to write

Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
Done	Function block completed state	BOOL	Depends on data types	-	Function block completed state
Busy	Function block execution flag	BOOL	Depends on data types	-	Function block execution flag
Error	Function block error flag	BOOL	Depends on data types	-	Function block error flag
ErrorID	Function block error ID	WORD	Depends on data types	-	Function block error ID
WriteSize	Number of elements actually written	UINT	Depends on data types	-	Number of elements actually written

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Execute	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
FileID	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
WriteBuf[] (array)	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√
	Arrays enumerations or structures can also be specified.																			

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Size	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-
Done	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Busy	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Error	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
ErrorID	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
WriteSize	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-

■ Function

1. This instruction writes the data written to WriteBuf[] to the position of the file position indicator in the file specified by FileID in the SD memory card.

2. The file position indicator is set at the desired location in advance with the FileSeek instruction.

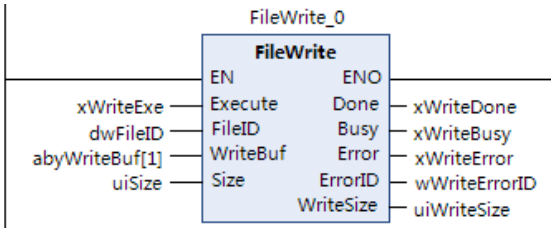
3. The amount of data that is written is the size of the data type of WriteBuf[] times Size.

4. The number of elements actually written is saved to WriteSize. In general, values of Size and ReadSize are the same.

■ Program example

Item	LD	ST

Defined variable	VAR
	FileWrite_0 : FileWrite;
	xWriteExe : BOOL;
	dwFileID : DWORD := 4459552;
	abyWriteBuf : ARRAY [1..5] OF BYTE := [12,34,5,67,89];
	uiSize : UINT := 5;
	xWriteDone : BOOL;
	xWriteBusy : BOOL;
	xWriteError : BOOL;
	wWriteErrorID : WORD;
	uiWriteSize : UINT;
	END_VAR	

Program		FileWrite_0(Execute := xWriteExe, FileID := dwFileID, WriteBuf := abyWriteBuf[1], Size := uiSize, Done => xWriteDone, Busy => xWriteBusy, Error => xWriteError, ErrorID => wWriteErrorID, WriteSize => uiWriteSize,);

Item	LD		ST
Running result	FileWrite_0	FileWrite	
	xWriteExe	BOOL	TRUE
	dwFileID	DWORD	4459552
	abyWriteBuf	ARRAY [1..5] OF BYTE	
	abyWriteBuf[1]	BYTE	12
	abyWriteBuf[2]	BYTE	34
	abyWriteBuf[3]	BYTE	5
	abyWriteBuf[4]	BYTE	67
	abyWriteBuf[5]	BYTE	89
	uiSize	UINT	5
	xWriteDone	BOOL	TRUE
	xWriteBusy	BOOL	FALSE
	xWriteError	BOOL	FALSE
	wWriteErrorID	WORD	0
	uiWriteSize	UINT	5

■ Precautions

· Execution of this instruction is continued until processing is completed even if the value of Execute changes to FALSE or the execution time exceeds the task period. The value of Done changes to TRUE when processing is completed. Use this to confirm normal completion of processing.

· When the value of Size exceeds the WriteBuf[] array, an program error occurs, Error is TRUE, and Error-ID is 16#406.

· Execute the FileOpen instruction in advance to obtain the value for FileID.

· If WriteBuf[] is an array of structures, adjustment areas between members may be inserted depending on the composition.

· An error occurs in the following cases. The value of Error is TRUE.

1) The SD memory card is unavailable.

2) The file specified by FileID is being accessed.

3) The file specified by FileID does not exist.

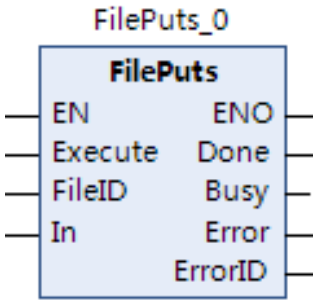
4) An error that prevents access occurs during SD memory card access.

5) The file specified by FileID is not opened in a writing mode.

4.6. 7 FilePuts

This instruction writes a text string to the specified file in the SD memory card.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
FilePuts	Put text string	FB		FilePuts_0(Execute :=, FileID :=, In :=, Done =>, Busy =>, Error =>, ErrorID =>,);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
Execute	Enable	BOOL	Depends on data types	-	(Triggered by edges) Execution of the function block switch
FileID	File ID	DWORD	Depends on data types	-	ID of the file to write
In	Text string to write	STRING	Depends on data types	-	Text string to write

Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
Done	Function block completed state	BOOL	Depends on data types	-	Function block completed state
Busy	Function block execution flag	BOOL	Depends on data types	-	Function block execution flag
Error	Function block error flag	BOOL	Depends on data types	-	Function block error flag
ErrorID	Function block error ID	WORD	Depends on data types	-	Function block error ID

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Execute	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
FileID	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
In	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√
Done	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Busy	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Error	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
ErrorID	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

■ Function

This instruction writes the text string In to the position of the file position indicator in the file specified by FileID in the SD memory card.

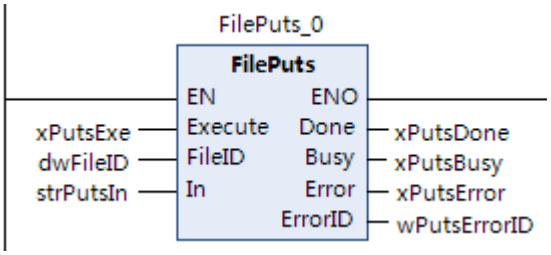
The file position indicator is set at the desired location in advance with the FileSeek instruction.

■ Program example

<table>
<tr><th>Item</th><th>LD</th><th>ST</th></tr>
<tr><td>Defined variable</td><td colspan="2">

```
VAR
    FilePuts_0      : FilePuts;
    xPutsExe        : BOOL;
    dwFileID        : DWORD := 4459552;
    strPutsIn       : STRING := '1234$R5678';
    xPutsDone       : BOOL;
    xPutsBusy       : BOOL;
    xPutsError      : BOOL;
    wPutsErrorID    : WORD;
END_VAR
```

</td></tr>
<tr><td>Program</td><td>



</td><td>

```
FilePuts_0(Execute := xPutsExe,
    FileID := dwFileID,
    In := strPutsIn,
    Done => xPutsDone,
    Busy => xPutsBusy,
    Error => xPutsError,
    ErrorID => wPutsErrorID,
    );
```

</td></tr>
<tr><td>Running result</td><td colspan="2">

FilePuts_0	FilePuts	
xPutsExe	BOOL	TRUE
dwFileID	DWORD	4459552
strPutsIn	STRING	'1234$R5678'
xPutsDone	BOOL	TRUE
xPutsBusy	BOOL	FALSE
xPutsError	BOOL	FALSE
wPutsErrorID	WORD	0

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

■ Precautions

Execution of this instruction is continued until processing is completed even if the value of Execute changes to FALSE or the execution time exceeds the task period. The value of Done changes to TRUE when processing is completed. Use this to confirm normal completion of processing.

Execute the FileOpen instruction in advance to obtain the value for FileID.

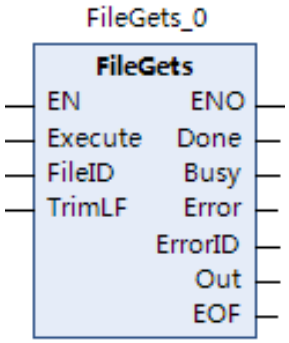
An error occurs in the following cases. The value of Error is TRUE.

1) The SD memory card is unavailable.
2) The SD memory card is write-protected.
3) There is insufficient space available in the SD memory card.
4) The file specified by FileID is being accessed.
5) The file specified by FileID does not exist.
6) An error that prevents access occurs during SD memory card access.
7) The file specified by FileID is not opened in a writing mode.

4.6. 8 FileGets

This instruction reads a text string of one line from the specified file in the SD memory card.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
FileGets	Get text string	FB		FileGets_0(Execute :=, FileID :=, TrimLF :=, Done =>, Busy =>, Error =>, ErrorID =>, Out =>, EOF =>,);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
Execute	Enable	BOOL	Depends on data types	-	(Triggered by edges) Execution of the function block switch
FileID	File ID	DWORD	Depends on data types	-	ID of the file to write
TrimLF	Line feed designation	BOOL	Depends on data types	-	Handling of the line feed code of the read text string TRUE: Delete FALSE: Do not delete

Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
Done	Function block completed state	BOOL	Depends on data types	-	Function block completed state
Busy	Function block execution flag	BOOL	Depends on data types	-	Function block execution flag
Error	Function block error flag	BOOL	Depends on data types	-	Function block error flag
ErrorID	Function block error ID	WORD	Depends on data types	-	Function block error ID
Out	Read text string	STRING	Depends on data types	-	Read text string
EOF	End of file	BOOL	Depends on data types	-	Whether the end of the file is reached TRUE: Reached FALSE: Not reached

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Execute	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
FileID	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
TrimLF	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Done	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Busy	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Error	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
ErrorID	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Out	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√
EOF	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

■ Function

This instruction reads one line of text string to the position of the file position indicator in the file specified by FileID in the SD memory card.

The file position indicator is set at the desired location in advance with the FileSeek instruction.

Line endings are determined by a line feed code. The text string that is read is written to read the text string Out.

The following three line feeds are automatically detected: CR, LF, and CR+LF. If line feed designation TrimLF is TRUE, the line feed code is deleted from the text string before it is written to Out.

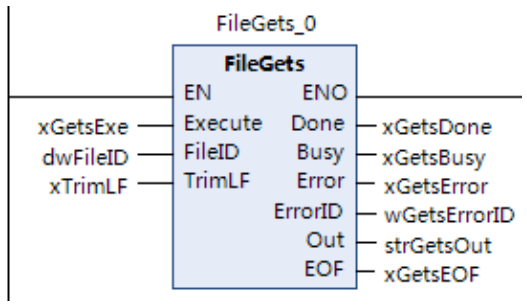
If data is read to the end of the file, EOF (end of file) changes to TRUE.

■ Program example

Item	LD	ST
Defined variable		

```
VAR
    FileGets_0      : FileGets;
    xGetsExe        : BOOL;
    dwFileID        : DWORD := 4459552;
    xTrimLF         : BOOL := FALSE;
    xGetsDone       : BOOL;
    xGetsBusy       : BOOL;
    xGetsError      : BOOL;
    wGetsErrorID    : WORD;
    strGetsOut      : STRING;
    xGetsEOF        : BOOL;
END_VAR
```

Program




```
FileGets_0(Execute := xGetsExe,
    FileID := dwFileID,
    TrimLF := xTrimLF,
    Done => xGetsDone,
    Busy => xGetsBusy,
    Error => xGetsError,
    ErrorID => wGetsErrorID,
    Out => strGetsOut,
    EOF => xGetsEOF,
    );
```

Running result

FileGets_0	FileGets	
xGetsExe	BOOL	TRUE
dwFileID	DWORD	4459552
xTrimLF	BOOL	FALSE
xGetsDone	BOOL	TRUE
xGetsBusy	BOOL	FALSE
xGetsError	BOOL	FALSE
wGetsErrorID	WORD	0
strGetsOut	STRING	'1234$R'
xGetsEOF	BOOL	FALSE

■ Precautions

Execution of this instruction is continued until processing is completed even if the value of Execute changes to FALSE or the execution time exceeds the task period. The value of Done changes to TRUE when processing is completed. Use this to confirm normal completion of processing.

If the length of the one-line text string exceeds 1985 bytes, the first 1985 bytes of the text string and the NULL text string at the end are stored in Out.

Execute the FileOpen instruction in advance to obtain the value for FileID.

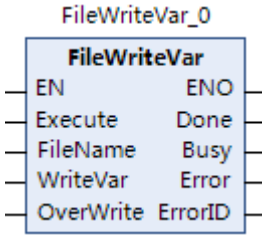
An error occurs in the following cases. The value of Error is TRUE.

1) The SD memory card is unavailable.
2) The file specified by FileID is being accessed.
3) The file specified by FileID does not exist.
4) An error that prevents access occurs during SD memory card access.
5) The file specified by FileID is not opened in a reading mode.

4.6. 9 FileWriteVar

This instruction writes the value of a variable to the specified file in the SD memory card in binary format.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
FileWriteVar	Write variable to file	FB		FileWriteVar_0(Execute :=, FileName :=, WriteVar :=, OverWrite :=, Done =>, Busy =>, Error =>, ErrorID =>,);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
Execute	Enable	BOOL	Depends on data types	-	(Triggered by edges) Execution of the function block switch
FileName	Specified file name	STRING	A maximum of 1985 characters	-	Name of the file to which to write variable data
WriteVar	Specified variable	-	Depends on data types	-	Variable to write
OverWrite	Overwrite enable	BOOL	Depends on data types	-	TRUE: Enable overwrite FALSE: Prohibit overwrite

Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
Done	Function block completed state	BOOL	Depends on data types	-	Function block completed state
Busy	Function block execution flag	BOOL	Depends on data types	-	Function block execution flag
Error	Function block error flag	BOOL	Depends on data types	-	Function block error flag
ErrorID	Function block error ID	WORD	Depends on data types	-	Function block error ID

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Execute	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
FileName	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√
WriteVar	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√
	An enumeration, an array, an array element, a structure, or a structure member can also be specified.																			
OverWrite	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Done	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Busy	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Error	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
ErrorID	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

■ Function

This instruction writes the value of variable WriteVar to the file specified by FileName in the SD memory card in binary format.

If a file with the name specified by FileName does not exist in the SD memory card, it is created.

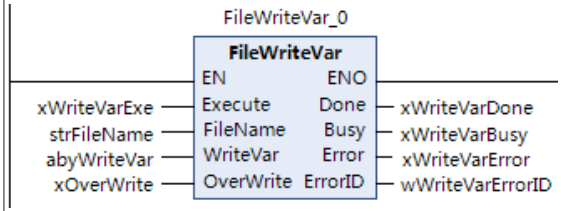
If a file with the name specified by FileName already exists in the SD memory card, the following processing is performed depending on the value of OverWrite.

Value of OverWrite	Operation
TRUE	The file is overwritten.
FALSE	The file is not overwritten and an error occurs.

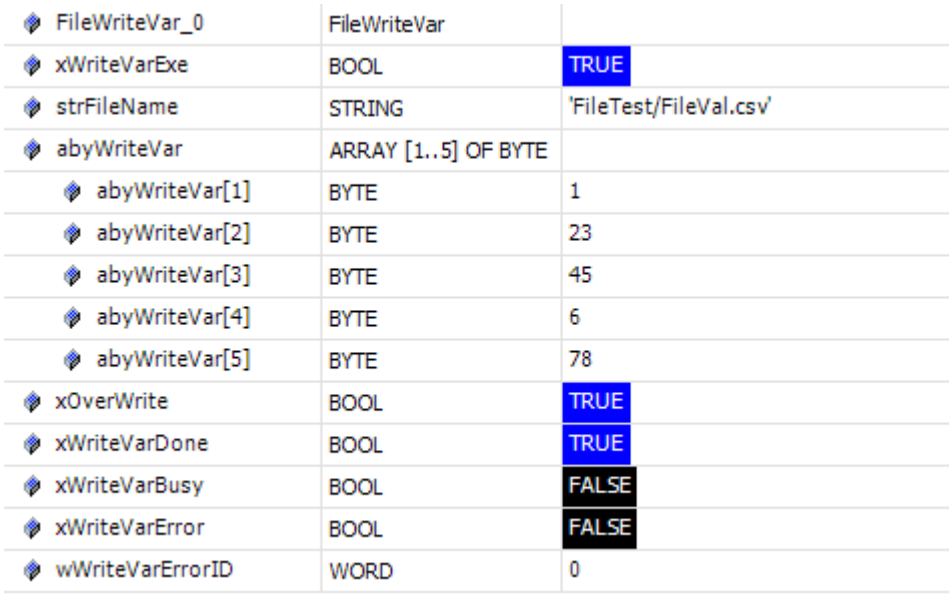
■ Program example

Item	LD	ST
Defined variable		
Program		

```
VAR
    FileWriteVar_0      : FileWriteVar;
    xWriteVarExe        : BOOL;
    strFileName         : STRING := 'FileTest/FileVal.csv';
    abyWriteVar         : ARRAY [1..5] OF BYTE := [1,23,45,6,78];
    xOverWrite          : BOOL;
    xWriteVarDone       : BOOL;
    xWriteVarBusy       : BOOL;
    xWriteVarError      : BOOL;
    wWriteVarErrorID    : WORD;
END_VAR
```




```
FileWriteVar_0(Execute := xWriteVarExe,
    FileName := strFileName,
    WriteVar := abyWriteVar,
    OverWrite := xOverWrite,
    Done => xWriteVarDone,
    Busy => xWriteVarBusy,
    Error => xWriteVarError,
    ErrorID => wWriteVarErrorID,
    );
```

Item	LD	ST
Running result		

■ Precautions

Execution of this instruction is continued until processing is completed even if the value of Execute changes to FALSE or the execution time exceeds the task period. The value of Done changes to TRUE when processing is completed. Use this to confirm normal completion of processing.

Execute the FileOpen instruction in advance to obtain the value for FileID.

If WriteVar is a structure, adjustment areas between members may be inserted depending on the composition.

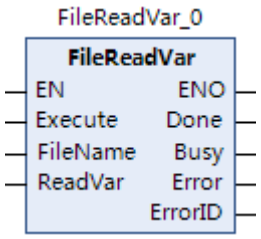
An error occurs in the following cases. The value of Error is TRUE.

1) The SD memory card is unavailable.
2) The SD memory card is write-protected.
3) There is insufficient space available in the SD memory card.
4) The file specified by FileID is being accessed.
5) The value of FileName is not a valid file name.
6) The maximum number of files or directories is exceeded.
7) A file with the name specified by FileName exits and the file is being accessed.
8) A file with the name specified by FileName exits and the value of OverWrite is FALSE.
9) A file with the name specified by FileName exits and is read-only.
10) An error that prevents access occurs during SD memory card access.

4.6. 10 FileReadVar

This instruction reads the values of the specified file in the SD memory card as binary data and writes the data to a variable.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
FileReadVar	Read variable from file	FB		FileReadVar_0(Execute :=, FileName :=, ReadVar :=, Done =>, Busy =>, Error =>, ErrorID =>,);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
Execute	Enable	BOOL	Depends on data types	-	(Triggered by edges) Execution of the function block switch
FileName	Specified file name	STRING	A maximum of 1985 characters	-	Name of the file from which to read variable data
ReadVar	Position variable to read	-	Depends on data types	-	Position variable from which to read variable data

Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
Done	Function block completed state	BOOL	Depends on data types	-	Function block completed state
Busy	Function block execution flag	BOOL	Depends on data types	-	Function block execution flag
Error	Function block error flag	BOOL	Depends on data types	-	Function block error flag
ErrorID	Function block error ID	WORD	Depends on data types	-	Function block error ID

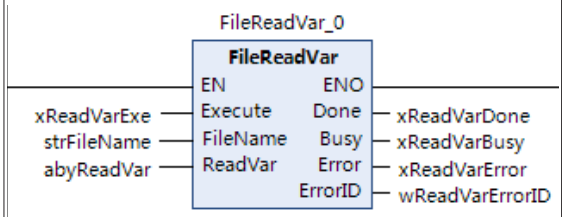
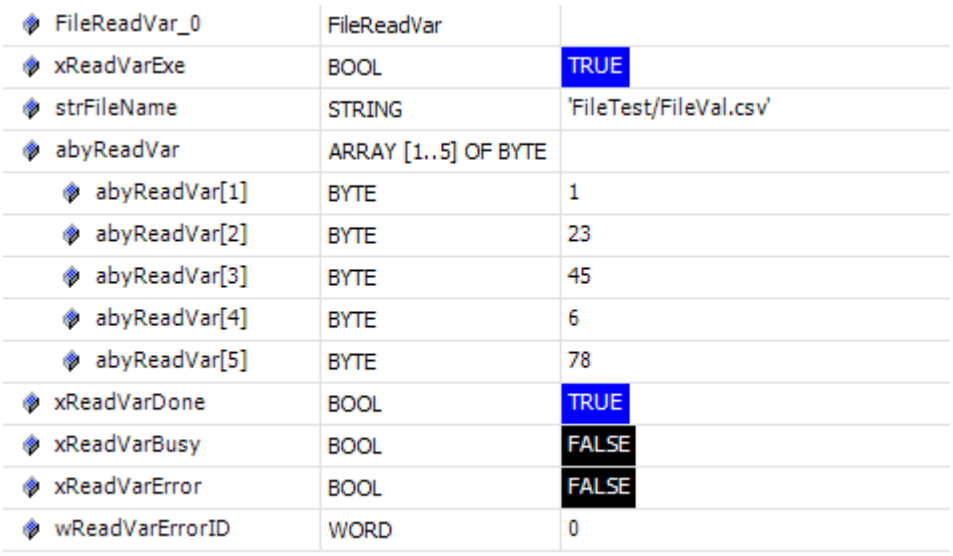
	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Execute	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
FileName	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√
ReadVar	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√
	An enumeration, an array, an array element, a structure, or a structure member can also be specified.																			
Done	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Busy	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Error	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
ErrorID	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

■ Function

This instruction reads variables from the file specified by FileName in the SD memory card in binary format and assigns the read data to the position variable to read ReadVar.

■ Program example

Item	LD	ST
Defined variable	VAR FileReadVar_0 : FileReadVar; xReadVarExe : BOOL; strFileName : STRING := 'FileTest/FileVal.csv'; abyReadVar : ARRAY [1..5] OF BYTE; xReadVarDone : BOOL; xReadVarBusy : BOOL; xReadVarError : BOOL; wReadVarErrorID : WORD; END_VAR	
Program		FileReadVar_0(Execute := xReadVarExe, FileName := strFileName, ReadVar := abyReadVar, Done => xReadVarDone, Busy => xReadVarBusy, Error => xReadVarError, ErrorID => wReadVarErrorID,);
Running result		

■ Precautions

Execution of this instruction is continued until processing is completed even if the value of Execute changes to FALSE or the execution time exceeds the task period. The value of Done changes to TRUE when processing is completed. Use this to confirm normal completion of processing.

If the size of the specified file is greater than that of ReadVar, no error occurs and only the data of the size of ReadVar is read.

If the size of the specified file is less than that of ReadVar, no error occurs and only the data of the size of the specified file is read. The remaining area in ReadVar retains the values used before execution of this instruction.

If ReadVar is a structure, adjustment areas between members may be inserted depending on the composition.

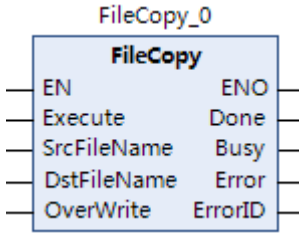
An error occurs in the following cases. The value of Error is TRUE.

1) The SD memory card is unavailable.
2) The value of FileName is not a valid file name.
3) The file specified by FileName does not exist.
4) The file specified by FileName is being accessed.
5) An error that prevents access occurs during SD memory card access.

4.6. 11 FileCopy

This instruction copies the specified file in the SD memory card.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
FileCopy	Copy file	FB		FileCopy_0(Execute :=, SrcFileName :=, DstFileName :=, OverWrite :=, Done =>, Busy =>, Error =>, ErrorID =>,);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
Execute	Enable	BOOL	Depends on data types	-	(Triggered by edges) Execution of the function block switch
SrcFileName	Name of source file to copy	STRING	A maximum of 1985 characters	-	Name of source file to copy
DstFileName	Name of destination file to paste	STRING	A maximum of 1985 characters	-	Name of destination file to paste
OverWrite	Overwrite enable	BOOL	Depends on data types	-	TRUE: Enable overwrite FALSE: Prohibit overwrite

Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
Done	Function block completed state	BOOL	Depends on data types	-	Function block completed state
Busy	Function block execution flag	BOOL	Depends on data types	-	Function block execution flag
Error	Function block error flag	BOOL	Depends on data types	-	Function block error flag
ErrorID	Function block error ID	WORD	Depends on data types	-	Function block error ID

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Execute	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
SrcFileName	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√
DstFileName	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√
OverWrite	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Done	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Busy	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Error	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
ErrorID	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

■ Function

This instruction copies the source file specified by SrcFileName in the SD memory card to DstFileName.

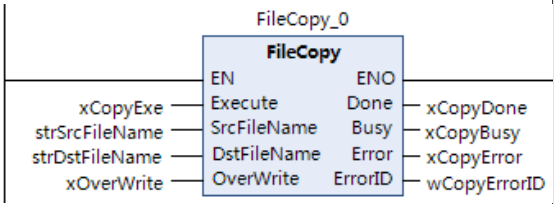
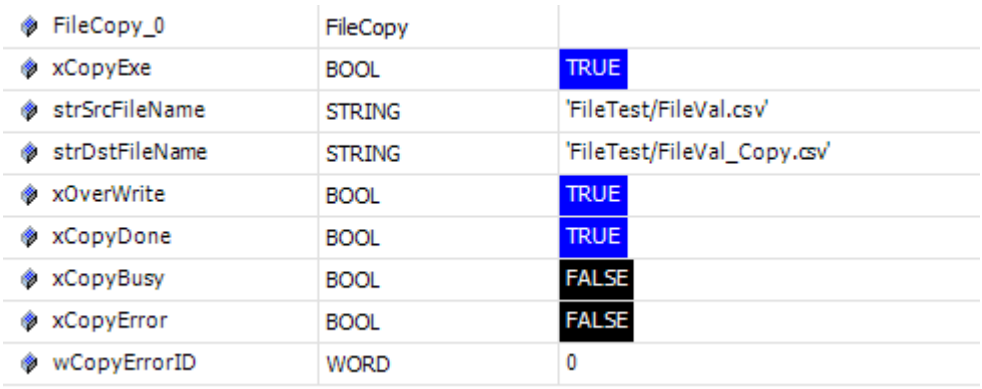
If a file with the name specified by DstFileName does not exist in the SD memory card, it is created.

If a file with the name specified by DstFileName already exists in the SD memory card, the following processing is performed depending on the value of OverWrite.

Value of OverWrite	Operation
TRUE	The file is overwritten.
FALSE	The file is not overwritten and an error occurs.

■ Program example

Item	LD	ST
Defined variable	VAR FileCopy_0 : FileCopy; xCopyExe : BOOL; strSrcFileName : STRING := 'FileTest/FileVal.csv'; strDstFileName : STRING := 'FileTest/FileVal_Copy.csv'; xOverWrite : BOOL; xCopyDone : BOOL; xCopyBusy : BOOL; xCopyError : BOOL; wCopyErrorID : WORD; END_VAR	

<table>
<tr><th>Item</th><th>LD</th><th>ST</th></tr>
<tr><td>Program</td><td> 
</td><td><pre>FileCopy_0(Execute := xCopyExe,
    SrcFileName := strSrcFileName,
    DstFileName := strDstFileName,
    OverWrite := xOverWrite,
    Done => xCopyDone,
    Busy => xCopyBusy,
    Error => xCopyError,
    ErrorID => wCopyErrorID,
    );</pre></td></tr>
<tr><td>Running result</td><td colspan="2"> 
</td></tr>
</table>

■ Precautions

Execution of this instruction is continued until processing is completed even if the value of Execute changes to FALSE or the execution time exceeds the task period. The value of Done changes to TRUE when processing is completed. Use this to confirm normal completion of processing.

When the file is opened, remove the SD memory card. Then the file remains the open state. When you insert the SD memory card again, you cannot read or write the file. To read and write the file, re-open the file.

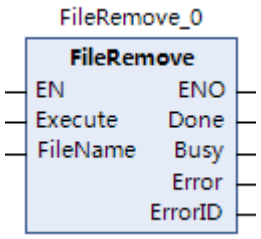
An error occurs in the following cases. The value of Error is TRUE.

1) The SD memory card is unavailable.
2) The SD memory card is write-protected.
3) There is insufficient space available in the SD memory card.
4) The file specified by SrcFileName does not exist.
5) The value of SrcFileName is not a valid file name.
6) The value of DstFileName is not a valid file name.
7) The maximum number of files or directories is exceeded.
8) A file with the name specified by DstFileName exits and the file is being accessed.
9) A file with the name specified by DstFileName exits and the value of OverWrite is FALSE.
10) A file with the name specified by DstFileName exits and is read-only.
11) An error that prevents access occurs during SD memory card access.

4.6. 12 FileRemove

This instruction deletes the specified file in the SD memory card.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
FileRemove	Delete file	FB		FileRemove_0(Execute :=, FileName :=, Done =>, Busy =>, Error =>, ErrorID =>,);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
Execute	Enable	BOOL	Depends on data types	-	(Triggered by edges) Execution of the function block switch
FileName	Specified file name	STRING	A maximum of 1985 characters	-	Name of the file to delete

Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
Done	Function block completed state	BOOL	Depends on data types	-	Function block completed state
Busy	Function block execution flag	BOOL	Depends on data types	-	Function block execution flag
Error	Function block error flag	BOOL	Depends on data types	-	Function block error flag
ErrorID	Function block error ID	WORD	Depends on data types	-	Function block error ID

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Execute	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
FileName	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√
Done	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Busy	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Error	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
ErrorID	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

■ Function

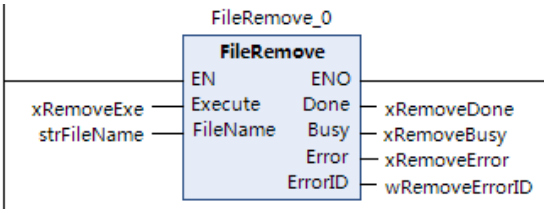
This instruction deletes the file specified by file name FileName from the SD memory card.

■ Program example

<table>
<tr><th>Item</th><th>LD</th><th>ST</th></tr>
<tr><td>Defined variable</td><td colspan="2">

```
VAR
    FileRemove_0    : FileRemove;
    xRemoveExe      : BOOL;
    strFileName     : STRING := 'FileTest/FileVal_Copy.csv';
    xRemoveDone     : BOOL;
    xRemoveBusy     : BOOL;
    xRemoveError    : BOOL;
    wRemoveErrorID  : WORD;
END_VAR
```

</td></tr>
<tr><td>Program</td><td>



</td><td>

```
FileRemove_0(Execute := xRemoveExe,
    FileName := strFileName,
    Done => xRemoveDone,
    Busy => xRemoveBusy,
    Error => xRemoveError,
    ErrorID => wRemoveErrorID,
    );
```

</td></tr>
<tr><td>Running result</td><td colspan="2">

FileRemove_0	FileRemove	
xRemoveExe	BOOL	TRUE
strFileName	STRING	'FileTest/FileVal_Copy.csv'
xRemoveDone	BOOL	TRUE
xRemoveBusy	BOOL	FALSE
xRemoveError	BOOL	FALSE
wRemoveErrorID	WORD	0

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

■ Precautions

Execution of this instruction is continued until processing is completed even if the value of Execute changes to FALSE or the execution time exceeds the task period. The value of Done changes to TRUE when processing is completed. Use this to confirm normal completion of processing.

When the file is opened, remove the SD memory card. Then the file remains the open state. When you insert the SD memory card again, you cannot read or write the file. To read and write the file, re-open the file.

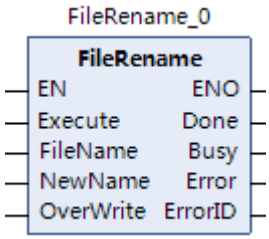
An error occurs in the following cases. The value of Error is TRUE.

1) The SD memory card is unavailable.
2) The SD memory card is write-protected.
3) The file specified by FileName does not exist.
4) The file specified by FileName is being accessed.
5) A file with the name specified by FileName exits and is read-only.
6) An error that prevents access occurs during SD memory card access.

4.6. 13 FileRename

This instruction changes the name of the specified file or directory in the SD memory card.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
FileRename	Change file name	FB		FileRename_0(Execute :=, FileName :=, NewName :=, OverWrite :=, Done =>, Busy =>, Error =>, ErrorID =>,);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
Execute	Enable	BOOL	Depends on data types	-	(Triggered by edges) Execution of the function block switch
FileName	Original file name	STRING	A maximum of 1985 characters	-	Original file name
NewName	New file name	STRING	A maximum of 1985 characters	-	New file name
OverWrite	Overwrite enable	BOOL	Depends on data types	-	TRUE: Enable overwrite FALSE: Prohibit overwrite

Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
Done	Function block completed state	BOOL	Depends on data types	-	Function block completed state
Busy	Function block execution flag	BOOL	Depends on data types	-	Function block execution flag
Error	Function block error flag	BOOL	Depends on data types	-	Function block error flag
ErrorID	Function block error ID	WORD	Depends on data types	-	Function block error ID

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Execute	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
FileName	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√
NewName	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√
OverWrite	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Done	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Busy	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Error	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
ErrorID	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

■ Function

This instruction changes the original name of the file or directory specified by FileName in the SD memory card to new file name specified by NewName.

If a file with the name specified by NewName does not exist in the SD memory card, it is created.

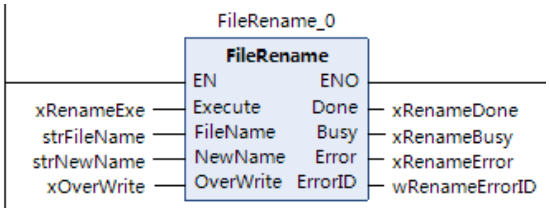
If a file with the name specified by NewName already exists in the SD memory card, the following processing is performed depending on the value of OverWrite.

Value of OverWrite	Operation
TRUE	The file or directory is overwritten.
FALSE	The file or directory is not overwritten and an error occurs.

■ Program example

Item	LD	ST
Defined variable		
Program		

```
VAR
    FileRename_0    : FileRename;
    xRenameExe      : BOOL;
    strFileName     : STRING := 'FileTest/FileVal.csv';
    strNewName      : STRING := 'FileTest/FileVal_Rename.csv';
    xOverWrite      : BOOL;
    xRenameDone     : BOOL;
    xRenameBusy     : BOOL;
    xRenameError    : BOOL;
    wRenameErrorID  : WORD;
END_VAR
```




```
FileRename_0(Execute := xRenameExe,
    FileName := strFileName,
    NewName := strNewName,
    OverWrite := xOverWrite,
    Done => xRenameDone,
    Busy => xRenameBusy,
    Error => xRenameError,
    ErrorID => wRenameErrorID,
    );
```

Item	LD	ST
Running result	FileRename_0 FileRename xRenameExe BOOL TRUE strFileName STRING 'FileTest/FileVal.csv' strNewName STRING 'FileTest/FileVal_Rename.csv' xOverWrite BOOL FALSE xRenameDone BOOL TRUE xRenameBusy BOOL FALSE xRenameError BOOL FALSE wRenameErrorID WORD 0	

■ Precautions

Execution of this instruction is continued until processing is completed even if the value of Execute changes to FALSE or the execution time exceeds the task period. The value of Done changes to TRUE when processing is completed. Use this to confirm normal completion of processing.

If the directories are different for FileName and NewName, the file is moved to the directory specified by NewName.

When the file is opened, remove the SD memory card. Then the file remains the open state. When you insert the SD memory card again, you cannot read or write the file. To read and write the file, re-open the file.

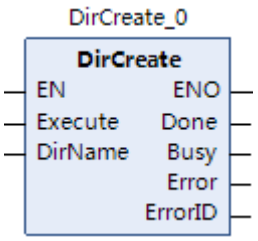
An error occurs in the following cases. The value of Error is TRUE.

1) The SD memory card is unavailable.
2) The SD memory card is write-protected.
3) The file specified by FileName does not exist.
4) The value of FileName or NewName is not a valid file name.
5) The file specified by FileName is being accessed.
6) The maximum number of files or directories is exceeded.
7) A file with the name specified by NewName exits and the value of OverWrite is FALSE.
8) A file with the name specified by DstFileName exits, the value of OverWrite is TRUE, and the file is read-only.
9) An error that prevents access occurs during SD memory card access.

4.6. 14 DirCreate

This instruction creates a directory with the specified name in the SD memory card.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
DirCreate	Create directory	FB		DirCreate_0(Execute :=, DirName :=, Done =>, Busy =>, Error =>, ErrorID =>,);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
Execute	Enable	BOOL	Depends on data types	-	(Triggered by edges) Execution of the function block switch
DirName	Specified directory name	STRING	A maximum of 1985 characters	-	Name of the directory to create

Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
Done	Function block completed state	BOOL	Depends on data types	-	Function block completed state
Busy	Function block execution flag	BOOL	Depends on data types	-	Function block execution flag
Error	Function block error flag	BOOL	Depends on data types	-	Function block error flag
ErrorID	Function block error ID	WORD	Depends on data types	-	Function block error ID

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Execute	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
DirName	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√
Done	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Busy	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Error	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
ErrorID	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Error	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
ErrorID	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

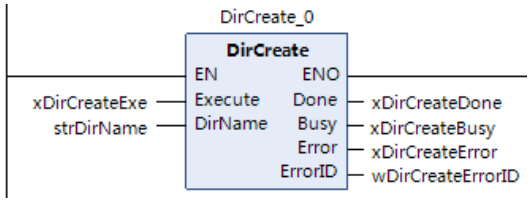
■ Function

This instruction creates a directory specified by DirName in the SD memory card.

■ Program example

Item	LD	ST
Defined variable		
Program		
Running result		

```
VAR
    DirCreate_0         : DirCreate;
    xDirCreateExe       : BOOL;
    strDirName          : STRING := 'FileTest/DirNew';
    xDirCreateDone      : BOOL;
    xDirCreateBusy      : BOOL;
    xDirCreateError     : BOOL;
    wDirCreateErrorID   : WORD;
END_VAR
```




```
DirCreate_0(Execute := xDirCreateExe,
    DirName := strDirName,
    Done => xDirCreateDone,
    Busy => xDirCreateBusy,
    Error => xDirCreateError,
    ErrorID => wDirCreateErrorID,
    );
```

DirCreate_0	DirCreate	
xDirCreateExe	BOOL	TRUE
strDirName	STRING	'FileTest/DirNew'
xDirCreateDone	BOOL	TRUE
xDirCreateBusy	BOOL	FALSE
xDirCreateError	BOOL	FALSE
wDirCreateErrorID	WORD	0

■ Precautions

Execution of this instruction is continued until processing is completed even if the value of Execute changes to FALSE or the execution time exceeds the task period. The value of Done changes to TRUE when processing is completed. Use this to confirm normal completion of processing.

When the file is opened, remove the SD memory card. Then the file remains the open state. When you insert the SD memory card again, you cannot read or write the file. To read and write the file, re-open the file.

An error occurs in the following cases. The value of Error is TRUE.

1) The SD memory card is unavailable.

2) The SD memory card is write-protected.

3) There is insufficient space available in the SD memory card.

4) The maximum number of directories is exceeded.

5) The directory specified by DirName exists.

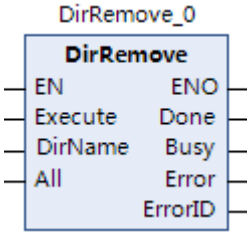
6) The value of DirName is not a valid directory name.

7) An error that prevents access occurs during SD memory card access.

4.6. 15 DirRemove

This instruction deletes the specified directory in the SD memory card.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
DirRemove	Delete directory	FB		DirRemove_0(Execute :=, DirName :=, All :=, Done =>, Busy =>, Error =>, ErrorID =>,);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
Execute	Enable	BOOL	Depends on data types	-	(Triggered by edges) Execution of the function block switch
FileName	Directory to delete	STRING	A maximum of 1985 characters	-	Directory to delete
All	All designation	BOOL	Depends on data types	-	Whether to delete a directory that contains files and subdirectories TRUE: Delete FALSE: Do not delete

Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
Done	Function block completed state	BOOL	Depends on data types	-	Function block completed state
Busy	Function block execution flag	BOOL	Depends on data types	-	Function block execution flag
Error	Function block error flag	BOOL	Depends on data types	-	Function block error flag
ErrorID	Function block error ID	WORD	Depends on data types	-	Function block error ID

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Execute	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
DirName	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√
All	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Done	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Busy	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Error	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
ErrorID	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

■ Function

This instruction deletes the directory specified by DirName in the SD memory card.

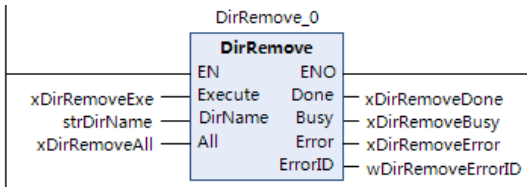
If there are files or subdirectories in the specified directory, the following processing is performed according to the value of All.

Value of All	Operation
TRUE	The specified directory is deleted, including all files and subdirectories in it.
FALSE	The specified directory is not deleted and an error occurs.

■ Program example

Item	LD	ST
Defined variable		
Program		

```
VAR
    DirRemove_0         : DirRemove;
    xDirRemoveExe       : BOOL;
    strDirName          : STRING := 'FileTest/DirNew';
    xDirRemoveAll       : BOOL;
    xDirRemoveDone      : BOOL;
    xDirRemoveBusy      : BOOL;
    xDirRemoveError     : BOOL;
    wDirRemoveErrorID   : WORD;
END_VAR
```




```
DirRemove_0(Execute := xDirRemoveExe,
    DirName := strDirName,
    All := xDirRemoveAll,
    Done => xDirRemoveDone,
    Busy => xDirRemoveBusy,
    Error => xDirRemoveError,
    ErrorID => wDirRemoveErrorID,
    );
```

Running result	DirRemove_0	DirRemove	
	xDirRemoveExe	BOOL	TRUE
	strDirName	STRING	'FileTest/DirNew'
	xDirRemoveAll	BOOL	FALSE
	xDirRemoveDone	BOOL	TRUE
	xDirRemoveBusy	BOOL	FALSE
	xDirRemoveError	BOOL	FALSE
	wDirRemoveErrorID	WORD	0

■ Precautions

Execution of this instruction is continued until processing is completed even if the value of Execute changes to FALSE or the execution time exceeds the task period. The value of Done changes to TRUE when processing is completed. Use this to confirm normal completion of processing.

When the file is opened, remove the SD memory card. Then the file remains the open state. When you insert the SD memory card again, you cannot read or write the file. To read and write the file, re-open the file.

An error occurs in the following cases. The value of Error is TRUE.

1) The SD memory card is unavailable.
2) The SD memory card is write-protected.
3) If the value of All is TRUE and the directory specified by DirName is being accessed by another instruction.
4) If the value of All is FALSE and the directory specified by DirName contains a file or subdirectory.
5) The directory specified by DirName is read-only.
6) The file specified by DirName does not exist.
7) An error that prevents access occurs during SD memory card access.

4.6. 16 FileOpen2

This instruction opens the specified file in the SD memory card.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
FileOpen2	Open file	FB	FileOpen2_0 FileOpen2 EN ENO Execute Done FileName Busy Mode Error ErrorID FileID	FileOpen2_0(Execute :=, FileName :=, Mode :=, Done =>, Busy =>, Error =>, ErrorID =>, FileID =>,);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
Execute	Enable	BOOL	Depends on data types	-	(Triggered by edges) Execution of the function block switch
FileName	Specified file name	STRING	Maximum 1,985 bytes	-	Name of the file to open
Mode	Open mode	ACCESS_MODE	Depends on data types	-	Mode in which to open file

Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
Done	Function block completed state	BOOL	Depends on data types	-	Function block completed state
Busy	Function block execution flag	BOOL	Depends on data types	-	Function block execution flag
Error	Function block error flag	BOOL	Depends on data types	-	Function block error flag
ErrorID	Function block error ID	HOU_ERROR	Depends on data types	-	Function block error ID
FileID	File ID	LWORD	Depends on data types	-	ID of the opened file

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Execute	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
FileName	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√
Mode	See "Function" for the enumerators for the enumerated type ACCESS_MODE.																			
Done	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Busy	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Error	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
ErrorID	See "Function" for the enumerators for the enumerated type HOU_ERROR.																			
FileID	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

■ Function

This instruction opens the file specified by FileName in the SD memory card in the mode specified by Mode.

The result outputs the file ID specified by FileID. FileID is used to specify the file in other instructions, such as FileRead2 and FileWrite2.

The data type of Mode is enumerated type ACCESS_MODE. The meanings of the enumerators are as follows.

Enumerator	Meaning
AM_READ	If the file exists in the target path, the file is opened in read-only mode. If the file does not exist, an error value is returned.

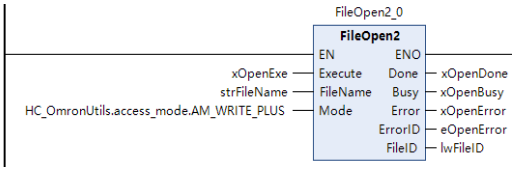
Enumerator	Meaning
AM_WRITE	If the file exists in the target path, the file is opened in re-write mode. If the file does not exist, a new file with the write permission is automatically created.
AM_APPEND	If the file exists in the target path, open the file in write-only mode and the content is appended to the end of the file. If the file does not exist, an error value is returned.
AM_READ_PLUS	If the file exists in the target path, open the file in read-write mode. If the file does not exist, an error value is returned.
AM_WRITE_PLUS	If the file exists in the target path, the file is opened in re-read-write mode. If the file does not exist, a new file with the read/write permission is automatically created.
AM_APPEND_PLUS	If the file exists in the target path, the file is opened in read-write mode and the content is appended to the end of the file. If the file does not exist, a new file with the read/write permission is automatically created.

■ Program example

<table>
<tr><th>Item</th><th>LD</th><th>ST</th></tr>
<tr><td>Defined variable</td><td colspan="2">

```
VAR
    FileOpen2_0 : FileOpen2;
    xOpenExe    : BOOL;
    strFileName : STRING(1985) := 'FileTest/File.txt';
    xOpenDone   : BOOL;
    xOpenBusy   : BOOL;
    xOpenError  : BOOL;
    eOpenError  : HOU_ERROR;
    lwFileID    : LWORD;
END_VAR
```

</td></tr>
<tr><td>Program</td><td>



</td><td>

```
FileOpen2_0(
    Execute:= xOpenExe,
    FileName:= strFileName,
    Mode:= HC_OmronUtils.access_mode.AM_WRITE_PLUS,
    Done=> xOpenDone,
    Busy=> xOpenBusy,
    Error=> xOpenError,
    ErrorID=> eOpenError,
    FileID=> lwFileID);
```

</td></tr>
<tr><td>Running result</td><td colspan="2">

FileOpen2_0	FileOpen2	
xOpenExe	BOOL	TRUE
strFileName	STRING(1985)	'FileTest/File.txt'
xOpenDone	BOOL	TRUE
xOpenBusy	BOOL	FALSE
xOpenError	BOOL	FALSE
eOpenError	HOU_ERROR	HOU_NO_ERROR
lwFileID	LWORD	3031437288

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

■ Precautions

Execution of this instruction is continued until processing is completed even if the value of Execute changes to FALSE or the execution time exceeds the task period. The value of Done changes to TRUE when processing is completed. Use this to confirm normal completion of processing.

This instruction must be executed before the FileSeek2, FileRead2, FileWrite2, FileGets2, and FilePuts2 instructions.

After using the file opened by this instruction, execute the FileClose2 instruction to close the file.

After this instruction is completed, save the value to FileID. The value of Done changes to TRUE.

When the file is opened, remove the SD memory card. Then the file remains the open state. When you insert the SD memory card again, you cannot read or write the file. To read and write the file, re-open the file.

An error occurs in the following cases. The value of Error is TRUE.

1) The SD memory card is unavailable.
2) The SD memory card is write-protected.
3) The value of Mode is AM_READ, AM_APPEND, or AM_READ_PLUS, and the file specified by FileName does not exist.
4) The value of FileName is not a valid file name.
5) The maximum number of files or directories is exceeded.
6) The file specified by FileName is being accessed.
7) The file specified by FileName is read-only.
8) An error that prevents access occurs during SD memory card access.

■ Error ID

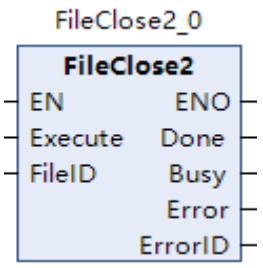
eErrorID	Value	Description
HOU_NO_ERROR	0	No error
HOU_INTERNAL_ERROR	1	General internal error
HOU_INVALIDPARAMETER_ERROR	2	Invalid parameter of the instruction
HOU_NOTINITIALIZED_ERROR	3	Component not initialized, failure to execute the instruction
HOU_VERSION_ERROR	4	Version conflict
HOU_TIMEOUT_ERROR	5	Instruction timeout
HOU_NOBUFFER_ERROR	6	Insufficient memory, failure to execute the instruction
HOU_PENDING_ERROR	10	Instruction calling not completed
HOU_NUMPENDING_ERROR	11	Excessive instruction calls
HOU_NOTIMPLEMENTED_ERROR	12	Instruction not implemented
HOU_INVALIDID_ERROR	13	Invalid ID input
HOU_OVERFLOW_ERROR	14	Integer overflow
HOU_BUFFERSIZE_ERROR	15	Buffer too small or invalid
HOU_NOOBJECT_ERROR	16	Input named object not existing
HOU_NOMEMORY_ERROR	17	No heap memory available
HOU_DUPLICATE_ERROR	18	Another object with the same name existing
HOU_MEMORYOVERWRITE_ERROR	19	Heap memory write out of bounds, memory overwrite error
HOU_INVALIDHANDLE_ERROR	20	Invalid object handle
HOU_ENDOFOBJECT_ERROR	21	Object end reached
HOU_NOCHANGE_ERROR	22	No change
HOU_INVALIDINTERFACE_ERROR	23	Invalid or unknown interface
HOU_NOTSUPPORTED_ERROR	24	Function not supported
HOU_NOACCESSRIGHTS_ERROR	25	No access permission
HOU_OUTOFLIMITS_ERROR	26	Resource limit exceeded

eErrorID	Value	Description
HOU_ENTRIESREMAINING_ERROR	27	Buffer limit causing transmission incompletion
HOU_INVALIDSESSIONID_ERROR	28	Invalid online session ID
HOU_EXCEPTION_ERROR	29	Exception
HOU_INVALIDPATHNAME_ERROR	33	Path name invalid
HOU_INVALIDFILENAME_ERROR	34	File name invalid
HOU_INVALIDLISTMODE_ERROR	35	Invalid control command
HOU_FILEOVERWRITE_ERROR	50	File overwrite error
HOU_OVERBUFFERSIZE_ERROR	1030	Data area too small
HOU_SAMEFILENAME_ERROR	1403	File with the same name existing

4.6. 17 FileClose2

This instruction closes the specified file in the SD memory card.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
FileClose2	Close file	FB		FileClose2_0(Execute :=, FileID :=, Done =>, Busy =>, Error =>, ErrorID =>,);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
Execute	Enable	BOOL	Depends on data types	-	(Triggered by edges) Execution of the function block switch
FileID	File ID	LWORD	Depends on data types	-	ID of the file to close

Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
Done	Function block completed state	BOOL	Depends on data types	-	Function block completed state
Busy	Function block execution flag	BOOL	Depends on data types	-	Function block execution flag
Error	Function block error flag	BOOL	Depends on data types	-	Function block error flag
ErrorID	Function block error ID	HOU_ERROR	Depends on data types	-	Function block error ID

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Execute	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
FileID	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√
Done	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Busy	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Error	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
ErrorID	See "Function" for the enumerators for the enumerated type HOU_ERROR.																			

■ Function

This instruction closes the file specified by FileID in the SD memory card.

■ Program example

Item	LD	ST
Defined variable		VAR FileClose2_0 : FileClose2; xCloseExe : BOOL; lwFileID : LWORD; xCloseDone : BOOL; xCloseBusy : BOOL; xCloseError : BOOL; eCloseErrorID : HOU_ERROR; END_VAR
Program	FileClose2_0 FileClose2 EN ENO xCloseExe — Execute Done — xCloseDone lwFileID — FileID Busy — xCloseBusy Error — xCloseError ErrorID — eCloseErrorID	FileClose2_0(Execute:= xCloseExe, FileID:= lwFileID, Done=> xCloseDone, Busy=> xCloseBusy, Error=> xCloseError, ErrorID=> eCloseErrorID);
Running result	FileClose2_0 FileClose2 xCloseExe BOOL TRUE xCloseDone BOOL TRUE xCloseBusy BOOL FALSE xCloseError BOOL FALSE eCloseErrorID HOU_ERROR HOU_NO_ERROR	

■ Precautions

Do not execute this instruction on a file for multiple consecutive times. Otherwise, the PLC fails to run and even breaks down.

Execution of this instruction is continued until processing is completed even if the value of Execute changes to FALSE or the execution time exceeds the task period. The value of Done changes to TRUE when processing is completed. Use this to confirm normal completion of processing.

Execute the FileOpen2 instruction in advance to obtain the value for FileID.

After using the file opened by the FileOpen2 instruction, execute this instruction to close the file. Otherwise, the result will be unexpected.

When the file is opened, remove the SD memory card. Then the file remains the open state. When you

insert the SD memory card again, you cannot read or write the file. To read and write the file, re-open the file.

An error occurs in the following cases. The value of Error is TRUE.

1) The SD memory card is unavailable.

2) The file specified by FileID is being accessed.

3) The file specified by FileID does not exist.

4) An error that prevents access occurs during SD memory card access.

■ Error ID

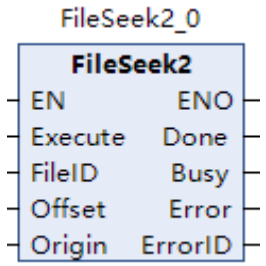
eErrorID	Value	Description
HOU_NO_ERROR	0	No error
HOU_INTERNAL_ERROR	1	General internal error
HOU_INVALIDPARAMETER_ERROR	2	Invalid parameter of the instruction
HOU_NOTINITIALIZED_ERROR	3	Component not initialized, failure to execute the instruction
HOU_VERSION_ERROR	4	Version conflict
HOU_TIMEOUT_ERROR	5	Instruction timeout
HOU_NOBUFFER_ERROR	6	Insufficient memory, failure to execute the instruction
HOU_PENDING_ERROR	10	Instruction calling not completed
HOU_NUMPENDING_ERROR	11	Excessive instruction calls
HOU_NOTIMPLEMENTED_ERROR	12	Instruction not implemented
HOU_INVALIDID_ERROR	13	Invalid ID input
HOU_OVERFLOW_ERROR	14	Integer overflow
HOU_BUFFERSIZE_ERROR	15	Buffer too small or invalid
HOU_NOOBJECT_ERROR	16	Input named object not existing
HOU_NOMEMORY_ERROR	17	No heap memory available
HOU_DUPLICATE_ERROR	18	Another object with the same name existing
HOU_MEMORYOVERWRITE_ERROR	19	Heap memory write out of bounds, memory overwrite error
HOU_INVALIDHANDLE_ERROR	20	Invalid object handle
HOU_ENDOFOBJECT_ERROR	21	Object end reached
HOU_NOCHANGE_ERROR	22	No change
HOU_INVALIDINTERFACE_ERROR	23	Invalid or unknown interface
HOU_NOTSUPPORTED_ERROR	24	Function not supported
HOU_NOACCESSRIGHTS_ERROR	25	No access permission
HOU_OUTOFLIMITS_ERROR	26	Resource limit exceeded
HOU_ENTRIESREMAINING_ERROR	27	Buffer limit causing transmission incompletion
HOU_INVALIDSESSIONID_ERROR	28	Invalid online session ID
HOU_EXCEPTION_ERROR	29	Exception
HOU_INVALIDPATHNAME_ERROR	33	Path name invalid
HOU_INVALIDFILENAME_ERROR	34	File name invalid
HOU_INVALIDLISTMODE_ERROR	35	Invalid control command
HOU_FILEOVERWRITE_ERROR	50	File overwrite error

eErrorID	Value	Description
HOU_OVERBUFFERSIZE_ERROR	1030	Data area too small
HOU_SAMEFILENAME_ERROR	1403	File with the same name existing

4.6. 18 FileSeek2

This instruction sets a file position indicator in the specified file in the SD memory card.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
FileSeek2	Seek file	FB		FileSeek2_0(Execute :=, FileID :=, Offset :=, Origin :=, Done =>, Busy =>, Error =>, ErrorID =>,);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
Execute	Enable	BOOL	Depends on data types	-	(Triggered by edges) Execution of the function block switch
FileID	File ID	LWORD	Depends on data types	-	ID of the file in which to set the file position indicator
Offset	Offset	DINT	Depends on data types	-	Offset from Origin
Origin	Reference position	_eFSEEK_ORIGIN	Depends on data types	-	Reference position for file position indicator

Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
Done	Function block completed state	BOOL	Depends on data types	-	Function block completed state
Busy	Function block execution flag	BOOL	Depends on data types	-	Function block execution flag
Error	Function block error flag	BOOL	Depends on data types	-	Function block error flag
ErrorID	Function block error ID	HOU_ERROR	Depends on data types	-	Function block error ID

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Execute	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
FileID	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Offset	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-
Origin	See "Function" for the enumerators for the enumerated type _eFSEEK_ORIGIN.																			
Done	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Busy	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Error	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
ErrorID	See "Function" for the enumerators for the enumerated type HOU_ERROR.																			

■ Function

This instruction sets a file position indicator in the file specified by file ID FileID in the SD memory card. A file position indicator is the position in a file at which to start reading or writing when an instruction such as the FileRead or FileWrite instruction is executed. For example, to read from the beginning of a file, set a file position indicator at the beginning of the file with the FileSeek2 instruction, and then execute the FileRead2 instruction.

The file position indicator is set at offset Offset from reference position Origin.

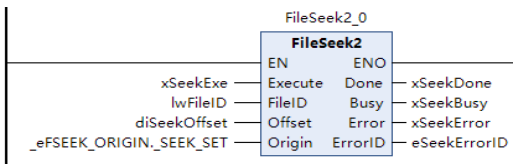
The data type of Origin is enumerated type _eFSEEK_ORIGIN. The meanings of the enumerators are as follows.

Enumerator	Meaning
_SEEK_SET	Beginning of the file
_SEEK_CUR	Location of the current file position indicator
_SEEK_END	End of the file

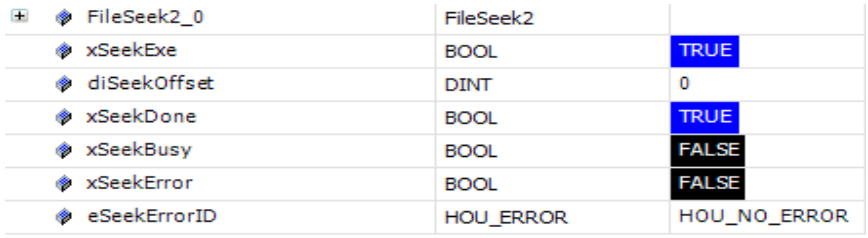
■ Program example

Item	LD	ST
Defined variable		
Program		

```
VAR
    FileSeek2_0     : FileSeek2;
    xSeekExe        : BOOL;
    lwFileID        : LWORD;
    diSeekOffset    : DINT;
    xSeekDone       : BOOL;
    xSeekBusy       : BOOL;
    xSeekError      : BOOL;
    eSeekErrorID    : HOU_ERROR;
END_VAR
```




```
FileSeek2_0(
    Execute:= xSeekExe,
    FileID:= lwFileID,
    Offset:= diSeekOffset,
    Origin:= _eFSEEK_ORIGIN._SEEK_SET,
    Done=> xSeekDone,
    Busy=> xSeekBusy,
    Error=> xSeekError,
    ErrorID=> eSeekErrorID);
```

Item	LD	ST
Running result		

■ Precautions

Execution of this instruction is continued until processing is completed even if the value of Execute changes to FALSE or the execution time exceeds the task period. The value of Done changes to TRUE when processing is completed. Use this to confirm normal completion of processing.

Execute the FileOpen2 instruction in advance to obtain the value for FileID.

An error occurs in the following cases. The value of Error is TRUE.

1) The SD memory card is unavailable.

2) The file specified by FileID is being accessed.

3) The file specified by FileID does not exist.

4) An error that prevents access occurs during SD memory card access.

■ Error ID

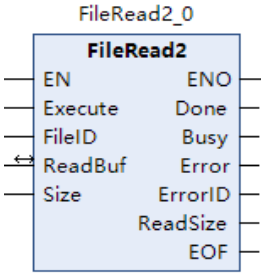
eErrorID	Value	Description
HOU_NO_ERROR	0	No error
HOU_INTERNAL_ERROR	1	General internal error
HOU_INVALIDPARAMETER_ERROR	2	Invalid parameter of the instruction
HOU_NOTINITIALIZED_ERROR	3	Component not initialized, failure to execute the instruction
HOU_VERSION_ERROR	4	Version conflict
HOU_TIMEOUT_ERROR	5	Instruction timeout
HOU_NOBUFFER_ERROR	6	Insufficient memory, failure to execute the instruction
HOU_PENDING_ERROR	10	Instruction calling not completed
HOU_NUMPENDING_ERROR	11	Excessive instruction calls
HOU_NOTIMPLEMENTED_ERROR	12	Instruction not implemented
HOU_INVALIDID_ERROR	13	Invalid ID input
HOU_OVERFLOW_ERROR	14	Integer overflow
HOU_BUFFERSIZE_ERROR	15	Buffer too small or invalid
HOU_NOOBJECT_ERROR	16	Input named object not existing
HOU_NOMEMORY_ERROR	17	No heap memory available
HOU_DUPLICATE_ERROR	18	Another object with the same name existing
HOU_MEMORYOVERWRITE_ERROR	19	Heap memory write out of bounds, memory overwrite error
HOU_INVALIDHANDLE_ERROR	20	Invalid object handle
HOU_ENDOFOBJECT_ERROR	21	Object end reached
HOU_NOCHANGE_ERROR	22	No change

eErrorID	Value	Description
HOU_INVALIDINTERFACE_ERROR	23	Invalid or unknown interface
HOU_NOTSUPPORTED_ERROR	24	Function not supported
HOU_NOACCESSRIGHTS_ERROR	25	No access permission
HOU_OUTOFLIMITS_ERROR	26	Resource limit exceeded
HOU_ENTRIESREMAINING_ERROR	27	Buffer limit causing transmission incompletion
HOU_INVALIDSESSIONID_ERROR	28	Invalid online session ID
HOU_EXCEPTION_ERROR	29	Exception
HOU_INVALIDPATHNAME_ERROR	33	Path name invalid
HOU_INVALIDFILENAME_ERROR	34	File name invalid
HOU_INVALIDLISTMODE_ERROR	35	Invalid control command
HOU_FILEOVERWRITE_ERROR	50	File overwrite error
HOU_OVERBUFFERSIZE_ERROR	1030	Data area too small
HOU_SAMEFILENAME_ERROR	1403	File with the same name existing

4.6. 19 FileRead2

This instruction reads data from the specified file in the SD memory card.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
FileRead2	Read file	FB		FileRead2_0(Execute :=, FileID :=, ReadBuf :=, Size :=, Done =>, Busy =>, Error =>, ErrorID =>, ReadSize =>, EOF =>,);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
Execute	Enable	BOOL	Depends on data types	-	(Triggered by edges) Execution of the function block switch
FileID	File ID	LWORD	Depends on data types	-	ID of the file to read
ReadBuf[]	Offset	DINT	Depends on data types	-	Offset from Origin
Array	Read buffer	—	Depends on data types	-	Buffer in which to write data that was read
Size	Number of elements to read	UINT	Depends on data types	-	Number of elements to read

Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
Done	Function block completed state	BOOL	Depends on data types	-	Function block completed state
Busy	Function block execution flag	BOOL	Depends on data types	-	Function block execution flag
Error	Function block error flag	BOOL	Depends on data types	-	Function block error flag
ErrorID	Function block error ID	HOU_ERROR	Depends on data types	-	Function block error ID
ReadSize	Number of elements actually read	UINT	Depends on data types	-	Number of elements actually read
EOF	End of file	BOOL	Depends on data types	-	Whether the end of the file is reached TRUE: Reached FALSE: Not reached

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Execute	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
FileID	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
ReadBuf[] (array)	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√
	Arrays enumerations or structures can also be specified.																			
Size	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-
Done	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Busy	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Error	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
ErrorID	See "Function" for the enumerators for the enumerated type HOU_ERROR.																			
ReadSize	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-
EOF	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

■ Function

This instruction reads data from the position of the file position indicator in the file specified by FileID in the SD memory card, and then stores the data in read buffer ReadBuf[].

The file position indicator is set at the desired location in advance with the FileSeek2 instruction.

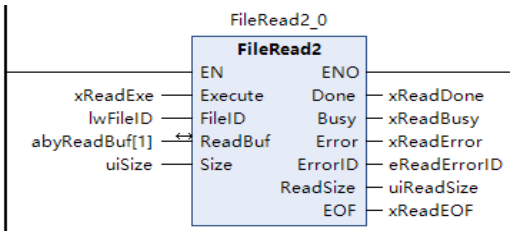
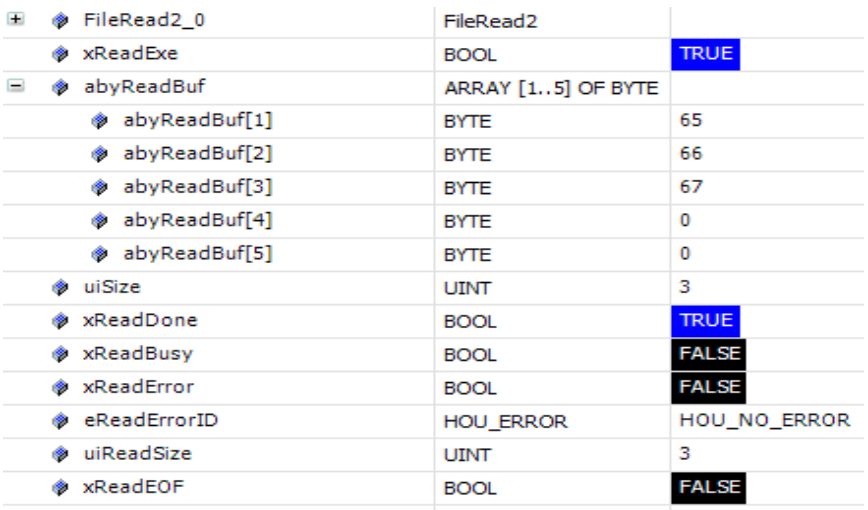
The amount of data that is read is the size of the data type of ReadBuf[] times Size.

The number of elements actually read is stored in ReadSize. In general, values of Size and ReadSize are the same. If the amount of data from the file position indicator to the end of the file is less than the value of Size, no error occurs and the data to the end of the file is stored in ReadBuf[]. If that occurs, the value of ReadSize is less than the value of Size.

If data is read to the end of the file, EOF (end of file) changes to TRUE. Otherwise, the value of EOF is FALSE.

Enumerator	Meaning
_SEEK_SET	Beginning of the file
_SEEK_CUR	Location of the current file position indicator
_SEEK_END	End of the file

■ Program example

Item	LD	ST
Defined variable	VAR FileRead2_0 : FileRead2; xReadExe : BOOL; lwFileID : LWORD; abyReadBuf : ARRAY [1..5] OF BYTE; uiSize : UINT := 3; xReadDone : BOOL; xReadBusy : BOOL; xReadError : BOOL; eReadErrorID : HOU_ERROR; uiReadSize : UINT; xReadEOF : BOOL; END_VAR	
Program		FileRead2_0(Execute:= xReadExe, FileID:= lwFileID, ReadBuf:= abyReadBuf[1], Size:= uiSize, Done=> xReadDone, Busy=> xReadBusy, Error=> xReadError, ErrorID=> eReadErrorID, ReadSize=> uiReadSize, EOF=> xReadEOF);
Running result		

■ Precautions

Execution of this instruction is continued until processing is completed even if the value of Execute changes to FALSE or the execution time exceeds the task period. The value of Done changes to TRUE when processing is completed. Use this to confirm normal completion of processing.

If the data is read to the end of the file and the size of the data is not evenly divisible by the size of the data type of ReadBuf[], the data that is insufficient for the data size of ReadBuf[] is discarded. The file position indicator advances to the end of the file, and the value of EOF changes to TRUE.

When the value of Size exceeds the ReadBuf[] array, an program error may occur and even the PLC may break down.

Elements beyond Size in ReadBuf[] (that is, the elements not overwritten when data is read) retain the values from before execution of this instruction.

Execute the FileOpen2 instruction in advance to obtain the value for FileID.

After this instruction is completed, save the value to EOF. The value of Done changes to TRUE.

If ReadBuf[] is an array of structures, adjustment areas between members may be inserted depending on the composition.

An error occurs in the following cases. The value of Error is TRUE.

1) The SD memory card is unavailable.
2) The file specified by FileID is being accessed.
3) The file specified by FileID does not exist.
4) An error that prevents access occurs during SD memory card access.
5) The file specified by FileID is not opened in a reading mode.

■ Error ID

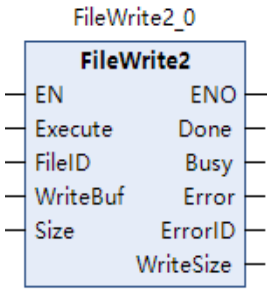
eErrorID	Value	Description
HOU_NO_ERROR	0	No error
HOU_INTERNAL_ERROR	1	General internal error
HOU_INVALIDPARAMETER_ERROR	2	Invalid parameter of the instruction
HOU_NOTINITIALIZED_ERROR	3	Component not initialized, failure to execute the instruction
HOU_VERSION_ERROR	4	Version conflict
HOU_TIMEOUT_ERROR	5	Instruction timeout
HOU_NOBUFFER_ERROR	6	Insufficient memory, failure to execute the instruction
HOU_PENDING_ERROR	10	Instruction calling not completed
HOU_NUMPENDING_ERROR	11	Excessive instruction calls
HOU_NOTIMPLEMENTED_ERROR	12	Instruction not implemented
HOU_INVALIDID_ERROR	13	Invalid ID input
HOU_OVERFLOW_ERROR	14	Integer overflow
HOU_BUFFERSIZE_ERROR	15	Buffer too small or invalid
HOU_NOOBJECT_ERROR	16	Input named object not existing
HOU_NOMEMORY_ERROR	17	No heap memory available
HOU_DUPLICATE_ERROR	18	Another object with the same name existing
HOU_MEMORYOVERWRITE_ERROR	19	Heap memory write out of bounds, memory overwrite error
HOU_INVALIDHANDLE_ERROR	20	Invalid object handle
HOU_ENDOFOBJECT_ERROR	21	Object end reached
HOU_NOCHANGE_ERROR	22	No change
HOU_INVALIDINTERFACE_ERROR	23	Invalid or unknown interface
HOU_NOTSUPPORTED_ERROR	24	Function not supported
HOU_NOACCESSRIGHTS_ERROR	25	No access permission
HOU_OUTOFLIMITS_ERROR	26	Resource limit exceeded

eErrorID	Value	Description
HOU_ENTRIESREMAINING_ERROR	27	Buffer limit causing transmission incompletion
HOU_INVALIDSESSIONID_ERROR	28	Invalid online session ID
HOU_EXCEPTION_ERROR	29	Exception
HOU_INVALIDPATHNAME_ERROR	33	Path name invalid
HOU_INVALIDFILENAME_ERROR	34	File name invalid
HOU_INVALIDLISTMODE_ERROR	35	Invalid control command
HOU_FILEOVERWRITE_ERROR	50	File overwrite error
HOU_OVERBUFFERSIZE_ERROR	1030	Data area too small
HOU_SAMEFILENAME_ERROR	1403	File with the same name existing

4.6. 20 FileWrite2

This instruction writes data to the specified file in the SD memory card.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
FileWrite2	Write file	FB		FileWrite2_0(Execute :=, FileID :=, WriteBuf :=, Size :=, Done =>, Busy =>, Error =>, ErrorID =>, WriteSize =>,);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
Execute	Enable	BOOL	Depends on data types	-	(Triggered by edges) Execution of the function block switch
FileID	File ID	LWORD	Depends on data types	-	ID of the file to write
WriteBuf[]	Offset	DINT	Depends on data types	-	Offset from Origin
Array	Write buffer	-	Depends on data types	-	Data to write
Size	Number of elements to write	UINT	Depends on data types	-	Number of elements to write

Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
Done	Function block completed state	BOOL	Depends on data types	-	Function block completed state

Output Variable	Name	Data Type	Value Range	Initial Value	Description
Busy	Function block execution flag	BOOL	Depends on data types	-	Function block execution flag
Error	Function block error flag	BOOL	Depends on data types	-	Function block error flag
ErrorID	Function block error ID	HOU_ERROR	Depends on data types	-	Function block error ID
WriteSize	Number of elements actually written	UINT	Depends on data types	-	Number of elements actually written

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Execute	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
FileID	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
WriteBuf[] (array)	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√
	Arrays enumerations or structures can also be specified.																			
Size	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-
Done	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Busy	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Error	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
ErrorID	See "Function" for the enumerators for the enumerated type HOU_ERROR.																			
WriteSize	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-

■ Function

This instruction writes the data written to WriteBuf[] to the position of the file position indicator in the file specified by FileID in the SD memory card.

The file position indicator is set at the desired location in advance with the FileSeek2 instruction.

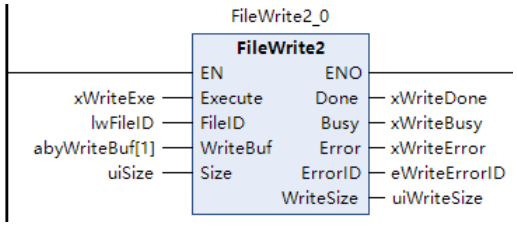
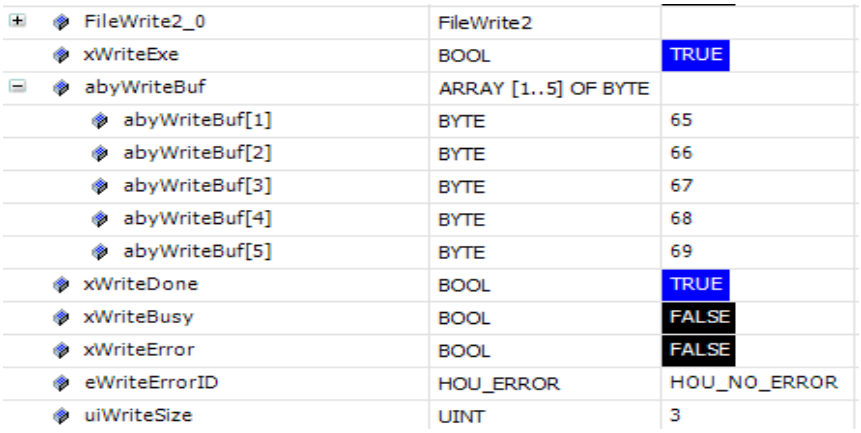
The amount of data that is written is the size of the data type of WriteBuf[] times Size.

The number of elements actually written is saved to WriteSize. In general, values of Size and WriteSize are the same.

■ Program example

Item	LD	ST
Defined variable		

```
VAR
    FileWrite2_0    : FileWrite2;
    xWriteExe       : BOOL;
    lwFileID        : LWORD;
    abyWriteBuf     : ARRAY [1..5] OF BYTE := [65,66,67,68,69];
    xWriteDone      : BOOL;
    xWriteBusy      : BOOL;
    xWriteError     : BOOL;
    eWriteErrorID   : HOU_ERROR;
    uiWriteSize     : UINT;
END_VAR
```

<table>
<tr><th>Item</th><th>LD</th><th>ST</th></tr>
<tr><td>Program</td><td> 
</td><td><pre>
FileWrite2_0(
    Execute:= xWriteExe,
    FileID:= lwFileID,
    WriteBuf:= abyWriteBuf[1],
    Size:= uiSize,
    Done=> xWriteDone,
    Busy=> xWriteBusy,
    Error=> xWriteError,
    ErrorID=> eWriteErrorID,
    WriteSize=> uiWriteSize);
</pre></td></tr>
<tr><td>Running result</td><td colspan="2"> 
</td></tr>
</table>

■ Precautions

Execution of this instruction is continued until processing is completed even if the value of Execute changes to FALSE or the execution time exceeds the task period. The value of Done changes to TRUE when processing is completed. Use this to confirm normal completion of processing.

When the value of Size exceeds the WriteBuf[] array, an program error may occur and the result is unexpected.

Execute the FileOpen2 instruction in advance to obtain the value for FileID.

If WriteBuf[] is an array of structures, adjustment areas between members may be inserted depending on the composition.

An error occurs in the following cases. The value of Error is TRUE.

The SD memory card is unavailable.

1) The SD memory card is unavailable.
2) The file specified by FileID is being accessed.
3) The file specified by FileID does not exist.
4) An error that prevents access occurs during SD memory card access.
5) The file specified by FileID is not opened in a writing mode.

■ Error ID

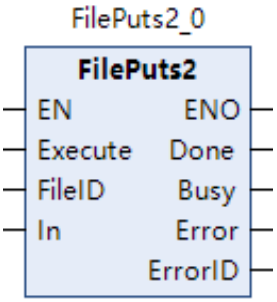
eErrorID	Value	Description
HOU_NO_ERROR	0	No error
HOU_INTERNAL_ERROR	1	General internal error
HOU_INVALIDPARAMETER_ERROR	2	Invalid parameter of the instruction

eErrorID	Value	Description
HOU_NOTINITIALIZED_ERROR	3	Component not initialized, failure to execute the instruction
HOU_VERSION_ERROR	4	Version conflict
HOU_TIMEOUT_ERROR	5	Instruction timeout
HOU_NOBUFFER_ERROR	6	Insufficient memory, failure to execute the instruction
HOU_PENDING_ERROR	10	Instruction calling not completed
HOU_NUMPENDING_ERROR	11	Excessive instruction calls
HOU_NOTIMPLEMENTED_ERROR	12	Instruction not implemented
HOU_INVALIDID_ERROR	13	Invalid ID input
HOU_OVERFLOW_ERROR	14	Integer overflow
HOU_BUFFERSIZE_ERROR	15	Buffer too small or invalid
HOU_NOOBJECT_ERROR	16	Input named object not existing
HOU_NOMEMORY_ERROR	17	No heap memory available
HOU_DUPLICATE_ERROR	18	Another object with the same name existing
HOU_MEMORYOVERWRITE_ERROR	19	Heap memory write out of bounds, memory overwrite error
HOU_INVALIDHANDLE_ERROR	20	Invalid object handle
HOU_ENDOFOBJECT_ERROR	21	Object end reached
HOU_NOCHANGE_ERROR	22	No change
HOU_INVALIDINTERFACE_ERROR	23	Invalid or unknown interface
HOU_NOTSUPPORTED_ERROR	24	Function not supported
HOU_NOACCESSRIGHTS_ERROR	25	No access permission
HOU_OUTOFLIMITS_ERROR	26	Resource limit exceeded
HOU_ENTRIESREMAINING_ERROR	27	Buffer limit causing transmission incompletion
HOU_INVALIDSESSIONID_ERROR	28	Invalid online session ID
HOU_EXCEPTION_ERROR	29	Exception
HOU_INVALIDPATHNAME_ERROR	33	Path name invalid
HOU_INVALIDFILENAME_ERROR	34	File name invalid
HOU_INVALIDLISTMODE_ERROR	35	Invalid control command
HOU_FILEOVERWRITE_ERROR	50	File overwrite error
HOU_OVERBUFFERSIZE_ERROR	1030	Data area too small
HOU_SAMEFILENAME_ERROR	1403	File with the same name existing

4.6. 21 FilePuts2

This instruction writes a text string to the specified file in the SD memory card.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
FilePuts2	Put text string	FB		FilePuts2_0(Execute :=, FileID :=, In :=, Done =>, Busy =>, Error =>, ErrorID =>,);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
Execute	Enable	BOOL	Depends on data types	-	(Triggered by edges) Execution of the function block switch
FileID	File ID	LWORD	Depends on data types	-	ID of the file to write
In	Text string to write	STRING	Depends on data types	-	Text string to write

Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
Done	Function block completed state	BOOL	Depends on data types	-	Function block completed state
Busy	Function block execution flag	BOOL	Depends on data types	-	Function block execution flag
Error	Function block error flag	HOU_ERROR	Depends on data types	-	Function block error flag
ErrorID	Function block error ID	WORD	Depends on data types	-	Function block error ID

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Execute	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
FileID	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
In	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√
Done	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Busy	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Error	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
ErrorID	See "Function" for the enumerators for the enumerated type HOU_ERROR.																			

■ Function

This instruction writes the text string In to the position of the file position indicator in the file specified by FileID in the SD memory card.

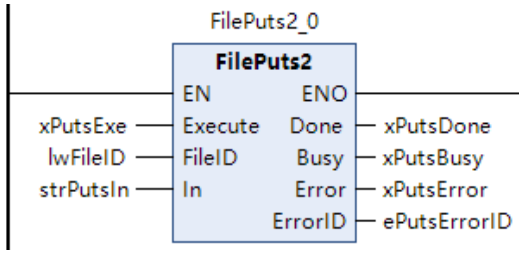
The file position indicator is set at the desired location in advance with the FileSeek2 instruction.

■ Program example

<table>
<tr><th>Item</th><th>LD</th><th>ST</th></tr>
<tr><td>Defined variable</td><td colspan="2">

```
VAR
    FilePuts2_0     : FilePuts2;
    xPutsExe        : BOOL;
    lwFileID        : LWORD;
    strPutsIn       : STRING(1985) := '1234$R5678';
    xPutsDone       : BOOL;
    xPutsBusy       : BOOL;
    xPutsError      : BOOL;
    ePutsErrorID    : HOU_ERROR;
END_VAR
```

</td></tr>
<tr><td>Program</td><td>



</td><td>

```
FilePuts2_0(
    Execute:= xPutsExe,
    FileID:= lwFileID,
    In:= strPutsIn,
    Done=> xPutsDone,
    Busy=> xPutsBusy,
    Error=> xPutsError,
    ErrorID=> ePutsErrorID);
```

</td></tr>
<tr><td>Running result</td><td colspan="2">

FilePuts2_0	FilePuts2	
xPutsExe	BOOL	TRUE
strPutsIn	STRING(1985)	'1234$R5678'
xPutsDone	BOOL	TRUE
xPutsBusy	BOOL	FALSE
xPutsError	BOOL	FALSE
ePutsErrorID	HOU_ERROR	HOU_NO_ERROR

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

■ Precautions

Execution of this instruction is continued until processing is completed even if the value of Execute changes to FALSE or the execution time exceeds the task period. The value of Done changes to TRUE when processing is completed. Use this to confirm normal completion of processing.

Execute the FileOpen2 instruction in advance to obtain the value for FileID.

An error occurs in the following cases. The value of Error is TRUE.

1) The SD memory card is unavailable.
2) The SD memory card is write-protected.
3) There is insufficient space available in the SD memory card.
4) The file specified by FileID is being accessed.
5) The file specified by FileID does not exist.
6) An error that prevents access occurs during SD memory card access.
7) The file specified by FileID is not opened in a writing mode.

■ Error ID

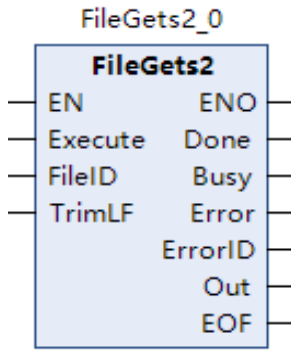
eErrorID	Value	Description
HOU_NO_ERROR	0	No error
HOU_INTERNAL_ERROR	1	General internal error
HOU_INVALIDPARAMETER_ERROR	2	Invalid parameter of the instruction

eErrorID	Value	Description
HOU_NOTINITIALIZED_ERROR	3	Component not initialized, failure to execute the instruction
HOU_VERSION_ERROR	4	Version conflict
HOU_TIMEOUT_ERROR	5	Instruction timeout
HOU_NOBUFFER_ERROR	6	Insufficient memory, failure to execute the instruction
HOU_PENDING_ERROR	10	Instruction calling not completed
HOU_NUMPENDING_ERROR	11	Excessive instruction calls
HOU_NOTIMPLEMENTED_ERROR	12	Instruction not implemented
HOU_INVALIDID_ERROR	13	Invalid ID input
HOU_OVERFLOW_ERROR	14	Integer overflow
HOU_BUFFERSIZE_ERROR	15	Buffer too small or invalid
HOU_NOOBJECT_ERROR	16	Input named object not existing
HOU_NOMEMORY_ERROR	17	No heap memory available
HOU_DUPLICATE_ERROR	18	Another object with the same name existing
HOU_MEMORYOVERWRITE_ERROR	19	Heap memory write out of bounds, memory overwrite error
HOU_INVALIDHANDLE_ERROR	20	Invalid object handle
HOU_ENDOFOBJECT_ERROR	21	Object end reached
HOU_NOCHANGE_ERROR	22	No change
HOU_INVALIDINTERFACE_ERROR	23	Invalid or unknown interface
HOU_NOTSUPPORTED_ERROR	24	Function not supported
HOU_NOACCESSRIGHTS_ERROR	25	No access permission
HOU_OUTOFLIMITS_ERROR	26	Resource limit exceeded
HOU_ENTRIESREMAINING_ERROR	27	Buffer limit causing transmission incompletion
HOU_INVALIDSESSIONID_ERROR	28	Invalid online session ID
HOU_EXCEPTION_ERROR	29	Exception
HOU_INVALIDPATHNAME_ERROR	33	Path name invalid
HOU_INVALIDFILENAME_ERROR	34	File name invalid
HOU_INVALIDLISTMODE_ERROR	35	Invalid control command
HOU_FILEOVERWRITE_ERROR	50	File overwrite error
HOU_OVERBUFFERSIZE_ERROR	1030	Data area too small
HOU_SAMEFILENAME_ERROR	1403	File with the same name existing

4.6. 22 FileGets2

This instruction reads a text string of one line from the specified file in the SD memory card.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
FileGets2	Get text string	FB		FileGets2_0(Execute :=, FileID :=, TrimLF :=, Done =>, Busy =>, Error =>, ErrorID =>, Out =>, EOF =>,);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
Execute	Enable	BOOL	Depends on data types	-	(Triggered by edges) Execution of the function block switch
FileID	File ID	LWORD	Depends on data types	-	ID of the file to write
TrimLF	Line feed designation	BOOL	Depends on data types	-	Handling of the line feed code of the read text string TRUE: Delete FALSE: Do not delete

Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
Done	Function block completed state	BOOL	Depends on data types	-	Function block completed state
Busy	Function block execution flag	BOOL	Depends on data types	-	Function block execution flag
Error	Function block error flag	BOOL	Depends on data types	-	Function block error flag
ErrorID	Function block error ID	HOU_ERROR	Depends on data types	-	Function block error ID
Out	Read text string	STRING	Depends on data types	-	Read text string
EOF	End of file	BOOL	Depends on data types	-	Whether the end of the file is reached TRUE: Reached FALSE: Not reached

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Execute	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
FileID	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
TrimLF	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Done	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Busy	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Error	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
ErrorID	See "Function" for the enumerators for the enumerated type HOU_ERROR.																			
Out	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√
EOF	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

■ Function

This instruction reads one line of text string to the position of the file position indicator in the file specified by FileID in the SD memory card.

The file position indicator is set at the desired location in advance with the FileSeek2 instruction.

Line endings are determined by a line feed code. The text string that is read is written to read the text string Out.

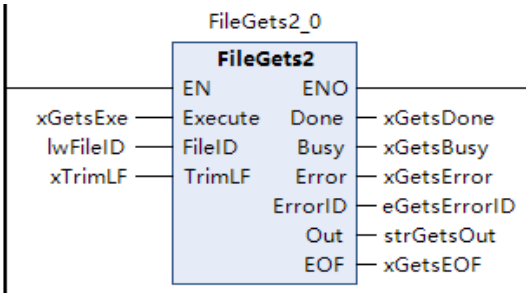
The following three line feeds are automatically detected: CR, LF, and CR+LF. If line feed designation TrimLF is TRUE, the line feed code is deleted from the text string before it is written to Out.

If data is read to the end of the file, EOF (end of file) changes to TRUE.

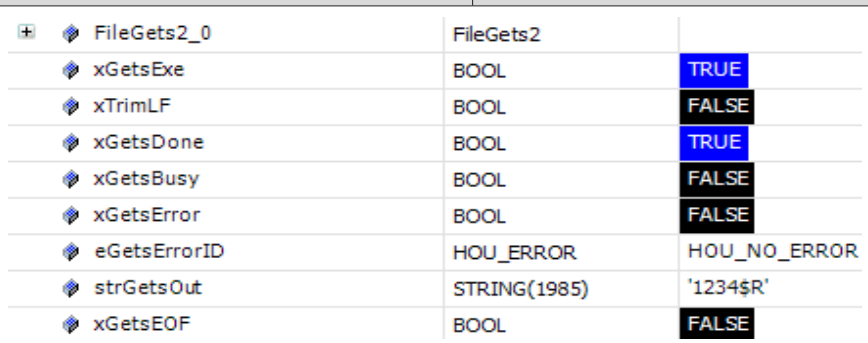
■ Program example

Item	LD	ST
Defined variable		
Program		

```
VAR
    FileGets2_0     : FileGets2;
    xGetsExe        : BOOL;
    lwFileID        : LWORD;
    xTrimLF         : BOOL;
    xGetsDone       : BOOL;
    xGetsBusy       : BOOL;
    xGetsError      : BOOL;
    eGetsErrorID    : HOU_ERROR;
    strGetsOut      : STRING(1985);
    xGetsEOF        : BOOL;
END_VAR
```




```
FileGets2_0(
    Execute:= xGetsExe,
    FileID:= lwFileID,
    TrimLF:= xTrimLF,
    Done=> xGetsDone,
    Busy=> xGetsBusy,
    Error=> xGetsError,
    ErrorID=> eGetsErrorID,
    Out=> strGetsOut,
    EOF=> xGetsEOF);
```

Item	LD	ST
Running result		

■ Precautions

Execution of this instruction is continued until processing is completed even if the value of Execute changes to FALSE or the execution time exceeds the task period. The value of Done changes to TRUE when processing is completed. Use this to confirm normal completion of processing.

If the length of the one-line text string exceeds 1,986 bytes, the first 1,985 bytes of the text string and the NULL text string at the end are stored in Out.

Execute the FileOpen2 instruction in advance to obtain the value for FileID.

An error occurs in the following cases. The value of Error is TRUE.

1) The SD memory card is unavailable.
2) The file specified by FileID is being accessed.
3) The file specified by FileID does not exist.
4) An error that prevents access occurs during SD memory card access.
5) The file specified by FileID is not opened in a reading mode.

■ Error ID

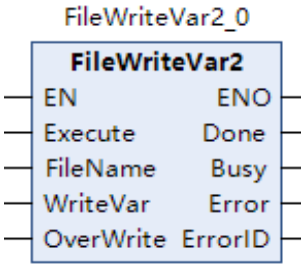
eErrorID	Value	Description
HOU_NO_ERROR	0	No error
HOU_INTERNAL_ERROR	1	General internal error
HOU_INVALIDPARAMETER_ERROR	2	Invalid parameter of the instruction
HOU_NOTINITIALIZED_ERROR	3	Component not initialized, failure to execute the instruction
HOU_VERSION_ERROR	4	Version conflict
HOU_TIMEOUT_ERROR	5	Instruction timeout
HOU_NOBUFFER_ERROR	6	Insufficient memory, failure to execute the instruction
HOU_PENDING_ERROR	10	Instruction calling not completed
HOU_NUMPENDING_ERROR	11	Excessive instruction calls
HOU_NOTIMPLEMENTED_ERROR	12	Instruction not implemented
HOU_INVALIDID_ERROR	13	Invalid ID input
HOU_OVERFLOW_ERROR	14	Integer overflow
HOU_BUFFERSIZE_ERROR	15	Buffer too small or invalid
HOU_NOOBJECT_ERROR	16	Input named object not existing
HOU_NOMEMORY_ERROR	17	No heap memory available
HOU_DUPLICATE_ERROR	18	Another object with the same name existing

eErrorID	Value	Description
HOU_MEMORYOVERWRITE_ERROR	19	Heap memory write out of bounds, memory overwrite error
HOU_INVALIDHANDLE_ERROR	20	Invalid object handle
HOU_ENDOFOBJECT_ERROR	21	Object end reached
HOU_NOCHANGE_ERROR	22	No change
HOU_INVALIDINTERFACE_ERROR	23	Invalid or unknown interface
HOU_NOTSUPPORTED_ERROR	24	Function not supported
HOU_NOACCESSRIGHTS_ERROR	25	No access permission
HOU_OUTOFLIMITS_ERROR	26	Resource limit exceeded
HOU_ENTRIESREMAINING_ERROR	27	Buffer limit causing transmission incompletion
HOU_INVALIDSESSIONID_ERROR	28	Invalid online session ID
HOU_EXCEPTION_ERROR	29	Exception
HOU_INVALIDPATHNAME_ERROR	33	Path name invalid
HOU_INVALIDFILENAME_ERROR	34	File name invalid
HOU_INVALIDLISTMODE_ERROR	35	Invalid control command
HOU_FILEOVERWRITE_ERROR	50	File overwrite error
HOU_OVERBUFFERSIZE_ERROR	1030	Data area too small
HOU_SAMEFILENAME_ERROR	1403	File with the same name existing

4.6. 23 FileWriteVar2

This instruction writes the value of a variable to the specified file in the SD memory card in binary format.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
FileWriteVar2	Write variable to file	FB		FileWriteVar2_0(Execute :=, FileName :=, WriteVar :=, OverWrite :=, Done =>, Busy =>, Error =>, ErrorID =>,);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
Execute	Enable	BOOL	Depends on data types	-	(Triggered by edges) Execution of the function block switch
FileName	Specified file name	STRING	Maximum 1,985 bytes	-	Name of the file to which to write variable data
WriteVar	Specified variable	-	Depends on data types	-	Variable to write

Input Variable	Name	Data Type	Value Range	Initial Value	Description
OverWrite	Overwrite enable	BOOL	Depends on data types	-	TRUE: Enable overwrite FALSE: Prohibit overwrite

Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
Done	Function block completed state	BOOL	Depends on data types	-	Function block completed state
Busy	Function block execution flag	BOOL	Depends on data types	-	Function block execution flag
Error	Function block error flag	BOOL	Depends on data types	-	Function block error flag
ErrorID	Function block error ID	HOU_ERROR	Depends on data types	-	Function block error ID

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Execute	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
FileName	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√
WriteVar	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√
	An enumeration, an array, an array element, a structure, or a structure member can also be specified.																			
OverWrite	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Done	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Busy	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Error	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
ErrorID	See "Function" for the enumerators for the enumerated type HOU_ERROR.																			

■ Function

This instruction writes the value of variable WriteVar to the file specified by FileName in the SD memory card in binary format.

If a file with the name specified by FileName does not exist in the SD memory card, it is created.

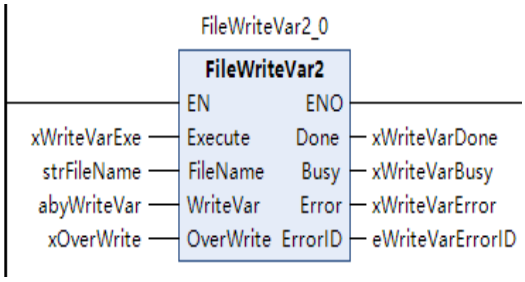
If a file with the name specified by FileName already exists in the SD memory card, the following processing is performed depending on the value of OverWrite.

Value of OverWrite	Operation
TRUE	The file is overwritten.
FALSE	The file is not overwritten and an error occurs.

■ Program example

<table>
<tr><th>Item</th><th>LD</th><th>ST</th></tr>
<tr><td>Defined variable</td><td colspan="2">

```
VAR
    FileWriteVar2_0     : FileWriteVar2;
    xWriteVarExe        : BOOL;
    strFileName         : STRING(1985) := 'FileTest/File.txt';
    abyWriteVar         : ARRAY [1..5] OF BYTE := [65,66,67,68,69];
    xOverWrite          : BOOL;
    xWriteVarDone       : BOOL;
    xWriteVarBusy       : BOOL;
    xWriteVarError      : BOOL;
    eWriteVarErrorID    : HOU_ERROR;
END_VAR
```

</td></tr>
<tr><td>Program</td><td>


</td><td>

```
FileWriteVar2_0(
    Execute:= xWriteVarExe,
    FileName:= strFileName,
    WriteVar:= abyWriteVar,
    OverWrite:= xOverWrite,
    Done=> xWriteVarDone,
    Busy=> xWriteVarBusy,
    Error=> xWriteVarError,
    ErrorID=> eWriteVarErrorID);
```

</td></tr>
<tr><td>Running result</td><td colspan="2">

FileWriteVar2_0	FileWriteVar2	
xWriteVarExe	BOOL	TRUE
strFileName	STRING(1985)	'FileTest/File.txt'
abyWriteVar	ARRAY [1..5] OF BYTE	
abyWriteVar[1]	BYTE	65
abyWriteVar[2]	BYTE	66
abyWriteVar[3]	BYTE	67
abyWriteVar[4]	BYTE	68
abyWriteVar[5]	BYTE	69
xOverWrite	BOOL	FALSE
xWriteVarDone	BOOL	TRUE
xWriteVarBusy	BOOL	FALSE
xWriteVarError	BOOL	FALSE
eWriteVarErrorID	HOU_ERROR	HOU_NO_ERROR

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

■ Precautions

Execution of this instruction is continued until processing is completed even if the value of Execute changes to FALSE or the execution time exceeds the task period. The value of Done changes to TRUE when processing is completed. Use this to confirm normal completion of processing.

If WriteVar is a structure, adjustment areas between members may be inserted depending on the composition.

An error occurs in the following cases. The value of Error is TRUE.

1) The SD memory card is unavailable.
2) The SD memory card is write-protected.
3) There is insufficient space available in the SD memory card.
4) The file specified by FileID is being accessed.
5) The value of FileName is not a valid file name.
6) The maximum number of files or directories is exceeded.

7) A file with the name specified by FileName exits and the file is being accessed.

8) A file with the name specified by FileName exits and the value of OverWrite is FALSE.

9) A file with the name specified by FileName exits and is read-only.

10) An error that prevents access occurs during SD memory card access.

■ Error ID

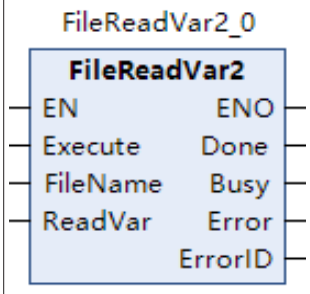
eErrorID	Value	Description
HOU_NO_ERROR	0	No error
HOU_INTERNAL_ERROR	1	General internal error
HOU_INVALIDPARAMETER_ERROR	2	Invalid parameter of the instruction
HOU_NOTINITIALIZED_ERROR	3	Component not initialized, failure to execute the instruction
HOU_VERSION_ERROR	4	Version conflict
HOU_TIMEOUT_ERROR	5	Instruction timeout
HOU_NOBUFFER_ERROR	6	Insufficient memory, failure to execute the instruction
HOU_PENDING_ERROR	10	Instruction calling not completed
HOU_NUMPENDING_ERROR	11	Excessive instruction calls
HOU_NOTIMPLEMENTED_ERROR	12	Instruction not implemented
HOU_INVALIDID_ERROR	13	Invalid ID input
HOU_OVERFLOW_ERROR	14	Integer overflow
HOU_BUFFERSIZE_ERROR	15	Buffer too small or invalid
HOU_NOOBJECT_ERROR	16	Input named object not existing
HOU_NOMEMORY_ERROR	17	No heap memory available
HOU_DUPLICATE_ERROR	18	Another object with the same name existing
HOU_MEMORYOVERWRITE_ERROR	19	Heap memory write out of bounds, memory overwrite error
HOU_INVALIDHANDLE_ERROR	20	Invalid object handle
HOU_ENDOFOBJECT_ERROR	21	Object end reached
HOU_NOCHANGE_ERROR	22	No change
HOU_INVALIDINTERFACE_ERROR	23	Invalid or unknown interface
HOU_NOTSUPPORTED_ERROR	24	Function not supported
HOU_NOACCESSRIGHTS_ERROR	25	No access permission
HOU_OUTOFLIMITS_ERROR	26	Resource limit exceeded
HOU_ENTRIESREMAINING_ERROR	27	Buffer limit causing transmission incompletion
HOU_INVALIDSESSIONID_ERROR	28	Invalid online session ID
HOU_EXCEPTION_ERROR	29	Exception
HOU_INVALIDPATHNAME_ERROR	33	Path name invalid
HOU_INVALIDFILENAME_ERROR	34	File name invalid
HOU_INVALIDLISTMODE_ERROR	35	Invalid control command
HOU_FILEOVERWRITE_ERROR	50	File overwrite error
HOU_OVERBUFFERSIZE_ERROR	1030	Data area too small

eErrorID	Value	Description
HOU_SAMEFILENAME_ERROR	1403	File with the same name existing

4.6. 24 FileReadVar2

This instruction reads the values of the specified file in the SD memory card as binary data and writes the data to a variable.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
FileReadVar2	Read variable from file	FB		FileReadVar2_0(Execute :=, FileName :=, ReadVar :=, Done =>, Busy =>, Error =>, ErrorID =>,);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
Execute	Enable	BOOL	Depends on data types	-	(Triggered by edges) Execution of the function block switch
FileName	Specified file name	STRING	Maximum 1,985 bytes	-	Name of the file from which to read variable data
ReadVar	Position variable to read	-	Depends on data types	-	Position variable from which to read variable data

Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
Done	Function block completed state	BOOL	Depends on data types	-	Function block completed state
Busy	Function block execution flag	BOOL	Depends on data types	-	Function block execution flag
Error	Function block error flag	BOOL	Depends on data types	-	Function block error flag
ErrorID	Function block error ID	HOU_ERROR	Depends on data types	-	Function block error ID

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Execute	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
File-Name	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√
ReadVar	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√
	An enumeration, an array, an array element, a structure, or a structure member can also be specified.																			
Done	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Busy	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Error	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
ErrorID	See "Function" for the enumerators for the enumerated type HOU_ERROR.																			

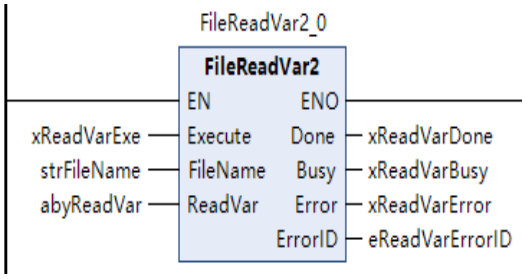
■ Function

This instruction reads variables from the file specified by FileName in the SD memory card in binary format and assigns the read data to the position variable to read ReadVar.

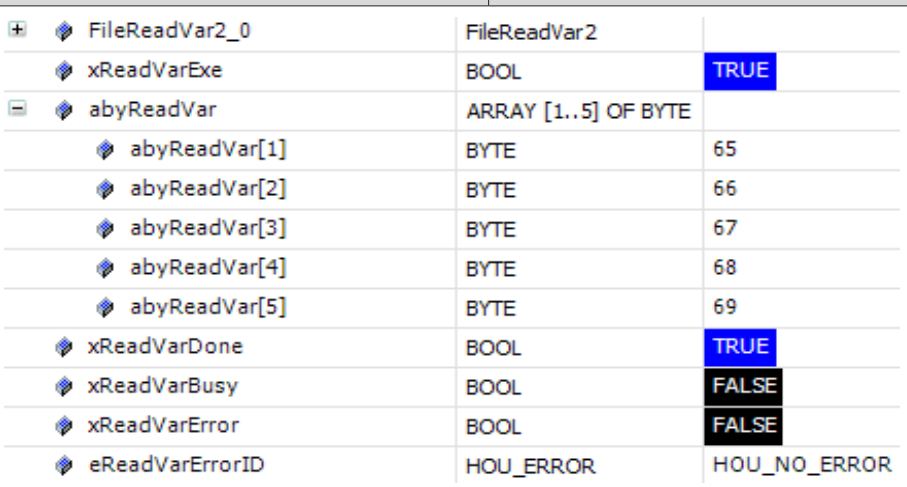
■ Program example

Item	LD	ST
Defined variable		
Program		

```
VAR
    FileReadVar2_0      : FileReadVar2;
    xReadVarExe         : BOOL;
    strFileName         : STRING(1985) := 'FileTest/File.txt';
    abyReadVar          : ARRAY [1..5] OF BYTE;
    xReadVarDone        : BOOL;
    xReadVarBusy        : BOOL;
    xReadVarError       : BOOL;
    eReadVarErrorID     : HOU_ERROR;
END_VAR
```




```
FileReadVar2_0(
    Execute:= xReadVarExe,
    FileName:= strFileName,
    ReadVar:= abyReadVar,
    Done=> xReadVarDone,
    Busy=> xReadVarBusy,
    Error=> xReadVarError,
    ErrorID=> eReadVarErrorID);
```

Item	LD	ST
Running result		

■ Precautions

Execution of this instruction is continued until processing is completed even if the value of Execute changes to FALSE or the execution time exceeds the task period. The value of Done changes to TRUE when processing is completed. Use this to confirm normal completion of processing.

If the size of the specified file is greater than that of ReadVar, no error occurs and only the data of the size of ReadVar is read.

If the size of the specified file is less than that of ReadVar, no error occurs and only the data of the size of the specified file is read. The remaining area in ReadVar retains the values used before execution of this instruction.

If ReadVar is a structure, adjustment areas between members may be inserted depending on the composition.

An error occurs in the following cases. The value of Error is TRUE.

1) The SD memory card is unavailable.

2) The value of FileName is not a valid file name.

3) The file specified by FileName does not exist.

4) The file specified by FileName is being accessed.

5) An error that prevents access occurs during SD memory card access.

■ Error ID

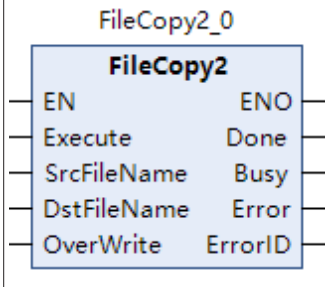
eErrorID	Value	Description
HOU_NO_ERROR	0	No error
HOU_INTERNAL_ERROR	1	General internal error
HOU_INVALIDPARAMETER_ERROR	2	Invalid parameter of the instruction
HOU_NOTINITIALIZED_ERROR	3	Component not initialized, failure to execute the instruction
HOU_VERSION_ERROR	4	Version conflict
HOU_TIMEOUT_ERROR	5	Instruction timeout
HOU_NOBUFFER_ERROR	6	Insufficient memory, failure to execute the instruction
HOU_PENDING_ERROR	10	Instruction calling not completed
HOU_NUMPENDING_ERROR	11	Excessive instruction calls

eErrorID	Value	Description
HOU_NOTIMPLEMENTED_ERROR	12	Instruction not implemented
HOU_INVALIDID_ERROR	13	Invalid ID input
HOU_OVERFLOW_ERROR	14	Integer overflow
HOU_BUFFERSIZE_ERROR	15	Buffer too small or invalid
HOU_NOOBJECT_ERROR	16	Input named object not existing
HOU_NOMEMORY_ERROR	17	No heap memory available
HOU_DUPLICATE_ERROR	18	Another object with the same name existing
HOU_MEMORYOVERWRITE_ERROR	19	Heap memory write out of bounds, memory overwrite error
HOU_INVALIDHANDLE_ERROR	20	Invalid object handle
HOU_ENDOFOBJECT_ERROR	21	Object end reached
HOU_NOCHANGE_ERROR	22	No change
HOU_INVALIDINTERFACE_ERROR	23	Invalid or unknown interface
HOU_NOTSUPPORTED_ERROR	24	Function not supported
HOU_NOACCESSRIGHTS_ERROR	25	No access permission
HOU_OUTOFLIMITS_ERROR	26	Resource limit exceeded
HOU_ENTRIESREMAINING_ERROR	27	Buffer limit causing transmission incompletion
HOU_INVALIDSESSIONID_ERROR	28	Invalid online session ID
HOU_EXCEPTION_ERROR	29	Exception
HOU_INVALIDPATHNAME_ERROR	33	Path name invalid
HOU_INVALIDFILENAME_ERROR	34	File name invalid
HOU_INVALIDLISTMODE_ERROR	35	Invalid control command
HOU_FILEOVERWRITE_ERROR	50	File overwrite error
HOU_OVERBUFFERSIZE_ERROR	1030	Data area too small
HOU_SAMEFILENAME_ERROR	1403	File with the same name existing

4.6. 25 FileCopy2

This instruction copies the specified file in the SD memory card.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
FileCopy2	Copy file	FB		FileCopy2_0(Execute :=, SrcFileName :=, DstFileName :=, OverWrite :=, Done =>, Busy =>, Error =>, ErrorID =>,);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
Execute	Enable	BOOL	Depends on data types	-	(Triggered by edges) Execution of the function block switch
SrcFileName	Name of source file to copy	STRING	Maximum 1,985 bytes	-	Name of source file to copy
DstFileName	Name of destination file to paste	STRING	Maximum 1,985 bytes	-	Name of destination file to paste
OverWrite	Overwrite enable	BOOL	Depends on data types	-	TRUE: Enable overwrite FALSE: Prohibit overwrite

Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
Done	Function block completed state	BOOL	Depends on data types	-	Function block completed state
Busy	Function block execution flag	BOOL	Depends on data types	-	Function block execution flag
Error	Function block error flag	BOOL	Depends on data types	-	Function block error flag
ErrorID	Function block error ID	HOU_ERROR	Depends on data types	-	Function block error ID

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Execute	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
SrcFileName	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√
DstFileName	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√
OverWrite	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Done	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Busy	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Error	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
ErrorID	See "Function" for the enumerators for the enumerated type HOU_ERROR.																			

■ Function

This instruction copies the source file specified by SrcFileName in the SD memory card to DstFileName.

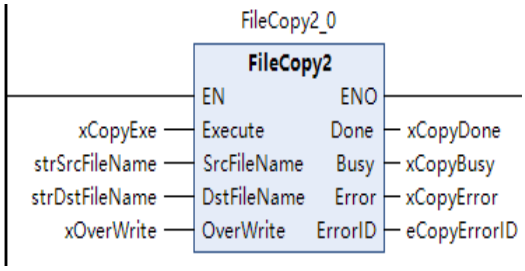
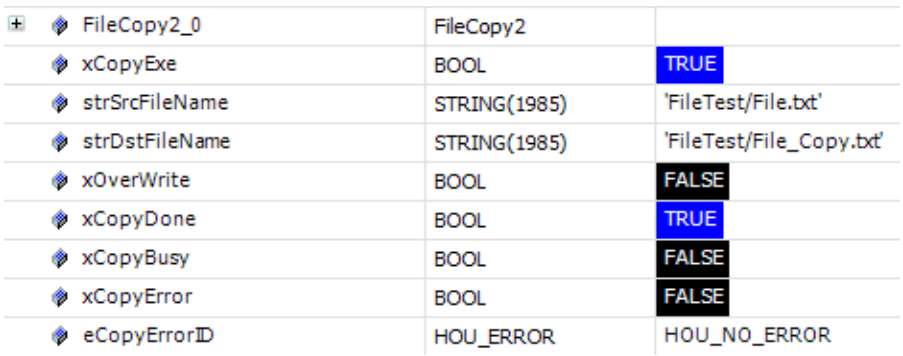
If a file with the name specified by DstFileName does not exist in the SD memory card, it is created.

If a file with the name specified by DstFileName already exists in the SD memory card, the following processing is performed depending on the value of OverWrite.

Value of OverWrite	Operation
TRUE	The file is overwritten.

Value of OverWrite	Operation
FALSE	The file is not overwritten and an error occurs.

■ Program example

Item	LD	ST
Defined variable	VAR FileCopy2_0 : FileCopy2; xCopyExe : BOOL; strSrcFileName : STRING(1985) := 'FileTest/File.txt'; strDstFileName : STRING(1985) := 'FileTest/File_Copy.txt'; xOverWrite : BOOL; xCopyDone : BOOL; xCopyBusy : BOOL; xCopyError : BOOL; eCopyErrorID : HOU_ERROR; END_VAR	
Program		FileCopy2_0(Execute:= xCopyExe, SrcFileName:= strSrcFileName, DstFileName:= strDstFileName, OverWrite:= xOverWrite, Done=> xCopyDone, Busy=> xCopyBusy, Error=> xCopyError, ErrorID=> eCopyErrorID);
Running result		

■ Precautions

Execution of this instruction is continued until processing is completed even if the value of Execute changes to FALSE or the execution time exceeds the task period. The value of Done changes to TRUE when processing is completed. Use this to confirm normal completion of processing.

When the file is opened, remove the SD memory card. Then the file remains the open state. When you insert the SD memory card again, you cannot read or write the file. To read and write the file, re-open the file.

An error occurs in the following cases. The value of Error is TRUE.

1) The SD memory card is unavailable.
2) The SD memory card is write-protected.
3) There is insufficient space available in the SD memory card.
4) The file specified by SrcFileName does not exist.

5) The value of SrcFileName is not a valid file name.

6) The value of DstFileName is not a valid file name.

7) The maximum number of files or directories is exceeded.

8) A file with the name specified by DstFileName exits and the file is being accessed.

9) A file with the name specified by DstFileName exits and the value of OverWrite is FALSE.

10) A file with the name specified by DstFileName exits and is read-only.

11) An error that prevents access occurs during SD memory card access.

■ Error ID

eErrorID	Value	Description
HOU_NO_ERROR	0	No error
HOU_INTERNAL_ERROR	1	General internal error
HOU_INVALIDPARAMETER_ERROR	2	Invalid parameter of the instruction
HOU_NOTINITIALIZED_ERROR	3	Component not initialized, failure to execute the instruction
HOU_VERSION_ERROR	4	Version conflict
HOU_TIMEOUT_ERROR	5	Instruction timeout
HOU_NOBUFFER_ERROR	6	Insufficient memory, failure to execute the instruction
HOU_PENDING_ERROR	10	Instruction calling not completed
HOU_NUMPENDING_ERROR	11	Excessive instruction calls
HOU_NOTIMPLEMENTED_ERROR	12	Instruction not implemented
HOU_INVALIDID_ERROR	13	Invalid ID input
HOU_OVERFLOW_ERROR	14	Integer overflow
HOU_BUFFERSIZE_ERROR	15	Buffer too small or invalid
HOU_NOOBJECT_ERROR	16	Input named object not existing
HOU_NOMEMORY_ERROR	17	No heap memory available
HOU_DUPLICATE_ERROR	18	Another object with the same name existing
HOU_MEMORYOVERWRITE_ERROR	19	Heap memory write out of bounds, memory overwrite error
HOU_INVALIDHANDLE_ERROR	20	Invalid object handle
HOU_ENDOFOBJECT_ERROR	21	Object end reached
HOU_NOCHANGE_ERROR	22	No change
HOU_INVALIDINTERFACE_ERROR	23	Invalid or unknown interface
HOU_NOTSUPPORTED_ERROR	24	Function not supported
HOU_NOACCESSRIGHTS_ERROR	25	No access permission
HOU_OUTOFLIMITS_ERROR	26	Resource limit exceeded
HOU_ENTRIESREMAINING_ERROR	27	Buffer limit causing transmission incompletion
HOU_INVALIDSESSIONID_ERROR	28	Invalid online session ID
HOU_EXCEPTION_ERROR	29	Exception
HOU_INVALIDPATHNAME_ERROR	33	Path name invalid
HOU_INVALIDFILENAME_ERROR	34	File name invalid
HOU_INVALIDLISTMODE_ERROR	35	Invalid control command

eErrorID	Value	Description
HOU_FILEOVERWRITE_ERROR	50	File overwrite error
HOU_OVERBUFFERSIZE_ERROR	1030	Data area too small
HOU_SAMEFILENAME_ERROR	1403	File with the same name existing

4.6. 26 FileRemove2

This instruction deletes the specified file in the SD memory card.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
FileRemove2	Delete file	FB	FileRemove2_0 FileRemove2 EN ENO Execute Done FileName Busy Error ErrorID	FileRemove2_0(Execute :=, FileName :=, Done =>, Busy =>, Error =>, ErrorID =>,);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
Execute	Enable	BOOL	Depends on data types	-	(Triggered by edges) Execution of the function block switch
FileName	Specified file name	STRING	Maximum 1,985 bytes	-	Name of the file to delete

Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
Done	Function block completed state	BOOL	Depends on data types	-	Function block completed state
Busy	Function block execution flag	BOOL	Depends on data types	-	Function block execution flag
Error	Function block error flag	BOOL	Depends on data types	-	Function block error flag
ErrorID	Function block error ID	HOU_ERROR	Depends on data types	-	Function block error ID

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Execute	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
FileName	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Done	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Busy	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Error	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
ErrorID	See "Function" for the enumerators for the enumerated type HOU_ERROR.																			

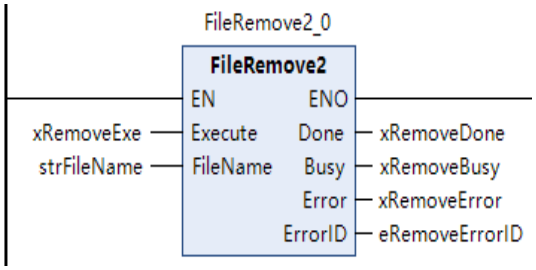
■ Function

This instruction deletes the file specified by file name FileName from the SD memory card.

■ Program example

Item	LD	ST
Defined variable		
Program		
Running result		

```
VAR
    FileRemove2_0   : FileRemove2;
    xRemoveExe      : BOOL;
    strFileName     : STRING(1985) := 'FileTest/File_Copy.txt';
    xRemoveDone     : BOOL;
    xRemoveBusy     : BOOL;
    xRemoveError    : BOOL;
    eRemoveErrorID  : HOU_ERROR;
END_VAR
```




```
FileRemove2_0(Execute:= xRemoveExe,
    FileName:= strFileName,
    Done=> xRemoveDone,
    Busy=> xRemoveBusy,
    Error=> xRemoveError,
    ErrorID=> eRemoveErrorID);
```

FileRemove2_0	FileRemove2	
xRemoveExe	BOOL	TRUE
strFileName	STRING(1985)	'FileTest/File_Copy.txt'
xRemoveDone	BOOL	TRUE
xRemoveBusy	BOOL	FALSE
xRemoveError	BOOL	FALSE
eRemoveErrorID	HOU_ERROR	HOU_NO_ERROR

■ Precautions

Execution of this instruction is continued until processing is completed even if the value of Execute changes to FALSE or the execution time exceeds the task period. The value of Done changes to TRUE when processing is completed. Use this to confirm normal completion of processing.

When the file is opened, remove the SD memory card. Then the file remains the open state. When you insert the SD memory card again, you cannot read or write the file. To read and write the file, re-open the file.

An error occurs in the following cases. The value of Error is TRUE.

1) The SD memory card is unavailable.
2) The SD memory card is write-protected.
3) The file specified by FileName does not exist.
4) The file specified by FileName is being accessed.
5) A file with the name specified by FileName exits and is read-only.
6) An error that prevents access occurs during SD memory card access.

■ Error ID

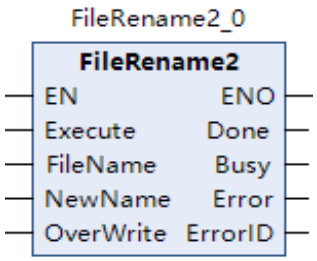
eErrorID	Value	Description
HOU_NO_ERROR	0	No error
HOU_INTERNAL_ERROR	1	General internal error
HOU_INVALIDPARAMETER_ERROR	2	Invalid parameter of the instruction
HOU_NOTINITIALIZED_ERROR	3	Component not initialized, failure to execute the instruction
HOU_VERSION_ERROR	4	Version conflict
HOU_TIMEOUT_ERROR	5	Instruction timeout
HOU_NOBUFFER_ERROR	6	Insufficient memory, failure to execute the instruction
HOU_PENDING_ERROR	10	Instruction calling not completed
HOU_NUMPENDING_ERROR	11	Excessive instruction calls
HOU_NOTIMPLEMENTED_ERROR	12	Instruction not implemented
HOU_INVALIDID_ERROR	13	Invalid ID input
HOU_OVERFLOW_ERROR	14	Integer overflow
HOU_BUFFERSIZE_ERROR	15	Buffer too small or invalid
HOU_NOOBJECT_ERROR	16	Input named object not existing
HOU_NOMEMORY_ERROR	17	No heap memory available
HOU_DUPLICATE_ERROR	18	Another object with the same name existing
HOU_MEMORYOVERWRITE_ERROR	19	Heap memory write out of bounds, memory overwrite error
HOU_INVALIDHANDLE_ERROR	20	Invalid object handle
HOU_ENDOFOBJECT_ERROR	21	Object end reached
HOU_NOCHANGE_ERROR	22	No change
HOU_INVALIDINTERFACE_ERROR	23	Invalid or unknown interface
HOU_NOTSUPPORTED_ERROR	24	Function not supported
HOU_NOACCESSRIGHTS_ERROR	25	No access permission
HOU_OUTOFLIMITS_ERROR	26	Resource limit exceeded
HOU_ENTRIESREMAINING_ERROR	27	Buffer limit causing transmission incompletion
HOU_INVALIDSESSIONID_ERROR	28	Invalid online session ID
HOU_EXCEPTION_ERROR	29	Exception
HOU_INVALIDPATHNAME_ERROR	33	Path name invalid
HOU_INVALIDFILENAME_ERROR	34	File name invalid
HOU_INVALIDLISTMODE_ERROR	35	Invalid control command

eErrorID	Value	Description
HOU_FILEOVERWRITE_ERROR	50	File overwrite error
HOU_OVERBUFFERSIZE_ERROR	1030	Data area too small
HOU_SAMEFILENAME_ERROR	1403	File with the same name existing

4.6. 27 FileRename2

This instruction deletes the specified file in the SD memory card.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
FileRename2	Change file name	FB		FileRename2_0(Execute :=, FileName :=, NewName :=, OverWrite :=, Done =>, Busy =>, Error =>, ErrorID =>,);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
Execute	Enable	BOOL	Depends on data types	-	(Triggered by edges) Execution of the function block switch
FileName	Original file name	STRING	Maximum 1,985 bytes	-	Original file name
NewName	New file name	STRING	Maximum 1,985 bytes	-	New file name
OverWrite	Overwrite enable	BOOL	Depends on data types	-	TRUE: Enable overwrite FALSE: Prohibit overwrite

Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
Done	Function block completed state	BOOL	Depends on data types	-	Function block completed state
Busy	Function block execution flag	BOOL	Depends on data types	-	Function block execution flag
Error	Function block error flag	BOOL	Depends on data types	-	Function block error flag
ErrorID	Function block error ID	HOU_ERROR	Depends on data types	-	Function block error ID

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Execute	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
FileName	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√
NewName	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√
OverWrite	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Done	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Busy	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Error	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
ErrorID	See "Function" for the enumerators for the enumerated type HOU_ERROR.																			

■ Function

This instruction changes the original name of the file or directory specified by FileName in the SD memory card to new file name specified by NewName.

If a file with the name specified by NewName does not exist in the SD memory card, it is created.

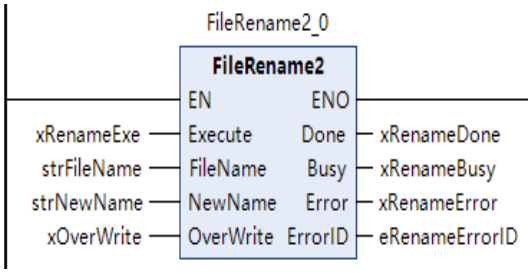
If a file with the name specified by NewName already exists in the SD memory card, the following processing is performed depending on the value of OverWrite.

Value of OverWrite	Operation
TRUE	The file or directory is overwritten.
FALSE	The file or directory is not overwritten and an error occurs.

■ Program example

Item	LD	ST
Defined variable		

```
VAR
    FileRename2_0       : FileRename2;
    xRenameExe          : BOOL;
    strFileName         : STRING(1985) := 'FileTest/File.txt';
    strNewName          : STRING(1985) := 'FileTest/File_New.txt';
    xOverWrite          : BOOL;
    xRenameDone         : BOOL;
    xRenameBusy         : BOOL;
    xRenameError        : BOOL;
    eRenameErrorID      : HOU_ERROR;
END_VAR
```

<table>
<tr><th>Item</th><th>LD</th><th>ST</th></tr>
<tr><td>Program</td><td> 
</td><td>

```
FileRename2_0(
    Execute:= xRenameExe,
    FileName:= strFileName,
    NewName:= strNewName,
    OverWrite:= xOverWrite,
    Done=> xRenameDone,
    Busy=> xRenameBusy,
    Error=> xRenameError,
    ErrorID=> eRenameErrorID);
```

</td></tr>
<tr><td>Running result</td><td colspan="2">
<table>
<tr><td>FileRename2_0</td><td>FileRename2</td><td></td></tr>
<tr><td>xRenameExe</td><td>BOOL</td><td>TRUE</td></tr>
<tr><td>strFileName</td><td>STRING(1985)</td><td>'FileTest/File.txt'</td></tr>
<tr><td>strNewName</td><td>STRING(1985)</td><td>'FileTest/File_New.txt'</td></tr>
<tr><td>xOverWrite</td><td>BOOL</td><td>FALSE</td></tr>
<tr><td>xRenameDone</td><td>BOOL</td><td>TRUE</td></tr>
<tr><td>xRenameBusy</td><td>BOOL</td><td>FALSE</td></tr>
<tr><td>xRenameError</td><td>BOOL</td><td>FALSE</td></tr>
<tr><td>eRenameErrorID</td><td>HOU_ERROR</td><td>HOU_NO_ERROR</td></tr>
</table>
</td></tr>
</table>

■ Precautions

Execution of this instruction is continued until processing is completed even if the value of Execute changes to FALSE or the execution time exceeds the task period. The value of Done changes to TRUE when processing is completed. Use this to confirm normal completion of processing.

If the directories are different for FileName and NewName, the file is moved to the directory specified by NewName.

When the file is opened, remove the SD memory card. Then the file remains the open state. When you insert the SD memory card again, you cannot read or write the file. To read and write the file, re-open the file.

An error occurs in the following cases. The value of Error is TRUE.

1) The SD memory card is unavailable.
2) The SD memory card is write-protected.
3) The file specified by FileName does not exist.
4) The value of FileName or NewName is not a valid file name.
5) The file specified by FileName is being accessed.
6) The maximum number of files or directories is exceeded.
7) A file with the name specified by NewName exits and the value of OverWrite is FALSE.
8) A file with the name specified by DstFileName exits, the value of OverWrite is TRUE, and the file is read-only.

9) An error that prevents access occurs during SD memory card access.

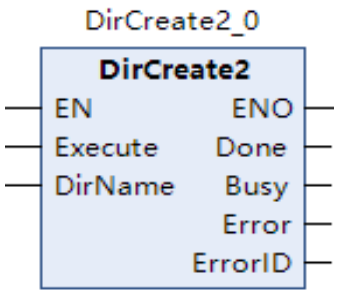
■ Error ID

eErrorID	Value	Description
HOU_NO_ERROR	0	No error
HOU_INTERNAL_ERROR	1	General internal error
HOU_INVALIDPARAMETER_ERROR	2	Invalid parameter of the instruction
HOU_NOTINITIALIZED_ERROR	3	Component not initialized, failure to execute the instruction
HOU_VERSION_ERROR	4	Version conflict
HOU_TIMEOUT_ERROR	5	Instruction timeout
HOU_NOBUFFER_ERROR	6	Insufficient memory, failure to execute the instruction
HOU_PENDING_ERROR	10	Instruction calling not completed
HOU_NUMPENDING_ERROR	11	Excessive instruction calls
HOU_NOTIMPLEMENTED_ERROR	12	Instruction not implemented
HOU_INVALIDID_ERROR	13	Invalid ID input
HOU_OVERFLOW_ERROR	14	Integer overflow
HOU_BUFFERSIZE_ERROR	15	Buffer too small or invalid
HOU_NOOBJECT_ERROR	16	Input named object not existing
HOU_NOMEMORY_ERROR	17	No heap memory available
HOU_DUPLICATE_ERROR	18	Another object with the same name existing
HOU_MEMORYOVERWRITE_ERROR	19	Heap memory write out of bounds, memory overwrite error
HOU_INVALIDHANDLE_ERROR	20	Invalid object handle
HOU_ENDOFOBJECT_ERROR	21	Object end reached
HOU_NOCHANGE_ERROR	22	No change
HOU_INVALIDINTERFACE_ERROR	23	Invalid or unknown interface
HOU_NOTSUPPORTED_ERROR	24	Function not supported
HOU_NOACCESSRIGHTS_ERROR	25	No access permission
HOU_OUTOFLIMITS_ERROR	26	Resource limit exceeded
HOU_ENTRIESREMAINING_ERROR	27	Buffer limit causing transmission incompletion
HOU_INVALIDSESSIONID_ERROR	28	Invalid online session ID
HOU_EXCEPTION_ERROR	29	Exception
HOU_INVALIDPATHNAME_ERROR	33	Path name invalid
HOU_INVALIDFILENAME_ERROR	34	File name invalid
HOU_INVALIDLISTMODE_ERROR	35	Invalid control command
HOU_FILEOVERWRITE_ERROR	50	File overwrite error
HOU_OVERBUFFERSIZE_ERROR	1030	Data area too small
HOU_SAMEFILENAME_ERROR	1403	File with the same name existing

4.6. 28 DirCreate2

This instruction creates a directory with the specified name in the SD memory card.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
DirCreate2	Create directory	FB		DirCreate2_0(Execute :=, DirName :=, Done =>, Busy =>, Error =>, ErrorID =>,);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
Execute	Enable	BOOL	Depends on data types	-	(Triggered by edges) Execution of the function block switch
DirName	Specified directory name	STRING	Maximum 1,985 bytes	-	Name of the directory to create

Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
Done	Function block completed state	BOOL	Depends on data types	-	Function block completed state
Busy	Function block execution flag	BOOL	Depends on data types	-	Function block execution flag
Error	Function block error flag	BOOL	Depends on data types	-	Function block error flag
ErrorID	Function block error ID	HOU_ERROR	Depends on data types	-	Function block error ID

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Execute	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
DirName	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√
Done	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Busy	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Error	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
ErrorID	See "Function" for the enumerators for the enumerated type HOU_ERROR.																			

■ Function

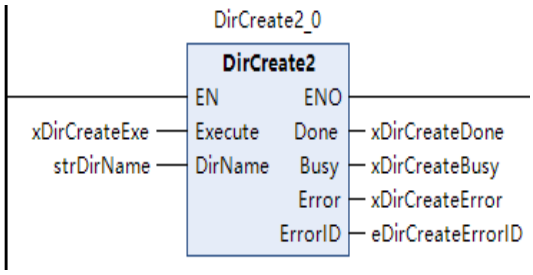
This instruction creates a directory specified by DirName in the SD memory card.

■ Program example

<table>
<tr><th>Item</th><th>LD</th><th>ST</th></tr>
<tr><td>Defined variable</td><td colspan="2">

```
VAR
    DirCreate2_0        : DirCreate2;
    xDirCreateExe       : BOOL;
    strDirName          : STRING(1985) := 'FileTest/DieNew';
    xDirCreateDone      : BOOL;
    xDirCreateBusy      : BOOL;
    xDirCreateError     : BOOL;
    eDirCreateErrorID   : HOU_ERROR;
END_VAR
```

</td></tr>
<tr><td>Program</td><td>



</td><td>

```
DirCreate2_0(Execute:= xDirCreateExe,
    DirName:= strDirName,
    Done=> xDirCreateDone,
    Busy=> xDirCreateBusy,
    Error=> xDirCreateError,
    ErrorID=> eDirCreateErrorID);
```

</td></tr>
<tr><td>Running result</td><td colspan="2">

DirCreate2_0	DirCreate2	
xDirCreateExe	BOOL	TRUE
strDirName	STRING(1985)	'FileTest/DieNew'
xDirCreateDone	BOOL	TRUE
xDirCreateBusy	BOOL	FALSE
xDirCreateError	BOOL	FALSE
eDirCreateErrorID	HOU_ERROR	HOU_NO_ERROR

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

■ Precautions

Execution of this instruction is continued until processing is completed even if the value of Execute changes to FALSE or the execution time exceeds the task period. The value of Done changes to TRUE when processing is completed. Use this to confirm normal completion of processing.

When the file is opened, remove the SD memory card. Then the file remains the open state. When you insert the SD memory card again, you cannot read or write the file. To read and write the file, re-open the file.

An error occurs in the following cases. The value of Error is TRUE.

1) The SD memory card is unavailable.
2) The SD memory card is write-protected.
3) There is insufficient space available in the SD memory card.
4) The maximum number of directories is exceeded.
5) The directory specified by DirName exists.
6) The value of DirName is not a valid directory name.
7) An error that prevents access occurs during SD memory card access.

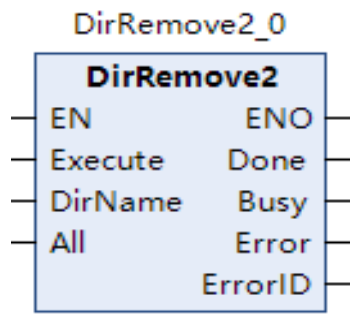
■ Error ID

eErrorID	Value	Description
HOU_NO_ERROR	0	No error
HOU_INTERNAL_ERROR	1	General internal error
HOU_INVALIDPARAMETER_ERROR	2	Invalid parameter of the instruction
HOU_NOTINITIALIZED_ERROR	3	Component not initialized, failure to execute the instruction
HOU_VERSION_ERROR	4	Version conflict
HOU_TIMEOUT_ERROR	5	Instruction timeout
HOU_NOBUFFER_ERROR	6	Insufficient memory, failure to execute the instruction
HOU_PENDING_ERROR	10	Instruction calling not completed
HOU_NUMPENDING_ERROR	11	Excessive instruction calls
HOU_NOTIMPLEMENTED_ERROR	12	Instruction not implemented
HOU_INVALIDID_ERROR	13	Invalid ID input
HOU_OVERFLOW_ERROR	14	Integer overflow
HOU_BUFFERSIZE_ERROR	15	Buffer too small or invalid
HOU_NOOBJECT_ERROR	16	Input named object not existing
HOU_NOMEMORY_ERROR	17	No heap memory available
HOU_DUPLICATE_ERROR	18	Another object with the same name existing
HOU_MEMORYOVERWRITE_ERROR	19	Heap memory write out of bounds, memory overwrite error
HOU_INVALIDHANDLE_ERROR	20	Invalid object handle
HOU_ENDOFOBJECT_ERROR	21	Object end reached
HOU_NOCHANGE_ERROR	22	No change
HOU_INVALIDINTERFACE_ERROR	23	Invalid or unknown interface
HOU_NOTSUPPORTED_ERROR	24	Function not supported
HOU_NOACCESSRIGHTS_ERROR	25	No access permission
HOU_OUTOFLIMITS_ERROR	26	Resource limit exceeded
HOU_ENTRIESREMAINING_ERROR	27	Buffer limit causing transmission incompletion
HOU_INVALIDSESSIONID_ERROR	28	Invalid online session ID
HOU_EXCEPTION_ERROR	29	Exception
HOU_INVALIDPATHNAME_ERROR	33	Path name invalid
HOU_INVALIDFILENAME_ERROR	34	File name invalid
HOU_INVALIDLISTMODE_ERROR	35	Invalid control command
HOU_FILEOVERWRITE_ERROR	50	File overwrite error
HOU_OVERBUFFERSIZE_ERROR	1030	Data area too small
HOU_SAMEFILENAME_ERROR	1403	File with the same name existing

4.6. 29 DirRemove2

This instruction deletes the specified directory in the SD memory card.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
DirRemove2	Delete directory	FB		DirRemove2_0(Execute :=, DirName :=, All :=, Done =>, Busy =>, Error =>, ErrorID =>,);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
Execute	Enable	BOOL	Depends on data types	-	(Triggered by edges) Execution of the function block switch
FileName	Directory to delete	STRING	Maximum 1,985 bytes	-	Directory to delete
All	Select all	BOOL	Depends on data types	-	Whether to delete a directory that contains files and subdirectories TRUE: Delete FALSE: Do not delete

Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
Done	Function block completed state	BOOL	Depends on data types	-	Function block completed state
Busy	Function block execution flag	BOOL	Depends on data types	-	Function block execution flag
Error	Function block error flag	BOOL	Depends on data types	-	Function block error flag
ErrorID	Function block error ID	HOU_ERROR	Depends on data types	-	Function block error ID

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Execute	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
DirName	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√
All	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Done	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Busy	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Error	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
ErrorID	See "Function" for the enumerators for the enumerated type HOU_ERROR.																			

■ Function

This instruction deletes the directory specified by DirName in the SD memory card.

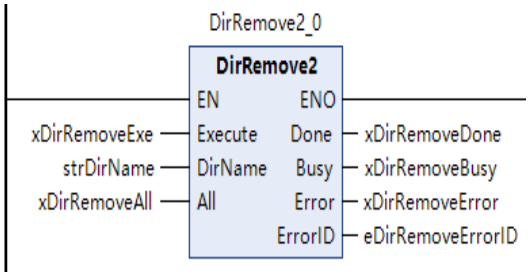
If there are files or subdirectories in the specified directory, the following processing is performed according to the value of All.

Value of All	Operation
TRUE	The specified directory is deleted, including all files and subdirectories in it.
FALSE	The specified directory is not deleted and an error occurs.

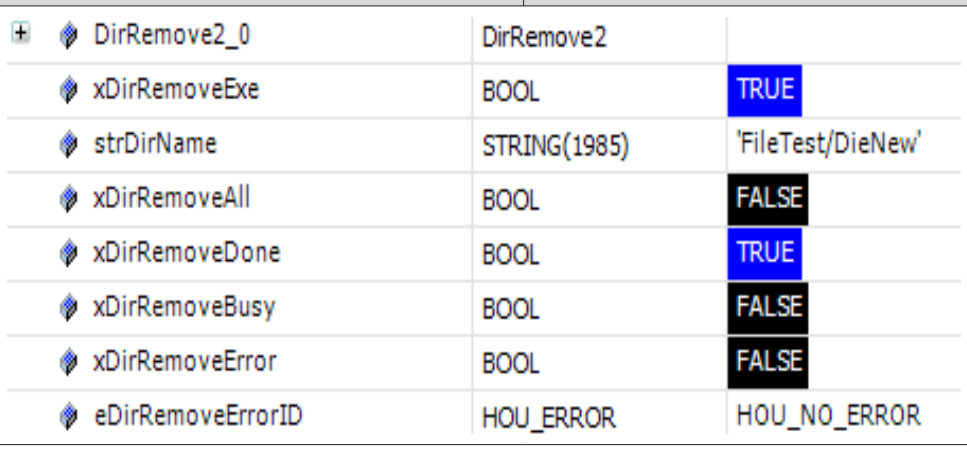
■ Program example

Item	LD	ST
Defined variable		
Program		

```
VAR
    DirRemove2_0        : DirRemove2;
    xDirRemoveExe       : BOOL;
    strDirName          : STRING(1985) := 'FileTest/DieNew';
    xDirRemoveAll       : BOOL;
    xDirRemoveDone      : BOOL;
    xDirRemoveBusy      : BOOL;
    xDirRemoveError     : BOOL;
    eDirRemoveErrorID   : HOU_ERROR;
END_VAR
```




```
DirRemove2_0(
    Execute:= xDirRemoveExe,
    DirName:= strDirName,
    All:= xDirRemoveAll,
    Done=> xDirRemoveDone,
    Busy=> xDirRemoveBusy,
    Error=> xDirRemoveError,
    ErrorID=> eDirRemoveErrorID);
```

Item	LD	ST
Running result		

■ Precautions

Execution of this instruction is continued until processing is completed even if the value of Execute changes to FALSE or the execution time exceeds the task period. The value of Done changes to TRUE when processing is completed. Use this to confirm normal completion of processing.

When the file is opened, remove the SD memory card. Then the file remains the open state. When you insert the SD memory card again, you cannot read or write the file. To read and write the file, re-open the file.

An error occurs in the following cases. The value of Error is TRUE.

1) The SD memory card is unavailable.
2) The SD memory card is write-protected.
3) If the value of All is TRUE and the directory specified by DirName is being accessed by another instruction.
4) If the value of All is FALSE and the directory specified by DirName contains a file or subdirectory.
5) The directory specified by DirName is read-only.
6) The file specified by DirName does not exist.
7) An error that prevents access occurs during SD memory card access.

■ Error ID

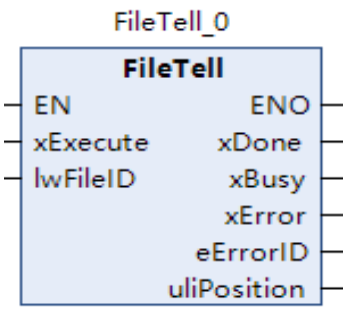
eErrorID	Value	Description
HOU_NO_ERROR	0	No error
HOU_INTERNAL_ERROR	1	General internal error
HOU_INVALIDPARAMETER_ERROR	2	Invalid parameter of the instruction
HOU_NOTINITIALIZED_ERROR	3	Component not initialized, failure to execute the instruction
HOU_VERSION_ERROR	4	Version conflict
HOU_TIMEOUT_ERROR	5	Instruction timeout
HOU_NOBUFFER_ERROR	6	Insufficient memory, failure to execute the instruction
HOU_PENDING_ERROR	10	Instruction calling not completed
HOU_NUMPENDING_ERROR	11	Excessive instruction calls
HOU_NOTIMPLEMENTED_ERROR	12	Instruction not implemented
HOU_INVALIDID_ERROR	13	Invalid ID input

eErrorID	Value	Description
HOU_OVERFLOW_ERROR	14	Integer overflow
HOU_BUFFERSIZE_ERROR	15	Buffer too small or invalid
HOU_NOOBJECT_ERROR	16	Input named object not existing
HOU_NOMEMORY_ERROR	17	No heap memory available
HOU_DUPLICATE_ERROR	18	Another object with the same name existing
HOU_MEMORYOVERWRITE_ERROR	19	Heap memory write out of bounds, memory overwrite error
HOU_INVALIDHANDLE_ERROR	20	Invalid object handle
HOU_ENDOFOBJECT_ERROR	21	Object end reached
HOU_NOCHANGE_ERROR	22	No change
HOU_INVALIDINTERFACE_ERROR	23	Invalid or unknown interface
HOU_NOTSUPPORTED_ERROR	24	Function not supported
HOU_NOACCESSRIGHTS_ERROR	25	No access permission
HOU_OUTOFLIMITS_ERROR	26	Resource limit exceeded
HOU_ENTRIESREMAINING_ERROR	27	Buffer limit causing transmission incompletion
HOU_INVALIDSESSIONID_ERROR	28	Invalid online session ID
HOU_EXCEPTION_ERROR	29	Exception
HOU_INVALIDPATHNAME_ERROR	33	Path name invalid
HOU_INVALIDFILENAME_ERROR	34	File name invalid
HOU_INVALIDLISTMODE_ERROR	35	Invalid control command
HOU_FILEOVERWRITE_ERROR	50	File overwrite error
HOU_OVERBUFFERSIZE_ERROR	1030	Data area too small
HOU_SAMEFILENAME_ERROR	1403	File with the same name existing

4.6. 30 FileTell

This instruction acquires the position of the file indicator in the specified file in the SD memory card.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
FileTell	Acquire position of file indicator	FB		FileTell_0(xExecute :=, lwFileID :=, xDone =>, xBusy =>, xError =>, eErrorID =>, uliPosition =>);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
xExecute	Enable	BOOL	Depends on data types	-	Enable
lwFileID	File ID	LWORD	Depends on data types	-	File ID

Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
xDone	Completion flag of the function block	BOOL	Depends on data types	-	Completion flag of the function block
xBusy	Function block execution flag	BOOL	Depends on data types	-	Function block execution flag
xError	Function block error flag	BOOL	Depends on data types	-	Function block error flag
eErrorID	Function block error ID	HOU_ERROR	Depends on data types	-	Function block error ID
uliPosition	Position in file	ULINT	Depends on data types	-	Position in file

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
xExecute	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
lwFileID	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
xDone	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
xBusy	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
xError	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Error	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
ErrorID	HOU_ERROR																			
uliPosition	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-

■ Function

This instruction acquires the position of the file position indicator in the file specified by file ID lwFileID in the SD memory card. A file position indicator is the position in a file at which to start reading or writing when an instruction such as the FileRead or FileWrite instruction is executed.

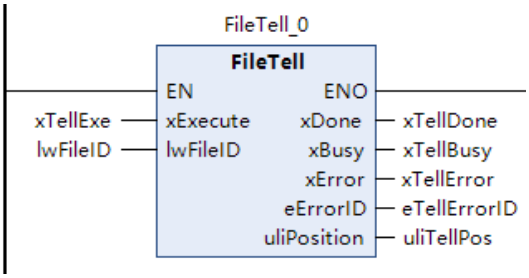
The file ID lwFileID is acquired by the FileOpen2 instruction.

■ Program example

<table>
<tr><th>Item</th><th>LD</th><th>ST</th></tr>
<tr><td>Defined variable</td><td colspan="2">

```
VAR
    FileTell_0      : FileTell;
    xTellExe        : BOOL;
    lwFileID        : LWORD := 3015706376;
    xTellDone       : BOOL;
    xTellBusy       : BOOL;
    xTellError      : BOOL;
    eTellErrorID    : HOU_ERROR;
    uliTellPos      : ULINT;
END_VAR
```

</td></tr>
<tr><td>Program</td><td>



</td><td>

```
FileTell_0(
    xExecute:= xTellExe,
    lwFileID:= lwFileID,
    xDone=> xTellDone,
    xBusy=> xTellBusy,
    xError=> xTellError,
    eErrorID=> eTellErrorID,
    uliPosition=> uliTellPos);
```

</td></tr>
<tr><td>Running result</td><td colspan="2">

FileTell_0	FileTell	
xTellExe	BOOL	TRUE
lwFileID	LWORD	3015706376
xTellDone	BOOL	TRUE
xTellBusy	BOOL	FALSE
xTellError	BOOL	FALSE
eTellErrorID	HOU_ERROR	HOU_NO_ERROR
uliTellPos	ULINT	15

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

■ Precautions

Execution of this instruction is continued until processing is completed even if the value of xExecute changes to FALSE or the execution time exceeds the task period. The value of xDone changes to TRUE when processing is completed. Use this to confirm normal completion of processing.

Execute the FileOpen2 instruction in advance to obtain the value for lwFileID.

An error occurs in the following cases. The value of xError is TRUE.

1) The SD memory card is unavailable.
2) The file specified by lwFileID is being accessed.
3) The file specified by lwFileID does not exist.
4) An error that prevents access occurs during SD memory card access.

■ Error ID

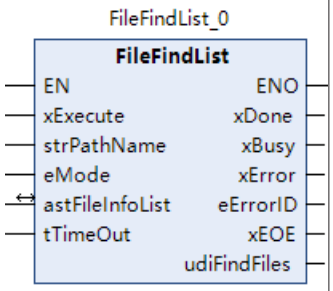
eErrorID	Value	Description
HOU_NO_ERROR	0	No error
HOU_INTERNAL_ERROR	1	General internal error

eErrorID	Value	Description
HOU_INVALIDPARAMETER_ERROR	2	Invalid parameter of the instruction
HOU_NOTINITIALIZED_ERROR	3	Component not initialized, failure to execute the instruction
HOU_VERSION_ERROR	4	Version conflict
HOU_TIMEOUT_ERROR	5	Instruction timeout
HOU_NOBUFFER_ERROR	6	Insufficient memory, failure to execute the instruction
HOU_PENDING_ERROR	10	Instruction calling not completed
HOU_NUMPENDING_ERROR	11	Excessive instruction calls
HOU_NOTIMPLEMENTED_ERROR	12	Instruction not implemented
HOU_INVALIDID_ERROR	13	Invalid ID input
HOU_OVERFLOW_ERROR	14	Integer overflow
HOU_BUFFERSIZE_ERROR	15	Buffer too small or invalid
HOU_NOOBJECT_ERROR	16	Input named object not existing
HOU_NOMEMORY_ERROR	17	No heap memory available
HOU_DUPLICATE_ERROR	18	Another object with the same name existing
HOU_MEMORYOVERWRITE_ERROR	19	Heap memory write out of bounds, memory overwrite error
HOU_INVALIDHANDLE_ERROR	20	Invalid object handle
HOU_ENDOFOBJECT_ERROR	21	Object end reached
HOU_NOCHANGE_ERROR	22	No change
HOU_INVALIDINTERFACE_ERROR	23	Invalid or unknown interface
HOU_NOTSUPPORTED_ERROR	24	Function not supported
HOU_NOACCESSRIGHTS_ERROR	25	No access permission
HOU_OUTOFLIMITS_ERROR	26	Resource limit exceeded
HOU_ENTRIESREMAINING_ERROR	27	Buffer limit causing transmission incompletion
HOU_INVALIDSESSIONID_ERROR	28	Invalid online session ID
HOU_EXCEPTION_ERROR	29	Exception
HOU_INVALIDPATHNAME_ERROR	33	Path name invalid
HOU_INVALIDFILENAME_ERROR	34	File name invalid
HOU_INVALIDLISTMODE_ERROR	35	Invalid control command
HOU_FILEOVERWRITE_ERROR	50	File overwrite error
HOU_OVERBUFFERSIZE_ERROR	1030	Data area too small
HOU_SAMEFILENAME_ERROR	1403	File with the same name existing

4.6. 31 FileFindList

This instruction queries all files or subdirectories with specified names in the path.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
FileFindList	Query all files	FB		FileFindList_0(xExecute :=, strPathName :=, eMode :=, astFileInfoList :=, tTimeOut :=, xDone =>, xBusy =>, xError =>, eErrorID =>, xEOE =>, udiFindFiles =>,);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
xExecute	Enable	BOOL	Depends on data types	-	Enable
strPathName	Path name	STRING[255]	Depends on data types	-	Path name
eMode	Query control command	FILELIST_MODE	Depends on data types	-	Query of the control command in the path: LISTCMD_FIRST LISTCMD_NEXT LISTCMD_ABORT
tTimeOut	Timeout	TIME	Depends on data types	T#5S	Timeout

In-out variables

In-Out Variable	Name	Data Type	Value Range	Initial Value	Description
astFileInfoList	File information array	Array [*] of FILEINFO	Depends on data types	-	File information array

Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
xDone	Completion flag of the function block	BOOL	Depends on data types	-	Completion flag of the function block
xBusy	Function block execution flag	BOOL	Depends on data types	-	Function block execution flag
xError	Function block error flag	BOOL	Depends on data types	-	Function block error flag
eErrorID	Function block error ID	HOU_ERROR	Depends on data types	-	Function block error ID
xEOE	End of path enumeration	BOOL	Depends on data types	-	End of path enumeration
udiFindFiles	Path enumeration count	UDINT	Depends on data types	-	Path enumeration count

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
xExecute	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
strPathName	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√
eMode	FILELIST_MODE enumerators																			
astFileInfoList	Array [*] of FILEINFO structure array																			
tTimeOut	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-
xDone	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
xBusy	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
xError	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
eErrorID	HOU_ERROR enumerators																			
xEOE	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
udiFindFiles	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-

■ Function

This instruction queries the files or subdirectories with names similar to the specified names in the path.

The input parameter strPathName specifies the target path and the name of the entry to be searched for. The file name to be searched for supports two types of wildcards (* and ?).

The input parameter eMode, whose data type is the FILELIST_MODE enumerator, specifies whether the search is conducted for the first time or in sequential mode. It is used to navigate in the entry list. The meanings of the enumerators are as follows.

Enumerator	Meaning
LISTCMD_FIRST	It means to search for the first time. The number of files found is determined by the size of the astFileInfoList array and the number of similar files in the path.
LISTCMD_NEXT	It means to search in sequential mode. After the first search is completed, it is called multiple times to find all similar files. The number of files found is determined by the size of the astFileInfoList array and the number of similar files in the path.
LISTCMD_ABORT	It means to abort the search and close the related path handle.

The in-out parameter astFileInfoList stores the information of the files found. Its data type is an array of structures. The meanings of the FILEINFO structure are as follows.

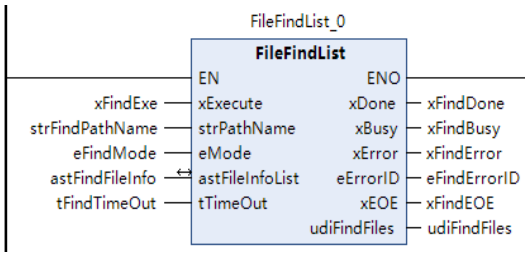
Structure Member	Data Type	Function Description
strFileName	STRING[255]	File or path name.
xDirectory	BOOL	File or directory.
uliFileSize	ULINT	File size, in bytes.
dtCreationTime	DT	Creation time of the file or directory.
dtLastAccessTime	DT	Last access time of the file or directory.
dtLastModificationTime	DT	Last write time of the file or directory.

■ Program example

<table>
<tr><th>Item</th><th>LD</th><th>ST</th></tr>
<tr><td>Defined variable</td><td colspan="2">

```
VAR
    FileFindList_0  : FileFindList;
    xFindExe        : BOOL;
    strFindPathName : STRING[255]             := 'Application/*txt';
    eFindMode       : FILELIST_MODE           := FILELIST_MODE.LISTCMD_FIRST;
    astFindFileInfo : ARRAY [1..5] OF FileInfo;
    tFindTimeOut    : TIME                    := T#5S;
    xFindDone       : BOOL;
    xFindBusy       : BOOL;
    xFindError      : BOOL;
    eFindErrorID    : HOU_ERROR;
    xFindEOE        : BOOL;
    udiFindFiles    : UDINT;
END_VAR
```

</td></tr>
<tr><td>Program</td><td>



</td><td>

```
FileFindList_0(
    xExecute:= xFindExe,
    strPathName:= strFindPathName,
    eMode:= eFindMode,
    astFileInfoList:= astFindFileInfo,
    tTimeOut:= tFindTimeOut,
    xDone=> xFindDone,
    xBusy=> xFindBusy,
    xError=> xFindError,
    eErrorID=> eFindErrorID,
    xEOE=> xFindEOE,
    udiFindFiles=> udiFindFiles);
```

</td></tr>
<tr><td>Running result</td><td colspan="2">

FileFindList_0	FileFindList	
xFindExe	BOOL	TRUE
strFindPathName	STRING(255)	'Application/*txt'
eFindMode	FILELIST_MODE	LISTCMD_FIRST
astFindFileInfo	ARRAY [1..5] OF Fil...	
astFindFileInfo[1]	FileInfo	
strFileName	STRING(255)	'test.txt'
xDirectory	BOOL	FALSE
uliFileSize	ULINT	90
dtCreationTi...	DATE_AND_TIME	DT#2023-12-11-15:28:28
dtLastAcces...	DATE_AND_TIME	DT#2023-12-11-15:28:28
dtLastModifi...	DATE_AND_TIME	DT#2023-12-11-15:28:28
astFindFileInfo[2]	FileInfo	
astFindFileInfo[3]	FileInfo	
astFindFileInfo[4]	FileInfo	
astFindFileInfo[5]	FileInfo	
tFindTimeOut	TIME	T#5s
xFindDone	BOOL	TRUE
xFindBusy	BOOL	FALSE
xFindError	BOOL	FALSE
eFindErrorID	HOU_ERROR	HOU_NO_ERROR
xFindEOE	BOOL	TRUE
udiFindFiles	UDINT	4

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

■ Precautions

Execution of this instruction is continued until processing is completed even if the value of xExecute changes to FALSE or the execution time exceeds the task period. The value of xDone changes to TRUE when processing is completed. Use this to confirm normal completion of processing.

To search for the path specified by strPathName, the user must have the access authority to the path and its subdirectories.

The path name specified by strPathName must consist of case-sensitive letters. Characters excluding

wildcards (* and ?) are valid data. Wildcards are only at the beginning or end of the file name. That is, the format for searching for the file name is *A, A*, or *A*.

When the found file name specified by strPathName is invalid, the value of xError is TRUE.

The file name specified by strPathName contains the path separators (/), and each path contains two entries "." and "..", respectively indicating the current directory and the parent directory.

xEOE indicates whether all similar files in the specified directory are found. The function block may need to be triggered multiple times to find all similar files in the specified folder. The search is completed only when xEOE is TRUE or when the search is prematurely terminated by setting eMode to LISTCMD_ABORT. If the required file or directory is found in the program, xEOE is FALSE. In this case, the value of the input parameter eCmd must be changed to LISTCMD_ABORT and the function block must be triggered again to release all internal resources.

bEOE turns to TRUE when the first attempt is made to read a non-existent entry.

tTimeOut specifies the timeout period for execution of the function block. When it is set to T#0S, the timeout function is not enabled.

A new search can begin only after the "search complete" signal xDone is set.

■ Error ID

eErrorID	Value	Description
HOU_NO_ERROR	0	No error
HOU_INTERNAL_ERROR	1	General internal error
HOU_INVALIDPARAMETER_ERROR	2	Invalid parameter of the instruction
HOU_NOTINITIALIZED_ERROR	3	Component not initialized, failure to execute the instruction
HOU_VERSION_ERROR	4	Version conflict
HOU_TIMEOUT_ERROR	5	Instruction timeout
HOU_NOBUFFER_ERROR	6	Insufficient memory, failure to execute the instruction
HOU_PENDING_ERROR	10	Instruction calling not completed
HOU_NUMPENDING_ERROR	11	Excessive instruction calls
HOU_NOTIMPLEMENTED_ERROR	12	Instruction not implemented
HOU_INVALIDID_ERROR	13	Invalid ID input
HOU_OVERFLOW_ERROR	14	Integer overflow
HOU_BUFFERSIZE_ERROR	15	Buffer too small or invalid
HOU_NOOBJECT_ERROR	16	Input named object not existing
HOU_NOMEMORY_ERROR	17	No heap memory available
HOU_DUPLICATE_ERROR	18	Another object with the same name existing
HOU_MEMORYOVERWRITE_ERROR	19	Heap memory write out of bounds, memory overwrite error
HOU_INVALIDHANDLE_ERROR	20	Invalid object handle
HOU_ENDOFOBJECT_ERROR	21	Object end reached
HOU_NOCHANGE_ERROR	22	No change
HOU_INVALIDINTERFACE_ERROR	23	Invalid or unknown interface
HOU_NOTSUPPORTED_ERROR	24	Function not supported
HOU_NOACCESSRIGHTS_ERROR	25	No access permission
HOU_OUTOFLIMITS_ERROR	26	Resource limit exceeded

eErrorID	Value	Description
HOU_ENTRIESREMAINING_ERROR	27	Buffer limit causing transmission incompletion
HOU_INVALIDSESSIONID_ERROR	28	Invalid online session ID
HOU_EXCEPTION_ERROR	29	Exception
HOU_INVALIDPATHNAME_ERROR	33	Path name invalid
HOU_INVALIDFILENAME_ERROR	34	File name invalid
HOU_INVALIDLISTMODE_ERROR	35	Invalid control command
HOU_FILEOVERWRITE_ERROR	50	File overwrite error
HOU_OVERBUFFERSIZE_ERROR	1030	Data area too small
HOU_SAMEFILENAME_ERROR	1403	File with the same name existing

4. 7 Analog Calculation Instructions

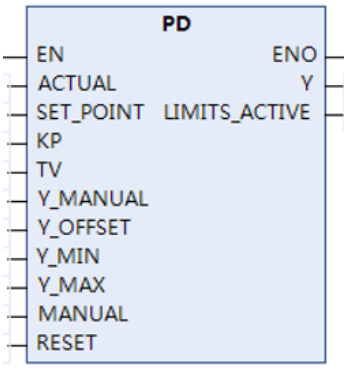
4.7. 1 Instruction List

Instruction Category	Name	FB/FC	Function
Analog calculation instructions	PD	FB	Universal PD control
	PID	FB	Universal PID control
	PID_FIXCYCLE	FB	Universal PID control that can manually set the cycle time

4.7. 2 PD

This instruction implements universal PD control.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
PD	Universal PD control	FB		PD_0(ACTUAL:= , SET_POINT:= , KP:= , TV:= , Y_MANUAL:= , Y_OFFSET:= , Y_MIN:= , Y_MAX:= , MANUAL:= , RESET:= , Y=> , LIMITS_ACTIVE=>);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
ACTUAL	Current value of variable to control	REAL	-	0	-
SET_POINT	Set point	REAL	-	0	-

KP	Scaling coefficient	REAL	-	0	-
TV	Differential time	REAL	-	0	-
Y_MANUAL	Manual output value	REAL	-	0	-
Y_OFFSET	Offset output of Y	REAL	-	0	-
Y_MIN	Minimum output value of Y	REAL	-	0	-
Y_MAX	Maximum output value of Y	REAL	-	0	-
MANUAL	Manual output	BOOL	[FALSE, TRUE]	FALSE	-
RESET	Reset	BOOL	[FALSE, TRUE]	FALSE	-

Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
Y	Output value	REAL	-	0	-
LIMITS_ACTIVE	Given limit reached	BOOL	[FALSE, TRUE]	FALSE	-

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	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	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

ACTUAL: Current value of the variable to control

SET_POINT: Set point of the variable to control

KP: Scaling coefficient, proportional gain of the P part

TV: Differential time, time of the D part in seconds. For example, 0.5 indicates 500 microseconds.

Y_MANUAL: When MANUAL is TRUE, the output value of Y is Y_MANUAL.

Y_OFFSET: Output offset of Y

Y_MIN: Minimum output value of Y

Y_MAX: Maximum output value of Y

MANUAL: Manual output control. If the value is TRUE, manual operation is activated and the output value of Y is Y_MANUAL.

RESET: When the value is TRUE, the controller is reset, and the value of Y is Y_OFFSET during re-initialization.

Y: Output value of the PD controller

LIMITS_ACTIVE: When the value is TRUE, the value of Y reaches the given limit (Y_MIN, Y_MAX).

■ Program example

This instruction implements incremental universal PD control.

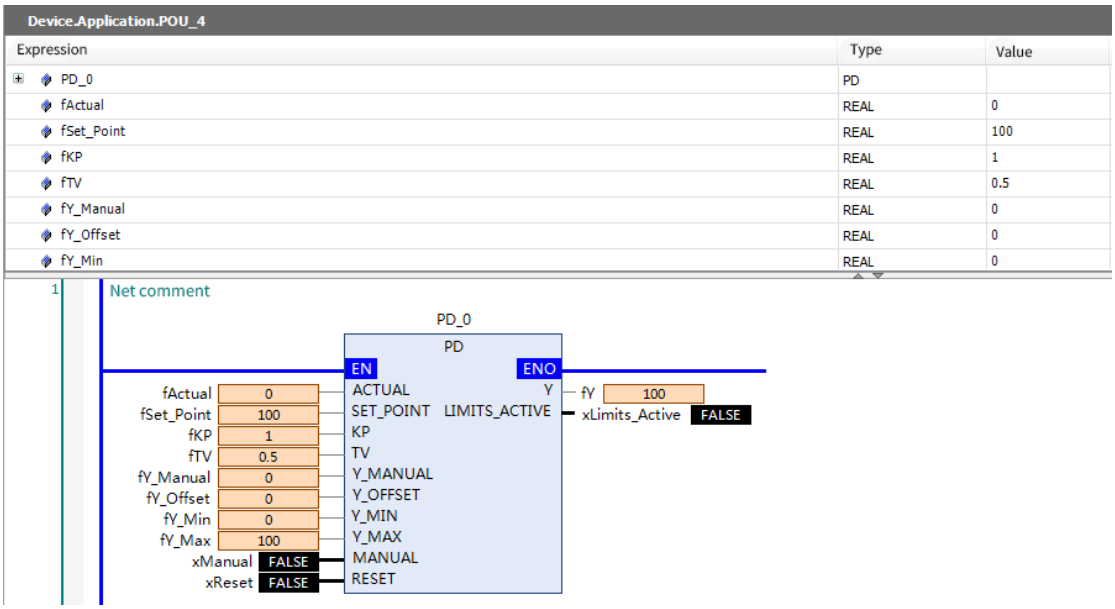
ST

Device.Application.POU_5

Expression	Type	Value
PD_0	PD	
fActual	REAL	0
fSet_Point	REAL	100
fKP	REAL	1
fTV	REAL	0.5
fY_Manual	REAL	0
fY_Offset	REAL	0
fY_Min	REAL	0
fY_Max	REAL	100
xManual	BOOL	FALSE
xReset	BOOL	FALSE
fY	REAL	100
xLimits_Active	BOOL	FALSE

```
1  PD_0(
2      ACTUAL:=fActual ,
3      SET_POINT:=fSet_Point ,
4      KP:=fKP ,
5      TV:= fTV ,
6      Y_MANUAL:=fY_Manual ,
7      Y_OFFSET:=fY_Offset ,
8      Y_MIN:=fY_Min ,
9      Y_MAX:=fY_Max ,
10     MANUAL:=xManual ,
11     RESET:=xReset ,
12     Y=>fY ,
13     LIMITS_ACTIVE=>xLimits_Active );RETURN
```

LD



■ Precautions

1) The output value of Y is calculated as follows:

$$Y = KP \cdot (\Delta + 1/TN \int e dt + TV\, \delta\Delta/\delta t) + Y\_OFFSET$$

2) If the output value is not limited, Y_MIN and Y_MAX must be set to 0.

3) Once MANUAL is set to TRUE, Y_MANUAL is written to Y.

4.7. 3 PID

This instruction implements universal PID control.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
PID	Universal PID control	FB	PID EN ENO ACTUAL Y SET_POINT LIMITS_ACTIVE KP OVERFLOW TN TV Y_MANUAL Y_OFFSET Y_MIN Y_MAX MANUAL RESET	PID_0(ACTUAL:= , SET_POINT:= , KP:= , TN:= , TV:= , Y_MANUAL:= , Y_OFFSET:= , Y_MIN:= , Y_MAX:= , MANUAL:= , RESET:= , Y=> , LIMITS_ACTIVE=> , OVERFLOW=>);

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
ACTUAL	Current value of variable to control	REAL	-	0	-
SET_POINT	Set point	REAL	-	0	-

Input Variable	Name	Data Type	Value Range	Initial Value	Description
KP	Scaling coefficient	REAL	-	0	-
TN	Integral time	REAL	-	0	-
TV	Differential time	REAL	-	0	-
Y_MANUAL	Manual output value	REAL	-	0	-
Y_OFFSET	Offset output of Y	REAL	-	0	-
Y_MIN	Minimum output value of Y	REAL	-	0	-
Y_MAX	Maximum output value of Y	REAL	-	FALSE	-
MANUAL	Manual output	BOOL	[FALSE, TRUE]	FALSE	-
RESET	Reset	BOOL	[FALSE, TRUE]	FALSE	-

Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
Y	Output value	REAL	-	0	-
LIMITS_ACTIVE	Given limit reached	BOOL	[FALSE, TRUE]	FALSE	-
OVERFLOW	Overflow flag	BOOL	[FALSE, TRUE]	FALSE	-

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	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	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

ACTUAL: Current value of the variable to control

SET_POINT: Set point of the variable to control

KP: Scaling coefficient, proportional gain of the P part

TN: Integral time, time of the I part in seconds. For example, 0.5 indicates 500 milliseconds. The value must be greater than 0. Otherwise, the function block does not perform any calculation.

TV: Differential time, time of the D part in seconds. For example, 0.5 indicates 500 milliseconds.

Y_MANUAL: When MANUAL is TRUE, the output value of Y is Y_MANUAL.

Y_OFFSET: Output offset of Y

Y_MIN: Minimum output value of Y

Y_MAX: Maximum output value of Y

MANUAL: Manual output control. If the value is TRUE, manual operation is activated and the output value of Y is Y_MANUAL.

RESET: When the value is TRUE, the controller is reset, and the value of Y is Y_OFFSET during re-initialization.

Y: Output value of the PD controller

LIMITS_ACTIVE: When the value is TRUE, the value of Y reaches the given limit (Y_MIN, Y_MAX).

OVERFLOW: Overflow flag. If the integral part of e is larger, incorrect parameter settings for the controller may cause overflow. For the sake of security, the output of OVERFLOW is TRUE in this case.

■ Program example

This instruction implements universal PID control.

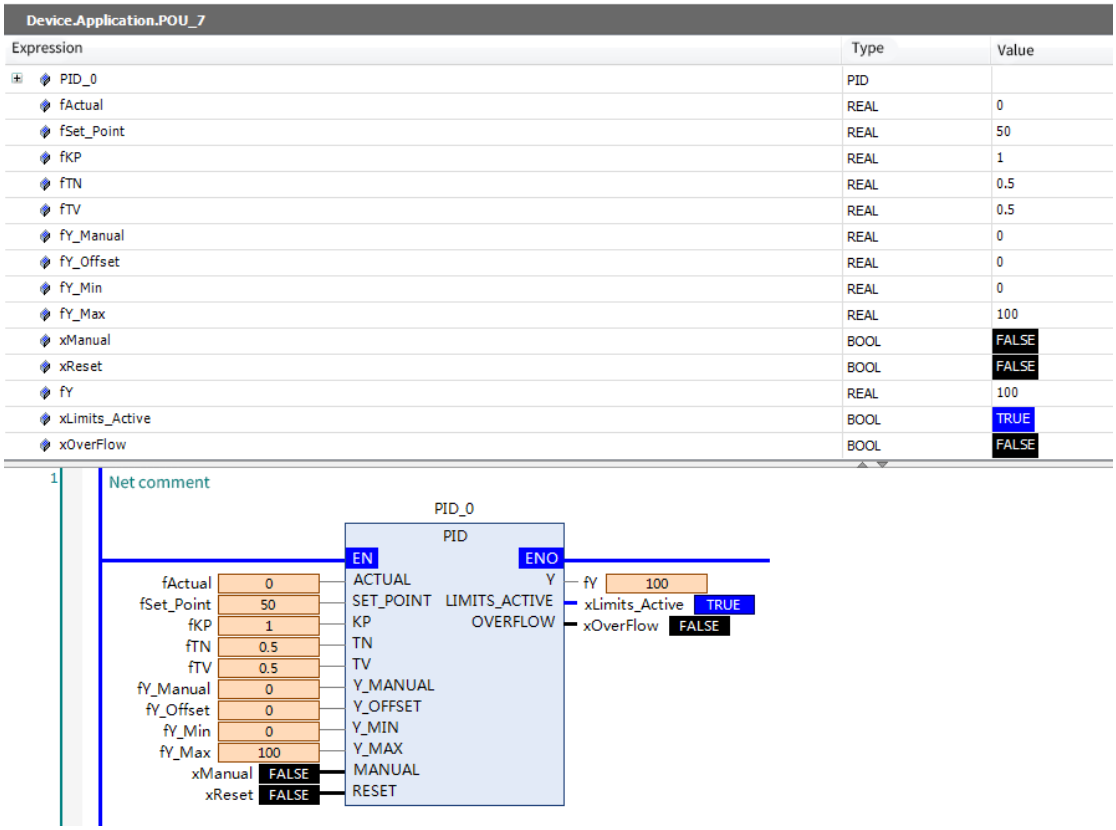
ST

Device.Application.POU_6

Expression	Type	Value
PID_0	PID	
fActual	REAL	0
fSet_Point	REAL	50
fKP	REAL	1
fTN	REAL	0.5
fTV	REAL	0.5
fY_Manual	REAL	0
fY_Offset	REAL	0
fY_Min	REAL	0
fY_Max	REAL	100
xManual	BOOL	FALSE
xReset	BOOL	FALSE
fY	REAL	100
xLimits_Active	BOOL	TRUE
xOverFlow	BOOL	FALSE

```
1  PID_0(
2      ACTUAL 0 :=fActual 0 ,
3      SET_POINT 50 :=fSet_Point 50 ,
4      KP 1 :=fKP 1 ,
5      TN 0.5 :=fTN 0.5 ,
6      TV 0.5 :=fTV 0.5 ,
7      Y_MANUAL 0 := fY_Manual 0 ,
8      Y_OFFSET 0 := fY_Offset 0 ,
9      Y_MIN 0 := fY_Min 0 ,
10     Y_MAX 100 := fY_Max 100 ,
11     MANUAL FALSE := xManual FALSE ,
12     RESET FALSE := xReset FALSE ,
13     Y 100 => fY 100 ,
14     LIMITS_ACTIVE TRUE => xLimits_Active TRUE ,
15     OVERFLOW FALSE => xOverFlow FALSE );
```

LD



■ Precautions

1) The output value of Y is calculated as follows:

$$Y = KP \cdot (e + 1/TN \int e dt + TV \cdot de/dt) + Y\_OFFSET$$

2) The control functions only when Y_MIN is less than Y_MAX.

3) Once MANUAL is set to TRUE, Y_MANUAL is written to Y.

4.7. 4 PID_FIXCYCLE

This instruction implements universal PID control that can manually set the cycle time.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
PID_FIXCY-CLE	Universal PID control that can manually set the cycle time	FB	PID_FIXCYCLE EN ENO ACTUAL Y SET_POINT LIMITS_ACTIVE KP OVERFLOW TN TV Y_MANUAL Y_OFFSET Y_MIN Y_MAX MANUAL RESET CYCLE	PID_FIXCYCLE_0(ACTUAL:= , SET_POINT:= , KP:= , TN:= , TV:= , Y_MANUAL:= , Y_OFFSET:= , Y_MIN:= ,Y_MAX:= , MANUAL:= , RESET:= , CYCLE:= , Y=> , LIMITS_ACTIVE=> , OVERFLOW=>);

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
ACTUAL	Current value of variable to control	REAL	-	0	-
SET_POINT	Set point	REAL	_	0	-
KP	Scaling coefficient	REAL	_	0	-
TN	Integral time	REAL	_	0	-
TV	Differential time	REAL	_	0	-
Y_MANUAL	Manual output value	REAL	_	0	-
Y_OFFSET	Offset output of Y	REAL	_	0	-
Y_MIN	Minimum output value of Y	REAL	_	0	-
Y_MAX	Maximum output value of Y	REAL	_	0	-
MANUAL	Manual output	BOOL	[FALSE, TRUE]	FALSE	-
RESET	Reset	BOOL	[FALSE, TRUE]	FALSE	-
CYCLE	Processing time	REAL	-	0	-

Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
Y	Output value	REAL	-	0	-
LIMITS_ACTIVE	Given limit reached	BOOL	[FALSE, TRUE]	FALSE	-
OVERFLOW	Overflow flag	BOOL	[FALSE, TRUE]	FALSE	-

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	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	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
MANUAL	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
RESET	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
CYCLE	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-
Y	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-
LIMITS_ACTIVE	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
OVERFLOW	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

ACTUAL: Current value of the variable to control

SET_POINT: Set point of the variable to control

KP: Scaling coefficient, proportional gain of the P part

TN: Integral time, time of the I part in seconds. For example, 0.5 indicates 500 milliseconds. The value must be greater than 0. Otherwise, the function block does not perform any calculation.

TV: Differential time, time of the D part in seconds. For example, 0.5 indicates 500 milliseconds.

Y_MANUAL: When MANUAL is TRUE, the output value of Y is Y_MANUAL.

Y_OFFSET: Output offset of Y

Y_MIN: Minimum output value of Y

Y_MAX: Maximum output value of Y

MANUAL: Manual output control. If the value is TRUE, manual operation is activated and the output value of Y is Y_MANUAL.

RESET: When the value is TRUE, the controller is reset, and the value of Y is Y_OFFSET during re-initialization.

CYCLE: Specified processing time

Y: Output value of the PD controller, in seconds. For example, 0.5 indicates 500 milliseconds.

LIMITS_ACTIVE: When the value is TRUE, the value of Y reaches the given limit (Y_MIN, Y_MAX).

OVERFLOW: Overflow flag. If the integral part of e is larger, incorrect parameter settings for the controller may cause overflow. For the sake of security, the output of OVERFLOW is TRUE in this case.

■ Program example

This instruction implements universal PID control that can manually set the cycle time.

ST

Device.Application.POU_8

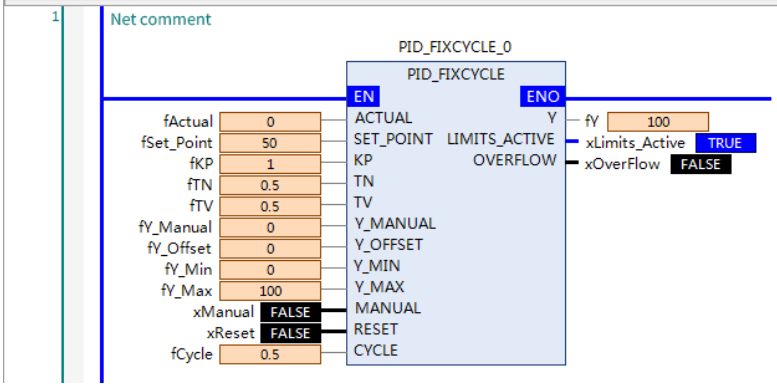
Expression	Type	Value
PID_FIXCYCLE_0	PID_FIXCYCLE	
fActual	REAL	0
fSet_Point	REAL	50
fKP	REAL	1
fTN	REAL	0.5
fTV	REAL	0.5
fY_Manual	REAL	0
fY_Offset	REAL	0
fY_Min	REAL	0
fY_Max	REAL	100
xManual	BOOL	FALSE
xReset	BOOL	FALSE
fCycle	REAL	0.5
fY	REAL	100
xLimits_Active	BOOL	TRUE
xOverFlow	BOOL	FALSE

```
1  PID_FIXCYCLE_0 (
2      ACTUAL 0 :=fActual 0 ,
3      SET_POINT 50 :=fSet_Point 50 ,
4      KP 1 := fKP 1 ,
5      TN 0.5 := fTN 0.5 ,
6      TV 0.5 := fTV 0.5 ,
7      Y_MANUAL 0 := fY_Manual 0 ,
8      Y_OFFSET 0 := fY_Offset 0 ,
9      Y_MIN 0 := fY_Min 0 ,
10     Y_MAX 100 :=fY_Max 100 ,
11     MANUAL FALSE := xManual FALSE ,
12     RESET FALSE := xReset FALSE ,
13     CYCLE 0.5 := fCycle 0.5 ,
14     Y 100 => fY 100 ,
15     LIMITS_ACTIVE TRUE => xLimits_Active TRUE ,
16     OVERFLOW FALSE =>xOverFlow FALSE ); RETURN
```

LD

Device.Application.POU_9

Expression	Type	Value
PID_FIXCYCLE_0	PID_FIXCYCLE	
fActual	REAL	0
fSet_Point	REAL	50
fKP	REAL	1
fTN	REAL	0.5
fTV	REAL	0.5
fY_Manual	REAL	0
fY_Offset	REAL	0
fY_Min	REAL	0
fY_Max	REAL	100
xManual	BOOL	FALSE
xReset	BOOL	FALSE
fCycle	REAL	0.5
fY	REAL	100
xLimits_Active	BOOL	TRUE
xOverFlow	BOOL	FALSE



■ Precautions

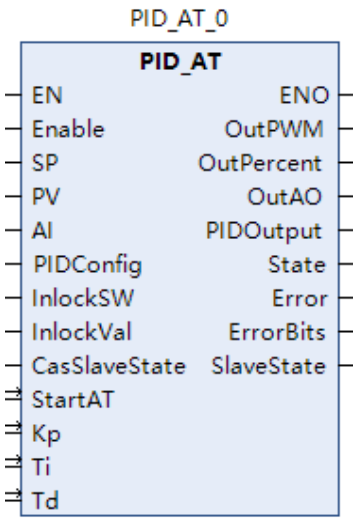
1) The control functions only when Y_MIN is less than Y_MAX.

2) Once MANUAL is set to TRUE, Y_MANUAL is written to Y.

4.7. 5 PID_AT

This instruction implements general PID control.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
PID_AT	Universal PID controller	FB		PID_AT_0(Enable:= Enable, SP:= , PV:= , AI:= , PIDConfig:= , InlockSW:= , InlockVal:= , CasSlaveState:= , Start AT:= Start AT, Kp:= Kp, Ti:= Ti. Td:= Td. OutPWM=> , OutPercent=> , OutAO=> , PIDOutput=> , State=> , Error=> , ErrorBits=> , SlaveState= >);

■ Variables

Input variables

Variable	Name	Data Type	Value Range	Initial Value	Description
Enable	Enable control	BOOL	[FALSE, TRUE]	FALSE	Enable PID control TRUE: Run FALSE: Stop
SP	Target reference	REAL	Depends on data types	0	Reference (target value)
PV	Measured value	REAL	Depends on data types	0	Current value (measured value)
AI	Analog input value	DINT	Depends on data types	0	Analog input value
PIDConfig	Parameter configuration structure	PID_ATC_Config	Depends on data types	-	Parameter configuration structure
InlockSW	Interlock switch	BOOL	[FALSE, TRUE]	FALSE	Interlock switch
InlockVal	Interlock security value	REAL	[–100, 100]	0.0	Interlock security value
CasSlaveState	Slave circuit status	Slave_Data	Depends on data types	-	Slave circuit status

Slave_Data structure

Structure	Element	Data Type	Initial Value	Description
Slave_Data	PV	REAL	0	Process value of slave circuit When the slave circuit is detected as non-cascade automatic mode, the output of the master circuit tracks the process value of the slave circuit.
	OVE	BYTE	0	Overlimit status of slave circuit 0: Normal 1: Upper limit reached 2: Lower limit reached
	CAS_State	UINT	0	Cascade use mode 0: Slave circuit non-automatic mode 1: Slave circuit automatic mode

In-out variables

Variable	Name	Data Type	Value Range	Initial Value	Description
StartAT	PID parameter auto-tuning enable	BOOL	[FALSE, TRUE]	FALSE	PID parameter auto-tuning enable. After auto-tuning is completed, internal auto reset occurs. TRUE: Enable auto-tuning FALSE: Disable auto-tuning
Kp	Scaling coefficient	REAL	[0.01, 1000]	60	Scaling coefficient
Ti	Integral time	REAL	[0, 10000]	400	Integral time (/s) A larger value causes less integral effect. When the value is 0, there is no integral action.
Td	Differential time	REAL	[0, 10000]	50	Differential time (/s) A larger value causes larger differential effect. When the value is 0, there is no differential action.

Output variables

Variable	Name	Data Type	Value Range	Initial Value	Description
OutPWM	PWM output	BOOL	[FALSE, TRUE]	FALSE	PWM output
OutPercent	Scaling coefficient	REAL	[0.01, 1000]	60	Scaling coefficient

Variable	Name	Data Type	Value Range	Initial Value	Description
OutAO	Integral time	REAL	[0, 10000]	400	Integral time (/s) A larger value causes less integral effect. When the value is 0, there is no integral action.
PIDOutput	Analog output value	PID_Output	[0, 10000]	50	Analog output value
State	Operation status of the analog output value	DINT	Depends on data types	0	Operation status of the analog output value: 0: Disable 1: Manual 2: Auto 3: Interlock (reserved) 4: Auto-tuning 5: Error value output 6: Alternative output for the incorrect security value 7: Master tracking (reserved)
Error	Running error	BOOL	[FALSE, TRUE]	FALSE	Running error
ErrorBits	Error code	DINT	For details, see error handling.	0	Error code. bit0: PV overlimit bit1: Parameter error
SlaveState	Slave circuit status	Slave_Data	Depends on data types	-	Slave circuit status

PID_Output structure

Structure	Element	Data Type	Initial Value	Description
PID_Output	PVOut	REAL	100	Actually calculated PV value
	ECMaxOut	REAL	1000	Fuzzy control, maximum temperature rise speed, in ° C/s
	ATState	DINT	0	Auto-tuning status. 0: No auto-tuning; 1: During auto-tuning; 2: Auto-tuning completed
	ErrorID	DINT	0	Parameter error code
	SPOut	REAL	FALSE	Actually calculated SP value

For details about parameter errors, see the output structure PID_Output.ErrorID.

The PIDConfig variable structure type is PID_ATC_Config.

Variable	Type	Description

GeneralConfig	PID_GeneralConfig	PID general configuration
InputConfig	PID_InputConfig	PID input configuration
OutputConfig	PID_OutputConfig	PID output configuration
SuperConfig	PID_SuperConfig	PID advanced configuration

The structure of the PID general configuration PID_GeneralConfig contains the following variables.

Variable	Name	Data Type	Value Range	Initial Value	Description
SampleTime	Heating sampling period	DINT	[1, 1000000]	100	Heating sampling period (ms)
CycleTime	Heating output period	DINT	[1, 1000000]	1000	Heating output period (ms)
PIDType	PID control type	DINT	0, 1, 2, and 3	0	0: PID 1: PI 2: PD 3: ID
ControlMode	Control mode	DINT	0 and 1	0	0: Reverse (unipolar heating) 1: Forward (unipolar cooling)
ManualEnable	Manual mode switch	BOOL	[FALSE, TRUE]	FALSE	Manual mode switch
ManualValue	Manual mode output	REAL	[OutMin, OutMax]	0	Manual mode output, with the value range following the maximum and minimum outputs (%)
ATCoefficient	Auto-tuning coefficient	REAL	[0, 1]	0.5	Auto-tuning coefficient
DelayStartUp	Delayed start time	DINT	[0, 1000000]	0	Delayed start time. PID is started after delay of the set time for off-peak start in case of multi-channel temperature control on the device. (ms)
Delay Shutdown	Delayed shutdown time	DINT	[0, 1000000]	0	Delayed shutdown time. PID is shut down after delay of the set time. (ms)
Physical Quantity	Physical object to control	DINT	0 and 1	1	Physical object to control 0: General 1: Temperature
PhysicalUnit	Physical object unit	DINT	Depends on data types	0	Unit of the physical object (reserved)
IsMaster	Master control	BOOL	[FALSE, TRUE]	FALSE	Whether the control is master control 0: No 1: Yes

IsSlave	Slave control	BOOL	[FALSE, TRUE]	FALSE	Whether the control is slave control 0: No 1: Yes

The structure of the PID input configuration PID_InputConfig contains the following variables.

Variable	Name	Data Type	Value Range	Initial Value	Description
PVType	PV input type selection	DINT	0 and 1	0	0: Use the current PV as the input 1: Use the peripheral analog as the input and convert it based on the analog range
AIMin	Minimum value for analog conversion	DINT	Depends on data types	0	Minimum value for analog conversion
AIMax	Maximum value for analog conversion	DINT	Depends on data types	20000	Maximum value for analog conversion
PVMin	Minimum value of PV	REAL	Depends on data types	0	Minimum value of PV
PVMax	Maximum value of PV	REAL	Depends on data types	100	Maximum value of PV
PVLowLimit	Lower alarm limit for PV	REAL	Depends on data types	–100	Lower alarm limit for PV
PVHighLimit	Higher alarm limit for PV	REAL	Depends on data types	1000	Higher alarm limit for PV
ErrorAction	Action for error alarm	BOOL	[FALSE, TRUE]	FALSE	Action for error alarm: FALSE: Remain the value upon an error alarm TRUE: Output the alternative value upon an error alarm
ErrorOut	Error output percentage	REAL	[OutMin, OutMax]	0	Output of an alternative percentage upon an error alarm
SPSel	Reference source	BOOL	[FALSE, TRUE]	FALSE	Reference source: FALSE: External reference (SP configuration) TRUE: Internal reference (panel operation reference)
SPRatio	SP reference change	REAL	Depends on data types	0	SP reference change
SPInt	Internal reference	REAL	Depends on data types	0	Internal reference
SPTrack	Internal reference tracking switch	BOOL	[FALSE, TRUE]	TRUE	Internal reference tracking switch The default value is TRUE.

The structure of the PID output configuration PID_OutputConfig contains the following variables.

Variable	Name	Data Type	Value Range	Initial Value	Description
OutType	Output type	DINT	0, 1, and 2	0	Output type: 0: PWM output 1: Analog output 2: Percent output
OutMin	Minimum value of output percentage	REAL	[–100, 100]	0	Minimum value of output percentage (%)
OutMax	Maximum value of output percentage	REAL	[–100, 100]	100	Maximum value of output percentage (%)
AOMin	Minimum value of analog output	DINT	Depends on data types	0	Minimum value of analog output
AOMax	Maximum value of analog output	DINT	Depends on data types	20000	Maximum value of analog output
OutputOffset	Output deviation compensation value	REAL	[–100, 100]	0	Output deviation compensation value (%)
MinPT	Minimum pulse value	DINT	[0, 10000]	0	Minimum pulse value (ms)

The structure of the PID advanced configuration PID_SuperConfig contains the following variables.

Variable	Name	Data Type	Value Range	Initial Value	Description
Fuzzy	Fuzzy self-adaption	BOOL	[FALSE, TRUE]	0	Heating sampling period (ms)
Integral Suspend	Integral pause	BOOL	[FALSE, TRUE]	0	Heating output period (ms)
Integral Division	Integral separation	BOOL	[FALSE, TRUE]	0	0: PID 1: PI 2: PD 3: ID
IntegralClear	Integral clear	BOOL	[FALSE, TRUE]	0	0: Normal integral 1: Clear integral
ECMax	Maximum temperature rise speed per second	REAL	Depends on data types	0	Maximum temperature rise speed per second
KdGain	Incomplete differential coefficient	REAL	[0, 1]	0	Incomplete differential coefficient (0.0 to 1.0) A larger value indicates smoother differential effect. When the value is 1, there is no differential action.
IntegralMax	Maximum integral value	REAL	[–100, 100]	75	Maximum integral value (%)
IntegralCut	Integral cutting coefficient	REAL	[0, 1.0]	0.75	Integral cutting coefficient for reference change
Integral DivLimit	Integral separation limit	REAL	[0, 100]	50	Integral separation limit

Variable	Name	Data Type	Value Range	Initial Value	Description
DeadBand	Deviation dead zone limit	REAL	[0, 100]	0	Deviation dead zone limit

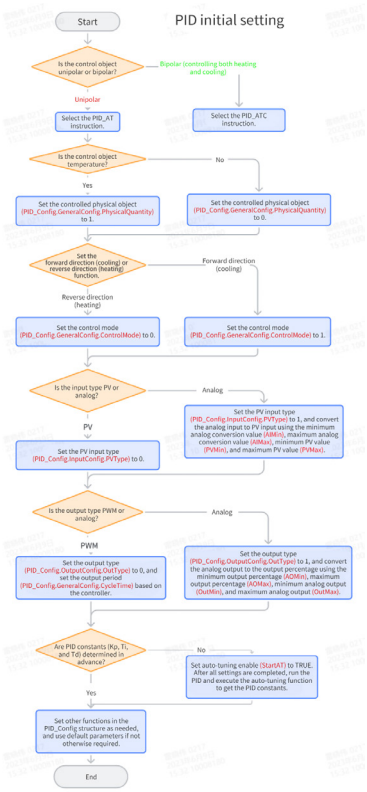
The structure of the PID output structure PID_Output contains the following variables.

Variable	Name	Data Type	Value Range	Initial Value	Description
PVOut	Analog PV value	REAL	Depends on data types	0	PV value converted from analog
ECMaxOut	Maximum temperature rise speed per second	REAL	Depends on data types	0	Maximum temperature rise speed per second
ATState	Auto-tuning status	DINT	0, 1, and 2	0	Auto-tuning status 0: No auto-tuning 1: During auto-tuning 2: Auto-tuning completed
ErrorID	Parameter error code	DINT	Depends on data types	0	Parameter error code
SPOut	Actual SP value	REAL	Depends on data types	0	Actually calculated SP value

The slave circuit status structure Slave_Data contains the following variables.

Variable	Name	Data Type	Value Range	Initial Value	Description
PV	Process value of slave circuit	REAL	Depends on data types	0	Process value of slave circuit When the slave circuit is detected as non-cascade automatic mode, the output of the master circuit tracks the process value of the slave circuit.
OVE	Overlimit status of slave circuit	BYTE	0, 1, and 2	0	Overlimit status of slave circuit 0: Normal 1: Upper limit reached 2: Lower limit reached
CAS_State	Cascade use mode	UINT	0 and 1	0	Cascade use mode 0: Slave non-automatic mode 1: Slave automatic mode In case of slave, the value should be returned to the master controller SUBCAS_MODE.

■ PID initial setting



■ Description

1. Enable

When the value of execution condition Enable changes to TRUE, PID calculation starts. When the value of Enable is TRUE, the following operations are periodically repeated:

Obtain the measured value PV → Perform PID calculation → Output the workload

After the value of Enable changes to FALSE, PID calculation is completed and the output is cleared.

2. StartAT (auto-tuning)

This instruction provides the auto-tuning function to automatically calculate the optimal values for PID constants Kp, Ti, and Td.

The auto-tuning execution conditions upon heating are as follows:

1) The current value is less than the reference. In general, the reference is the common temperature set for working.

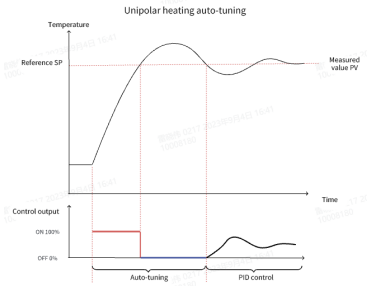
2) When the value of StartAT is TRUE, auto-tuning is executed from the beginning of operation if the value of Enable changes from FALSE to TRUE. When the value of Enable is TRUE and PID calculation is in progress, auto-tuning is executed during the operation if the value of StartAT changes from FALSE to TRUE.

After auto-tuning starts, the value of State.1 changes to 1. When auto-tuning is completed, the value of State.1 changes to 0 and the value of State.2 changes to 1.

In either case, when the value of StartAT changes from TRUE to FALSE during auto-tuning, auto-tuning is terminated, with State.1 of 0 and State.2 of 0.

Note: After auto-tuning is completed, StartAT is automatically set to FALSE.

The following figure shows the tuning process.



3. Scaling coefficient Kp

The scaling coefficient can be set by a user or automatically obtained through auto-tuning.

The value of Kp must be greater than 0. Otherwise, the value of Error is TRUE and eErrorID is an error number. A larger value indicates stronger scaling.

4. Integral time Ti

The integral time can be set by a user or automatically obtained through auto-tuning. If the value of Ti is 0, there is no integral action. A larger value indicates a weaker integral action.

The value of Ti cannot be less than 0. Otherwise, the value of Error is TRUE and ErrorID is an error number.

5. Differential time Td

The differential time can be set by a user or automatically obtained through auto-tuning. If the value of Td is 0, there is no differential action. A larger value indicates a stronger differential action.

The value of Td cannot be less than 0. Otherwise, the value of Error is TRUE and ErrorID is an error number.

6. InlockSW (interlock switch)

TRUE: When the interlock switch is triggered, the interlock security value is output.

7. InlockVal (interlock security value)

8. CasSlaveState (slave status structure Slave_Data)

· CasSlaveState.PV: Process value of slave circuit

When the slave circuit is detected as non-cascade automatic mode, the output of the master circuit tracks the process value of the slave circuit.

· CasSlaveState.OVE: Overlimit status of slave circuit

0: Normal

1: Upper limit reached

2: Lower limit reached

· CasSlaveState.CAS_State: Cascade use mode

0: Slave non-automatic mode

1: Slave automatic mode

In case of slave, the value should be returned to the master controller SUBCAS_MODE.

9. PID_Config (PID parameter configuration)

The options include the following:

GeneralConfig (general configuration), InputConfig (input configuration), OutputConfigHeat (heating output configuration), OutputConfigCool (cooling output configuration), and SupterConfig (advanced configuration)

10. Parameter type PID_GeneralConfig for general configuration GeneralConfig:

· GeneralConfig.SampleTime: Sampling period

The current PV value is obtained once a sampling period for once PID calculation. In general, the sampling period is that of the temperature collection module. For general requirements, the default value 100 ms can be used.

· GeneralConfig.CycleTime: Output period

Output period of HeatPWM. If the HeatPercent output value of the PID is 80% and the value of GeneralConfig.CycleTime is 10000 milliseconds, the period for HeatPWM is 10000 milliseconds, with 8s on and 2s off.

· GeneralConfig.PIDType: Current PID type

0: PID calculation

1: PI calculation

2: PD calculation

3: ID calculation

Perform corresponding calculation control based on the selected PID type. The default value is 0.

· GeneralConfig.ControlMode: Control mode

0: Reverse (unipolar heating)

1: Forward (unipolar cooling)

2: Bipolar control based on cooling coefficient (available for PID_ATC)

≫ The refrigerating parameter is calculated based on the PID parameters and cooling coefficient of the heating process.

≫ It is applicable when the time response is similar but gains are different for the heating activator and cooling activator.

3: Bipolar control based on independent PID parameter
Heating and cooling are separately controlled based on the configured parameter (available for PID_ATC).

≫ Independent cooling PID parameters are used.

≫ It is applicable when the time response is similar but gains are different for the heating activator and cooling activator.

· GeneralConfig.ManualEnable: Manual mode switch

0: Automatic mode

1: Manual mode

When the manual mode is enabled, the value of GeneralConfig.ManualValue is output.

In bipolar mode, the positive value of GeneralConfig.ManualValue corresponds to the heating output value, and the negative value corresponds to the cooling output value.

· GeneralConfig.ATCoefficient: Auto-tuning coefficient

For the auto-tuning coefficient, the larger the Kp parameter value obtained from auto-tuning, the faster the set temperature can be reached, but it may cause greater overshoot. The smaller the auto-tuning

coefficient, the slower the set temperature parameter obtained from auto-tuning can be reached, and the temperature curve will be very smooth, without overshoot or with very small overshoot. Use the default value 0.5 for the parameter of initial tuning. If the customer requires to reach the set temperature faster, increase the auto-tuning coefficient GeneralConfig.ATCoefficient for another tuning based on the control effect.

If the customer needs a smooth curve and small overshoot, decrease the auto-tuning coefficient GeneralConfig.ATCoefficient for another tuning.

· GeneralConfig.DelayStartUp: Delayed start time

PID is started after delay of different set time for off-peak start in case of multi-channel temperature control on the device. This prevents a large current fluctuation during multi-channel start.

· GeneralConfig.Delayshutdown: Delayed shutdown time

Delayed shutdown time. PID is shut down after delay of the set time.

· GeneralConfig.PhysicalQuantity: Physical object to control

0: General

1: Temperature

· GeneralConfig.PhysicalUnit: Unit of the physical object

· GeneralConfig.IsMaster: Whether the control is master control

0: No

1: Yes

· GeneralConfig.IsSlave: Whether the control is slave control

0: No

1: Yes

11. Parameter type PID_InputConfig for input configuration InputConfig:

· InputConfig.PVType: Type of sampling PV

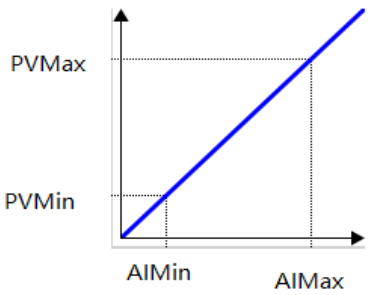
0: Use the current PV as the input

1: Use the peripheral analog as the input and convert it based on the analog range InputConfig.AIMin and InputConfig.AIMax as well as the corresponding PVMin and PVMax

· InputConfig.AIMin: Minimum value for analog conversion
InputConfig.AIMax: Maximum value for analog conversion

They are used to convert an analog to a PV value.

Use the peripheral analog as the input and convert it based on the analog range InputConfig.AIMin and InputConfig.AIMax as well as the corresponding PVMin and PVMax



· InputConfig.PVMin and InputConfig.PVMax

InputConfig.PVMin: Minimum PV value corresponding to InputConfig.AIMin

InputConfig.PVMax: Minimum PV value corresponding to InputConfig.AIMax

Use the peripheral analog as the input and convert it based on the analog range InputConfig.AIMin and InputConfig.AIMax as well as the corresponding PVMin and PVMax

· InputConfig.InputMin and InputConfig.InputMax: Input upper limit and lower limit - alarm

When the PV value or the PV value converted from the analog is lower than the value of InputConfig. InputMin, an alarm is generated for exceeding the lower limit.

When the PV value or the PV value converted from the analog is higher than the value of InputConfig. InputMax, an alarm is generated for exceeding the upper limit.

Alarms are output based on InputConfig.ErrorAction and ErrorOut.

· InputConfig.ErrorAction: Action for error alarm
InputConfig.ErrorOut: Output of an alternative value upon an error alarm

FALSE: Remain the output value upon an error alarm

TRUE: Output the alternative value InputConfig.ErrorOut upon an error alarm

1: Use the peripheral analog as the input and convert it based on the analog range InputConfig.AIMin and InputConfig.AIMax as well as the corresponding PVMin and PVMax

· InputConfig.SPSel: Reference source

FALSE: External reference (SP configuration)

TRUE: Internal reference (panel operation reference SPInt)

· InputConfig.SPRatio: SP reference change

If SPRatio is greater than 0, the final value turns closer to the changing reference based on the change of SPRatio. It is calculated once every task period.

· InputConfig.SPInt: SP internal reference

If SPSel is TRUE, select the internal reference of SPInt as the current reference (panel operation reference SPInt)

· InputConfig.SPTrack: Internal reference tracking switch
The default value is TRUE.

FALSE: The internal reference does not track the external reference.

TRUE: The internal reference tracks the external reference.

12. Parameter type PID_OutputConfig for output configuration OutputConfig:

· OutputConfig.OutType: Output type

0: PWM output
OutPercent indicates the PWM output percentage.

1: Analog output
OutPercent indicates the analog output percentage.

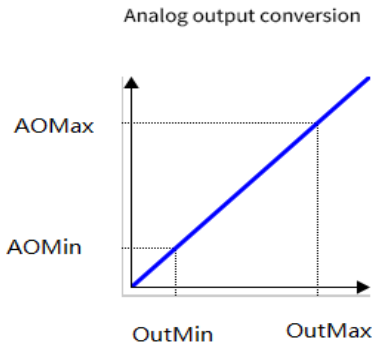
2: Percent output
OutPercent indicates the percent output.

· OutputConfig.OutMin and OutputConfig.OutMax: minimum output percentage and maximum output percentage

It is used to limit the output of PID calculation and convert the analog output.

Convert the output percentage of PID calculation based on the analog range OutputConfig.OutMin and

OutputConfig.OutMax as well as the corresponding OutputConfig.AOMin and OutputConfig.AOMax.



· OutputConfig.AOMin and OutputConfig.AOMax: minimum analog output and maximum analog output

It is used to convert the analog output.

Convert the output percentage of PID calculation based on the analog range OutputConfig.OutMin and OutputConfig.OutMax as well as the corresponding OutputConfig.AOMin and OutputConfig.AOMax.

· OutputConfig.OutputOffset: Output deviation compensation value

OUT = PID calculation value + OutputOffset: The final output is equal to the PID calculation value plus the value of OutputOffset. It can be used for the feedforward compensation.

· OutputConfig.MinPT: Minimum pulse enable/disable time of PWM, in milliseconds

The pulse is not enabled when the enable time is less than MinPT.

The pulse is not disabled when the disable time is less than MinPT.

13. Parameter type PID_SuperConfig for advanced configuration SuperConfig:

· SuperConfig.Fuzzy: Fuzzy self-adaption enable

0: Fuzzy self-adaption disabled

1: Fuzzy self-adaption enabled

PID parameters can be dynamically adjusted based on fuzzy self-adaption (available for unipolar heating).

· SuperConfig.ECMax: Maximum temperature rise speed per second

It is used in fuzzy self-adaptive mode. The maximum temperature rise speed of the device can be input by the user or obtained through auto-tuning (PIDOutput.ECMaxOut).

· SuperConfig.KdGain: Incomplete differential coefficient

The value range is 0.0 to 1.0. A larger value causes smoother differential effect.

When the value is 1, there is no differential action.

· SuperConfig.IntegralSuspend: Integral pause

0: Integral enabled (default)

1: Integral paused

· SuperConfig.IntegralDivision: Integral separation

0: Integral separated (default)

1: Integral not separated

· SuperConfig.IntegralClear: Integral clear

0: Normal integral (default)

1: Clear integral

· SuperConfig.IntegralMax: Maximum integral value

The maximum integral value is 75% by default.

· SuperConfig.IntegralCut: Integral cutting coefficient for reference change

The integral cutting coefficient is 0.75 by default.

· SuperConfig.IntegralDivLimit: Integral separation limit

When the deviation is greater than IntegralDivLimit, no integration is performed.

· SuperConfig.DeadBand: Deviation dead zone limit

When deviation |SP – PV| is less than DeadBand, the deviation is considered as 0 (no PID calculation).

14. Output structure:

· PIDOutput.PVOut: Actually calculated PV value

· PIDOutput.ECMaxOut: Maximum temperature rise speed per second

The output is auto-tuning output, which is used for fuzzy control.

· PIDOutput.ATState: Auto-tuning status

0: No auto-tuning

1: During auto-tuning

2: Auto-tuning completed

· PIDOutput.ErrorID: Parameter error code

For details, see the error codes.

· PIDOutput.SPOut: Actually calculated SP value

■ Function

A single circuit, also known as a simple control system, refers to a closed circuit feedback control system composed of a controlled object, a detection transmitter, a controller, and an activator. It is often used when the hysteresis time of the controlled object is small, the load and interference change little, and the control quality requirements are not very high.

≫ Calculation formula

$$\mathbf{AV} = \boldsymbol{K_p} * \left[(\boldsymbol{SP} - \boldsymbol{PV}) + \frac{\mathbf{1}}{\boldsymbol{T_i} * \boldsymbol{S}} (\boldsymbol{SP} - \boldsymbol{PV}) + \frac{T_d}{\boldsymbol{S}} (\boldsymbol{PV}) \right]$$

Discretization formula

$$\mathbf{AV(k)} = \boldsymbol{K_p} * \mathbf{E}(\boldsymbol{k}) + \boldsymbol{K_p} * \frac{\boldsymbol{cycle}}{\boldsymbol{T_i}} \sum \mathbf{E}(\boldsymbol{k}) + \frac{\boldsymbol{K_p} * \boldsymbol{T_D}}{\boldsymbol{cycle}} (\boldsymbol{P}(\boldsymbol{k}) - \mathbf{P}(\boldsymbol{k} - \mathbf{1}))$$

≫ Working modes

Working mode	Action
Automatic	The output value of the PID module is calculated based on the PID rule. The calculation result of the PID module is restricted by the output amplitude.

Manual	The output is manually controlled (operator write value or online logical write value). The output of this mode is restricted by the output amplitude.
Interlock	In the working mode, PID stops automatic calculation, and the output value of the PID module is a pre-set value of InlockVal, which is generally referred to as the interlock point. The interlock functions are also used for tracking. When a certain condition (interlock switch InlockSW is TRUE) is met, the PID module automatically switches to the working mode. When the interlock condition is not met, the PID module switches back to the previous working mode.

1) Interlock has the highest priority, followed by manual, and finally automatic.

2) Automatic mode: For much control with fixed set-points, references can be configured on the interface.

3) Choose to use the external reference or internal reference based on the configuration of PIDConfig. InputConfig.SPSel.

SPSel = FALSE: External reference of the logical configuration

SPSel = TRUE: Internally given reference

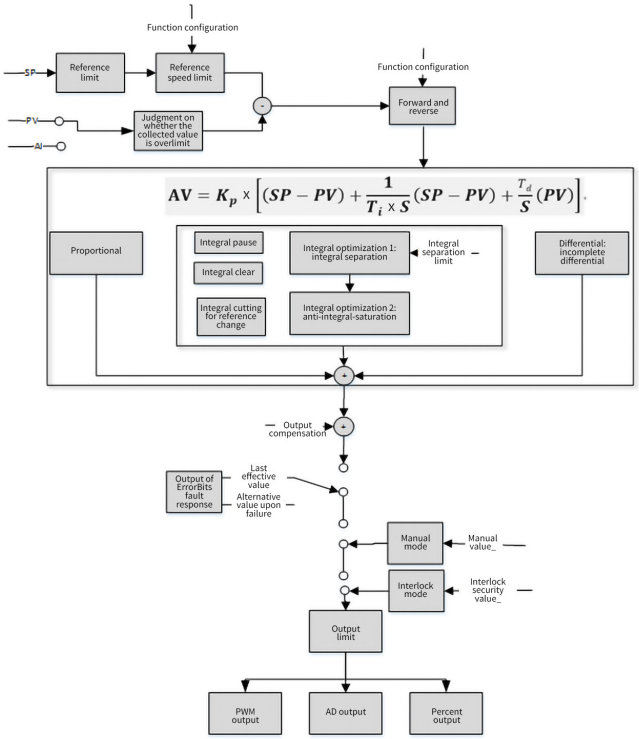
Note: To prevent circuit fluctuations that may occur during the switchover process of internal and external references, PIDConfig.InputConfig.SPTrack (whether the internal reference follows the external reference) can be set to TRUE. With the external reference selected, the internal reference can be interlocked with the external reference to ensure the unperturbed switchover of the internal reference.

If real-time values of the logical configuration are directly used during the switchover from the internal reference to the external reference, perturbation may occur.

≫ Unperturbed switchover

Interlock to manual	In interlock mode, the interlock value is directly output. In this case, the manual value tracks the output value to ensure the unperturbed switchover.
Interlock to automatic	In interlock mode, the interlock value is directly output. After the interlock mode is switched to the automatic mode, if the external reference is used, the logic operation value is directly used for the PID calculation output, which may cause perturbation.
Manual to interlock	The manual output is the manual value. After the manual mode is switched to the interlock mode, the output is the interlock value, and perturbation occurs.
Manual to automatic	The manual output is the operator reference. If the external reference is used, the logic operation value is directly used for the PID calculation output. In this case, the switchover is unperturbed (the incremental part is perturbed).
Auto to interlock	In automatic mode, the output is the normal operation value. After the automatic mode is switched to the interlock mode, the output is the interlock value, and perturbation occurs.
Automatic to manual	In automatic mode, the output is the normal operation value. In this case, the manual value tracks the output value to ensure the unperturbed switchover.

≫ Logic diagram of working (taking reverse as an example)



≫ Forward and reverse supported

Forward and reverse: In automatic mode, when the output change direction is consistent with the feedback value, it is considered as forward; when the output change direction is opposite to the feedback value, it is considered as reverse. In simple terms, if the output needs to be increased to eliminate the deviation when the feedback value is greater than the reference, it is considered as forward. Conversely, if the output needs to be decreased, it is considered as reverse.

≫ Reference change limit function

In automatic mode, the reference (including external and internal references) change is checked. If the change exceeds a preset limit (specified by PIDConfig.InputConfig.SPRatio), the actual reference used in the calculation is restricted to the change value and the output will gradually approach the reference based on the magnitude of the change.

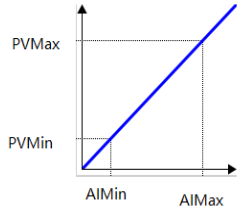
$$\Delta SP > SPRatio \text{ (Reference change): } SP = LastSP + SPRatio$$

$$\Delta SP < SPRatio \text{ (Reference change): } SP = LastSP - SPRatio$$

≫ Input signal selection function

Input signals of two modes can be selected.

PIDConfig.InputConfig.PVType	Input	-
0	Current input (PV)	-

1	Analog input (AI)	Linear conversion is calibrated based on the upper limits and lower limits of the analog and PV value.  PIDConfig.InputConfig.AIMax: Upper limit of analog PIDConfig.InputConfig.AIMin: Lower limit of analog PIDConfig.InputConfig.AIMax: Upper limit of PV PIDConfig.InputConfig.AIMin: Lower limit of PV

≫ Integral function

· Integral separation

Integral saturation is mainly caused by the integral term. To overcome integral saturation, it is crucial to limit the integral action to prevent excessive integral accumulation. To eliminate integral saturation, the integral separation function can be employed to provide the anti-integral-saturation capability to the function block.

For non-temperature objects, the configuration for integral separation is as follows:

Integral separation enable PIDConfig.GeneralConfig.IntegralDivision	Integral separation limit PIDConfig.GeneralConfig.IntegralDivLimit
TRUE	When |Deviation| is greater than or equal to the integral separation limit, no integral component operation is performed. In this case, the control is PD control.
	When |Deviation| is less than the integral separation limit, the full integral operation is performed. In this case, the control is PID control.
FALSE	The full integral operation is always performed.

· Integral pause

Set PIDConfig.GeneralConfig.IntegralSuspend to TRUE to trigger integral pause. The integral action remains unchanged and no more integral accumulation occurs.

· Integral clear

Set PIDConfig.GeneralConfig.IntegralClear to TRUE to trigger integral clear. The integral action is cleared to 0.

· Integral cutting coefficient

When the reference changes, set PIDConfig.SuperConfig.IntegralCut to reduce the initial integral action. The value range is from 0 to 1. The default value is 0.75.

≫ Incomplete differential function

The incomplete differential function adds a first-order inertial element to the differential action, which helps to smooth the differential effect and effectively improves the control performance of the system.

Parameter: PIDConfig.SuperConfig.KdGain	Differential action
0.0	The differential action takes effect only within this adjustment period. The default value is 0.0.

(0.0, 1.0)	A larger value indicates a smoother differential action.
1.0	The differential action does not take effect.

≫ Dead zone control function

The PIDConfig.SuperConfig.DeadBand parameter (specifying the dead zone width) can be set. Within the dead zone range, the adjustment is stable and the PID output does not change. If the value exceeds the dead zone range, PID calculation is normal.

≫ Delayed start/stop function

This function is used to set different delayed start/stop time for off-peak start/stop in case of multi-circuit control for the device. This prevents a large current fluctuation during multi-channel start.

PIDConfig.GeneralConfig.DelayStartUp	After the time specified by DelayStartUp, restart PID calculation.
PIDConfig.GeneralConfig.Delayshutdown	After the time specified by Delayshutdown, stop PID calculation.

≫ Output compensation function

The output compensation function directly adds the output compensation value to the result of automatic PID calculation to obtain the output value. When the controller is in automatic mode, the output compensation functions. When the controller is in manual mode, the output compensation does not function.

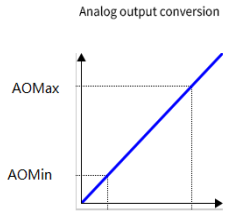
≫ Output amplitude limit function

When the output value is greater than the output upper limit specified by PIDConfig.OutputConfig.OutMax, the output value is equal to the output upper limit.

When the output value is less than the output lower limit specified by PIDConfig.OutputConfig.OutMin, the output value is equal to the output lower limit.

≫ Output signal selection function

Three modes are supported for output control:

PIDConfig.OutputConfig.OutType	Output	-
0	PWM output	The digital is converted and output based on the PWM output period.
1	Analog output	Linear conversion is calibrated and output based on the upper limits and lower limits of the analog and PID output.  PIDConfig.OutputConfig.AOMax: Upper limit of analog output PIDConfig.OutputConfig.AOMin: Lower limit of analog output PIDConfig.OutputConfig.OutMax: Upper limit of PID output PIDConfig.OutputConfig.OutMin: Lower limit of PID output
2	Percent output	Output within 0% to 100%

≫ Fault detection

When any bit of ErrorBits is triggered, Error is TRUE.

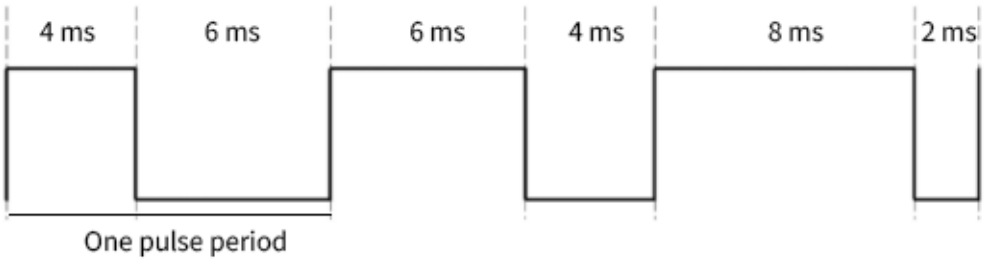
Bit of ErrorBits		Error output configuration parameter: PIDConfig.InputConfig.ErrorAction
Bit0 = 1	PV overlimit alarm	FALSE: The output remains
Bit1 = 1	parameter configuration error alarm	TRUE: The alternative value PIDConfig.InputConfig.ErrorOut is output

For details about parameter configuration errors, see ErrorID.

≫ PWM output

For PWM, the output is ON or OFF. The analog of PID output needs to be converted to digital output (ON/OFF) based on the duty cycle. The duty cycle is the percentage of the high-level pulse time to a whole pulse period.

For example, if the opening of a PID regulator's output is 40% and the PWM output period PIDConfig.GeneralConfig.CycleTime is set to 10s, the ON time for the 10s period would be 10s × 40% = 4s, and the OFF time would be 10s – 4s = 6s. The duty cycle of 40%. Therefore, within the 10s period, the output is the ON signal for 4s, followed by the OFF signal for 6s, and then the next period calculation begins.

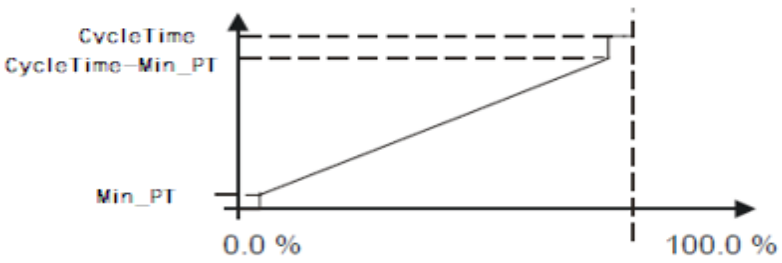


· Minimum pulse on/off time

To prevent frequent operation on the actuator, PIDConfig.OutputConfig.Min_PT can be set to specify the minimum pulse on/off time.

When the calculated high-level pulse time is less than the minimum pulse on/off time, the output duty cycle is 0%.

When the calculated high-level pulse time is greater than the minimum pulse on/off time in the PWM output period, the output duty cycle is 100%.



≫ Cascade function

Cascade control involves two or more cascaded PID controllers. The control variable of the main controller is interconnected with the reference of the slave controller, creating a nested control circuit. Cascade control in a slave circuit can compensate for interference in the internal circuit faster than in the slower main circuit.

Note: Cascade control is possible only when there are other measurable variables in the process, and the internal control circuit is much faster than the external circuit.

Unperturbed switchover: When the slave circuit is not in cascade mode (automatic mode of the external reference), the master circuit switches to the tracking mode. The output value of the master circuit will tracks the process value of the slave circuit, allowing for a smooth transition back to the cascade mode.

Priority (Error response > Interlock > Manual > Master tracking mode triggered by slave > Automatic)

■ Description

1. First, select the physical object to control GeneralConfig.PhysicalQuantity. The options include General and Temperature.

0: General, which is mainly for pressure and flow control.
During general control, IntegralDivision and IntegralDivLimit in advanced configuration can be set for integral separation.

1: Temperature, for which various optimization is carried out.

2. Enable the auto-tuning function at room temperature. When auto-tuning is completed, xStartAT is automatically set to FALSE, and you can obtain the set Kp, Ti and Td values. Let the program continue to run and check whether the controlled object can reach steady state.

3. If the current usage needs are met after the controlled object reaches the steady state, record the values of Kp, Ti, and Td, and write these values into the program as defaults.

You can manually adjust the Kp, Ti, and Td values to meet your usage needs.

4. The larger the auto-tuning coefficient rATCoefficient, the faster the set temperature can be reached, but it may cause greater overshoot. The smaller the auto-tuning coefficient, the slower the set temperature can be reached, and the temperature curve will be very smooth, without overshoot or with very small overshoot. Unless otherwise required, use the default 0.5. If the customer requires to reach the set temperature faster, increase the auto-tuning coefficient rATCoefficient for another tuning.

If the customer needs a smooth curve and small overshoot, decrease the auto-tuning coefficient rATCoefficient for another tuning.

5. With the interlock function, when the InlockSW interlock switch is triggered, the interlock security value InlockVal is output.

6. The output priority is error output > interlock output > manual output > automatic output.

Note:

1. Before running the PID, check whether the parameters are correct.

2. It is recommended that the auto-tuning process starts at the room temperature.

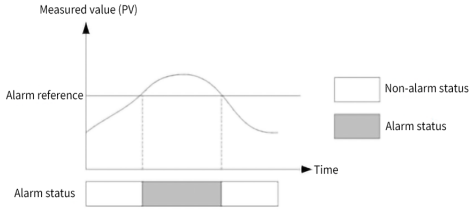
3. If the common temperature range for the customer is 150° C to 200° C, tuning can be carried out at 175° C. It is important to select a common temperature for more accurate tuning parameter.

4. In general, auto-tuning can be carried out first to obtain the tuned PID parameters. Then, you can adjust the PID parameters as needed based on the control effect.

■ Upper and lower limit alarms

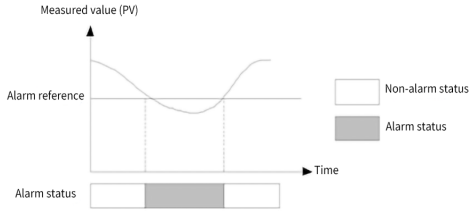
1. Upper limit alarm of the input value PV

When the measured value PV (of the PV value converted from the analog) is greater than the upper limit, an alarm is reported.



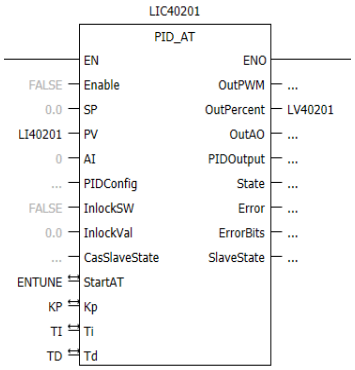
2. Lower limit alarm of the input value PV

When the measured value PV (of the PV value converted from the analog) is less than the lower limit, an alarm is reported.



■ Application examples

Control scheme configuration



Examples

No.	Example	Description
1	LIC40201	Liquid level controller
2	LI40201	Liquid level AI detection signal
3	LV40201	Liquid level control valve

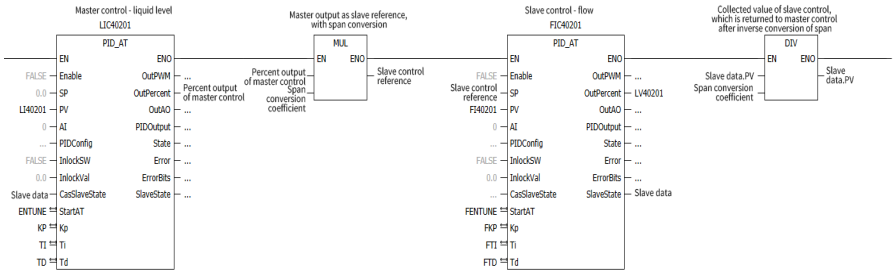
Cascade control scheme configuration:

LIC40201.PIDConfig.GeneralConfig.IsMaster = TRUE; // The control for the LIC40201 liquid level is master control.

LIC40201.PIDConfig.GeneralConfig.PhysicalQuantity = liquid level; // The LIC40201 object to control is the liquid level.

FIC40201.PIDConfig.GeneralConfig. IsSlave = TRUE; // The control for the FIC40201 liquid level is slave control.

FIC40201.PIDConfig.GeneralConfig.PhysicalQuantity = flow; // The LIC40201 object to control is the flow.



Examples

No.	Example	Description
1	LIC40201	Liquid level controller
2	LI40201	Liquid level AI detection signal
3	FIC40201	Flow controller
4	LV40201	Flow control valve
5	FI40201	Flow AI detection signal

■ Error codes (ErrorID)

Group	Error No.	Error code	Description
No error	0	NO_ERROR	No error
Parameter errors in the input interface	1	KP_ERROR	Kp configuration error
	2	TI_ERROR	Ti configuration error
	3	TD_ERROR	Td configuration error
	4	KPCOOL_ERROR	Configuration error of Kp for cooling
	5	TICOOL_ERROR	Configuration error of Ti for cooling
	6	TDCOOL_ERROR	Configuration error of Td for cooling
	7	COOL_COEFFICIENT_ERROR	Configuration error of cooling coefficients
General configuration	8	SAMPLE_TIME_ERROR	Sampling period - General configuration error
	9	CYCLE_TIME_ERROR	Output period - General configuration error
	10	PID_TYPE_ERROR	PID type - General configuration error
	11	MODE_ERROR	PID control mode - General configuration error
	12	MANUAL_VALUE_ERROR	Manual output value - General configuration error
	13	ATCOEFFICIENT_ERROR	Auto-tuning coefficient - General configuration error
	14	DELAY_STARTUP_ERROR	Delayed startup - General configuration error
	15	DELAY_SHUTDOWN_ERROR	Delayed shutdown - General configuration error

Group	Error No.	Error code	Description
Input configuration	16	PVTYPE_ERROR	PV type - Input configuration error
	17	AI_MIN_ERROR	Minimum value for analog conversion - Input configuration error
	18	AI_MAX_ERROR	Maximum value for analog conversion - Input configuration error
	19	PV_MIN_ERROR	Minimum value of PV - Input configuration error
	20	PV_MAX_ERROR	Maximum value of PV - Input configuration error
	21	PV_LOW_LIMIT_ERROR	Input lower limit - Input configuration error
	22	PV_HIGH_LIMIT_ERROR	Input upper limit - Input configuration error
	23	ERROR_OUT_ERROR	Output of an alternative value upon an error alarm - Input configuration error
	24	PV_UNIT_ERROR	Configuration error of PV unit
Heating output configuration	25	OUT_TYPE_ERROR	Heating output type - Output configuration error
	26	OUT_MIN_ERROR	Minimum value of heating output percentage - Output configuration error
	27	OUT_MAX_ERROR	Maximum value of heating output percentage - Output configuration error
	28	AO_MIN_ERROR	Minimum value of heating analog output - Output configuration error
	29	AO_MAX_ERROR	Maximum value of heating analog output - Output configuration error
	30	OUTPUT_OFFSET_ERROR	Heating output compensation - Output configuration error
Cooling output configuration (available for PID_ATC)	31	COOL_OUT_TYPE_ERROR	Cooling output type - Cooling output configuration error
	32	COOL_OUT_MIN_ERROR	Minimum value of cooling output percentage - Cooling output configuration error
	33	COOL_OUT_MAX_ERROR	Maximum value of cooling output percentage - Cooling output configuration error
	34	COOL_AO_MIN_ERROR	Minimum value of cooling analog output - Cooling output configuration error
	35	COOL_AO_MAX_ERROR	Maximum value of cooling analog output - Cooling output configuration error
	36	COOL_OUTPUT_OFFSET_ERROR	Cooling output compensation - Cooling output configuration error
Advanced configuration	37	ECMAX_ERROR	Maximum temperature rise speed - Advanced configuration error
	38	KD_GAIN_ERROR	Incomplete differential coefficient - Advanced configuration error
	39	INTEGRAL_MAX_ERROR	Maximum integral value - Advanced configuration error
	40	INTEGRAL_CUT_ERROR	Integral cutting coefficient for reference change - Advanced configuration error

Group	Error No.	Error code	Description
Cooling configuration (available for PID_ATC)	41	COOL_PID_TYPE_ERROR	Cooling PID type - Cooling configuration error
	42	SAMPLE_COOL_TIME_ERROR	Cooling sampling period - Cooling configuration error
	43	CYCLE_COOL_TIME_ERROR	Cooling output period - Cooling configuration error
	44	COOL_OFFSET_ERROR	Cooling reference deviation - Cooling configuration error
PV limit alarm	45	PV_LOW_LIMIT_ALARM	Lower limit alarm of PV. Check the PIDConfig.InputConfig.PVLowLimit and PIDConfig.InputConfig.PVHighLimit parameters to ensure that the PV value is in the value range.
	46	PV_HIGH_ LIMIT_ALARM	Upper limit alarm of PV. Check the PIDConfig.InputConfig.PVLowLimit and PIDConfig.InputConfig.PVHighLimit parameters to ensure that the PV value is in the value range.
Input interface configuration error	47	PV_AND_AI_NOINPUT	No input configuration for PV and AI
Input configuration error	48	SP_RAT_ERROR	Reference change configuration error - Input configuration error
Advanced configuration error	49	INTEGRAL_DIV_ERROR	Integral separation limit configuration error - Advanced configuration error
Advanced configuration error	50	DEADBAND_ERROR	Control dead zone configuration error - Advanced configuration error
Cooling output configuration error (available for PID_ATC)	51	COOL_MINPT_ERROR	Minimum cooling on/off time configuration error - Cooling output configuration error
Output configuration error	52	OUT_MINPT_ERROR	Minimum on/off time configuration error - Output configuration error
Parameter errors in the input interface	53	INLOCKVAL_ERR	Interlock security value configuration error
Parameter errors in the input interface	54	CASSLAVESTATE_PV_ERR	PV overlimit in slave data
Parameter errors in the input interface	55	CASSLAVESTATE_OVE_ERR	OVE configuration error in slave data
Parameter errors in the input interface	56	CASSLAVESTATE_CASSTATE_ERR	Slave status configuration error in slave data

For details about parameter errors, see the output structure PID_Output.ErrorID.

For all configuration errors, check the upper and lower limits of parameters.

■ Error handling

PIDConfig.InputConfig.ErrorAction: Action upon an error alarm

1. When PIDConfig.InputConfig.ErrorAction is FALSE, the input remains unchanged upon an error alarm. When the error is recovered, the current input is recalculated.

2. When PIDConfig.InputConfig.ErrorAction is TRUE, the alternative value of ErrorOut is output upon an error alarm. When the error is recovered, the current input is recalculated.

4.7. 6 PID_ATC

This instruction functions as the bipolar temperature PID controller.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
PID_ATC	Bipolar temperature PID controller PID controller	FC	PID_ATC_0 **PID_ATC** EN — ENO Enable — HeatPWM SP — HeatPercent PV — HeatAO AI — CoolPWM PIDConfig — CoolPercent InlockSW — CoolAO InlockVal — PIDOutput CasSlaveState — State StartAT — Error Kp — ErrorBits Ti — SlaveState Td KpCool TiCool TdCool CoolCoefficient	PID_ATC_0(Enable:= Enable1, SP:= , PV:= , AI:= , PIDConfig:= , InlockSW:= , InlockVal:= , CasSlaveState:= , Start AT:= Start AT1, Kp:= Kp1, Ti:= Ti1. Td:= Td1. KpCool:= KpCool, TiCool:= TiCool. TdCool:= TdCool. CoolCoefficient:= CoolCoefficient. HeatPWM=> , HeatPercent=> , HeatAO=> , CoolPWM=> , CoolPercent=> , CoolAO=> , PIDOutput=> , State=> , Error=> , ErrorBits=> , SlaveState= >);

■ Variables

Input variables

Variable	Name	Data Type	Value Range	Initial Value	Description
Enable	Enable control	BOOL	[FALSE, TRUE]	FALSE	Enable PID control TRUE: Run FALSE: Stop
SP	Target reference	REAL	Depends on data types	0	Reference (target value)
PV	Measured value	REAL	Depends on data types	0	Current value (measured value)
AI	Analog input value	DINT	Depends on data types	0	Analog input value
PIDConfig	Parameter configuration structure	PID_ATC_Config	Depends on data types	-	Parameter configuration structure
InlockSW	Interlock switch	BOOL	[FALSE, TRUE]	FALSE	Interlock switch

Variable	Name	Data Type	Value Range	Initial Value	Description
InlockVal	Interlock security value	REAL	[–100, 100]	0.0	Interlock security value
CasSlaveState	Slave circuit status	Slave_Data	Depends on data types	-	Slave circuit status

Slave_Data structure

Structure	Element	Data Type	Initial Value	Description
Slave_Data	PV	REAL	0	Process value of slave circuit When the slave circuit is detected as non-cascade automatic mode, the output of the master circuit tracks the process value of the slave circuit.
	OVE	BYTE	0	Overlimit status of slave circuit 0: Normal 1: Upper limit reached 2: Lower limit reached
	CAS_State	UINT	0	Cascade use mode 0: Slave circuit non-automatic mode 1: Slave circuit automatic mode

In-out variables

Variable	Name	Data Type	Value Range	Initial Value	Description
StartAT	Enable control	BOOL	[FALSE, TRUE]	FALSE	PID parameter auto-tuning enable. After auto-tuning is completed, internal auto reset occurs. TRUE: Enable auto-tuning FALSE: Disable auto-tuning
Kp	Scaling coefficient	REAL	[0.01, 1000]	60	Scaling coefficient
Ti	Integral time	REAL	[0, 10000]	400	Integral time (/s) A larger value causes less integral effect. When the value is 0, there is no integral action.
Td	Differential time	REAL	[0, 10000]	50	Differential time (/s) A larger value causes larger differential effect. When the value is 0, there is no differential action.
KpCool	Cooling scaling coefficient	REAL	[0.01, 1000]	0	Cooling scaling coefficient

Variable	Name	Data Type	Value Range	Initial Value	Description
TiCool	Cooling integral time	REAL	[0, 10000]	0	Cooling integral time (/s) A larger value causes less integral effect. When the value is 0, there is no integral action.
TdCool	Cooling differential time	REAL	[0, 10000]	0	Cooling differential time (/s) A larger value causes larger differential effect. When the value is 0, there is no differential action.
Cool Coefficient	Cooling coefficient	REAL	[0, 100]	0.1	Cooling coefficient

Output variables

Variable	Name	Data Type	Value Range	Initial Value	Description
HeatPWM	Heating PWM output	BOOL	[FALSE, TRUE]	FALSE	Heating PWM output TRUE: Enabled FALSE: Disabled
HeatPercent	Heating output percentage	REAL	[0, 100%]	0	Heating output percentage (selected based on output types)
HeatAO	Heating analog output value	DINT	Depends on data types	0	Heating analog output value
CoolPWM	Cooling PWM output	BOOL	[FALSE, TRUE]	FALSE	Cooling PWM output TRUE: Enabled FALSE: Disabled
CoolPercent	Cooling output percentage	REAL	[0, 100%]	0	Cooling output percentage (selected based on output types)
CoolAO	Cooling analog output value	DINT	Depends on data types	0	Cooling analog output value
PIDOutput	Analog output value	PID_Output	[0, 10000]	50	Analog output value

State	Operation status of the analog output value	DINT	Depends on data types	0	PID operation status: 0: Disable 1: Manual 2: Auto 3: Interlock (reserved) 4: Auto-tuning 5: Error value output 6: Alternative output for the incorrect security value 7: Master tracking (reserved)
Error	Running error	BOOL	[FALSE, TRUE]	FALSE	Running error
ErrorBits	Error code	DINT	For details, see error handling.	0	Error code. bit0: PV overlimit bit1: Parameter error
SlaveState	Slave circuit status	Slave_Data	Depends on data types	-	Slave circuit status

PID_Output structure

Structure	Element	Data Type	Initial Value	Description
PID_Output	PVOut	REAL	100	Actually calculated PV value
	ECMaxOut	REAL	1000	Fuzzy control, maximum temperature rise speed, in ° C/s
	ATState	DINT	0	Auto-tuning status. 0: No auto-tuning; 1: During auto-tuning; 2: Auto-tuning completed
	ErrorID	DINT	0	Parameter error code
	SPOut	REAL	FALSE	Actually calculated SP value

For details about parameter errors, see the output structure PID_Output.ErrorID.

The PIDConfig variable structure type is PID_ATC_Config.

Variable	Type	Description
GeneralConfig	PID_GeneralConfig	PID general configuration
InputConfig	PID_InputConfig	PID input configuration
Output ConfigHeat	PID_OutputConfig	PID heating output configuration
Output ConfigCool	PID_OutputConfig	PID cooling output configuration
CoolConfig	PID_CoolConfig	PID cooling configuration
SuperConfig	PID_SuperConfig	PID advanced configuration

The structure of the PID general configuration PID_GeneralConfig contains the following variables.

Variable	Name	Data Type	Value Range	Initial Value	Description
SampleTime	Heating sampling period	DINT	[1, 1000000]	100	Heating sampling period (ms)
CycleTime	Heating output period	DINT	[1, 1000000]	1000	Heating output period (ms)
PIDType	PID control type	DINT	0, 1, 2, and 3	0	0: PID 1: PI 2: PD 3: ID
ControlMode	Control mode	DINT	0, 1, 2, and 3	0	0: Reverse (unipolar heating) 1: Forward (unipolar cooling) 2: Bipolar control based on cooling coefficient 3: Bipolar control based on independent PID parameter
ManualEnable	Manual mode switch	BOOL	[FALSE, TRUE]	FALSE	Manual mode switch
ManualValue	Manual mode output	REAL	[OutMin, OutMax]	0	Manual mode output, with the value range following the maximum and minimum outputs (%)
ATCoefficient	Auto-tuning coefficient	REAL	[0, 1]	0.5	Auto-tuning coefficient
DelayStartUp	Delayed start time	DINT	[0, 1000000]	0	Delayed start time. PID is started after delay of the set time for off-peak start in case of multi-channel temperature control on the device. (ms)
Delay shutdown	Delayed shutdown time	DINT	[0, 1000000]	0	Delayed shutdown time. PID is shut down after delay of the set time. (ms)
Physical Quantity	Physical object to control	DINT	0 and 1	1	Physical object to control 0: General 1: Temperature
PhysicalUnit	Physical object unit	DINT	Depends on data types	0	Unit of the physical object (reserved)
IsMaster	Master control	BOOL	[FALSE, TRUE]	FALSE	Whether the control is master control 0: No 1: Yes
IsSlave	Slave control	BOOL	[FALSE, TRUE]	FALSE	Whether the control is slave control 0: No 1: Yes

The structure of the PID input configuration PID_InputConfig contains the following variables.

Variable	Name	Data Type	Value Range	Initial Value	Description
PVType	PV input type selection	DINT	0 and 1	0	0: Use the current PV as the input 1: Use the peripheral analog as the input and convert it based on the analog range
AIMin	Minimum value for analog conversion	DINT	Depends on data types	0	Minimum value for analog conversion
AIMax	Maximum value for analog conversion	DINT	Depends on data types	20000	Maximum value for analog conversion
PVMin	Minimum value of PV	REAL	Depends on data types	0	Minimum value of PV
PVMax	Maximum value of PV	REAL	Depends on data types	100	Maximum value of PV
PVLowLimit	Lower alarm limit for PV	REAL	Depends on data types	–100	Lower alarm limit for PV
PVHighLimit	Higher alarm limit for PV	REAL	Depends on data types	1000	Higher alarm limit for PV
ErrorAction	Action for error alarm	BOOL	[FALSE, TRUE]	FALSE	Action for error alarm: FALSE: Remain the value upon an error alarm TRUE: Output the alternative value upon an error alarm
ErrorOut	Error output percentage	REAL	[OutMin, OutMax]	0	Output of an alternative percentage upon an error alarm
SPSel	Reference source	BOOL	[FALSE, TRUE]	FALSE	Reference source: FALSE: External reference (SP configuration) TRUE: Internal reference (panel operation reference)
SPRatio	SP reference change	REAL	Depends on data types	0	SP reference change
SPInt	Internal reference	REAL	Depends on data types	0	Internal reference
SPTrack	Internal reference tracking switch	BOOL	[FALSE, TRUE]	TRUE	Internal reference tracking switch The default value is TRUE.

The structure of the PID output configuration PID_OutputConfig contains the following variables.

Variable	Name	Data Type	Value Range	Initial Value	Description
OutType	Output type	DINT	0, 1, and 2	0	Output type: 0: PWM output 1: Analog output 2: Percent output

OutMin	Minimum value of output percentage	REAL	[–100, 100]	0	Minimum value of output percentage (%)
OutMax	Maximum value of output percentage	REAL	[–100, 100]	100	Maximum value of output percentage (%)
AOMin	Minimum value of analog output	DINT	Depends on data types	0	Minimum value of analog output
AOMax	Maximum value of analog output	DINT	Depends on data types	20000	Maximum value of analog output
OutputOffset	Output deviation compensation value	REAL	[–100, 100]	0	Output deviation compensation value (%)
MinPT	Minimum pulse value	DINT	[0, 10000]	0	Minimum pulse value (ms)

The structure of the PID output configuration PID_CoolConfig contains the following variables.

Variable	Name	Data Type	Value Range	Initial Value	Description
CoolPIDType	Controller type	DINT	0, 1, 2, and 3	0	0: PID 1: PI 2: PD 3: ID
Cool SampleTime	Cooling sampling period	DINT	[1, 1000000]	100	Cooling sampling period (ms)
CoolCycleTime	Cooling output period	DINT	[1, 1000000]	5000	Cooling output period (ms)
CoolOffset	Cooling deviation	REAL	[–1000, 1000]	0	Cooling deviation Cooling reference = SP + rOffsetCool

The structure of the PID advanced configuration PID_SuperConfig contains the following variables.

Variable	Name	Data Type	Value Range	Initial Value	Description
Fuzzy	Fuzzy self-adaption	BOOL	[FALSE, TRUE]	0	0: Fuzzy self-adaption disabled 1: Fuzzy self-adaption enabled
Integral Suspend	Integral pause	BOOL	[FALSE, TRUE]	0	0: Integral enabled 1: Integral paused
Integral Division	Integral separation	BOOL	[FALSE, TRUE]	0	0: Integral separated 1: Integral not separated
IntegralClear	Integral clear	BOOL	[FALSE, TRUE]	0	0: Normal integral 1: Clear integral
ECMax	Maximum temperature rise speed per second	REAL	Depends on data types	0	Maximum temperature rise speed per second

Variable	Name	Data Type	Value Range	Initial Value	Description
KdGain	Incomplete differential coefficient	REAL	[0, 1]	0	Incomplete differential coefficient (0.0 to 1.0) A larger value indicates smoother differential effect. When the value is 1, there is no differential action.
IntegralMax	Maximum integral value	REAL	[–100, 100]	75	Maximum integral value (%)
IntegralCut	Integral cutting coefficient	REAL	[0, 1.0]	0.75	Integral cutting coefficient for reference change
Integral DivLimit	Integral separation limit	REAL	[0, 100]	50	Integral separation limit
DeadBand	Deviation dead zone limit	REAL	[0, 100]	0	Deviation dead zone limit

The structure of the PID output structure PID_Output contains the following variables.

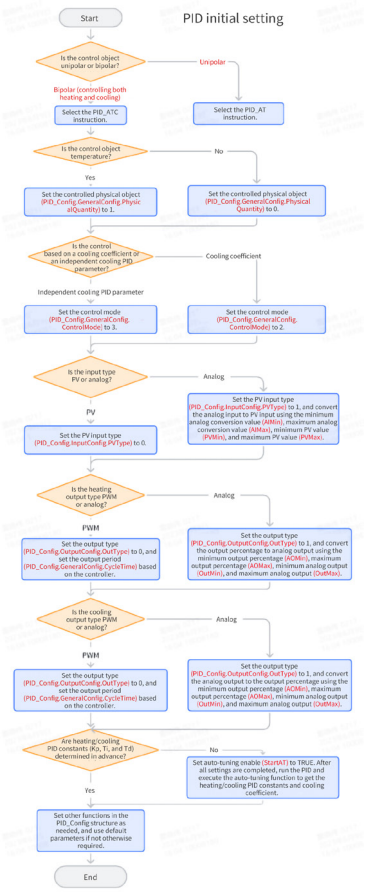
Variable	Name	Data Type	Value Range	Initial Value	Description
PVOut	Analog PV value	REAL	Depends on data types	0	PV value converted from analog
ECMaxOut	Maximum temperature rise speed per second	REAL	Depends on data types	0	Maximum temperature rise speed per second
ATState	Auto-tuning status	DINT	0, 1, and 2	0	Auto-tuning status 3: No auto-tuning 4: During auto-tuning 5: Auto-tuning completed
ErrorID	Parameter error code	DINT	Depends on data types	0	Parameter error code
SPOut	Actual SP value	REAL	Depends on data types	0	Actually calculated SP value

The slave circuit status structure Slave_Data contains the following variables.

Variable	Name	Data Type	Value Range	Initial Value	Description
PV	Process value of slave circuit	REAL	Depends on data types	0	Process value of slave circuit When the slave circuit is detected as non-cascade automatic mode, the output of the master circuit tracks the process value of the slave circuit.

Variable	Name	Data Type	Value Range	Initial Value	Description
OVE	Overlimit status of slave circuit	BYTE	0, 1, and 2	0	Overlimit status of slave circuit 0: Normal 1: Upper limit reached 2: Lower limit reached
CAS_State	Cascade use mode	UINT	0 and 1	0	Cascade use mode 0: Slave non-automatic mode 1: Slave automatic mode In case of slave, the value should be returned to the master controller SUBCAS_MODE.

■ PID initial setting



■ Description

1. Enable

When the value of execution condition Enable changes to TRUE, PID calculation starts. When the value of Enable is TRUE, the following operations are periodically repeated:

Obtain the measured value PV → Perform PID calculation → Output the workload

After the value of Enable changes to FALSE, PID calculation is completed and the output is cleared.

2. StartAT (auto-tuning)

This instruction provides the auto-tuning function to automatically calculate the optimal values for PID constants Kp, Ti, and Td.

The auto-tuning execution conditions upon heating are as follows:

1) The current value is less than the reference. In general, the reference is the common temperature set for working.

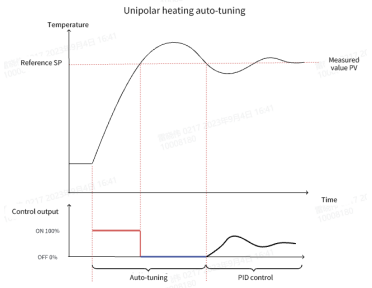
2) When the value of StartAT is TRUE, auto-tuning is executed from the beginning of operation if the value of Enable changes from FALSE to TRUE. When the value of Enable is TRUE and PID calculation is in progress, auto-tuning is executed during the operation if the value of StartAT changes from FALSE to TRUE.

After auto-tuning starts, the value of State.1 changes to 1. When auto-tuning is completed, the value of State.1 changes to 0 and the value of State.2 changes to 1.

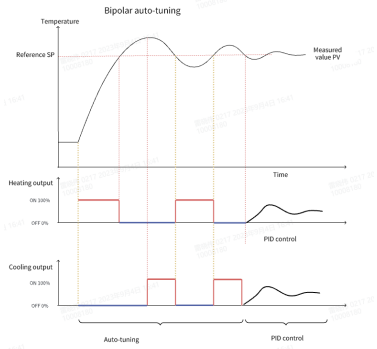
In either case, when the value of StartAT changes from TRUE to FALSE during auto-tuning, auto-tuning is terminated, with State.1 of 0 and State.2 of 0.

Note: After auto-tuning is completed, StartAT is automatically set to FALSE.

The following figure shows the tuning process of unipolar heating.



During bipolar auto-tuning, two waveforms are generated near the reference. After auto-tuning is completed, StartAT is set to FALSE.



Note: During the auto-tuning process, parameter configuration cannot be modified. Otherwise, the parameters may be improper.

3. Scaling coefficient Kp

The scaling coefficient can be set by a user or automatically obtained through auto-tuning.

The value of Kp must be greater than 0. Otherwise, the value of Error is TRUE and eErrorID is an error number. A larger value indicates stronger scaling.

4. Integral time Ti

The integral time can be set by a user or automatically obtained through auto-tuning. If the value of Ti is 0, there is no integral action. A larger value indicates a weaker integral action.

The value of Ti cannot be less than 0. Otherwise, the value of Error is TRUE and ErrorID is an error number.

5. Differential time Td

The differential time can be set by a user or automatically obtained through auto-tuning. If the value of Td is 0, there is no differential action. A larger value indicates a stronger differential action.

The value of Td cannot be less than 0. Otherwise, the value of Error is TRUE and ErrorID is an error number.

6. Cooling scaling coefficient KpCool

The cooling scaling coefficient can be set by a user or automatically obtained through auto-tuning. A larger value indicates stronger scaling.

The value of KpCool must be greater than 0. Otherwise, the value of Error is TRUE and eErrorID is an error number.

7. Cooling integral time TiCool

The cooling integral time can be set by a user or automatically obtained through auto-tuning. If the value of TiCool is 0, there is no integral action. A larger value indicates a weaker integral action.

The value of TiCool cannot be less than 0. Otherwise, the value of Error is TRUE and ErrorID is an error number.

8. Cooling differential time TdCool

The cooling differential time can be set by a user or automatically obtained through auto-tuning. If the value of TdCool is 0, there is no differential action. A larger value indicates a stronger differential action.

The value of TdCool cannot be less than 0. Otherwise, the value of Error is TRUE and ErrorID is an error number.

9. Cooling coefficient CoolCoefficient

When the value of GeneralConfig.Mode is 2, the cooling output is calculated based on the cooling efficient CoolCoefficient. The cooling efficient can be set by a user or automatically obtained through auto-tuning. If the value of CoolCoefficient is 0, there is no cooling action.

The value of CoolCoefficient cannot be less than 0. Otherwise, the value of Error is TRUE and ErrorID is an error number.

10. InlockSW (interlock switch)

When the interlock switch is triggered, the interlock security value is output.

11. InlockVal (interlock security value)

12. CasSlaveState (slave status structure Slave_Data)

· CasSlaveState.PV: Process value of slave circuit

· CasSlaveState.OVE: Overlimit status of slave circuit

0: Normal

1: Upper limit reached

2: Lower limit reached

· CasSlaveState.CAS_State: Cascade use mode

2: Slave non-automatic mode

3: Slave automatic mode

In case of slave, the value should be returned to the master controller SUBCAS_MODE.

13. PID_Config (PID parameter configuration)

The options include the following:

GeneralConfig (general configuration), InputConfig (input configuration), OutputConfigHeat (heating output configuration), OutputConfigCool (cooling output configuration), and SupterConfig (advanced configuration)

14. Parameter type PID_GeneralConfig for general configuration GeneralConfig:

· GeneralConfig.SampleTime: Sampling period

The current PV value is obtained once a sampling period for once PID calculation. In general, the sampling period is that of the temperature collection module. For general requirements, the default value 100 ms can be used.

· GeneralConfig.CycleTime: Output period

Output period of HeatPWM. If the HeatPercent output value of the PID is 80% and the value of GeneralConfig.CycleTime is 10000 milliseconds, the period for HeatPWM is 10000 milliseconds, with 8s on and 2s off.

· GeneralConfig.PIDType: Current PID type

0: PID calculation

1: PI calculation

2: PD calculation

3: ID calculation

Perform corresponding calculation control based on the selected PID type. The default value is 0.

· GeneralConfig.ControlMode: Control mode

0: Reverse (unipolar heating)

1: Forward (unipolar cooling)

2: Bipolar control based on cooling coefficient

≫ The refrigerating parameter is calculated based on the PID parameters and cooling coefficient of the heating process.

≫ It is applicable when the time response is similar but gains are different for the heating activator and cooling activator.

3: Bipolar control based on independent PID parameter
Heating and cooling are separately controlled based on the configured parameter.

≫ Independent cooling PID parameters are used.

≫ It is applicable when the time response is similar but gains are different for the heating activator and cooling activator.

· GeneralConfig.ManualEnable: Manual mode switch

0: Automatic mode

1: Manual mode

When the manual mode is enabled, the value of GeneralConfig.ManualValue is output.

In bipolar mode, the positive value of GeneralConfig.ManualValue corresponds to the heating output value, and the negative value corresponds to the cooling output value.

· GeneralConfig.ATCoefficient: Auto-tuning coefficient

For the auto-tuning coefficient, the larger the Kp parameter value obtained from auto-tuning, the faster the set temperature can be reached, but it may cause greater overshoot. The smaller the auto-tuning coefficient, the slower the set temperature parameter obtained from auto-tuning can be reached, and the temperature curve will be very smooth, without overshoot or with very small overshoot. Use the default value 0.5 for the parameter of initial tuning. If the customer requires to reach the set temperature faster, increase the auto-tuning coefficient GeneralConfig.ATCoefficient for another tuning based on the control effect.

If the customer needs a smooth curve and small overshoot, decrease the auto-tuning coefficient GeneralConfig.ATCoefficient for another tuning.

· GeneralConfig.DelayStartUp: Delayed start time

PID is started after delay of different set time for off-peak start in case of multi-channel temperature control on the device. This prevents a large current fluctuation during multi-channel start.

· GeneralConfig.Delayshutdown: Delayed shutdown time

Delayed shutdown time. PID is shut down after delay of the set time.

· GeneralConfig.PhysicalQuantity: Physical object to control

2: General

3: Temperature

· GeneralConfig.PhysicalUnit: Unit of the physical object

· GeneralConfig.IsMaster: Whether the control is master control

2: No

3: Yes

· GeneralConfig.IsSlave: Whether the control is slave control

2: No

3: Yes

15. Parameter type PID_InputConfig for input configuration InputConfig:

· InputConfig.PVType: Type of sampling PV

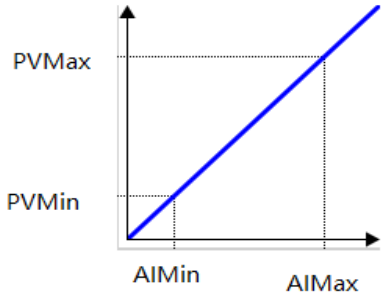
0: Use the current PV as the input

1: Use the peripheral analog as the input and convert it based on the analog range InputConfig.AIMin and InputConfig.AIMax as well as the corresponding PVMin and PVMax

· InputConfig.AIMin: Minimum value for analog conversion
InputConfig.AIMax: Maximum value for analog conversion

They are used to convert an analog to a PV value.

Use the peripheral analog as the input and convert it based on the analog range InputConfig.AIMin and InputConfig.AIMax as well as the corresponding PVMin and PVMax



· InputConfig.PVMin and InputConfig.PVMax

InputConfig.PVMin: Minimum PV value corresponding to InputConfig.AIMin

InputConfig.PVMax: Minimum PV value corresponding to InputConfig.AIMax

Use the peripheral analog as the input and convert it based on the analog range InputConfig.AIMin and InputConfig.AIMax as well as the corresponding PVMin and PVMax

· InputConfig.InputMin and InputConfig.InputMax: Input upper limit and lower limit - alarm

When the PV value or the PV value converted from the analog is lower than the value of InputConfig. InputMin, an alarm is generated for exceeding the lower limit.

When the PV value or the PV value converted from the analog is higher than the value of InputConfig. InputMax, an alarm is generated for exceeding the upper limit.

Alarms are output based on InputConfig.ErrorAction and ErrorOut.

· InputConfig.ErrorAction: Action for error alarm
InputConfig.ErrorOut: Output of an alternative value upon an error alarm

FALSE: Remain the output value upon an error alarm

TRUE: Output the alternative value InputConfig.ErrorOut upon an error alarm

1: Use the peripheral analog as the input and convert it based on the analog range InputConfig.AIMin and InputConfig.AIMax as well as the corresponding PVMin and PVMax

· InputConfig.SPSel: Reference source

Select the SP value used for PID calculation:

FALSE: External reference (SP configuration)

TRUE: Internal reference (panel operation reference SPInt)

· InputConfig.SPRatio: SP reference change

If SPRatio is greater than 0, the final value turns closer to the changing reference based on the change of SPRatio. It is calculated once every task period.

· InputConfig.SPInt: SP internal reference

If SPSel is TRUE, select the internal reference of SPInt as the current reference (panel operation reference SPInt)

· InputConfig.SPTrack: Internal reference tracking switch
The default value is TRUE.

FALSE: The internal reference does not track the external reference.

TRUE: The internal reference tracks the external reference.

16. Parameter type PID_OutputConfig for heating output configuration OutputConfigHeat:

· OutputConfigHeat.OutType: Output type

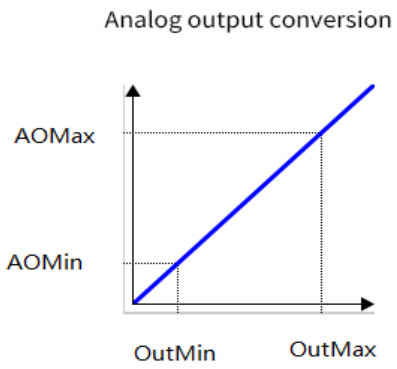
0: PWM output
HeatPercent indicates the percentage for PWM output.

1: Analog output
HeatPercent indicates the percentage for analog output.

· OutputConfigHeat.OutMin and OutputConfigHeat.OutMax: minimum output percentage and maximum output percentage

It is used to limit the output of PID calculation and convert the analog output.

Convert the output percentage of PID calculation based on the analog range OutputConfigHeat.OutMin and OutputConfigHeat.OutMax as well as the corresponding OutputConfigHeat.AOMin and OutputConfigHeat.AOMax.



· OutputConfigHeat.AOMin and OutputConfigHeat.AOMax: minimum analog output and maximum analog output

It is used to convert the analog output.

Convert the output percentage of PID calculation based on the analog range OutputConfigHeat.OutMin and OutputConfigHeat.OutMax as well as the corresponding OutputConfigHeat.AOMin and OutputConfigHeat.AOMax.

· OutputConfigHeat.OutputOffset: Output deviation compensation value

OUT = PID calculation value + OutputOffset: The final output is equal to the PID calculation value plus the value of OutputOffset. It can be used for the feedforward compensation.

· OutputConfigHeat.MinPT: Minimum pulse enable/disable time of PWM, in milliseconds

The pulse is not enabled when the enable time is less than MinPT.

The pulse is not disabled when the disable time is less than MinPT.

17. Parameter type PID_OutputConfig for cooling output configuration OutputConfigCool:

· OutputConfigCool.OutType: Output type

0: PWM output
CoolPercent indicates the percentage for PWM output.

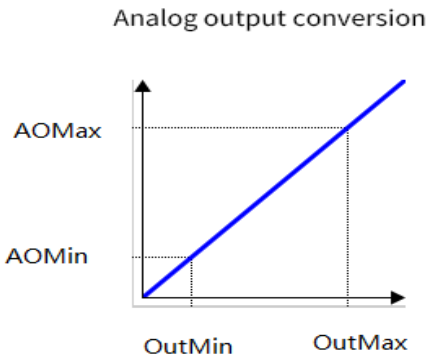
1: Analog output
CoolPercent indicates the percentage for analog output.

· OutputConfigCool.OutMin and OutputConfigCool.OutMax: minimum output percentage and maximum output percentage

It is used to limit the output of PID calculation and convert the analog output.

Convert the output percentage of PID calculation based on the analog range OutputConfigCool.Out-

Min and OutputConfigCool.OutMax as well as the corresponding OutputConfigCool.AOMin and Output-ConfigCool.AOMax.



· OutputConfigCool.AOMin and OutputConfigCool.AOMax: minimum analog output and maximum analog output

It is used to convert the analog output.

Convert the output percentage of PID calculation based on the analog range OutputConfigCool.Out-Min and OutputConfigCool.OutMax as well as the corresponding OutputConfigCool.AOMin and Output-ConfigCool.AOMax.

· OutputConfigCool.OutputOffset: Output deviation compensation value

OUT = PID calculation value + OutputOffset: The final output is equal to the PID calculation value plus the value of OutputOffset.

· OutputConfigCool.MinPT: Minimum pulse enable/disable time of PWM, in milliseconds

The pulse is not enabled when the enable time is less than MinPT.

The pulse is not disabled when the disable time is less than MinPT.

18. Parameter type PID_CoolConfig for cooling configuration CoolConfig:

· CoolConfig.CoolPIDType: Current PID type

0: PID calculation

1: PI calculation

2: PD calculation

3: ID calculation

Perform corresponding calculation control based on the selected PID type. The default value is 0.

· CoolConfig.CoolSampleTime: Sampling period

The current PV value is obtained once a sampling period for once PID calculation. In general, the sampling period is that of the temperature collection module. For general requirements, the default value 100 ms can be used.

· CoolConfig.CoolCycleTime: Output period

Output period specified by CoolPWM. If the CoolPercent output value of the PID is 80% and the value of CoolConfig.CoolCycleTime is 10000 milliseconds, the period for CoolPWM is 10000 milliseconds, with 8s on and 2s off.

· CoolConfig.CoolOffset: Cooling deviation

Cooling reference = Heating reference SP + CoolOffset

19. Parameter type PID_SuperConfig for advanced configuration SuperConfig:

· SuperConfig.Fuzzy: Fuzzy self-adaption enable

0: Fuzzy self-adaption disabled

1: Fuzzy self-adaption enabled

PID parameters can be dynamically adjusted based on fuzzy self-adaption (available for unipolar heating).

· SuperConfig.ECMax: Maximum temperature rise speed per second

It is used in fuzzy self-adaptive mode. The maximum temperature rise speed of the device can be input by the user or obtained through auto-tuning (PIDOutput.ECMaxOut).

· SuperConfig.KdGain: Incomplete differential coefficient

The value range is 0.0 to 1.0. A larger value causes smoother differential effect.

When the value is 1, there is no differential action.

· SuperConfig.IntegralSuspend: Integral pause

0: Integral enabled (default)

1: Integral paused

· SuperConfig.IntegralDivision: Integral separation

0: Integral separated (default)

1: Integral not separated

· SuperConfig.IntegralClear: Integral clear

0: Normal integral (default)

1: Clear integral

· SuperConfig.IntegralMax: Maximum integral value

The maximum integral value is 75% by default.

· SuperConfig.IntegralCut: Integral cutting coefficient for reference change

The integral cutting coefficient is 0.75 by default.

· SuperConfig.IntegralDivLimit: Integral separation limit

When the deviation is greater than IntegralDivLimit, no integration is performed.

· SuperConfig.DeadBand: Deviation dead zone limit

When deviation |SP – PV| is less than DeadBand, the deviation is considered as 0 (no PID calculation).

· PIDOutput.SPOut: Actually calculated SP value

20. Output structure:

· PIDOutput.PVOut: Actually calculated PV value

· PIDOutput.ECMaxOut: Maximum temperature rise speed per second

The output is auto-tuning output, which is used for fuzzy control.

· PIDOutput.ATState: Auto-tuning status

0: No auto-tuning

1: During auto-tuning

2: Auto-tuning completed

· PIDOutput.ErrorID: Parameter error code

For details, see the error codes.

■ Function

PID_ATC is a dedicated temperature controller designed for heating, cooling, and heating/cooling applications. It provides two independent outputs, one for heating and one for cooling. It supports cascade configuration applications and parameter auto-tuning.

≫ Calculation formula

$$\mathbf{AV} = \boldsymbol{K_p} * \left[(\boldsymbol{SP} - \boldsymbol{PV}) + \frac{\mathbf{1}}{\boldsymbol{T_i} * \boldsymbol{S}} (\boldsymbol{SP} - \boldsymbol{PV}) + \frac{\boldsymbol{T_d}}{\boldsymbol{S}} (\boldsymbol{SP} - \boldsymbol{PV}) \right]$$

Discretization formula

$$\mathbf{AV(k)} = \boldsymbol{K_p} * \mathbf{E}(\boldsymbol{k}) + \boldsymbol{K_p} * \frac{\boldsymbol{cycle}}{\boldsymbol{T_i}} \sum \mathbf{E}(\boldsymbol{k}) + \frac{\boldsymbol{K_p} * \boldsymbol{T_D}}{\boldsymbol{cycle}} (\mathbf{E}(\boldsymbol{k}) - \mathbf{E}(\boldsymbol{k} - \mathbf{1}))$$

≫ Working modes

Working mode	Action
Automatic	The output value of the PID module is calculated based on the PID rule. The calculation result of the PID module is restricted by the output amplitude.
Manual	The output is manually controlled (operator write value or online logical write value). The output of this mode is restricted by the output amplitude.
Interlock	In the working mode, PID stops automatic calculation, and the output value of the PID module is a pre-set value of InlockVal, which is generally referred to as the interlock point. The interlock functions are also used for tracking. When a certain condition (interlock switch InlockSW is TRUE) is met, the PID module automatically switches to the working mode. When the interlock condition is not met, the PID module switches back to the previous working mode.

1) Interlock has the highest priority, followed by manual, and finally automatic.
2) Automatic mode: For much control with fixed set-points, references can be configured on the interface.
3) Choose to use the external reference or internal reference based on the configuration of PIDConfig. InputConfig.SPSel.

SPSel = FALSE: External reference of the logical configuration

SPSel = TRUE: Internally given reference

Note: To prevent circuit fluctuations that may occur during the switchover process of internal and external references, PIDConfig.InputConfig.SPTrack (whether the internal reference follows the external reference) can be set to TRUE. With the external reference selected, the internal reference can be interlocked with the external reference to ensure the unperturbed switchover of the internal reference.

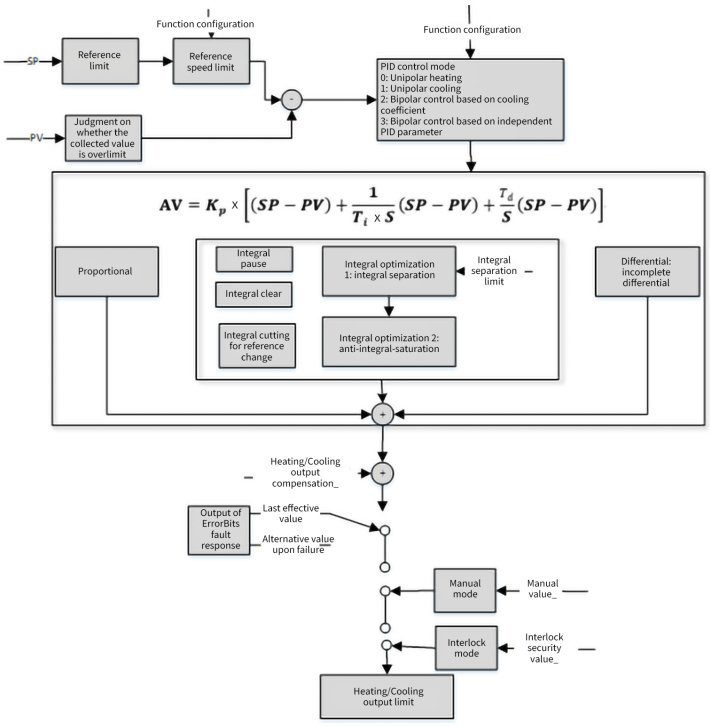
If real-time values of the logical configuration are directly used during the switchover from the internal reference to the external reference, perturbation may occur.

≫ Unperturbed switchover

Interlock to manual	In interlock mode, the interlock value is directly output. In this case, the manual value tracks the output value to ensure the unperturbed switchover.
Interlock to automatic	In interlock mode, the interlock value is directly output. After the interlock mode is switched to the automatic mode, if the external reference is used, the logic operation value is directly used for the PID calculation output, which may cause perturbation.
Manual to interlock	The manual output is the manual value. After the manual mode is switched to the interlock mode, the output is the interlock value, and perturbation occurs.

Manual to automatic	The manual output is the operator reference. If the external reference is used, the logic operation value is directly used for the PID calculation output. In this case, the switchover is unperturbed (the incremental part is perturbed).
Auto to interlock	In automatic mode, the output is the normal operation value. After the automatic mode is switched to the interlock mode, the output is the interlock value, and perturbation occurs.
Automatic to manual	In automatic mode, the output is the normal operation value. In this case, the manual value tracks the output value to ensure the unperturbed switchover.

≫ Logic diagram of working (taking reverse as an example)



≫ Various control modes:

Four control modes can be selected as needed.

Control Mode PIDConfig.GeneralConfig. ControlMode	Control Mode	Description
0	Unipolar heating	It is applicable to heating only.
1	Unipolar cooling	It is applicable to cooling only.
2	Bipolar control based on cooling coefficient	It is applicable when the time response is similar but gains are different for the heating activator and cooling activator. The refrigerating output is calculated based on the PID parameters and cooling coefficient CoolCoefficient of the heating control.
3	Bipolar control based on independent PID parameter	It is applicable when the time response is similar but gains are different for the heating activator and cooling activator. The output is calculated based on independent heating and cooling PID parameters.

≫ Reference change limit function

In automatic mode, the reference (including external and internal references) change is checked. If the change exceeds a preset limit (specified by PIDConfig.InputConfig.SPRatio), the actual reference used

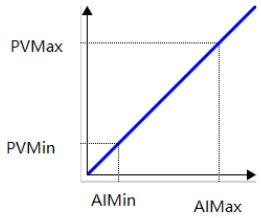
in the calculation is restricted to the change value and the output will gradually approach the reference based on the magnitude of the change.

$$\Delta SP > SPRatio\left(设定值变化量\right): SP = LastSP + SPRatio$$

$$\Delta SP < SPRatio\left(设定值变化量\right): SP = LastSP - SPRatio$$

≫ Input signal selection function

Input signals of two modes can be selected.

PIDConfig.InputConfig.PVType	Input	-
0	Current input (PV)	-
1	Analog input (AI)	Linear conversion is calibrated based on the upper limits and lower limits of the analog and PV value.  PIDConfig.InputConfig.AIMax: Upper limit of analog PIDConfig.InputConfig.AIMin: Lower limit of analog PIDConfig.InputConfig.AIMax: Upper limit of PV PIDConfig.InputConfig.AIMin: Lower limit of PV

≫ Integral function

· Integral separation

Integral saturation is mainly caused by the integral term. To overcome integral saturation, it is crucial to limit the integral action to prevent excessive integral accumulation. To eliminate integral saturation, the integral separation function can be employed to provide the anti-integral-saturation capability to the function block.

For non-temperature objects, the configuration for integral separation is as follows:

Integral separation enable PIDConfig.GeneralConfig.IntegralDivision	Integral separation limit PIDConfig.GeneralConfig.IntegralDivLimit
TRUE	When |Deviation| is greater than or equal to the integral separation limit, no integral component operation is performed. In this case, the control is PD control.
	When |Deviation| is less than the integral separation limit, the full integral operation is performed. In this case, the control is PID control.
FALSE	The full integral operation is always performed.

· Integral pause

Set PIDConfig.GeneralConfig.IntegralSuspend to TRUE to trigger integral pause. The integral action remains unchanged and no more integral accumulation occurs.

· Integral clear

Set PIDConfig.GeneralConfig.IntegralClear to TRUE to trigger integral clear. The integral action is cleared to 0.

· Integral cutting coefficient

When the reference changes, set PIDConfig.SuperConfig.IntegralCut to reduce the initial integral action. The value range is from 0 to 1. The default value is 0.75.

≫ Incomplete differential function

The incomplete differential function adds a first-order inertial element to the differential action, which helps to smooth the differential effect and effectively improves the control performance of the system.

Parameter PIDConfig.SuperConfig.KdGain	Differential action
0.0	The differential action takes effect only within this adjustment period. The default value is 0.0.
(0.0, 1.0)	A larger value indicates a smoother differential action.
1.0	The differential action does not take effect.

≫ Dead zone control function

The PIDConfig.SuperConfig.DeadBand parameter (specifying the dead zone width) can be set. Within the dead zone range, the adjustment is stable and the PID output does not change. If the value exceeds the dead zone range, PID calculation is normal.

≫ Delayed start/stop function

This function is used to set different delayed start/stop time for off-peak start/stop in case of multi-circuit control for the device. This prevents a large current fluctuation during multi-channel start.

PIDConfig.GeneralConfig.DelayStartUp	After the time specified by DelayStartUp, restart PID calculation.
PIDConfig.GeneralConfig.Delayshutdown	After the time specified by Delayshutdown, stop PID calculation.

≫ Output compensation function

The output compensation function directly adds the output compensation value to the result of automatic PID calculation to obtain the output value. When the controller is in automatic mode, the output compensation functions. When the controller is in manual mode, the output compensation does not function.

Heating output = Heating PID calculation output + Heating output compensation (PIDConfig.OutputConfigHeat.OutputOffset)

Cooling output = Cooling PID calculation output + Cooling output compensation (PIDConfig.OutputConfigCool.OutputOffset)

≫ Output amplitude limit function

When the heating output value is greater than the heating output upper limit specified by PIDConfig.OutputConfigHeat.OutMax, the heating output value is equal to the heating output upper limit.

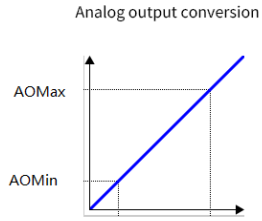
When the heating output value is less than the heating output lower limit specified by PIDConfig.OutputConfigHeat.OutMin, the heating output value is equal to the heating output lower limit.

When the cooling output value is greater than the cooling output upper limit specified by PIDConfig.OutputConfigHeat.OutMax, the cooling output value is equal to the cooling output upper limit.

When the cooling output value is less than the cooling output lower limit specified by PIDConfig.OutputConfigHeat.OutMin, the cooling output value is equal to the cooling output lower limit.

≫ Output signal selection function

Three modes are supported for output control:

PIDConfig.OutputConfig-Heat.OutType: Heating output type PIDConfig.OutputConfig-Cool.OutType: Cooling output type	Output	-
0	PWM output	The digital is converted and output based on the PWM output period.
1	Analog output	Linear conversion is calibrated and output based on the upper limits and lower limits of the analog and PID output.  PIDConfig.OutputConfig.AOMax: Upper limit of analog output PIDConfig.OutputConfig.AOMin: Lower limit of analog output PIDConfig.OutputConfig.OutMax: Upper limit of PID output PIDConfig.OutputConfig.OutMin: Lower limit of PID output
2	Percent output	Output within 0% to 100%

≫ Fault detection

When any bit of ErrorBits is triggered, Error is TRUE.

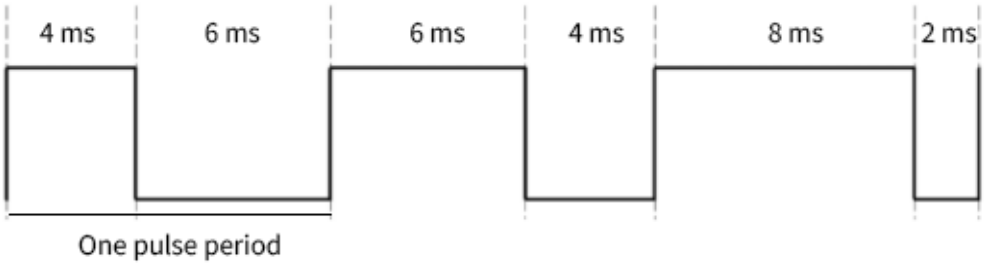
Bit of ErrorBits		Error output configuration parameter: PIDConfig.InputConfig.ErrorAction
Bit0 = 1	PV overlimit alarm	FALSE: The output remains TRUE: The alternative value PIDConfig.InputConfig.ErrorOut is output
Bit1 = 1	parameter configuration error alarm	

For details about parameter configuration errors, see ErrorID.

≫ PWM output

For PWM, the output is ON or OFF. The analog of PID output needs to be converted to digital output (ON/OFF) based on the duty cycle. The duty cycle is the percentage of the high-level pulse time to a whole pulse period.

For example, if the opening of a PID regulator's output is 40% and the PWM output period PIDConfig.GeneralConfig.CycleTime is set to 10s, the ON time for the 10s period would be 10s × 40% = 4s, and the OFF time would be 10s – 4s = 6s. The duty cycle of 40%. Therefore, within the 10s period, the output is the ON signal for 4s, followed by the OFF signal for 6s, and then the next period calculation begins.

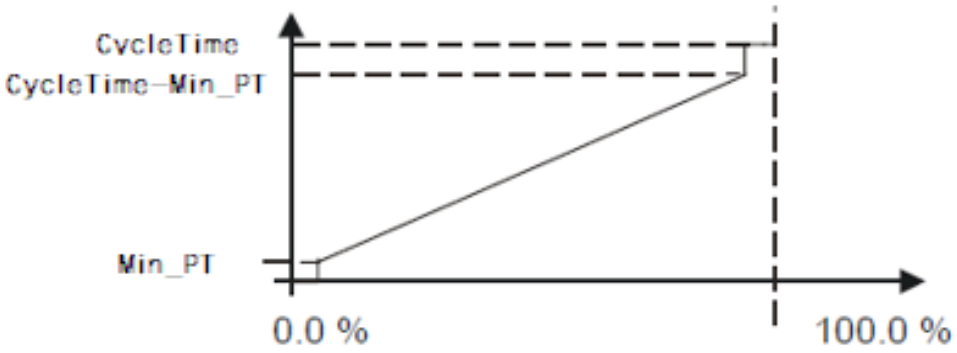


· Minimum pulse on/off time

To prevent frequent operation on the actuator, PIDConfig.OutputConfigHeat.Min_PT can be set to specify the minimum pulse on/off time for heating, and PIDConfig.OutputConfigCool.Min_PT can be set to specify the minimum pulse on/off time for cooling.

When the calculated high-level pulse time is less than the minimum pulse on/off time, the output duty cycle is 0%.

When the calculated high-level pulse time is greater than the minimum pulse on/off time in the PWM output period, the output duty cycle is 100%.



≫ Cascade function

Cascade control involves two or more cascaded PID controllers. The control variable of the main controller is interconnected with the reference of the slave controller, creating a nested control circuit. Cascade control in a slave circuit can compensate for interference in the internal circuit faster than in the slower main circuit.

Note: Cascade control is possible only when there are other measurable variables in the process, and the internal control circuit is much faster than the external circuit.

Unperturbed switchover: When the slave circuit is not in cascade mode (automatic mode of the external reference), the master circuit switches to the tracking mode. The output value of the master circuit will tracks the process value of the slave circuit, allowing for a smooth transition back to the cascade mode.

Priority (Error response > Interlock > Manual > Master tracking mode triggered by slave > Automatic)

■ Description

1. First, select the physical object to control GeneralConfig.PhysicalQuantity. The options include General and Temperature.

2: General, which is mainly for pressure and flow control. During general control, IntegralDivision and IntegralDivLimit in advanced configuration can be set for integral separation.

3: Temperature, for which various optimization is carried out.

2. Enable the auto-tuning function at room temperature. When auto-tuning is completed, xStartAT is au-

tomatically set to FALSE, and you can obtain the set Kp, Ti and Td values. Let the program continue to run and check whether the controlled object can reach steady state.

3. If the current usage needs are met after the controlled object reaches the steady state, record the values of Kp, Ti, and Td, and write these values into the program as defaults.

You can manually adjust the Kp, Ti, and Td values to meet your usage needs.

4. The larger the auto-tuning coefficient rATCoefficient, the faster the set temperature can be reached, but it may cause greater overshoot. The smaller the auto-tuning coefficient, the slower the set temperature can be reached, and the temperature curve will be very smooth, without overshoot or with very small overshoot. Unless otherwise required, use the default 0.5. If the customer requires to reach the set temperature faster, increase the auto-tuning coefficient rATCoefficient for another tuning.

If the customer needs a smooth curve and small overshoot, decrease the auto-tuning coefficient rATCoefficient for another tuning.

5. With the interlock function, when the InlockSW interlock switch is triggered, the interlock security value InlockVal is output.

6. The output priority is error output > interlock output > manual output > automatic output.

Note:

1. Before running the PID, check whether the parameters are correct.

2. It is recommended that the auto-tuning process starts at the room temperature.

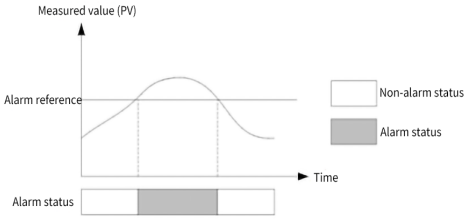
3. If the common temperature range for the customer is 150° C to 200° C, tuning can be carried out at 175° C. It is important to select a common temperature for more accurate tuning parameter.

4. In general, auto-tuning can be carried out first to obtain the tuned PID parameters. Then, you can adjust the PID parameters as needed based on the control effect.

■ Upper and lower limit alarms

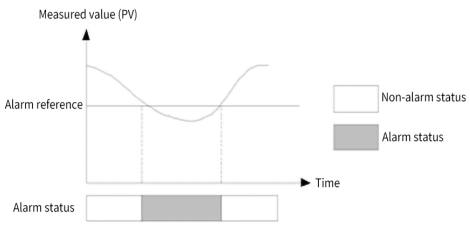
1. Upper limit alarm of the input value PV

When the measured value PV (of the PV value converted from the analog) is greater than the upper limit, an alarm is reported.



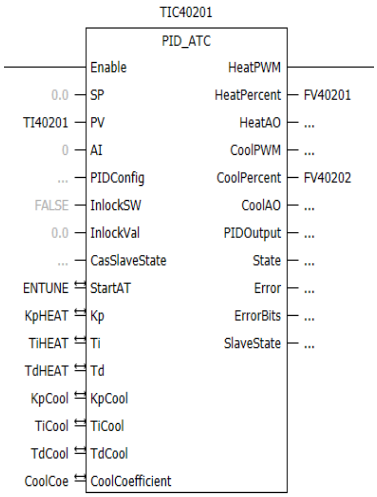
2. Lower limit alarm of the input value PV

When the measured value PV (of the PV value converted from the analog) is less than the lower limit, an alarm is reported.



■ Application examples

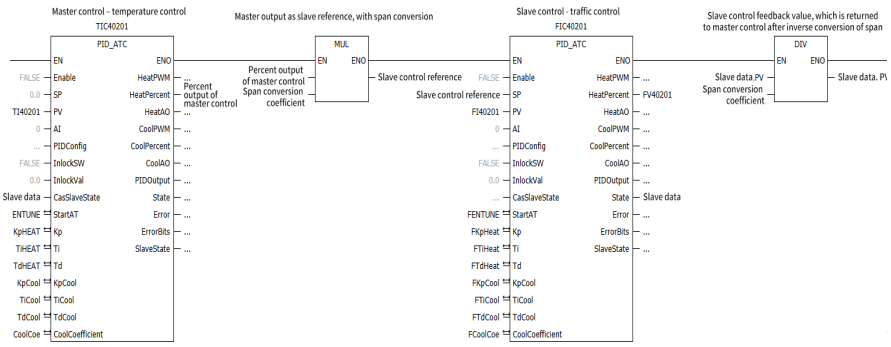
Control scheme configuration



Examples

No.	Example	Description
1	TIC40201	Temperature controller
2	TI40201	Temperature TC detection signal
3	FV40201	Hot water flow control valve
4	FV40202	Cold water flow control valve

Cascade control scheme configuration:



Examples

No.	Example	Description
1	TIC40201	Temperature controller
2	TI40201	Temperature TC detection signal
3	FIC40201	Flow controller
4	LV40201	Liquid level control valve
5	FI40201	Flow AI detection signal

■ Error codes (ErrorID)

For details, see the error codes (ErrorID) in 4.7.5 PID_AT.

■ Error handling

PIDConfig.InputConfig.ErrorAction: Action upon an error alarm

1. When PIDConfig.InputConfig.ErrorAction is FALSE, the input remains unchanged upon an error alarm. When the error is recovered, the current input is recalculated.

2. When PIDConfig.InputConfig.ErrorAction is TRUE, the alternative value of ErrorOut is output upon an error alarm. When the error is recovered, the current input is recalculated.

4. 8 BCD Conversion Instructions

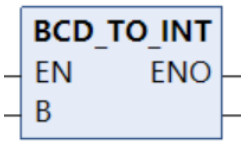
4.8. 1 Instruction List

Instruction Category	Name	FB/FC	Function
BCD conversion instructions	BCD_TO_INT	FC	BCD to signed integer
	INT_TO_BCD	FC	Signed integer to BCD
	BCD_TO_BYTE	FC	BCD to BYTE
	BYTE_TO_BCD	FC	BYTE to BCD
	BCD_TO_WORD	FC	BCD to word
	WORD_TO_BCD	FC	Word to BCD
	BCD_TO_DWORD	FC	BCD to double word
	DWORD_TO_BCD	FC	Double word to BCD

4.8. 2 BCD_TO_INT

This instruction converts BCD of the source data to integers.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
BCD_TO_INT	BCD to signed integer	FC		BCD_TO_INT(B:=);

■ Relevant variable BCD_TO_INT:

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
B	Input BCD	BYTE	0 to 255	0	Binary format (or corresponding decimal and hexadecimal format) of the input BCD

Output variables

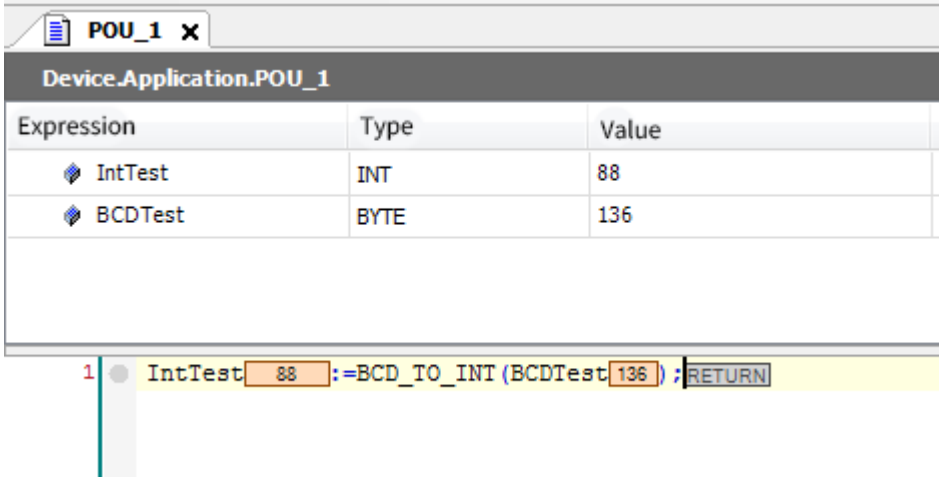
Output Variable	Name	Data Type	Value Range	Initial Value	Description
OUT	Actual value of input BCD	INT	0 to 65535	0	If the input is not BCD, the output is –1.

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
B	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
OUT	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-

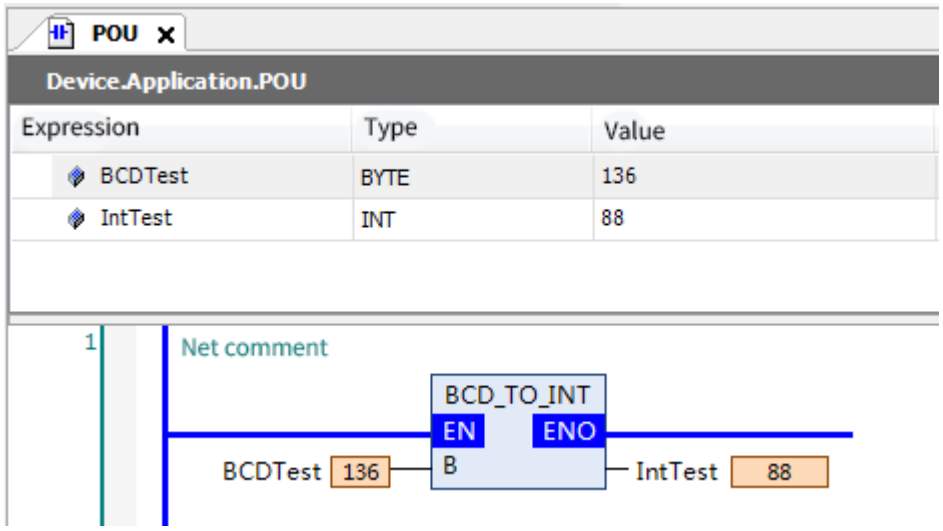
■ Program example

This instruction converts BCD of the source data to integers.

ST



LD



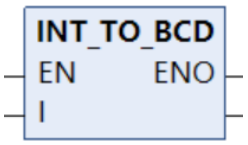
■ Precautions

This instruction can only convert data between 0 and 99. When the data to convert is not in the range, the conversion result is 255.

4.8. 3 INT_TO_BCD

This instruction converts integers of the source data to BCD.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
INT_TO_BCD	Signed integer to BCD	FC		INT_TO_BCD(I:=);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
I	Input data of INT type	INT	0 to 99	0	If the integer is 39, 39 is input.

Output variables

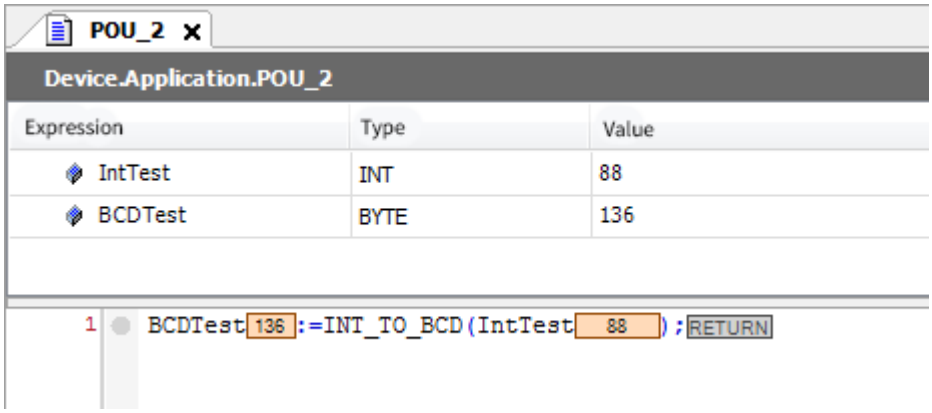
Output Variable	Name	Data Type	Value Range	Initial Value	Description
OUT	BCD after conversion	BYTE	0 to 255	0	If the integer 39 is converted to BCD 2#00101001, the output is 2#00101001 (10#57 or 16#39).

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
I	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-
OUT	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

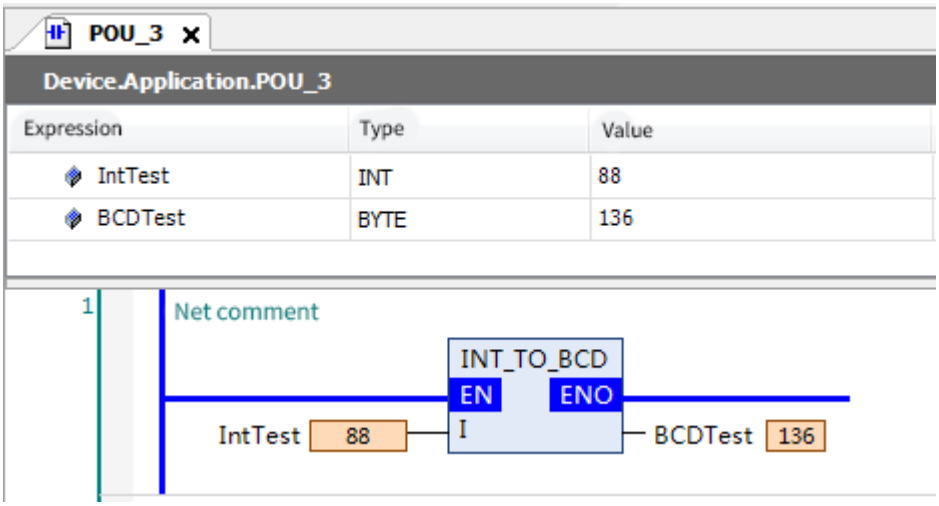
■ Program example

This instruction converts integers of the source data to BCD.

ST



LD



■ Precautions

This instruction can only convert data between 0 and 99. When the data to convert is not in the range, the conversion result is 255.

4.8. 4 BCD_TO_BYTE

This instruction converts BCD of the source data to bytes.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
BCD_TO_BYTE	BCD to BYTE	FC	BCD_TO_BYTE EN ENO B	BCD_TO_BYTE(B:=);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
B	Input data	BYTE	0 to 255	0	-

Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
OUT	Output data	BYTE	0 to 255	0	-

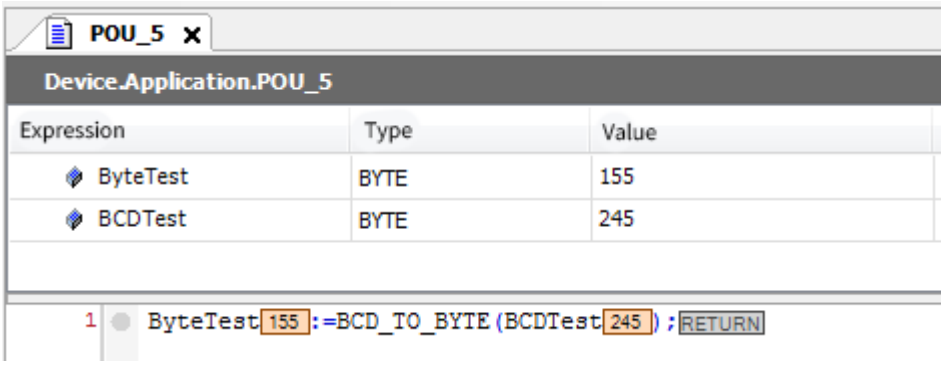
	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
B	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
OUT	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

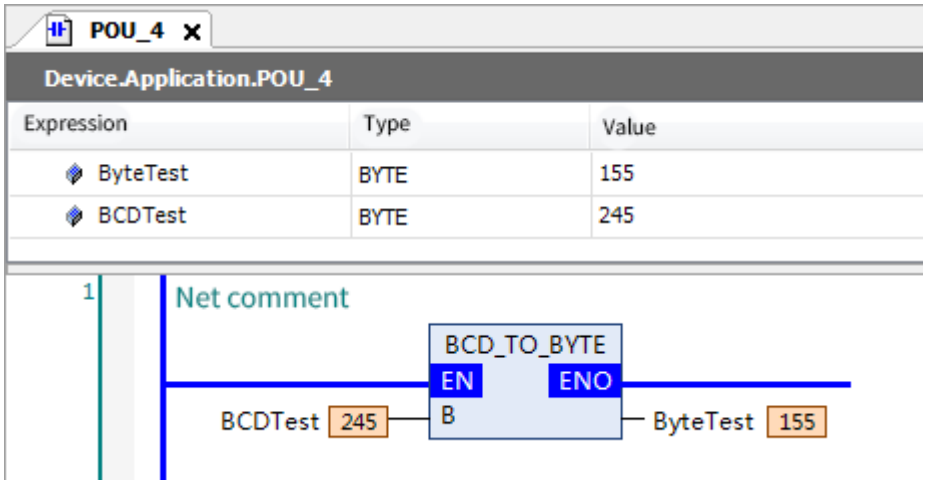
■ Program example

This instruction converts BCD of the source data to bytes.

ST



LD



■ Precautions

The BCD_TO_BYTE instruction can implement carry. For example, if BCD is set to 2#11110101, where the high 4 bits 2#1111 represent 15 and the low 4 bits 2#0101 represent 5, it indicates 155.

4.8. 5 BYTE_TO_BCD

This instruction converts bytes of the source data to BCD.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
BYTE_TO_BCD	BYTE to BCD	FC	BYTE_TO_BCD EN ENO B	BYTE_TO_BCD(B:=);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
B	Input data	BYTE	0 to 255	0	-

Output variables

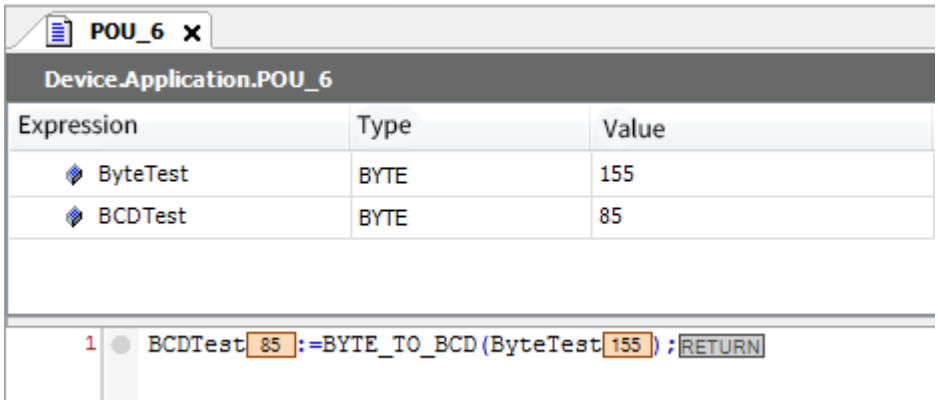
Output Variable	Name	Data Type	Value Range	Initial Value	Description
OUT	Output data	BYTE	0 to 255	0	-

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
B	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
OUT	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

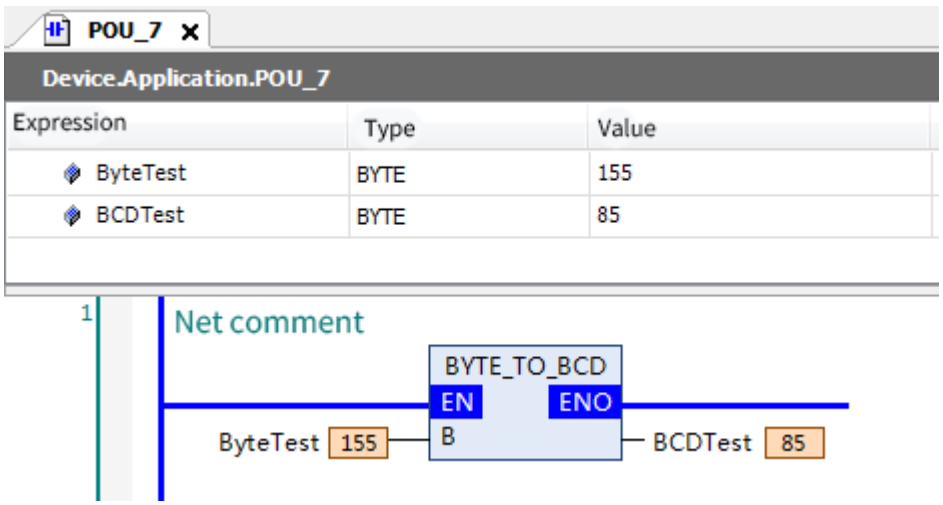
■ Program example

This instruction converts bytes of the source data to BCD.

ST



LD



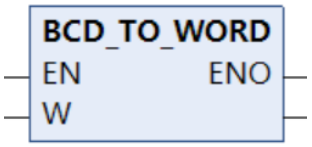
■ Precautions

The BYTE_TO_BCD instruction only converts the tens and ones digits.

4.8. 6 BCD_TO_WORD

This instruction converts BCD of the source data to words.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
BCD_TO_WORD	BCD to word	FC		BCD_TO_WORD(W:=);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
OUT	Output data	WORD	0-FFFF	0	-

Output variables

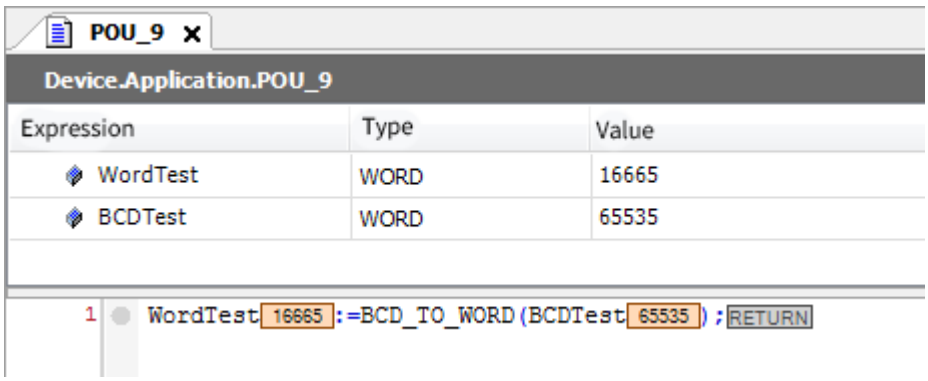
Output Variable	Name	Data Type	Value Range	Initial Value	Description
OUT	Output data	WORD	0-FFFF	0	-

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
W	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
OUT	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

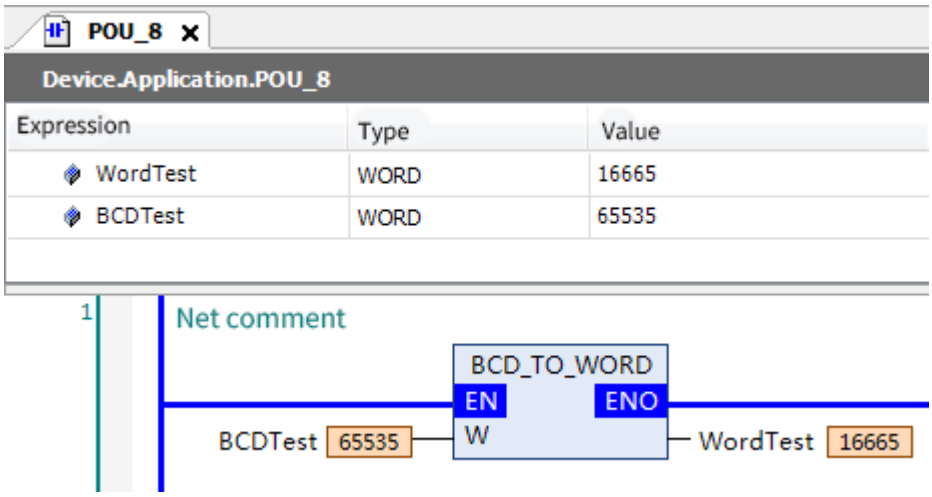
■ Program example

This instruction converts BCD of the source data to words.

ST



LD



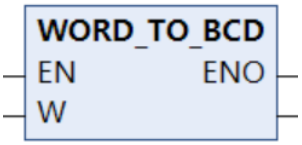
■ Precautions

The BCD_TO_WORD instruction can implement carry. For example, if BCD is set to 2#1111111111111111, where the high bytes 2#11111111 represent 165 and the low bytes 2#1111111 represent 165, it indicates 16665.

4.8. 7 WORD_TO_BCD

This instruction converts words of the source data to BCD.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
WORD_TO_BCD	Word to BCD	FC		WORD_TO_BCD(W:=);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
W	Input data	WORD	0-FFFF	0	-

■ Output variables

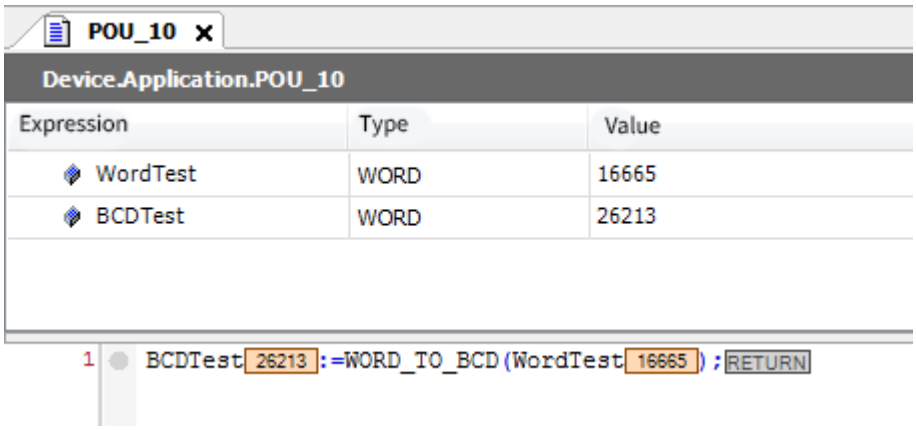
Output Variable	Name	Data Type	Value Range	Initial Value	Description
OUT	Output data	WORD	0-FFFF	0	-

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
W	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
OUT	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

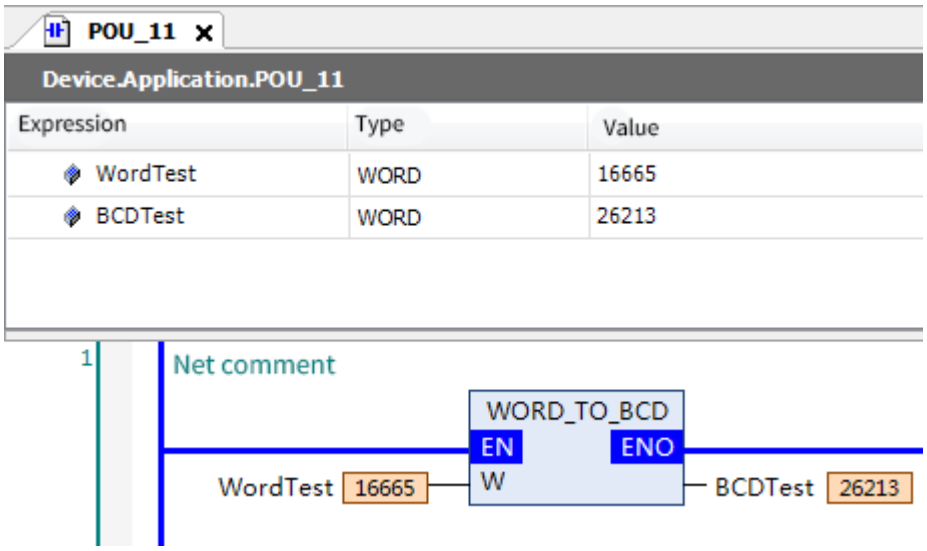
■ Program example

This instruction converts words of the source data to BCD.

ST



LD



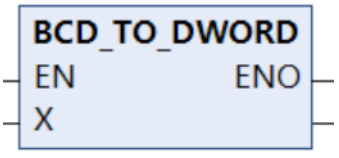
■ Precautions

The WORD_TO_BCD instruction only converts the thousands, hundreds, tens, and ones digits.

4.8. 8 BCD_TO_DWORD

This instruction converts BCD of the source data to doublewords.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
BCD_TO_DWORD	BCD to word	FC		BCD_TO_DWORD(X:=);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
X	Input data	DWORD	0 to FFFF FFFF	0	-

Output variables

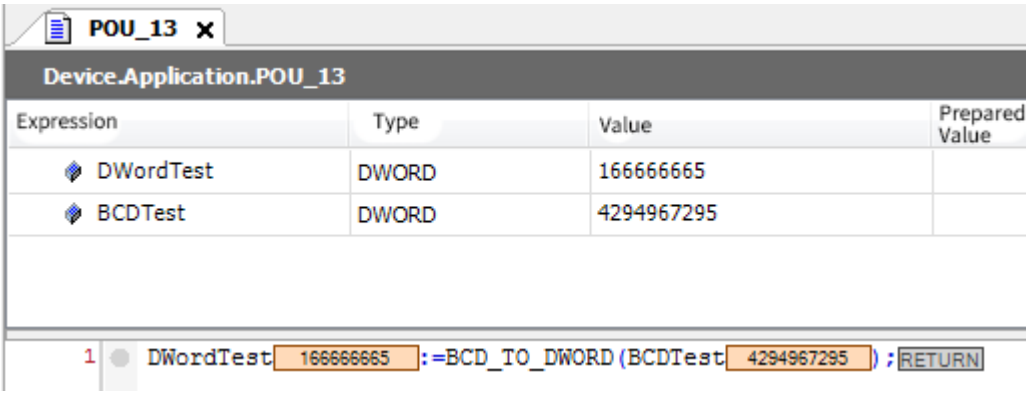
Output Variable	Name	Data Type	Value Range	Initial Value	Description
OUT	Output data	DWORD	0 to FFFF FFFF	0	-

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
X	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
OUT	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

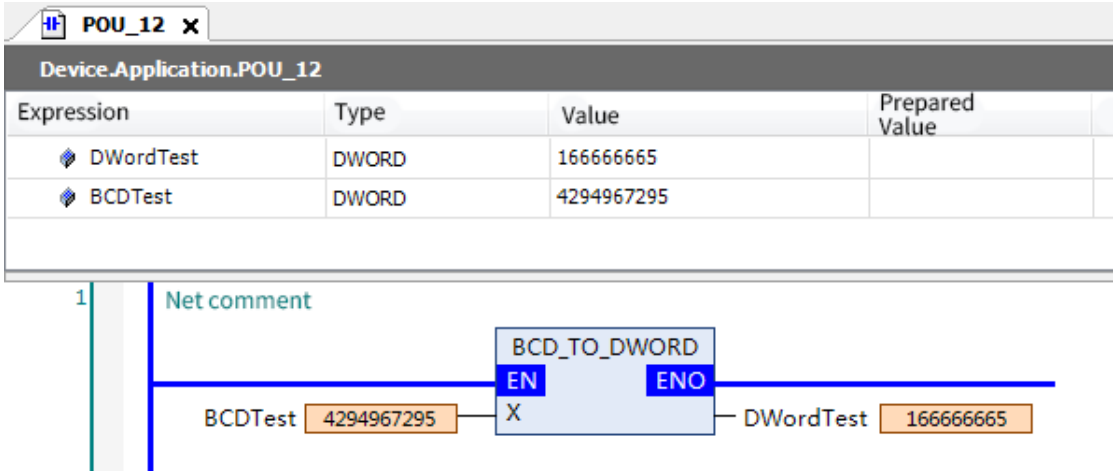
■ Program example

This instruction converts BCD of the source data to doublewords.

ST



LD



■ Precautions

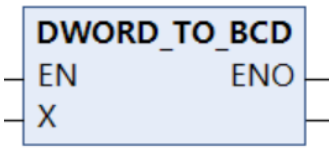
The BCD_TO_DWORD instruction can implement carry. For example, if BCD is set to 2#1111111111111111111111111111111, where the high words 2#1111111111111111 represent 16665 and the low words

2#11111111111111 represent 16665, it indicates 166666665.

4.8. 9 DWORD_TO_BCD

This instruction converts doublewords of the source data to BCD.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
DWORD_TO_BCD	Double word to BCD	FC		DWORD_TO_BCD(X:=);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
X	Input data	DWORD	0 to FFFF FFFF	0	-

Output variables

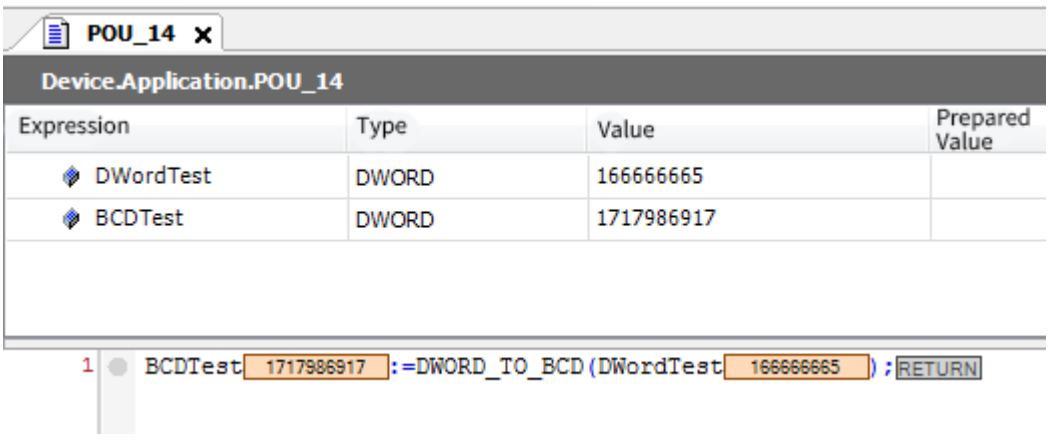
Output Variable	Name	Data Type	Value Range	Initial Value	Description
OUT	Output data	DWORD	0 to FFFF FFFF	0	-

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
X	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
OUT	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

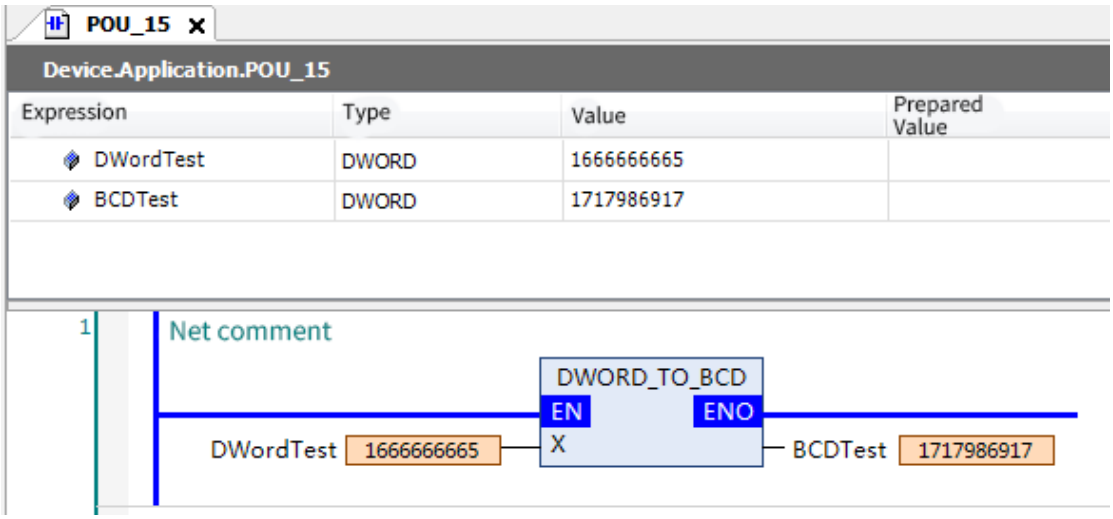
■ Program example

This instruction converts doublewords of the source data to BCD.

ST



LD



■ Precautions

The DWORD_TO_BCD instruction only converts to the thousands digit.

4. 9 Filter Instructions

4.9. 1 Instruction List

Instruction Category	Name	FB/FC	Function
Filter instructions	LimitingFilter	FB	Limiting filter
	MedianFilter	FB	Median value filtering
	ArithmeticAverageFilter	FB	One-dimensional arithmetic mean filtering
	RecursiveAverageFilter	FB	Recursive averaging filter
	MedianAverageFilter	FB	Median average filtering
	LimitingAverageFilter	FB	Limiting average filtering
	FirstOrderLagFilter	FB	First order lag filtering
	WeightRecursiveAverageFilter	FB	Weighted recursive average filtering
	DebounceFilter	FB	Debounce filter
	LimitingDebounceFilter	FB	Limit debounce filter

4.9. 2 LimitingFilter

When xEnable is set to TRUE, the sampling variable fSample is input based on the given scan period count uiSampleCycle, reference value fReference, and maximum allowable deviation fDeviation for two consecutive samples. Each time a new value is sampled, the following conditions are checked:

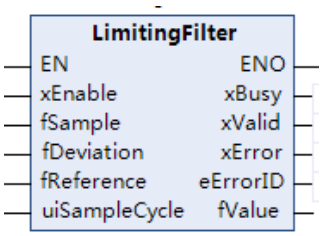
If the difference between the current value and the previous value is less than or equal to the value of fDeviation, the current value is valid and is output as the filter value.

If the difference between the current value and the previous value is greater than the value of fDeviation, the current value is invalid. In this case, the current value is discarded, and the previous value is output as the filter value.

Advantages: It effectively overcomes pulse interference caused by accidental factors.

Disadvantages: It is unable to suppress periodic interference, causing poor smoothness.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
LimitingFilter	Limiting filter	FB		LimitingFilter(xEnable:= , fSample:= , fDeviation:= , fReference:= , uiSampleCycle:= , xBusy=> , xValid=> , xError=> , eErrorID=> , fValue=>);

■ Variables

Input variables

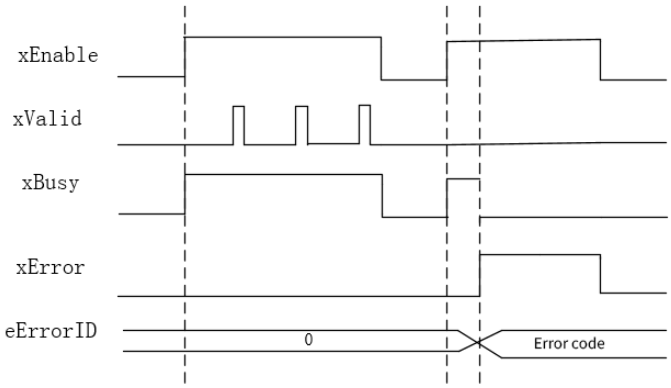
Input Variable	Name	Data Type	Value Range	Initial Value	Description
xEnable	Filter enable	BOOL	[TRUE, FALSE]	FALSE	TRUE: Execute the function block FALSE: Not execute the function block
fSample	Input sampling value	REAL	-	0	Input sampling value
fDeviation	Maximum allowable deviation for two consecutive samples	REAL	-	0	Maximum allowable deviation for two consecutive samples
fReference	Input reference value	REAL	-	0	Input reference value
uiSampleCycle	Input sampling period	UINT	1 to 1000	0	Input sampling period

Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
xBusy	Executing the instruction	BOOL	[TRUE, FALSE]	FALSE	TRUE: The function block is being executed
xValid	Output active	BOOL	[TRUE, FALSE]	FALSE	TRUE: Instruction execution is valid
xError	Error	BOOL	[TRUE, FALSE]	FALSE	TRUE: An error occurs
eErrorID	Error ID	UTIL_ERROR	See UTIL_ERROR.	0	Error ID For detail, see UTIL_ERROR.
fValue	Filter output effective value	REAL	0	0	Filter output effective value

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
xEnable	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
fSample	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-
fDeviation	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-
fReference	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-
uiSample-Cycle	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-
xBusy	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
xValid	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
xError	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
eErrorID	UTIL_ERROR																			
fValue	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-

■ Timing Diagram



■ Program example

The sampling type is consecutive sampling. The raw data for sampling is sourced from the consecutively updated program variables.

The amplitude reference value is 10.0 (user-defined reference value of the floating-point type).

The maximum allowable deviation in amplitude is 1.5 (deviation range of the floating-point type).

The filter calculation period is 1 (the sampling filter is executed once every given number of running period, with the default value of 1).

When the input xFilter is set as energy flow effective, the limiting filter function block starts the sampling filter. Set the initial reference value to 10.0 as the effective value. Compare the sampling value with the previous effective value. If the absolute difference between them is greater than the set maximum deviation value of 1.5, use the previous effective value as the current effective value. If the absolute difference is less than or equal to the maximum deviation value, use the current value as the effective value. Then output the effective value as the filter value, and set the filter execution valid flag xValid to TRUE.

ST

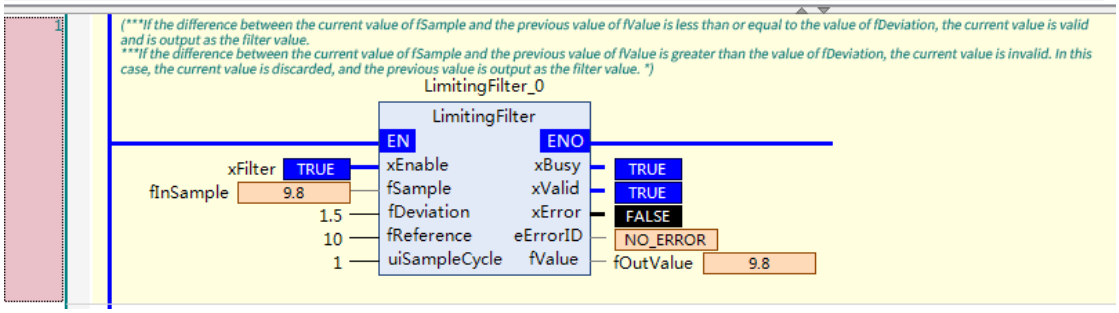
Expression	Type	Value	Prepared Value	Address	Comment
LimitingFilter_0	LimitingFilter				Limiting filter instantiation
xFilter	BOOL	TRUE			Filter enable bit
fInSample	REAL	8			Input value
fOutValue	REAL	10			Output value

```
(***If the difference between the current value of fSample and the previous value of fValue is less than or equal to the value of fDeviation, the current value is valid and is output as the filter value.
***If the difference between the current value of fSample and the previous value of fValue is greater than the value of fDeviation, the current value is invalid. In this case, the current value is
discarded, and the previous value is output as the filter value. *)
LimitingFilter_0(
    xEnable:=xFilter ,      // Enable condition
    fSample:= fInSample ,   // Input sampling value
    fDeviation:=1.5 ,       // Maximum allowable deviation for two consecutive samples
    fReference:=10 ,        // Sampling reference value
    uiSampleCycle:=1 ,      // Number of scan periods
    xBusy=> ,
    xValid=> ,              // Filter execution valid flag, xValid
    xError=> ,
    eErrorID=> ,
    fValue=>fOutValue );    // Filter output value

RETURN
```

LD

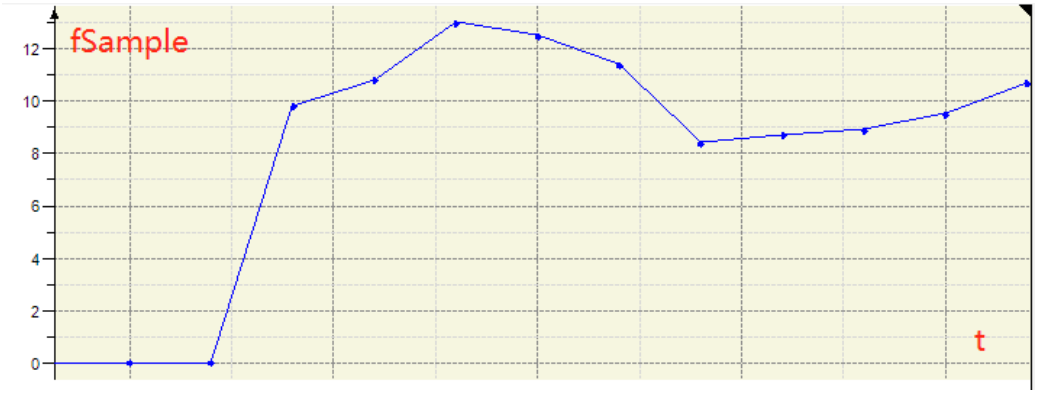
Expression	Type	Value	Prepared Value	Address	Comment
LimitingFilter_0	LimitingFilter				Limiting filter instantiation
xFilter	BOOL	TRUE			Filter enable bit
fInSample	REAL	9.8			Input value
fOutValue	REAL	9.8			Output value



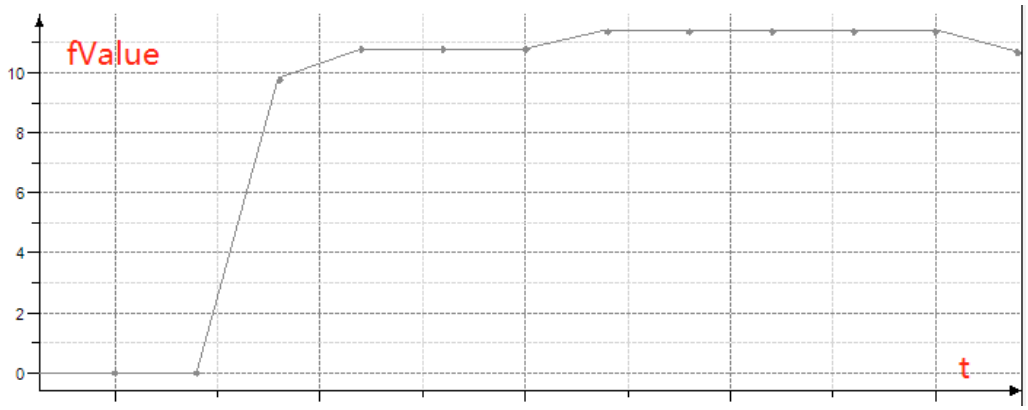
The sampling values of 10 consecutive periods are respectively 9.8, 10.8, 13.0, 12.5, 11.4, 8.4, 8.7, 8.9, 9.5, and 10.7.

After filtering, the values are respectively 9.8, 10.8, 10.8, 10.8, 11.4, 11.4, 11.4, 11.4, 11.4, and 10.7.

The following figure shows the curve before filtering.



The following figure shows the curve after filtering.



■ Precautions

1) In the filter function block, the input parameter uiSampleCycle indicates the sampling period. When the value of uiSampleCycle is 1, sampling is performed every period. When the value of uiSampleCycle is 3, sampling is performed once every three periods. If the output of the function block flag LimitingFilter.xValid is TRUE, the filter is valid. The filter output value of fValue is updated when the filter output is valid.

2) If the sampled data is stored in an array, the data needs to be put in the filter function block one by one and handled by the user.

3) For details about errors, see UTIL_ERROR.

4.9. 3 MedianFilter

When xEnable is set to TRUE, the sampling variable fValue is input based on the given scan period count uiSampleCycle and sampling count uiSampleNum. Sample data for consecutively uiSampleNum times, sort the sampled data by size, and take the middle value as the filter value for this period.

Advantages: It effectively overcomes fluctuation interference and pulse interference caused by accidental factors, and has good filtering effect on slowly changing parameters such as the temperature and liquid level.

Disadvantages: It is not suitable for rapidly changing parameters such as the flow rate and speed.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
MedianFilter	Median value filtering	FB	MedianFilter EN ENO xEnable xBusy fSample xValid uiSampleNum xError uiSampleCycle eErrorID fValue	MedianFilter(xEnable:= , fSample:= , uiSampleNum:= , uiSampleCycle:= , xBusy=> , xValid=> , xError=> , eErrorID=> , fValue=>);

■ Variables

Input variables

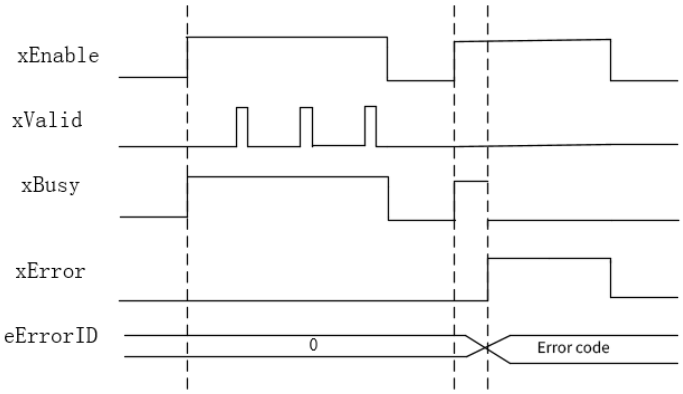
Input Variable	Name	Data Type	Value Range	Initial Value	Description
xEnable	Filter enable	BOOL	[TRUE, FALSE]	FALSE	TRUE: Execute the function block FALSE: Not execute the function block
fSample	Input sampling value	REAL	-	0	Input sampling value
uiSampleNum	Input sampling count N	UINT	1 to 1000	0	Input sampling count N
uiSampleCycle	Input sampling period	UINT	1 to 1000	0	Input sampling period

Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
xBusy	Executing the instruction	BOOL	[TRUE, FALSE]	FALSE	TRUE: The function block is being executed
xValid	Output active	BOOL	[TRUE, FALSE]	FALSE	TRUE: Instruction execution is valid
xError	Error	BOOL	[TRUE, FALSE]	FALSE	TRUE: An error occurs
eErrorID	Error ID	UTIL_ERROR	See UTIL_ERROR.	0	Error ID For detail, see UTIL_ERROR.
fValue	Filter output effective value	REAL	0	0	Filter output effective value

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
xEnable	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
fSample	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-
uiSampleNum	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-
uiSampleCycle	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-
xBusy	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
xValid	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
xError	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
eErrorID	UTIL_ERROR																			
fValue	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-

■ Timing Diagram



■ Program example

The sampling type is consecutive sampling. The raw data for sampling is sourced from the consecutively updated program variables.

The number of samples for the sampling calculation is 3 (unsigned integer ranging from 1 to 3000).

The sampling period is 1 (data sampling is executed once every given number of running periods, with the default value of 1).

When the input value of xFilter is TRUE, the function block starts sampling and filtering with energy flow effective. Set the number of samples for the sampling calculation to 3, indicating that filter calculation is performed once for every three sampling values. Sort the three sampling values by numerical value, output the middle value as the filter value, and set the filter execution valid flag xValid to TRUE.

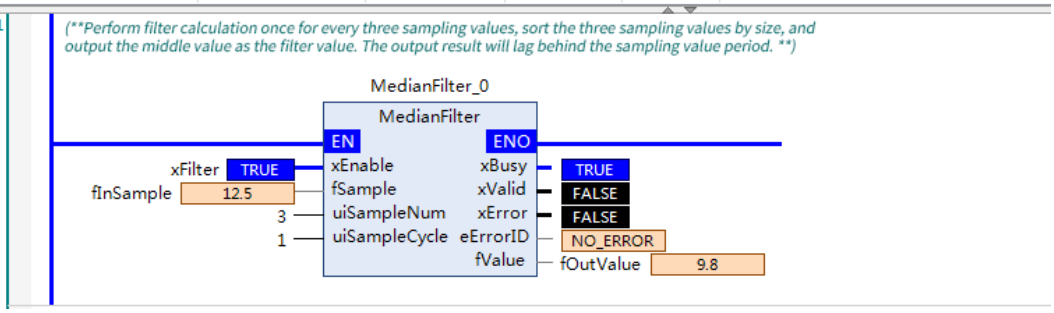
ST

Expression	Type	Value	Prepared Value	Address	Comment
MedianFilter_0	MedianFilter				Median value filtering instantiation
xFilter	BOOL	TRUE			Filter enable bit
fInSample	REAL	13			Input value

```
2     The output result will lag behind the sampling value period.
3   MedianFilter_0(
4       xEnable:=xFilter ,
5       fSample:=fInSample ,
6       uiSampleNum:= 3,
7       uiSampleCycle:=1 ,
8       xBusy=> ,
9       xValid=> ,
10      xError=> ,
11      eErrorID=> ,
12      fValue=> fOutValue );
13      RETURN
```

LD

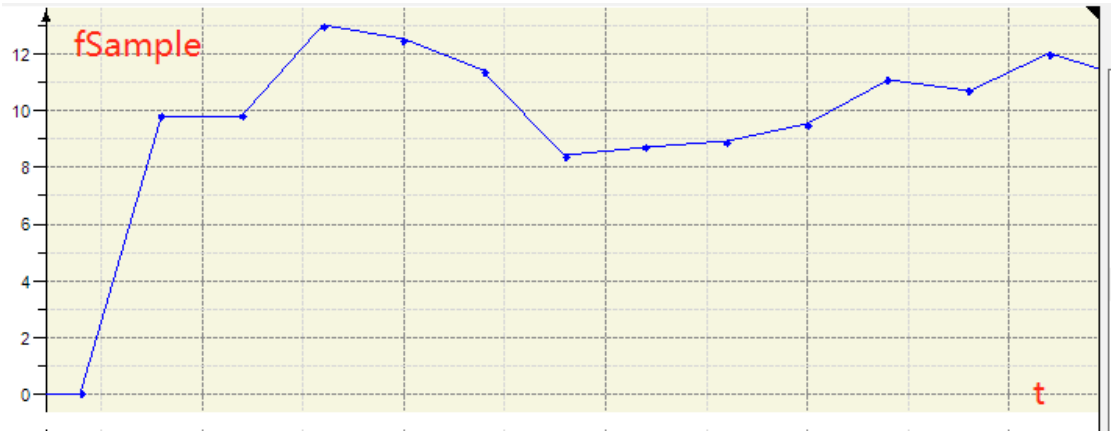
Expression	Type	Value	Prepared Value	Address	Comment
MedianFilter_0	MedianFilter				Median value filtering instantiation
xFilter	BOOL	TRUE			Filter enable bit
fInSample	REAL	12.5			Input value



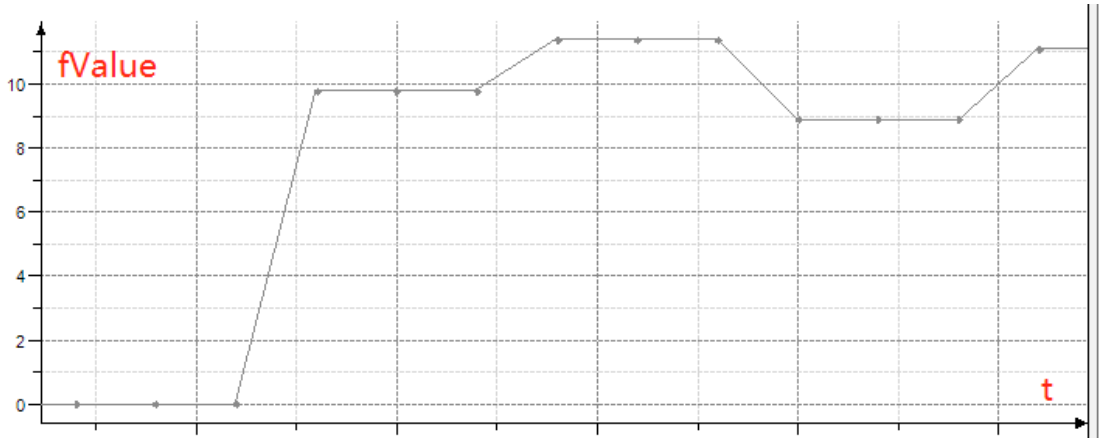
The sampling values of 12 consecutive periods are respectively 9.8, 9.8, 13.0, 12.5, 11.4, 8.4, 8.7, 8.9, 9.5, 11.1, 10.7, and 12.0.

After filtering, the values are respectively 0, 0, 9.8, 9.8, 9.8, 11.4, 11.4, 11.4, 8.9, 8.9, 8.9, and 11.1.

The following figure shows the curve before filtering.



The following figure shows the curve after filtering.



■ Precautions

1) In the filter function block, the input parameter uiSampleCycle indicates the sampling period. When the value of uiSampleCycle is 1, sampling is performed every period. When the value of uiSampleCycle is 3, sampling is performed once every three periods. If the output of MedianFilter.xValid is TRUE, the filter is valid. The filter output value of fValue is updated when the filter output is valid.

2) If the sampled data is stored in an array, the data needs to be put in the filter function block one by one and handled by the user.

3) For details about errors, see UTIL_ERROR.

4.9. 4 ArithmeticAverageFilter

When xEnable is set to TRUE, the sampling variable fValue is input based on the given scan period count uiSampleCycle and sampling count uiSampleNum. Sample data for consecutively uiSampleNum times and average the sampled data.

When the value of uiSampleNum is larger, the signal smoothness is higher but the sensitivity is lower.

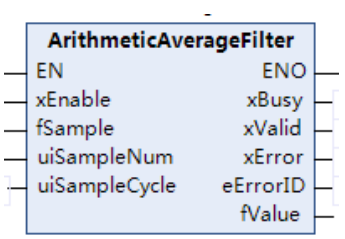
When the value of uiSampleNum is smaller, the signal smoothness is lower but the sensitivity is higher.

For the general flow, set uiSampleNum to 12. For the pressure, set uiSampleNum to 4.

Advantages: It is suitable for filtering signals with general random interference. Such signals have an average value and fluctuate around a certain range.

Disadvantages: It is not suitable for real-time control with a slow measurement speed or demanding a fast data calculation speed, and it may consume RAM unnecessarily.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
ArithmeticAverageFilter	One-dimensional arithmetic mean filtering	FB		ArithmeticAverageFilter(xEnable:= , fSample:= , uiSampleNum:= , uiSampleCycle:= , xBusy=> , xValid=> , xError=> , eErrorID=> , fValue=>);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
xEnable	Filter enable	BOOL	[TRUE, FALSE]	FALSE	TRUE: Execute the function block FALSE: Not execute the function block
fSample	Input sampling value	REAL	-	0	Input sampling value
uiSampleNum	Input sampling count N	UINT	1 to 1000	0	Input sampling count N
uiSampleCycle	Input sampling period	UINT	1 to 1000	0	Input sampling period

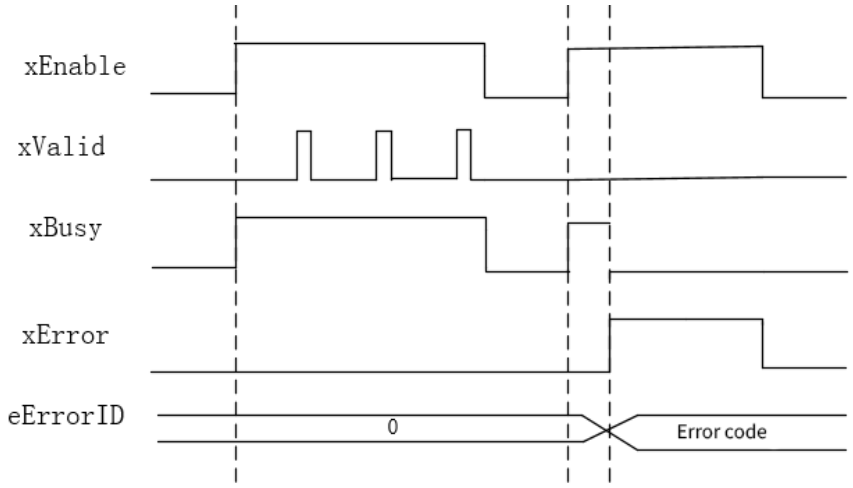
Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
xBusy	Executing the instruction	BOOL	[TRUE, FALSE]	FALSE	TRUE: The function block is being executed
xValid	Output active	BOOL	[TRUE, FALSE]	FALSE	TRUE: Instruction execution is valid
xError	Error	BOOL	[TRUE, FALSE]	FALSE	TRUE: An error occurs
eErrorID	Error ID	UTIL_ERROR	See UTIL_ERROR.	0	Error ID For detail, see UTIL_ERROR.
fValue	Filter output effective value	REAL	0	0	Filter output effective value

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
xEnable	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
fSample	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-
uiSampleNum	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-
uiSampleCycle	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-
xBusy	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
xValid	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
xError	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
eErrorID	UTIL_ERROR																			
fValue	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-

■ Timing Diagram



■ Program example

The sampling type is consecutive sampling. The raw data for sampling is sourced from the consecutively updated program variables.

The number of samples for the sampling calculation is 3 (unsigned integer ranging from 1 to 3000).

The sampling period is 1 (data sampling is executed once every given number of running periods, with the default value of 1).

When the input value of xFilter is TRUE, the function block starts sampling and filtering with energy flow effective. Set the number of samples for the sampling calculation to 3, indicating that filter calculation is performed once for every three sampling values. Average the three sampling values, output the result as the filter value, and set the filter execution valid flag xValid to TRUE.

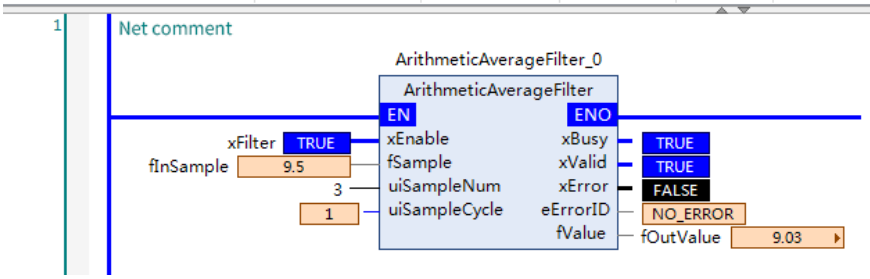
ST

Expression	Type	Value	Prepared Value	Address	Comment
ArithmeticAverageFilter_0	ArithmeticAverageFil...				
xFilter	BOOL	TRUE			Filter enable bit
fInSample	REAL	13			Input value

```
ArithmeticAverageFilter_0(
    xEnable TRUE := xFilter TRUE,
    fSample 13 := fInSample 13,
    uiSampleNum 3 :=3 ,
    uiSampleCycle 1 :=1 ,
    xBusy=> ,
    xValid=> ,
    xError=> ,
    eErrorID=> ,
    fValue 10.9 => fOutValue 10.9 );RETURN
```

LD

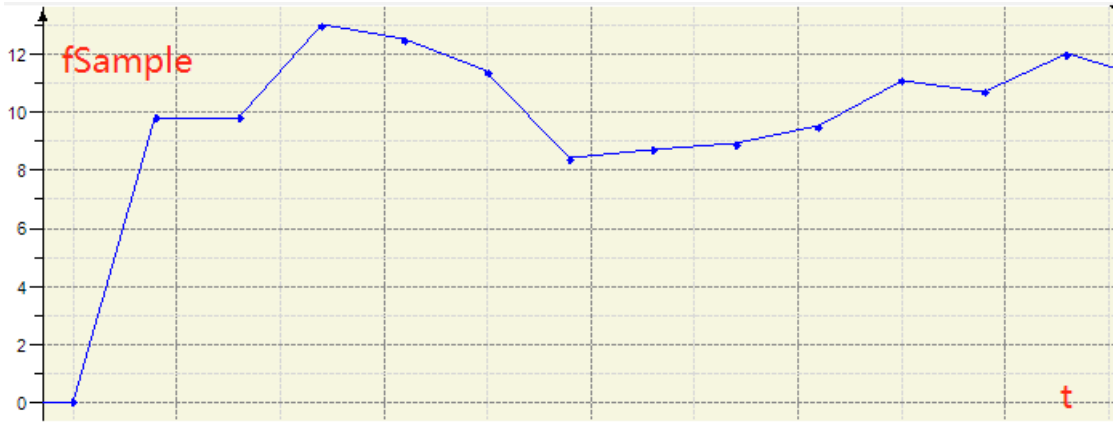
Expression	Type	Value	Prepared Value	Address	Comment
ArithmeticAverageFilter_0	ArithmeticAverageFil...				
xFilter	BOOL	TRUE			Filter enable bit
fInSample	REAL	9.5			Input value



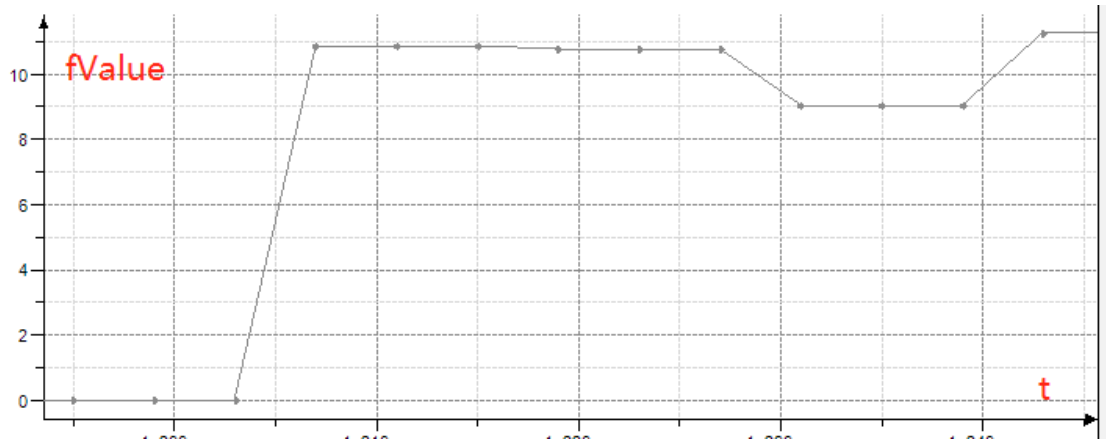
The sampling values of 12 consecutive periods are respectively 9.8, 9.8, 13.0, 12.5, 11.4, 8.4, 8.7, 8.9, 9.5, 11.1, 10.7, and 12.0.

After filtering, the values are respectively 0, 0, 10.87, 10.87, 10.87, 10.77, 10.77, 10.77, 9.03, 9.03, 9.03, and 11.27.

The following figure shows the curve before filtering.



The following figure shows the curve after filtering.



■ Precautions

1) In the filter function block, the input parameter uiSampleCycle indicates the sampling period. When the value of uiSampleCycle is 1, sampling is performed every period. When the value of uiSampleCycle is 3, sampling is performed once every three periods. If the output of the function block flag ArithmeticAverageFilter.xValid is TRUE, the filter is valid. The filter output value of fValue is updated when the filter output is valid.

2) If the sampled data is stored in an array, the data needs to be put in the filter function block one by one and handled by the user.

3) If the value of the input parameter uiSampleNum is not within the valid range, the xError flag is set to TRUE, and the function block generates an error code eErrorID indicating the error message "SAMPLENUM_PARAMETER_SET_ERROR". To handle the error, you need to reset the uiSampleNum parameter within the valid range and re-trigger a flow to reset the fault and the xError signal.

4) For details about errors, see UTIL_ERROR.

4.9. 5 RecursiveAverageFilter

When xEnable is set to TRUE, the sampling variable fSample is input based on the given scan period count uiSampleCycle and queue length uiQueueCount. Consider consecutively uiQueueCount samples as a queue with the fixed length of uiQueueCount. Place each newly sampled piece of data to the end of the queue, and discard the first piece of data in the queue, following the FIFO principle. Afterward, average N data samples in the queue for a new filter result.

When the value of uiQueueCount is larger, the signal smoothness is higher but the sensitivity is lower.

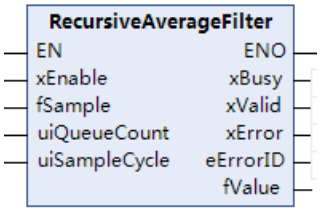
When the value of uiQueueCount is smaller, the signal smoothness is lower but the sensitivity is higher.

For the general flow, set uiQueueCount to 12. For the pressure, set uiQueueCount to 4. For the liquid level, set uiQueueCount to 4 to 12. For the temperature, set uiQueueCount to 4.

Advantages: It suppresses periodic interference well and provides high smoothness, and it is suitable for systems with high-frequency oscillations.

Disadvantages: It has low sensitivity and provides poor suppression for occasional pulse interference. It is not effective in eliminating sampling value deviations caused by pulse interference. It is not suitable for situations with significant pulse interference. It may consume RAM unnecessarily.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
RecursiveAverageFilter	Recursive averaging filter	FB		RecursiveAverageFilter(xEnable:= , fSample:= , uiQueueCount:= , uiSampleCycle:= , xBusy=> , xValid=> , xError=> , eErrorID=> , fValue=>);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
xEnable	Filter enable	BOOL	[TRUE, FALSE]	FALSE	TRUE: Execute the function block FALSE: Not execute the function block
fSample	Input sampling value	REAL	-	0	Input value to filter
uiQueueCount	Input queue length N	UINT	1 to 3000	0	Input queue length N
uiSampleCycle	Input sampling period	UINT	1 to 1000	0	Input sampling period

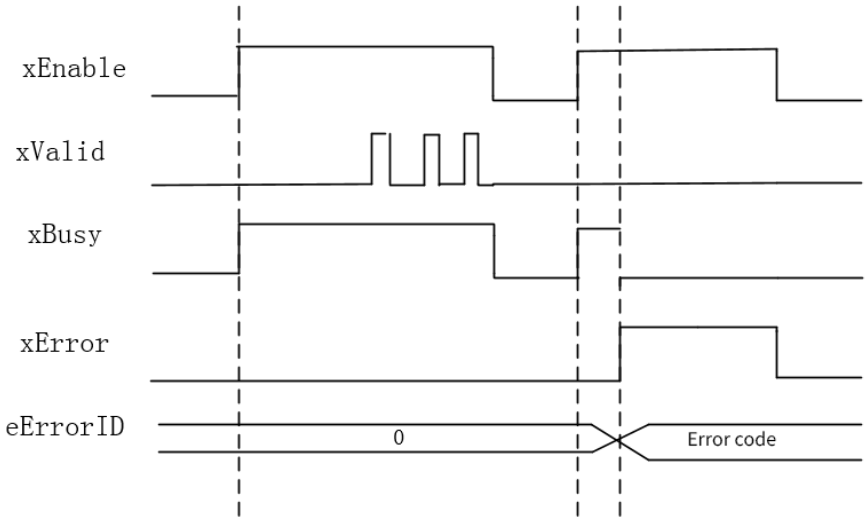
Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
xBusy	Executing the instruction	BOOL	[TRUE, FALSE]	FALSE	TRUE: The function block is being executed
xValid	Output active	BOOL	[TRUE, FALSE]	FALSE	TRUE: Instruction execution is valid
xError	Error	BOOL	[TRUE, FALSE]	FALSE	TRUE: An error occurs
eErrorID	Error ID	UTIL_ERROR	See UTIL_ERROR.	0	Error ID For detail, see UTIL_ERROR.
fValue	Filter output effective value	REAL	0	0	Filter output effective value

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
xEnable	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
fSample	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-
uiSampleNum	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-
uiSampleCycle	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
xBusy	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
xValid	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
xError	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
eErrorID	UTIL_ERROR																			
fValue	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-

■ Timing Diagram



■ Program example

1) The sampling type is consecutive sampling. The raw data for sampling is sourced from the consecutively updated program variables.

2) The number of samples for the sampling calculation is 3 (unsigned integer ranging from 1 to 3000).

3) The sampling period is 1 (data sampling is executed once every given number of running periods, with the default value of 1).

4) When the input value of xFilter is TRUE, the function block starts sampling and filtering with the energy flow effective. Set the number of sample queues for the sampling calculation to 3. Place each newly sampled piece of data to the end of the queue, and discard the first piece of data in the queue, following the FIFO principle. Average the three sampling values in the queue, output the result as the filter value, and set the filter execution valid flag xValid to TRUE.

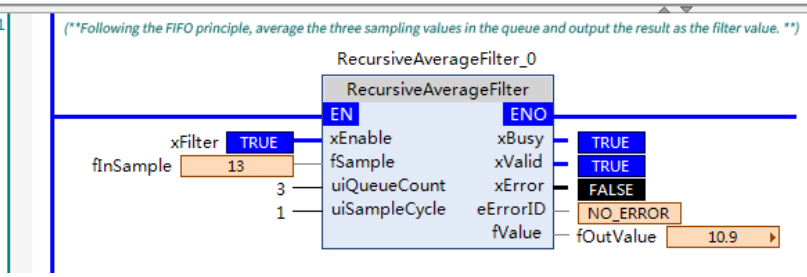
ST

Expression	Type	Value	Prepared Value	Address	Comment
RecursiveAverageFilter_0	RecursiveAverageFil...				
xFilter	BOOL	TRUE			Filter enable bit
fInSample	REAL	12.5			Input value

```
1  (**Following the FIFO principle, average the three sampling values in the queue and output the result as the filter value. **)
2  RecursiveAverageFilter_0(
3      xEnable TRUE :=xFilter TRUE ,
4      fSample 12.5 :=fInSample 12.5 ,
5      uiQueueCount 3 :=3 ,
6      uiSampleCycle 1 :=1 ,
7      xBusy=> ,
8      xValid=> ,
9      xError=> ,
10     eErrorID=> ,
11     fValue 11.8 => fOutValue 11.8 );RETURN
```

LD

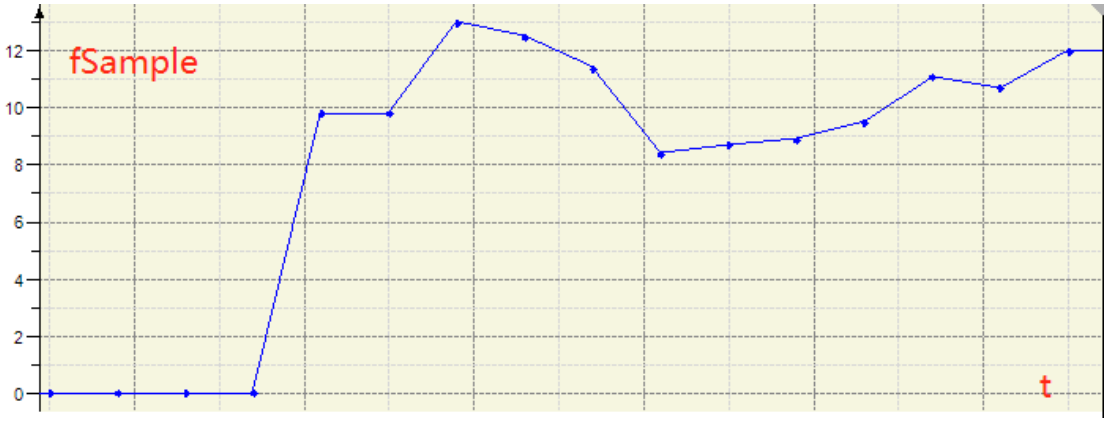
Expression	Type	Value	Prepared Value	Address	Comment
RecursiveAverageFilter_0	RecursiveAverageFil...				
xFilter	BOOL	TRUE			Filter enable bit
fInSample	REAL	13			Input value
fOutValue	REAL	10.8666658			Output value



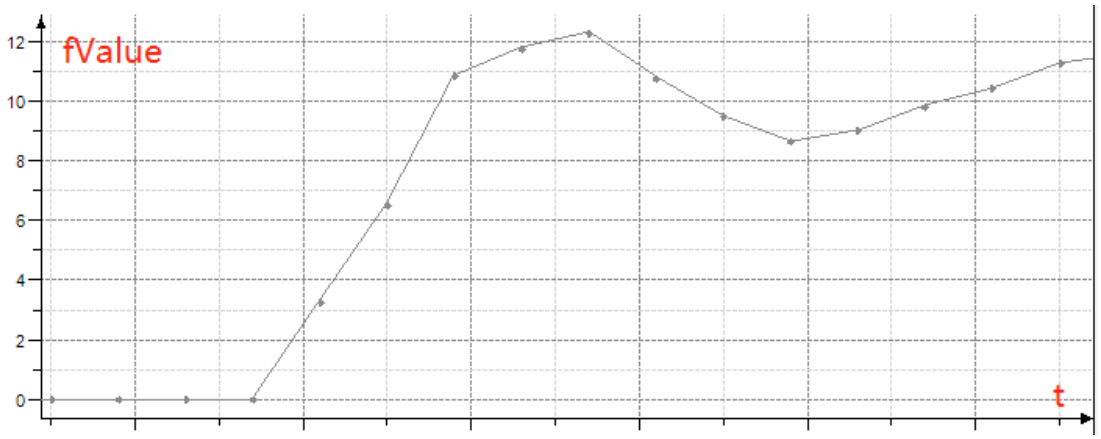
The sampling values of 12 consecutive periods are respectively 9.8, 9.8, 13.0, 12.5, 11.4, 8.4, 8.7, 8.9, 9.5, 11.1, 10.7, and 12.0.

After filtering, the values are respectively 0, 0, 10.87, 11.77, 12.30, 10.77, 9.50, 8.67, 9.03, 9.83, 10.43, and 11.27.

The following figure shows the curve before filtering.



The following figure shows the curve after filtering.



■ Precautions

1) In the filter function block, the input parameter uiSampleCycle indicates the sampling period. When the value of uiSampleCycle is 1, sampling is performed every period. When the value of uiSampleCycle is 3, sampling is performed once every three periods. If the output of RecursiveAverageFilter.xValid is TRUE, the filter is valid. The filter output value of fValue is updated when the filter output is valid.

2) If the sampled data is stored in an array, the data needs to be put in the filter function block one by one and handled by the user.

3) For details about errors, see UTIL_ERROR.

4.9. 6 MedianAverageFilter

When xEnable is set to TRUE, the sampling variable fSample is input based on the given scan period count uiSampleCycle and sampling count uiSampleNum. Sample data for consecutively uiSampleNum times, sort the sampled data, exclude the maximum value and minimum value, and average the rest values. This means using both median value filtering and one-dimensional arithmetic mean filtering. In general, uiSampleNum is set to 3 to 14.

Advantages: It combines the advantages of both median value filtering and one-dimensional arithmetic mean filtering. It can eliminate sampling value deviations caused by occasional pulse interference, suppress periodic interference well, and provide high smoothness. It is suitable for systems with high-frequency oscillations.

Disadvantages: It provides a slower calculation speed, similar to one-dimensional arithmetic mean filtering. It may consume RAM unnecessarily.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
MedianAverageFilter	Median average filtering	FB	MedianAverageFilter EN ENO xEnable xBusy fSample xValid uiSampleNum xError uiSampleCycle eErrorID fValue	MedianAverageFilter(xEnable:= , fSample:= , uiSampleNum:= , uiSampleCycle:= , xBusy=> , xValid=> , xError=> , eErrorID=> , fValue=>);

■ Variables

Input variables

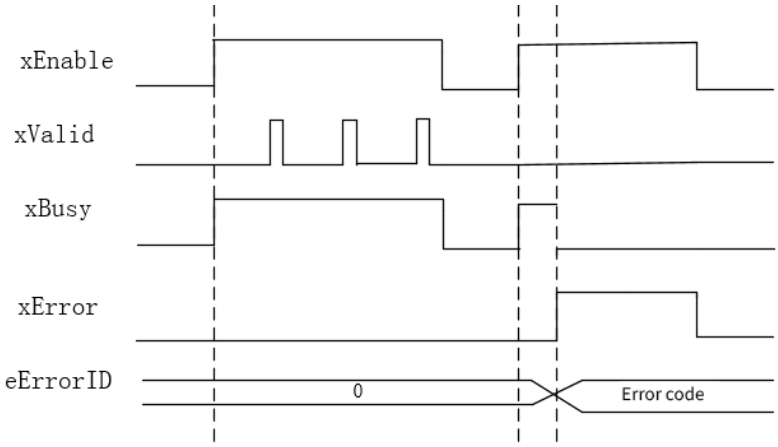
Input Variable	Name	Data Type	Value Range	Initial Value	Description
xEnable	Filter enable	BOOL	[TRUE, FALSE]	FALSE	TRUE: Execute the function block FALSE: Not execute the function block
fSample	Input sampling value	REAL	-	0	Input value to filter
uiSampleNum	Input sampling count	UINT	3 to 3000	0	Input sampling count
uiSampleCycle	Input sampling period	UINT	1 to 1000	0	Input sampling period

Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
xBusy	Executing the instruction	BOOL	[TRUE, FALSE]	FALSE	TRUE: The function block is being executed
xValid	Output active	BOOL	[TRUE, FALSE]	FALSE	TRUE: Instruction execution is valid
xError	Error	BOOL	[TRUE, FALSE]	FALSE	TRUE: An error occurs
eErrorID	Error ID	UTIL_ERROR	See UTIL_ERROR.	0	Error ID For detail, see UTIL_ERROR.
fValue	Filter output effective value	REAL	0	0	Filter output effective value

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
xEnable	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
fSample	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-
uiSampleNum	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-
uiSampleCycle	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-
xBusy	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
xValid	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
xError	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
eErrorID	UTIL_ERROR																			
fValue	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-

■ Timing Diagram



■ Program example

1) The sampling type is consecutive sampling. The raw data for sampling is sourced from the consecutively updated program variables.

2) The number of samples for the sampling calculation is 4 (unsigned integer ranging from 3 to 3000).

3) The sampling period is 1 (data sampling is executed once every given number of running periods, with the default value of 1).

4) When the input value of xFilter is TRUE, the function block starts sampling and filtering with energy flow effective. Set the number of samples for the sampling calculation to 4, indicating that filter calculation is performed once for every four sampling values. Sort the four sampling values by numerical value, exclude the maximum value and minimum value, and average the rest two values. Output the average value as the filter value, and set the filter execution valid flag xValid to TRUE.

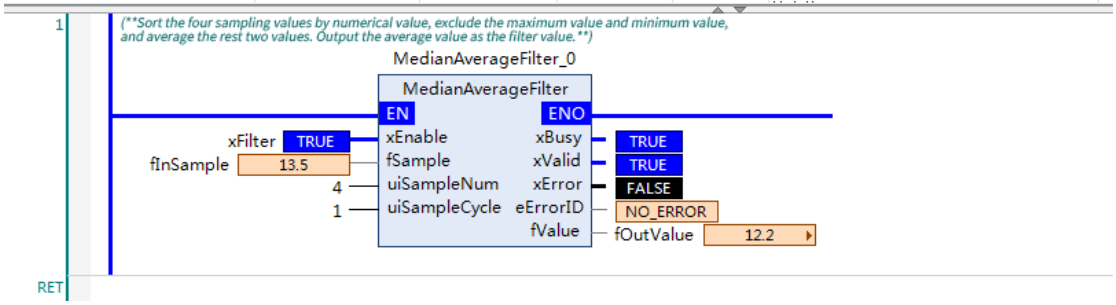
ST

Expression	Type	Value	Prepared Value	Address	Comment
MedianAverageFilter_0	MedianAverageFilter				
xFilter	BOOL	TRUE			Filter enable bit
fInSample	REAL	4.4			Input value

```
1  (**Sort the four sampling values by numerical value, exclude the maximum value and minimum value,
   and average the rest two values. Output the average value as the filter value.**)
2  MedianAverageFilter_0(
3      xEnable:=xFilter ,
4      fSample:= fInSample,
5      uiSampleNum:= 4,
6      uiSampleCycle:=1 ,
7      xBusy=> ,
8      xValid=> ,
9      xError=> ,
10     eErrorID=> ,
11     fValue=>fOutValue );RETURN
```

LD

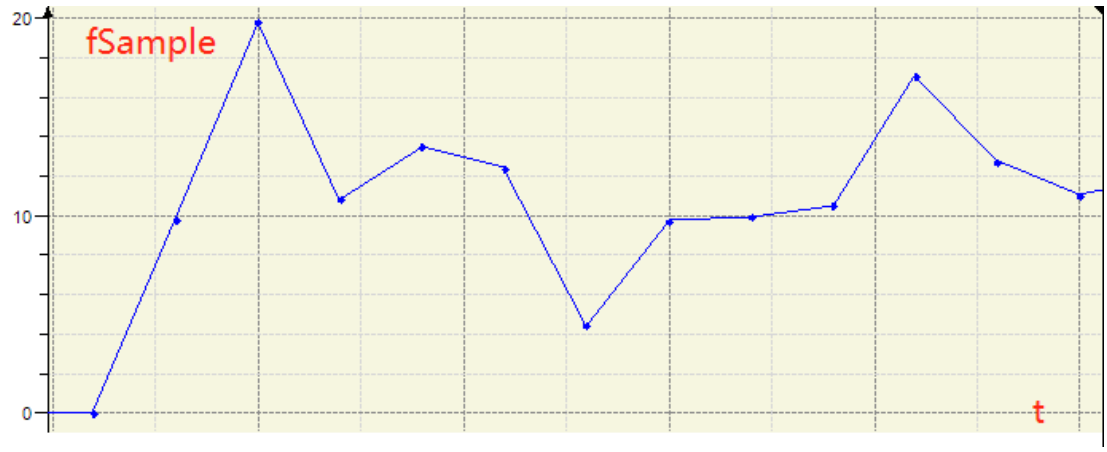
Expression	Type	Value	Prepared Value	Address	Comment
MedianAverageFilter_0	MedianAverageFilter				
xFilter	BOOL	TRUE			Filter enable bit
fInSample	REAL	13.5			Input value



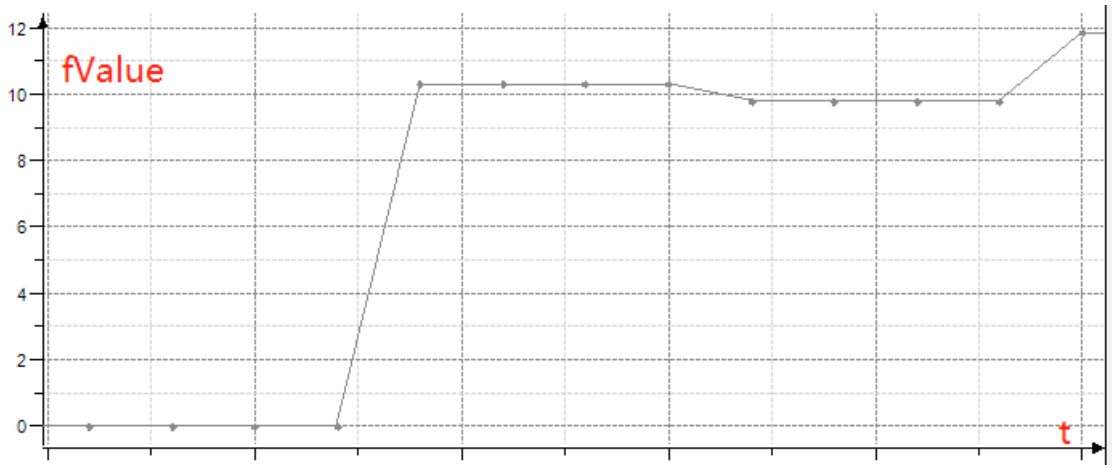
The sampling values of 12 consecutive periods are respectively 9.8, 19.8, 10.8, 13.5, 12.4, 4.4, 9.7, 9.9, 10.5, 17.1, 12.7, and 11.0.

After filtering, the values are respectively 0, 0, 0, 12.15, 12.15, 12.15, 12.15, 9.80, 9.80, 9.80, 9.80, and 11.85.

The following figure shows the curve before filtering.



The following figure shows the curve after filtering.



■ Precautions

1) In the filter function block, the input parameter uiSampleCycle indicates the sampling period. When the value of uiSampleCycle is 1, sampling is performed every period. When the value of uiSampleCycle is 3, sampling is performed once every three periods. If the output of MedianAverageFilter.xValid is TRUE, the filter is valid. The filter output value of fValue is updated when the filter output is valid.

2) If the sampled data is stored in an array, the data needs to be put in the filter function block one by one and handled by the user.

3) For details about errors, see UTIL_ERROR.

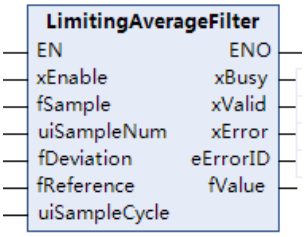
4.9. 7 LimitingAverageFilter

When xEnable is set to TRUE, the sampling variable fSample is input based on the given scan period count uiSampleCycle, sampling count uiSampleNum, sampling reference effective value fReference, and maximum allowable deviation fDeviation for two consecutive samples. Both limiting filter and recursive averaging filter are used. Limit each new sampled data and then add it to the queue for recursive averaging filter.

Advantages: It combines the advantages of both limiting filter and recursive averaging filter. It can eliminate sampling value deviations caused by occasional pulse interference.

Disadvantages: It may consume RAM unnecessarily.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
LimitingAverageFilter	Limiting average filtering	FB		LimitingAverageFilter(xEnable:= , fSample:= , uiSampleNum:= , fDeviation:= , fReference:= , uiSampleCycle:= , xBusy=> , xValid=> , xError=> , eErrorID=> , fValue=>);

■ Variables

Input variables

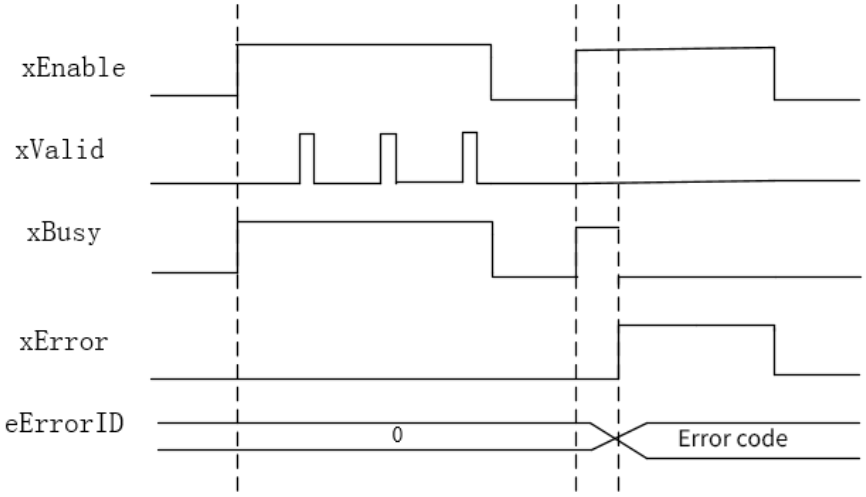
Input Variable	Name	Data Type	Value Range	Initial Value	Description
xEnable	Filter enable	BOOL	[TRUE, FALSE]	FALSE	TRUE: Execute the function block FALSE: Not execute the function block
fSample	Input sampling value	REAL	-	0	Input value to filter
uiSampleNum	Input sampling count	UINT	1 to 1000	0	Input sampling count
fDeviation	Maximum allowable deviation for two consecutive samples	REAL	-	0	Maximum allowable deviation for two consecutive samples
fReference	Input effective value	REAL	-	0	Input effective value
uiSampleCycle	Input sampling period	UINT	1 to 1000	0	Input sampling period

Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
xBusy	Executing the instruction	BOOL	[TRUE, FALSE]	FALSE	TRUE: The function block is being executed
xValid	Output active	BOOL	[TRUE, FALSE]	FALSE	TRUE: Instruction execution is valid
xError	Error	BOOL	[TRUE, FALSE]	FALSE	TRUE: An error occurs
eErrorID	Error ID	UTIL_ERROR	See UTIL_ERROR.	0	Error ID For detail, see UTIL_ERROR.
fValue	Filter output effective value	REAL	0	0	Filter output effective value

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
xEnable	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
fSample	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-
uiSampleNum	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-
fDeviation	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-
fReference	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-
uiSampleCycle	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-
xBusy	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
xValid	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
xError	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
eErrorID	UTIL_ERROR																			
fValue	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-

■ Timing Diagram



■ Program example

The sampling type is consecutive sampling. The raw data for sampling is sourced from the consecutively updated program variables.

The maximum allowable deviation in amplitude is 1.5 (deviation range of the floating-point type).

The amplitude reference value is 10.0 (user-defined reference value of the floating-point type).

The number of samples for the sampling calculation is 3 (unsigned integer ranging from 1 to 3000).

The sampling period is 1 (data sampling is executed once every given number of running periods, with the default value of 1).

When the input xFilter is set to TRUE, the function block starts sampling and filtering with energy flow effective. Set the initial reference value to 10.0 as the effective value. Compare the sampling value with the previous effective value. If the absolute difference between them is greater than the set maximum deviation value of 1.5, use the previous effective value as the current effective value. If it is less than or equal to the maximum deviation value, use the current value as the effective value. Then output the effective value as the filter value, and set the filter execution valid flag xValid to TRUE.

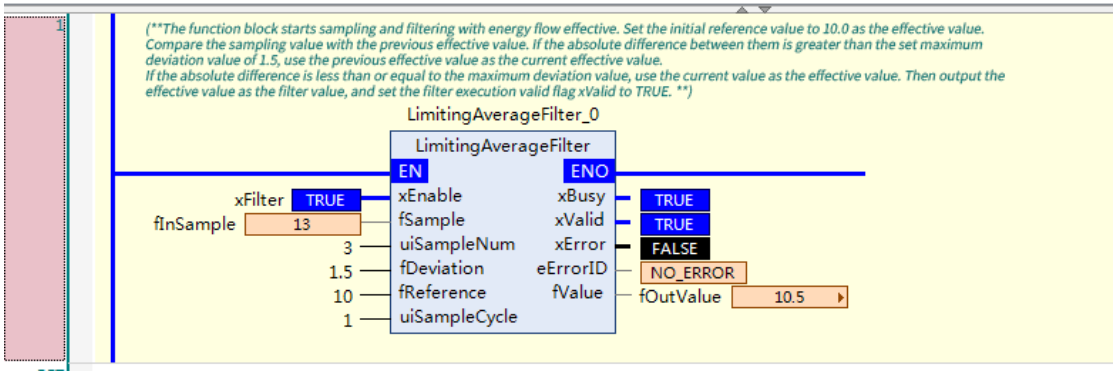
ST

Expression	Type	Value	Prepared Value	Address	Comment
LimitingAverageFilter_0	LimitingAverageFilter				
xFilter	BOOL	TRUE			Filter enable bit
fInSample	REAL	10.8			Input value
fOutValue	REAL	10.2			Output value

```
(**The function block starts sampling and filtering with energy flow effective. Set the initial reference value to 10.0 as the effective value. Compare the sampling
value with the previous effective value. If the absolute difference between them is greater than the set maximum deviation value of 1.5, use the previous effective
value as the current effective value.
If the absolute difference is less than or equal to the maximum deviation value, use the current value as the effective value. Then output the effective value as the
filter value, and set the filter execution valid flag xValid to TRUE. **)
LimitingAverageFilter_0(
    xEnable:=xFilter,
    fSample:=fInSample,
    uiSampleNum:= 3, // Input sampling count
    fDeviation:= 1.5, // Maximum allowable deviation for two consecutive samples
    fReference:= 10, //Input effective value
    uiSampleCycle:= 1, //Input sampling period
    xBusy=> ,
    xValid=> ,
    xError=> ,
    eErrorID=> ,
    fValue=>fOutValue );RETURN
```

LD

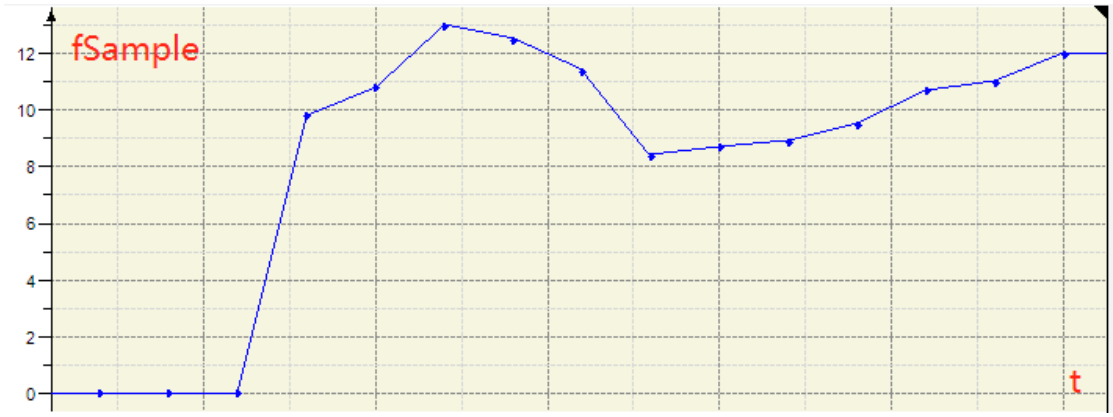
Expression	Type	Value	Prepared Value	Address	Comment
LimitingAverageFilter_0	LimitingAverageFilter				
xFilter	BOOL	TRUE			Filter enable bit
fInSample	REAL	13			Input value
fOutValue	REAL	10.4666672			Output value



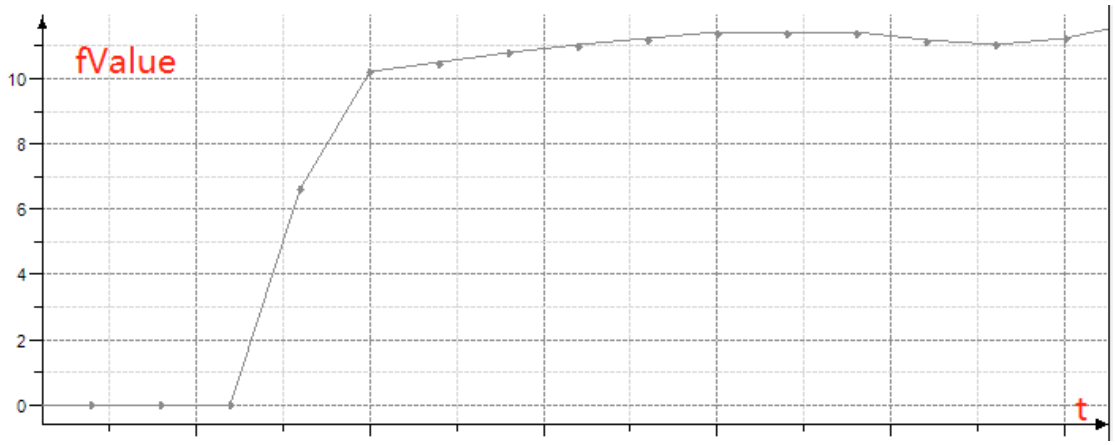
The sampling values of 12 consecutive periods are respectively 9.8, 10.8, 13.0, 12.5, 11.4, 8.4, 8.7, 8.9, 9.5, 10.7, 11.0, and 12.0.

After filtering, the values are respectively 6.6, 10.2, 10.47, 10.8, 11, 11.2, 11.4, 11.4, 11.4, 11.17, 11.03, and 11.23.

The following figure shows the curve before filtering.



The following figure shows the curve after filtering.



■ Precautions

1) In the filter function block, the input parameter uiSampleCycle indicates the sampling period. When the value of uiSampleCycle is 1, sampling is performed every period. When the value of uiSampleCycle is 3, sampling is performed once every three periods. If the output of the function block flag LimitingAverageFilter.xValid is TRUE, the filter is valid. The filter output value of fValue is updated when the filter output is valid.

2) If the sampled data is stored in an array, the data needs to be put in the filter function block one by one and handled by the user.

3) For details about errors, see UTIL_ERROR.

4.9. 8 FirstOrderLagFilter

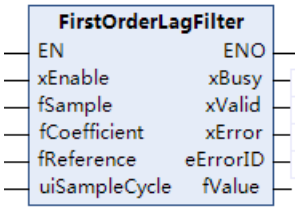
When xEnable is set to TRUE, the sampling variable fSample is input based on the given scan period count uiSampleCycle, sampling reference value fReference, and input first-order low-pass filter coefficient fCoefficient (0 to 1). Each time a new value is sampled, the calculation is as follows:

Current filter result = fCoefficient x Current sampling value + (1 – fCoefficient) x Previous filter result

Advantages: It suppresses periodic interference well, and it is suitable for systems with high-frequency oscillations.

Disadvantages: It has phase lag and low sensitivity. The lagging program depends on the value of the fCoefficient. It cannot eliminate interference signals with the filter frequency higher than half of the sampling frequency.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
FirstOrderLagFilter	First order lag filtering	FB		FirstOrderLagFilter(xEnable:= , fSample:= , fCoefficient:= , fReference:= , uiSampleCycle:= , xBusy=> , xValid=> , xError=> , eErrorID=> , fValue=>);

■ Variables

Input variables

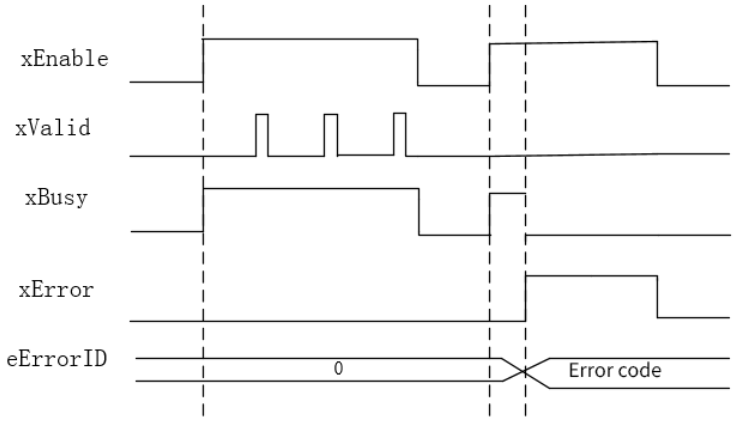
Input Variable	Name	Data Type	Value Range	Initial Value	Description
xEnable	Filter enable	BOOL	[TRUE, FALSE]	FALSE	TRUE: Execute the function block FALSE: Not execute the function block
fSample	Input sampling value	REAL	-	0	Input value to filter
fCoefficient	Input first-order low-pass filter coefficient (a = 0 to 1)	REAL	0 to 1	0	Input first-order low-pass filter coefficient
fReference	Input effective value	REAL	-	0	Input effective value
uiSampleCycle	Input sampling period	UINT	1 to 1000	0	Input sampling period

Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
xBusy	Executing the instruction	BOOL	[TRUE, FALSE]	FALSE	TRUE: The function block is being executed
xValid	Output active	BOOL	[TRUE, FALSE]	FALSE	TRUE: Instruction execution is valid
xError	Error	BOOL	[TRUE, FALSE]	FALSE	TRUE: An error occurs
eErrorID	Error ID	UTIL_ERROR	See UTIL_ERROR.	0	Error ID For detail, see UTIL_ERROR.
fValue	Filter output effective value	REAL	0	0	Filter output effective value

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
xEnable	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
fSample	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-
fCoefficient	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-
fReference	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-
uiSample-Cycle	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-
xBusy	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
xValid	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
xError	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
eErrorID	UTIL_ERROR																			
fValue	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-

■ Timing Diagram



■ Program example

The sampling type is consecutive sampling. The raw data for sampling is sourced from the consecutively updated program variables.

The first-order low-pass filter coefficient is 0.01, which is of the floating-point type.

The filter reference value is 10.0.

The sampling period is 1, indicating that data is sampled once a period.

When the input xFilter is set to TRUE, the function block starts sampling and filtering with energy flow effective. Set the initial reference value to 10.0 as the effective value and the first-order low-pass filter coefficient to 0.01. Calculate the filter value as follows: Current filter result = fCoefficient x Current sampling value + (1 – fCoefficient) x Previous filter result. Output the filter value, and set the filter execution valid flag xValid to TRUE.

ST

Expression	Type	Value	Prepared Value	Address	Comment
FirstOrderLagFilter_0	FirstOrderLagFilter				
xFilter	BOOL	TRUE			Filter enable bit
fInSample	REAL	9.8			Input value
fOutValue	REAL	9.99602			Output value

```
1  (**Current filter result = fCoefficient x Current sampling value + (1 – fCoefficient) x Last filter resulte current value as the effective value. Then
   output the effective value as the filter value, and set the filter execution valid flag xValid to TRUE. **)
2  FirstOrderLagFilter_0 (
3      xEnable TRUE :=xFilter TRUE ,
4      fSample 9.8 :=fInSample 9.8 ,
5      fCoefficient 0.01 :=0.01 ,
6      fReference 10 := 10,
7      uiSampleCycle 1 :=1 ,
8      xBusy=> ,
9      xValid=> ,
10     xError=> ,
11     eErrorID=> ,
12     fValue 10 => fOutValue 10 );RETURN
```

LD

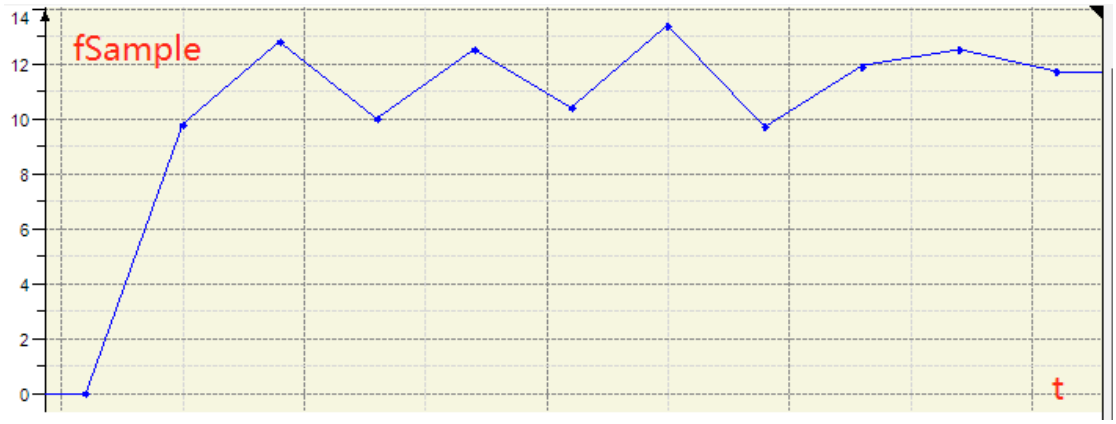
Expression	Type	Value	Prepared Value	Address	Comment
FirstOrderLagFilter_0	FirstOrderLagFilter				
xFilter	BOOL	TRUE			Filter enable bit
fInSample	REAL	12.8			Input value
fOutValue	REAL	10.02602			Output value

```
(**Current filter result = fCoefficient x Current sampling value + (1 – fCoefficient) x Last filter resulte current value as the effective value. Then
output the effective value as the filter value, and set the filter execution valid flag xValid to TRUE. **)
FirstOrderLagFilter_0
                FirstOrderLagFilter
                EN                  ENO
xFilter  TRUE   xEnable     xBusy   TRUE
fInSample 12.8  fSample     xValid  TRUE
0.01            fCoefficient xError FALSE
10              fReference  eErrorID NO_ERROR
1               uiSampleCycle fValue fOutValue 10
```

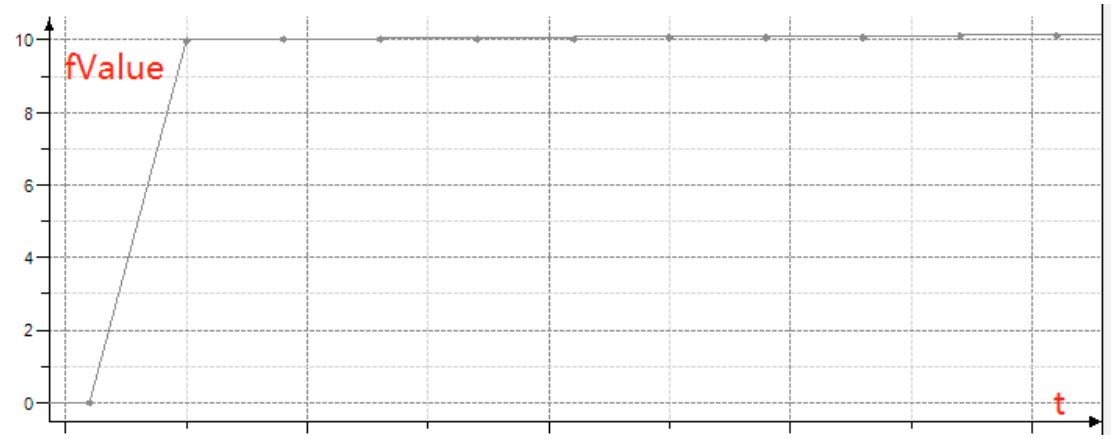
The sampling values of 10 consecutive periods are respectively 9.8, 12.8, 10.0, 12.5, 10.4, 13.4, 9.7, 11.9, 12.5, and 11.7.

After filtering, the values are respectively 9.998, 10.03, 10.03, 10.05, 10.05, 10.09, 10.08, 10.10, 10.13, and 10.14.

The following figure shows the curve before filtering.



The following figure shows the curve after filtering.



■ Precautions

1) In the filter function block, the input parameter uiSampleCycle indicates the sampling period. When the value of uiSampleCycle is 1, sampling is performed every period. When the value of uiSample-Cycle is 3, sampling is performed once every three periods. If the output of the function block flag FirstOrderLagFilter.xValid is TRUE, the filter is valid. The filter output value of fValue is updated when the filter output is valid.

2) If the sampled data is stored in an array, the data needs to be put in the filter function block one by

one and handled by the user.

3) For details about errors, see UTIL_ERROR.

4.9. 9 WeightRecursiveAverageFilter

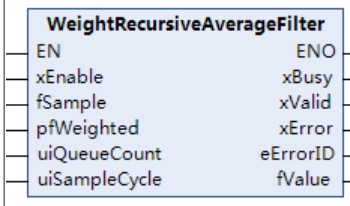
When xEnable is set to TRUE, the sampling variable fSample and corresponding weighting coefficient array address pfWeighted are input based on the given scan period count uiSampleCycle and queue length uiQueueCount.

This feature is an improvement for recursive averaging filter, where different weights are assigned to data at different time points. Typically, a higher weight is assigned to the data closer to the current time point. When the weight coefficient assigned to a new sampled value is larger, the sensitivity increases but the signal smoothness decreases.

Advantages: It is suitable for objects with a large pure time-delay constant and systems with a short sampling period.

Disadvantages: It is not suitable for signals with a small pure time-delay constant, long sampling period, and slow changes. It cannot quickly reflect the severity of interference the system is experiencing, resulting in a poor filter effect.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
WeightRecursiveAverageFilter	Weighted recursive average filtering	FB		WeightRecursiveAverageFilter(xEnable:= , fSample:= , pfWeighted:= , uiQueueCount:= , uiSampleCycle:= , xBusy=> , xValid=> , xError=> , eErrorID=> , fValue=>);

■ Variables

Input variables

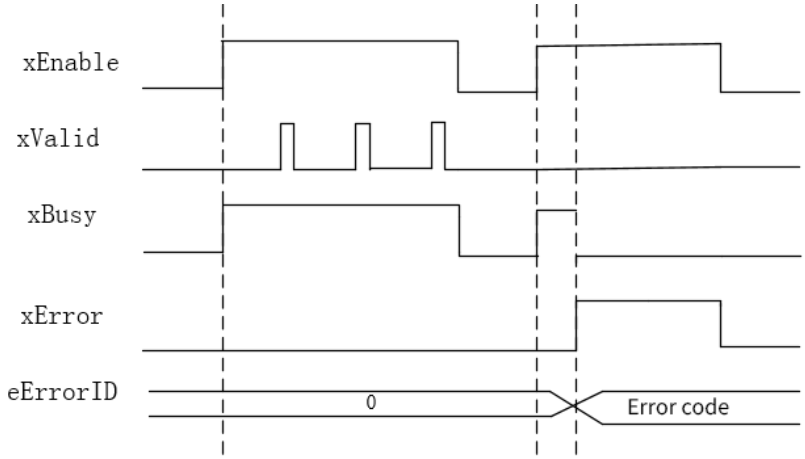
Input Variable	Name	Data Type	Value Range	Initial Value	Description
xEnable	Filter enable	BOOL	[TRUE, FALSE]	FALSE	TRUE: Execute the function block FALSE: Not execute the function block
fSample	Input sampling value	REAL	-	0	Input value to filter
pfWeighted	Input weighting coefficient, stored in an array	POINTER TO REAL	0	0	Input weighting coefficient, stored in an array, ensuring that not all weighting coefficients are 0
uiQueueCount	Input queue length N	UINT	1 to 3000	0	Input effective value
uiSampleCycle	Input sampling period	UINT	1 to 1000	0	Input sampling period

Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
xBusy	Executing the instruction	BOOL	[TRUE, FALSE]	FALSE	TRUE: The function block is being executed
xValid	Output active	BOOL	[TRUE, FALSE]	FALSE	TRUE: Instruction execution is valid
xError	Error	BOOL	[TRUE, FALSE]	FALSE	TRUE: An error occurs
eErrorID	Error ID	UTIL_ERROR	See UTIL_ERROR.	0	Error ID For detail, see UTIL_ERROR.
fValue	Filter output effective value	REAL	0	0	Filter output effective value

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
xEnable	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
fSample	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-
pfWeighted	POINTER TO REAL																			
uiQueueCount	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-
uiSampleCycle	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-
xBusy	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
xValid	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
xError	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
eErrorID	UTIL_ERROR																			
fValue	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-

■ Timing Diagram



■ Program example

The sampling type is consecutive sampling. The raw data for sampling is sourced from the consecutively updated program variables.

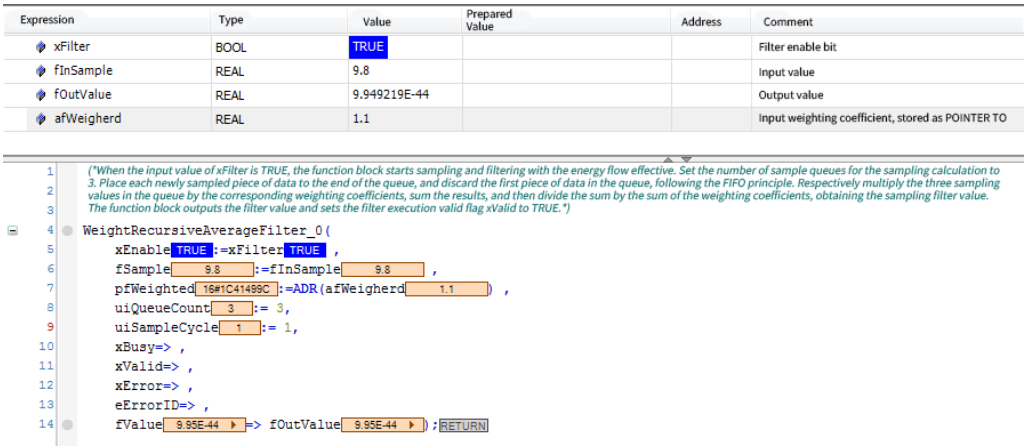
The weighting coefficients are floating-point numbers 1.0, 1.2, 1.3, and they correspond to the weights of

sampled data. The weighting coefficient quantity m is equal to the queue length n. If m is greater than n, only the first n weighting coefficients are used. The number of sample queues for the sampling calculation is 3 (unsigned integer ranging from 1 to 3000).

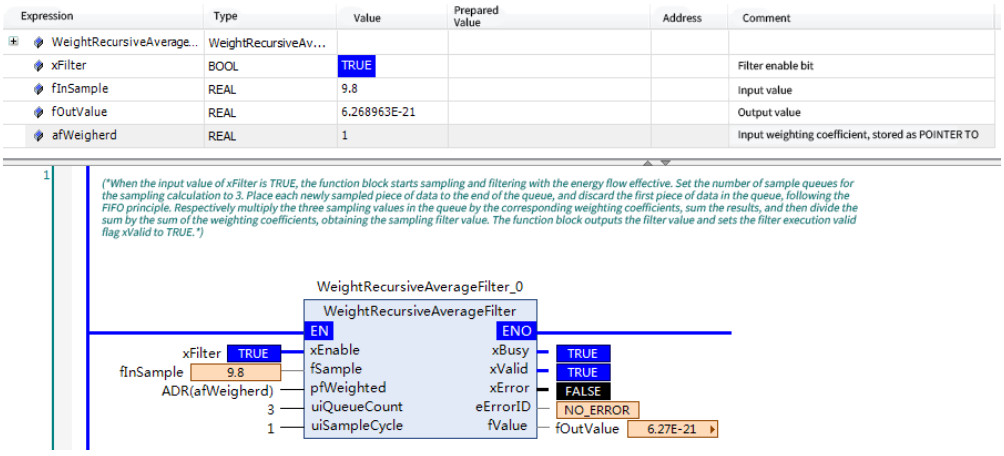
The filter calculation period is 1 (the sampling filter is executed once every given number of running period, with the default value of 1).

When the input value of xFilter is TRUE, the function block starts sampling and filtering with the energy flow effective. Set the number of sample queues for the sampling calculation to 3. Place each newly sampled piece of data to the end of the queue, and discard the first piece of data in the queue, following the FIFO principle. Respectively multiply the three sampling values in the queue by the corresponding weighting coefficients, sum the results, and then divide the sum by the sum of the weighting coefficients, obtaining the sampling filter value. The function block outputs the filter value and sets the filter execution valid flag xValid to TRUE.

ST



LD

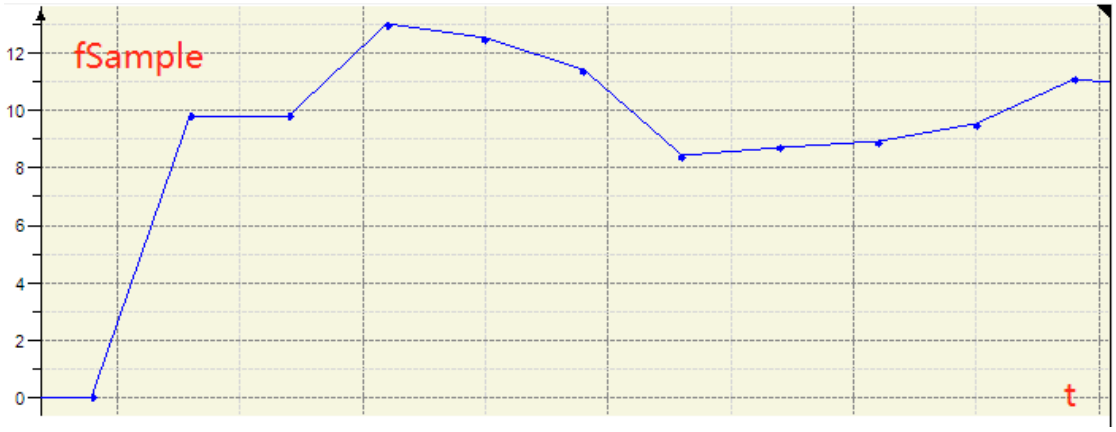


The weighting coefficients of 10 consecutive running periods are 1.0, 1.1, and 1.2.

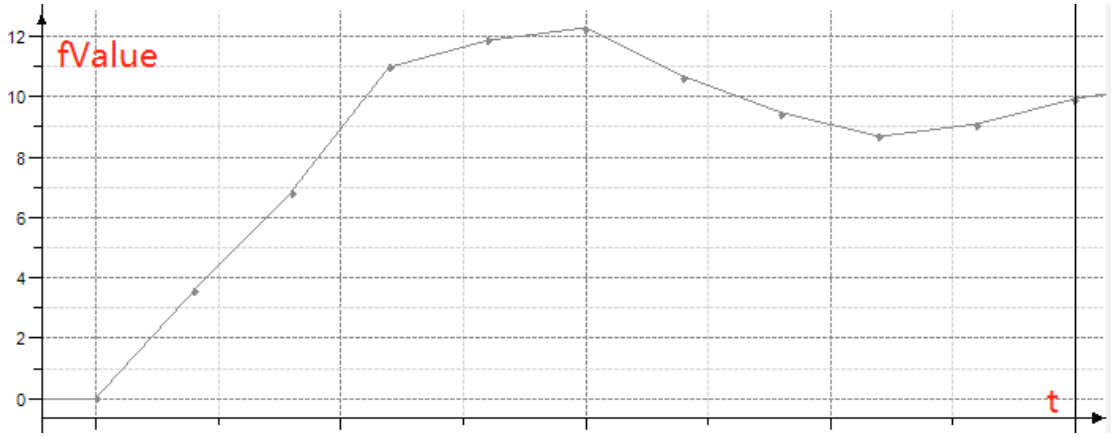
The sampling values are respectively 9.8, 9.8, 13.0, 12.5, 11.4, 8.4, 8.7, 8.9, 9.5, and 11.1.

After filtering, the values are respectively 0, 0, 10.96, 11.85, 12.25, 10.64, 9.42, 8.68, 9.06, and 9.9.

The following figure shows the curve before filtering.



The following figure shows the curve after filtering.



■ Precautions

1) In the filter function block, the input parameter uiSampleCycle indicates the sampling period. When the value of uiSampleCycle is 1, sampling is performed every period. When the value of uiSampleCycle is 3, sampling is performed once every three periods. If the output of WeightRecursiveAverageFilter.xValid is TRUE, the filter is valid. The filter output value of fValue is updated when the filter output is valid.

2) If the sampled data is stored in an array, the data needs to be put in the filter function block one by one and handled by the user.

3) Ensure that not all weighting coefficients are 0.

4) For details about errors, see UTIL_ERROR.

4.9. 10 DebounceFilter

When xEnable is set to TRUE, the sampling variable fSample is input based on the given scan period count uiSampleCycle, sampling reference value fReference, and filter upper limit uiUpLimit.

Compare each sampled value with the current effective value:

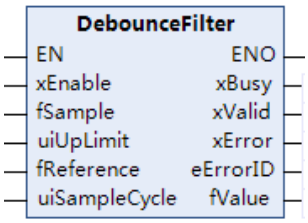
If the sampling value is equal to the current effective value, set the current count value to zero.

If the sampling value is not equal to the current effective value, increment the count value by 1 and check whether the count value exceeds the upper limit uiUpLimit.

If the count value exceeds the upper limit, replace the current filter value with the current sampling value, output the value, and reset the count value to zero.

Advantages: It provides good filter effect for slowly changing parameters, and prevents repeated on/off switchovers on the controller or numerical jitters on the display near critical values.

Disadvantages: It is not suitable for rapidly changing parameters. If the value sampled at the time of counter overflow happens to be an interference value, it may be mistakenly introduced into the system as an effective value.

Instruction	Name	FB/FC	LD Expression	ST Expression
DebounceFilter	Debounce filter	FB		DebounceFilter(xEnable:= , fSample:= , uiUpLimit:= , fReference:= , uiSampleCycle:= , xBusy=> , xValid=> , xError=> , eErrorID=> , fValue=>);

■ Variables

Input variables

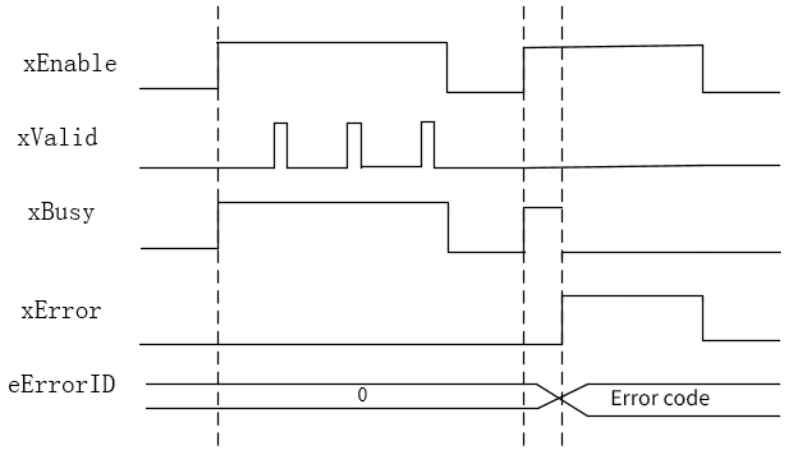
Input Variable	Name	Data Type	Value Range	Initial Value	Description
xEnable	Filter enable	BOOL	[TRUE, FALSE]	FALSE	TRUE: Execute the function block FALSE: Not execute the function block
fSample	Input sampling value	REAL	-	0	Input value to filter
uiUpLimit	Filter counter upper limit to set	UINT	1 to 65535	0	Filter counter upper limit to set
fReference	Input reference value	REAL	-	0	Input reference value
uiSampleCycle	Input sampling period	UINT	1 to 1000	0	Input sampling period

Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
xBusy	Executing the instruction	BOOL	[TRUE, FALSE]	FALSE	TRUE: The function block is being executed
xValid	Output active	BOOL	[TRUE, FALSE]	FALSE	TRUE: Instruction execution is valid
xError	Error	BOOL	[TRUE, FALSE]	FALSE	TRUE: An error occurs
eErrorID	Error ID	UTIL_ERROR	See UTIL_ERROR.	0	Error ID For detail, see UTIL_ERROR.
fValue	Filter output effective value	REAL	0	0	Filter output effective value

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
xEnable	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
fSample	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-
uiUpLimit	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-
fReference	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-
uiSampleCycle	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-
xBusy	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
xValid	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
xError	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
eErrorID	UTIL_ERROR																			
fValue	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-

■ Timing Diagram



■ Program example

The sampling type is consecutive sampling. The raw data for sampling is sourced from the consecutively updated program variables.

The reference effective value of the sample is 10.0 (user-defined reference value of the floating-point type).

The maximum number of jitters for the sample value on the counter is 3 (unsigned integer).

The sampling period is 1 (data sampling is executed once every given number of running periods, with the default value of 1).

When the input xFilter is set to TRUE, the function block starts sampling and filtering with energy flow effective. Set the reference value to 10.0 as the effective value. Compare the sampling value with the current effective value. If they are the same, reset the jitter counter value to zero. If they are different, increase the counter value by 1. When the counter value exceeds the counter value upper limit, replace the current effective value with the current sampling value as the filter effective value, output the effective value, and set the filter execution valid flag xValid to TRUE.

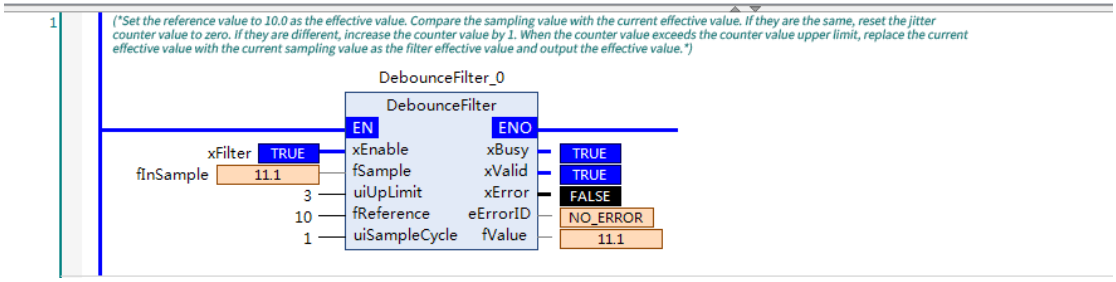
ST

Expression	Type	Value	Prepared Value	Address	Comment
DebounceFilter_0	DebounceFilter				
xFilter	BOOL	TRUE			Filter enable bit
fInSample	REAL	9.8			Input value
fOutValue	REAL	10			Output value

```
(*Set the reference value to 10.0 as the effective value. Compare the sampling value with the current effective value. If they are the same, reset the jitter counter value to zero. If they are different, increase the counter value by 1. When the counter value exceeds the counter value upper limit, replace the current effective value with the current sampling value as the filter effective value and output the effective value.*)
DebounceFilter_0(
    xEnable:=xFilter,
    fSample:= fInSample,
    uiUpLimit:= 3,
    fReference:= 10,
    uiSampleCycle:=1 ,
    xBusy=> ,
    xValid=> ,
    xError=> ,
    eErrorID=> ,
    fValue=>fOutValue );RETURN
```

LD

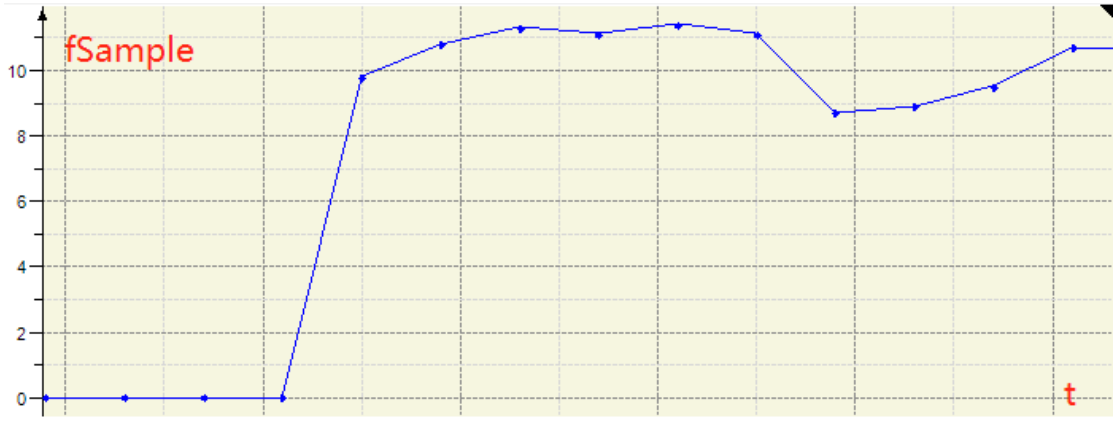
Expression	Type	Value	Prepared Value	Address	Comment
DebounceFilter_0	DebounceFilter				
xFilter	BOOL	TRUE			Filter enable bit
fInSample	REAL	11.1			Input value
fOutValue	REAL	0			Output value



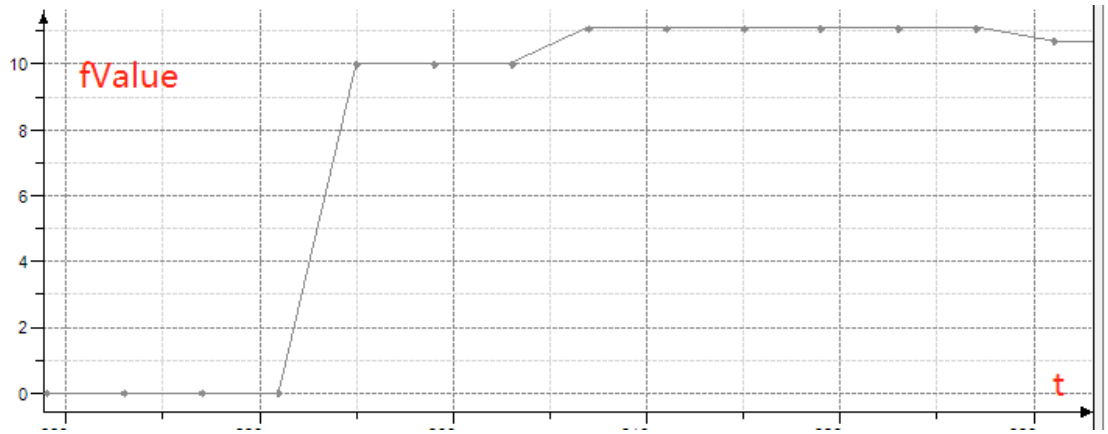
The sampling values of 10 consecutive periods are respectively 9.8, 10.8, 11.3, 11.1, 11.4, 11.1, 8.7, 8.9, 9.5, and 10.7.

After filtering, the values are respectively 10.0, 10.0, 10.0, 11.1, 11.1, 11.1, 11.1, 11.1, 11.1, and 10.7.

The following figure shows the curve before filtering.



The following figure shows the curve after filtering.



■ Precautions

1) In the filter function block, the input parameter uiSampleCycle indicates the sampling period. When the value of uiSampleCycle is 1, sampling is performed every period. When the value of uiSampleCycle is 3, sampling is performed once every three periods. If the output of the function block flag DebounceFilter.xValid is TRUE, the filter is valid. The filter output value of fValue is updated when the filter output is valid.

2) If the sampled data is stored in an array, the data needs to be put in the filter function block one by one and handled by the user.

3) For details about errors, see UTIL_ERROR.

4.9. 11 LimitingDebounceFilter

When xEnable is set to TRUE, the sampling variable fSample is input based on the given scan period count uiSampleCycle, filter counter upper limit uiUpLimit, reference value fReference, and maximum allowable deviation fDeviation for two consecutive samples. Both limiting filter and debounce filter are used. Limiting filter is used before debounce filter.

If the difference between the current value and the previous value is less than or equal to the value of fDeviation, the current value is valid and is used as the sampling value for debounce filter.

If the difference between the current value and the previous value is greater than the value of fDeviation, the current value is invalid. In this case, the current value is discarded, and the previous value is used as the sampling value for debounce filter.

Use the output of limiting filter as the sampling value for debounce filter, and compare it with the current effective value:

If the sampling value is equal to the current effective value, set the current count value to zero.

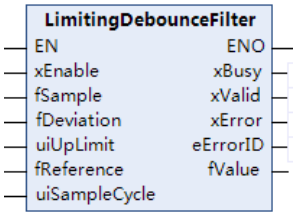
If the sampling value is not equal to the current effective value, increment the count value by 1 and check whether the count value exceeds the upper limit uiUpLimit.

If the count value exceeds the upper limit, replace the current filter value with the current sampling value, output the value, and reset the count value to zero.

Advantages: It combines the advantages of both limiting filter and debounce filter. It overcomes certain defects of debounce filter, preventing introduction of interference values into the system.

Disadvantages: It is not suitable for rapidly changing parameters.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
LimitingDebounceFilter	Limit debounce filter	FB		LimitingDebounceFilter(xEnable:= , fSample:= , fDeviation:= , uiUpLimit:= , fReference:= , uiSampleCycle:= , xBusy=> , xValid=> , xError=> , eErrorID=> , fValue=>);

■ Variables

Input variables

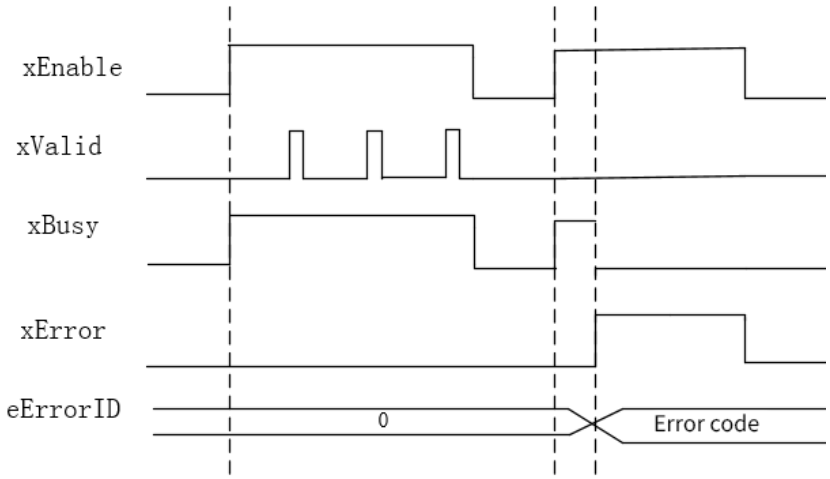
Input Variable	Name	Data Type	Value Range	Initial Value	Description
xEnable	Filter enable	BOOL	[TRUE, FALSE]	FALSE	TRUE: Execute the function block FALSE: Not execute the function block
fSample	Input sampling value	REAL	-	0	Input value to filter
fDeviation	Maximum allowable deviation for two consecutive samples	REAL	-	0	Maximum allowable deviation for two consecutive samples
uiUpLimit	Filter counter upper limit to set	UINT	1 to 65535	0	Filter counter upper limit to set
fReference	Input reference value	REAL	-	0	Input reference value
uiSampleCycle	Input sampling period	UINT	1 to 1000	0	Input sampling period

Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
xBusy	Executing the instruction	BOOL	[TRUE, FALSE]	FALSE	TRUE: The function block is being executed
xValid	Output active	BOOL	[TRUE, FALSE]	FALSE	TRUE: Instruction execution is valid
xError	Error	BOOL	[TRUE, FALSE]	FALSE	TRUE: An error occurs
eErrorID	Error ID	UTIL_ERROR	See UTIL_ERROR.	0	Error ID For detail, see UTIL_ERROR.
fValue	Filter output effective value	REAL	0	0	Filter output effective value

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
xEnable	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
fSample	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-
fDeviation	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-
uiUpLimit	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-
fReference	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-
uiSampleCycle	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-
xBusy	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
xValid	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
xError	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
eErrorID	UTIL_ERROR																			
fValue	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-

■ Timing Diagram



■ Program example

The sampling type is consecutive sampling. The raw data for sampling is sourced from the consecutively updated program variables.

The reference effective value of the sample is 10.0 (user-defined reference value of the floating-point type).

The maximum number of jitters for the sample value on the counter is 3 (unsigned integer).

The sampling period is 1 (data sampling is executed once every given number of running periods, with the default value of 1).

When the input xFilter is set to TRUE, the function block starts sampling and filtering with energy flow effective. Set the reference value to 10.0 as the effective value. Compare the sampling value with the current effective value. If they are the same, reset the jitter counter value to zero. If they are different, increase the counter value by 1. When the counter value exceeds the counter value upper limit, replace the current effective value with the current sampling value as the filter effective value, output the effective value, and set the filter execution valid flag xValid to TRUE.

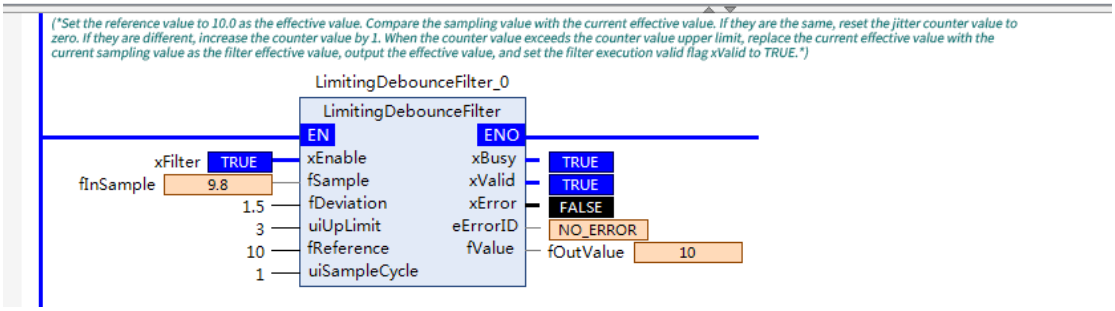
ST

Expression	Type	Value	Prepared Value	Address	Comment
LimitingDebounceFilter_0	LimitingDebounceFilter				
xFilter	BOOL	TRUE			Filter enable bit
fInSample	REAL	11.1			Input value
fOutValue	REAL	11.1			Output value

```
(*Set the reference value to 10.0 as the effective value. Compare the sampling value with the current effective value. If they are the same, reset the jitter counter value to zero. If they are different, increase the counter value by 1. When the counter value exceeds the counter value upper limit, replace the current effective value with the current sampling value as the filter effective value, output the effective value, and set the filter execution valid flag xValid to TRUE.*)
LimitingDebounceFilter_0(
    xEnable:=xFilter ,
    fSample:= fInSample,
    fDeviation:= 1.5,
    uiUpLimit:=3 ,
    fReference:=10 ,
    uiSampleCycle:=1 ,
    xBusy=> ,
    xValid=> ,
    xError=> ,
    eErrorID=> ,
    fValue=> fOutValue);RETURN
```

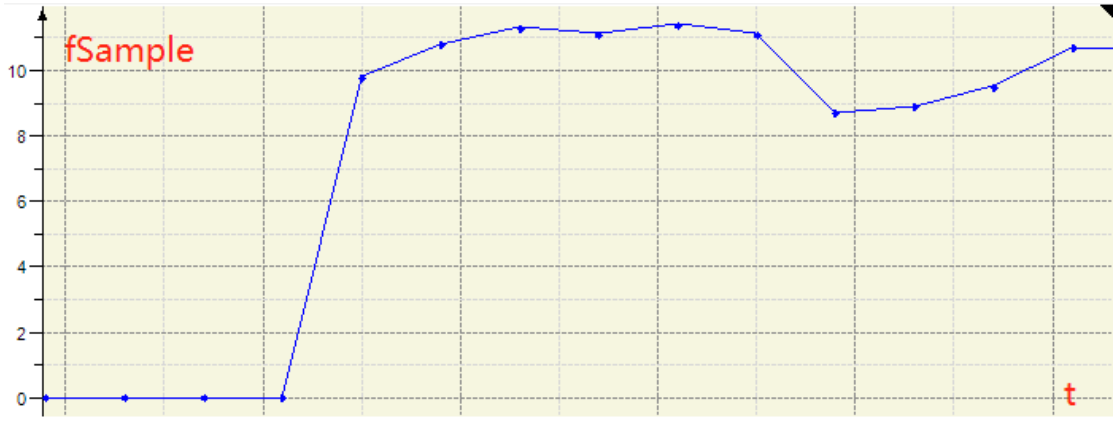
LD

Expression	Type	Value	Prepared Value	Address	Comment
LimitingDebounceFilter_0	LimitingDebounceFilter				
xFilter	BOOL	TRUE			Filter enable bit
fInSample	REAL	9.8			Input value
fOutValue	REAL	10			Output value

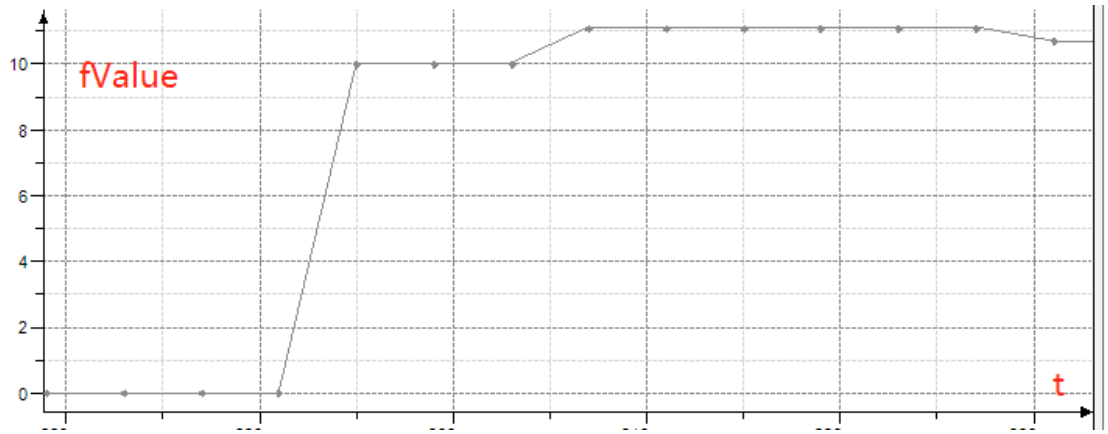


The sampling values of 10 consecutive periods are respectively 9.8, 10.8, 11.3, 11.1, 11.4, 11.1, 8.7, 8.9, 9.5, and 10.7.

After filtering, the values are respectively 10.0, 10.0, 10.0, 11.1, 11.1, 11.1, 11.1, 11.1, 11.1, and 11.1.



The following figure shows the curve after filtering.



■ Precautions

1) In the filter function block, the input parameter uiSampleCycle indicates the sampling period. When the value of uiSampleCycle is 1, sampling is performed every period. When the value of uiSampleCycle is 3, sampling is performed once every three periods. If the output of the function block flag LimitingDebounceFilter.xValid is TRUE, the filter is valid. The filter output value of fValue is updated when the filter output is valid.

2) If the sampled data is stored in an array, the data needs to be put in the filter function block one by one and handled by the user.

3) For details about errors, see UTIL_ERROR.

Failure Information	Range	Data Type	Description
NO_ERROR	UTIL_ERROR	VAR	No fault
DEVIATION_PARAMETER_SET_ERROR	UTIL_ERROR	VAR	Incorrect absolute value for the maximum deviation of two consecutive samples, which should be greater than 0
UPLIMIT_PARAMETER_SET_ERROR	UTIL_ERROR	VAR	Incorrect upper limit of the jitter count value
MAF_SAMPLENUM_PARAMETER_SET_ERROR	UTIL_ERROR	VAR	Incorrect consecutive sampling number of median average filtering, which should be 3 to 3000
SAMPLENUM_PARAMETER_SET_ERROR	UTIL_ERROR	VAR	Incorrect consecutive sampling number, which should be 1 to 1000
QUEUECOUNT_PARAMETER_SET_ERROR	UTIL_ERROR	VAR	Incorrect length of the queue for storing sampling values
COEFFICIENT_PARAMETER_SET_ERROR	UTIL_ERROR	VAR	Incorrect first-order low-pass filter coefficient
WEIGHTED_PARAMETER_SET_ERROR	UTIL_ERROR	VAR	Incorrect weighting coefficients, which cannot be all 0
_PARAMETER_SET_ERROR	UTIL_ERROR	VAR	-
_PARAMETER_SET_ERROR	UTIL_ERROR	VAR	-
_PARAMETER_SET_ERROR	UTIL_ERROR	VAR	-

4. 10 System Instructions

4.10. 1 Instruction List

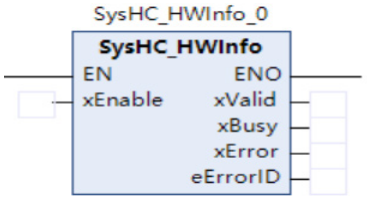
Instruction Category	Name	FB/FC	Function
System instructions	SysHC_HWInfo	FB	Get PLC hardware information
	SysHC_SWInfo	FB	Get PLC software information
	SysHC_CPUInfo	FB	Get the CPU, memory, and boot time information of the PLC
	SysHC_CPUDiagnose	FB	Get the CPU hardware diagnosis information
	DiagnosticMessage	FB	Get CPU diagnostic message
	SysHC_NetworkConfig	FB	Configure PLC network ports
	SysHC_NetworkInfo	FB	Obtain PLC network configuration
	SysHC_UDiskPath	FB	Obtain the path of the U disk
	SysHC_SetSerialParam	FB	Parameter setting for serial port free protocol
	SysHC_SetSerialSendRecvParam	FB	Send/Receive parameter setting for serial port free protocol
	SysHC_GatewayConfig2	FB	Gateway setting
	GetFPGALogicVersion	FB	Get the FPGA logic version number
	GetFPGASoftwareVersion	FB	Get the FPGA software version number
	GetBootVersion	FB	Get the boot version
	GetPLCVersion	FB	Get the PLC firmware version
	GetProductName	FB	Get the PLC name
	GetRuntimeVersion	FB	Get runtime version
	GetSerialNumber	FB	Get the PLC serial number (unique)
	GET_CPU_IOMODULE_DIAGNOSE	FB	Get diagnostic information of CPU local modules

4.10. 2 SysHC_HWInfo

This instruction gets PLC hardware information.

This instruction saves the PLC boot time, CPU load, CPU temperature, CPU utilization, memory size, memory usage, product name, serial number, and external USB flash disk path to internal attribute variables.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
SysHC_HWInfo	Get PLC hardware information	FB		SysHC_HWInfo(xEnable: , xValid=> , xBusy=> , xError=> , eErrorID=>);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
xEnable	Active high	BOOL	0 to 1	0	Active high

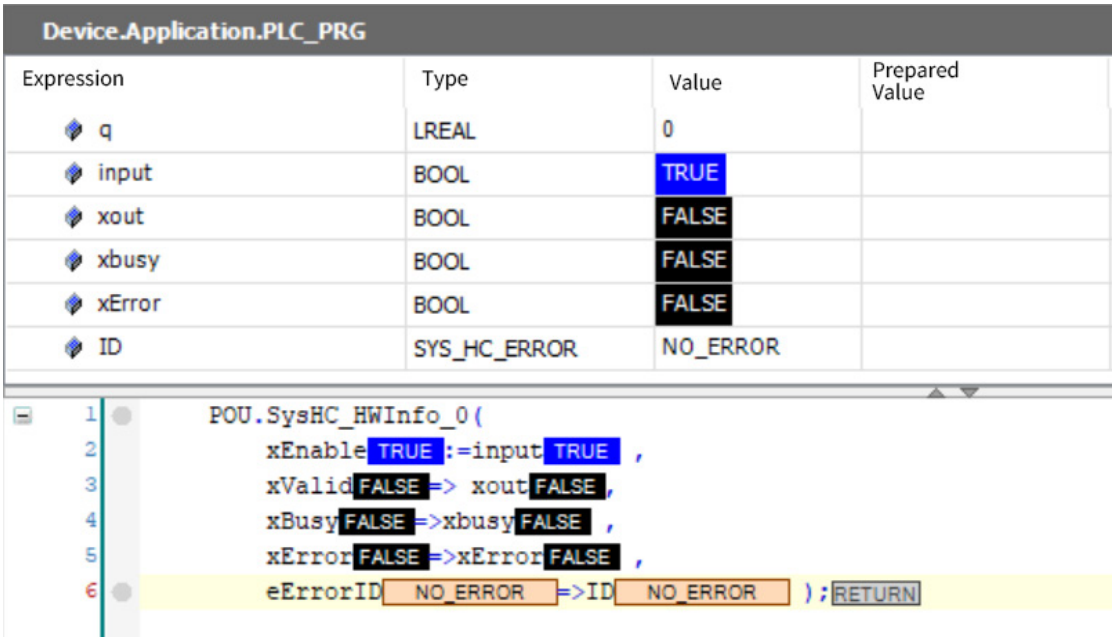
Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
xValid	Output active	BOOL	0 to 1	0	Output active
xBusy	Executing the instruction	BOOL	0 to 1	0	Executing the instruction
xError	Error	BOOL	0 to 1	0	Error
eErrorID	Error ID	SYS_HC_ERROR	-	NO_ERROR	Error ID

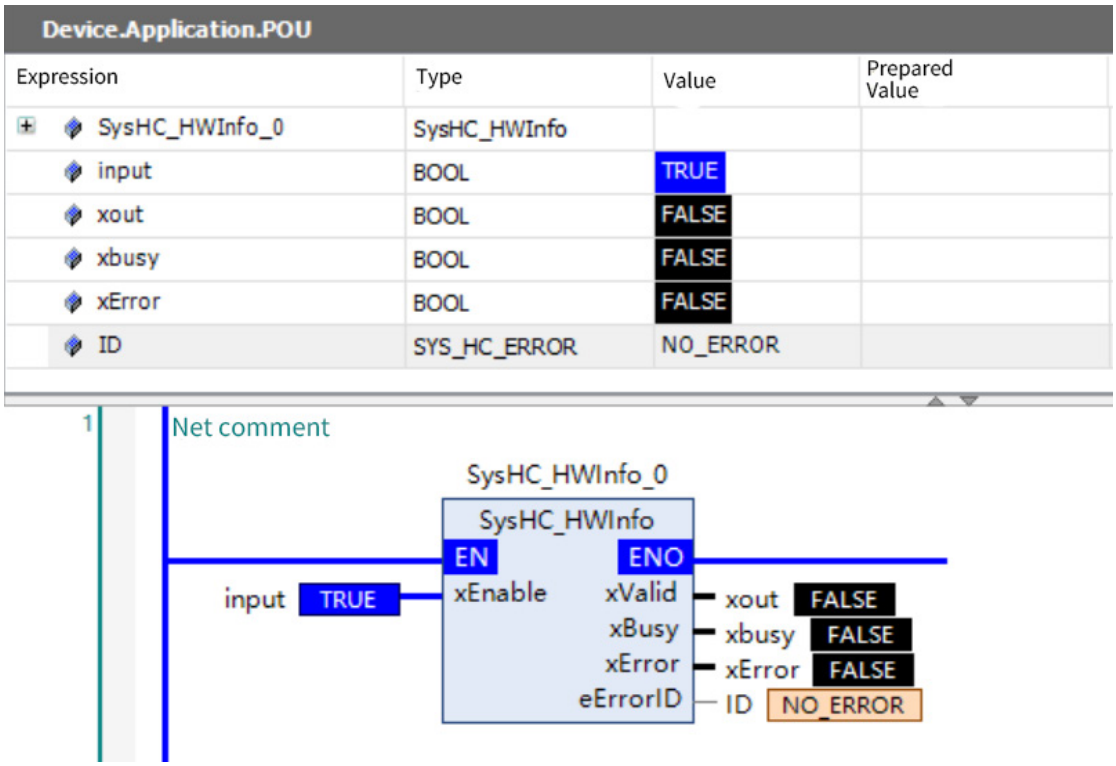
	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
xEnable	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
xValid	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
xBusy	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
xError	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
eErrorID	SYS_HC_ERROR																			

■ Program example

ST



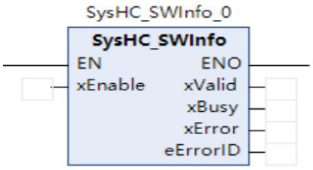
LD



4.10. 3 SysHC_SWInfo

This instruction saves the boot version, diagnosis information, PLC version, and runtime version to internal attribute variables.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
SysHC_SWInfo	Get PLC software information	FB		SysHC_SWInfo(xEnable: , xValid=> , xBusy=> , xError=> , eErrorID=>);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
xEnable	Active high	BOOL	0 to 1	0	Active high

Output variables

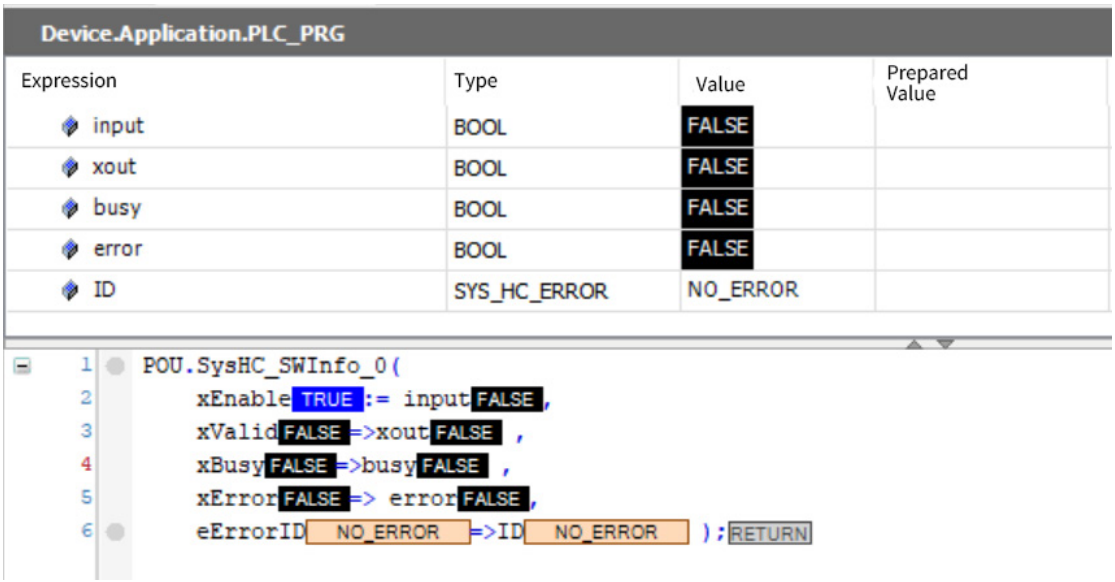
Output Variable	Name	Data Type	Value Range	Initial Value	Description
xValid	Output active	BOOL	0 to 1	0	Output active
xBusy	Executing the instruction	BOOL	0 to 1	0	Executing the instruction
xError	Error	BOOL	0 to 1	0	Error

Output Variable	Name	Data Type	Value Range	Initial Value	Description
eErrorID	Error ID	SYS_HC_ERROR	-	NO_ERROR	Error ID

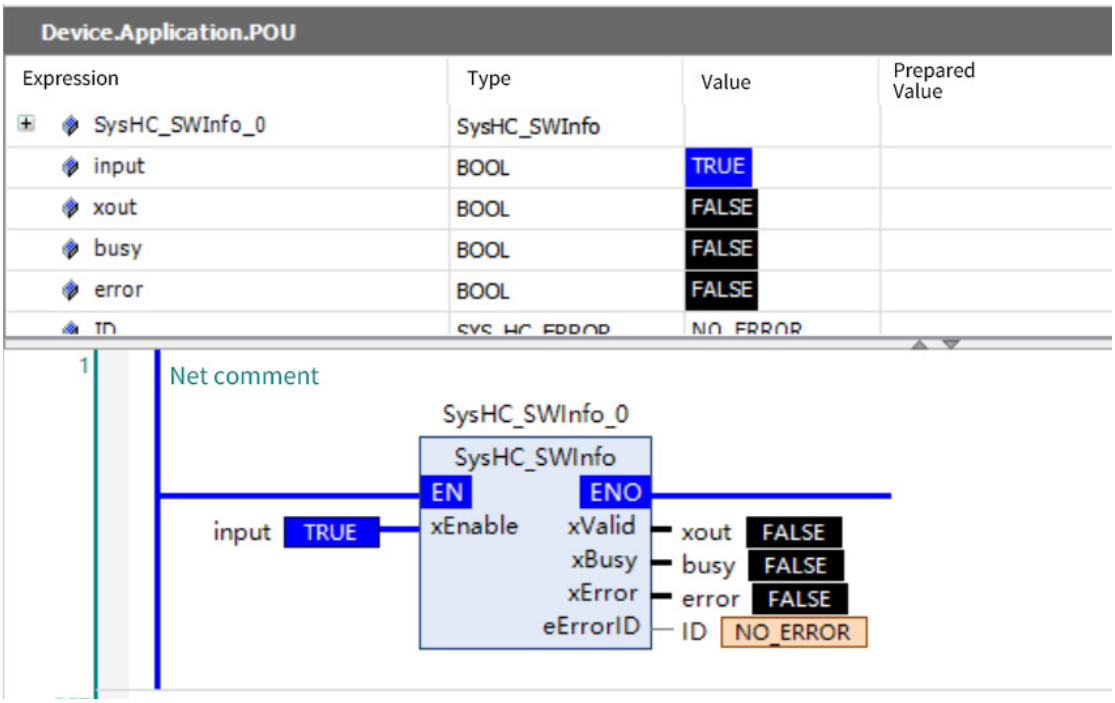
	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
xEnable	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
xValid	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
xBusy	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
xError	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
eErrorID	SYS_HC_ERROR																			

■ Program example

ST



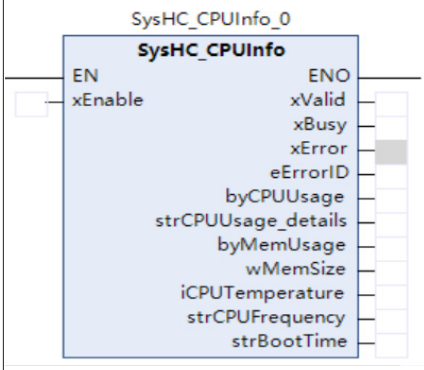
LD



4.10. 4 SysHC_CPUInfo

This instruction gets the CPU memory, primary frequency, and temperature and saves them to corresponding variables.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
SysHC_CPUInfo	Get the CPU, memory, and boot time information of the PLC	FB		SysHC_CPUInfo(xEnable:= , xValid=> , xBusy=> , xError=> , eErrorID=> , byCPUUsage=> , strCPUUsage_details=> , byMemUsage=> , wMemSize=> , iCPUTemperature=> , strCPUFrequency=> , strBootTime=>);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
xEnable	Active high	BOOL	0 to 1	0	Active high

Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
xValid	Output active	BOOL	0 to 1	0	Output active
xBusy	Executing the instruction	BOOL	0 to 1	0	Executing the instruction
xError	Error	BOOL	0 to 1	0	Error
eErrorID	Error ID	SYS_HC_ERROR	-	NO_ERROR	Error ID
byCPUUsage	CPU load rate	BYTE	0 to 255	0	CPU load rate
strCPUUsage_details	Separate CPU load rate	STRING	-	''	Separate CPU load rate
byMemUsage	Memory utilization	BYTE	0 to 255	0	Memory utilization
wMemSize	Memory size	WORD	0 to 65535	0	Memory size
iCPUTemperature	CPU temperature	BYTE	0 to 255	0	CPU temperature
strCPUFrequency	Primary frequency of CPU	STRING	-	''	Primary frequency of CPU
strBootTime	Startup time	STRING	-	''	Startup time

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
xEnable	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
xValid	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
xBusy	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
xError	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
eErrorID	SYS_HC_ERROR																			
byCPUUsage	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
strCPUUsage_details	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√
byMemUsage	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
wMemSize	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
iCPUTemperature	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
strCPUFrequency	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√
strBootTime	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√

■ Program example

ST

Device.Application.PLC_PRG

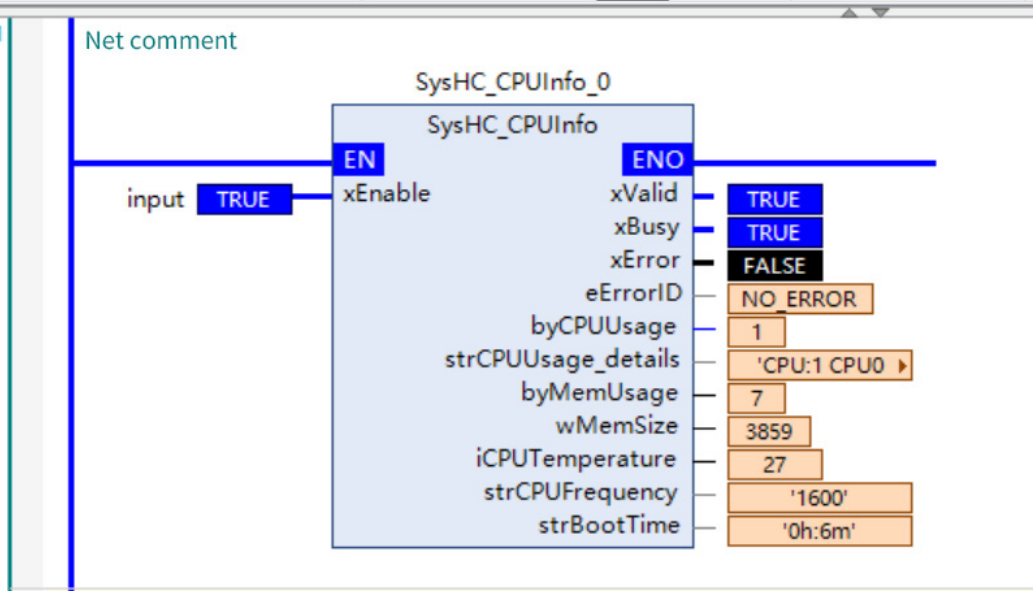
Expression	Type	Value	Prepared Value
task	ARRAY [1..10] OF T...		
input	BOOL	TRUE	
error	BOOL	FALSE	
busy	BOOL	TRUE	

```
1
2  POU.SysHC_CPUInfo_0(
3      xEnable:=input,
4      xValid=>xout,
5      xBusy=>busy,
6      xError=>error,
7      eErrorID=>ID,
8      byCPUUsage=> sage,
9      strCPUUsage_details=>details,
10     byMemUsage=> memusage,
11     wMemSize=>size,
12     iCPUTemperature=>temperature,
13     strCPUFrequency=>frequency,
14     strBootTime=> boottime);
15 RETURN
```

LD

Device.Application.POU

Expression	Type	Value	Prepared Value
xout	STRING	''	
busy	BOOL	FALSE	
error	BOOL	FALSE	

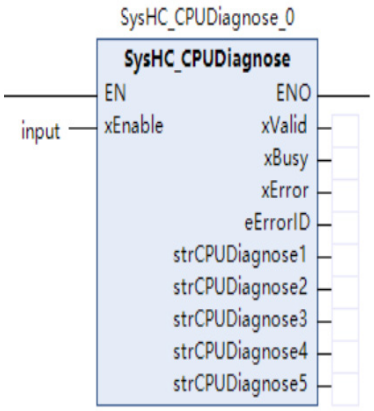


■ Precautions

AM600 does not provide the temperature detection function.

4.10. 5 SysHC_CPUDiagnose

This instruction gets the fault diagnosis information of the PLC.

Instruction	Name	FB/FC	LD Expression	ST Expression
SysHC_CPUDiagnose	Get the CPU hardware diagnosis information	FB		SysHC_CPUDiagnose(xEnable:= , xValid=> , xBusy=> , xError=> , eErrorID=> , strCPUDiagnose1=> , strCPUDiagnose2=> , strCPUDiagnose3=> , strCPUDiagnose4=> , strCPUDiagnose5=>);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
xEnable	Active high	BOOL	0 to 1	0	Active high

Output variables

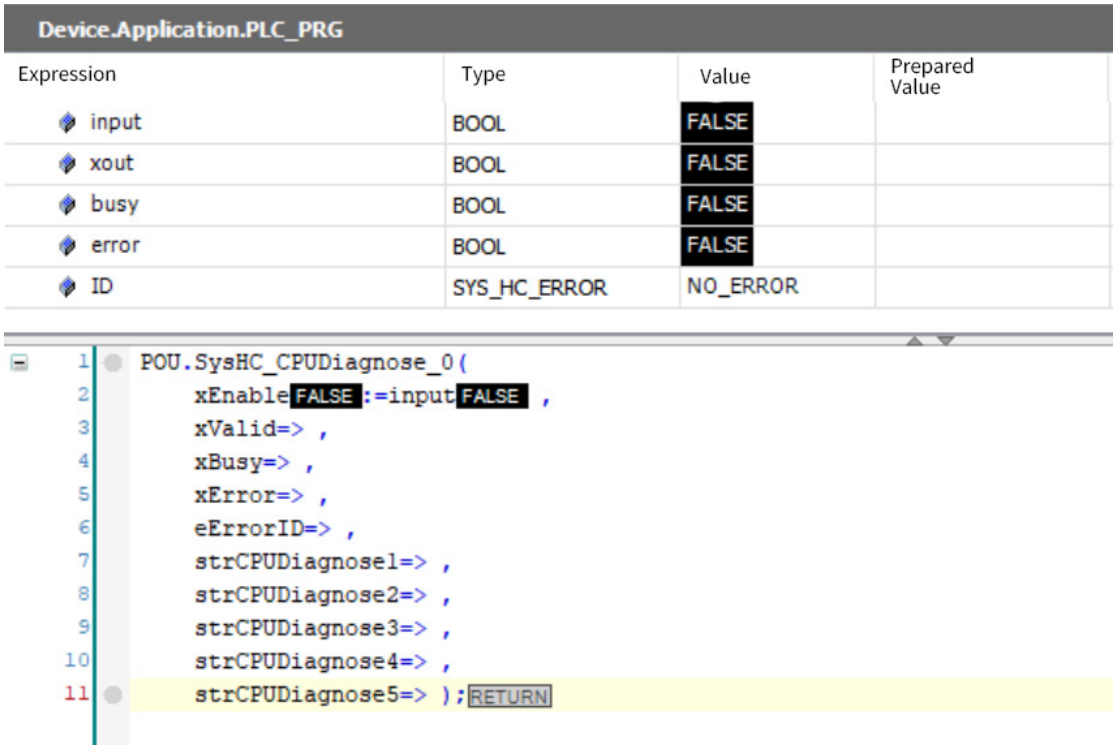
Output Variable	Name	Data Type	Value Range	Initial Value	Description
xValid	Output active	BOOL	0 to 1	0	Output active
xBusy	Executing the instruction	BOOL	0 to 1	0	Executing the instruction
xError	Error	BOOL	0 to 1	0	Error
eErrorID	Error ID	SYS_HC_ERROR	-	NO_ERROR	Error ID
strCPUDiagnose1	Diagnosis information 1	STRING	-	''	Diagnosis information 1
strCPUDiagnose2	Diagnosis information 2	STRING	-	''	Diagnosis information 2
strCPUDiagnose3	Diagnosis information 3	STRING	-	''	Diagnosis information 3
strCPUDiagnose4	Diagnosis information 4	STRING	-	''	Diagnosis information 4
strCPUDiagnose5	Diagnosis information 5	STRING	-	''	Diagnosis information 5

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
xEnable	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

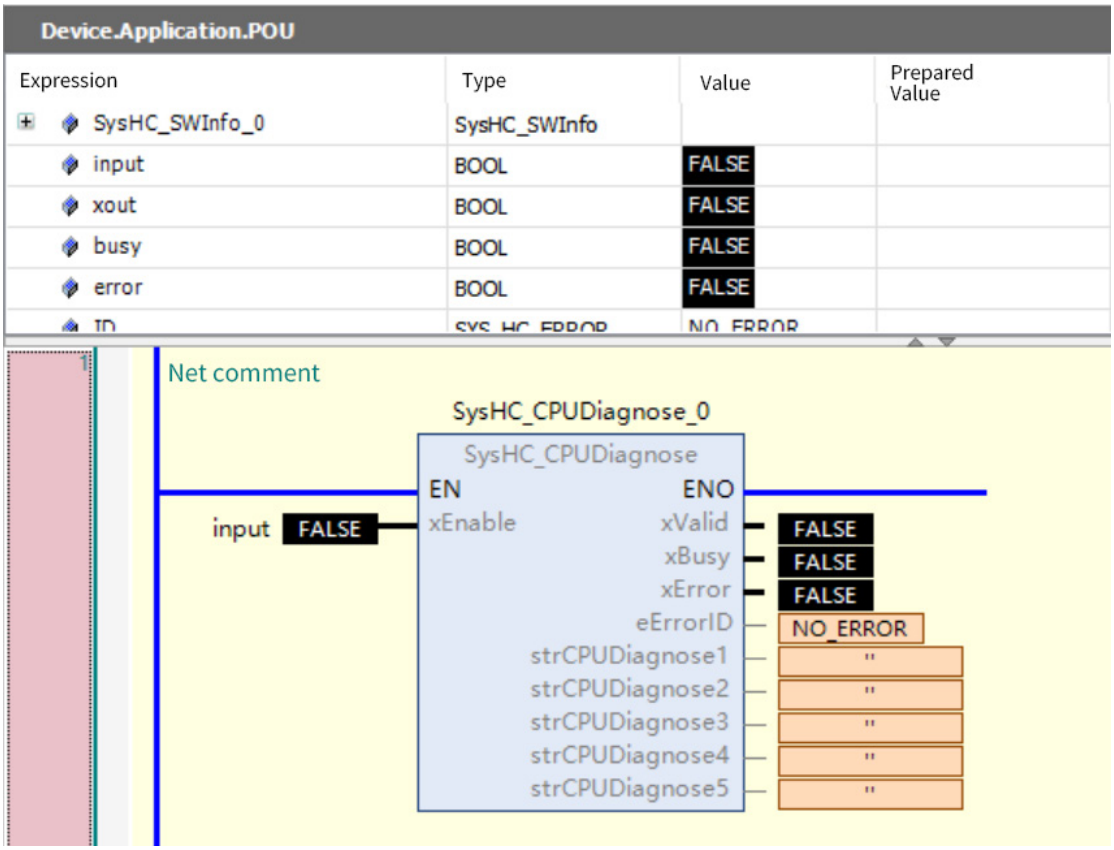
	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
xValid	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
xBusy	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
xError	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
eErrorID	SYS_HC_ERROR																			
strCPUDiagnose1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√
strCPUDiagnose2	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√
strCPUDiagnose3	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√
strCPUDiagnose4	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√
strCPUDiagnose5	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√

■ Program example

ST



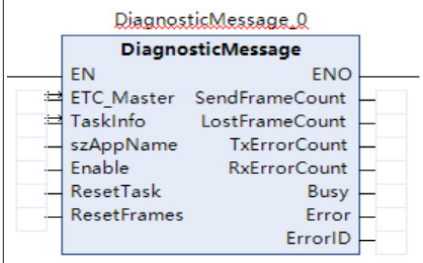
LD



4.10. 6 DiagnosticMessage

This instruction gets the current CPU diagnostic message.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
DiagnosticMessage	Get CPU diag-nostic message	FB		DiagnosticMessage(ETC_Master:= , TaskInfo:= , szAppName:= , Enable:= , ResetTask:= , ResetFrames:= , SendFrameCount=> , LostFrameCount=> , TxErrorCount=> , RxErrorCount=> , Busy=> , Error=> , ErrorID=>);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
ETC_Master	EtherCAT task	IoDrvEtherCAT	-	-	EtherCAT task

Input Variable	Name	Data Type	Value Range	Initial Value	Description
TaskInfo	Task information	ARRAY[1 to 10] OF Task_info2	-	-	Task information
szAppName	PLC application name	STRING	-	'Application'	PLC application name
Enable	Function block enable, with level active	BOOL	0 to 1	0	Function block enable, with level active
ResetTask	Reset task data, activated at the rising edge	BOOL	0 to 1	0	Reset task data, activated at the rising edge
ResetFrames	Reset EtherCAT frame diagnostic data, activated at the rising edge	BOOL	0 to 1	0	Reset EtherCAT frame diagnostic data, activated at the rising edge

Output variables

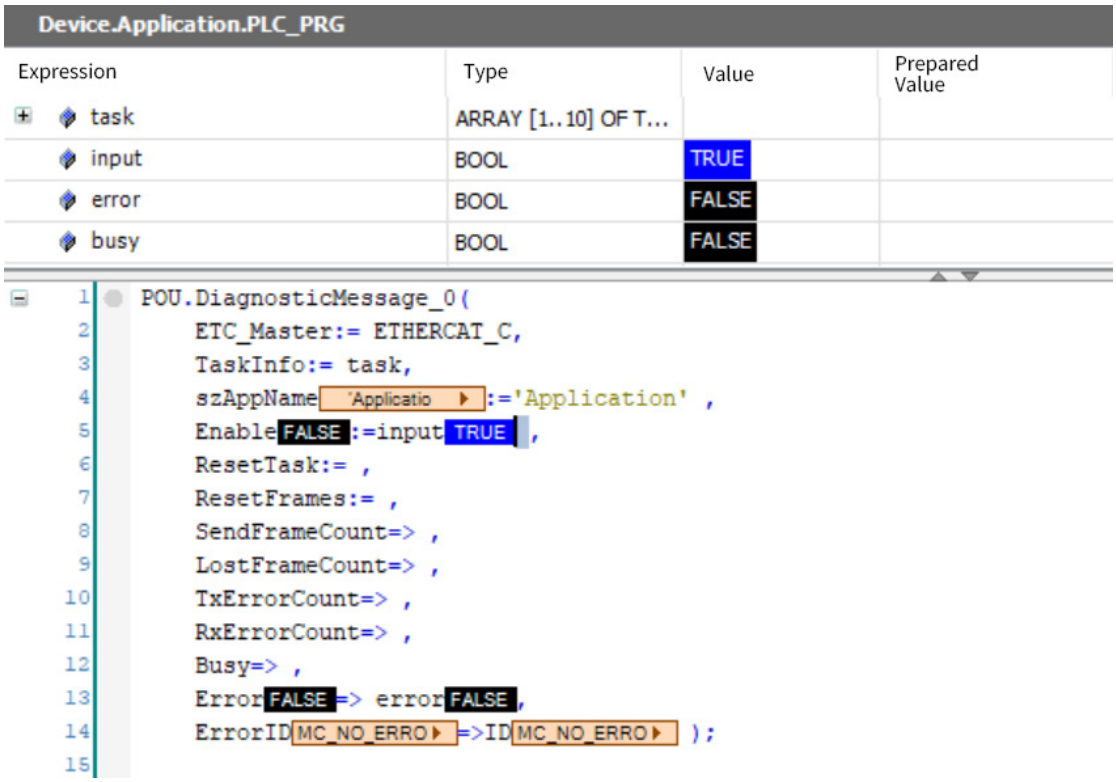
Output Variable	Name	Data Type	Value Range	Initial Value	Description
SendFrameCount	Send frame data	UDINT	0 to 4294967295	0	Output active
LostFrameCount	Lost frame count	UDINT	0 to 4294967295	0	Executing the instruction
TxErrorCount	Frame sending error count	UDINT	0 to 4294967295	0	Error
RxErrorCount	Frame receiving error count	UDINT	0 to 4294967295	0	Error ID
xBusy	Executing the instruction	BOOL	0 to 1	0	Output active
xError	Error	BOOL	0 to 1	0	Executing the instruction
eErrorID	Error ID	SYS_HC_ERROR	-	NO_ERROR	Error

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
ETC_Master	IoDrvEtherCAT																			
TaskInfo	ARRAY[1 to 10] OF Task_info2																			
szAppName	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√
Enable	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
ResetTask	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
ResetFrames	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
SendFrameCount	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-
LostFrameCount	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-

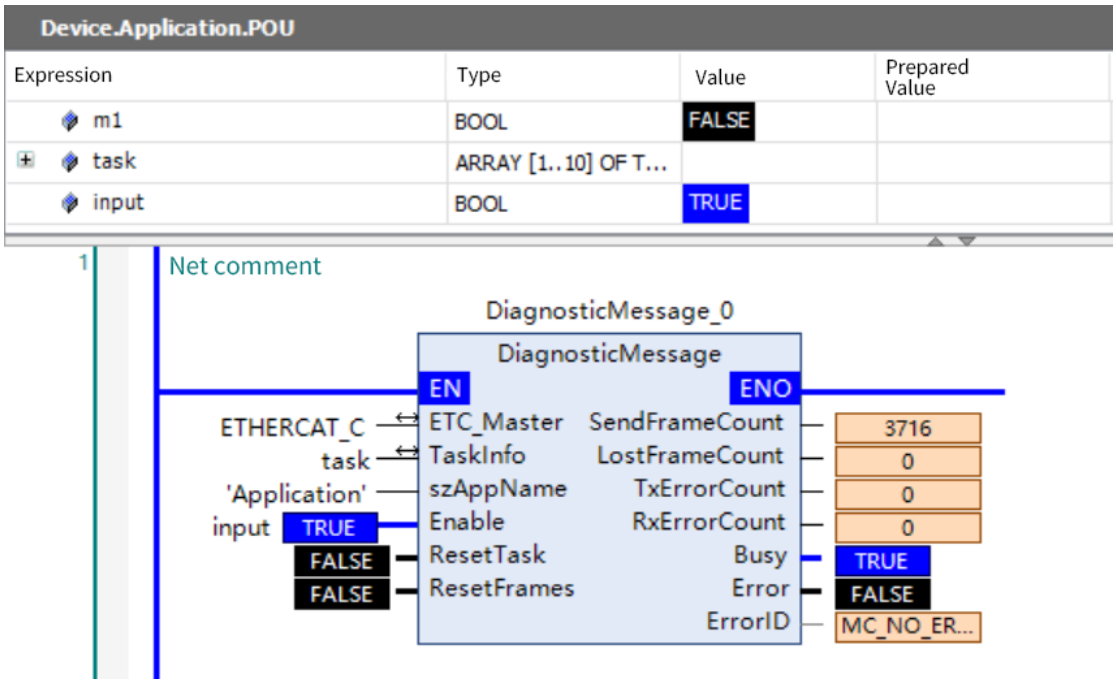
	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
TxErrorCount	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-
RxErrorCount	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-
xBusy	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
xError	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
eErrorID	SYS_HC_ERROR																			

■ Program example

ST



LD



■ Precautions

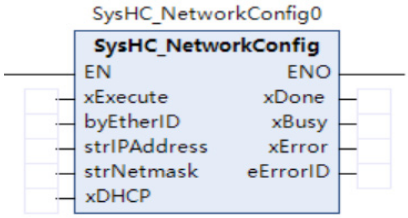
Taskinfo contains information about a maximum of 10 tasks. Each task contains information in the following figure.

Expression	Type	Value	Prepared Value	Address	Comment
task[1]	Task_Info2				
dwVersion	DWORD	2			[c] Version of the structure (2)
pszName	POINTER TO STRING	16#00007F5357...			[c] Name of the task, mandatory
pszName^	STRING	'ETHERCAT_C'			[c] Name of the task, mandatory
nPriority	INT	0			[c] IEC priority of the task (0..31)
KindOfTask	INT	0			[c] Kind of task. See c...sponding category |...
bWatchdog	BOOL	FALSE			[c] Is TRUE, if watchdog is enabled
bProfiling	BOOL	FALSE			[c] Is TRUE, if profilin...nabled (not used, F...
dwEventFunctionPointer	POINTER TO BYTE	16#0000000000...			[c] Function pointer to...e event check routin...
pszExternalEvent	POINTER TO STRING	16#0000000000...			[c] Name of the event, if it is an external eve...
dwTaskEntryFunctionPoin...	POINTER TO BYTE	16#00007F5357...			[c] Function pointer to...e task code, mandat...
dwWatchdogSensitivity	DWORD	1			[c] Watchdog sensitivity
dwInterval	DWORD	1000			[c] Interval in microseconds
dwWatchdogTime	DWORD	0			[c] Watchdog time in microseconds
dwCycleTime	DWORD	10			[s] Cycle time in micro...nds (last execution ...
dwAverageCycleTime	DWORD	8			[s] Average cycle time in microseconds
dwMaxCycleTime	DWORD	44			[s] Maximum cycle time in microseconds
dwMinCycleTime	DWORD	1			[s] Minumum cycle time in microseconds
iJitter	DINT	153			[s] Jitter in microseconds

4.10. 7 SysHC_NetworkConfig

This instruction configures PLC network parameters. It configures the IP address of the PLC network port, subnet mask, and dynamic IP address by using external libraries.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
SysHC_NetworkConfig	Config-ure PLC network ports	FB		SysHC_NetworkConfig(xExecute:= , byEtherID:= , strIPAddress:= , strNetmask:= , xDHCP:= , xDone=> , xBusy=> , xError=> , eErrorID=>);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
xExecute	Active high	BOOL	0 to 1	0	Active high
byEtherID	Network port ID	BYTE	0 to 255	0	Network port ID
strIPAddress	IP Address	STRING	-	'192.168.1.88'	IP Address
strNetmask	Subnet mask address	STRING	-	255.255.255.0	Subnet mask address
xDHCP	DHCP attribute	BOOL	0 to 1	0	DHCP attribute

Output variables

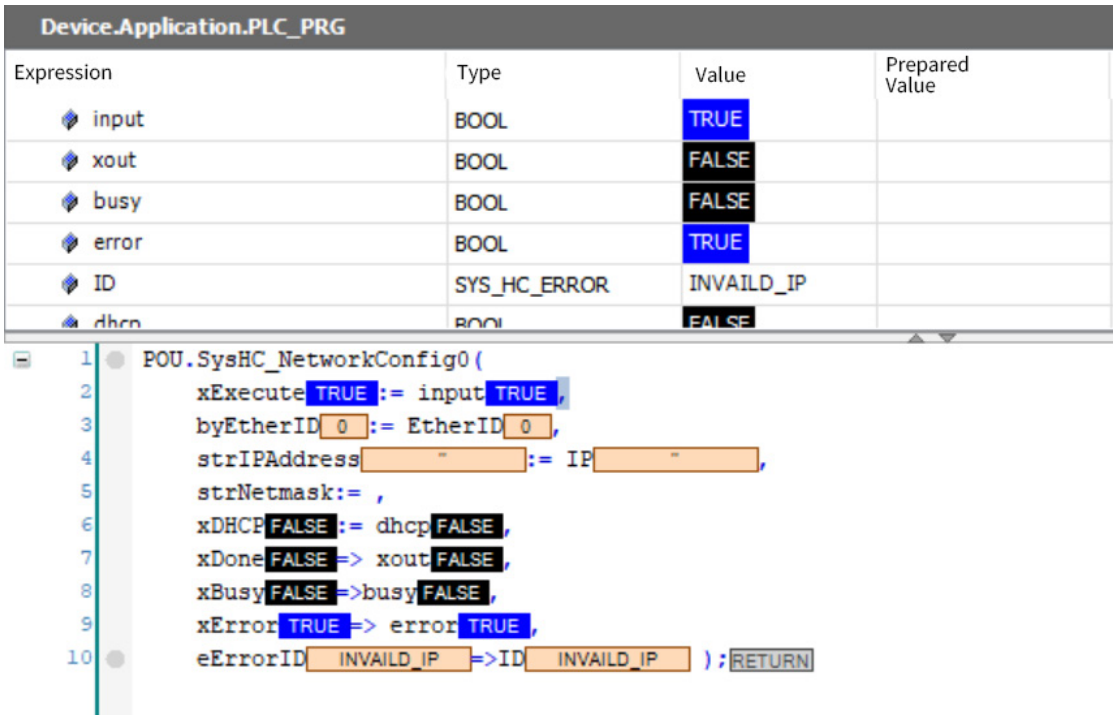
Output Variable	Name	Data Type	Value Range	Initial Value	Description
xDone	Output active	BOOL	0 to 1	0	Output active
xBusy	Executing the instruction	BOOL	0 to 1	0	Executing the instruction
xError	Error	BOOL	0 to 1	0	Error
eErrorID	Error ID	SYS_HC_ERROR	-	NO_ERROR	Error ID

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
xExecute	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
byEtherID	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
strIPAddress	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√
strNetmask	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√
xDHCP	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
xDone	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
xBusy	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
xError	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

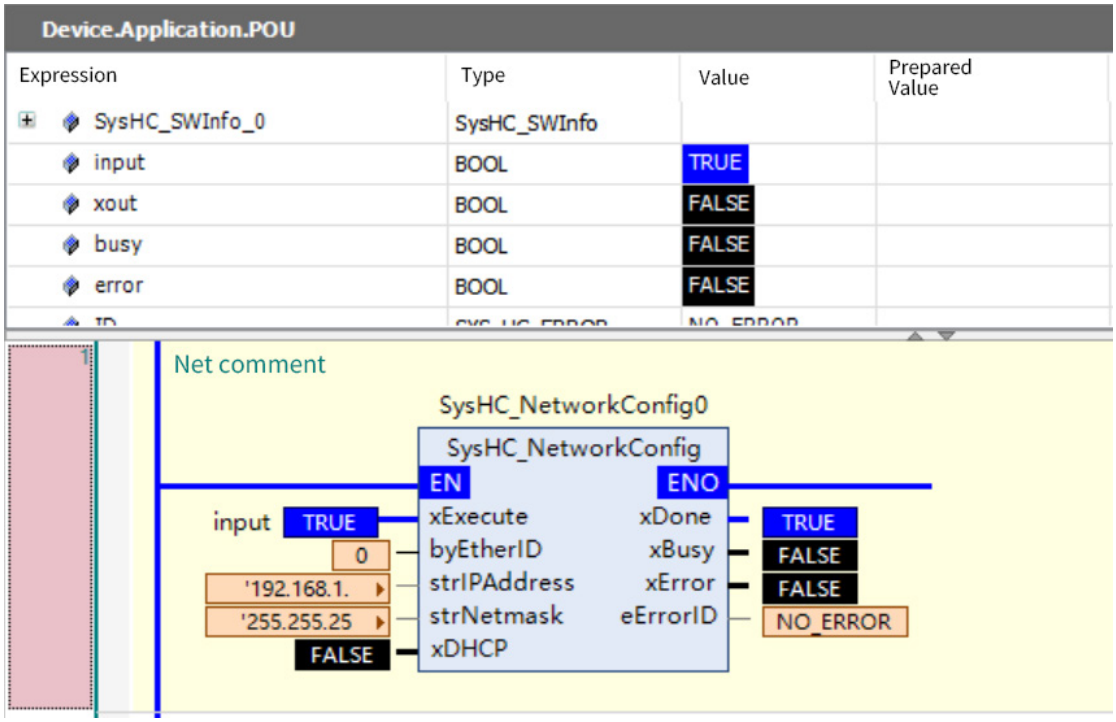
	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
eErrorID	SYS_HC_ERROR																			

■ Program example

ST



LD



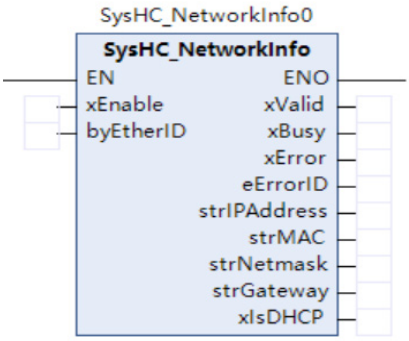
■ Precautions

This instruction cannot be used for ETC tasks.

4.10. 8 SysHC_NetworkInfo

This instruction gets PLC network parameters. It configures the IP address of the PLC network port, subnet mask, dynamic IP address, gateway, and physical address of the network card by using external libraries.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
SysHC_NetworkInfo	Obtain PLC network configuration	FB		SysHC_NetworkInfo(xEnable:= , byEtherID:= , xValid=> , xBusy=> , xError=> , eErrorID=> , strIPAddress=> , strMAC=> , strNetmask=> , strGateway=> , xIsDHCP=>);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
xEnable	Active high	BOOL	0 to 1	0	Active high
byEtherID	Network port ID	BYTE	0 to 255	0	Network port ID

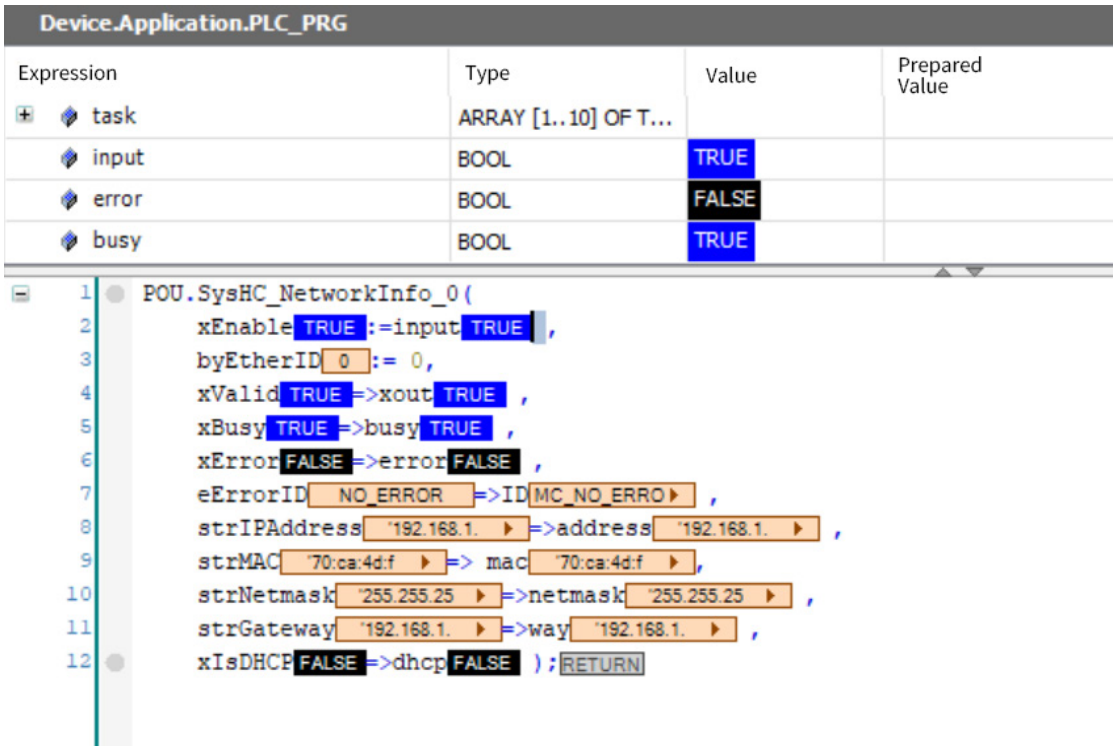
Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
xValid	Output active	BOOL	0 to 1	0	Output active
xBusy	Executing the instruction	BOOL	0 to 1	0	Executing the instruction
xError	Error	BOOL	0 to 1	0	Error
eErrorID	Error ID	SYS_HC_ERROR	-	NO_ERROR	Error ID
strIPAddress	IP Address	STRING	-	'192.168.1.88'	IP Address
strMAC	Stored MAC address	STRING	-	''	Stored MAC address
strNetmask	Subnet mask address	STRING	-	255.255.255.0	Subnet mask address
strGateway	Stored gateway information	STRING	-	''	Stored gateway information
xIsDHCP	DHCP status	STRING	-	''	DHCP status

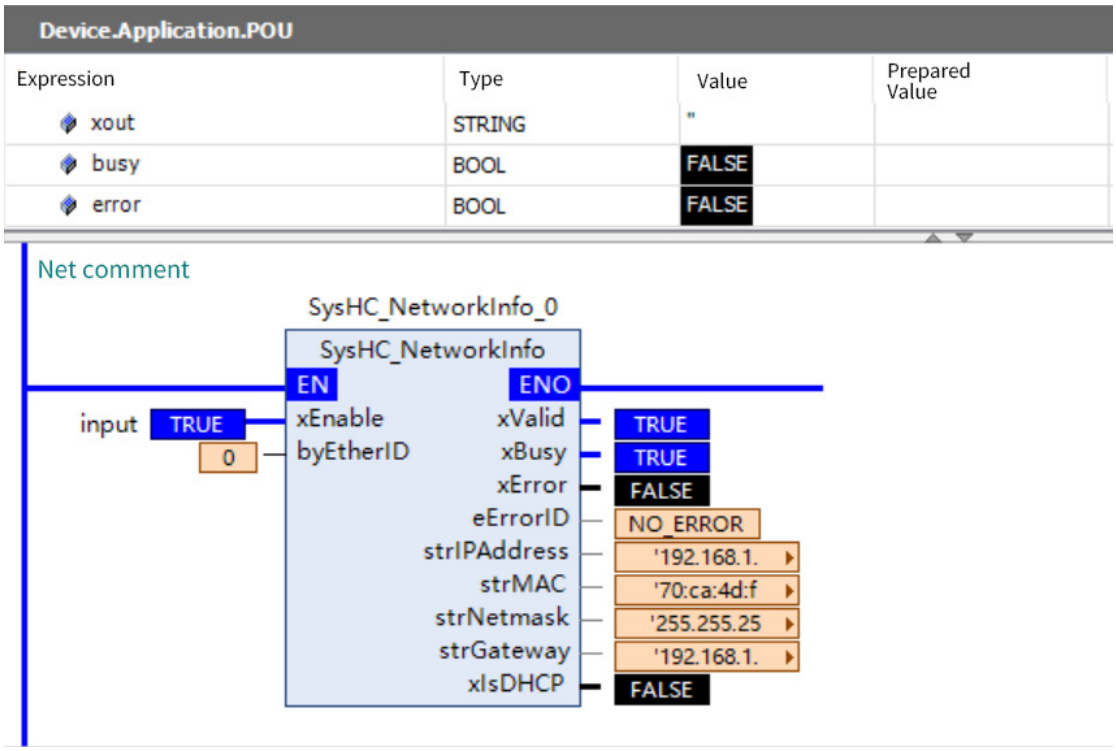
	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
xEnable	√	-	-	-	-	-	-	-	-	-	-	-	-	_	_	-	-	-	-	-
byEtherID	-	√	-	-	-	-	-	-	-	-	-	-	-	_	_	-	-	-	-	-
xValid	√	-	-	-	-	-	-	-	-	-	-	-	-	_	_	-	-	-	-	-
xBusy	√	-	-	-	-	-	-	-	-	-	-	-	-	_	_	-	-	-	-	-
xError	√	-	-	-	-	-	-	-	-	-	-	-	-	_	_	-	-	-	-	-
eErrorID	SYS_HC_ERROR																			
strIPAddress	-	-	-	-	-	-	-	-	-	-	-	-	-	_	_	-	-	-	-	√
strMAC	-	-	-	-	-	-	-	-	-	-	-	-	-	_	_	-	-	-	-	√
strNetmask	-	-	-	-	-	-	-	-	-	-	-	-	-	_	_	-	-	-	-	√
strGateway	-	-	-	-	-	-	-	-	-	-	-	-	-	_	_	-	-	-	-	√
xIsDHCP	-	-	-	-	-	-	-	-	-	-	-	-	-	_	_	-	-	-	-	√

■ Program example

ST



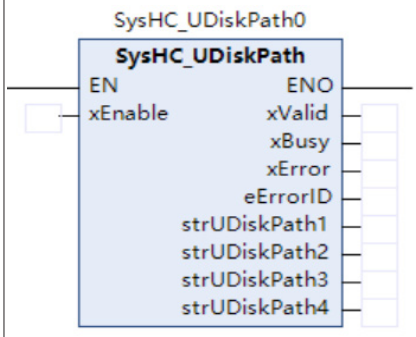
LD



4.10. 9 SysHC_UDiskPath

This instruction obtains the path of the USB flash disk.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
SysHC_UDiskPath	Obtain the path of the U disk	FB		SysHC_UDiskPath(xEnable:= , xValid=> , xBusy=> , xError=> , eErrorID=> , strUDiskPath1=> , strUDiskPath2=> , strUDiskPath3=> , strUDiskPath4=>)

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
xEnable	Active high	BOOL	0 to 1	0	Active high

Output variables

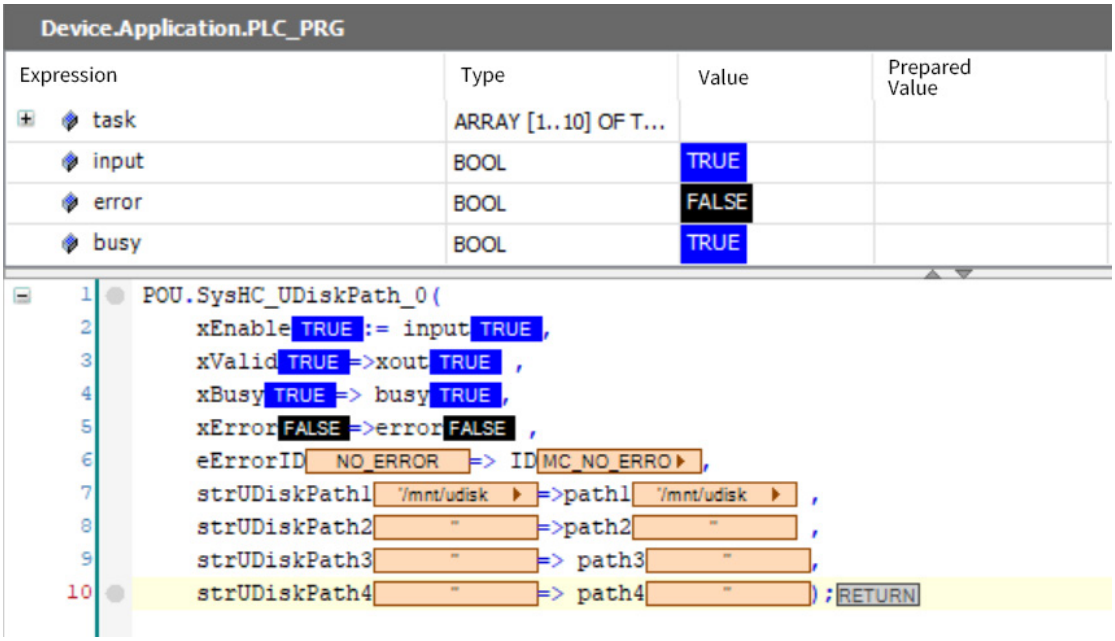
Output Variable	Name	Data Type	Value Range	Initial Value	Description
xValid	Output active	BOOL	0 to 1	0	Output active
xBusy	Executing the instruction	BOOL	0 to 1	0	Executing the instruction

Output Variable	Name	Data Type	Value Range	Initial Value	Description
xError	Error	BOOL	0 to 1	0	Error
eErrorID	Error ID	SYS_HC_ERROR	-	NO_ERROR	Error ID
strUDiskPath1	USB flash disk 1	STRING	-	''	USB flash disk 1
strUDiskPath2	USB flash disk 2	STRING	-	''	USB flash disk 2
strUDiskPath3	USB flash disk 3	STRING	-	''	USB flash disk 3
strUDiskPath4	USB flash disk 4	STRING	-	''	USB flash disk 4

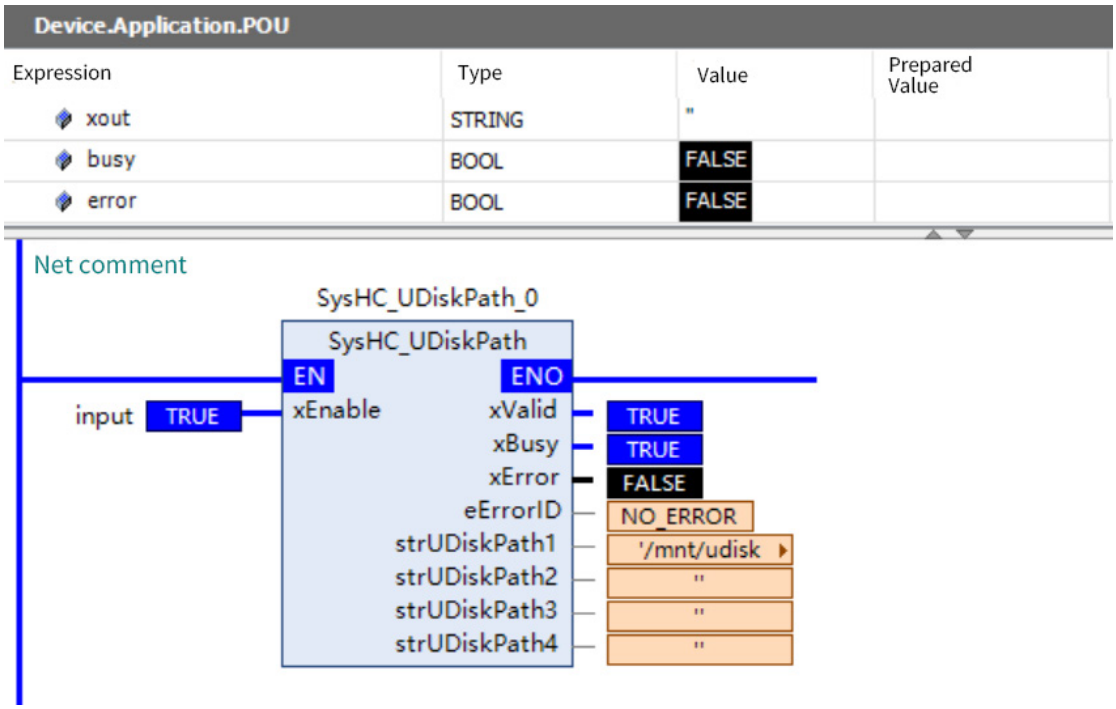
	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
xEnable	√	-	-	-	-	-	-	-	-	-	-	-	-	_	_	-	-	-	-	-
xValid	√	-	-	-	-	-	-	-	-	-	-	-	-	_	_	-	-	-	-	-
xBusy	√	-	-	-	-	-	-	-	-	-	-	-	-	_	_	-	-	-	-	-
xError	√	-	-	-	-	-	-	-	-	-	-	-	-	_	_	-	-	-	-	-
eErrorID	SYS_HC_ERROR																			
strUDiskPath1	-	-	-	-	-	-	-	-	-	-	-	-	-	_	_	-	-	-	-	√
strUDiskPath2	-	-	-	-	-	-	-	-	-	-	-	-	-	_	_	-	-	-	-	√
strUDiskPath3	-	-	-	-	-	-	-	-	-	-	-	-	-	_	_	-	-	-	-	√
strUDiskPath4	-	-	-	-	-	-	-	-	-	-	-	-	-	_	_	-	-	-	-	√

■ Program example

ST



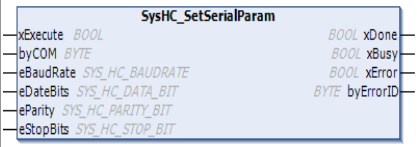
LD



4.10. 10 SysHC_SetSerialParam

This instruction sets parameters for the serial port free protocol.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
SysHC_SetSerialParam	Parameter setting for serial port free protocol	FB		SysHC_SetSerialParam(xExecute:= , byCOM:= , eBaudRate:= , eDateBits:= , eParity:= , eStopBits:= , xDone=> , xBusy=> , xError=> , byErrorID=>);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
xExecute	Function block enable	BOOL	[FALSE, TRUE]	FALSE	Function block enable
byCOM	COM number	BYTE	0 to 1	0	Value 0 indicates COM0 and value 1 indicates COM1.
eBaudRate	Baud rate	TYPE SYS_HC_BAUDRATE	[B4800, B9600, B19200, B38400, B57600, B115200]	B9600	Baud rate
eDateBits	Data bit	TYPE SYS_HC_DATA_BIT	BIT7 to BIT8	BIT8	Data bit

Input Variable	Name	Data Type	Value Range	Initial Value	Description
eParityBits	Parity bit	TYPE SYS_HC_PARITY_BIT	[NONE, ODD, EVEN]	NONE	Parity bit
eStopBits	Stop bit	TYPE SYS_HC_STOP_BIT	BIT1 to BIT2	BIT1	Stop bit

Output variables

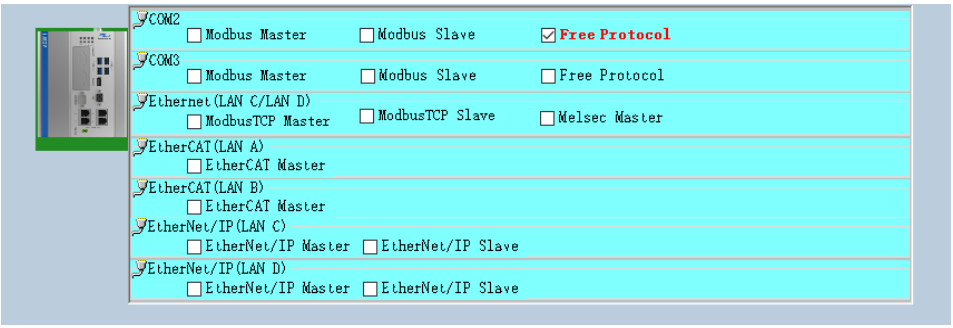
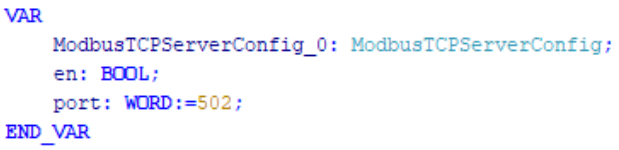
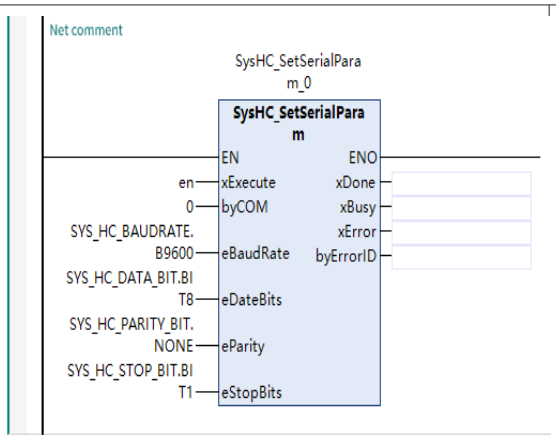
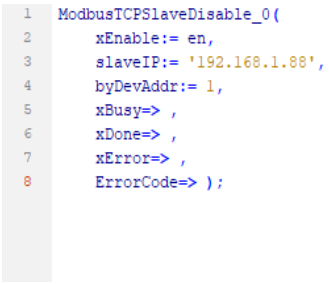
Output Variable	Name	Data Type	Value Range	Initial Value	Description
xDone	Output active	BOOL	[FALSE, TRUE]	-	Output active
xBusy	Executing the instruction	BOOL	[FALSE, TRUE]	-	Executing the instruction
xError	Execution error	BOOL	[FALSE, TRUE]	-	Execution error
byErrorID	Error code	BYTE	0 to 255	-	Error code

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
xExecute	√	-	-	-	-	-	-	-	-	-	-	-	-	_	_	-	-	-	-	-
byCOM	-	√	-	-	-	-	-	-	-	-	-	-	-	_	_	-	-	-	-	-
eBaudRate	-	-	-	-	-	-	-	-	-	-	-	-	-	_	_	-	-	-	-	-
eDateBits	-	-	-	-	-	-	-	-	-	-	-	-	-	_	_	-	-	-	-	-
eParityBits	-	-	-	-	-	-	-	-	-	-	-	-	-	_	_	-	-	-	-	-
eStopBits	-	-	-	-	-	-	-	-	-	-	-	-	-	_	_	-	-	-	-	-
xDone	√	-	-	-	-	-	-	-	-	-	-	-	-	_	_	-	-	-	-	-
xBusy	√	-	-	-	-	-	-	-	-	-	-	-	-	_	_	-	-	-	-	-
xError	-	-	-	-	-	-	-	-	-	-	-	-	-	_	_	-	-	-	-	-
byErrorID	-	√	-	-	-	-	-	-	-	-	-	-	-	_	_	-	-	-	-	-

■ Function

This function block sets communication parameters for the serial port free protocol.

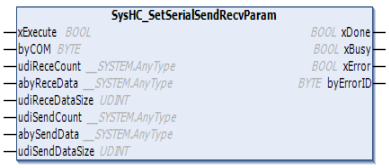
■ Program example

	LD	ST
Defined variable		
Program		

4.10. 11 SysHC_SetSerialSendRecvParam

This instruction sets send/receive parameters for the serial port free protocol.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
SysHC_SetSerialSendRecvParam	Send/Receive parameter setting for serial port free protocol	FB		SysHC_SetSerialSendRecvParam(xExecute:= , byCOM:= , udiReceCount:= , abyReceData:= , udiReceDataSize:= , udiSendCount:= , abySendData:= , udiSendDataSize:= , xDone=> , xBusy=> , xError=> , byErrorID=>);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
xExecute	Function block enable	BOOL	[FALSE, TRUE]	FALSE	Function block enable
byCOM	COM number	BYTE	0 to 1	0	COM number Value 0 indicates COM0 and value 1 indicates COM1.
udiReceCount	Number of bytes to receive	UDINT	By data type size	-	Number of bytes to receive
abyReceData	Cache to receive	ARRAY[*] OF BYTE	-	-	Cache to receive, which is the first element of an array
udiReceData Size	Size of bytes to receive	UDINT	By data type size	-	Size of bytes to receive, which cannot exceed the size of cache to receive
udiSendCount	Number of bytes to transmit	UDINT	By data type size		Number of bytes to transmit
abySendData	Cache to transmit	ARRAY[*] OF BYTE	-	-	Cache to transmit, which is the first element of an array
udiSendData Size	Size of bytes to transmit	UDINT	By data type size	-	Size of bytes to transmit, which cannot exceed the size of cache to transmit

Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
xDone	Output active	BOOL	[FALSE, TRUE]	-	Output active
xBusy	Executing the instruction	BOOL	[FALSE, TRUE]	-	Executing the instruction
xError	Execution error	BOOL	[FALSE, TRUE]	-	Execution error
byErrorID	Error code	BYTE	0 to 255	-	Error code

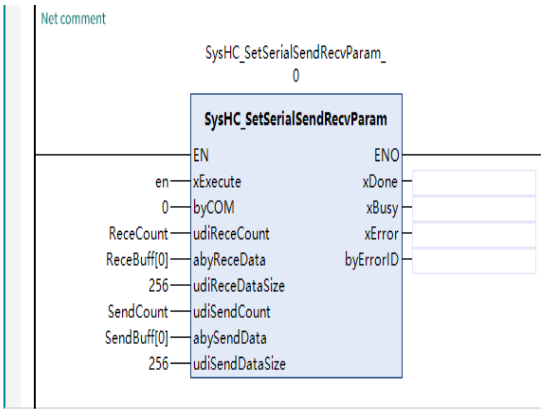
	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
xExecute	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
byCOM	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
udiReceCount	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-
abyReceData	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
udiReceData Size	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-
udiSendCount	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-
abySendData	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
udiSendDataSize	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
xDone	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
xBusy	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
xError	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
byErrorID	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

■ Function

- This function block sets send/receive parameters for the serial port free protocol.
- The instruction is not supported for PLCs of the AC series.

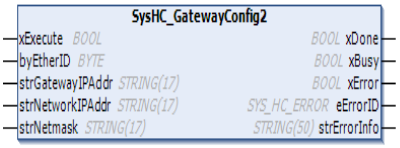
■ Program example

	LD	ST
Defined variable	 VAR SysHC_SetSerialSendRecvParam_0: SysHC_SetSerialSendRecvParam; en: BOOL; ReceCount: UDINT; ReceBuff:ARRAY[0..255] OF BYTE; SendCount: UDINT; SendBuff:ARRAY[0..255] OF BYTE; END_VAR	
Program		SysHC_SetSerialParam_0(xExecute:= en, byCOM:= 0, eBaudRate:= SYS_HC_BAUDRATE.B9600, eDateBits:= SYS_HC_DATA_BIT.BIT8, eParity:= SYS_HC_PARITY_BIT.NONE, eStopBits:= SYS_HC_STOP_BIT.BIT1, xDone=> , xBusy=> , xError=> , byErrorID=>);

4.10. 12 SysHC_GatewayConfig2

This instruction sets the gateway.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
SysHC_Gateway-Config2	Gateway setting	FB		SysHC_GatewayConfig2(xExecute:= , byEtherID:= , strGatewayIPAddr:= , strNetworkIPAddr:= , strNetmask:= xDone=> , xBusy=> , xError=> , eErrorID=> , strErrorInfo=>);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
xExecute	Function block enable	BOOL	[FALSE, TRUE]	FALSE	Function block enable
byEtherID	Network port ID	BYTE	0 to 1	-	Network port ID, which is determined by the number of EtherNET network ports. The value is 1 for the AM series and 0 to 1 for the AC series (network port A/B).
StrGateway IPAddr	Gateway IP address	STRING[17]	0.0.0.0 to 255.255.255.255	192.168.1.1	Gateway IP address
StrNetwork IPAddr	Target segment	STRING[17]	0.0.0.0 to 255.255.255.255	192.168.1.0	Address of specified segment to access (valid only for network port B)
StrNetmask	Subnet mask	STRING[17]	0.0.0.0 to 255.255.255.255	255.255.255.0	Subnet mask (valid only for network port B)

Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
xDone	Execution flag	BOOL	[FALSE, TRUE]	-	Output active
xBusy	Completion flag	BOOL	[FALSE, TRUE]	-	Executing the instruction
xError	Error flag	BOOL	[FALSE, TRUE]	-	Execution error
eErrorID	Error ID	SYS_HC_ERROT	Depends on data types	-	Error code
strErrorInfo	Error information	STRING[50]	Depends on data types	-	Error information

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
xExecute	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
byEtherID	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
StrGateway IPAddr	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√
StrNetwork IPAddr	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√
StrNetmask	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√
xDone	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
xBusy	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
xError	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
eErrorID	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
strErrorInfo	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√

■ Function

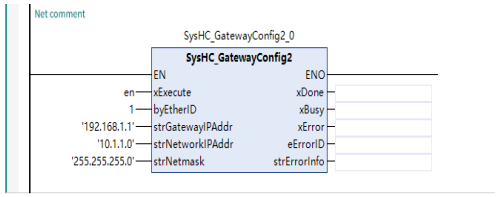
• This function block sets the default gateway and plaintext routing for the system A PLC of the AC series has two network ports, where network port A supports setting of the default gateway for external communication, while network port B supports setting of the plaintext routing for internal communication.

• The instruction is not supported for PLCs of the AM600 series.

■ Program example

	LD	ST
Defined variable		
Program		

```
VAR
    SysHC_GatewayConfig2_0: SysHC_GatewayConfig2;
    en: BOOL;
END_VAR
```

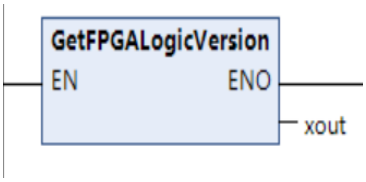



```
SysHC_GatewayConfig2_0(
    xExecute:= en,
    byEtherID:= 1,
    strGatewayIPAddr:= '192.168.1.1',
    strNetworkIPAddr:= '10.1.1.0',
    strNetmask:= '255.255.255.0',
    xDone=> ,
    xBusy=> ,
    xError=> ,
    eErrorID=> ,
    strErrorInfo=> );
```

4.10. 13 GetFPGALogicVersion

This instruction gets the FPGA logic version number.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
GetFPGALogicVersion	Get the FPGA logic version number	FC		xout:=GetFPGALogicVersion();

■ Variables

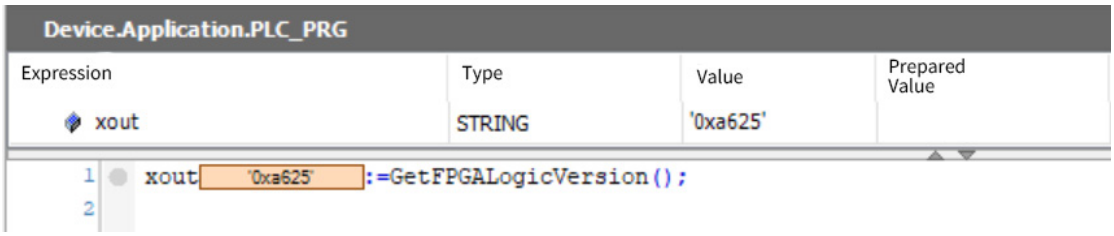
Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
xout	Storage variable	STRING	-	''	Storage variable

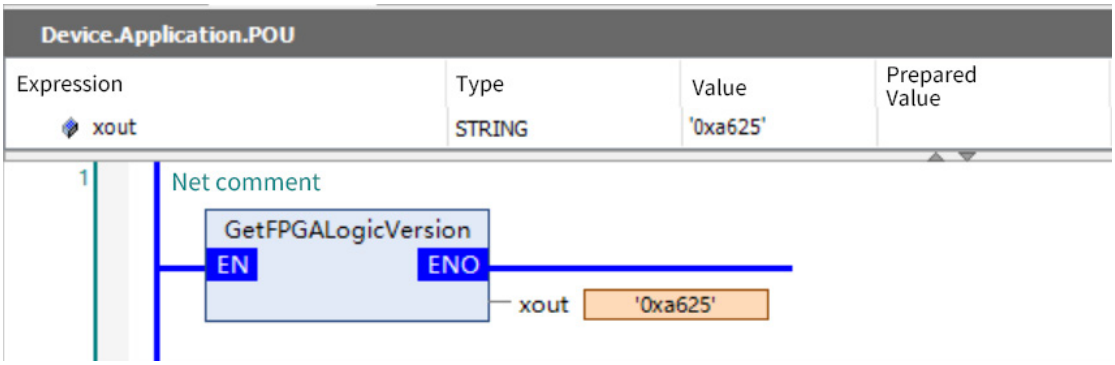
	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
xout	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√

■ Program example

ST



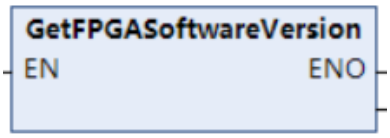
LD



4.10. 14 GetFPGASoftwareVersion

This instruction gets the FPGA software version number.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
GetFPGASoftwareVersion	Get the FPGA software version number	FC		GetFPGASoftwareVersion()

■ Variables

Output variables

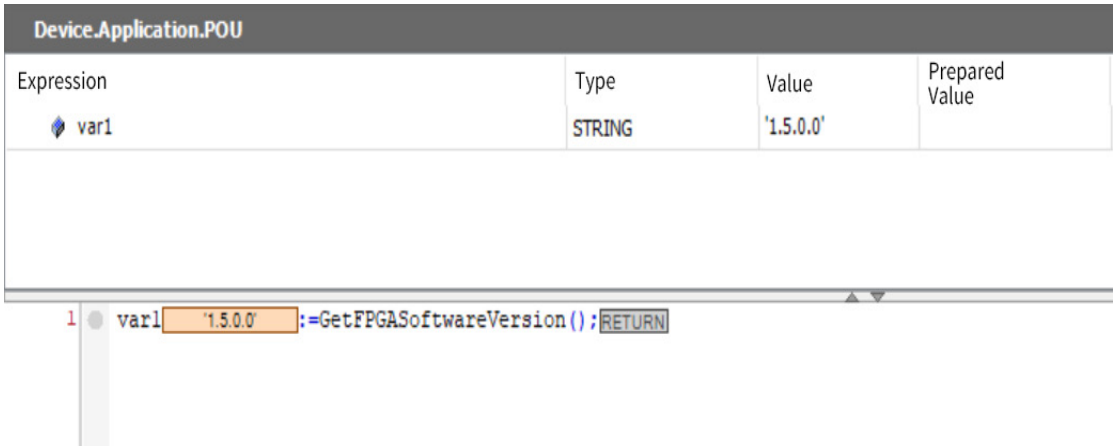
Output Variable	Name	Data Type	Value Range	Initial Value	Description
xout	Storage variable	STRING	-	''	Storage variable

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
xout	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√

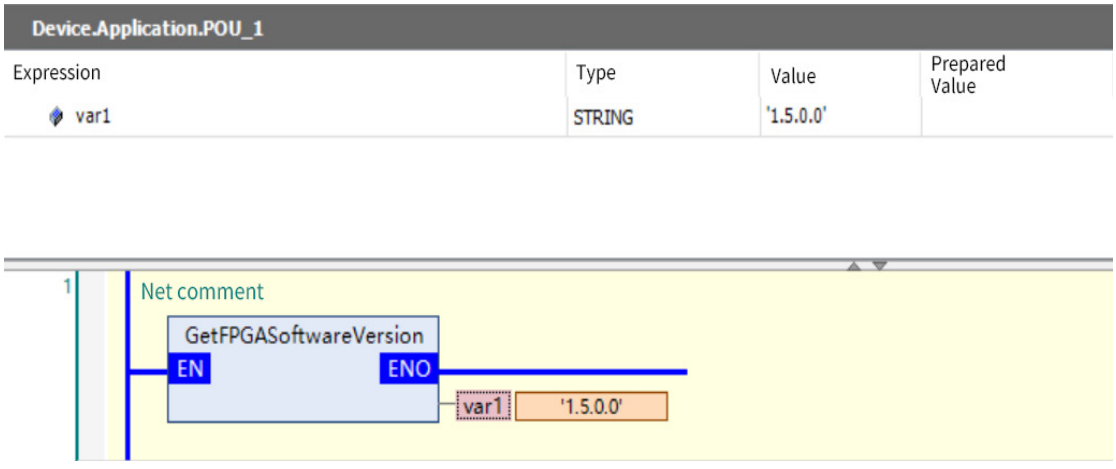
■ Program example

This instruction connects to the PLC to read the FPGA software version number.

ST



LD



■ Precautions

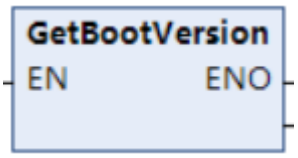
1) The type of the output data is STRING.

2) In case of direct simulation without connection to the PLC, no data is output.

4.10. 15 GetBootVersion

This instruction gets the boot version.

■ Instruction format

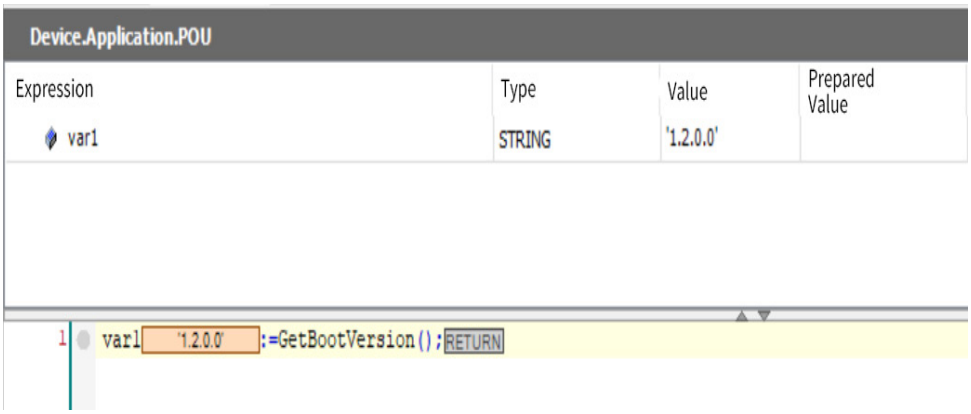
Instruction	Name	FB/FC	LD Expression	ST Expression
GetBootVersion	Get the boot version	FC		GetBootVersion()

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
xout	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√

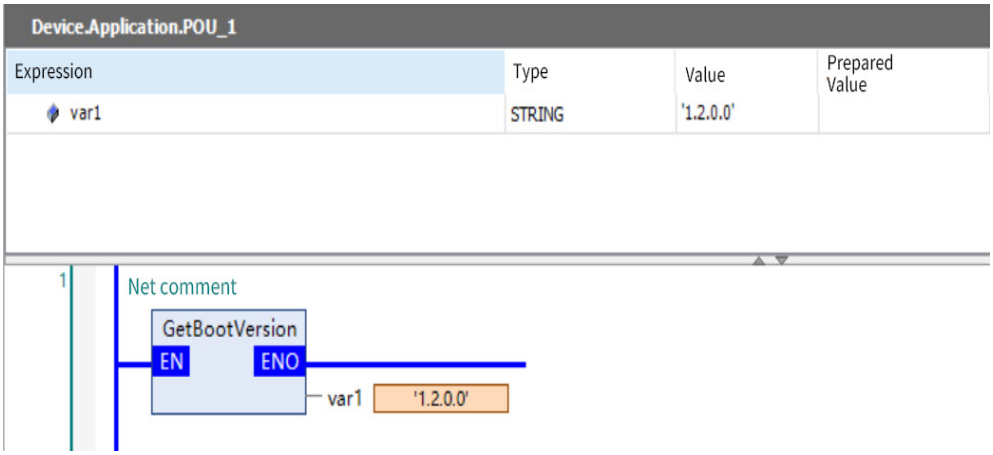
■ Program example

This instruction connects to the PLC to read the Boot software version number.

ST



LD



■ Precautions

The type of the output data is STRING.

4.10. 16 GetPLCVersion

This instruction gets the PLC version.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
GetPLCVersion	Get the PLC firmware version	FC	GetPLCVersion EN ENO	GetPLCVersion()

■ Variables

Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
Out	Output result	STRING	Depends on data types	-	Output result

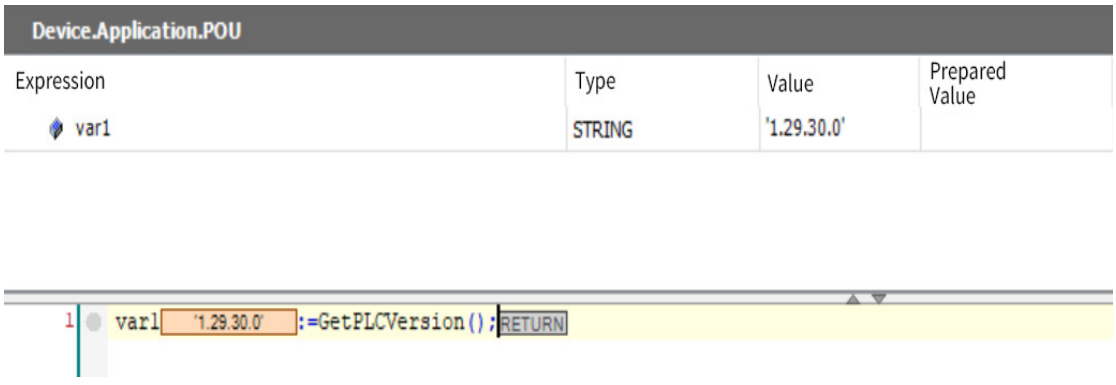
	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
out	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√

■ Function

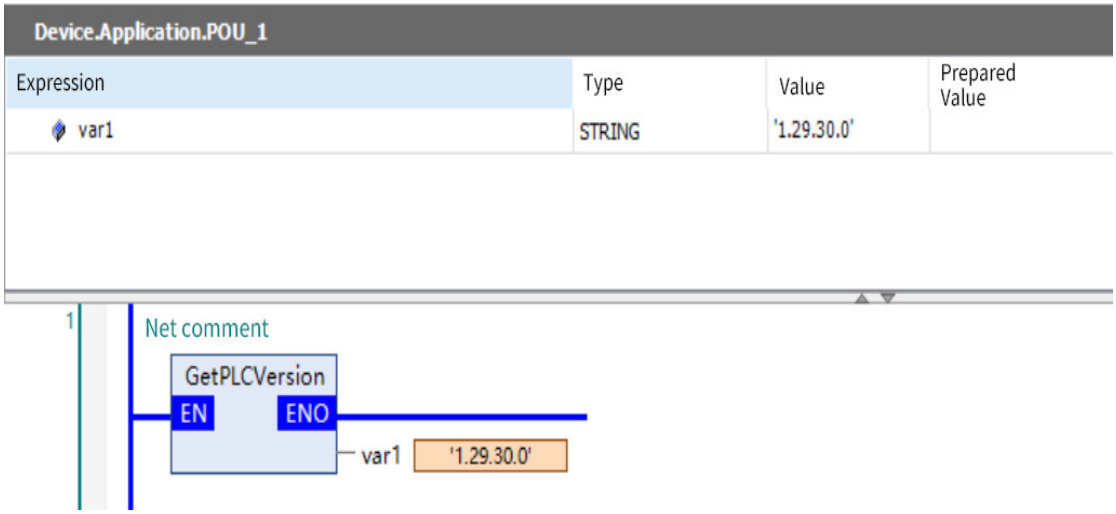
This instruction reads the PLC software version.

■ Program example

ST



LD



4.10. 17 GetProductName

This instruction gets the PLC name.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
GetProductName	Get the PLC name	FC	GetProductName EN ENO	GetProductName ()

■ Variables

Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
Out	Output result	STRING	Depends on data types	-	Output result

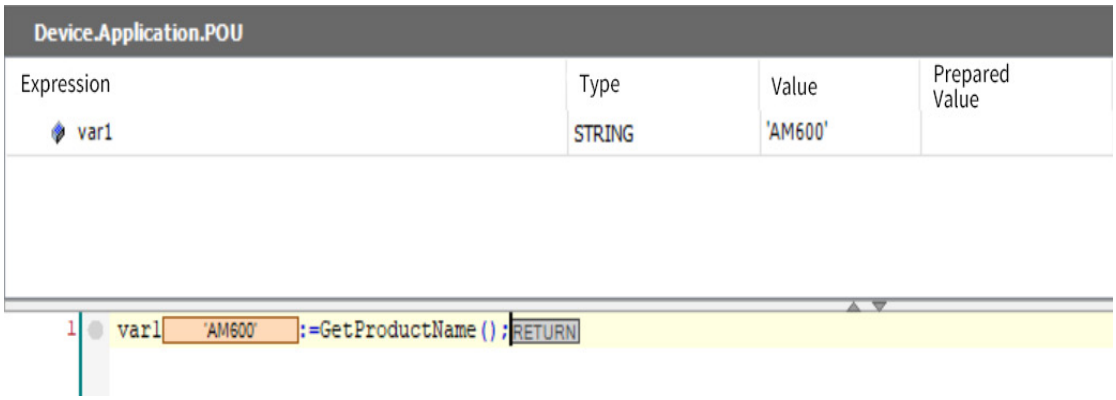
	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
out	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√

■ Function

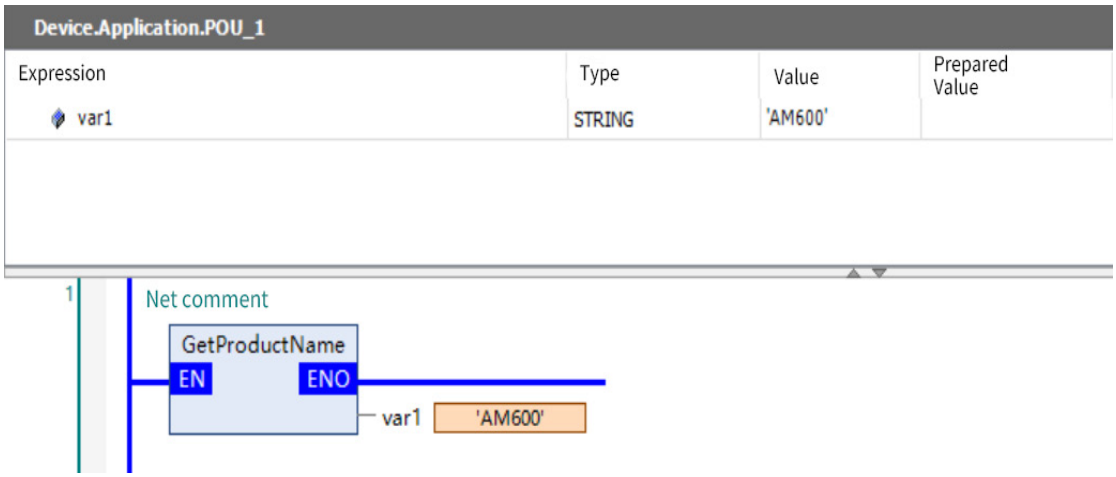
This instruction gets the PLC name.

■ Program example

ST



LD



4.10. 18 GetRuntimeVersion

This instruction gets the runtime version.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
GetRuntimeVersion	Get runtime version	FC	GetRuntimeVersion EN ENO	GetRuntimeVersion ()

■ Variables

Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
Out	Output result	STRING	Depends on data types	-	Output result

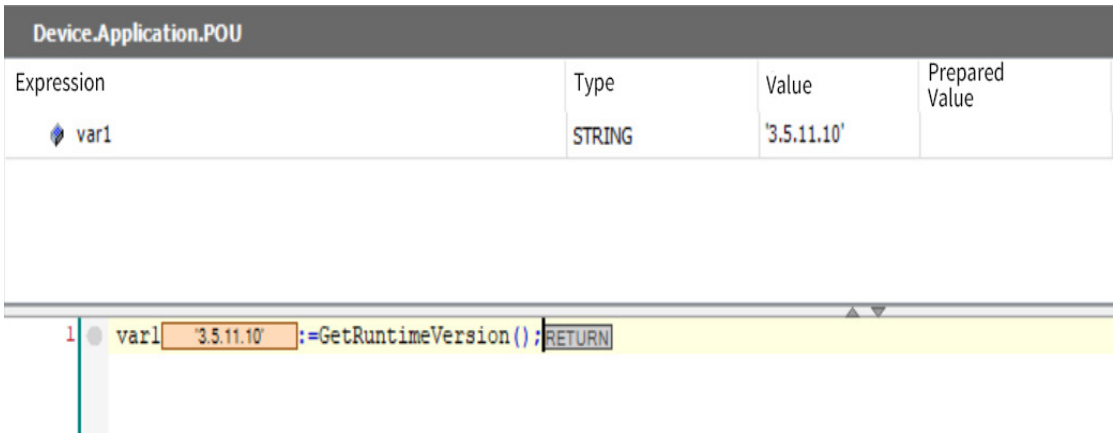
	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
out	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√

■ Function

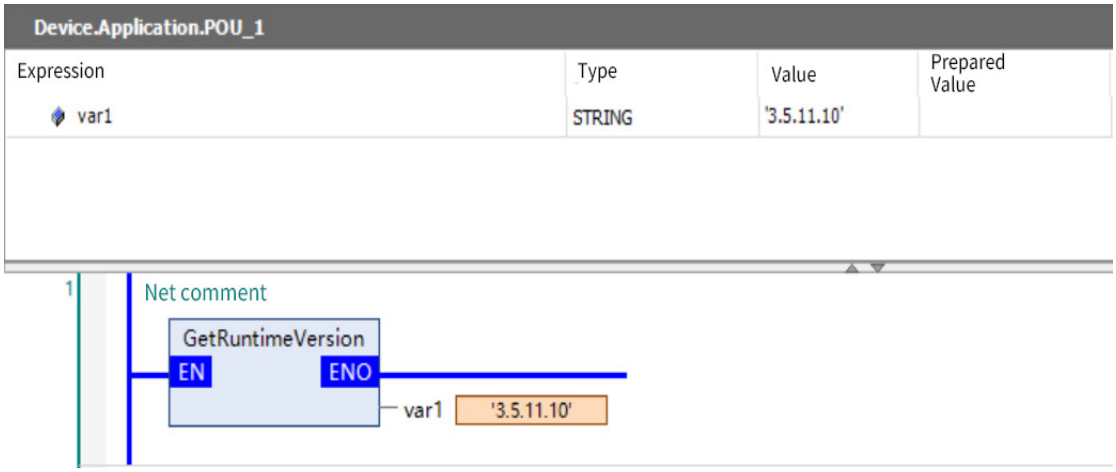
This instruction gets the runtime version.

■ Program example

ST



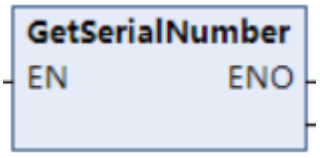
LD



4.10. 19 GetSerialNumber

This instruction gets the PLC serial number.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
GetSerialNumber	Get the PLC serial number (unique)	FC		GetSerialNumber()

■ Variables

Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
Out	Output result	STRING	Depends on data types	-	Output result

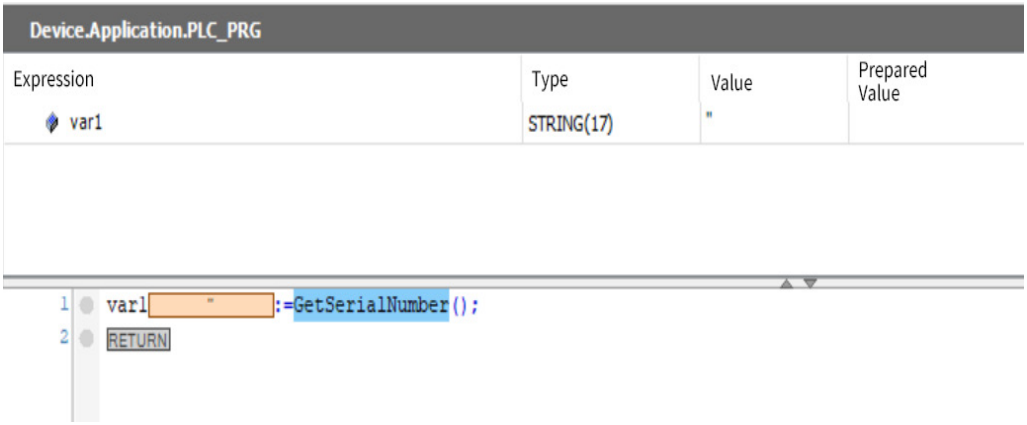
	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
out	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√

■ Function

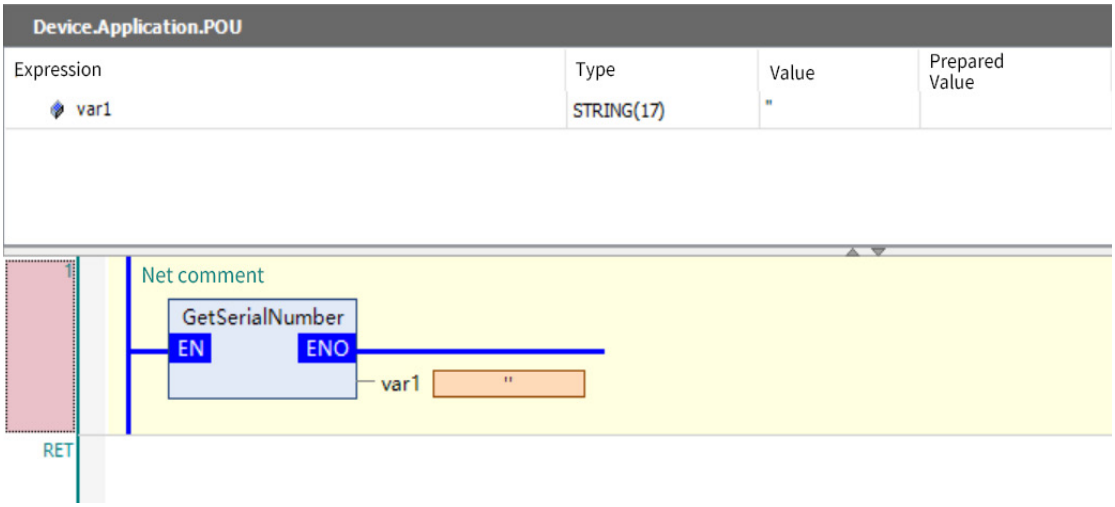
This instruction gets the PLC serial number.

■ Program example

ST



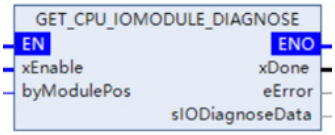
LD



4.10. 20 GET_CPU_IOMODULE_DIAGNOSE

This instruction gets diagnostic information of CPU local modules.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
GET_CPU_IOMODULE_DIAGNOSE	Get diagnostic information of CPU local modules	FB		GET_CPU_IOMODULE_DIAGNOSE(xEnable:= , byModulePos:=, xDone=> , eError=> , sIODiagnoseData=>);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
xEnable	Input Signal	BOOL	[FALSE, TRUE]	FALSE	Input Signal
bymodulepos	Input Signal	BYTE	-	0	Local module to obtain

Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
xDone	Function block completed state	BOOL	Depends on data types	-	Function block completed state
eError	Function block error ID	SYS_HC_ERROR	Depends on data types	-	Function block error ID
sIODignoseDate	Diagnostic data	-	Depends on data types	-	Diagnostic data

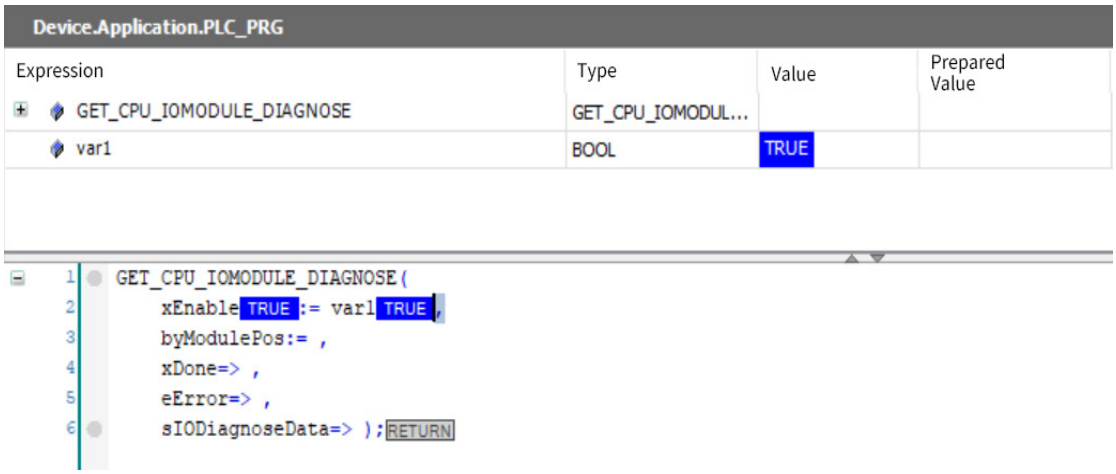
	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
xEnable	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
bymodulepos	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
xDone	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
eError	Enumerator SYS_HC_ERROR																			
sIODignose-Date	GET_CPU_IOMODULE_DIAGNOSE																			

■ Function

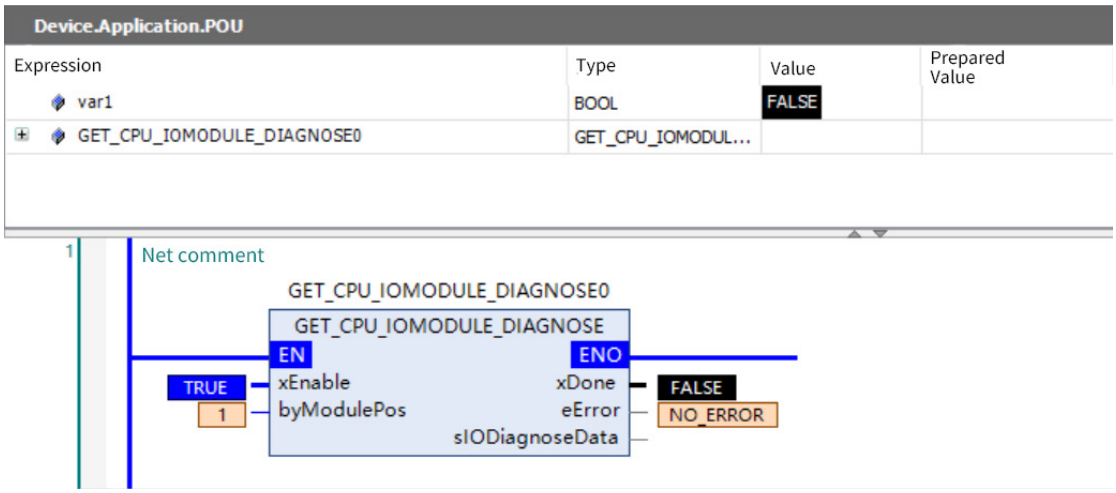
This instruction gets diagnostic information of CPU local modules.

■ Program example

ST



LD



4.10. 21 SysHC_GetAllGL20ModuleInfo

This instruction gets information about all GL20 series modules.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
SysHC_GetAllGL20 ModuleInfo	Get information about all GL20 modules	FB		SysHC_GetAllGL20ModuleInfo(xEnable:= , xValid=> , xBusy=> , xError=> , eErrorID=> , byNumConfDevice=> , stModuleInfo=>);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
xEnable	Function block enable	BOOL	0 to 1	0	Enable instruction running

Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
xValid	Output active	BOOL	0 to 1	0	Output active
xBusy	Executing the instruction	BOOL	0 to 1	0	Executing the instruction
xError	Error	BOOL	0 to 1	0	Error
eErrorID	Error ID	SYS_HC_ERROR	-	0	Error ID
byNumConfDevice	Number of modules configured in the project	BYTE	32	0	Number of modules configured in the project
stModuleInfo	All module information	ARRAY [0..31] OF GL20_MODULE_INFO	31	-	All module information that is received

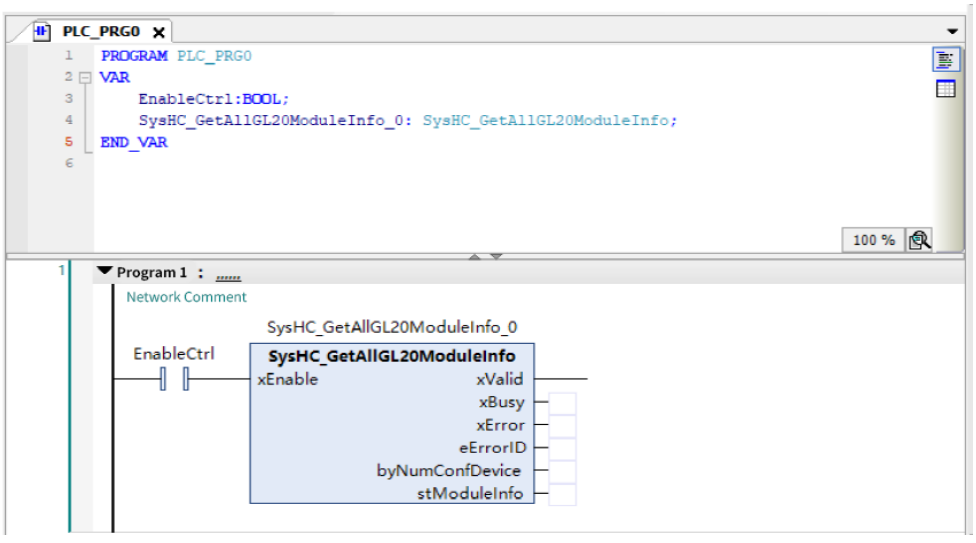
■ Program example

ST

```
PROGRAM PLC_PRG
VAR
    SysHC_GetAllGL20ModuleInfo_0: SysHC_GetAllGL20ModuleInfo;
END_VAR
```

```
SysHC_GetAllGL20ModuleInfo_0(
        xEnable:= ,
        xValid=> ,
        xBusy=> ,
        xError=> ,
        eErrorID=> ,
        byNumConfDevice=> ,
        stModuleInfo=> );
```

LD



■ Precautions

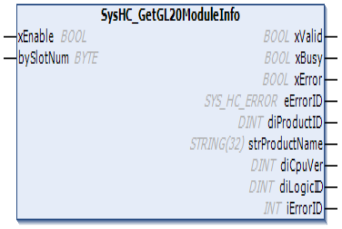
· This instruction applies to AM780-N PLCs.

· This instruction applies to SysHCPlcInfo.library of version 1.0.84.0 and later versions.

4.10. 22 SysHC_GetGL20ModuleInfo

This instruction gets information about a specific GL20 series module located in a specified slot.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
SysHC_GetGL20 ModuleInfo	Get information about a specific GL20 module	FB		SysHC_GetGL20ModuleInfo (xEnable:= , bySlotNum:= , xValid=> , xBusy=> , xError=> , eErrorID=> , diProductID=> , strProductName=> , diCpuVer=> , diLogicID=> , iErrorID=>);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
xEnable	Function block enable	BOOL	0 to 1	0	Enable instruction running
bySlotNum	Slot number	BYTE	31	0	Number of the slot where the local module to obtain is located

Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
xValid	Output active	BOOL	0 to 1	0	Output active
xBusy	Executing the instruction	BOOL	0 to 1	0	Executing the instruction
xError	Error	BOOL	0 to 1	0	Error
eErrorID	Error ID	SYS_HC_ERROR	-	0	Error ID
diProductID	Product ID	DINT	32	0	Product ID
strProductName	Device name	STRING(32)	31	-	Device name
diCpuVer	CPU version	DINT	-	-	Firmware version of the CPU in the device
diLogicID	Logic firmware version	DINT	-	-	Version of FPGA logic firmware in the device
iErrorID	Device error code	DINT	-	-	Device error code

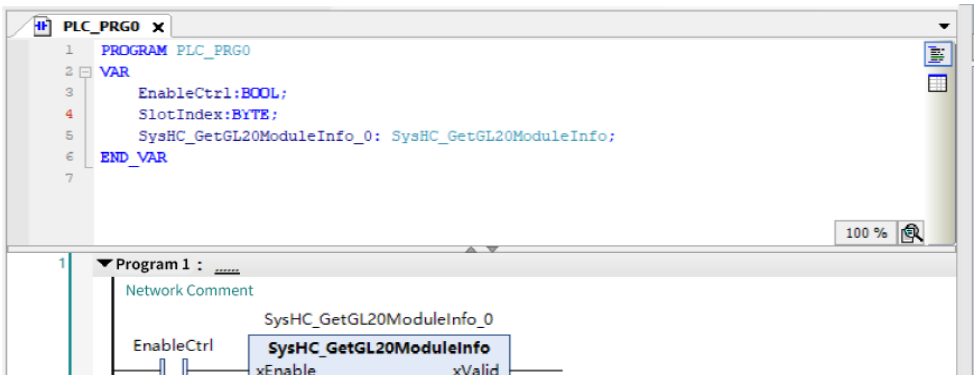
■ Program example

ST

```
PROGRAM PLC_PRG
VAR
    SysHC_GetGL20ModuleInfo_0: SysHC_GetGL20ModuleInfo;
END_VAR
```

```
SysHC_GetGL20ModuleInfo_0(
    xEnable:= ,
    bySlotNum:= ,
    xValid=> ,
    xBusy=> ,
```

LD



■ Precautions

· This instruction applies to AM780-N PLCs.

· This instruction applies to SysHCPlcInfo.library of version 1.0.84.0 and later versions.

4.10. 23 SysHC_NTPClient

This instruction sets the system time through NTP for the PLC.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
SysHC_NTPClient	Set system time through ntp	FB	SysHC_NTPClient_0 SysHC_NTPClient xEnable xDone strIPAddress xBusy uiPort xError uiSyncTime eErrorID	SysHC_NTPClient (xEnable:= , strIPAddress:= , uiPort:= , uiSyncTime:= , xDone=> , xBusy=> , xError=> , eErrorID=>);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
xEnable	Active high	BOOL	0 to 1	0	Active high
strIPAddress	IP address	STRING	-	-	IP address
uiPort	Network port	UINT	-	123	Network port address

Input Variable	Name	Data Type	Value Range	Initial Value	Description
uiSyncTime	PLC synchroniza-tion time	UDINT	10 to 86400	-	PLC synchroniza-tion time

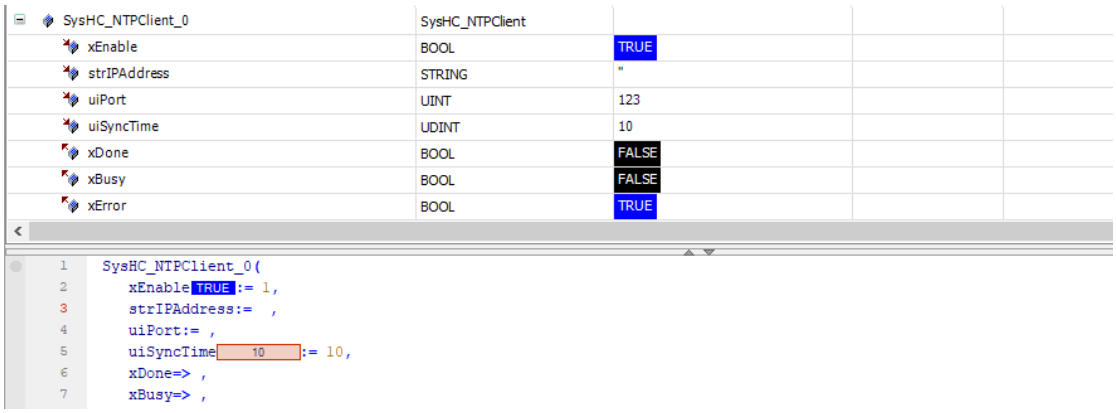
Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
xDone	Output active	BOOL	0 to 1	0	Output active
xBusy	Executing the instruction	BOOL	0 to 1	0	Executing the instruction
xError	Error	BOOL	0 to 1	0	Error
eErrorID	Error ID	SYS_HC_ERROR	-	NO_ERROR	Error ID

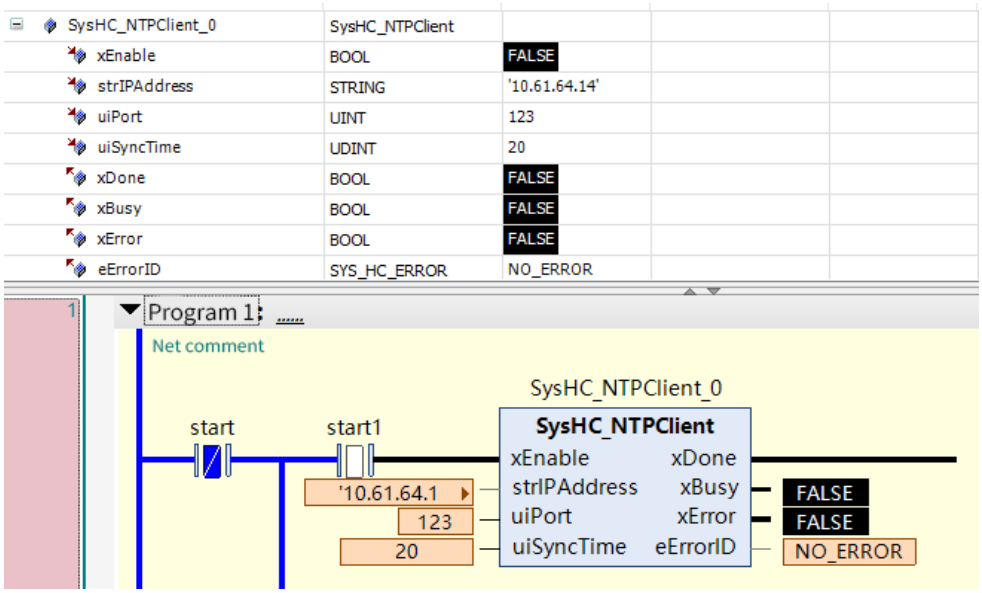
	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Enable	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
strIPAddress	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√
uiPort	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-
uiSyncTime	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-
xEnable	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
xValid	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
xBusy	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
xError	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
eErrorID	SYS_HC_ERROR																			

■ Program example

ST



LD



■ Precautions

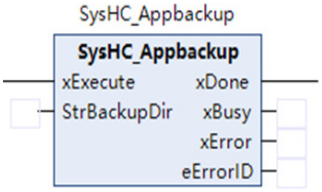
Either the SysHC_NTPClient or SysHC_SetSystemDate function block can be used to set the system time.

Only one SysHC_NTPClient function block can be configured in the project.

4.10. 24 SysHC_Appbackup

This instruction backs up the user program. The function block triggers backup of the user program by using the external library.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
SysHC_Appbackup	Complete user program backup	FB		SysHC_Appbackup(xExecute:=, StrBackupDir:=, xDone=>, xBusy=>, xError=>, eErrorID=>);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
xExecute	Function block enable	BOOL	0 to 1	0	Enable of triggering at the rising edge, to obtain the backup result
StrBackupDir	Name of the directory for storing the backup file, up to 32 bytes	STRING[32]	-	-	Specified directory name to start with the "/", without any illegal characters such as @ # , !

Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description

xDone	Output active	BOOL	0 to 1	0	Output active
xBusy	Executing the instruction	BOOL	0 to 1	0	Executing the instruction
xError	Error	BOOL	0 to 1	0	Error
eErrorID	Error ID	SYS_HC_ERROR	-	0	Error ID

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Enable	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
StrBackupDir	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√
xDone	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
xBusy	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
xError	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
eErrorID	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

■ Function

• This instruction applies to AC700, AC800, AM780-N, AM300, and AM500 medium-sized PLCs.

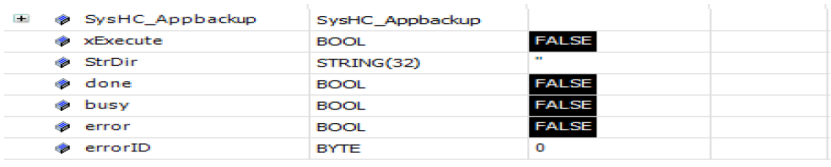
• To write backup data to an external USB storage device (USB flash disk), call the SysHC_UDiskPath instruction to get the file path of the USB storage device connected to the PLC. The StrBackupDir parameter can be set to the path to call the SysHC_AppBackup backup instruction.

• When the input parameter StrBackupDir is empty, the default storage root directory is /App/FB_AppBackup_Data. When the input parameter is not empty, /App/ and the StrBackupDir parameter are concatenated as the complete storage path for backup data.

• Backup data retrieval capabilities are provided to the client by using SFTP. The SFTP account management and root directory control policies for user access are as follows: The SFTP client public account is inovance, and the default password is admin.

■ Program example

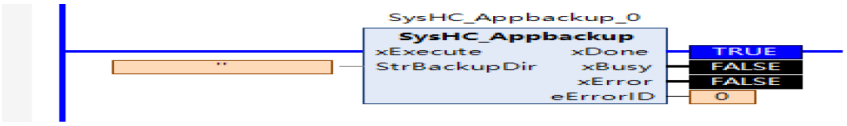
ST




```
SysHC_Appbackup(xExecute FALSE :=xExecute FALSE ,
                StrBackupDir "" :=StrDir "" ,
                xDone FALSE =>done FALSE ,
                xBusy FALSE =>busy FALSE ,
                xError FALSE =>error FALSE ,
                eErrorID 0 =>errorID 0 );
```

LD

SysHC_Appbackup	SysHC_Appbackup	
xExecute	BOOL	FALSE
StrBackupDir	STRING(32)	''
xDone	BOOL	FALSE
xBusy	BOOL	FALSE
xError	BOOL	FALSE
eErrorID	BYTE	0
xLastExecute	BOOL	FALSE



■ Precautions

1) AM300 and AM500 PLCs have limited built-in flash storage space, so an external SD expansion card is required (which needs to be expanded by using the GE20-TF-RTC expansion card) to store the data backed up by the backup instruction. AM300 and AM500 PLCs do not support hot swapping recognition of SD cards, so the SD card must be connected before the device is powered on and then started.

2) When the system memory usage exceeds 80%, backup is not allowed, and calling the backup instruction returns the error code 0x12.

3) When the available storage space in the backup data storage directory is less than the total size of the files to be backed up plus 100 MB, calling the backup instruction returns the error code 0xDD.

4) Only the latest backup file package is retained. Multiple backups will clear other file packages in the backup directory.

5) If an IEC task enables backup instruction pins by event triggering, events need to be continuously triggered to refresh the instruction execution result. This prevents pin occupancy from causing the failure to call instructions in other tasks.

6) The backup instruction function block is not suitable for calling under FUN and METH.

7) When the backup instruction is called for backup, it is necessary to ensure that all the .app and .crc files of the app exist. If any one is missing, the error code 0x11 will be returned.

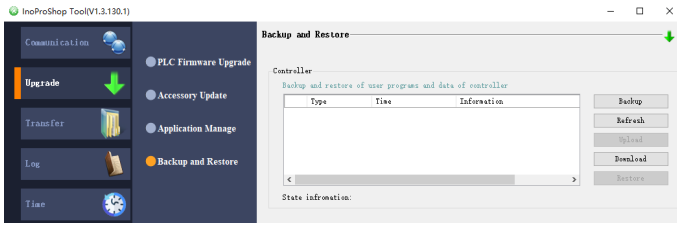
8) The instruction cannot be triggered for backup when there are multiple apps in the PLC.

9) This instruction applies to SysHCPlcInfo.library of version 1.0.83.0 and later versions.

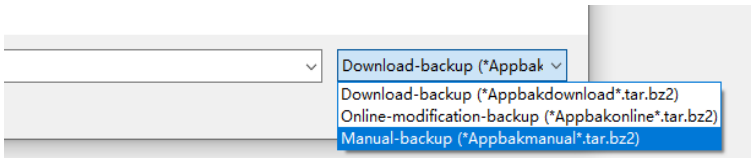
10) The file package Appbakmanual.tar.bz2 backed up by the SysHC_AppBackup instruction allows program restoration by using the Inoproshop Tool. The specific restoration operation is as follows:

a. Select the target device on the Inoproshop Tool.

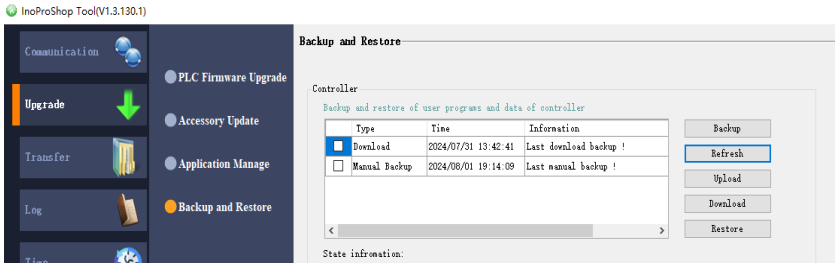
b. Choose **Upgrade** > **Backup and Restore** and click **Download**.



c. To select a file in the pop-up page box, select **Manual-backup (*Appbakmanual*.tar.bz2)** from the drop-down menu in the lower right corner, as shown in the figure below. Then open the file package backed up by the instruction and download it from the PC to the PLC.



d. After downloading the file package to the PLC, click **Refresh**, select **Manual Backup** corresponding to the time in the table, and then click **Restore** to complete the program restoration.



■ Error code

Error codes of the program backup instruction

Name	Diagnosis	Diagnosis Information
SysHC_Appbackup	0xFF	An internal error occurs.
	0xEE	No SD card or USB flash disk is detected.
	0xDD	The storage space is insufficient.
	0xAA	The directory name specified by StrBackupDir is invalid.
	0x11	The source file for backup does not exist.
	0x12	The system memory usage exceeds the limit, causing the failure to start backup.
	0x13	This backup instruction is repeatedly called for different examples.
	0x21	Data storage at power failure triggered before backup times out.
	0x31	The backup directory fails to be created before backup execution.
	0x32	The backup file fails to be packed during the backup process.
	0x33	No corresponding backup file is generated after backup ends.

4. 11 Analog Waveform Instructions

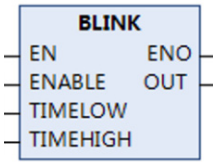
4.11. 1 Instruction List

Instruction Category	Name	FB/FC	Function
Analog waveform instructions	BLINK	FB	Simulate a blinking signal
	GEN	FB	Analog waveform generator
	FREQ_MEASURE	FB	Analog frequency measurement

4.11. 2 BLINK

If ENABLE is set to TRUE, this instruction sets the output to TRUE within the period specified by TIMEHIGH and then sets the output to FALSE in the period specified by TIMELOW.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
BLINK	Simulate a blinking signal	FB		BLINK(ENABLE:=,TIMELOW:=, TIMEHIGH:= , OUT=>);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
ENABLE	Instruction enable	BOOL	[FALSE, TRUE]	FALSE	When the value is TRUE, the instruction starts working.
TIMELOW	Low-level duration	TIME	-	-	Duration for remaining the low level for pulse signals
TIMEHIGH	High-level duration	TIME	-	-	Duration for remaining the high level for pulse signals

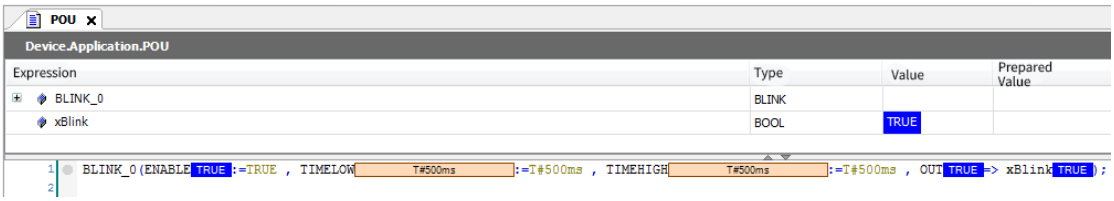
Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
OUT	Pulse signal output	BOOL	[TRUE, FALSE]	FALSE	Output value of a pulse signal

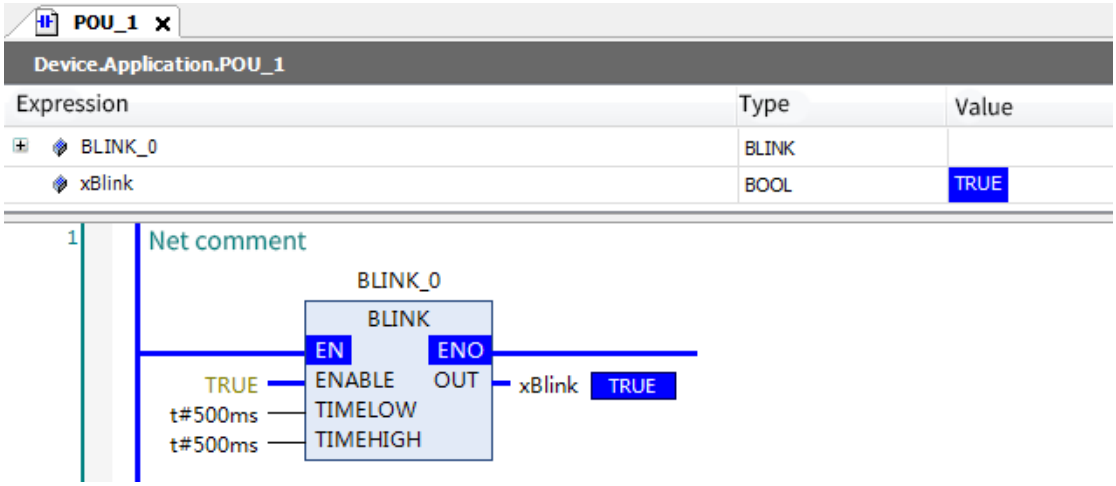
	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
ENABLE	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
TIMELOW	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-
TIMEHIGH	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-
OUT	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

■ Program example

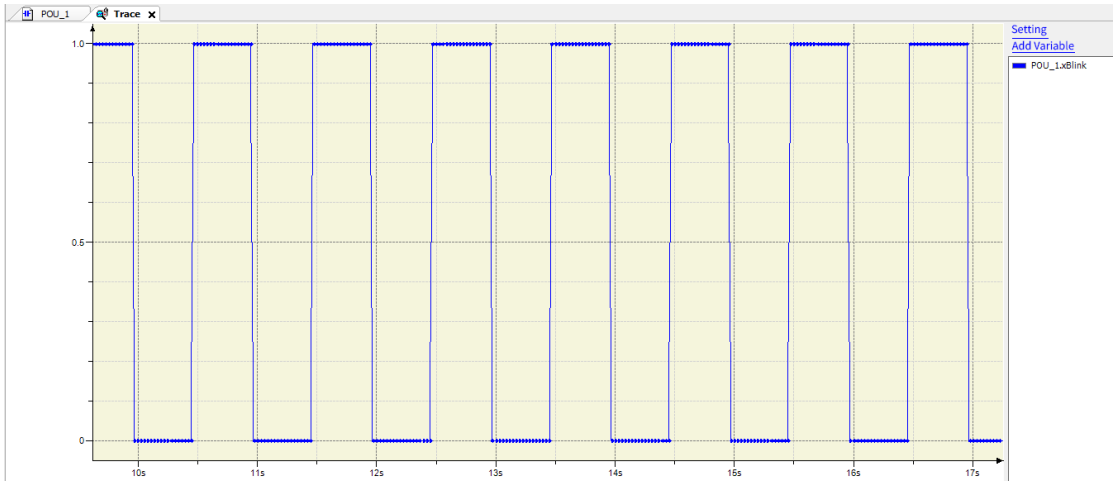
ST



LD



Trace



■ Precautions

When ENABLE is set to FALSE, OUT does not change and no more signal is generated.

4.11. 3 GEN

This instruction generates specified function waveforms based on the input parameters.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
GEN	Analog waveform generator	FB	GEN EN ENO MODE OUT BASE PERIOD CYCLES AMPLITUDE RESET	GEN(MODE:= , BASE:= , PERIOD:= , CYCLES:= , AMPLITUDE:= , RESET:= , OUT=>);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
MODE	Signal type	GEN_MODE	-	TRIANGLE	MODE is an enumerator of the GEN_MODE type, with the following seven enumeration values: 0: TRIANGLE, indicating triangular waves with the output range from –AMPLITUDE to +AMPLITUDE 1: TRIANGLE_POS, indicating triangular waves with the output range from 0 to +AMPLITUDE 2: SAWTOOTH_RISE, indicating rising sawtooth waves with the output range from –AMPLITUDE to +AMPLITUDE 3: SAWTOOTH_FALL, indicating falling sawtooth waves with the output range from +AMPLITUDE to –AMPLITUDE 4: RECTABGLE, indicating square waves with the output range from –AMPLITUDE to +AMPLITUDE 5: SINE, indicating sine waves 6: COSINE, indicating cosine waves
BASE	Circulating mode	BOOL	[FALSE, TRUE]	FALSE	TRUE: Output signals by the cycle period specified by PERIOD FALSE: Output signals by the cycle times specified by CYCLES
PERIOD	Cycle period	TIME	-	TIME#1s0ms	This variable is valid when BASE is TRUE, specifying the cycle period for output signals.
CYCLES	Cycle times	INT	0 to 65535	1000	This variable is valid when BASE is FALSE, specifying the cycle times for output signals.
AMPLITUDE	Signal vibration amplitude	INT	0 to 65535	0	Vibration amplitude for output signals
RESET	Reset	BOOL	[FALSE, TRUE]	FALSE	When the value is TRUE, output signals are reset to 0. When the value is FALSE, signals can be properly output.

Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
OUT	Signal output value	INT	0 to 65535	0	Output value of a periodic signal

Note:

MODE: TRIANGLE, specifying the trigonometric function ranging from –AMPLITUDE to +AMPLITUDE

TRIANGLE_POS, specifying the trigonometric function ranging from 0 to +AMPLITUDE

SAWTOOTH_RISE, specifying the increment of the sawtooth wave function ranging from –AMPLITUDE to +AMPLITUDE

SAWTOOTH_FALL, specifying the decrement of the sawtooth wave function ranging from –AMPLITUDE to +AMPLITUDE

RECTANGLE, specifying the conversion of the sawtooth wave function ranging from –AMPLITUDE to +AMPLITUDE

SINE, specifying the sine function

COSINE: specifying the cosine function

BASE: When the value is FALSE, the function period is the value of CYCLES multiplied by the task period. When the value is TRUE, the function period is set to the duration specified by PERIOD.

PERIOD: period duration, which takes effect when BASE is TRUE

CYCLES: number of periods, which takes effect when BASE is FALSE

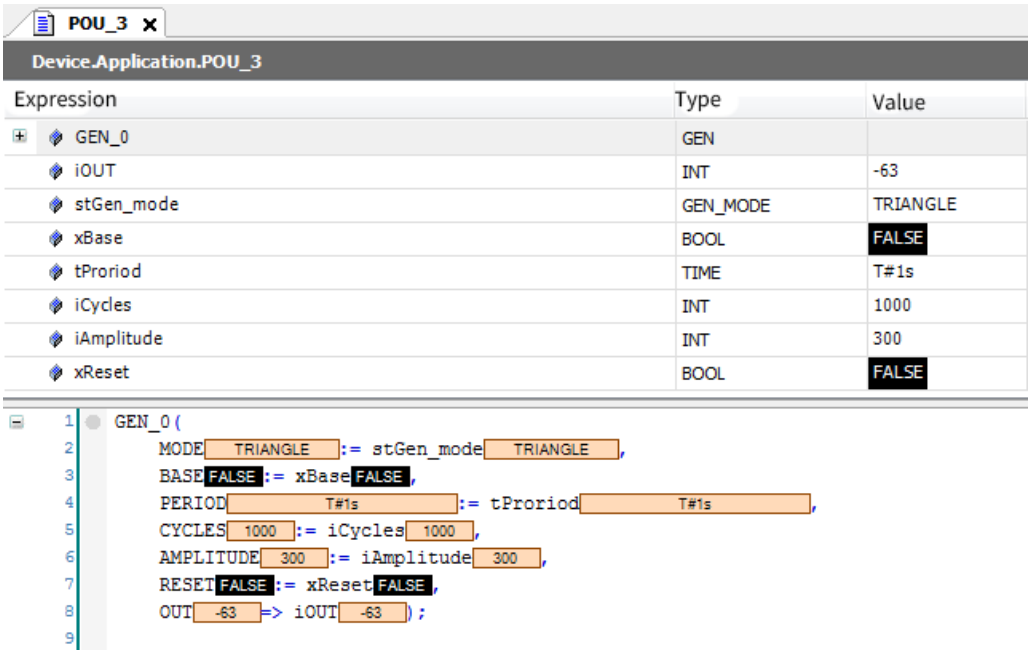
AMPLITUDE: vibration amplitude of a function, which determines the oscillation range of the curve

RESET: When the value is TRUE, the function block output is 0.

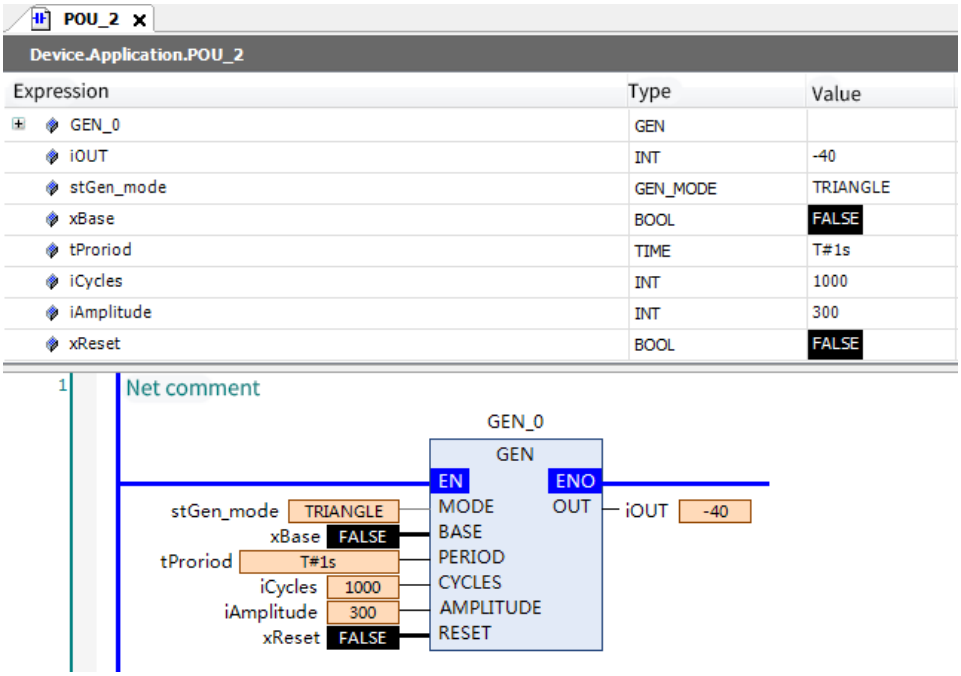
OUT: function block output

■ Program example

ST

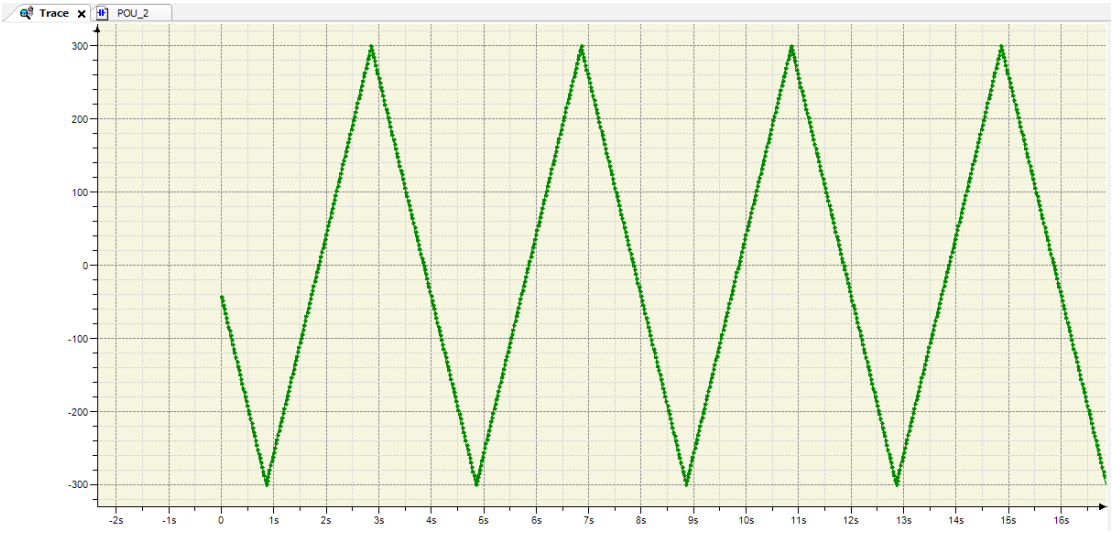


LD



TRACE

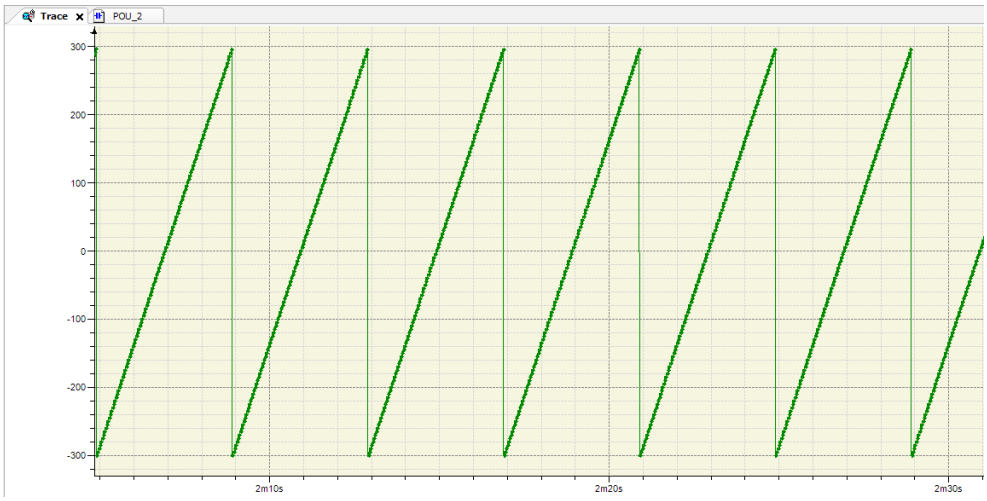
MODE = TRIANGLE



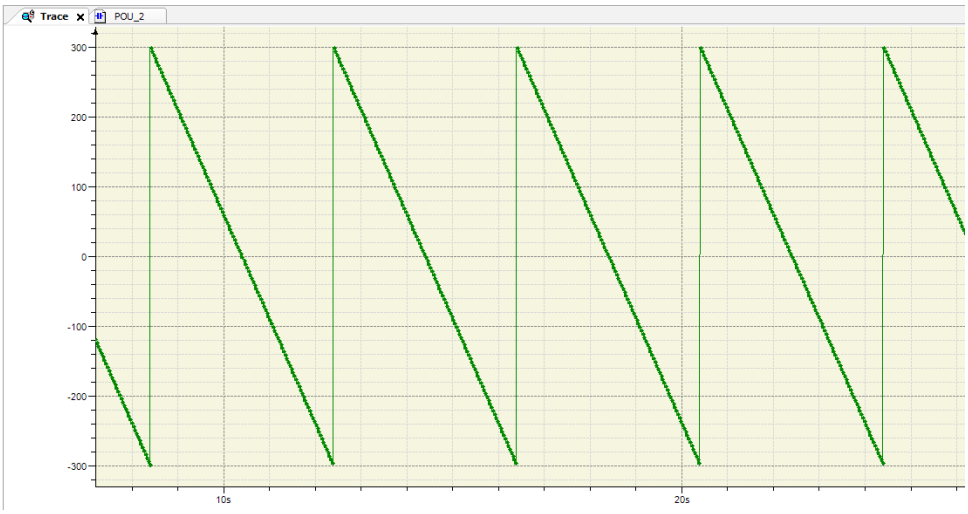
MODE = TRIANGLE_POS



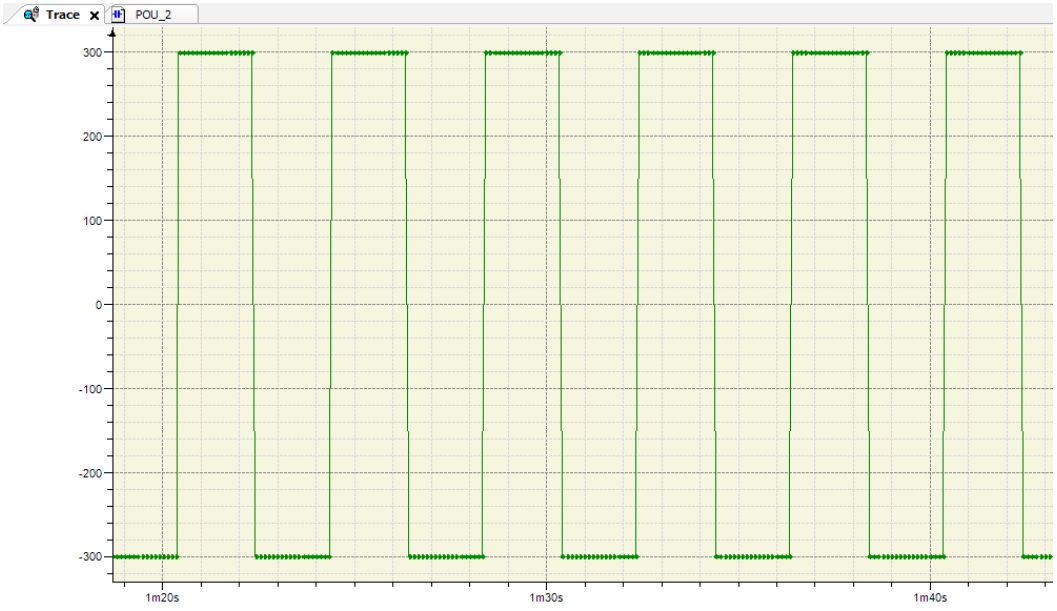
MODE = SAWTOOTH_RISE



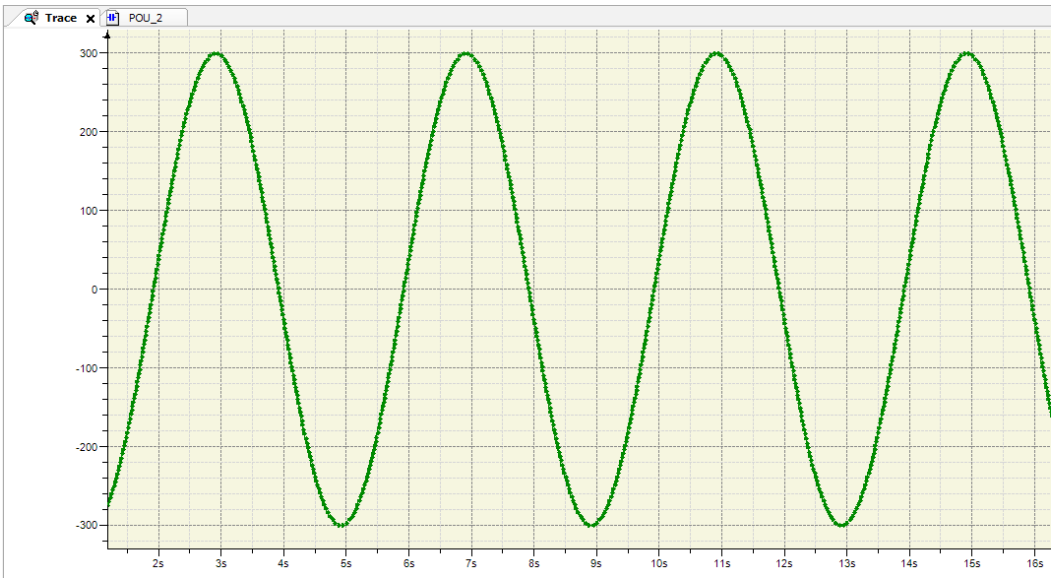
MODE = SAWTOOTH_ FALL



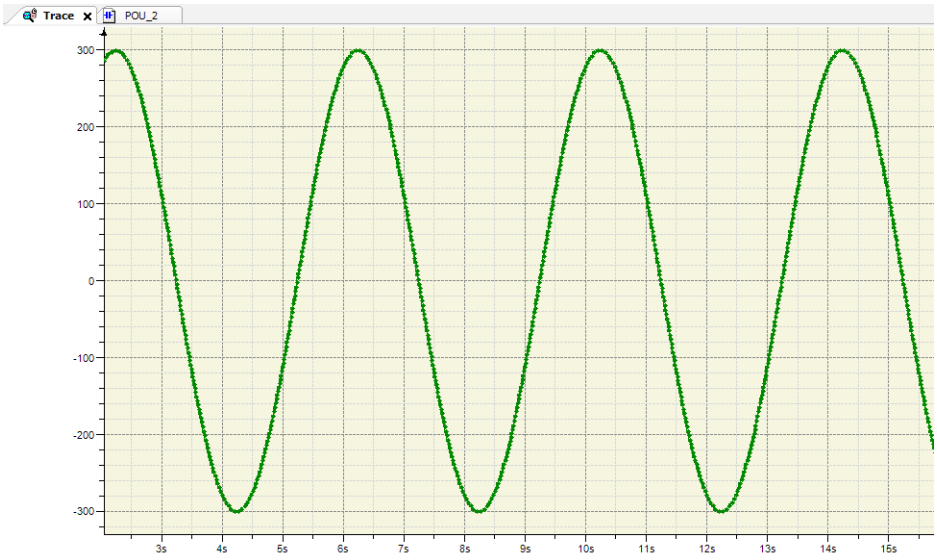
MODE = RECTANGLE



MODE = SINE



MODE = COSINE



4.11. 4 FREQ_MEASURE

This instruction measures the (average) frequency (Hz) of the input Boolean signal.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
FREQ_MEASURE	Analog frequency measurement	FB	FREQ_MEASURE EN ENO IN OUT PERIODS VALID RESET	FREQ_MEASURE(IN:= , PERIODS:= , RESET:= , OUT=> , VALID=>);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
IN	Input Signal	BOOL	[FALSE, TRUE]	FALSE	When the value is TRUE, the instruction starts working.
PERIODS	Measurement period	INT	0 to 65535	1	The value range is from 1 to 10. The default value is 1. The time interval between two rising edges of the input signal is one period. By averaging the frequency values over N periods, we can obtain the frequency of the input signal. This variable specifies the number of periods used for the averaging operation.
RESET	Reset	BOOL	[FALSE, TRUE]	FALSE	When the value is TRUE, the frequency is re-measured. During normal execution, the value should be FALSE.

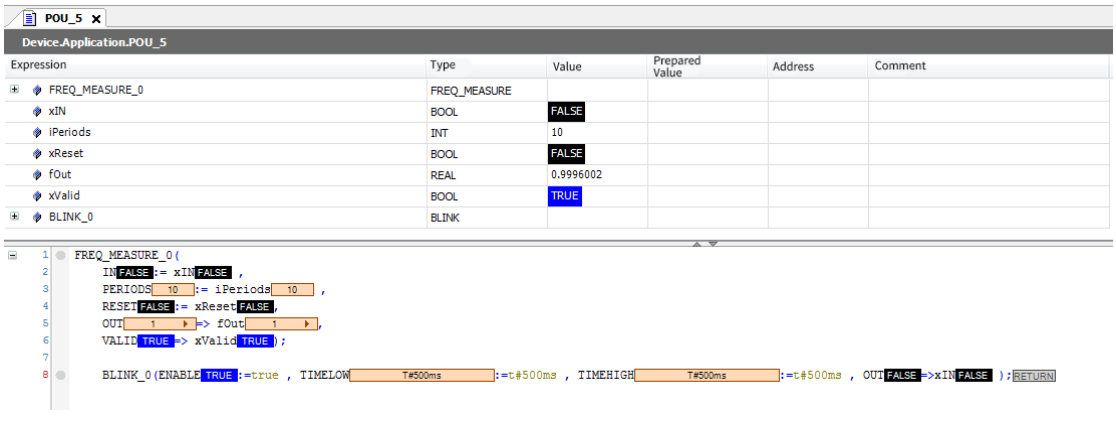
Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
OUT	Frequency output	REAL	–2147483648 to +2147483647	0	Frequency of the input signal, in the unit of Hz
VALID	Operation flag	BOOL	[TRUE, FALSE]	FALSE	The value is TRUE when the first operation is completed or the operation fails. The value is FALSE for other time.

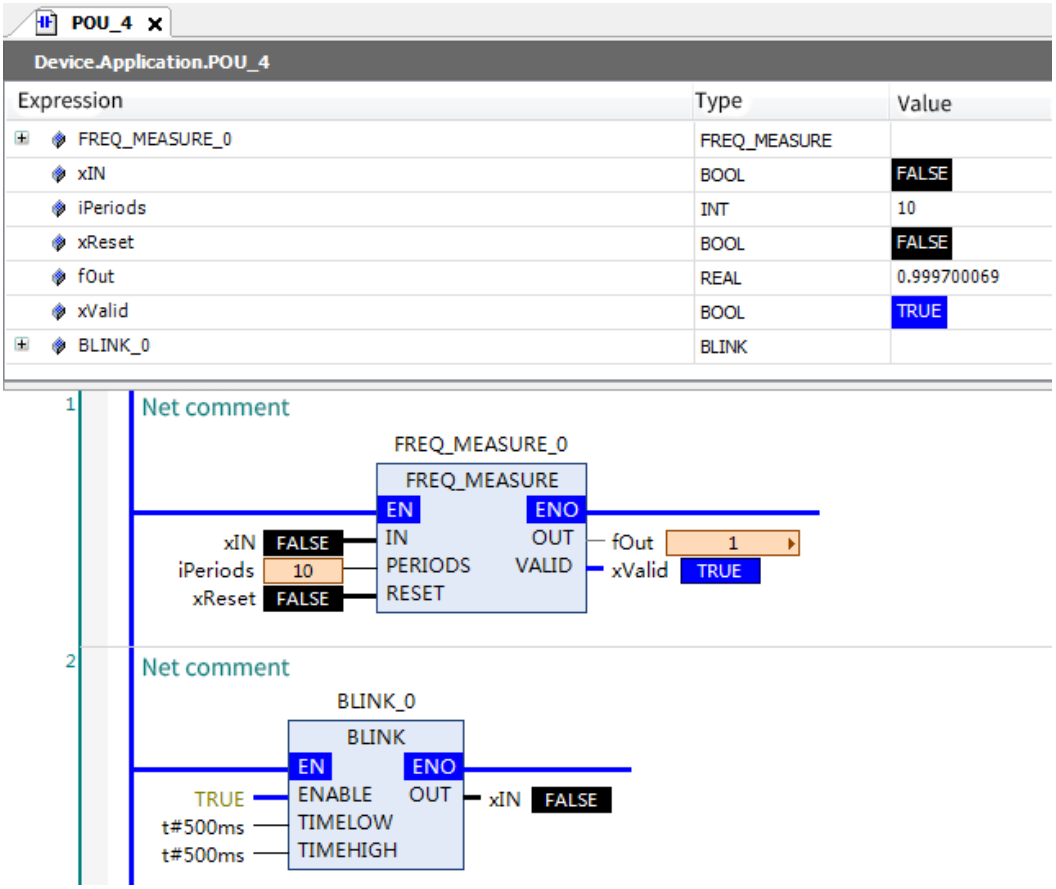
	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
IN	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
PERIODS	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-		-	-	-	-
RESET	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
OUT	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-
VALID	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

■ Program example

ST



LD


POU_4
Device.Application.POU_4
Expression
Type
Value
FREQ_MEASURE_0
FREQ_MEASURE
xIN
BOOL
FALSE
iPeriods
INT
10
xReset
BOOL
FALSE
fOut
REAL
0.999700069
xValid
BOOL
TRUE
BLINK_0
BLINK
1
Net comment
FREQ_MEASURE_0
FREQ_MEASURE
EN
ENO
xIN
FALSE
IN
OUT
fOut
1
iPeriods
10
PERIODS
VALID
xValid
TRUE
xReset
FALSE
RESET
2
Net comment
BLINK_0
BLINK
EN
ENO
TRUE
ENABLE
OUT
xIN
FALSE
t#500ms
TIMELOW
t#500ms
TIMEHIGH

Chapter 5 High-Speed I/O Instructions

5. 1 AM300/AM500/AC700 Local Counter Instructions

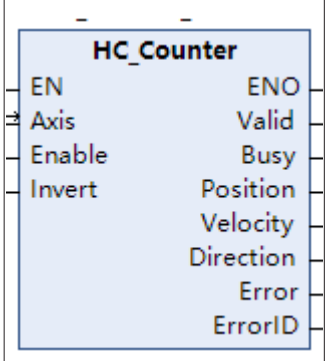
5.1. 1 Instruction List

Instruction Category	Name	FB/FC	Function
AM300/ AM500/AC700 local counter instructions	HC_Counter	FB	High-speed counter enable
	HC_Preset	FB	High-speed counter preset value
	HC_TouchProbe	FB	High-speed counter probe
	HC_Compare	FB	High-speed counter comparison
	HC_ArrayCompare	FB	High-speed counter array comparison
	HC_StepCompare	FB	High-speed counter equal distance comparison
	HC_VirtualTouchProbe	FB	Virtual probe
	HC_ChangeGearingRatio	FB	High-speed counter set gearing ratio
	HC_PWM	FB	High-speed counter PWM
	HC_ReadStatus	FB	High-speed counter read status

5.1. 2 HC_Counter

This instruction enables the high-speed counter.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
HC_Counter	High-speed counter enable	FB		HC_Counter(Axis:= , Enable:= , Invert:= , Valid=> , Busy=> , Position=> , Velocity=> , Direction=> , Error=> , ErrorID=>);

■ Variables

In-out variables

In-Out Variable	Name	Data Type	Value Range	Initial Value	Description
Axis	Encoder axis	ENCODER_REF_INOVANCE	-	-	ENCODER_REF_INOVANCE encoder axis example

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
Enable	Enable	BOOL	[FALSE, TRUE]	FALSE	Triggered by level TRUE: Enable the counter
Invert	Reverse	BOOL	[FALSE, TRUE]	FALSE	Triggered by level TRUE: Change the count direction

Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
Valid	Enabled	BOOL	[FALSE, TRUE]	FALSE	Count enabled state
Busy	Execution flag	BOOL	[FALSE, TRUE]	FALSE	Execution of the count enable instruction
Position	Count value	LREAL	(− 1.7E 308, 1.7E 308)	0.0	Current count value, which is converted from pulse count based on the gearing ratio
Velocity	Velocity/Frequency	LREAL	(− 1.7E 308, 1.7E 308)	0	Pulse frequency
Direction	Direction	HSC_DIR	[− 1, 1]	− 1	Count direction
Error	Error flag	BOOL	[FALSE, TRUE]	FALSE	TRUE: An internal error occurs in the function block
ErrorID	Error ID	HSC_ERROR	-	HSC_NO_ERROR	Error ID For detail, see HSC_ERROR.

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Axis	ENCODER_REF_INOVANCE																			
Enable	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Invert	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Valid	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Busy	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Position	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–
Velocity	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–
Direction	HSC_DIR																			
Error	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
ErrorID	HSC_ERROR																			

■ Function

This function block primarily controls the start/stop of a local counter. When Enable is set to TRUE, the counter responds to external X pulse signals and starts counting. When Enable is set to FALSE, the counting stops.

The counter position changes within the range defined by the counter mode, and the position unit is Unit.

The frequency of the counter is measured in Unit/s. The minimum velocity that can be measured by the counter axis is the velocity corresponding to one pulse of the counter within 1s. If one pulse of the counter

corresponds to 0.01 Unit, the minimum velocity that can be measured is 0.01 Unit/s.

■ Precautions

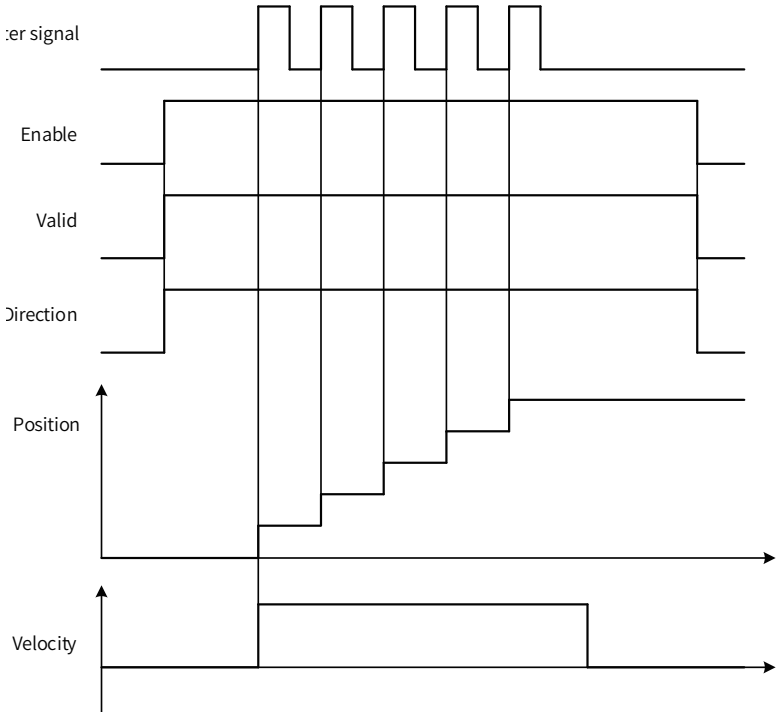
Invert specifies the count direction of the counter. The following table lists the count directions of different count modes. Modification on Invert takes effect only after this function block instruction is enabled again.

Invert	A/B Phase	Pulse+Direction	CW/CCW	Sin-gle-phase Counter
0	Phase A leading phase B, counting up Phase B leading phase A, counting down	Direction signal = OFF, counting down Direction signal = ON, counting up	Phase A, counting up Phase B, counting down	Counting up
1	Phase A leading phase B, counting down Phase B leading phase A, counting up	Direction signal = OFF, counting up Direction signal = ON, counting down	Phase A, counting down Phase B, counting up	Counting down

■ Program Example

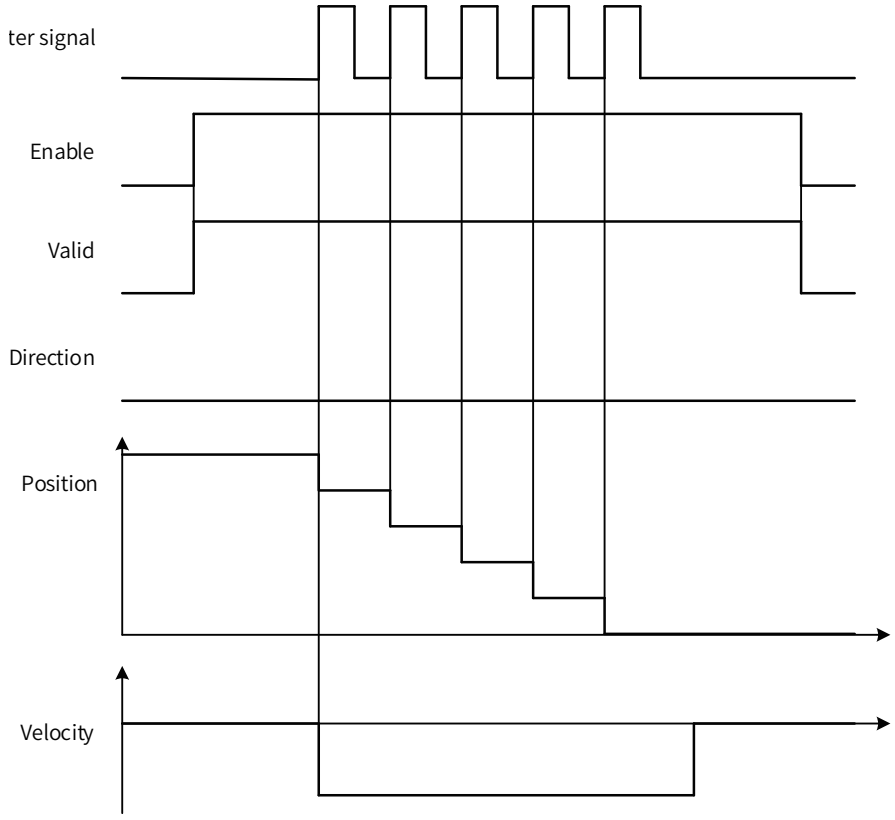
The pulse+direction mode is used as an example in the HC_Counter instruction timing diagram.

If the direction signal is ON and Invert is set to 0, or the direction signal is OFF and Invert is set to 1, the counter counts up, as shown in the following figure.



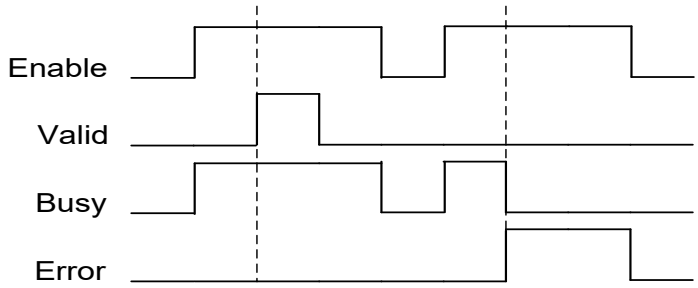
Timing diagram for the HC_Counter incremental count instruction

If the direction signal is ON and Invert is set to 1, or the direction signal is OFF and Invert is set to 0, the counter counts up, as shown in the following figure.



Timing diagram for the HC_Counter decremental count instruction

■ Timing diagram



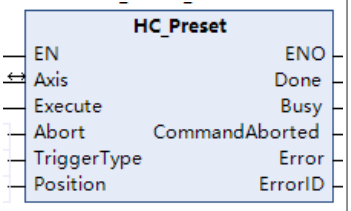
■ Error description

ErrorID in HSC_ERROR specifies the reason for the corresponding error.

5.1.3 HC_Preset

This instruction implements the high-speed counter preset.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
HC_Preset	High-speed counter preset value	FB		HC_Preset(Axis:= , Execute:= , Abort:= , TriggerType:= , Position:= , Done=> , Busy=> , CommandAborted=> , Error=> , ErrorID=>);

■ Variables

In-out variables

In-Out Variable	Name	Data Type	Value Range	Initial Value	Description
Axis	Encoder axis	ENCODER_REF_INOVANCE	-	-	ENCODER_REF_INOVANCE encoder axis example

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
Execute	Enable	BOOL	[FALSE, TRUE]	FALSE	(Triggered by edges) Execution of the preset function switch
Abort	Abort	BOOL	[FALSE, TRUE]	FALSE	Triggered by level TRUE: Function block is aborted
TriggerType	Trigger type	BYTE	[0, 3]	0	0: Triggered on the rising edge of the instruction 1: Triggered on the rising edge of the external input X 2: Triggered on the falling edge of the external input X 3: Triggered on the rising or falling edge of the external input X
Position	Preset value	LREAL	(− 1.7E 308, 1.7E 308)	0.0	Preset value of the counter, in the unit of unit Encoder unit: − 2147483648 to +2147483647

Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
Done	Preset success flag	BOOL	[FALSE, TRUE]	FALSE	Preset success flag
Busy	Execution flag	BOOL	[FALSE, TRUE]	FALSE	Execution of the preset command

Output Variable	Name	Data Type	Value Range	Initial Value	Description
CommandAborted	Abort flag	BOOL	[FALSE, TRUE]	FALSE	Aborted state, which is set when Abort is TRUE
Error	Error flag	BOOL	[FALSE, TRUE]	FALSE	TRUE: An internal error occurs in the function block
ErrorID	Error ID	HSC_ERROR	-	HSC_NO_ERROR	Error ID For detail, see HSC_ERROR.

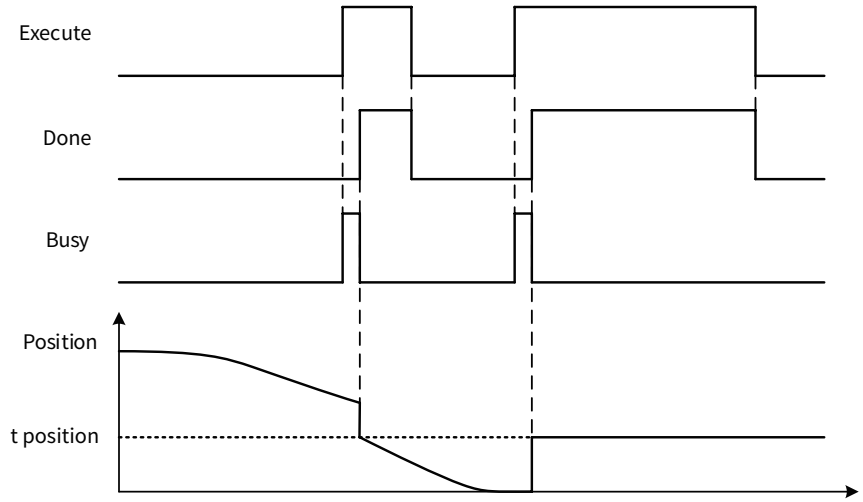
	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Axis	ENCODER_REF_INOVANCE																			
Execute	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Abort	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
TriggerType	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Position	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–
Done	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Busy	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
CommandAborted	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Error	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
ErrorID	HSC_ERROR																			

■ Function

This function block implements the preset function of the counter. Execution of the function block is triggered at the rising edge, and the Done signal is output after the counter is preset.

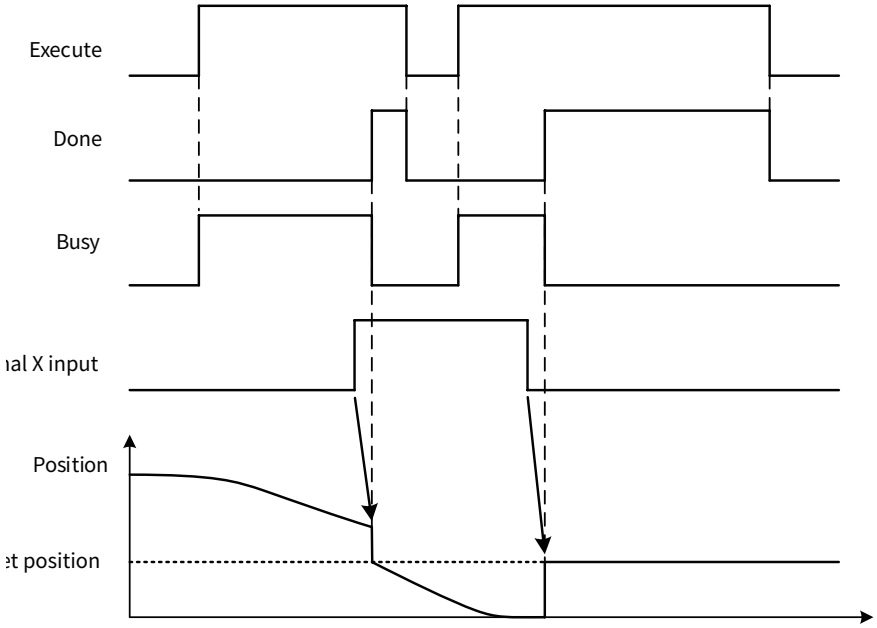
TriggerType can be set to four preset types, including the trigger by the rising edge of the instruction or by external X input.

When the preset condition is set to the trigger by external X input, you need to select the preset function in counter parameter settings and select the input terminal and trigger condition. The input terminal can be any of X0 to X7. The timing diagram of the instruction is as follows when TriggerType is set to the trigger by the rising edge of the instruction (TriggerType = 0).



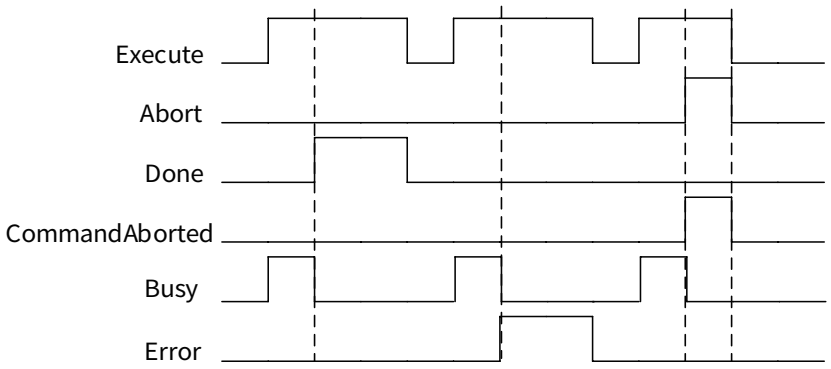
HC_Preset instruction timing diagram when TriggerType is 0

The timing diagram of the instruction is as follows when TriggerType is set to the trigger by the rising or falling edge of the external input X (TriggerType = 3).



HC_Preset instruction timing diagram when TriggerType is 3

■ Timing diagram



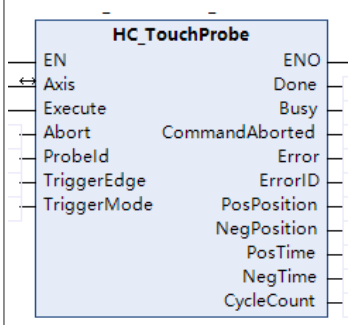
■ Error description

ErrorID in HSC_ERROR specifies the reason for the corresponding error.

5.1.4 HC_TouchProbe

This instruction implements high-speed counter probes.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
HC_TouchProbe	High-speed counter probe	FB		HC_TouchProbe(Axis:= , Execute:= , Abort:= , ProbeId:= , TriggerEdge:= , TriggerMode:= , Done=> , Busy=> , CommandAborted=> , Error=> , ErrorID=> , PosPosition=> , NegPosition=> , PosTime=> , NegTime=> , CycleCount=>);

■ Variables

In-out variables

In-Out Variable	Name	Data Type	Value Range	Initial Value	Description
Axis	Encoder axis	ENCODER_REF_INOVANCE	-	-	ENCODER_REF_INOVANCE encoder axis example

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
Execute	Enable	BOOL	[FALSE, TRUE]	FALSE	(Triggered by edges) Execution of the probe function switch
Abort	Abort	BOOL	[FALSE, TRUE]	FALSE	Triggered by level TRUE: Function block is aborted
ProbeId	Probe ID	WORD	[1, 2]	1	Probe ID

Input Variable	Name	Data Type	Value Range	Initial Value	Description
TriggerEdge	Trigger edge	WORD	[1, 3]	1	Trigger edge 1: Triggered on the rising edge of the external input X 2: Triggered on the falling edge of the external input X 3: Triggered on the rising and falling edges of the external input X
TriggerMode	Trigger mode	BYTE	[0, 1]	0	Trigger type 0: Single 1: Continuous

Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
Done	Probe execution completion flag	BOOL	[FALSE, TRUE]	FALSE	Probe function execution completed
Busy	Execution flag	BOOL	[FALSE, TRUE]	FALSE	Execution of the count probe function
CommandAborted	Aborted state	BOOL	[FALSE, TRUE]	FALSE	Aborted state, which is set when Abort is TRUE
Error	Error flag	BOOL	[FALSE, TRUE]	FALSE	TRUE: An internal error occurs in the function block
ErrorID	Error ID	HSC_ERROR	-	HSC_NO_ERROR	Error ID For detail, see HSC_ERROR.
PosPosition	Position latched on the rising edge	LREAL	(− 1.7E 308, 1.7E 308)	0.0	Position latched on the rising edge, in the unit of unit
NegPosition	Position latched on the falling edge	LREAL	(− 1.7E 308, 1.7E 308)	0.0	Position latched on the falling edge, in the unit of unit
PosTime	Rising edge latch time	LINT	[0, 2^63)	0	Rising edge latch time, in the unit of ns
NegTime	Falling edge latch time	LINT	[0, 2^63)	0	Falling edge latch time, in the unit of ns
CycleCount	Latch times count	WORD	[0, 3]	0	In consecutive latch mode, CycleCount accumulates once upon each successful latching, with the value ranging from 0 to 3 in a cyclic manner.

	Bool-ean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Axis	ENCODER_REF_INOVANCE																			
Execute	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Abort	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
ProbeId	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
TriggerEdge	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
TriggerMode	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Done	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Busy	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
CommandAborted	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Error	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
ErrorID	HSC_ERROR																			
PosPosition	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–
NegPosition	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–
PosTime	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–
NegTime	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–
CycleCount	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

■ Function

This function block latches the count value upon the external X trigger signal. It also supports latching the internal timestamp of the high-speed I/O module. Execution of the function block is triggered at the rising edge, and the Done signal is output after the value is latched.

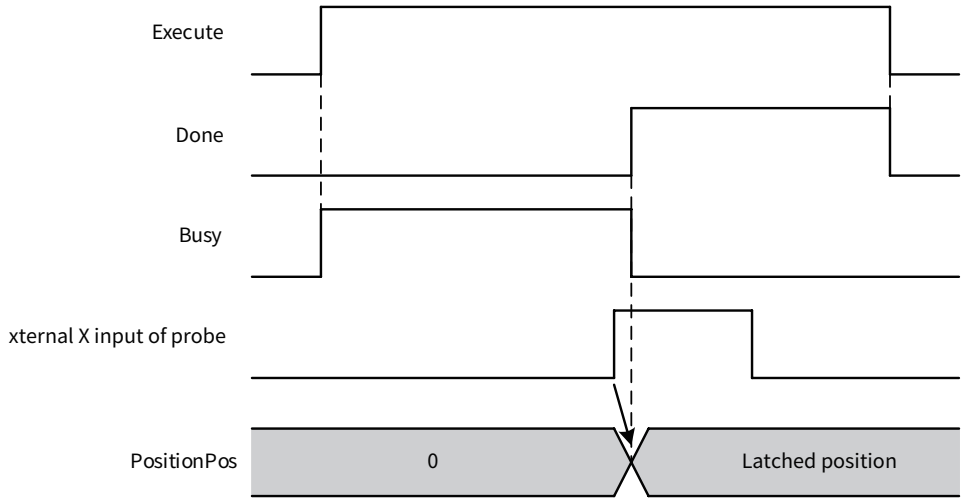
Each counter supports two probes, and each probe supports three modes of trigger at edges: latch at the rising edge, latch at the falling edge, and latch at both rising and falling edges.

Each probe supports two latch modes, including single latch and continuous latch.

Before using the probe function block, the probe signal input terminals must be configured on the corresponding counter interface, and the input terminals can be configured to any from X0 to X7.

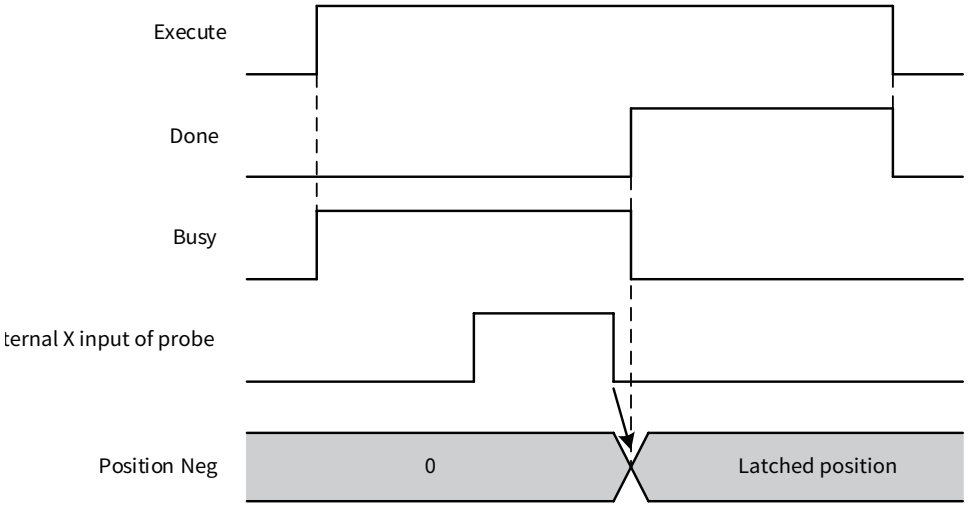
Only counter 0 supports both time and position latching, while other counters only support position latching.

The timing diagram of the instruction is as follows when TriggerEdge is set to the rising edge of the external input X (TriggerEdge = 1) and TriggerMode is set to the single trigger mode (TriggerMode = 0).



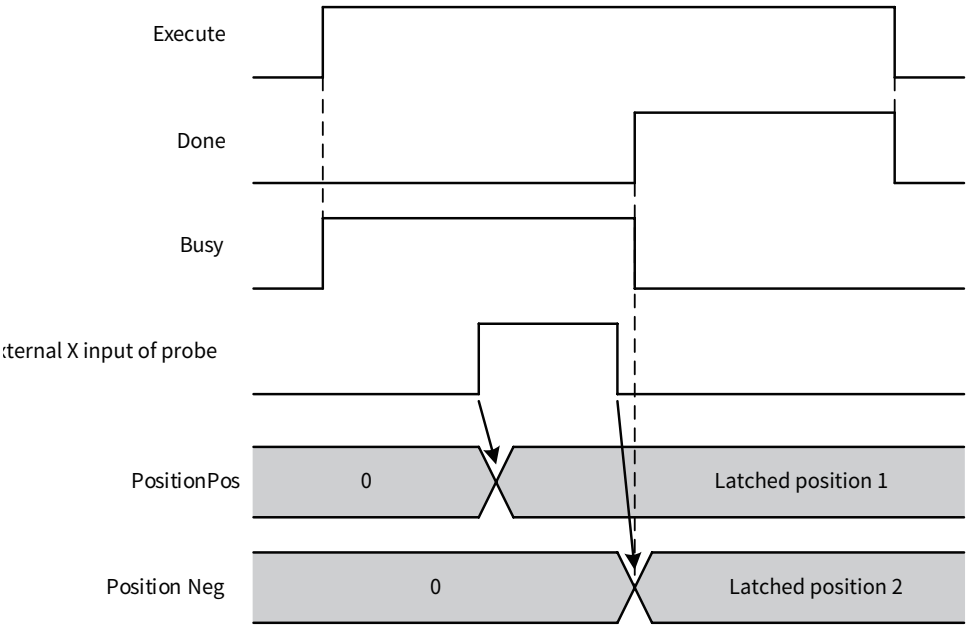
Timing diagram for the HC_TouchProbe instruction

The timing diagram of the instruction is as follows when TriggerEdge is set to the falling edge of the external input X (TriggerEdge = 2) and TriggerMode is set to the single trigger mode (TriggerMode = 0).



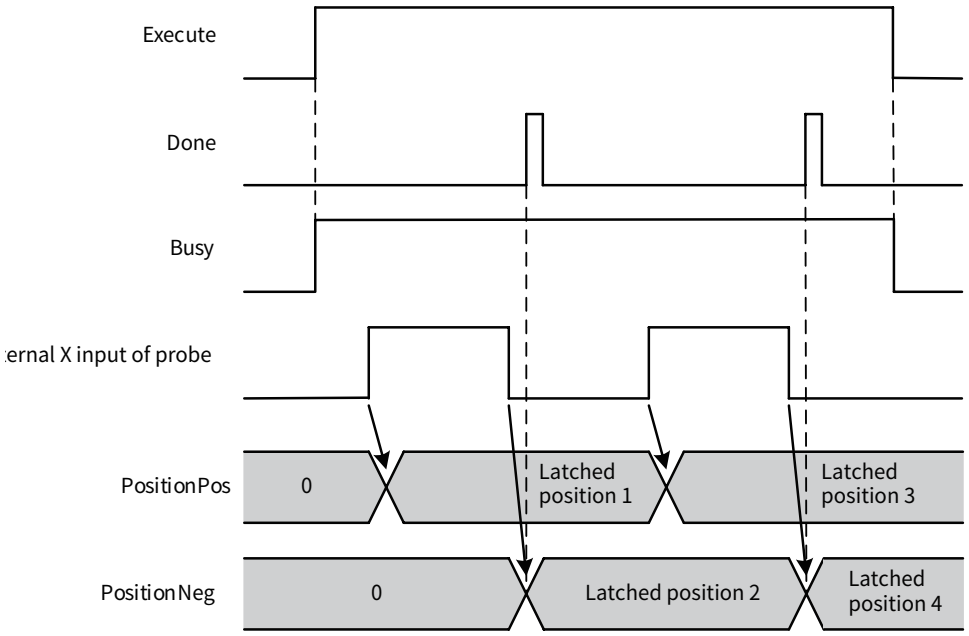
Timing diagram for the HC_TouchProbe instruction

The timing diagram of the instruction is as follows when TriggerEdge is set to the rising and falling edges (rising before falling) of the external input X (TriggerEdge = 3) and TriggerMode is set to the single trigger mode (TriggerMode = 0).



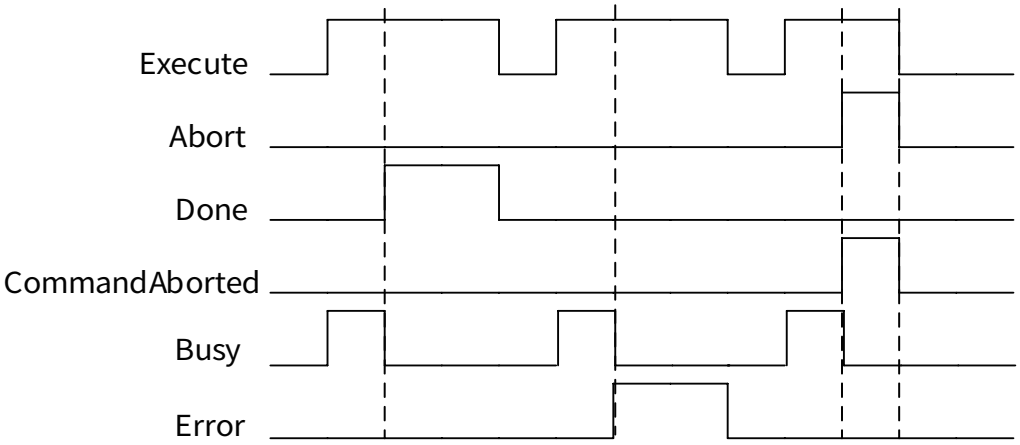
Timing diagram for the HC_TouchProbe instruction

The timing diagram of the instruction is as follows when TriggerEdge is set to the rising and falling edges (falling before rising) of the external input X (TriggerEdge = 3) and TriggerMode is set to the continuous trigger mode (TriggerMode = 1).



Timing diagram for the HC_TouchProbe instruction

■ Timing diagram



■ Error description

ErrorID in HSC_ERROR specifies the reason for the corresponding error.

5.1.5 HC_Compare

This instruction outputs the single-point comparison result of the high-speed counter.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
HC_Compare	High-speed counter comparison	FB	HC_Compare EN ENO Axis Done Execute Busy Abort CommandAborted Position Error OutputEnable ErrorID OutputType OutputValue	HC_Compare(Axis:= , Execute:= , Abort:= , Position:= , OutputType:= , OutputType:= , OutputValue:= , Done=> , Busy=> , CommandAborted=> , Error=> , ErrorID=>);

■ Variables

In-out variables

In-Out Variable	Name	Data Type	Value Range	Initial Value	Description
Axis	Encoder axis	ENCODER_REF_INOVANCE	-	-	ENCODER_REF_INOVANCE encoder axis example

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
Execute	Enable	BOOL	[FALSE, TRUE]	FALSE	(Triggered by edges) Execution of the function block switch

Input Variable	Name	Data Type	Value Range	Initial Value	Description
Abort	Abort	BOOL	[FALSE, TRUE]	FALSE	Triggered by level TRUE: Function block is aborted
Position	Comparison value	LREAL	(− 1.7E 308, 1.7E 308)	0.0	Comparison value of the float type, in the unit of unit
OutputEnable	Output enable	BOOL	[FALSE, TRUE]	FALSE	Enable of the hardware port output function. To use this function, configure the corresponding output port on the encoder interface.
OutputType	Output type	UINT	[0, 1]	0	Output control type 0: Time mode 1: Pulse mode
OutputValue	Output holding parameter	UDINT	[1, 65535]	1000	The holding time and pulse of the output port are enabled. The minimum unit of the time is 100 us. Default value: 1000 (1000 x 100 us = 100 ms)

Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
Done	Comparison consistence completion flag	BOOL	[FALSE, TRUE]	FALSE	The holding time and pulse count reach the set comparison value, and the DO output retains for a period before ending.
CommandAborted	Aborted state	BOOL	[FALSE, TRUE]	FALSE	Aborted state, which is set when Abort is TRUE
Busy	Execution flag	BOOL	[FALSE, TRUE]	FALSE	Execution of the comparison consistence function
Error	Error flag	BOOL	[FALSE, TRUE]	FALSE	TRUE: An internal error occurs in the function block
ErrorID	Error ID	HSC_ERROR	-	HSC_NO_ERROR	Error ID For detail, see HSC_ERROR.

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Axis	ENCODER_REF_INOVANCE																			
Execute	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Abort	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Position	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–
OutputEnable	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

	Bool-ean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
OutputType	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–
OutputValue	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–
Done	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
CommandAborted	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Busy	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Error	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
ErrorID	HSC_ERROR																			

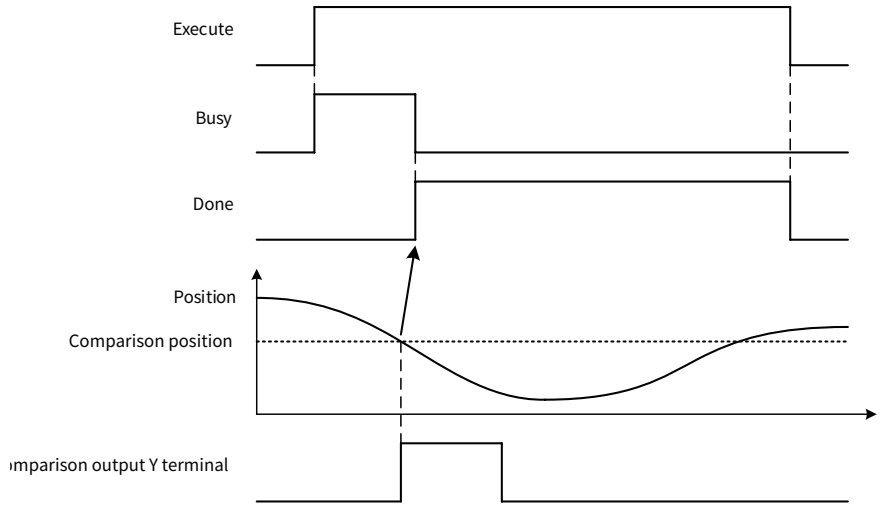
■ Function

This function block triggers a high-speed output Y point and retains it for a certain period when the current count value of the counter is equal to the set comparison value.

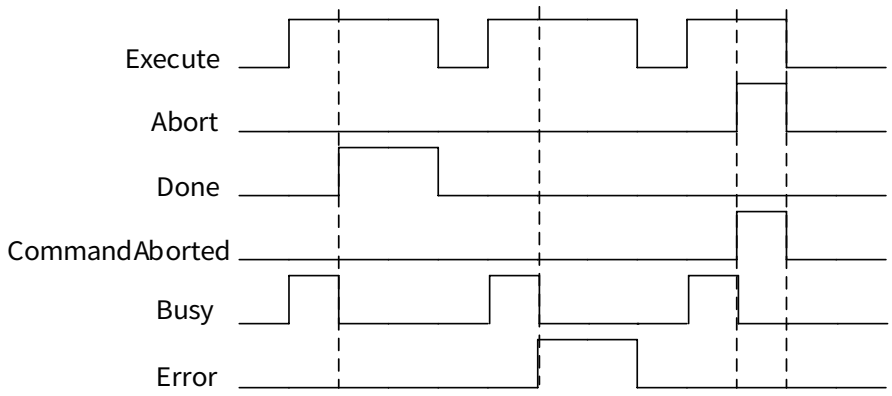
The comparison consistence output function is a single trigger. When Done is TRUE, the comparison consistency function is completed. To continue using the comparison function, trigger Execute again.

Comparison output width can be calculated based on time and pulse. When the count value is equal to the set comparison value, Done is set to TRUE, and the comparison output Y point is opened. It retains open for a period of OutputValue x 100 us (time-based) or OutputValue (pulse-based), and then the Y point is closed.

Before using the comparison function block, the output terminal Y must be configured on the corresponding counter interface, and the input terminals can be configured to any from Y0 to Y3.



■ Timing diagram



■ Error description

ErrorID in HSC_ERROR specifies the reason for the corresponding error.

5.1.6 HC_ArrayCompare

This instruction outputs the high-speed counter array comparison result.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
HC_ArrayCompare	High-speed counter array comparison	FB	HC_ArrayCompare EN ENO Axis Done Execute Busy Abort CommandAborted ArrayAddr Error ArrayLength ErrorID OutputEnable Index OutputType Position OutputValue	HC_ArrayCompare(Axis:= , Execute:= , Abort:= , ArrayAddr:= , ArrayLength:= , OutputEnable:=, OutputType:= , OutputValue:= , Done=> , Busy=> , CommandAborted=> , Error=> , ErrorID=> , Index=> , Position=>);

■ Variables

In-out variables

In-Out Variable	Name	Data Type	Value Range	Initial Value	Description
Axis	Encoder axis	ENCODER_REF_INOVANCE	-	-	ENCODER_REF_INOVANCE encoder axis example

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
Execute	Enable	BOOL	[FALSE, TRUE]	FALSE	(Triggered by edges) Execution of the function block switch
Abort	Abort	BOOL	[FALSE, TRUE]	FALSE	Triggered by level TRUE: Function block is aborted
ArrayAddr	Pointer address	POINTER TO LREAL	-	0	Head address of the comparison point array
Arraylength	Qty	UINT	1 to 1000	1	Array length
OutputEnable	Output enable	BOOL	[FALSE, TRUE]	FALSE	Enable of the hardware port output function. To use this function, configure the corresponding output port on the encoder interface.
OutputType	Output type	UINT	[0, 2]	0	Output control type 0: Time mode 1: Pulse mode
OutputValue	Output holding parameter	UDINT	[1, 65535]	1000	The holding time and pulse of the output port are enabled. The minimum unit of the time is 100 us. Default value: 1000 (1000 x 100 us = 100 ms)

Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
Done	Comparison consistence completion flag	BOOL	[FALSE, TRUE]	FALSE	The holding time and pulse count reach the set comparison value, and the DO output retains for a period before ending.
CommandAborted	Aborted state	BOOL	[FALSE, TRUE]	FALSE	Aborted state, which is set when Abort is TRUE
Busy	Execution flag	BOOL	[FALSE, TRUE]	FALSE	Execution of the comparison consistence function
Error	Error flag	BOOL	[FALSE, TRUE]	FALSE	TRUE: An internal error occurs in the function block
ErrorID	Error ID	HSC_ERROR	-	HSC_NO_ERROR	Error ID For detail, see HSC_ERROR.
Index	Current comparison point No.	UINT	[0, 65535]	0	Number of the current comparison point
Position	Current comparison point position	LREAL	(− 1.7E 308, 1.7E 308)	0	Current comparison point position of the float type, in the unit of unit

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Axis	ENCODER_REF_INOVANCE																			
Execute	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Abort	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Array	POINTER TO LREAL																			
Arraylength	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–
OutputEnable	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
OutputType	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–
OutputValue	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–
Done	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
CommandAborted	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Busy	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Error	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
ErrorID	HSC_ERROR																			
Index	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–
Position	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–

■ Function

This function block triggers a hardware high-speed output port and retains it for a certain period when the current count value of the counter is equal to the comparison value set in the array.

The array comparison consistence output function is a single trigger. When Done is TRUE, the comparison consistency function is completed. To continue using the comparison function, trigger Execute again.

Comparison output width can be calculated based on time and pulse. When the count value is equal to the set comparison value, Done is set to TRUE, and the comparison output Y point is opened. It retains open for a period of OutputValue x 100 us (time-based) or OutputValue (pulse-based), and then the Y point is closed.

Before using the comparison function block, the output terminal Y must be configured on the corresponding counter interface, and the output terminals can be configured to any from Y0 to Y3.

1) In the user program, the length of the comparison point array must be greater than the value of ArrayLength. Otherwise, array out-of-bounds may cause program execution faults and even lead to PLC breakdown.

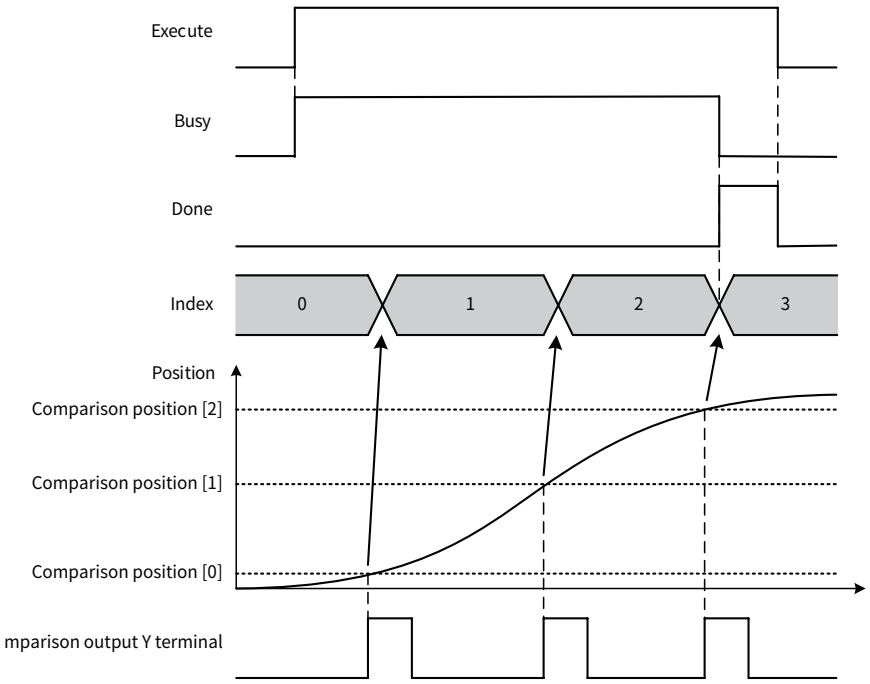
2) For array comparison, the distance between any two adjacent points must be greater than the minimum distance (minimum distance = count frequency x scan period). Otherwise, the function block cannot execute the remaining comparison points in linear mode.

3) During the array comparison, when Execute is triggered at a rising edge again, the function block will restart the comparison output function from the first point.

If the interval between two adjacent trigger points is less than the comparison output holding time, the output port will continuously retain the output state. The total output holding time is equal to the current output holding time plus the next output holding time. For the total output holding time of multiple

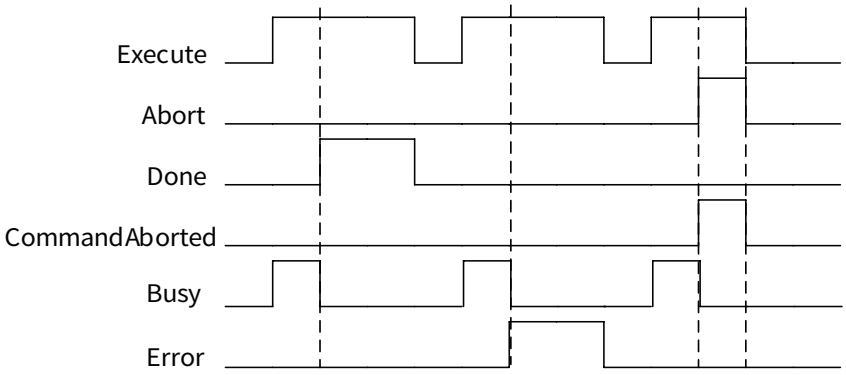
comparison points, use the calculation formula for two adjacent points and then accumulate the output holding time.

The timing diagram of the instruction is as follows for the comparison of three positions (ArrayLength = 3).



Timing diagram for the HC_ArrayCompare instruction

■ Timing diagram



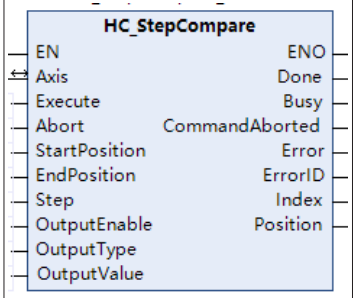
■ Error description

ErrorID in HSC_ERROR specifies the reason for the corresponding error.

5.1.7 HC_StepCompare

This instruction implements the high-speed counter step comparison.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
HC_StepCompare	High-speed counter equal distance comparison	FB		HC_StepCompare(Axis:= , Execute:= , Abort:= , StartPosition:= , EndPosition:= , Step:= , OutputEnable:=, OutputType:= , OutputValue:= , Done=> , Busy=> , CommandAborted=> , Error=> , ErrorID=> , Index=> , Position=>);

■ Variables

In-out variables

In-Out Variable	Name	Data Type	Value Range	Initial Value	Description
Axis	Encoder axis	ENCODER_REF_INOVANCE	-	-	ENCODER_REF_INOVANCE encoder axis example

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
Execute	Enable	BOOL	[FALSE, TRUE]	FALSE	(Triggered by edges) Execution of the function block switch
Abort	Abort	BOOL	[FALSE, TRUE]	FALSE	Triggered by level TRUE: Function block is aborted
StartPosition	Start position	LREAL	(− 1.7E 308, 1.7E 308)	0.0	Position of the first comparison point
EndPosition	End position	LREAL	(− 1.7E 308, 1.7E 308)	0.0	Position of the last comparison point
Step	Step	LREAL	(− 1.7E 308, 1.7E 308)	0.0	Equal distance step
OutputEnable	Output enable	BOOL	[FALSE, TRUE]	FALSE	Enable of the hardware port output function. To use this function, configure the corresponding output port on the encoder interface.
OutputType	Output type	UINT	[0, 1]	1000	Output control type 0: Time mode 1: Pulse mode

Input Variable	Name	Data Type	Value Range	Initial Value	Description
OutputValue	Output holding parameter	UDINT	[1, 65535]	1000	The holding time and pulse of the output port are enabled. The minimum unit of the time is 100 us. Default value: 1000 (1000 x 100 us = 100 ms)

Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
Done	Comparison consistence completion flag	BOOL	[FALSE, TRUE]	FALSE	The holding time and pulse count reach the set comparison value, and the DO output retains for a period before ending.
CommandAborted	Aborted state	BOOL	[FALSE, TRUE]	FALSE	Aborted state, which is set when Abort is TRUE
Busy	Execution flag	BOOL	[FALSE, TRUE]	FALSE	Execution of the comparison consistence function
Error	Error flag	BOOL	[FALSE, TRUE]	FALSE	TRUE: An internal error occurs in the function block
ErrorID	Error ID	HSC_ERROR	-	HSC_NO_ERROR	Error ID For detail, see HSC_ERROR.
Index	Current comparison point No.	UINT	[0, 65535]	0	Number of the current comparison point
Position	Current comparison point position	LREAL	(− 1.7E 308, 1.7E 308)	0	Current comparison point position of the float type, in the unit of unit

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Axis	ENCODER_REF_INOVANCE																			
Execute	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Abort	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
StartPosition	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–
EndPosition	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–
OutputEnable	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Step	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–
OutputType	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
OutputValue	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–
Done	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Busy	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
CommandAborted	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Error	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
ErrorID	HSC_ERROR																			
Index	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–
Position	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–

■ Function

This function block triggers a hardware high-speed output port and retains it for a certain period when the current count value of the counter is equal to the set comparison value.

The step comparison consistence output function is a single trigger. When Done is TRUE, the comparison consistency function is completed. To continue using the comparison function, trigger Execute again.

Comparison output width can be calculated based on time and pulse. When the count value is equal to the set comparison value, Done is set to TRUE, and the comparison output Y point is opened. It retains open for a period of OutputValue x 100 us (time-based) or OutputValue (pulse-based), and then the Y point is closed.

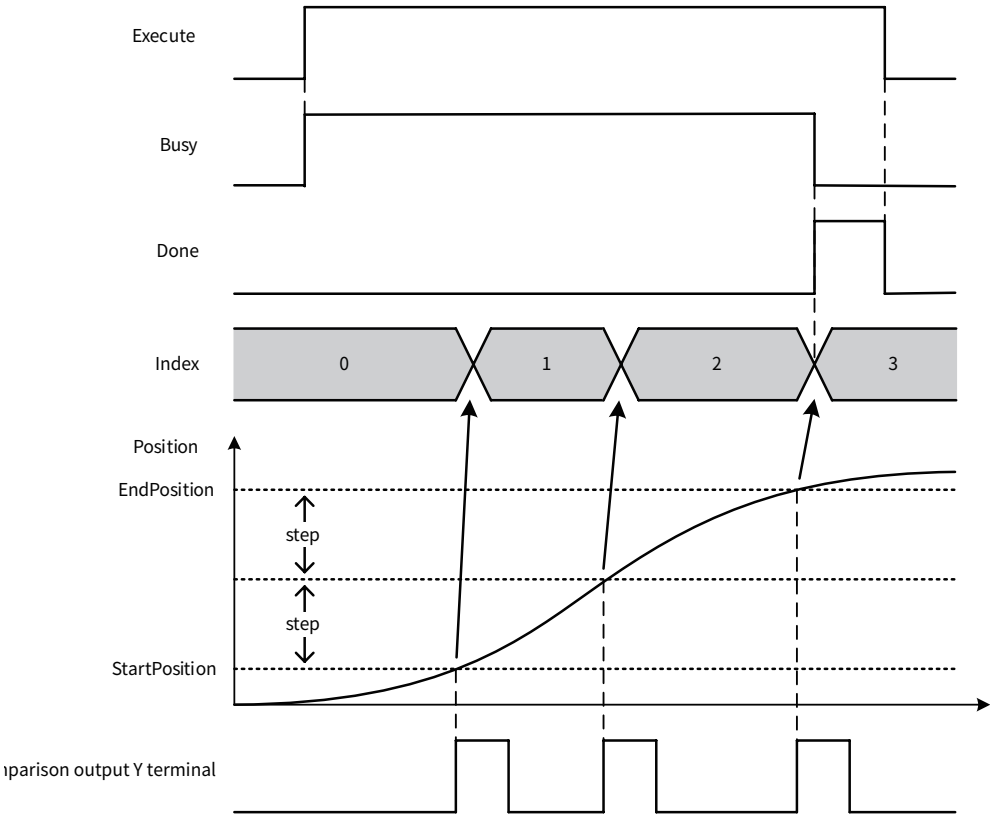
Before using the comparison function block, the output terminal Y must be configured on the corresponding counter interface, and the input terminals can be configured to any from Y0 to Y3.

For step comparison, the distance between any two adjacent points must be greater than the minimum distance (minimum distance = count frequency x scan period). Otherwise, the function block cannot execute the remaining comparison points in linear mode.

During the step comparison, when Execute is triggered at a rising edge again, the function block will restart the comparison output function from the first point.

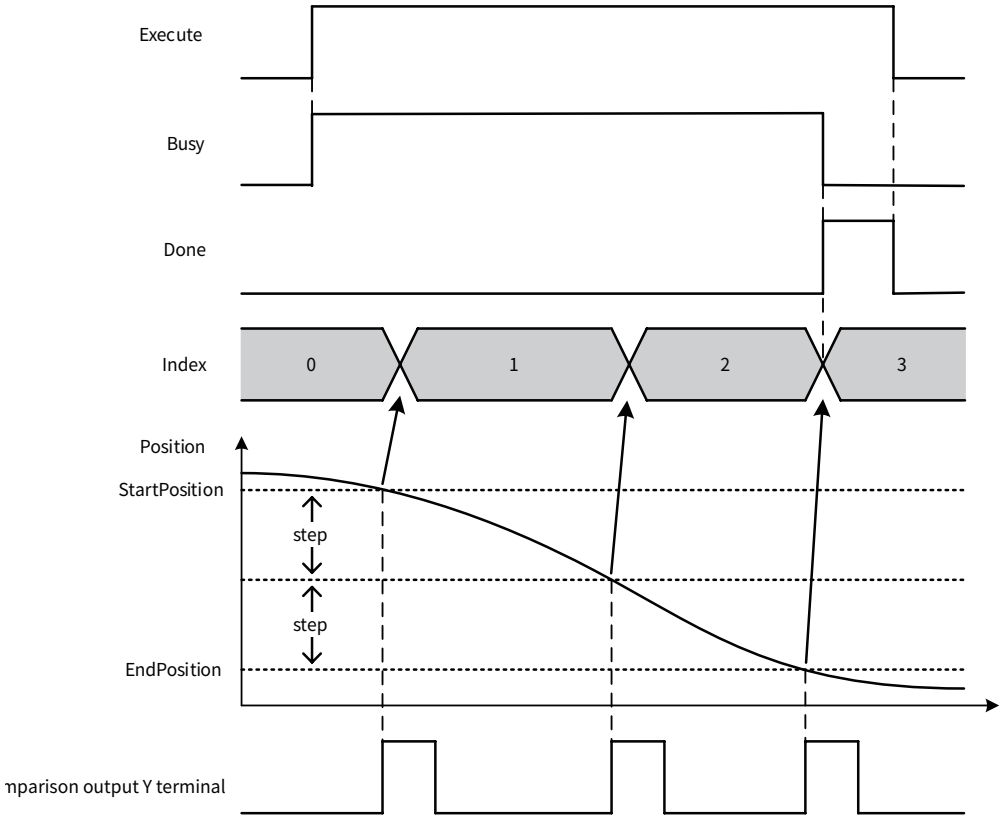
If the interval between two adjacent trigger points is less than the comparison output holding time, the output port will continuously retain the output state. The total output holding time is equal to the current output holding time plus the next output holding time. For the total output holding time of multiple comparison points, use the calculation formula for two adjacent points and then accumulate the output holding time.

The timing diagram of the instruction is as follows when StartPosition is less than EndPosition.



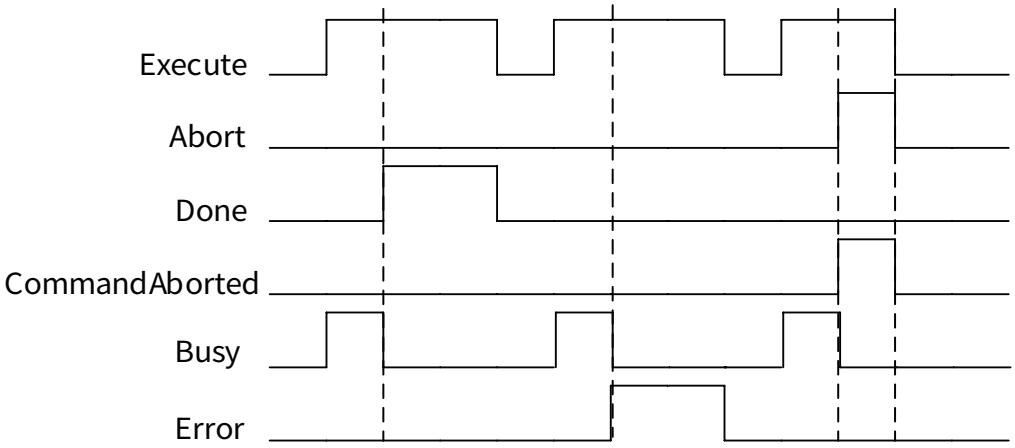
Timing diagram 1 for the HC_StepCompare instruction

The timing diagram of the instruction is as follows when StartPosition is greater than EndPosition.



Timing diagram 2 for the HC_StepCompare instruction

■ Timing diagram



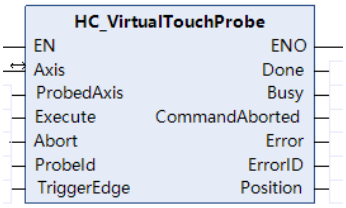
■ Error description

ErrorID in HSC_ERROR specifies the reason for the corresponding error.

5.1.8 HC_VirtualTouchProbe

This instruction implements the virtual probe of the high-speed counter.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
HC_VirtualTouchProbe	Virtual probe	FB		HC_VirtualTouchProbe(Axis:= , ProbedAxis:= , Execute:= , Abort:= , ProbeId:= , TriggerEdge:= , Done=> , Busy=> , CommandAborted=>, Error=> , ErrorID=> , Position=>);

■ Variables

In-out variables

In-Out Variable	Name	Data Type	Value Range	Initial Value	Description
Axis	Encoder axis	ENCODER_REF_INOVANCE	-	-	ENCODER_REF_INOVANCE encoder axis example

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
ProbedAxis	Axis to detect	AIXS_REF_SM3	-	-	Imaginary axis, encoder axis, and EtherCAT real axis
Execute	Enable	BOOL	[FALSE, TRUE]	FALSE	(Triggered by edges) Execution of the probe function switch
Abort	Abort	BOOL	[FALSE, TRUE]	FALSE	Triggered by level TRUE: Function block is aborted
ProbeId	Probe ID	WORD	[1, 2]	1	Two probes are supported.
TriggerEdge	Trigger edge	WORD	[1, 2]	1	Trigger edge 1: Rising edge of the external input X 2: Falling edge of the external input X

Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
Done	Probe execution comple-tion flag	BOOL	[FALSE, TRUE]	FALSE	Latch function execution completed
Busy	Execution flag	BOOL	[FALSE, TRUE]	FALSE	Execution of the count latch function
CommandAborted	Aborted state	BOOL	[FALSE, TRUE]	FALSE	Aborted state, which is set when Abort is TRUE
Error	Error flag	BOOL	[FALSE, TRUE]	FALSE	TRUE: An internal error occurs in the function block
ErrorID	Error ID	HSC_ERROR	-	HSC_NO_ERROR	Error ID For detail, see HSC_ER-ROR.
Position	Latched position	LREAL	(– 1.7E 308, 1.7E 308)	0.0	Latched position, in the unit of unit

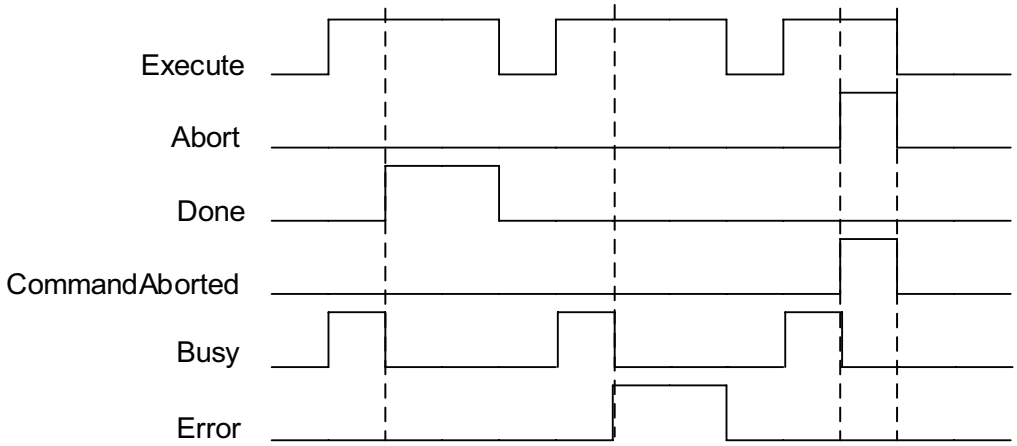
	Bool-ean	Bit String				Integer								Real Num-ber		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Axis	ENCODER_REF_INOVANCE																			
ProbedAxis	AXIS_REF_SM3																			
Execute	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Abort	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
ProbeId	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
TriggerEdge	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Done	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Busy	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

	Bool-ean	Bit String				Integer								Real Num-ber		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
CommandAborted	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Error	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
ErrorID	HSC_ERROR																			
Position	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–

■ Function

This function block utilizes the counter probe function to latch the position of ProbedAxis when the signal triggered at the external X terminal edge is active. There may be a certain accuracy deviation between the latched position and the actual value.

■ Timing diagram



■ Error description

ErrorID in HSC_ERROR specifies the reason for the corresponding error.

5.1.9 HC_ChangeGearingRatio

This instruction implements the high-speed counter set gearing ratio.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
HC_ChangeGearingRatio	High-speed counter set gearing ratio	FB	HC_ChangeGearingRatio EN ENO Axis Done Execute Busy RatioNum Error RatioDenom ErrorID	HC_ChangeGearingRatio(Axis:= , Execute:= , RatioNum:= , RatioDenom:= , Done=> , Busy=> , Error=> , ErrorID=>);

■ Variables

In-out variables

In-Out Variable	Name	Data Type	Value Range	Initial Value	Description
Axis	Encoder axis	ENCODER_REF_INOVANCE	-	-	ENCODER_REF_INOVANCE encoder axis example

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
Execute	Enable	BOOL	[FALSE, TRUE]	FALSE	(Triggered by edges) Execution of the probe function switch
RatioNum	Scaling numerator	UDINT	[1, 2^32 − 1]	10000	Numerator of the gearing ratio for conversion between the unit in application and pulse unit
RatioDenom	Scaling denominator	LREAL	(− 1.7E 308, 1.7E 308)	1.0	Denominator of the gearing ratio for conversion between the unit in application and pulse unit

Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
Done	Probe execution completion flag	BOOL	[FALSE, TRUE]	FALSE	Latch function execution completed
Busy	Execution flag	BOOL	[FALSE, TRUE]	FALSE	Execution of the count latch function
Error	Error flag	BOOL	[FALSE, TRUE]	FALSE	TRUE: An internal error occurs in the function block
ErrorID	Error ID	HSC_ERROR	-	HSC_NO_ERROR	Error ID For detail, see HSC_ERROR.

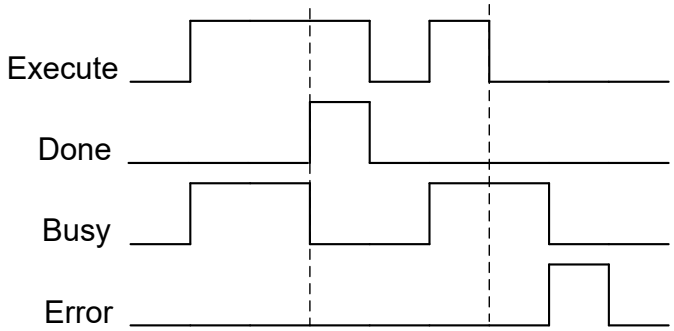
	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Axis	ENCODER_REF_INOVANCE																			
Execute	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
RatioNum	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–
RatioDenom	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–
Done	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Busy	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Error	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
ErrorID	HSC_ERROR																			

■ Function

This function block implements online modification of the gearing ratio.

Before modification, the counter must be disconnected.

■ Timing diagram



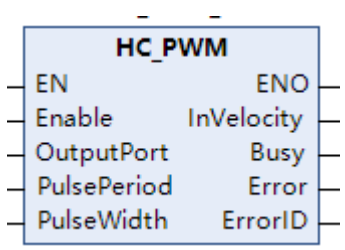
■ Error description

ErrorID in HSC_ERROR specifies the reason for the corresponding error.

5.1.10 HC_PWM

This instruction implements high-speed counter PWM.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
HC_PWM	High-speed counter PWM	FB		HC_PWM(Enable:= , OutputPort:= , PulsePeriod:= , PulseWidth:= , InVelocity=> , Busy=> , Error=> , ErrorID=>);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
Enable	Enable	BOOL	[FALSE, TRUE]	FALSE	Triggered by level
OutputPort	Pulse output port	BYTE	[0, 2]	0	PWM output port. The options are only Y0 and Y2.
PulsePeriod	Pulse width	DINT	[0, 2^31 − 1]	0	Period: Width of a complete square wave Unit: 100 ns
PulseWidth	High-level pulse width	DINT	[0, 2^31 − 1]	0	Square-wave high-level pulse width Unit: 100 ns

Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
InVelocity	Frequency reference reached flag	BOOL	[FALSE, TRUE]	FALSE	Pulse frequency reaching the reference
Busy	Execution flag	BOOL	[FALSE, TRUE]	FALSE	Execution of the output PWM square wave
Error	Error flag	BOOL	[FALSE, TRUE]	FALSE	TRUE: An internal error occurs in the function block
ErrorID	Error ID	HSC_ERROR	-	HSC_NO_ERROR	Error ID For detail, see HSC_ERROR.

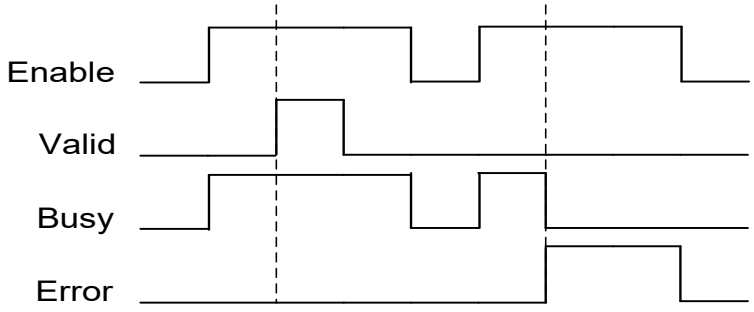
	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Enable	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
OutputPort	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
PulsePeriod	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–
PulseWidth	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–
InVelocity	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Busy	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Error	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
ErrorID	HSC_ERROR																			

■ Function

This function block implements dynamic PWM.

When the PWM function is configured for the Y0 or Y2 terminal, Y0+Y1/Y2+Y3 output cannot be controlled on the device I/O mapping interface of the user program or programming software during execution of the HC_PWM function block.

■ Timing diagram



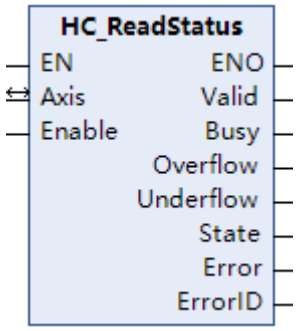
■ Error description

ErrorID in HSC_ERROR specifies the reason for the corresponding error.

5.1.11 HC_ReadStatus

This instruction reads the status of the high-speed counter.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
HC_ReadStatus	High-speed counter read status	FB		HC_ReadStatus(Axis:= , Enable:= , Valid=> , Busy=> , Overflow=> , Underflow=> , State=> , Error=> , ErrorID=>)

■ Variables

In-out variables

In-Out Variable	Name	Data Type	Value Range	Initial Value	Description
Axis	Encoder axis	ENCODER_REF_INOVANCE	-	-	ENCODER_REF_INOVANCE encoder axis example

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
Enable	Enable	BOOL	[FALSE, TRUE]	FALSE	Triggered by level

Output variables

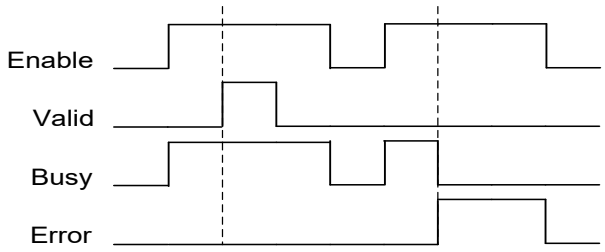
Output Variable	Name	Data Type	Value Range	Initial Value	Description
Valid	Output data valid flag	BOOL	[FALSE, TRUE]	FALSE	Data output valid flag
Busy	Execution flag	BOOL	[FALSE, TRUE]	FALSE	Reading the status
Overflow	Overflow flag	BOOL	[FALSE, TRUE]	FALSE	Count overflow flag
Underflow	Underflow flag	BOOL	[FALSE, TRUE]	FALSE	Count underflow flag
State	State	SMC_AXIS_STATE	-	power_off	Counter enable status
Error	Error flag	BOOL	[FALSE, TRUE]	FALSE	TRUE: An internal error occurs in the function block
ErrorID	Error ID	HSC_ERROR	-	HSC_NO_ERROR	Error ID For detail, see HSC_ERROR.

	Bool-ean	Bit String				Integer								Real Num-ber		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Axis	ENCODER_REF_INOVANCE																			
Enable	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Valid	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Busy	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Overflow	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Underflow	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Error	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
ErrorID	HSC_ERROR																			

■ Function

This function block reads the status of the high-speed counter.

■ Timing diagram



■ Error description

ErrorID in HSC_ERROR specifies the reason for the corresponding error.

Errors specified by HSC_ERROR are listed in the following table.

Error Code	Enumerator	Description
0x00	HSC_NO_ERROR	No error occurs.
0x01	HSC_ERROR_COMMUNICATION	Communication fault
0x02	HSC_ERROR_NOT_SUPPORT_ONLINE_CHANGE	Online error handling for the counter
0x03	HSC_ERROR_UNDEFINED_COUNTER	Use of an undefined counter
0x10	HSC_ERROR_DISABLE	Counter disabled
0x11	HSC_ERROR_PV_DI_CONFIGURE_ERROR	Preset, preset function not configured for the DI terminal
0x12	HSC_ERROR_PV_TYPE_INVALID	Preset, invalid preset type
0x13	HSC_ERROR_PV_PRESET_VALUE_INVALID	Preset, invalid preset value
0x20	HSC_ERROR_CP_DO_CONFIGURE_ERROR	Comparison consistence output function not configured for the DO output terminal
0x21	HSC_ERROR_CP_CHANNEL_INVALID	Invalid channel parameter setting for the comparison consistence function
0x22	HSC_ERROR_CP_COMPARE_VAULE_INVALID	Invalid comparison position

Error Code	Enumerator	Description
0x23	HSC_ERROR_CP_START_POS_INVALID	Invalid start position for isometric comparison
0x24	HSC_ERROR_CP_END_POS_INVALID	Invalid end position for isometric comparison
0x25	HSC_ERROR_CP_STEP_LEN_INVALID	Invalid step for isometric comparison
0x26	HSC_ERROR_CP_ARRAY_ADDR_INVALID	Invalid array head address for array comparison
0x27	HSC_ERROR_CP_ARRAY_COUNT_INVALID	Invalid number of arrays for array comparison
0x28	HSC_ERROR_CP_OUTPUT_TYPE_INVALID	Invalid DO output type for comparison consistence
0x29	HSC_ERROR_CP_OUTPUT_VALUE_INVALID	Invalid DO output holding parameters for comparison consistence
0x30	HSC_ERROR_CP_CHANNEL_ACTIVE	Failed to interrupt the locked channels for comparison consistence
0x40	HSC_ERROR_TP_DI_CONFIGURE_ERROR	Probe function not configured for the DI terminal
0x41	HSC_ERROR_TP_ID_INVALID	Invalid probe ID
0x42	HSC_ERROR_TP_TRGIGGER_TYPE_INVALID	Trigger type of the probe exceeding the valid range
0x43	HSC_ERROR_TP_EDG_TYPE_INVALID	Edge-based trigger type of the probe exceeding the valid range
0x44	HSC_ERROR_TP_MODE_INVALID	Trigger mode value of the probe exceeding the valid range
0x45	HSC_ERROR_TP_CHANNEL_ACTIVE	Failed to interrupt the locked channels for the probe
0x50	HSC_ERROR_LP_ID_INVALID	Invalid ID for the latched position
0x51	HSC_ERROR_LP_TRGIGGER_TYPE_INVALID	Trigger type of the latched position exceeding the valid range
0x60	HSC_ERROR_CGR_COUNTER_MUST_DISABLE	Counter must be disabled before gearing ratio modification
0x61	HSC_ERROR_CGR_RATIO_VAULE_ERROR	Invalid ratio for gearing ratio modification
0x70	HSC_ERROR_R_NO_ERROR_CLEAR	No error cleared
0x71	HSC_ERROR_R_ERROR_CANNOT_RESET	Errors cannot be cleared
0x80	HSC_ERROR_SDO_READ_ERROR	SDO failed to read parameters
0x500	HSC_ERROR_INPUT_PUSLE_FREQUENCY_TOO_HIGH	Input frequency greater than 202 kHz
0x501	HSC_ERROR_OVERFLOW	Count value overflow
0x502	HSC_ERROR_UNDERFLOW	Count value underflow
0x503	HSC_ERROR_FILT_PARAM_EXCEED	Filter coefficient exceeding the valid range
0x520	HSC_ERROR_PARAM_NO_PDO_MAPPING	No PDO configured for the instruction

5. 2 AM600 Local Counter Instructions

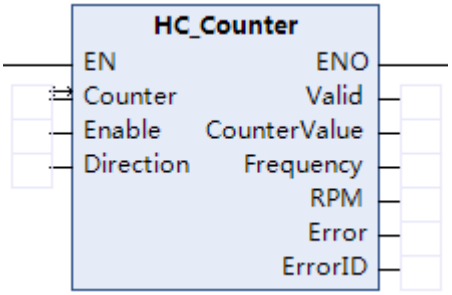
5.2. 1 Instruction List

Instruction Category	Name	FB/FC	Function
AM600 local counter instructions	HC_Counter	FB	Enable counter
	HC_SetCompare	FB	Set counter comparison output
	HC_PresetValue	FB	Set counter value from preset value
	HC_EnableInterrupt	FB	Interrupt counter
	HC_TouchProbe	FB	Counter probe latch
	HC_MeasurePulseWidth	FB	Measure counter pulse width
	HC_Sample	FB	Counter start to sample
	HC_ReadBoolParameter	FB	Read counter boolean parameters
	HC_WriteBoolParameter	FB	Write counter boolean parameters
	HC_SetCompareM	FB	Counter multi-segment comparison
	HC_SetRing	FB	Set counter mode
	HC_ResetCmpOutput	FB	Reset comparison output
	HC_WriteInterruptParameter	FB	Write counter interrupt parameters

5.2. 2 HC_Counter

This instruction enables the counter.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
HC_Counter	Enable counter	FB		HC_Counter(Counter := , Enable := , Direction := , Valid => , CounterValue=> , Frequency => , RPM => , Error=> , ErrorID=>);

■ Variables

In-out variables

In-Out Variable	Name	Data Type	Value Range	Initial Value	Description
Counter	Counter	COUNTER_REF	-	-	Number of the specified counter to use, ranging from 0 to 7

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
Enable	Enable	BOOL	[FALSE, TRUE]	FALSE	Triggered by level TRUE: Enable the counter
Direction	Direction	BOOL	[FALSE, TRUE]	FALSE	0: Incremental count 1: Decremental count This variable is valid only for the single-phase input or input mode. This variable is valid only when the energy flow changes.

Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
CounterValue	Current count value	DINT	Depends on data types	-	Current value of the counter
Frequency	Pulse frequency	UDINT	Depends on data types	-	Pulse frequency measured value, in the unit of Hz
RPM	Rotation speed	REAL	Depends on data types	-	Revolution per minute, in the unit of r/min
Valid	Enabled	BOOL	Depends on data types	-	1: Enabled 0: Disabled
Error	Error flag	BOOL	[FALSE, TRUE]	-	TRUE: An internal error occurs in the function block
ErrorID	Error ID	UINT	Depends on data types	-	Error code

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Counter	COUNTER_REF																			
Enable	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Direction	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
CounterValue	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–
Frequency	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–
RPM	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–
Valid	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Error	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
ErrorID	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–

■ Function

This function block primarily controls the start/stop of a local counter. When Enable is set to TRUE, the counter starts counting.

The following table lists the resource allocation for counters and count ports.

Count Mode	Terminal Counter	X0	X1	X2	X3	X4	X5	X6	X7	X8	X9	Xa	Xb	Xc	Xd	Xe	Xf
AB Phase A/B count	Counter0	A	B							T							
	Counter1			A	B						T						
	Counter2 (△)					A	B					T					
	Counter3 (△)							A	B				T				
	Counter4 (△)									A	B						
	Counter5 (△)											A	B				
	Counter6 (△)													A	B		
	Counter7 (△)															A	B
Phase A Count	Counter0	A															
	Counter1		A														
	Counter2			A													
	Counter3				A												
	Counter4 (△)					A											
	Counter5 (△)						A										
	Counter6 (△)							A									
	Counter7 (△)								A								

(△): indicates that FrequencyValue and RotationRateValue are calculated by using software.

When phase A/B count is selected for Counter0 and Counter1, only phase A frequency is displayed.

T indicates an externally triggered signal.

■ Precautions

The first four counters and last four counters are different as follows:

Forward count: When the count value reaches 2147483647, the first four counters stop counting and the last four counters start counting from －2147483648.

Reverse count: When the count value reaches －2147483648, the first four counters stop counting and the last four counters start counting from 2147483647.

Linear count and ring count are different as follows:

Linear count: (DownLimitValue, UpLimitValue)

Ring count: [iRingDownValue, iRingUpValue]

Counters 0 to 3: The linear count range is －2147483648 to +2147483647, excluding －2147483648 and +2147483647.

The ring count range is －2147483648 to +2147483647, including －2147483648 and +2147483647.

Counters 4 to 7: The linear count range is －2147483648 to +2147483647, excluding －2147483648 and +2147483647.

Counters 4 to 7: If the count value is reaching the boundary value, no error may occur upon counter overflow due to the scan period.

During program download, the counter value is not cleared.

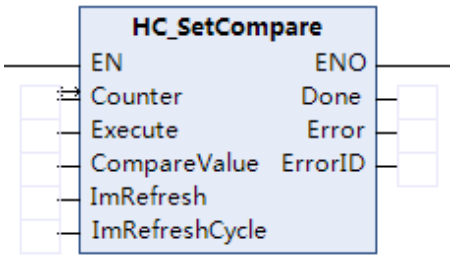
Either the HC_Counter or HC_MeasurePulseWidth function block can be used in the program each time.

The HC_Sample function block can be used only after HC_Counter is enabled.

5.2. 3 HC_SetCompare

This instruction sets the comparison consistence output.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
HC_SetCompare	Set counter comparison output	FB		HC_SetCompare(Counter := , Execute := , CompareValue:= , ImRefresh:= , ImRefreshCycle:=, Done => , Error => , ErrorID =>);

■ Variables

In-out variables

In-Out Variable	Name	Data Type	Value Range	Initial Value	Description
Counter	Counter	COUNTER_REF	-	-	Number of the specified counter to use, ranging from 0 to 7

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
Execute	Enable	BOOL	[FALSE, TRUE]	FALSE	Trigger at the rising edge TRUE: Enable the counter
CompareValue	Comparison value	DINT	(– 2147483648, 2147483647)	-	Setting the comparison value for the counter
ImRefresh	Output type	BOOL	[FALSE, TRUE]	-	0: Software output, with the output port configured in the background 1: Hardware output Counter 0 outputs Y0. Counter 1 outputs Y1. ... Counter 7 outputs Y7.
ImRefreshCycle	Output time	UINT	[0, 30000]	-	The unit is 100 us, and the maximum output time is 3000 ms. For example, if the value is 10000, the output time is 1000 ms.

Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
Done	Completion flag	BOOL	[FALSE, TRUE]	-	Completion flag of the function block

Output Variable	Name	Data Type	Value Range	Initial Value	Description
Error	Error flag	BOOL	[FALSE, TRUE]	-	TRUE: An internal error occurs in the function block
ErrorID	Error ID	UINT	Depends on data types	-	Error code

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Counter	COUNTER_REF																			
Execute	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
CompareValue	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–
ImRefresh	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
ImRefreshCycle	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–
Done	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Error	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
ErrorID	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–

■ Function

This function block implements comparison consistence output for the local counter. The HC_EnableInterrupt instruction is required to interrupt comparison consistence.

■ Precautions

To control Y0 to Y7, ImRefresh can be set to 0 or 1.

ImRefresh = 0: Software output, configured in the background. Each counter can select Y0 to Y7 without limit, but the output may be delayed.

The output time range is from 100 us to 3000 ms.

If ImRefreshCycle is set to 0, the HC_ResetCmpOutput instruction is used to reduce the output.

ImRefresh = 1: Immediate hardware output. The counters cannot freely select Y0 to Y7, and there is no delay in the output. The output time is determined by ImRefreshCycle.

The output time range is from 0 to 3000 ms.

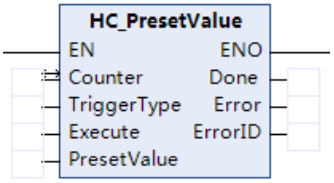
The HC_EnableInterrupt instruction needs to be used in advance to interrupt comparison consistence.

After a hot or cold reset, the previous comparison value is maintained.

5.2. 4 HC_PresetValue

This instruction writes the preset value.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
HC_PresetValue	Set counter value from preset value	FB		HC_PresetValue(Counter := , TriggerType := , Execute := , PresetValue:= , Done => , Error => , ErrorID =>);

■ Variables

In-out variables

In-Out Variable	Name	Data Type	Value Range	Initial Value	Description
Counter	Counter	COUNTER_REF	-	-	Number of the specified counter to use, ranging from 0 to 7

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
TriggerType	Trigger type	BYTE	Depends on data types	-	0: Trigger at the rising edge 1: Trigger by external input External signals: Couter0 -> x8 Couter1 -> x9 Couter2 -> xa Couter3 -> xb 2: Trigger at the rising edge, which is preset upon comparison consistence output
Execute	Enable	BOOL	[FALSE, TRUE]	FALSE	Trigger at the rising edge TRUE: Enable the function block
PresetValue	Preset value	DINT	(– 2147483648, 2147483647)	-	Preset value of the counter

Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
Done	Completion flag	BOOL	[FALSE, TRUE]	-	Completion flag of the function block
Error	Error flag	BOOL	[FALSE, TRUE]	-	TRUE: An internal error occurs in the function block
ErrorID	Error ID	UINT	Depends on data types	-	Error code

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Counter	COUNTER_REF																			
TriggerType	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Execute	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
PresetValue	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–
Done	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Error	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
ErrorID	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–

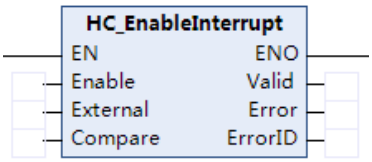
■ Function

This function block implements the preset function of the counter. Execution of the function block is triggered at the rising edge, and the Done signal is output after the counter is preset.

5.2. 5 HC_EnableInterrupt

This instruction enables interruption.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
HC_EnableInterrupt	Interrupt counter	FB		HC_EnableInterrupt(Enable := , External := , Compare := , Valid => , Error => , ErrorID =>);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
Enable	Action instruction	BOOL	[FALSE, TRUE]	FALSE	Action interrupt enable TRUE: Enabled FALSE: Disabled
External	External input interrupt	UINT	Depends on data types	-	External input interrupt enable For example, the binary mode for digit 3 is 2#11, indicating that bits 0 and 1 of the port are enabled.
Compare	Comparison consistence interrupt	UINT	Depends on data types	-	Comparison consistence interrupt enable For example, the binary mode for digit 3 is 2#11, indicating that bits 0 and 1 of the port are enabled.

Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
Valid	Interrupt enabled	BOOL	[FALSE, TRUE]	-	Interrupt enabled
Error	Error flag	BOOL	[FALSE, TRUE]	-	TRUE: An internal error occurs in the function block
ErrorID	Error ID	UINT	Depends on data types	-	Error code

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Counter	COUNTER_REF																			
Execute	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

	Bool-ean	Bit String				Integer								Real Num-ber		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
CompareValue	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–
ImRefresh	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
ImRefreshCycle	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–
Done	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Error	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
ErrorID	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–

■ Function

This function block enables external interrupt and comparison consistence interrupt for the local counter.

■ Precautions

If HC_WriteInterruptParameter is executed before HC_EnableInterrupt, the HC_WriteInterruptParameter parameter is valid.

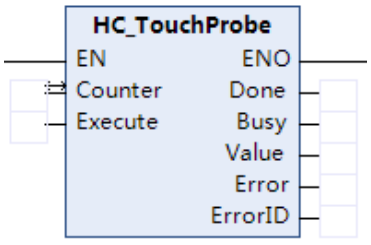
If HC_EnableInterrupt is executed before HC_WriteInterruptParameter, background interrupt parameters are valid.

When HC_WriteInterruptParameter is executed again, the HC_WriteInterruptParameter parameter is valid.

5.2. 6 HC_TouchProbe

This instruction implements the probe function.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
HC_TouchProbe	Counter probe latch	FB		HC_TouchProbe (Counter := , Execute := , Done => , Busy => , Value => , Error => , ErrorID =>);

■ Variables

In-out variables

In-Out Variable	Name	Data Type	Value Range	Initial Value	Description
Counter	Counter	COUNTER_REF	-	-	Number of the specified counter to use, ranging from 0 to 7

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
Execute	Enable	BOOL	[FALSE, TRUE]	FALSE	Trigger at the rising edge TRUE: Enable the function block

Output variables

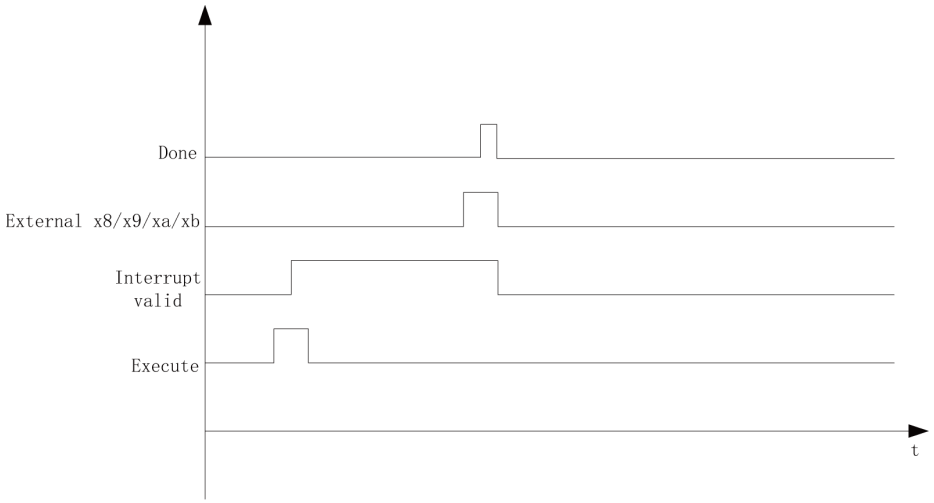
Output Variable	Name	Data Type	Value Range	Initial Value	Description
Done	Completion flag	BOOL	[FALSE, TRUE]	-	Completion flag of the function block
Busy	Execution flag	BOOL	[FALSE, TRUE]	-	Function block execution flag
Value	Latch value	DINT	Depends on data types	-	Latch value
Error	Error flag	BOOL	[FALSE, TRUE]	-	TRUE: An internal error occurs in the function block
ErrorID	Error ID	UINT	Depends on data types	-	Error code

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Counter	COUNTER_REF																			
Execute	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Done	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Busy	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Value	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–
Error	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
ErrorID	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–

■ Function

This function block implements the probe function for the counter. In case of external interrupt, the current value (0 to 3) of the counter is read and latched.

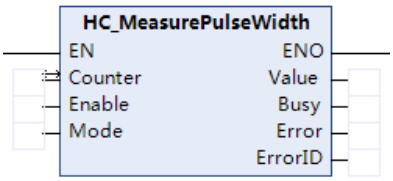
■ Timing diagram


Done
External x8/x9/xa/xb
Interrupt valid
Execute
t

5.2. 7 HC_MeasurePulseWidth

This instruction measures the pulse width.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
HC_MeasurePulseWidth	Measure counter pulse width	FB		HC_MeasurePulseWidth(Counter := , Enable := , Mode := , Value => , Busy => , Error => , ErrorID =>);

■ Variables

In-out variables

In-Out Variable	Name	Data Type	Value Range	Initial Value	Description
Counter	Counter	COUNTER_REF	-	-	Number of the specified counter to use, ranging from 0 to 7

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
Enable	Enable	BOOL	[FALSE, TRUE]	FALSE	Triggered by level TRUE: Enable the function block
Mode	Mode	BOOL	[FALSE, TRUE]	FALSE	FALSE: External signal high level (measuring the high level pulse width) TRUE: External signal low level (measuring the low level pulse width)

Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
Value	Pulse width value	UDINT	Depends on data types	-	Read measured value of the pulse width. The unit is us and the value range is [0, 4000000].
Busy	Execution flag	BOOL	[FALSE, TRUE]	-	Function block execution flag
Error	Error flag	BOOL	[FALSE, TRUE]	-	TRUE: An internal error occurs in the function block
ErrorID	Error ID	UINT	Depends on data types	-	Error code

	Bool-ean	Bit String				Integer								Real Num-ber		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Counter	COUNTER_REF																			
Enable	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Mode	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Value	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–
Busy	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Error	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
ErrorID	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–

■ Function

This function block implements the pulse width measurement function for the counter. When external triggering is valid, the measured pulse width values of x8, x9, xa, and xb are read.

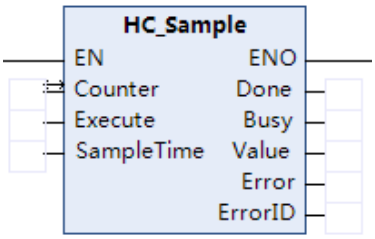
■ Precautions

Either the HC_Counter or HC_MeasurePulseWidth function block can be used in the program each time.

5.2. 8 HC_Sample

This instruction implements sampling for the counter.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
HC_Sample	Counter start to sample	FB		HC_Sample (Counter := , Execute := , SampleTime := , Done => , Busy => , Value => , Error => , ErrorID =>);

■ Variables

In-out variables

In-Out Variable	Name	Data Type	Value Range	Initial Value	Description
Counter	Counter	COUNTER_REF	-	-	Number of the specified counter to use, ranging from 0 to 7

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
Execute	Enable	BOOL	[FALSE, TRUE]	FALSE	Trigger at the rising edge TRUE: Enable the function block
SampleTime	Sampling time	UINT	Depends on data types	-	Sampling time. The unit is ms and the value range is from 10 to 65535.

Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
Done	Completion flag	BOOL	[FALSE, TRUE]	-	Completion flag of the function block
Busy	Execution flag	BOOL	[FALSE, TRUE]	-	Function block execution flag
Value	Sampling value	DINT	Depends on data types	-	Sampling value
Error	Error flag	BOOL	[FALSE, TRUE]	-	TRUE: An internal error occurs in the function block
ErrorID	Error ID	UINT	Depends on data types	-	Error code

	Bool-ean	Bit String				Integer								Real Num-ber		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Counter	COUNTER_REF																			
Execute	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
SampleTime	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–
Done	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Busy	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Value	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–
Error	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
ErrorID	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–

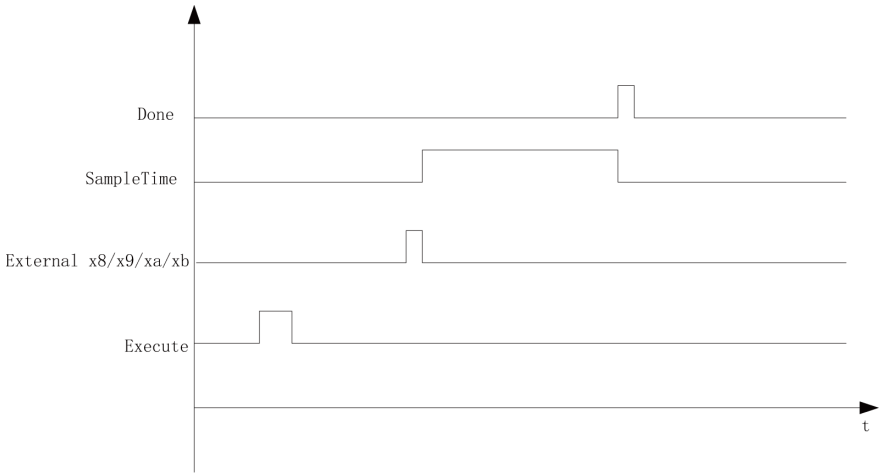
■ Function

This function block implements the sampling function of the counter to count the number within a period.

■ Precautions

The HC_Sample function block can be used only after HC_Counter is enabled.

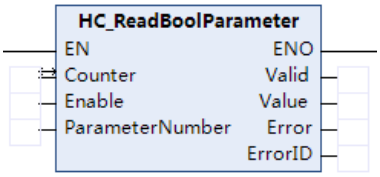
■ Timing diagram


Done
SampleTime
External x8/x9/xa/xb
Execute
t

5.2. 9 HC_ReadBoolParameter

This instruction reads parameters for the counter.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
HC_ReadBoolParameter	Read counter boolean parameters	FB		HC_ReadBoolParameter(Counter := , Enable := , ParameterNumber:= , Valid => , Value => , Error => , ErrorID =>);

■ Variables

In-out variables

In-Out Variable	Name	Data Type	Value Range	Initial Value	Description
Counter	Counter	COUNTER_REF	-	-	Number of the specified counter to use, ranging from 0 to 7

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
Enable	Enable	BOOL	[FALSE, TRUE]	FALSE	Triggered by level TRUE: Enable the function block
ParameterNumber	Parameter Number	DINT	Depends on data types	-	Parameter Number

Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
Valid	Enable flag	BOOL	[FALSE, TRUE]	-	Function block valid flag
Value	Value	BOOL	Depends on data types	-	Value
Error	Error flag	BOOL	[FALSE, TRUE]	-	TRUE: An internal error occurs in the function block
ErrorID	Error ID	UINT	Depends on data types	-	Error code

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Counter	COUNTER_REF																			
Enable	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
ParameterNumber	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–

	Bool-ean	Bit String				Integer								Real Num-ber		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Valid	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Value	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Error	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
ErrorID	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–

■ Function

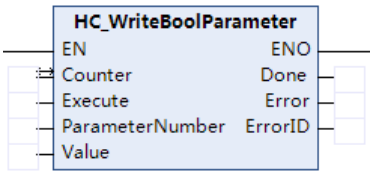
This function block reads counter parameters. The following table lists the PN parameter mappings.

PN	Name	DataByte	R/W	Comments
10001	bDisableCmpEvent	Bool	R/W	External comparison event of user tasks 0: Enabled 1: Disabled
10002	bDisableOutput	Bool	R/W	Output of the output port for comparison consistence 0: No output 1: Output This parameter is invalid for ImRefresh.
10008	Direction	Bool	R/W	Counter direction
10009	bDisableFrequency	Bool	R/W	This parameter disables FrequencyValue for the HC_ Counter instruction.
10010	bDisableRotationRate	Bool	R/W	This parameter disables RotationRateValue for the HC_ Counter instruction.

5.2. 10 HC_WriteBoolParameter

This instruction writes parameters for the counter.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
HC_WriteBoolParameter	Write counter boolean parame-ters	FB		HC_WriteBoolParameter(Counter := , Execute := , ParameterNumber:= , Value := , Done => , Error => , ErrorID =>);

■ Variables

In-out variables

In-Out Variable	Name	Data Type	Value Range	Initial Value	Description
Counter	Counter	COUNTER_REF	-	-	Number of the specified counter to use, ranging from 0 to 7

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
Execute	Enable	BOOL	[FALSE, TRUE]	FALSE	Trigger at the rising edge TRUE: Enable the function block
ParameterNumber	Parameter Number	DINT	Depends on data types	-	Parameter Number
Value	Value	BOOL	Depends on data types	-	Value

Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
Done	Completion flag	BOOL	[FALSE, TRUE]	-	Completion flag of the function block
Error	Error flag	BOOL	[FALSE, TRUE]	-	TRUE: An internal error occurs in the function block
ErrorID	Error ID	UINT	Depends on data types	-	Error code

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Counter	COUNTER_REF																			
Execute	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
ParameterNumber	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–
Value	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Done	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Error	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
ErrorID	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–

■ Function

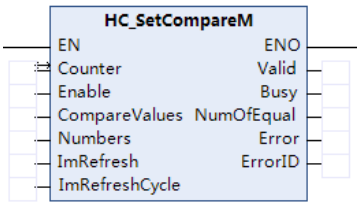
This function block writes counter parameters. The following table lists the PN parameter mappings.

PN	Name	DataByte	R/W	Comments
10001	bDisableCmpEvent	Bool	R/W	External comparison event of user tasks 0: Enabled 1: Disabled
10002	bDisableOutput	Bool	R/W	Output of the output port for comparison consistence 0: No output 1: Output This parameter is invalid for ImRefresh.
10008	Direction	Bool	R/W	Counter direction
10009	bDisableFrequency	Bool	R/W	This parameter disables FrequencyValue for the HC_ Counter instruction.
10010	bDisableRotationRate	Bool	R/W	This parameter disables RotationRateValue for the HC_ Counter instruction.

5.2. 11 HC_SetCompareM

This instruction implements comparison consistence output.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
HC_SetCompareM	Counter multi-segment comparison	FB		HC_SetCompareM(Counter := , Enable := , CompareValues:= , Numbers := , ImRefresh:= , ImRefreshCycle:= , Valid => , Busy => , NumOfEqual => , Error => , ErrorID =>);

■ Variables

In-out variables

In-Out Variable	Name	Data Type	Value Range	Initial Value	Description
Counter	Counter	COUNTER_REF	-	-	Number of the specified counter to use, ranging from 0 to 7

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
Enable	Enable	BOOL	[FALSE, TRUE]	FALSE	Triggered by level TRUE: Enable the function block
CompareValues	Comparison value array	-	-	-	One-dimensional array of comparison values for the counter. A maximum of 100 comparison values are supported. The range for each value is (– 2147483648, 2147483647).
Numbers	Comparison value array size	DWORD	Depends on data types	-	Number of comparison values in a one-dimensional array. The maximum value is 100.
ImRefresh	Output type	BOOL	[FALSE, TRUE]	-	0: Software output, with the output port configured in the background 1: Hardware output Counter 0 outputs Y0. Counter 1 outputs Y1. ... Counter 7 outputs Y7.

ImRefreshCycle	Output time	UINT	[0, 30000]	-	The unit is 100 us, and the maximum output time is 3000 ms. For example, if the value is 10000, the output time is 1000 ms.

Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
Valid	Enable flag	BOOL	[FALSE, TRUE]	-	Function block valid flag
Busy	Execution flag	BOOL	[FALSE, TRUE]	-	Function block execution flag
NumOfEqual	Number of equal values	DWORD	Depends on data types	-	Number of equal values
Error	Error flag	BOOL	[FALSE, TRUE]	-	TRUE: An internal error occurs in the function block
ErrorID	Error ID	UINT	Depends on data types	-	Error code

	Bool-ean	Bit String				Integer								Real Num-ber		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Counter	COUNTER_REF																			
Enable	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
CompareValues[]	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–
Numbers	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
ImRefresh	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
ImRefreshCycle	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–
Valid	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Busy	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
NumOfEqual	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Error	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
ErrorID	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–

■ Function

This function block implements the comparison consistence output function for the counter. A maximum of 100 comparison values can be set.

■ Precautions

When Enable is TRUE, the count direction cannot be changed.

Array elements must be gradually larger in the forward direction or smaller in the reverse direction. No equal ones are allowed in an array.

Forward direction: The start value of the counter must be less than the first comparison value. Otherwise, there will be no enough time to execute the first comparison value.

Reverse direction: The start value of the counter must be greater than the first comparison value. Otherwise, there will be no enough time to execute the first comparison value.

The HC_EnableInterrupt instruction needs to be used in advance to interrupt comparison consistence.

The format in the forward direction is as follows:

arr1 :ARRAY [0..99] OF DINT := [2000,4000,6000,8000,10000,12000,14000,16000,18000,20000];

The format in the reverse direction is as follows:

arr2 :ARRAY [0..99] OF DINT := [20000,18000,16000,14000,12000,10000,8000,6000,5000,3000];

This function block depends on HC_Counter.

To control Y0 to Y7, ImRefresh can be set to 0 or 1.

ImRefresh = 0: Software output, configured in the background. Each counter can select Y0 to Y7 without limit, but the output may be delayed.

The output time range is from 100 us to 3000 ms.

If ImRefreshCycle is set to 0, the HC_ResetCmpOutput instruction is used to reduce the output.

ImRefresh = 1: Immediate hardware output. The counters cannot freely select Y0 to Y7, and there is no delay in the output. The output time is determined by ImRefreshCycle.

The output time range is from 0 to 3000 ms.

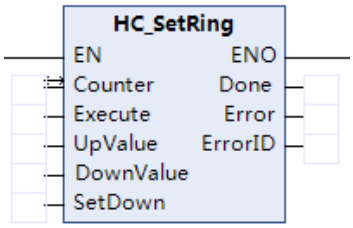
The HC_EnableInterrupt instruction needs to be used in advance to interrupt comparison consistence.

After a hot or cold reset, the previous comparison value is maintained.

5.2. 12 HC_SetRing

This instruction sets the ring count function.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
HC_SetRing	Set counter mode	FB		HC_SetRing(Counter := , Execute := , UpValue:= , DownValue := , SetDown := , Done => , Error => , ErrorID =>);

■ Variables

In-out variables

In-Out Variable	Name	Data Type	Value Range	Initial Value	Description
Counter	Counter	COUNTER_REF	-	-	Number of the specified counter to use, ranging from 0 to 7

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
Execute	Enable	BOOL	[FALSE, TRUE]	FALSE	Trigger at the rising edge TRUE: Enable the function block
UpValue	Upper limit	DINT	Depends on data types	-	Upper limit for ring count
DownValue	Lower limit	DINT	Depends on data types	-	Lower limit for ring count
SetDown	Forced counting from the lower limit	BOOL	Depends on data types	-	Forced counting from the lower limit

Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
Done	Completion flag	BOOL	[FALSE, TRUE]	-	Completion flag of the function block
Error	Error flag	BOOL	[FALSE, TRUE]	-	TRUE: An internal error occurs in the function block
ErrorID	Error ID	UINT	Depends on data types	-	Error code

	Bool-ean	Bit String				Integer								Real Num-ber		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Counter	COUNTER_REF																			
Execute	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
UpValue	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–
DownValue	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–
SetDown	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Done	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Error	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
ErrorID	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–

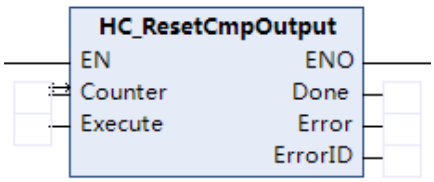
■ Function

This function block implements the ring count function for the counter.

5.2. 13 HC_ResetCmpOutput

This instruction implements the reset comparison output function.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
HC_ResetCmpOutput	Reset comparison output	FB		HC_ResetCmpOutput(Counter := , Execute := , Done => , Error => , ErrorID =>);

■ Variables

In-out variables

In-Out Variable	Name	Data Type	Value Range	Initial Value	Description
Counter	Counter	COUNTER_REF	-	-	Number of the specified counter to use, ranging from 0 to 7

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
Execute	Enable	BOOL	[FALSE, TRUE]	FALSE	Trigger at the rising edge TRUE: Enable the function block

Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
Done	Completion flag	BOOL	[FALSE, TRUE]	-	Completion flag of the function block
Error	Error flag	BOOL	[FALSE, TRUE]	-	TRUE: An internal error occurs in the function block
ErrorID	Error ID	UINT	Depends on data types	-	Error code

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Counter	COUNTER_REF																			
Execute	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Done	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Error	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
ErrorID	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–

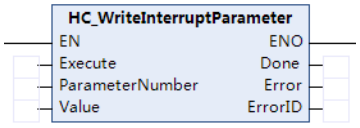
■ Function

This function block implements the reset comparison consistence output function for the counter. The ports (set in the background) for comparison consistence output as well as the hardware output ports (ImRefresh and ImRefreshCycle for HC_SetCompare) are invalid.

5.2. 14 HC_WriteInterruptParameter

This instruction writes interrupt parameters for the counter.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
HC_WriteInterruptParameter	Write counter interrupt parameters	FB		HC_WriteInterruptParameter(Execute := , ParameterNumber => , Value => , Done => , Error => , ErrorID =>);

■ Variables

In-out variables

In-Out Variable	Name	Data Type	Value Range	Initial Value	Description
Counter	Counter	COUNTER_REF	-	-	Number of the specified counter to use, ranging from 0 to 7

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
Execute	Enable	BOOL	[FALSE, TRUE]	FALSE	Trigger at the rising edge TRUE: Enable the function block
ParameterNumber	Parameter Number	DINT	Depends on data types	-	Parameter Number
Value	Value	BOOL	Depends on data types	-	Value

Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
Done	Completion flag	BOOL	[FALSE, TRUE]	-	Completion flag of the function block
Error	Error flag	BOOL	[FALSE, TRUE]	-	TRUE: An internal error occurs in the function block
ErrorID	Error ID	UINT	Depends on data types	-	Error code

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Counter	COUNTER_REF																			
Execute	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
ParameterNumber	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–
Value	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Done	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Error	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
ErrorID	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–

■ Function

This function block writes counter interrupt parameters. The following table lists the PN parameter mappings.

PN	Name	DataByte	R/W	Comments
10001	bDisableCmpEvent	Bool	R/W	External comparison event of user tasks 0: Enabled 1: Disabled
10002	bDisableOutput	Bool	R/W	Output of the output port for comparison consistence 0: No output 1: Output This parameter is invalid for ImRefresh.
10008	Direction	Bool	R/W	Counter direction
10009	bDisableFrequency	Bool	R/W	This parameter disables FrequencyValue for the HC_ Counter instruction.

PN	Name	DataByte	R/W	Comments
10010	bDisableRotationRate	Bool	R/W	This parameter disables RotationRateValue for the HC_ Counter instruction.

5. 3 AM600 Local Pulse Output Instructions

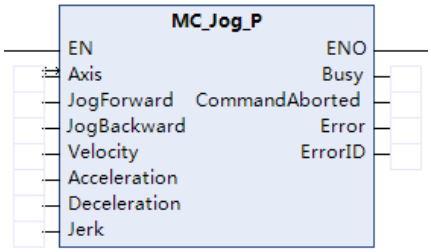
5.3. 1 Instruction List

Instruction Category	Name	FB/FC	Function
AM600 local pulse output instructions	MC_Jog_P	FB	Jog local pulse axis
	MC_Home_P	FB	Local pulse axis homing
	MC_MoveAbsolute_P	FB	Local pulse axis move absolute
	MC_MoveRelative_P	FB	Local pulse axis move relative
	MC_MoveVelocity_P	FB	Local pulse axis move velocity
	MC_Stop_P	FB	Stop pulse axis motion
	MC_Reset_P	FB	Reset pulse axis errors
	MC_Power_P	FB	Enable local pulse axis
	MC_ReadParameter_P	FB	Read pulse axis parameters
	MC_WriteParameter_P	FB	Write pulse axis parameters
	MC_SetPosition_P	FB	Set pulse axis position
	MC_ReadStatus_P	FB	Read pulse axis status

5.3. 2 MC_Jog_P

This instruction implements jogging of high-speed axes.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
MC_Jog_P	Jog local pulse axis	FB		MC_Jog_P (Axis := , JogForward := , JogBackward := , Velocity := , Acceleration:= , Deceleration := , Jerk := , Busy => , CommandAborted=> , Error => , ErrorID =>);

■ Variables

In-out variables

In-Out Variable	Name	Data Type	Value Range	Initial Value	Description
Axis	High-speed axis	HS_AXIS_REF	-	-	Number of the specified high-speed axis to use, ranging from 0 to 3

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
JogForward	Forward direction	BOOL	[FALSE, TRUE]	FALSE	Jogging in forward direction, triggered by level
JogBackward	Negative direction	BOOL	[FALSE, TRUE]	FALSE	Jogging in reverse direction, triggered by level
Velocity	Speed	DWORD	Depends on data types	-	Jogging speed. The unit is pulse/s and the maximum value is 200000.
Acceleration	Acceleration	DWORD	Depends on data types	-	Acceleration rate, which can be changed by dwMaxAcceleration. The unit is pulse/s☒ and the maximum value is 20000000.
Deceleration	Deceleration	DWORD	Depends on data types	-	Deceleration rate, which can be changed by dwMaxDeceleration. The unit is pulse/s☒ and the maximum value is 20000000.
Jerk	Jerk	DWORD	Depends on data types	-	Jerk (pulse/s☒)

Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
Busy	Execution flag	BOOL	[FALSE, TRUE]	-	Function block execution flag
CommandAborted	Aborted state	BOOL	[FALSE, TRUE]	-	Aborted state of the function block
Error	Error flag	BOOL	[FALSE, TRUE]	-	TRUE: An internal error occurs in the function block
ErrorID	Error ID	UINT	Depends on data types	-	Error code

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Axis	HS_AXIS_REF																			
JogForward	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
JogBackward	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Velocity	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Acceleration	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Deceleration	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Jerk	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Busy	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

	Bool-ean	Bit String				Integer								Real Num-ber		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
CommandAborted	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Error	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
ErrorID	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–

■ Function

This function block implements jogging of high-speed axes based on the target velocity specified by Velocity.

Values of Acceleration and Deceleration are integer multiples of 125 x ratio.

$$\text{Rate} = \begin{cases} 1 & \text{Vmax} \le 8000 \\ \frac{\text{Vmax}}{8000} & \text{Vmax} > 8000 \end{cases}$$

$$\text{Vel} = \left[\frac{\text{Vel}}{\text{Rate}}\right] \times \text{Rate}$$

$$\text{Actual acceleration} = \begin{cases} 125 \times \text{Rate} & \text{Acc} \le 125 \times \text{Rate} \\ \left[\frac{\text{Acc}}{125 \times \text{Rate}}\right] \times 125 \times \text{Rate} & \text{Acc} > 125 \times \text{Rate} \end{cases}$$

Values in square brackets ([]) shall be rounded to the nearest integers.

The velocity cannot be changed during acceleration and deceleration.

When the velocity curve is S-curve, jerk is calculated internally. Therefore, all parameters involving Jerk are calculated internally.

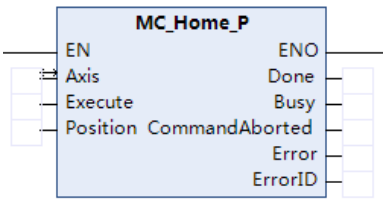
When the velocity curve is S-curve, the velocity, acceleration rate, and deceleration rate cannot be changed during jogging.

When the velocity curve is S-curve and the deceleration rate is greater than or equal to 5 x Velocity, no trailing and truncation occurs.

5.3. 3 MC_Home_P

This instruction implements high-speed axis homing.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
MC_Home_P	Local pulse axis homing	FB		MC_Home_P (Axis := , Execute:= , Position := , Done => , Busy => , CommandAborted=> , Error => , ErrorID =>);

■ Variables

In-out variables

In-Out Variable	Name	Data Type	Value Range	Initial Value	Description
Axis	High-speed axis	HS_AXIS_REF	-	-	Number of the specified high-speed axis to use, ranging from 0 to 3

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
Execute	Enable	BOOL	[FALSE, TRUE]	FALSE	(Triggered by edges) Execution of the function block switch
Position	Homing position	DINT	Depends on data types	-	Homing position of the axis

Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
Done	Completion flag	BOOL	[FALSE, TRUE]	-	Completion flag of the function block
Busy	Execution flag	BOOL	[FALSE, TRUE]	-	Function block execution flag
CommandAborted	Aborted state	BOOL	[FALSE, TRUE]	-	Aborted state of the function block
Error	Error flag	BOOL	[FALSE, TRUE]	-	TRUE: An internal error occurs in the function block
ErrorID	Error ID	UINT	Depends on data types	-	Error code

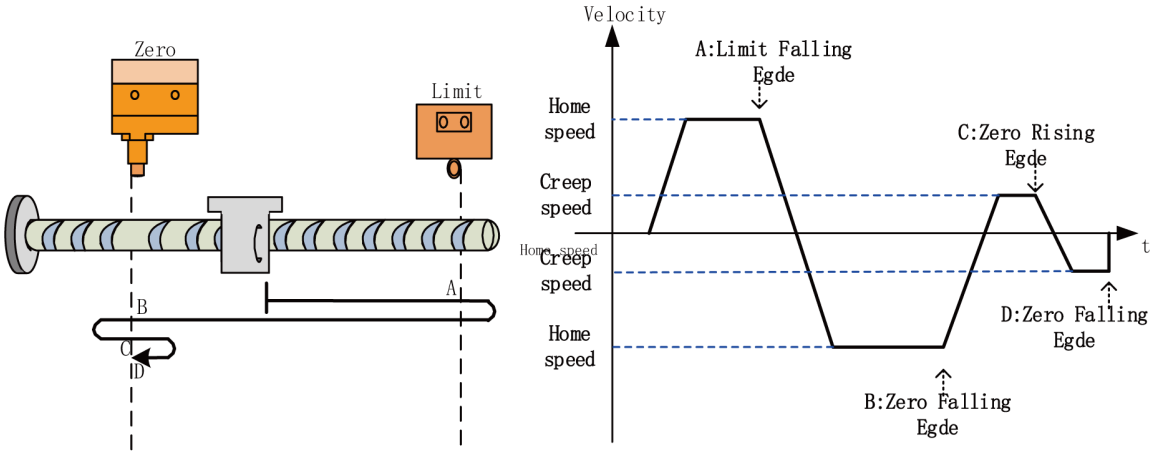
	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Axis	HS_AXIS_REF																			
Execute	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Position	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–
Done	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Busy	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
CommandAborted	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Error	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
ErrorID	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–

■ Function

This function block implements high-speed axis homing. Data specified by Position is the homing position.

mode0

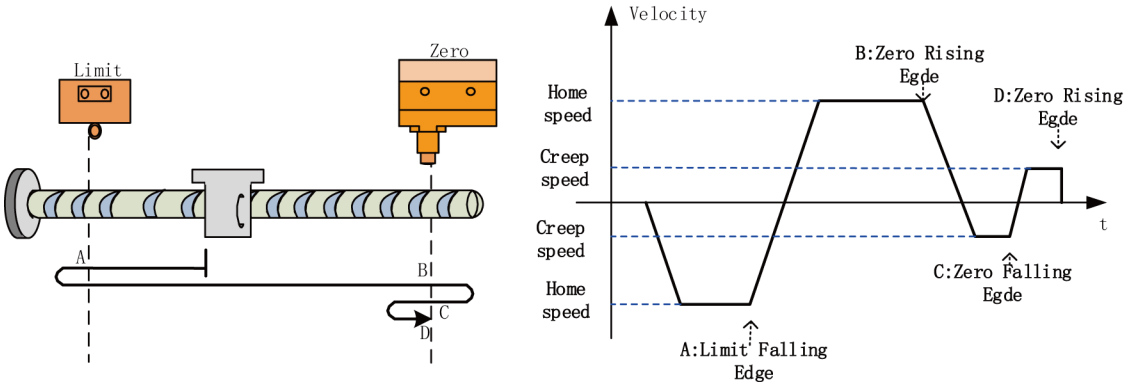
Forward homing is selected, and the home switch acts as the deceleration point and the home. The homing right limit must be set.



Note: If the current position is on the right of the right extreme position, homing fails to start.

mode1

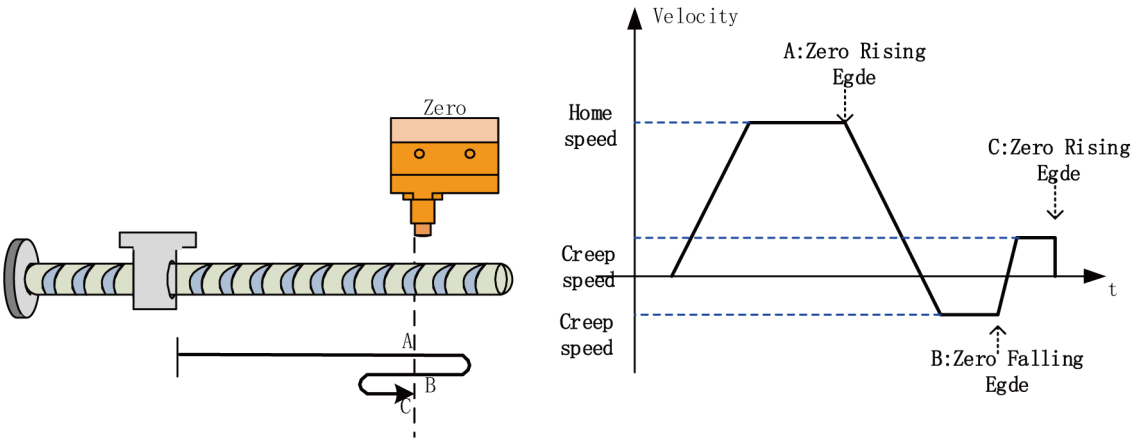
Reverse homing is selected, and the home switch acts as the deceleration point and the home. The homing left limit must be set.



Note: If the current position is on the left of the left extreme position, homing fails to start.

mode2

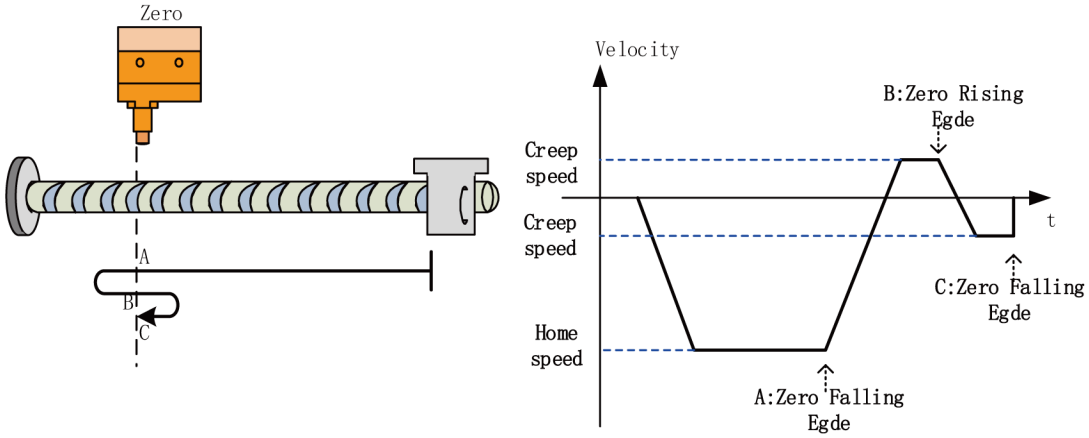
Forward homing is selected, and the home switch acts as the deceleration point and the home.



Note: If the current position is in the middle or on the right of the home position, homing fails to start.

mode3

Reverse homing is selected, and the home switch acts as the deceleration point and the home.

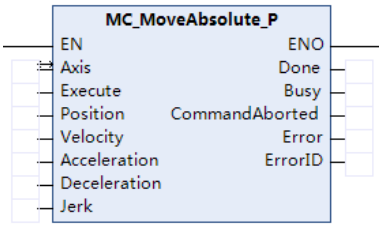


Note: If the current position is in the middle or on the left of the home position, homing fails to start.

5.3. 4 MC_MoveAbsolute_P

This instruction controls the absolute position of the high-speed axis.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
MC_MoveAbsolute_P	Local pulse axis move absolute	FB		MC_MoveAbsolute_P (Axis := , Execute := , Position := , Velocity := , Acceleration:= , Deceleration := , Jerk := , Done => , Busy => , CommandAborted=>, Error => , ErrorID =>);

■ Variables

In-out variables

In-Out Variable	Name	Data Type	Value Range	Initial Value	Description
Axis	High-speed axis	HS_AXIS_REF	-	-	Number of the specified high-speed axis to use, ranging from 0 to 3

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
Execute	Enable	BOOL	[FALSE, TRUE]	FALSE	(Triggered by edges) Execution of the function block switch
Position	Site	DINT	Depends on data types	-	Site
Velocity	Speed	DWORD	Depends on data types	-	Velocity. The unit is pulse/s and the maximum value is 200000.
Acceleration	Acceleration	DWORD	Depends on data types	-	Acceleration rate, which can be changed by dwMaxAcceleration. The unit is pulse/s□ and the maximum value is 20000000.
Deceleration	Deceleration	DWORD	Depends on data types	-	Deceleration rate, which can be changed by dwMaxDeceleration. The unit is pulse/s□ and the maximum value is 20000000.
Jerk	Jerk	DWORD	Depends on data types	-	Jerk (pulse/s□)

Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
Done	Completion flag	BOOL	[FALSE, TRUE]	-	Completion flag of the function block
Busy	Execution flag	BOOL	[FALSE, TRUE]	-	Function block execution flag
CommandAborted	Aborted state	BOOL	[FALSE, TRUE]	-	Aborted state of the function block
Error	Error flag	BOOL	[FALSE, TRUE]	-	TRUE: An internal error occurs in the function block
ErrorID	Error ID	UINT	Depends on data types	-	Error code

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Axis	HS_AXIS_REF																			
Execute	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Position	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–
Velocity	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Acceleration	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Deceleration	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Jerk	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Done	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Busy	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
CommandAborted	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Error	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
ErrorID	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–

■ Function

This function block implements absolute axis positioning of the high-speed axis. Data specified by Position is the absolute position.

Values of Acceleration and Deceleration are integer multiples of 125 x ratio.

$$\text{Rate} = \begin{cases} 1 & \text{Vmax} \le 8000 \\ \frac{\text{Vmax}}{8000} & \text{Vmax} > 8000 \end{cases}$$

$$\text{Vel} = \left[\frac{\text{Vel}}{\text{Rate}}\right] \times \text{Rate}$$

$$\text{Actual acceleration} = \begin{cases} 125 \times \text{Rate} & \text{Acc} \le 125 \times \text{Rate} \\ \left[\frac{\text{Acc}}{125 \times \text{Rate}}\right] \times 125 \times \text{Rate} & \text{Acc} > 125 \times \text{Rate} \end{cases}$$

Values in square brackets ([]) shall be rounded to the nearest integers.

The velocity cannot be changed during acceleration and deceleration.

The positioning distance cannot exceed 2147483647. Otherwise, jogging is in the reverse direction.

When the velocity curve is S-curve, jerk is calculated internally. Therefore, all parameters involving Jerk are calculated internally.

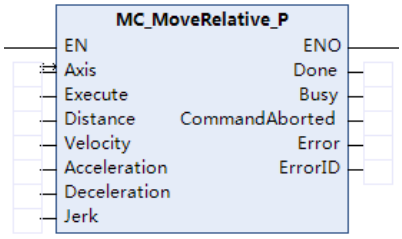
When the velocity curve is S-curve, the velocity, acceleration rate, and deceleration rate cannot be changed during jogging.

When the velocity curve is S-curve and the deceleration rate is greater than or equal to 5 x Velocity, no trailing and truncation occurs.

5.3. 5 MC_MoveRelative_P

This instruction controls the relative position of the high-speed axis.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
MC_MoveRelative_P	Local pulse axis move relative	FB		MC_MoveRelative_P (Axis := , Execute := , Distance := , Velocity := , Acceleration:= , Deceleration := , Jerk := , Done => , Busy => , CommandAborted=>, Error => , ErrorID =>);

■ Variables

In-out variables

In-Out Variable	Name	Data Type	Value Range	Initial Value	Description
Axis	High-speed axis	HS_AXIS_REF	-	-	Number of the specified high-speed axis to use, ranging from 0 to 3

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
Execute	Enable	BOOL	[FALSE, TRUE]	FALSE	(Triggered by edges) Execution of the function block switch
Distance	Distance	DINT	Depends on data types	-	Relative distance
Velocity	Speed	DWORD	Depends on data types	-	Velocity. The unit is pulse/s and the maximum value is 200000.
Acceleration	Acceleration	DWORD	Depends on data types	-	Acceleration rate, which can be changed by dwMaxAcceleration. The unit is pulse/s□ and the maximum value is 20000000.
Deceleration	Deceleration	DWORD	Depends on data types	-	Deceleration rate, which can be changed by dwMaxDeceleration. The unit is pulse/s□ and the maximum value is 20000000.
Jerk	Jerk	DWORD	Depends on data types	-	Jerk (pulse/s□)

Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
Done	Completion flag	BOOL	[FALSE, TRUE]	-	Completion flag of the function block
Busy	Execution flag	BOOL	[FALSE, TRUE]	-	Function block execution flag
CommandAborted	Aborted state	BOOL	[FALSE, TRUE]	-	Aborted state of the function block
Error	Error flag	BOOL	[FALSE, TRUE]	-	TRUE: An internal error occurs in the function block
ErrorID	Error ID	UINT	Depends on data types	-	Error code

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Axis	HS_AXIS_REF																			
Execute	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Distance	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–
Velocity	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Acceleration	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Deceleration	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Jerk	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Done	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Busy	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
CommandAborted	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Error	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
ErrorID	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–

■ Function

This function block implements absolute axis positioning of the high-speed axis. Data specified by Position is the absolute position.

Values of Acceleration and Deceleration are integer multiples of 125 x ratio.

$$\text{Rate} = \begin{cases} 1 & \text{Vmax} \le 8000 \\ \frac{\text{Vmax}}{8000} & \text{Vmax} > 8000 \end{cases}$$

$$\text{Vel} = \left[\frac{\text{Vel}}{\text{Rate}}\right] \times \text{Rate}$$

$$\text{Actual acceleration} = \begin{cases} 125 \times \text{Rate} & \text{Acc} \le 125 \times \text{Rate} \\ \left[\frac{\text{Acc}}{125 \times \text{Rate}}\right] \times 125 \times \text{Rate} & \text{Acc} > 125 \times \text{Rate} \end{cases}$$

Values in square brackets ([]) shall be rounded to the nearest integers.

The velocity cannot be changed during acceleration and deceleration.

The positioning distance cannot exceed 2147483647. Otherwise, jogging is in the reverse direction.

When the velocity curve is S-curve, jerk is calculated internally. Therefore, all parameters involving Jerk are calculated internally.

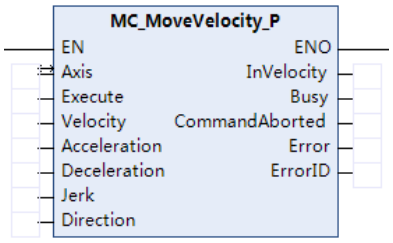
When the velocity curve is S-curve, the velocity, acceleration rate, and deceleration rate cannot be changed during jogging.

When the velocity curve is S-curve and the deceleration rate is greater than or equal to 5 x Velocity, no trailing and truncation occurs.

5.3. 6 MC_MoveVelocity_P

This instruction controls the high-speed axis velocity.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
MC_MoveVelocity_P	Local pulse axis move velocity	FB		MC_MoveVelocity_P (Axis := , Execute := , Velocity := , Acceleration:= , Deceleration := , Jerk := , Direction := , InVelocity => , Busy => , CommandAborted=>, Error => , ErrorID =>);

■ Variables

In-out variables

In-Out Variable	Name	Data Type	Value Range	Initial Value	Description
Axis	High-speed axis	HS_AXIS_REF	-	-	Number of the specified high-speed axis to use, ranging from 0 to 3

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
Execute	Enable	BOOL	[FALSE, TRUE]	FALSE	(Triggered by edges) Execution of the function block switch
Velocity	Speed	DWORD	Depends on data types	-	Velocity. The unit is pulse/s and the maximum value is 200000.
Acceleration	Acceleration	DWORD	Depends on data types	-	Acceleration rate, which can be changed by dwMaxAcceleration. The unit is pulse/s□ and the maximum value is 20000000.
Deceleration	Deceleration	DWORD	Depends on data types	-	Deceleration rate, which can be changed by dwMaxDeceleration. The unit is pulse/s□ and the maximum value is 20000000.
Jerk	Jerk	DWORD	Depends on data types	-	Jerk (pulse/s□)
Direction	Direction	BOOL	[FALSE, TRUE]	FALSE	Direction

Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
InVelocity	Speed reference reached flag	BOOL	[FALSE, TRUE]	-	Speed reference reached flag
Busy	Execution flag	BOOL	[FALSE, TRUE]	-	Function block execution flag
CommandAborted	Aborted state	BOOL	[FALSE, TRUE]	-	Aborted state of the function block
Error	Error flag	BOOL	[FALSE, TRUE]	-	TRUE: An internal error occurs in the function block
ErrorID	Error ID	UINT	Depends on data types	-	Error code

	Bool-ean	Bit String				Integer								Real Num-ber		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Axis	HS_AXIS_REF																			
Execute	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Velocity	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Acceleration	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Deceleration	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Jerk	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Direction	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
InVelocity	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Busy	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
CommandAborted	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Error	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
ErrorID	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–

■ Function

This function block controls the high-speed axis velocity.

Values of Acceleration and Deceleration are integer multiples of 125 x ratio.

$$\text{Rate} = \begin{cases} 1 & \text{Vmax} \le 8000 \\ \frac{\text{Vmax}}{8000} & \text{Vmax} > 8000 \end{cases}$$

$$\text{Vel} = \left[\frac{\text{Vel}}{\text{Rate}}\right] \times \text{Rate}$$

$$\text{Actual acceleration} = \begin{cases} 125 \times \text{Rate} & \text{Acc} \le 125 \times \text{Rate} \\ \left[\frac{\text{Acc}}{125 \times \text{Rate}}\right] \times 125 \times \text{Rate} & \text{Acc} > 125 \times \text{Rate} \end{cases}$$

Values in square brackets ([]) shall be rounded to the nearest integers.

The velocity cannot be changed during acceleration and deceleration.

The positioning distance cannot exceed 2147483647. Otherwise, jogging is in the reverse direction.

When the velocity curve is S-curve, jerk is calculated internally. Therefore, all parameters involving Jerk are calculated internally.

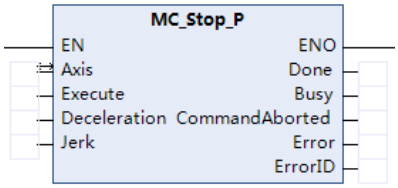
When the velocity curve is S-curve, the velocity, acceleration rate, and deceleration rate cannot be changed during jogging.

When the velocity curve is S-curve and the deceleration rate is greater than or equal to 5 x Velocity, no trailing and truncation occurs.

5.3. 7 MC_Stop_P

This instruction stops the high-speed axis.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
MC_Stop_P	Stop pulse axis motion	FB		MC_Stop_P (Axis := , Execute:= , Deceleration := , Jerk := , Done => , Busy => , CommandAborted=> , Error => , ErrorID =>);

■ Variables

In-out variables

In-Out Variable	Name	Data Type	Value Range	Initial Value	Description
Axis	High-speed axis	HS_AXIS_REF	-	-	Number of the specified high-speed axis to use, ranging from 0 to 3

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
Execute	Enable	BOOL	[FALSE, TRUE]	FALSE	(Triggered by edges) Execution of the function block switch
Deceleration	Deceleration	DWORD	Depends on data types	-	Deceleration of the axis
Jerk	Jerk	DWORD	Depends on data types	-	Jerk of the axis

Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
Done	Completion flag	BOOL	[FALSE, TRUE]	-	Completion flag of the function block

Output Variable	Name	Data Type	Value Range	Initial Value	Description
Busy	Execution flag	BOOL	[FALSE, TRUE]	-	Function block execution flag
CommandAborted	Aborted state	BOOL	[FALSE, TRUE]	-	Aborted state of the function block
Error	Error flag	BOOL	[FALSE, TRUE]	-	TRUE: An internal error occurs in the function block
ErrorID	Error ID	UINT	Depends on data types	-	Error code

	Bool-ean	Bit String				Integer								Real Num-ber		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Axis	HS_AXIS_REF																			
Execute	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Deceleration	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Jerk	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Done	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Busy	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
CommandAborted	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Error	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
ErrorID	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–

■ Function

This function block stops the high-speed axis.

Values of Acceleration and Deceleration are integer multiples of 125 x ratio.

$$\text{Rate} = \begin{cases} 1 & \text{Vmax} \le 8000 \\ \frac{\text{Vmax}}{8000} & \text{Vmax} > 8000 \end{cases}$$

$$\text{Vel} = \left[\frac{\text{Vel}}{\text{Rate}}\right] \times \text{Rate}$$

$$\text{Actual acceleration} = \begin{cases} 125 \times \text{Rate} & \text{Acc} \le 125 \times \text{Rate} \\ \left[\frac{\text{Acc}}{125 \times \text{Rate}}\right] \times 125 \times \text{Rate} & \text{Acc} > 125 \times \text{Rate} \end{cases}$$

Values in square brackets ([]) shall be rounded to the nearest integers.

The velocity cannot be changed during acceleration and deceleration.

The positioning distance cannot exceed 2147483647. Otherwise, jogging is in the reverse direction.

When the velocity curve is S-curve, jerk is calculated internally. Therefore, all parameters involving Jerk are calculated internally.

When the velocity curve is S-curve, the velocity, acceleration rate, and deceleration rate cannot be changed during jogging.

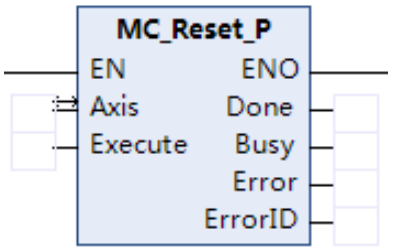
When the velocity curve is S-curve and the deceleration rate is greater than or equal to 5 x Velocity, no

trailing and truncation occurs.

5.3. 8 MC_Reset_P

This instruction resets high-speed axis errors.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
MC_Reset_P	Reset pulse axis errors	FB		MC_Reset_P (Axis := , Execute:= , Done => , Busy => , Error => , ErrorID =>);

■ Variables

In-out variables

In-Out Variable	Name	Data Type	Value Range	Initial Value	Description
Axis	High-speed axis	HS_AXIS_REF	-	-	Number of the specified high-speed axis to use, ranging from 0 to 3

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
Execute	Enable	BOOL	[FALSE, TRUE]	FALSE	(Triggered by edges) Execution of the function block switch

Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
Done	Comple-tion flag	BOOL	[FALSE, TRUE]	-	Completion flag of the func-tion block
Busy	Execution flag	BOOL	[FALSE, TRUE]	-	Function block execution flag
Error	Error flag	BOOL	[FALSE, TRUE]	-	TRUE: An internal error occurs in the function block
ErrorID	Error ID	UINT	Depends on data types	-	Error code

	Bool-ean	Bit String				Integer								Real Num-ber		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Axis	HS_AXIS_REF																			
Execute	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

	Bool-ean	Bit String				Integer								Real Num-ber		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Done	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Busy	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Error	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
ErrorID	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–

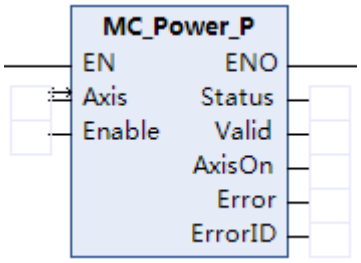
■ Function

This function block resets high-speed axis errors.

5.3. 9 MC_Power_P

This instruction enables the high-speed axis.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
MC_Power_P	Enable local pulse axis	FB		MC_Power_P (Axis := , Enable := , Status => , Valid => , AxisOn => , Error => , ErrorID =>);

■ Variables

In-out variables

In-Out Variable	Name	Data Type	Value Range	Initial Value	Description
Axis	High-speed axis	HS_AXIS_REF	-	-	Number of the specified high-speed axis to use, ranging from 0 to 3

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
Enable	Enable	BOOL	[FALSE, TRUE]	FALSE	Function block enable bit, triggered by level

Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
Status	Ready-to-run state	BOOL	[FALSE, TRUE]	-	If the axis is ready to run, the value is TRUE.
Valid	Enable flag	BOOL	[FALSE, TRUE]	-	Execution valid flag of the function block

Output Variable	Name	Data Type	Value Range	Initial Value	Description
AxisOn	Axis enable	BOOL	[FALSE, TRUE]	-	Axis enable
Error	Error flag	BOOL	[FALSE, TRUE]	-	TRUE: An internal error occurs in the function block

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Axis	HS_AXIS_REF																			
Enable	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Status	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Valid	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
AxisOn	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Error	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
ErrorID	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–

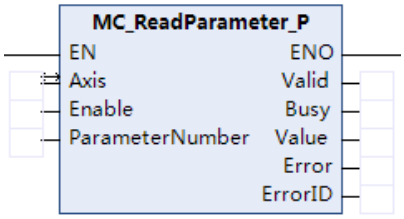
■ Function

This function block enables the high-speed axis.

5.3. 10 MC_ReadParameter_P

This instruction reads high-speed axis parameters.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
MC_ReadParameter_P	Read pulse axis parameters	FB		MC_ReadParameter_P(Axis := , Enable := , ParameterNumber:= , Valid => , Busy => , Value => , Error => , ErrorID =>);

■ Variables

In-out variables

In-Out Variable	Name	Data Type	Value Range	Initial Value	Description
Axis	High-speed axis	HS_AXIS_REF	-	-	Number of the specified high-speed axis to use, ranging from 0 to 3

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
Enable	Enable	BOOL	[FALSE, TRUE]	FALSE	Function block enable bit, triggered by level
ParameterNumber	Parameter Number	DINT	Depends on data types	-	Parameter Number

Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
Valid	Active state	BOOL	[FALSE, TRUE]	-	Execution valid flag of the function block
Busy	Execution flag	BOOL	[FALSE, TRUE]	-	Function block execution flag
Value	Return value	DINT	Depends on data types	-	Return value
Error	Error flag	BOOL	[FALSE, TRUE]	-	TRUE: An internal error occurs in the function block
ErrorID	Error ID	UINT	Depends on data types	-	Error code

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Axis	HS_AXIS_REF																			
Enable	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
ParameterNumber	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–
Valid	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Busy	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Value	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–
Error	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
ErrorID	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–

■ Function

This function block reads high-speed axis parameters. The following table lists the PN parameter mappings.

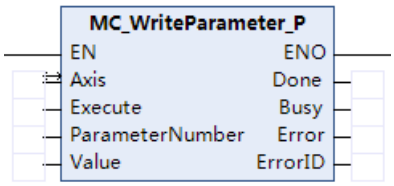
PN	Name	DataByte	R/W	Comments
10030	diSWLimitPos	Dint	R/W	Soft limit maximum value
10031	diSWLimitNeg	Dint	R/W	Soft limit minimum value
10032	bEnableLimitPos	Bool	R/W	Soft limit minimum value enable 0: FALSE
10033	bEnableLimitNeg	Bool	R/W	Soft limit maximum value enable 0: FALSE
10034	Velocity	Dint	R	Actual velocity
10035	Position	Dint	R	Actual position

PN	Name	DataByte	R/W	Comments
10036	dwMaxAcceleration	Dword	R/W	Maximum acceleration
10037	dwMinAcceleration	Dword	R/W	Minimum acceleration rate
10038	dwMaxDeceleration	Dword	R/W	Max. deceleration
10039	dwMinDeceleration	Dword	R/W	Minimum deceleration rate
10040	dwMaxVelocity	Dint	R/W	Maximum speed
10041	dwHomeVelocity	Dword	R/W	Homing velocity
10042	dwHomeCreepVelocity	Dword	R/W	Homing creeping velocity
10043	dwHomeAcceleration	Dword	R/W	Homing acceleration rate
10044	dwHomeDeceleration	Dword	R/W	Homing deceleration rate
10045	bHomeMode	Byte	R/W	Homing mode The value range is [0, 3].
10046	bErrorStopMode	Bool	R/W	Stop mode upon an axis error 0: FALSE
10046	bSoftLimitStopMode	Bool	R/W	Stop mode upon software limit 0: FALSE

5.3. 11 MC_WriteParameter_P

This instruction writes high-speed axis parameters.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
MC_WriteParameter_P	Write pulse axis parameters	FB		MC_WriteParameter_P(Axis := , Execute := , ParameterNumber:= , Value := , Done => , Busy => , Error => , ErrorID =>);

■ Variables

In-out variables

In-Out Variable	Name	Data Type	Value Range	Initial Value	Description
Axis	High-speed axis	HS_AXIS_REF	-	-	Number of the specified high-speed axis to use, ranging from 0 to 3

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
Execute	Enable	BOOL	[FALSE, TRUE]	FALSE	(Triggered by edges) Execution of the function block switch

Input Variable	Name	Data Type	Value Range	Initial Value	Description
ParameterNumber	Parameter Number	DINT	Depends on data types	-	Parameter Number
Value	Value	DINT	Depends on data types	-	Value

Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
Done	Completion flag	BOOL	[FALSE, TRUE]	-	Completion flag of the function block
Busy	Execution flag	BOOL	[FALSE, TRUE]	-	Function block execution flag
Error	Error flag	BOOL	[FALSE, TRUE]	-	TRUE: An internal error occurs in the function block
ErrorID	Error ID	UINT	Depends on data types	-	Error code

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Axis	HS_AXIS_REF																			
Execute	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
ParameterNumber	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–
Value	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–
Done	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Busy	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Error	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
ErrorID	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–

■ Function

This function block writes high-speed axis parameters. The following table lists the PN parameter mappings.

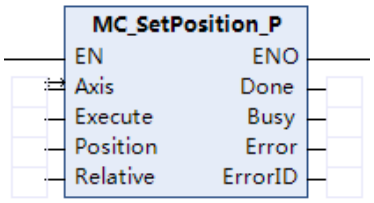
PN	Name	DataByte	R/W	Comments
10030	diSWLimitPos	Dint	R/W	Soft limit maximum value
10031	diSWLimitNeg	Dint	R/W	Soft limit minimum value
10032	bEnableLimitPos	Bool	R/W	Soft limit minimum value enable 0: FALSE
10033	bEnableLimitNeg	Bool	R/W	Soft limit maximum value enable 0: FALSE
10034	Velocity	Dint	R	Actual velocity
10035	Position	Dint	R	Actual position
10036	dwMaxAcceleration	Dword	R/W	Maximum acceleration
10037	dwMinAcceleration	Dword	R/W	Minimum acceleration rate

PN	Name	DataByte	R/W	Comments
10038	dwMaxDeceleration	Dword	R/W	Max. deceleration
10039	dwMinDeceleration	Dword	R/W	Minimum deceleration rate
10040	dwMaxVelocity	Dint	R/W	Maximum speed
10041	dwHomeVelocity	Dword	R/W	Homing velocity
10042	dwHomeCreepVelocity	Dword	R/W	Homing creeping velocity
10043	dwHomeAcceleration	Dword	R/W	Homing acceleration rate
10044	dwHomeDeceleration	Dword	R/W	Homing deceleration rate
10045	bHomeMode	Byte	R/W	Homing mode The value range is [0, 3].
10046	bErrorStopMode	Bool	R/W	Stop mode upon an axis error 0: FALSE
10046	bSoftLimitStopMode	Bool	R/W	Stop mode upon software limit 0: FALSE

5.3. 12 MC_SetPosition_P

This instruction set the logical address for the high-speed axis.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
MC_SetPosition_P	Set pulse axis position	FB		MC_SetPosition_P(Axis := , Execute := , Position := , Relative := , Done => , Busy => , Error => , ErrorID =>);

■ Variables

In-out variables

In-Out Variable	Name	Data Type	Value Range	Initial Value	Description
Axis	High-speed axis	HS_AXIS_REF	-	-	Number of the specified high-speed axis to use, ranging from 0 to 3

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
Execute	Enable	BOOL	[FALSE, TRUE]	FALSE	(Triggered by edges) Execution of the function block switch
Position	Address	DINT	Depends on data types	-	Address

Input Variable	Name	Data Type	Value Range	Initial Value	Description
Relative	Relative flag	BOOL	[FALSE, TRUE]	FALSE	0: Absolute position of the logical address 1: Relative position of the logical address

Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
Done	Completion flag	BOOL	[FALSE, TRUE]	-	Completion flag of the function block
Busy	Execution flag	BOOL	[FALSE, TRUE]	-	Function block execution flag
Error	Error flag	BOOL	[FALSE, TRUE]	-	TRUE: An internal error occurs in the function block
ErrorID	Error ID	UINT	Depends on data types	-	Error code

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Axis	HS_AXIS_REF																			
Execute	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Position	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–
Relative	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Done	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Busy	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Error	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
ErrorID	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–

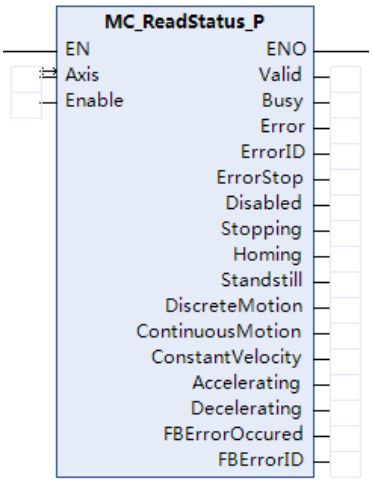
■ Function

This function block sets the logical address for the high-speed axis.

5.3. 13 MC_ReadStatus_P

This instruction read the high-speed axis status.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
MC_ReadStatus_P	Read pulse axis status	FB		MC_ReadStatus_P(Axis := , Enable := , Valid => , Busy => , Error => , ErrorID => , ErrorStop => , Disabled => , Stopping => , Homing => , Standstill => , DiscreteMotion=> , ContinuousMotion=> , ConstantVelocity=> , Accelerating => , Decelerating => , FBErrorOccured => , FBErrorID => ,);

■ Variables

In-out variables

In-Out Variable	Name	Data Type	Value Range	Initial Value	Description
Axis	High-speed axis	HS_AXIS_REF	-	-	Number of the specified high-speed axis to use, ranging from 0 to 3

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
Enable	Enable	BOOL	[FALSE, TRUE]	FALSE	Function block enable bit, triggered by level

Output variables

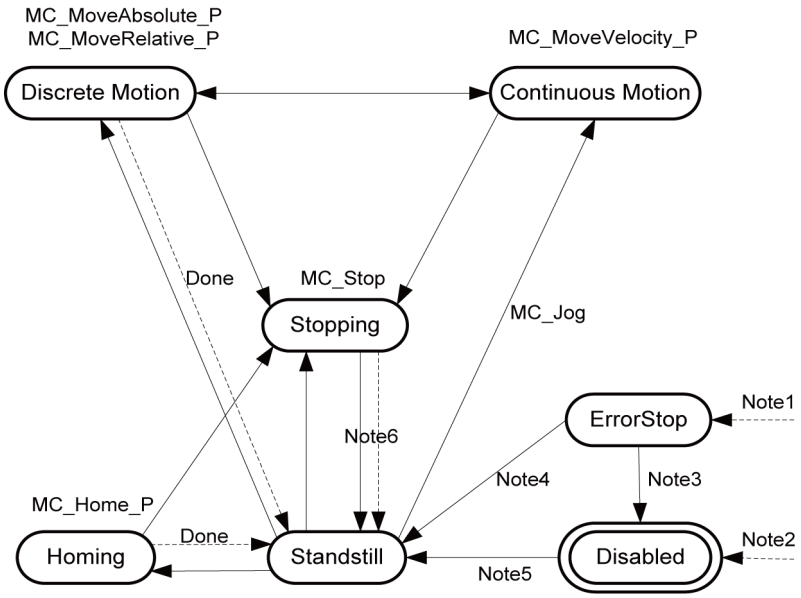
Output Variable	Name	Data Type	Value Range	Initial Value	Description
Valid	Active state	BOOL	[FALSE, TRUE]	-	Execution valid flag of the function block
Busy	Execution flag	BOOL	[FALSE, TRUE]	-	Function block execution flag
Error	Error flag	BOOL	[FALSE, TRUE]	-	TRUE: An internal error occurs in the function block
ErrorID	Error ID	UINT	Depends on data types	-	Error code
ErrorStop	Error stop mode	BOOL	[FALSE, TRUE]	-	Error stop mode
Disabled	Disabled mode	BOOL	[FALSE, TRUE]	-	Disabled mode

Output Variable	Name	Data Type	Value Range	Initial Value	Description
Stopping	Stopping mode	BOOL	[FALSE, TRUE]	-	Stopping mode
Homing	Homing mode	BOOL	[FALSE, TRUE]	-	Homing mode
StandStill	Standstill mode	BOOL	[FALSE, TRUE]	-	Standstill mode
DiscreteMotion	Positioning mode	BOOL	[FALSE, TRUE]	-	Positioning mode
ContinuousMotion	Speed mode	BOOL	[FALSE, TRUE]	-	Speed mode
ConstantVelocity	Constant velocity phase	BOOL	[FALSE, TRUE]	-	Constant velocity phase
Accelerating	Acceleration phase	BOOL	[FALSE, TRUE]	-	Acceleration phase
Decelerating	Deceleration phase	BOOL	[FALSE, TRUE]	-	Deceleration phase
FBErrorOccured	Function block error	BOOL	[FALSE, TRUE]	-	Function block error
FBErrorID	Error ID of function block	DWORD	Depends on data types	-	Error ID of function block

	Bool-ean	Bit String				Integer								Real Num-ber		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Axis	HS_AXIS_REF																			
Enable	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Valid	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Busy	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Error	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
ErrorID	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–
ErrorStop	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Disabled	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Stopping	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Homing	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
StandStill	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
DiscreteMotion	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
ContinuousMotion	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
ConstantVelocity	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Accelerating	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Decelerating	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
FBErrorOccured	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
FBErrorID	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

■ Function

This function block reads the high-speed axis status. The following figure shows the axis state machine.



Note 1: From any state. An error in the axis occurred. Reference 2.3

Note 2: From any state. MC_Power.Enable = FALSE and there is no error in the axis.

Note 3: MC_Reset AND MC_Power.Status = FALSE

Note 4: MC_Reset AND MC_Power.Status = TRUE AND MC_Power.Enable = TRUE

Note 5: MC_Power.Enable = TRUE AND MC_Power.Status = TRUE

Note 6: MC_Stop.Done = TRUE AND MC_Stop.Execute = FALSE

Note 7: MC_Jog only join Continuous Motion.

The basic format of ErrorID is as follows:

Library + Function block + Error code

Library: High-speed I/O. Default: 0

Function block number: Starting from 01

Error code: Starting from 01. For details, see the following table. When the error code is less than 500, the error is a critical error. When the error code is greater than 500, the error is a function block error.

For example, in 31520, 31 indicates MC_WriteParameter_P and 520 indicates the parameter setting error.

Error code	Error	Description
001	ERR_NOT_POWER	MC_Power is not enabled.
002	ERR_UP_SOFTWARE_LIMIT	The current position exceeds the software trip limit +.
003	ERR_DOWN_SOFTWARE_LIMIT	The current position exceeds the software trip limit -.
004	ERR_AXIS_FUNC_UNUSED	The high-speed axis function is disabled, which is enabled in the background.
005	ERR_INPUT_CHANNAL_NUM_INVALID	The axis number is invalid. The valid range is [0, 3].

Error code	Error	Description
006	ERR_DEST_POS_OVER_SOFT_UP_LIMIT	The target position exceeds the software upper limit.
007	ERR_DEST_POS_OVER_SOFT_DOWN_LIMIT	The target position exceeds the software lower limit.
010	ERR_POS_DECPOINT_OVERLOW	The deceleration point is invalid: In position mode, the deceleration length is greater than the actual distance during re-positioning.
011	ERR_VEL_DECPOINT_OVERLOW	The deceleration point is invalid: In velocity mode, the deceleration length is greater than the actual distance during switching to positioning.
012	ERR_POS_PLSNUM_OVERLOW	The length exceeds the PLSNUM maximum positioning length 2147483647.
013	ERR_POS_DECPOINT2_OVERLOW	The deceleration point fails to be calculated for twice.
501	ERR_ACC_SET_OVERFLOW	The acceleration rate exceeds the allowed range and the maximum acceleration rate specified by MC_WriteParameter_P.
502	ERR_ACC_SET_LOW	The acceleration rate is too low and is less than the minimum acceleration rate specified by MC_WriteParameter_P.
503	ERR_DEC_SET_OVERFLOW	The deceleration rate exceeds the allowed range and the maximum deceleration rate set by MC_WriteParameter_P.
504	ERR_DEC_SET_LOW	The deceleration rate is too low and is less than the minimum deceleration rate set by MC_WriteParameter_P.
505	ERR_VEL_SET_OVERFLOW	The velocity exceeds the allowed range set in the background or by MC_WriteParameter_P.
506	ERR_VEL_SET_LOW	The velocity is too low.
508	ERR_VEL_LESS_THAN_STARTVEL	The velocity is lower than the startup offset velocity set in the background.
509	ERR_STARTVEL_SET_LOW	The starting velocity is too low.
510	ERR_FBD_MOVEMODE_INVALIAD	The motion mode of the function block is invalid, and the status is incorrect.
511	ERR_WASNT_STANDSTILL	The current state is not STANDSTILL.
512	ERR_WASNT_DISABLED	The current state is not DISABLE.
513	ERR_IN_ERRORSTOP	The current state is not ERRORSTOP.
514	ERR_NOT_READY_FOR_MOTION	The axis is not ready to run.
515	ERR_INVLALID_VELOCITY_MODE	The velocity mode is invalid.
516	ERR_INVLALID_POSTION_MODE	The position mode is invalid.

Error code	Error	Description
520	ERR_AXIS_WRITEPARAMETER_UNVALIAD	The MC_WriteParameter_P instruction is invalid.
521	ERR_AXIS_READPARAMETER_UNVALIAD	The MC_ReadParameter_P instruction is invalid.
522	ERR_HOME_MODE_UNVALIAD	The homing mode is invalid, which is selected in the background.
523	ERR_AXIS_WRITEPARAMETER_HOME_MODE_UNVALIAD	The homing mode is invalid.

The errors are divided into axis errors and function block errors.

The conditions for setting the ErrorStop state are as follows:

An axis error occurs, or a function block error occurs in the DiscreteMotion, ContinuousMotion, or Homing state.

The following table lists the velocity change timing.

Mode	Ladder Acceleration/Deceleration Mode			S-Shape Acceleration/Deceleration Mode		
	Acceleration	Constant Velocity	Deceleration	Acceleration	Constant Velocity	Deceleration
Speed mode	-	The velocity can be changed.	-	-	-	-
Position mode	-	The velocity can be changed.	-	-	-	-
Velocity -> Position	-	The velocity can be changed.	-	-	-	-
Position -> Velocity	-	The velocity can be changed.	-	-	-	-
Homing mode	-	-	-	-	-	-
Jog	-	-	-	-	-	-

Note: The symbols "---"indicate not supported.

5. 4 EtherCAT Remote Counter Instructions

5.4. 1 Instruction List

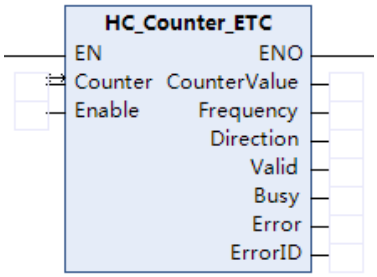
Instruction Category	Name	FB/FC	Function
GR10-2HCE remote counter instructions	HC_Counter_ETC	FB	Enable 2HCE counter
	HC_SetCompare_ETC	FB	2HCE counter comparison output
	HC_Presetvalue_ETC	FB	Preset 2HCE counter value
	HC_TouchProbe_ETC	FB	2HCE counter probe
	HC_Reset_ETC	FB	Reset 2HCE counter

Instruction Category	Name	FB/FC	Function
GR10-8PBE remote counter instructions	Counter_ETC	FB	Enable 8PBE counter
	Counter_SetCompare_ETC	FB	8PBE counter comparison output
	Counter_Presetvalue_ETC	FB	Preset 8PBE counter value
	Counter_TouchProbe_ETC	FB	8PBE counter probe
	Counter_Reset_ETC	FB	Reset 8PBE counter

5.4. 2 HC_Counter_ETC

This instruction enables the 2HCE counter.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
HC_Counter_ETC	Enable 2HCE counter	FB		HC_Counter_ETC(Counter := , Enable := , CounterValue=> , Frequency => , Direction => , Valid => , Busy => , Error=> , ErrorID=>);

■ Variables

In-out variables

In-Out Variable	Name	Data Type	Value Range	Initial Value	Description
Counter	2HCE counter	AXIS_REF_ENCODER_ETC	-	-	AXIS_REF_ENCODER_ETC counter type

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
Enable	Enable	BOOL	[FALSE, TRUE]	FALSE	Triggered by level TRUE: Enable the counter

Output variables

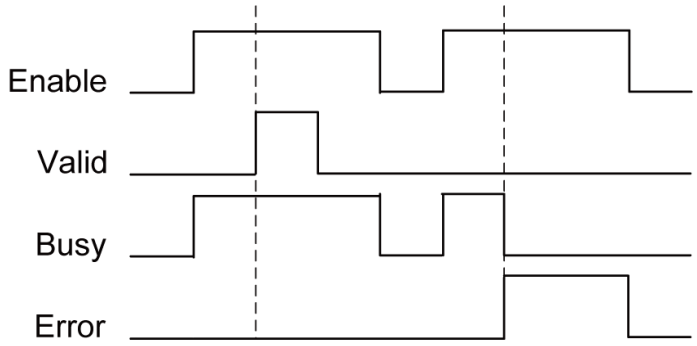
Output Variable	Name	Data Type	Value Range	Initial Value	Description
CounterValue	Current count value	LREAL	(− 1.7E 308, 1.7E 308)	0.0	Current count value, which is converted from pulse count
Frequency	Frequency	UDINT	[0, 2^32)	0	Frequency of the collected pulse
Direction	Direction	COUNTER_ DIR	[− 1, 1]	− 1	Count direction
Valid	Enabled	BOOL	[FALSE, TRUE]	FALSE	Count enabled state
Busy	Execution flag	BOOL	[FALSE, TRUE]	FALSE	Execution of the count enable instruction
Error	Error flag	BOOL	[FALSE, TRUE]	FALSE	TRUE: An internal error occurs in the function block
ErrorID	Error ID	COUNTER_ ERROR	-	COUNTER_ NO_ERROR	Error ID For details, see COUNTER_ ERROR.

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Counter	AXIS_REF_ENCODER_ETC																			
Enable	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
CounterValue	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–
Frequency	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–
Direction	COUNTER_DIR																			
Valid	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Busy	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Error	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
ErrorID	COUNTER_ERROR																			
Counter	AXIS_REF_ENCODER_ETC																			
Enable	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
CounterValue	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–
Frequency	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–
Direction	COUNTER_DIR																			
Valid	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Busy	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Error	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
ErrorID	COUNTER_ERROR																			

■ Function

This function block implements the enable, count, frequency measurement, and count direction output functions of the 2HCE counter.

■ Timing diagram



■ Error description

When an error occurs, locate the error specified by ErrorID in COUNTER_ERROR according to the help document, and find the reason for the error.

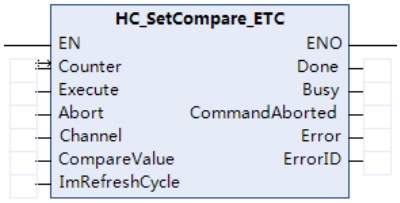
Error ID	Enumerator	Description
0x00	COUNTER_NO_ERROR	No error occurs.
0x01	COUNTER_ERROR_ECT_COMMUNICATION	ETC communication fault
0x02	COUNTER_ERROR_PV_PRESETVALUE_EXCEED	Preset value exceeding the limit
0x03	COUNTER_ERROR_CP_COMPARE_VAULE_INVALID	Invalid setting for the comparison value
0x04	COUNTER_ERROR_FILT_PARAM_EXCEED	Filter coefficient exceeding the valid range
0x10	COUNTER_ERROR_PV_DI_CONFIGURE_ERROR	Preset function not configured for the DI terminal
0x11	-	Counter reset to zero, which is not used
0x12	COUNTER_ERROR_TP_DI_CONFIGURE_ERROR	Probe function not configured for the DI terminal
0x30	COUNTER_ERROR_CP_DO_CONFIGURE_ERROR	Comparison consistence output function not configured for the DO output terminal
0x50	COUNTER_ERROR_INPUT_PUSLE_FREQUENCY_TOO_HIGH	Input frequency greater than 202 kHz
0x51	COUNTER_ERROR_OVERFLOW	Count value overflow
0x52	COUNTER_ERROR_UNDERFLOW	Count value underflow
0x1001	COUNTER_ERROR_PV_TYPE_INVALID	Preset type exceeding the valid range
0x1002	COUNTER_ERROR_DISABLE	Counter disabled
0x1003	COUNTER_ERROR_TP_PARAM_INVALID	Invalid probe parameters
0x1004	COUNTER_ERROR_CP_PARAM_INVALID	Incorrect comparison channel parameters

Error ID	Enumerator	Description
0x1005	COUNTER_ERROR_R_NO_ERROR_CLEAR	No error cleared
0x1006	COUNTER_ERROR_R_ERROR_CANNOT_RESET	Errors cannot be cleared
0x1100	COUNTER_ERROR_PARAM_NO_PDO_MAPPING	No PDO configured for the instruction

5.4. 3 HC_SetCompare_ETC

This instruction implements the comparison consistence output for the 2HCE counter.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
HC_SetCompare_ETC	2HCE counter comparison output	FB		HC_SetCompare_ETC(Counter := , Execute:= , Abort:= , Channel:= , CompareValue:= , ImRefreshCycle:= , Done => , Busy => , CommandAborted=> , Error=> , ErrorID=>);

■ Variables

In-out variables

In-Out Variable	Name	Data Type	Value Range	Initial Value	Description
Counter	2HCE counter	AXIS_REF_ENCODER_ETC	-	-	AXIS_REF_ENCODER_ETC counter type

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
Execute	Enable	BOOL	[FALSE, TRUE]	FALSE	(Triggered by edges) Execution of the function block switch
Abort	Abort	BOOL	[FALSE, TRUE]	FALSE	Triggered by level TRUE: Function block is aborted
Channel	Comparison channel	BYTE	[1, 2]	1	Comparison channel of the counter Two channels are available.
CompareValue	Comparison value	LREAL	(– 1.7E 308, 1.7E 308)	0.0	Comparison value of the float type, in the unit of unit

Input Variable	Name	Data Type	Value Range	Initial Value	Description
ImRefreshCycle	Output holding time	UINT	(0, 2^32)	1000	The holding time of the output port is enabled. The minimum unit of the time is 100 us. Default value: 1000 (1000 x 100 us = 100 ms)

Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
Done	Comparison consistence completed state	BOOL	[FALSE, TRUE]	FALSE	The pulse count reaches the set comparison value, and the output holding time ends.
CommandAborted	Aborted state	BOOL	[FALSE, TRUE]	FALSE	Aborted state, which is set when Abort is TRUE
Busy	Execution flag	BOOL	[FALSE, TRUE]	FALSE	Execution of the comparison consistence function
Error	Error flag	BOOL	[FALSE, TRUE]	FALSE	TRUE: An internal error occurs in the function block
ErrorID	Error ID	COUNTER_ERROR	-	COUNTER_NO_ERROR	Error ID For details, see COUNTER_ERROR.

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Counter	AXIS_REF_ENCODER_ETC																			
Execute	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Abort	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Channel	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
CompareValue	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–
ImRefreshCycle	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–
Done	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
CommandAborted	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Busy	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Error	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
ErrorID	COUNTER_ERROR																			

■ Function

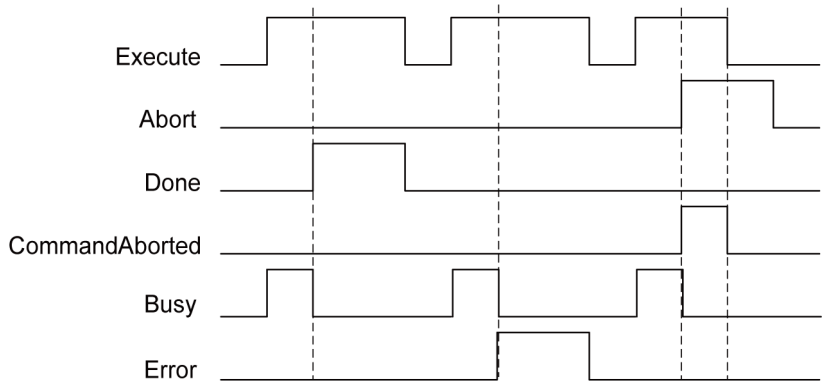
This function block triggers a hardware high-speed output port and retains it for a certain period when the current count value of the counter is equal to the set comparison value.

The comparison consistence output function is a single trigger. When Done is TRUE, the comparison consistency function is completed. To continue using the comparison function, trigger Execute again.

Confirm the output point attribute used to configure the counter comparison consistence function in de-

velopment software.

■ Timing diagram



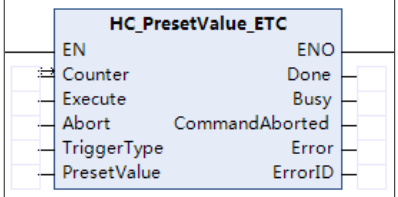
■ Error description

When an error occurs, locate the error specified by ErrorID in COUNTER_ERROR according to the help document, and find the reason for the error.

5.4. 4 HC_Presetvalue_ETC

This instruction presets the 2HCE counter.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
HC_Presetvalue_ETC	Preset 2HCE counter value	FB		HC_Presetvalue_ETC(Counter := , Execute:= , Abort:= , TriggerType:= , PresetValue:= , Done => , Busy => , CommandAborted=> , Error=> , ErrorID=>);

■ Variables

In-out variables

In-Out Variable	Name	Data Type	Value Range	Initial Value	Description
Counter	2HCE counter	AXIS_REF_ENCODER_ETC	-	-	AXIS_REF_ENCODER_ETC counter type

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
Execute	Enable	BOOL	[FALSE, TRUE]	FALSE	(Triggered by edges) Execution of the preset function switch

Input Variable	Name	Data Type	Value Range	Initial Value	Description
Abort	Abort	BOOL	[FALSE, TRUE]	FALSE	Triggered by level TRUE: Function block is aborted
TriggerType	Trigger type	PRESET_TRIGGER	[1, 7]	1	Internal trigger: 0x01 DI trigger: 0x02 Internal trigger and DI trigger: 0x03 Z phase trigger: 0x04 Internal trigger and Z phase trigger: 0x05 DI trigger and Z phase trigger: 0x06 Internal trigger, DI trigger, and Z phase trigger: 0x07
PresetValue	Preset value	LREAL	(– 1.7E 308, 1.7E 308)	0.0	Preset value of the counter

Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
Done	Comparison consistence completed state	BOOL	[FALSE, TRUE]	FALSE	The pulse count reaches the set comparison value, and the output holding time ends.
CommandAborted	Aborted state	BOOL	[FALSE, TRUE]	FALSE	Aborted state, which is set when Abort is TRUE
Busy	Execution flag	BOOL	[FALSE, TRUE]	FALSE	Execution of the preset function
Error	Error flag	BOOL	[FALSE, TRUE]	FALSE	TRUE: An internal error occurs in the function block
ErrorID	Error ID	COUNTER_ERROR	-	COUNTER_NO_ERROR	Error ID For details, see COUNTER_ERROR.

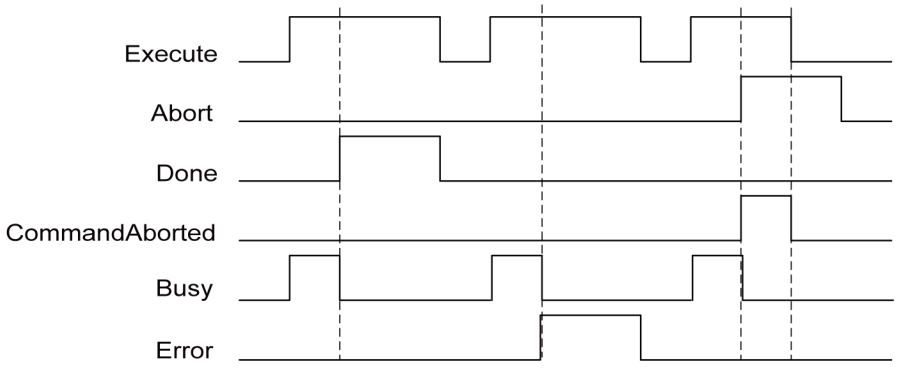
	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Counter	AXIS_REF_ENCODER_ETC																			
Execute	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Abort	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
TriggerType	PRESET_TRIGGER																			
PresetValue	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–
Done	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Busy	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
CommandAborted	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Error	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
ErrorID	COUNTER_ERROR																			

■ Function

This function block implements the preset function of a 2HCE counter, including seven available preset methods. When two or more trigger types are selected as a combination, the counter will be preset if any condition in the combination is met. After the preset is completed, the Done signal is output. The execution of the preset function block is valid only upon single trigger at the rising edge.

■ Timing diagram



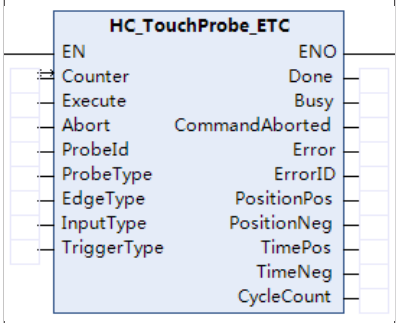
■ Error description

When an error occurs, locate the error specified by ErrorID in COUNTER_ERROR according to the help document, and find the reason for the error.

5.4. 5 HC_TouchProbe_ETC

This instruction implements the 2HCE counter probe function.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
HC_TouchProbe_ETC	2HCE counter probe	FB		HC_TouchProbe_ETC(Counter := , Execute:= , Abort:= , ProbeId:= , ProbeType:= , EdgeType:= , InputType:= , TriggerType:= , Done => , Busy => , CommandAborted=> , Error => , ErrorID => , PositionPos => , PositionNeg => , TimePos => , TimeNeg => , CycleCount =>);

■ Variables

In-out variables

In-Out Variable	Name	Data Type	Value Range	Initial Value	Description
Counter	2HCE counter	AXIS_REF_ENCODER_ETC	-	-	AXIS_REF_ENCODER_ETC counter type

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
Execute	Enable	BOOL	[FALSE, TRUE]	FALSE	(Triggered by edges) Execution of the preset function switch
Abort	Abort	BOOL	[FALSE, TRUE]	FALSE	Triggered by level TRUE: Function block is aborted
ProbeId	Probe ID	WORD	[1, 2]	1	Probe ID Only two probes are supported.
ProbeType	Probe type	TOUCH_PROBE_TYPE	-	PositionLatch	Probe type, including the time and position types
EdgeType	Trigger edge	TOUCH_PROBE_EDGE	[1, 3]	RisingEdge	1: Rising edge 2: Falling edge 3: Both rising edge and falling edge

Input Variable	Name	Data Type	Value Range	Initial Value	Description
InputType	Hardware trigger type	TOUCH_PROBE_INPUT	[0, 1]	DigtalInput	0: The probe input terminal is an external input terminal 1: The probe input terminal is a phase Z terminal
TriggerType	Trigger type	TOUCH_PROBE_TRIGGER	[0, 1]	TrigSingle	0: Single trigger for a probe 1: Continuous trigger for a probe

Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
Done	Probe execution status	BOOL	[FALSE, TRUE]	FALSE	Probe function execution completed
CommandAborted	Aborted state	BOOL	[FALSE, TRUE]	FALSE	Aborted state, which is set when Abort is TRUE
Busy	Execution flag	BOOL	[FALSE, TRUE]	FALSE	Execution of the count probe function
Error	Error flag	BOOL	[FALSE, TRUE]	FALSE	TRUE: An internal error occurs in the function block
ErrorID	Error ID	COUNTER_ERROR	-	COUNTER_NO_ERROR	Error ID For details, see COUNTER_ERROR.
PositionPos	Position latched on the rising edge	LREAL	(− 1.7E 308, 1.7E 308)	0.0	Position latched on the rising edge, in the unit of unit
PositionNeg	Position latched on the falling edge	LREAL	(− 1.7E 308, 1.7E 308)	0.0	Position latched on the falling edge, in the unit of unit
TimePos	Rising edge latch time	LINT	[0, 2^63)	0	Rising edge latch time, in the unit of ns
TimeNeg	Falling edge latch time	LINT	[0, 2^63)	0	Falling edge latch time, in the unit of ns
CycleCount	Probe check count value	WORD	[0, 2]	0	Valid count value of a probe in continuous trigger mode CycleCount accumulates once upon each successful trigger, with the value ranging from 0 to 2 in a cyclic manner.

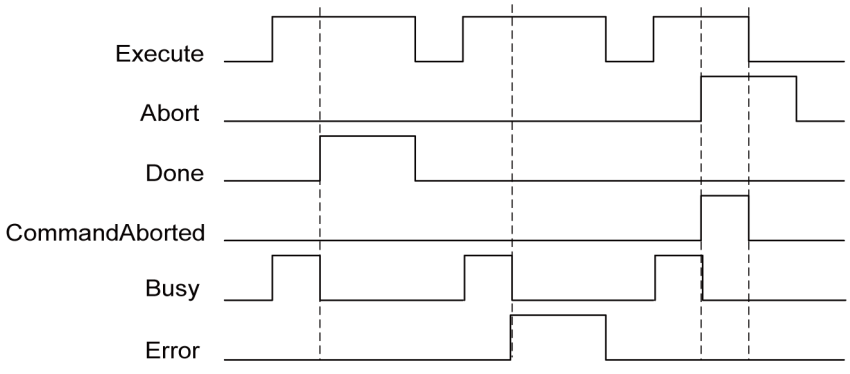
	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Counter	AXIS_REF_ENCODER_ETC																			
Execute	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Abort	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
ProbeId	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
ProbeType	TOUCH_PROBE_TYPE																			
EdgeType	TOUCH_PROBE_EDGE																			
InputType	TOUCH_PROBE_INPUT																			
TriggerType	TOUCH_PROBE_TRIGGER																			
Done	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Busy	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
CommandAborted	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Error	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
ErrorID	COUNTER_ERROR																			
PositionPos	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–
PositionNeg	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–
TimePos	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–
TimeNeg	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–
CycleCount	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

■ Function

This function block implements the probe function of a 2HCE counter, supporting two probes, both of which support the single trigger and continuous trigger modes.

The position probe and time probe can be used simultaneously. When using the probe function, check the configuration of the counter's probe attributes and PDO mapping. If the configuration is not reasonable, Error is set to TRUE and the COUNTER_ERROR error table is checked by ErrorID when the Execute function block is triggered.

■ Timing diagram



■ Error description

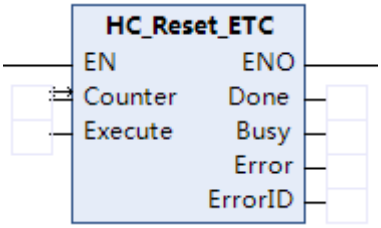
When an error occurs, locate the error specified by ErrorID in COUNTER_ERROR according to the help doc-

ument, and find the reason for the error.

5.4. 6 HC_Reset_ETC

This instruction resets the 2HCE counter for an error.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
HC_Reset_ETC	Reset 2HCE counter	FB		HC_Reset_ETC(Counter := , Execute:= , Done => , Busy => , Error => , ErrorID => ,);

■ Variables

In-out variables

In-Out Variable	Name	Data Type	Value Range	Initial Value	Description
Counter	2HCE counter	AXIS_REF_ENCODER_ETC	-	-	AXIS_REF_ENCODER_ETC counter type

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
Execute	Enable	BOOL	[FALSE, TRUE]	FALSE	(Triggered by edges) Execution of the error count function switch

Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
Done	Reset completed state	BOOL	[FALSE, TRUE]	FALSE	Reset function execution completed
Busy	Execution flag	BOOL	[FALSE, TRUE]	FALSE	TRUE: The counter is being reset
Error	Error flag	BOOL	[FALSE, TRUE]	FALSE	TRUE: An internal error occurs in the function block
ErrorID	Error ID	COUNTER_ERROR	-	COUNTER_NO_ERROR	Error ID For details, see COUNTER_ERROR.

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING

Counter	AXIS_REF_ENCODER_ETC																		
Execute	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Done	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Busy	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Error	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
ErrorID	COUNTER_ERROR																		

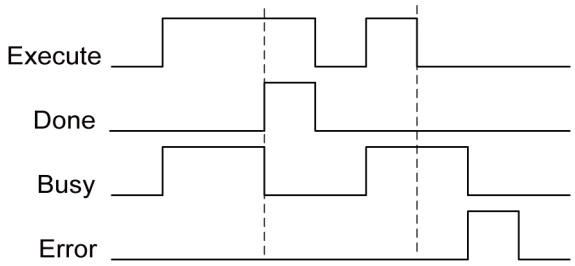
■ Function

This function block resets the 2HCE counter for a fault or an error.

When the counter is faulty, use the HC_Reset_ETC function block to clear the fault. If the function block cannot clear the fault, restart the count module.

If the function block fails, change Execute to FALSE to clear the error flag of the function block.

■ Timing diagram



■ Error description

When an error occurs, locate the error specified by ErrorID in COUNTER_ERROR according to the help document, and find the reason for the error.

5.4. 7 Counter_ETC

This function block implements the enable, count, frequency measurement, and count direction output functions of the 8PBE counter.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
Counter_ETC	Enable 8PBE counter	FB	Counter_ETC_0 Counter_ETC EN ENO Counter CounterValue Enable Frequency Direction Valid Busy Error ErrorID	Counter_ETC_0(Counter :=, Enable :=, CounterValue =>, Frequency =>, Direction =>, Valid =>, Busy =>, Error =>, ErrorID =>,)

■ Variables

In-out variables

In-Out Variable	Name	Data Type	Value Range	Initial Value	Description
Counter	8PBE counter	COUNTER_REF_ETC	Depends on data types	-	COUNTER_REF_ETC counter type

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
Enable	Enable	BOOL	[FALSE, TRUE]	FALSE	Triggered by level TRUE: Enable the counter

Output variables

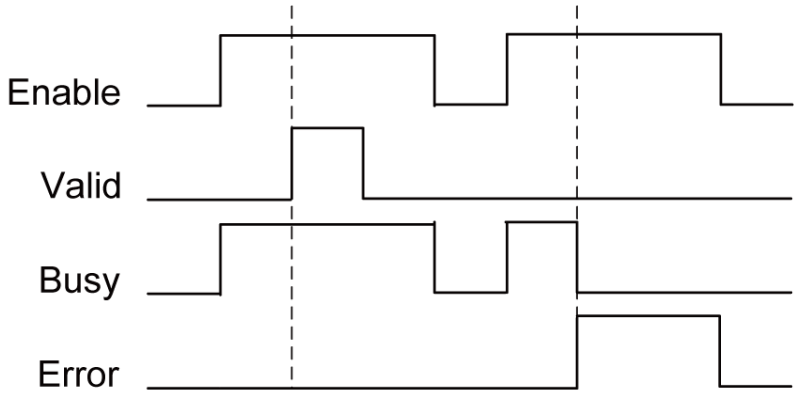
Output Variable	Name	Data Type	Value Range	Initial Value	Description
CounterValue	Current count value	LREAL	(− 1.7E 308, 1.7E 308)	0.0	Current count value, which is converted from pulse count
Frequency	Frequency	UDINT	[0, 2^32)	0	Frequency of the collected pulse
Direction	Direction	BOOL	[FALSE, TRUE]	FALSE	Count direction
Valid	Enabled	BOOL	[FALSE, TRUE]	FALSE	Count enabled state
Busy	Execution flag	BOOL	[FALSE, TRUE]	FALSE	Running the function block
Error	Error flag	BOOL	[FALSE, TRUE]	FALSE	TRUE: An internal error occurs in the function block
ErrorID	Error ID	CT_ERROR	Depends on data types	-	Error ID

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Counter	COUNTER_REF_ETC																			
Enable	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
CounterValue	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–
Frequency	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–
Direction	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Valid	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Busy	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Error	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
ErrorID	CT_ERROR																			

■ Function

This function block implements the enable, count, frequency measurement, and count direction output functions of the 8PBE counter.

■ Timing diagram



■ Error description

When an error occurs, locate the error specified by ErrorID in CT_ERROR according to the help document, and find the reason for the error.

5.4. 8 Counter_SetCompare_ETC

This instruction implements the comparison consistence output for the 8PBE counter.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
Counter_SetCompare_ETC	8PBE counter comparison output	FB	Counter_SetCompare_ETC_0 Counter_SetCompare_ETC EN ENO Counter Done Execute Busy Abort CommandAborted Channel Error CompareValue ErrorID ImRefreshCycle	Counter_SetCompare_ETC(Counter := , Execute:= , Abort:= , Channel:= , CompareValue:= , ImRefreshCycle:= , Done => , Busy => , CommandAborted=> , Error=> , ErrorID=>);

■ Variables

In-out variables

In-Out Variable	Name	Data Type	Value Range	Initial Value	Description
Counter	8PBE counter	COUNTER_REF_ETC	Depends on data types	-	COUNTER_REF_ETC counter type

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
Execute	Enable	BOOL	[FALSE, TRUE]	FALSE	(Triggered by edges) Execution of the function block switch

Input Variable	Name	Data Type	Value Range	Initial Value	Description
Abort	Abort	BOOL	[FALSE, TRUE]	FALSE	Triggered by level TRUE: Function block is aborted
Channel	Comparison channel	BYTE	[1, 2]	1	Comparison channel of the counter Two channels are available.
CompareValue	Comparison value	LREAL	(− 1.7E 308, 1.7E 308)	0.0	Comparison value of the float type, in the unit of unit
ImRefreshCycle	Output holding time	UINT	Depends on data types	-	The holding time of the output port is enabled.

Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
Done	Comparison consistence completed state	BOOL	[FALSE, TRUE]	FALSE	The pulse count reaches the set comparison value, and the output holding time ends.
CommandAborted	Aborted state	BOOL	[FALSE, TRUE]	FALSE	Aborted state, which is set when Abort is TRUE
Busy	Execution flag	BOOL	[FALSE, TRUE]	FALSE	Execution of the comparison consistence function
Error	Error flag	BOOL	[FALSE, TRUE]	FALSE	TRUE: An internal error occurs in the function block
ErrorID	Error ID	CT_ERROR	Depends on data types	-	Error ID

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Counter	COUNTER_REF_ETC																			
Execute	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Abort	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Channel	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
CompareValue	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–
ImRefreshCycle	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–
Done	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
CommandAborted	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Busy	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Error	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
ErrorID	CT_ERROR																			

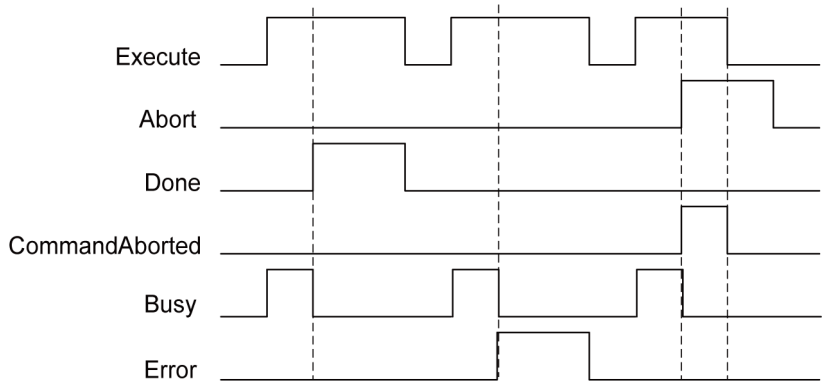
■ Function

This function block triggers a hardware high-speed output port and retains it for a certain period when the current count value of the counter is equal to the set comparison value.

The comparison consistence output function is a single trigger. When Done is TRUE, the comparison consistency function is completed. To continue using the comparison function, trigger Execute again.

Confirm the output point attribute used to configure the counter comparison consistence function in development software.

■ Timing diagram



■ Error description

When an error occurs, locate the error specified by ErrorID in CT_ERROR according to the help document, and find the reason for the error.

5.4. 9 Counter_Presetvalue_ETC

This instruction presets the 8PBE counter.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
Counter_Presetvalue_ETC	Preset 8PBE counter value	FB	Counter_PresetValue_ETC_0 Counter_PresetValue_ETC EN ENO Counter Done Execute Busy Abort CommandAborted TriggerType Error PresetValue ErrorID	Counter_Presetvalue_ETC(Counter := , Execute:= , Abort:= , TriggerType:= , PresetValue:= , Done => , Busy => , CommandAborted=> , Error=> , ErrorID=>);

■ Variables

In-out variables

In-Out Variable	Name	Data Type	Value Range	Initial Value	Description
Counter	8PBE counter	COUNTER_REF_ETC	Depends on data types	-	COUNTER_REF_ETC counter type

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
Execute	Enable	BOOL	[FALSE, TRUE]	FALSE	(Triggered by edges) Execution of the preset function switch
Abort	Abort	BOOL	[FALSE, TRUE]	FALSE	Triggered by level TRUE: Function block is aborted
TriggerType	Trigger type	BYTE	Depends on data types	-	Internal trigger: 0x01 DI trigger: 0x02 Internal trigger and DI trigger: 0x03 Z phase trigger: 0x04 Internal trigger and Z phase trigger: 0x05 DI trigger and Z phase trigger: 0x06 Internal trigger, DI trigger, and Z phase trigger: 0x07
PresetValue	Preset value	LREAL	(– 1.7E 308, 1.7E 308)	0.0	Preset value of the counter

Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
Done	Comparison consistence completed state	BOOL	[FALSE, TRUE]	FALSE	The pulse count reaches the set comparison value, and the output holding time ends.
CommandAborted	Aborted state	BOOL	[FALSE, TRUE]	FALSE	Aborted state, which is set when Abort is TRUE
Busy	Execution flag	BOOL	[FALSE, TRUE]	FALSE	Execution of the preset function
Error	Error flag	BOOL	[FALSE, TRUE]	FALSE	TRUE: An internal error occurs in the function block
ErrorID	Error ID	CT_ERROR	Depends on data types	-	Error ID

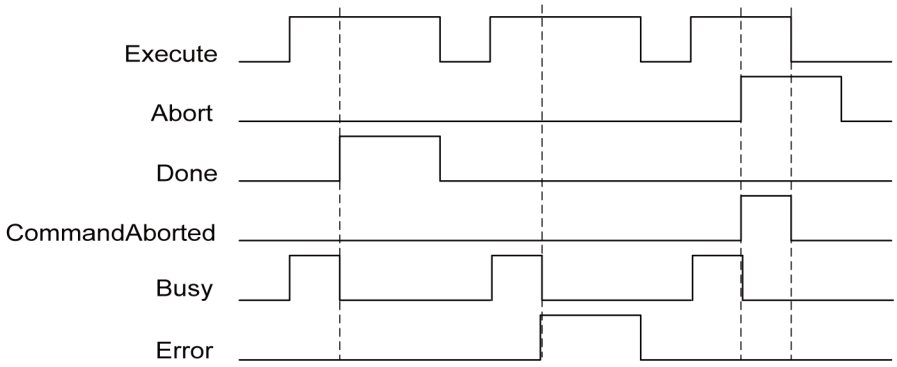
	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Counter	COUNTER_REF_ETC																			
Execute	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Abort	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
TriggerType	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
PresetValue	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–
Done	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Busy	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
CommandAborted	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Error	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
ErrorID	CT_ERROR																			

■ Function

This function block implements the preset function of an 8PBE counter, including seven available preset methods. When two or more trigger types are selected as a combination, the counter will be preset if any condition in the combination is met. After the preset is completed, the Done signal is output. The execution of the preset function block is valid only upon single trigger at the rising edge.

■ Timing diagram



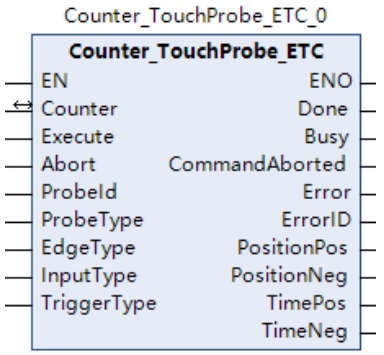
■ Error description

When an error occurs, locate the error specified by ErrorID in CT_ERROR according to the help document, and find the reason for the error.

5.4. 10 Counter_TouchProbe_ETC

This instruction implements the 8PBE counter probe function.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
Counter_TouchProbe_ETC	8PBE counter probe	FB		Counter_TouchProbe_ETC(Counter := , Execute:= , Abort:= , ProbeId:= , ProbeType:= , EdgeType:= , InputType:= , TriggerType:= , Done => , Busy => , CommandAborted=> , Error => , ErrorID => , PositionPos => , PositionNeg => , TimePos => , TimeNeg =>);

■ Variables

In-out variables

In-Out Variable	Name	Data Type	Value Range	Initial Value	Description
Counter	8PBE counter	COUNTER_REF_ETC	Depends on data types	-	COUNTER_REF_ETC counter type

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
Execute	Enable	BOOL	[FALSE, TRUE]	FALSE	(Triggered by edges) Execution of the preset function switch
Abort	Abort	BOOL	[FALSE, TRUE]	FALSE	Triggered by level TRUE: Function block is aborted
ProbeId	Probe ID	WORD	(0, 7)	-	Probe ID
ProbeType	Probe type	TOUCH_PROBE_TYPE	-	PositionLatch	Probe type, including the time and position types
EdgeType	Trigger edge	TOUCH_PROBE_EDGE	[1, 3]	RisingEdge	1: Rising edge 2: Falling edge 3: Both rising edge and falling edge
InputType	Hardware trigger type	TOUCH_PROBE_INPUT	[0, 1]	DigtalInput	0: The probe input terminal is an external input terminal 1: The probe input terminal is a phase Z terminal

Input Variable	Name	Data Type	Value Range	Initial Value	Description
TriggerType	Trigger type	TOUCH_ PROBE_ TRIGGER	[0, 1]	TrigSingle	0: Single trigger for a probe 1: Continuous trigger for a probe

Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
Done	Probe execution status	BOOL	[FALSE, TRUE]	FALSE	Probe function execution completed
CommandAborted	Aborted state	BOOL	[FALSE, TRUE]	FALSE	Aborted state, which is set when Abort is TRUE
Busy	Execution flag	BOOL	[FALSE, TRUE]	FALSE	Execution of the count probe function
Error	Error flag	BOOL	[FALSE, TRUE]	FALSE	TRUE: An internal error occurs in the function block
ErrorID	Error ID	CT_ERROR	Depends on data types	-	Error ID
PositionPos	Position latched on the rising edge	LREAL	(− 1.7E 308, 1.7E 308)	0.0	Position latched on the rising edge, in the unit of unit
PositionNeg	Position latched on the falling edge	LREAL	(− 1.7E 308, 1.7E 308)	0.0	Position latched on the falling edge, in the unit of unit
TimePos	Rising edge latch time	LINT	[0, 2^63)	0	Rising edge latch time, in the unit of ns
TimeNeg	Falling edge latch time	LINT	[0, 2^63)	0	Falling edge latch time, in the unit of ns

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Counter	COUNTER_REF_ETC																			
Execute	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Abort	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
ProbeId	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
ProbeType	TOUCH_PROBE_TYPE																			
EdgeType	TOUCH_PROBE_EDGE																			
InputType	TOUCH_PROBE_INPUT																			
TriggerType	TOUCH_PROBE_TRIGGER																			
Done	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Busy	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
CommandAborted	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

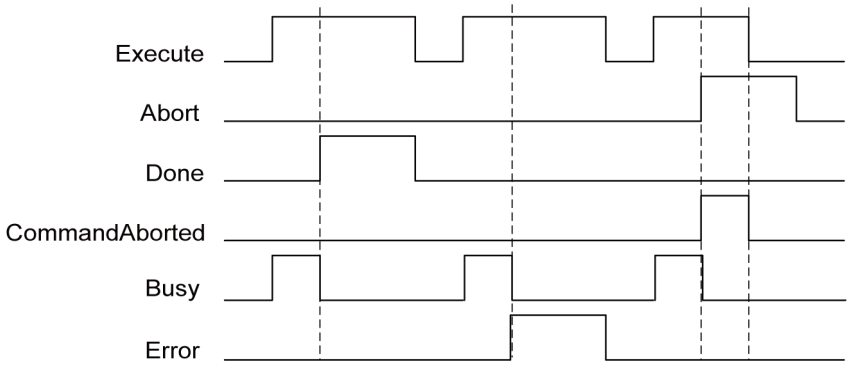
	Bool-ean	Bit String				Integer								Real Num-ber		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Error	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
ErrorID	CT_ERROR																			
PositionPos	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–
PositionNeg	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–
TimePos	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–
TimeNeg	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–

4) Function

This function block implements the probe function of an 8PBE counter. All the probes support the single trigger and continuous trigger modes.

The position probe and time probe can be used simultaneously. When using the probe function, check the configuration of the counter's probe attributes and PDO mapping. If the configuration is not reasonable, Error is set to TRUE and the COUNTER_ERROR error table is checked by ErrorID when the Execute function block is triggered.

■ Timing diagram



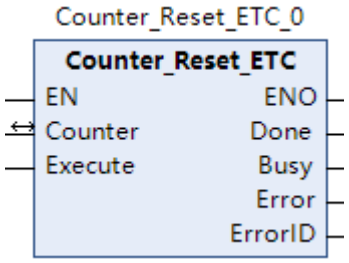
■ Error description

When an error occurs, locate the error specified by ErrorID in CT_ERROR according to the help document, and find the reason for the error.

5.4. 11 Counter_Reset_ETC

This instruction resets the 8PBE counter for an error.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
Counter_Reset_ETC	Reset 8PBE counter	FB		Counter_Reset_ETC(Counter := , Execute:= , Done => , Busy => , Error => , ErrorID => ,);

■ Variables

In-out variables

In-Out Variable	Name	Data Type	Value Range	Initial Value	Description
Counter	8PBE counter	COUNTER_REF_ETC	Depends on data types	-	COUNTER_REF_ETC counter type

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
Execute	Enable	BOOL	[FALSE, TRUE]	FALSE	(Triggered by edges) Execution of the error count function switch

Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
Done	Reset completed state	BOOL	[FALSE, TRUE]	FALSE	Reset function execution completed
Busy	Execution flag	BOOL	[FALSE, TRUE]	FALSE	TRUE: The counter is being reset
Error	Error flag	BOOL	[FALSE, TRUE]	FALSE	TRUE: An internal error occurs in the function block
ErrorID	Error ID	CT_ERROR	Depends on data types	-	Error ID

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Counter	COUNTER_REF_ETC																			
Execute	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Done	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Busy	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Error	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
ErrorID	CT_ERROR																			

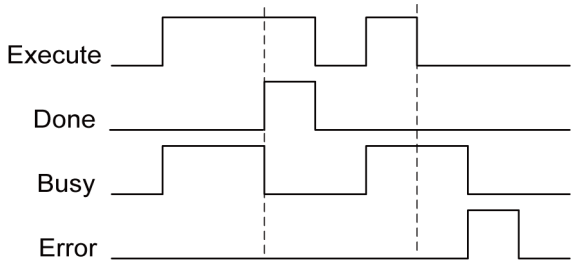
■ Function

This function block resets the 8PBE counter for a fault or an error.

When the counter is faulty, use the HC_Reset_ETC function block to clear the fault. If the function block cannot clear the fault, restart the count module.

If the function block fails, change Execute to FALSE to clear the error flag of the function block.

■ Timing diagram



■ Error description

When an error occurs, locate the error specified by ErrorID in CT_ERROR according to the help document, and find the reason for the error.

Chapter 6 Communication Instructions

6. 1 Free TCP Communication Instructions

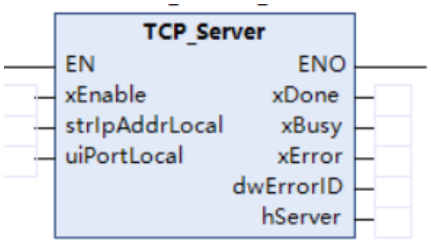
6.1. 1 Instruction List

Instruction Category	Name	FB/FC	Function
Free TCP communication instructions	TCP_Server	FB	Create TCP server communication
	TCP_Client	FB	Create TCP client communication
	TCP_Connect	FB	Create TCP client, and connect to server
	TCP_Receive	FB	Receive data
	TCP_Send	FB	Send data

6.1. 2 TCP_Server

This instruction creates communication services on the TCP server. When xEnable is set to TRUE, a valid TCP communication handle is created for the local server to communicate with a remote client (the value hServer is not 0), and xBusy remains TRUE. If any service-related errors occur during the communication, xDone is set to TRUE, and the function blocks on the server side get unavailable. In such cases, TCP_Server enable needs to be re-triggered.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
TCP_Server	Create TCP server communication	FB		TCP_Server(xEnable:= , strIpAddrLocal:= , uiPortLocal:= , xDone=> , xBusy=> , xError=> , dwErrorID=> , hServer=>);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
xEnable	Function block enable	BOOL	-	-	Enabling the function block at the input high level
strIpAddrLocal	Local IP address	STRING	-	-	IP address of the local PLC communication network port
uiPortLocal	Local port number	UINT	-	-	Local communication port number

Output variables

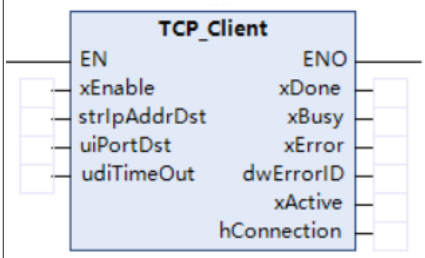
Output Variable	Name	Data Type	Value Range	Initial Value	Description
xDone	Ending flag	BOOL	-	-	Ending flag of communication on the server
xBusy	Busy flag	BOOL	-	-	Busy state
xError	Error flag	BOOL	-	-	Error state
dwErrorID	Error code	DWORD	-	-	Error code
hServer	Server handle	_XWORD	-	-	TCP server handle

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
xEnable	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
strIpAddrLocal	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√
uiPortLocal	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-
xDone	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
xBusy	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
xError	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
dwErrorID	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
hServer	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

6.1. 3 TCP_Client

This instruction creates communication services on the TCP client. When a high level is detected for xEnable, the local client requests to connect to the destination server. When the connection is successful, the communication handle hConnection between the server and the client is created, and xActive is set to TRUE.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
TCP_Client	Create TCP client communication	FB		TCP_Client(xEnable:= , strIpAddrDst:= , uiPortDst:= , udiTimeOut:= , xDone=> , xBusy=> , xError=> , dwErrorID=> , xActive=> , hConnection=>);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
xEnable	Function block enable	BOOL	-	-	Enabling the function block at the input high level
strIpAddrDst	Destination device IP address	STRING	-	-	IP address for destination device communication
uiPortDst	Destination device port	UINT	-	-	Port number for destination device communication
udiTimeout	Timeout period	UDINT	-	-	Timeout period for requesting a connection The unit is us and the default value is 500 ms.

Output variables

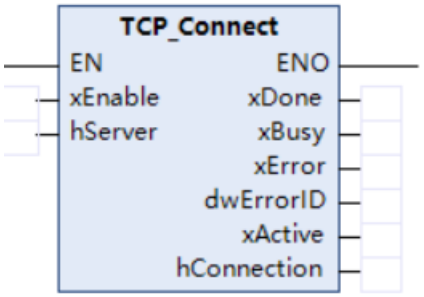
Output Variable	Name	Data Type	Value Range	Initial Value	Description
xDone	Ending signal	BOOL	-	-	Client communication ending signal
xBusy	Busy flag	BOOL	-	-	Busy state
xError	Error flag	BOOL	-	-	Error state
dwErrorID	Error code	DWORD	-	-	Error code
xActive	Successful connection flag	BOOL	-	-	Flag of successful connection between the client and the destination server
hConnection	Connection handle	_XWORD	-	-	Connection handle for communication between the server and the client

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
xEnable	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
strIpAddrDst	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√
uiPortDst	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-
udiTimeout	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-
xDone	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
xBusy	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
xError	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
dwErrorID	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
xActive	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
hConnection	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

6.1. 4 TCP_Connect

When a high level is detected for xEnable, the local server monitors the connection request of the destination client. After successful connection between the client and the server, this instruction creates the connection handle hConnection for the communication between them.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
TCP_Connect	Create TCP client, and connect to server	FB		TCP_Connect(xEnable:=, hServer:=, xDone=> , xBusy=> , xError=> , dwErrorID=> , xActive=> , hConnection=>);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
xEnable	Function block enable	BOOL	-	-	Enabling the function block at the input high level
hServer	Server handle	_XWORD	-	-	TCP server handle

Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
xDone	Ending signal	BOOL	-	-	Monitoring ending signal
xBusy	Busy flag	BOOL	-	-	Busy state
xError	Error flag	BOOL	-	-	Error state
dwErrorID	Error code	DWORD	-	-	Error code
xActive	Connection flag	BOOL	-	-	Connection handle flag for the server to monitor the destination client
hConnection	Connection handle	_XWORD	-	-	Connection handle for communication between the server and the client

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
xEnable	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
hServer	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
xDone	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
xBusy	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
xError	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
dwErrorID	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
xActive	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
hConnection	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

■ Precautions

When the communication is connection, xActive is set to TRUE. When the communication is disconnected, xActive is automatically reset.

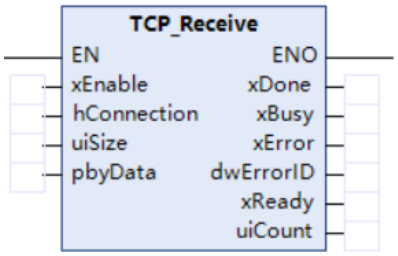
To monitor the communication disconnection, the TCP_Receive function block is also required. The disconnection can be monitored only when xBusy of the TCP_Receive instruction is set to TRUE.

6.1. 5 TCP_Receive

When the input pin xEnable is set to TRUE for the TCP_Receive instruction, this instruction reads data from the TCP communication buffer, and xBusy is set to TRUE.

If data reading is successful, the read data is stored in the array with pbyData as the head address. In addition, uiCount specifies the volume of output data, xReady is set to TRUE, and two variables remain a scan period. If data reading fails due to any network communication errors, xDone is set to TRUE, the data reading function fails, and even TCP_Connection needs to be triggered again and xEnable needs to be reset to TRUE for the TCP_Client function block.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
TCP_Receive	Receive data	FB		TCP_Receive(xEnable:= , hConnection:= , uiSize:=, pbyData:=, xDone=> , xBusy=> , xError=> , dwErrorID=> , xReady=> , uiCount=>);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
xEnable	Function block enable	BOOL	-	-	Enabling the function block at the input high level
hConnection	Connection handle	DWORD	-	-	Connection handle for TCP communication
uiSize	Size of data to receive	UINT	-	-	Size of the area for receiving data, in the unit of byte
PbyData	Pointer to the receiving area	POINTER_ TO_BYTE	-	-	Pointer to the first byte of the receiving area

Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
xDone	Ending signal	BOOL	-	-	Data receiving ending signal
xBusy	Busy flag	BOOL	-	-	Busy state
xError	Error flag	BOOL	-	-	Error state
dwErrorID	Error code	DWORD	-	-	Error code

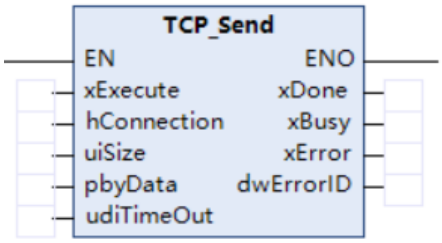
Output Variable	Name	Data Type	Value Range	Initial Value	Description
xReady	Successful connection flag	BOOL	-	-	Flag of setting a scan period when data is read from the buffer
uiCount	Connection handle	_UXINT	-	-	Volume of data read from the reception buffer

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
xEnable	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
hConnection	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
uiSize	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-
PbyData	POINTER_TO_BYTE																			
xDone	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
xBusy	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
xError	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
dwErrorID	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
xReady	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
uiCount	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-

6.1. 6 TCP_Send

When xExecute is set to TRUE, this instruction sends the user-defined data of which the length is uiSize and the head address is pbyData in the sending buffer to the destination device. If the data is successfully sent within the timeout period, xDone is set to TRUE.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
TCP_Send	Send data	FB		TCP_Send(xExecute:= , hConnection:= , uiSize:= , pbyData:= , udiTimeOut:= , xDone=> , xBusy=> , xError=> , dwErrorID=>);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
xExecute	Function block enable	BOOL	-	-	Enabling the function block at the input high level
hConnection	Connection handle	_XWORD	-	-	Connection handle for TCP communication

Input Variable	Name	Data Type	Value Range	Initial Value	Description
uiSize	Size of data to send	UINT	-	-	Size of the data sending area, in the unit of bytes
PbyData	Pointer to the sending area	POINTER_ TO_BYTE	-	-	Pointer to the first byte of the sending area
uditimeOut	Timeout period	UDINT	-	-	Timeout period for data sending The unit is us and the default value is 500 ms.

Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
xDone	Done signal	BOOL	-	-	Data sent signal
xBusy	Busy flag	BOOL	-	-	Busy state
xError	Error flag	BOOL	-	-	Error state
dwErrorID	Error code	DWORD	-	-	Error code

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
xEnable	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
hConnection	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
uiSize	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-
PbyData	POINTER_TO_BYTE																			
uditimeOut	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-
xDone	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
xBusy	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
xError	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
dwErrorID	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

6.1. 7 TCP Communication Instruction Examples

AM600 as the server and commissioning assistant as the client

Example: AM600 functions as the server (IP address: 192.168.1.88) and the commissioning assistant functions as the client.

ST

To establish a connection, trigger TCP_Server and TCP_Connect. The input pin hServer of the TCP_ Connect instruction is the output pin hServer of the TCP_Server instruction.

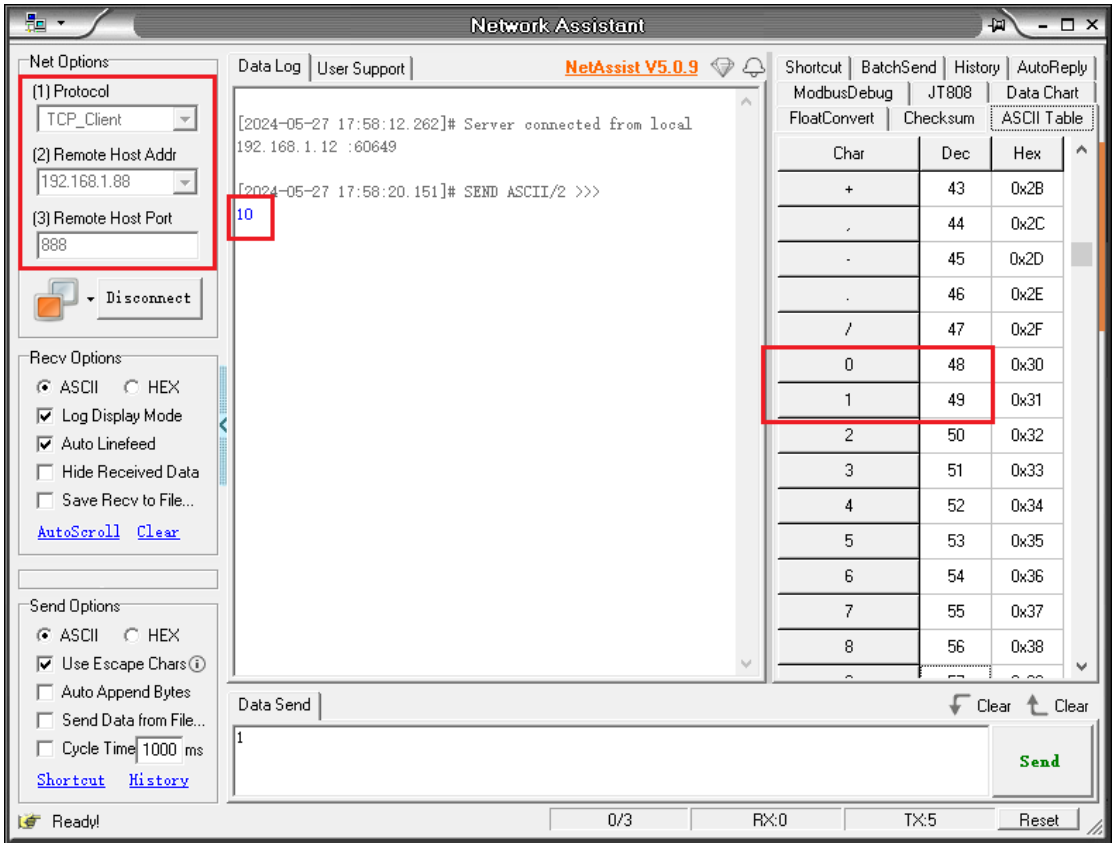
Trigger connection to the commissioning assistant. Ensure that the IP address and port number are consistent. For example, the port number is 888.

Expression	Type	Value
TCP_Server_0	TCP_Server	
TCP_Connect_0	TCP_Connect	

```
1   //AM600 as the server
2   TCP_Server_0(
3       xEnable TRUE :=TRUE ,
4       strIpAddrLocal '192.168.1. ▸ :='192.168.1.88' ,
5       uiPortLocal 888 :=888 ,
6       xDone=> ,
7       xBusy=> ,
8       xError=> ,
9       dwErrorID=> ,
10      hServer=> );
11
12  TCP_Connect_0(
13      xEnable TRUE :=TRUE ,
14      hServer 3032623684 :=TCP_Server_0.hServer 3032623684 ,
15      xDone=> ,
16      xBusy=> ,
17      xError=> ,
18      dwErrorID=> ,
19      xActive=> ,
20      hConnection=> );
```

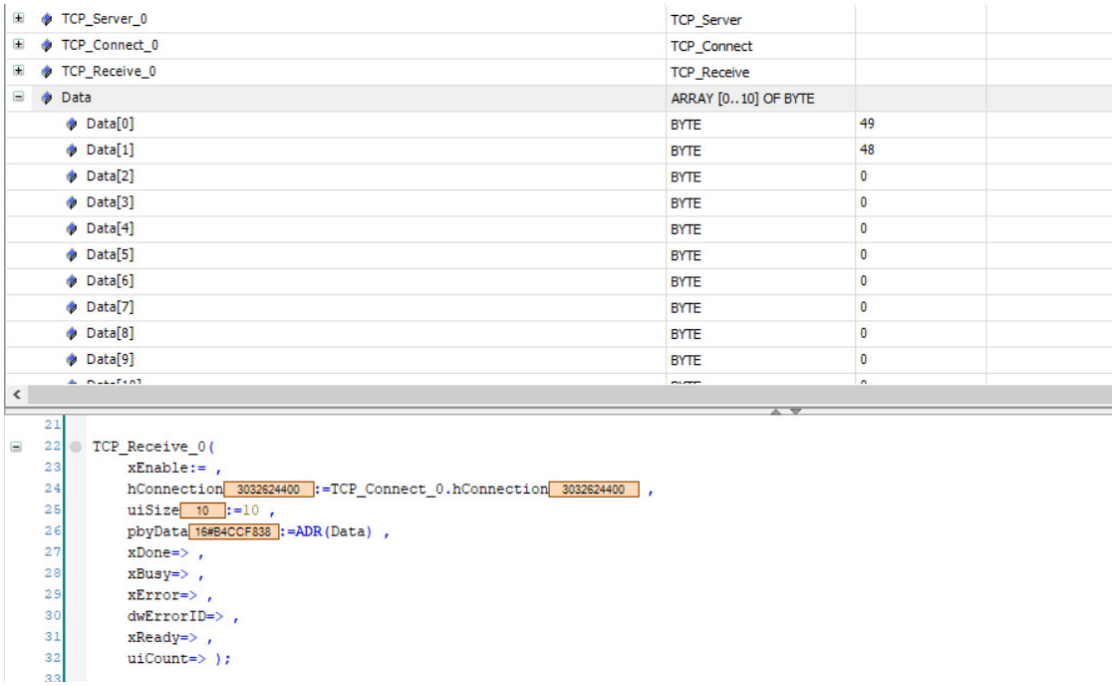
Receive data

Use the commissioning assistant to send data "1" and "0".



The corresponding decimal values are "48" and "49".

Trigger TCP_Receive and ensure that the received data is in the address pointed to by the pbyData pointer. As shown in the following figure, the two values in the array pointed to by the pointer are respectively "48" and "49", indicating successful receiving.

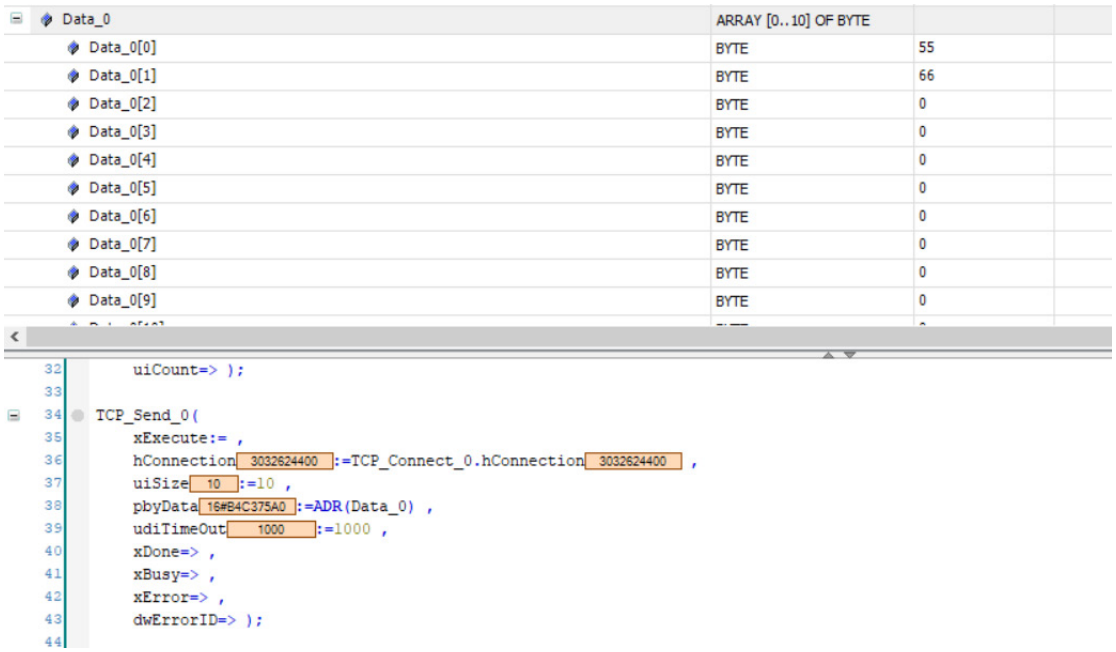


TCP_Server_0	TCP_Server	
TCP_Connect_0	TCP_Connect	
TCP_Receive_0	TCP_Receive	
Data	ARRAY [0..10] OF BYTE	
Data[0]	BYTE	49
Data[1]	BYTE	48
Data[2]	BYTE	0
Data[3]	BYTE	0
Data[4]	BYTE	0
Data[5]	BYTE	0
Data[6]	BYTE	0
Data[7]	BYTE	0
Data[8]	BYTE	0
Data[9]	BYTE	0

```
21
22  TCP_Receive_0(
23      xEnable:= ,
24      hConnection 3032624400 :=TCP_Connect_0.hConnection 3032624400 ,
25      uiSize 10 :=10 ,
26      pbyData 16#B4CCF838 :=ADR(Data) ,
27      xDone=> ,
28      xBusy=> ,
29      xError=> ,
30      dwErrorID=> ,
31      xReady=> ,
32      uiCount=> );
33
```

Send data

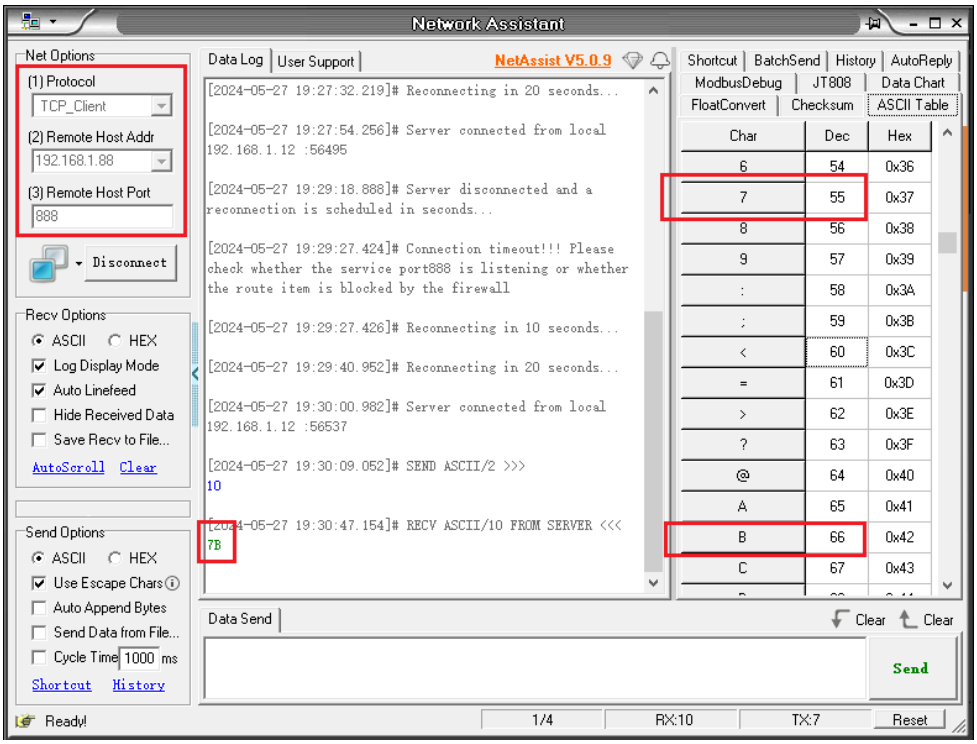
Trigger TCP_Send and ensure that the sent data is in the address pointed to by the pbyData pointer. As shown in the following figure, the two values in the array pointed to by the pointer are respectively "55" and "66".



Data_0	ARRAY [0..10] OF BYTE	
Data_0[0]	BYTE	55
Data_0[1]	BYTE	66
Data_0[2]	BYTE	0
Data_0[3]	BYTE	0
Data_0[4]	BYTE	0
Data_0[5]	BYTE	0
Data_0[6]	BYTE	0
Data_0[7]	BYTE	0
Data_0[8]	BYTE	0
Data_0[9]	BYTE	0

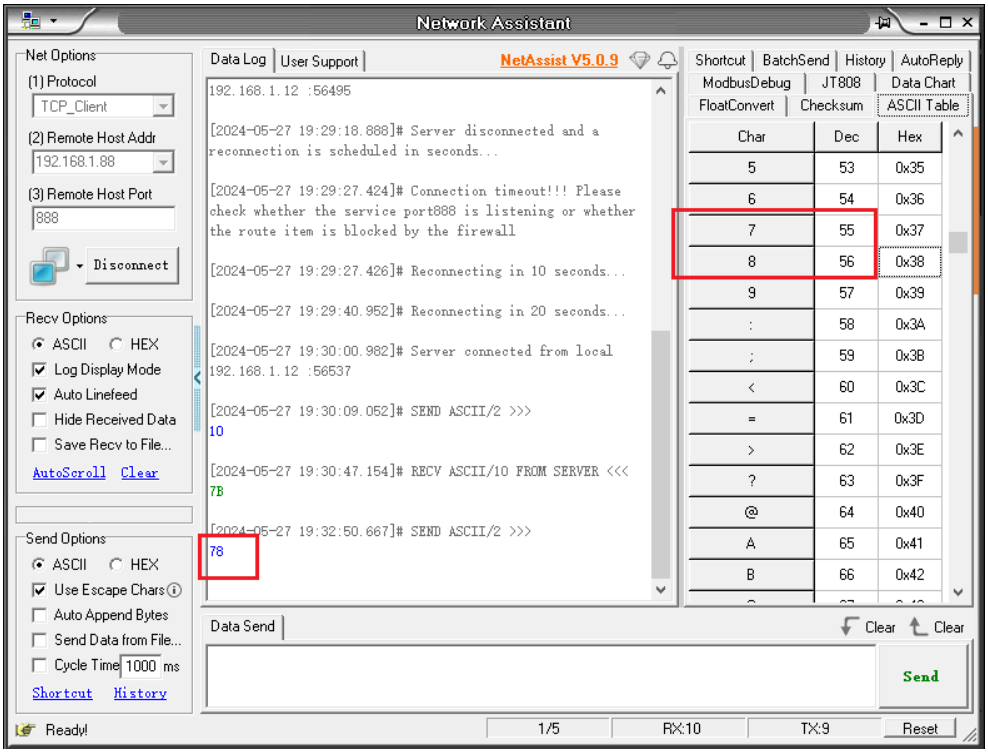
```
32      uiCount=> );
33
34  TCP_Send_0(
35      xExecute:= ,
36      hConnection 3032624400 :=TCP_Connect_0.hConnection 3032624400 ,
37      uiSize 10 :=10 ,
38      pbyData 16#B4C375A0 :=ADR(Data_0) ,
39      udiTimeOut 1000 :=1000 ,
40      xDone=> ,
41      xBusy=> ,
42      xError=> ,
43      dwErrorID=> );
44
```

Check the commissioning assistant. The received data "7" and "B" corresponds to "55" and "66" respectively, indicating successful sending.

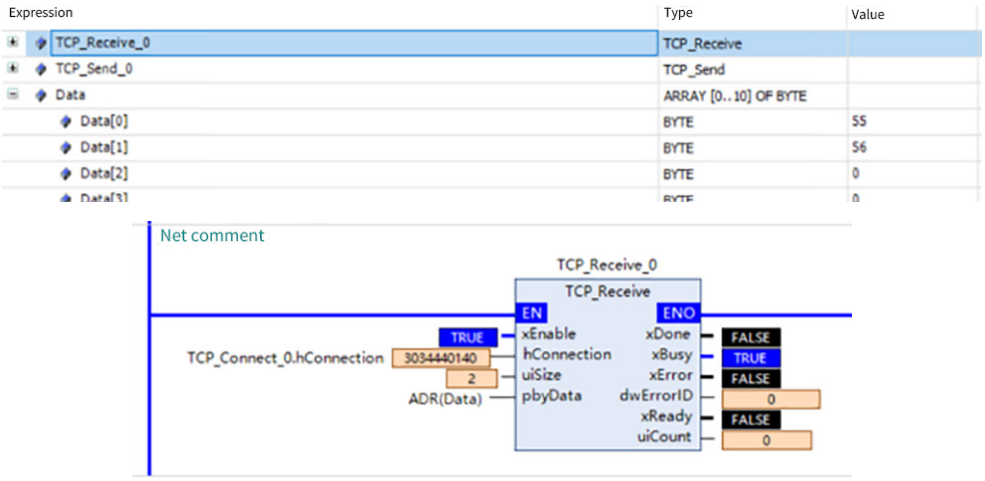


LD

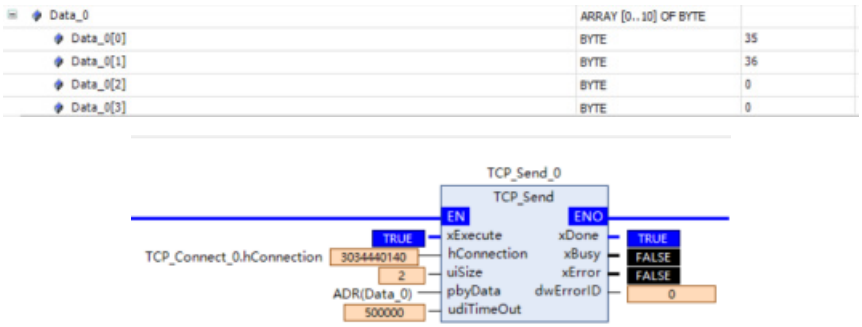
Receive data and use the commissioning assistant to send "7" and "8".



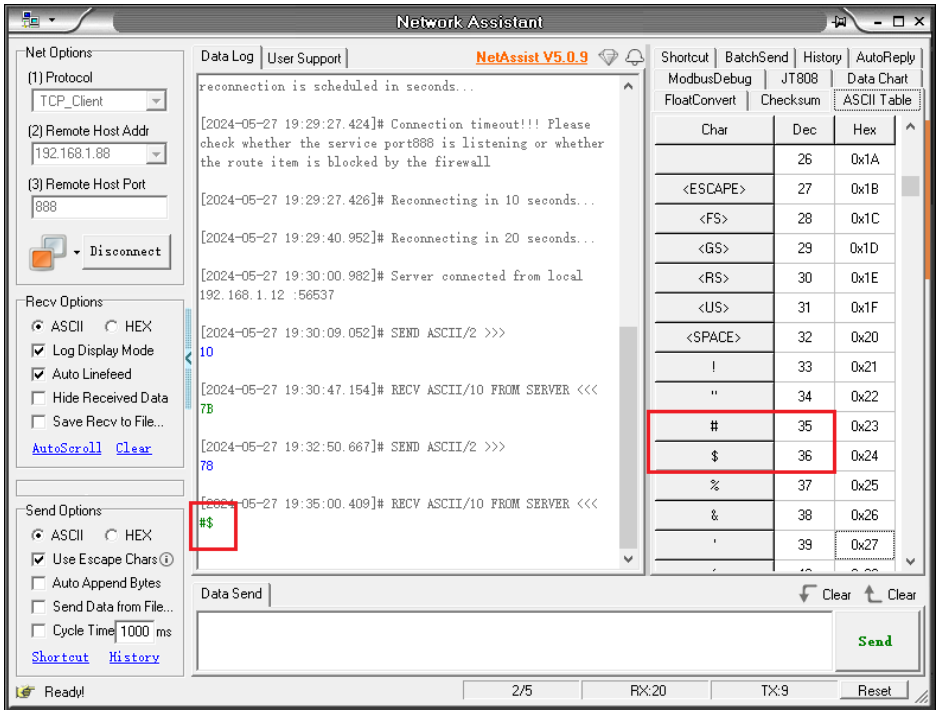
The two values in the array pointed to by the pbyData pin pointer of TCP_Receive instruction are respectively "55" and "56", indicating successful receiving.



Send data, write "35" and "36" to the first two array elements in the sending buffer of TCP_Send, and set Size to 2 for the sent bytes.



Check the commissioning assistant. The values obtained are "#" and "$", which respectively correspond to values "35" and "36", indicating successful sending.

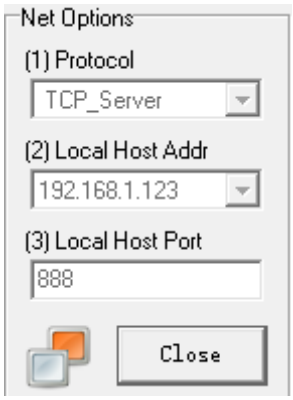


AM600 as the client and commissioning assistant as the server

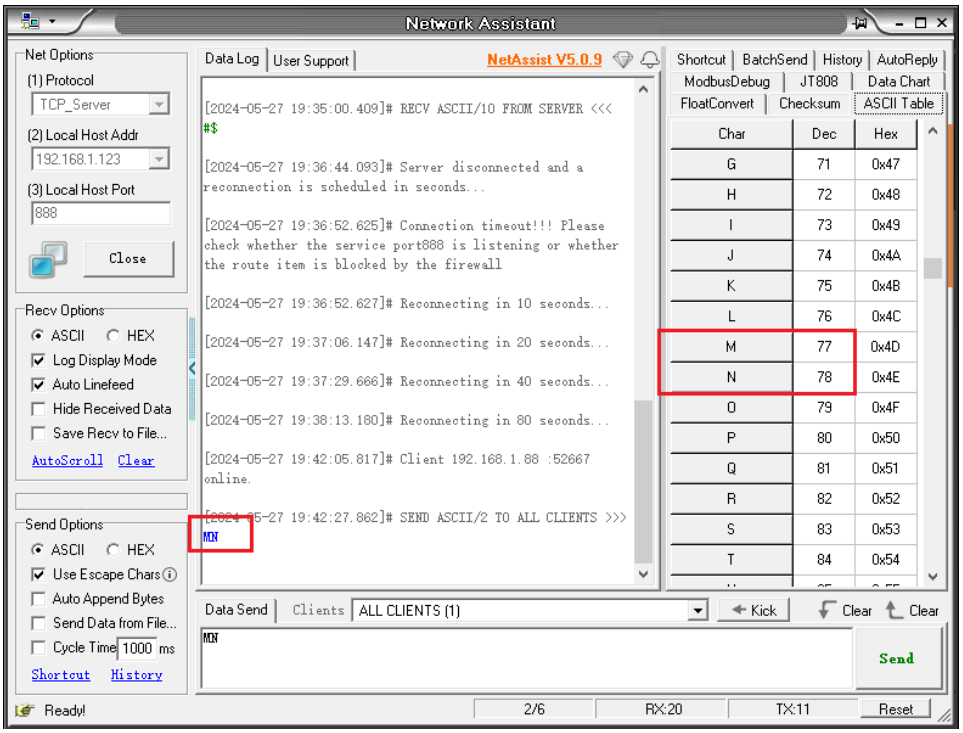
ST

To establish a connection, trigger the xEnable pint of TCP_Client. Ensure that the destination IP address and port number are consistent with those of the commissioning assistant.

```
TCP_Client_0(
    xEnable:= ,
    strIpAddrDst '192.168.1.  ':='192.168.1.123' ,
    uiPortDst 888 :=888 ,
    udiTimeOut 10000 :=10000 ,
    xDone=> ,
    xBusy=> ,
    xError=> ,
    dwErrorID=> ,
    xActive=> ,
    hConnection=> );
```



Receive data and use the commissioning assistant to send "M" and "N".



Trigger the xEnable pin of TCP_Receive to receive data. Ensure that uiSize is set to a valid value, and that the address pointed to by the pbyData pointer is the array to receive the data. Values of the first two elements are respectively "77" and "78", which correspond to the values sent by the commissioning assistant, indicating successful receiving.

Data	ARRAY [0..10] OF BYTE	
Data[0]	BYTE	77
Data[1]	BYTE	78
Data[2]	BYTE	0
Data[3]	BYTE	0
Data[4]	BYTE	0
Data[5]	BYTE	0

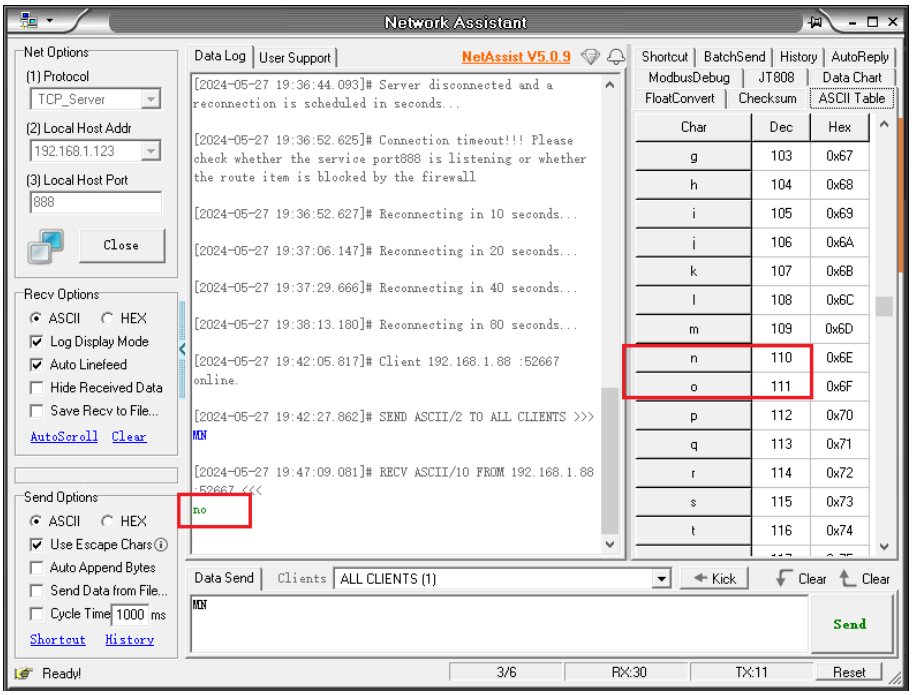
```
12
13  TCP_Receive_0(
14      xEnable:= ,
15      hConnection 3034441928 :=TCP_Client_0.hConnection 3034441928 ,
16      uiSize 10 :=10 ,
17      pbyData 16#B4D33D1C :=ADR(Data) ,
18      xDone=> ,
19      xBusy=> ,
20      xError=> ,
21      dwErrorID=> ,
22      xReady=> ,
23      uiCount=> );
```

Send data. In the array of the data sending area pointed to by the pbyData pin pointer of TCP_Send, the first two elements are respectively assigned with values "110" and "111", and uiSize is set to a proper value.

Data_0	ARRAY [0..10] OF BYTE	
Data_0[0]	BYTE	110
Data_0[1]	BYTE	111
Data_0[2]	BYTE	0
Data_0[3]	BYTE	0
Data_0[4]	BYTE	0
Data_0[5]	BYTE	0
Data_0[6]	BYTE	0
Data_0[7]	BYTE	0
Data_0[8]	BYTE	0

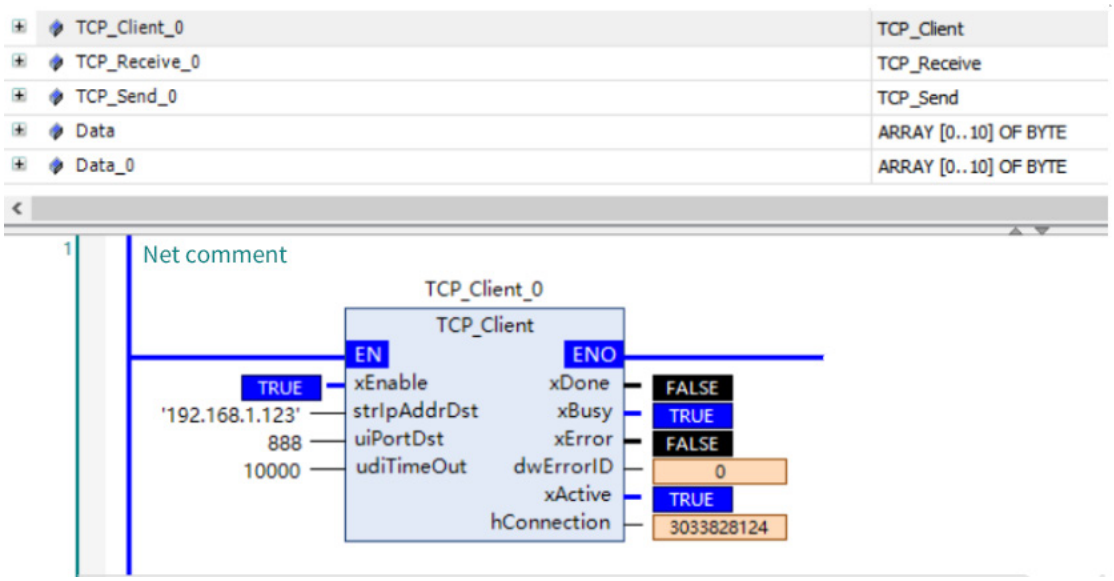
```
17      pbyData 16#B4D33D1C :=ADR(Data) ,
18      xDone=> ,
19      xBusy=> ,
20      xError=> ,
21      dwErrorID=> ,
22      xReady=> ,
23      uiCount=> );
24
25  TCP_Send_0(
26      xExecute:= ,
27      hConnection 3034441928 :=TCP_Client_0.hConnection 3034441928 ,
28      uiSize 10 :=10 ,
29      pbyData 16#B4D33D27 :=ADR(Data_0) ,
30      udiTimeOut 1000 :=1000 ,
31      xDone=> ,
32      xBusy=> ,
33      xError=> ,
34      dwErrorID=> );RETURN
```

The commissioning assistant receives "n" and "o", which match the sent data, indicating successful sending.

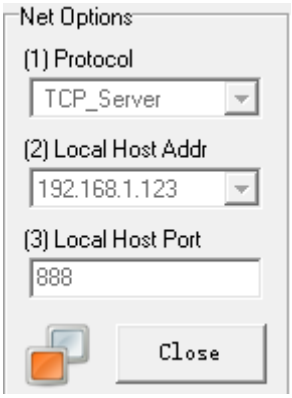


LD

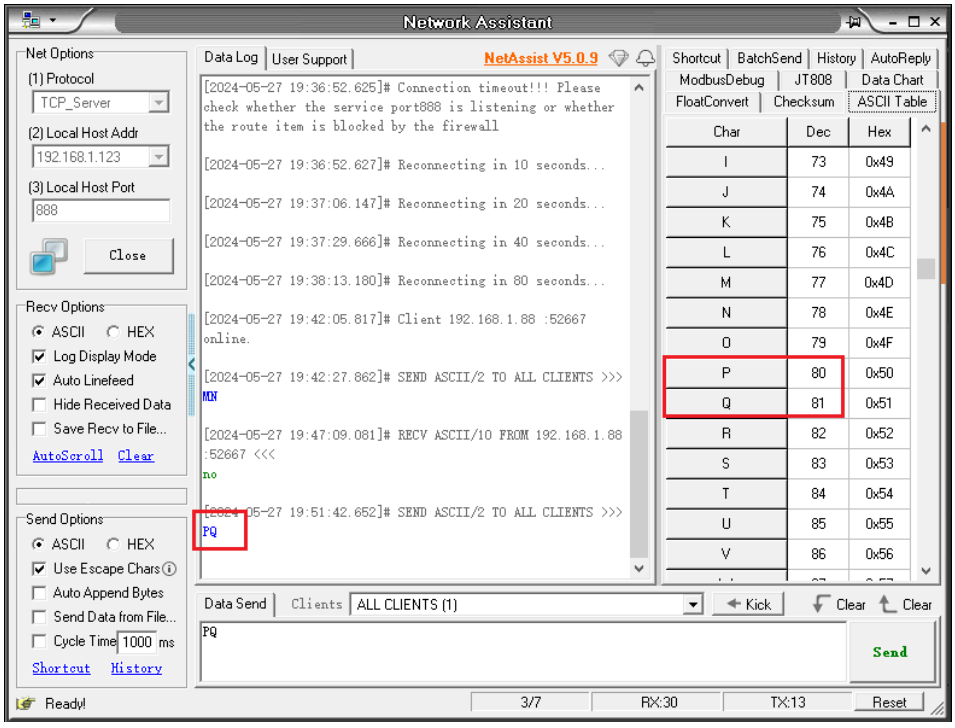
To establish a connection, trigger the xEnable pint of TCP_Client. Ensure that the destination IP address and port number are consistent with those of the commissioning assistant.



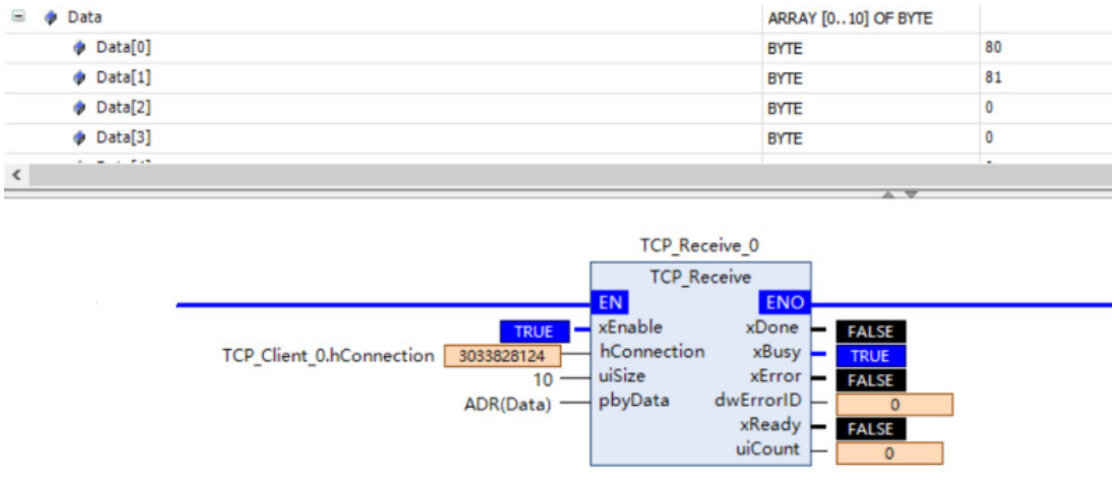
Commissioning assistant interface



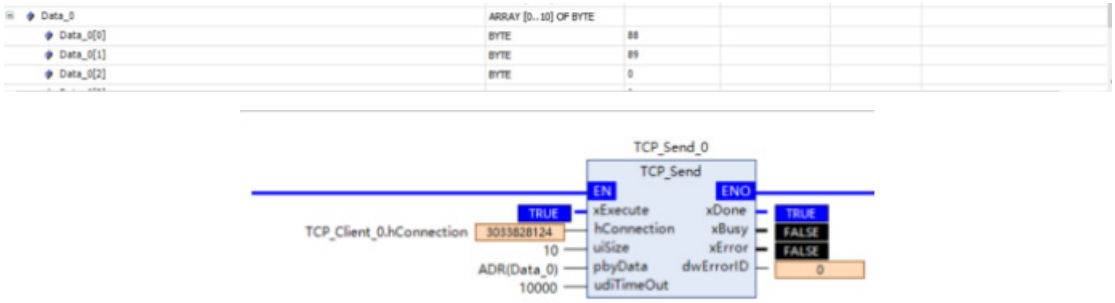
Receive data and use the commissioning assistant to send "P" and "Q".



Trigger the xEnable pin of TCP_Receive to receive data. Ensure that uiSize is set to a valid value, and that the address pointed to by the pbyData pointer is the array to receive the data. Values of the first two elements are respectively "80" and "81", which correspond to the values sent by the commissioning assistant, indicating successful receiving.



Send data. In the array of the data sending area pointed to by the pbyData pin pointer of TCP_Send, the first two elements are respectively assigned with values "88" and "89", and uiSize is set to a proper value.



The commissioning assistant receives "X" and "Y", which match the sent data, indicating successful sending.

■ Precautions

Disconnection detection function

When the TCP connection is on, the xActive pin of the TCP_Client and TCP_Connect function blocks is set to ON. When the TCP connection is off, the xActive pin is set to OFF. When the LAN cable is disconnected, the client program stops running, and the client is disconnected during running, set Enable of the TCP_Receive function block to ON to detect the disconnection status. In addition, after the client is reconnected to the server, set Enable to OFF and then ON to activate the xActive flag. To reconnect to the disconnected client, set the Enable pin of TCP_Connect to ON again for the server.

Error code

Error ID	Error code	Description
NO_ERROR	0	Proper running without error
FIRST_ERROR	6000	-
TIME_OUT	6001	Timeout
INVALID_ADDR	6002	Invalid IP address
INVALID_HANDLE	6003	Invalid handle
INVALID_DATAPOINTER	6004	Invalid data pointer
INVALID_DATASIZE	6005	Invalid data length
UDP_RECEIVE_ERROR	6006	UDP receiving error
UDP_SEND_ERROR	6007	UDP sending error
UDP_SEND_NOT_COMPLETE	6008	UDP sending not completed
UDP_OPEN_ERROR	6009	UDP opening error
UDP_CLOSE_ERROR	6010	UDP closing error
TCP_SEND_ERROR	6011	TCP sending error
TCP_RECEIVE_ERROR	6012	TCP receiving error
TCP_OPEN_ERROR	6013	TCP opening error
TCP_CONNECT_ERROR	6014	TCP connection error
TCP_CLOSE_ERROR	6015	TCP closing error
TCP_SERVER_ERROR	6016	TCP server error
WRONG_PARAMETER	6017	Incorrect parameter
ERROR_UNKNOWN	6018	Unknown error
TCP_NO_CONNECTION	6019	TCP not connected
IOCTL_ERROR	6020	IOCTL

Error ID	Error code	Description
FIRST_MF	6050	-
LAST_ERROR	6099	Last error

6. 2 Free UDP Communication Instructions

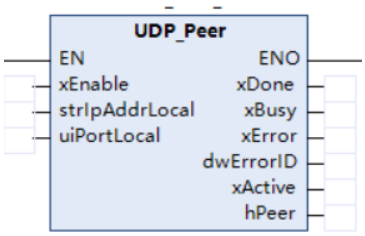
6.2. 1 Instruction List

Instruction Category	Name	FB/FC	Function
Free UDP communication instructions	UDP_Peer	FB	Activate UDP peer
	UDP_Receive	FB	Receive UDP data
	UDP_Send	FB	Send UDP data

6.2. 2 UDP_Peer

This instruction creates a UDP communication connection.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
UDP_Peer	Activate UDP peer	FB		UDP_Peer(xEnable:= , strIpAddrLocal:= , uiPortLocal:= , xDone=> , xBusy=> , xError=> , dwErrorID=> , xActive=> , hPeer=>);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
xEnable	Function block enable	BOOL	-	-	Input high level to enable the function block
strIpAddrLocal	Local IP address	STRING (80)	-	-	IP address of the local PLC communication network port

Input Variable	Name	Data Type	Value Range	Initial Value	Description
uiPortLocal	Communication port number	UINT	-	-	UDP communication port number

Output variables

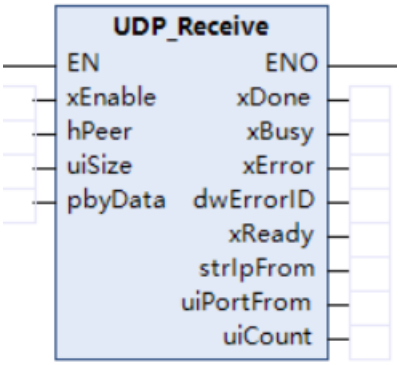
Output Variable	Name	Data Type	Value Range	Initial Value	Description
xDone	Ending signal	BOOL	-	-	Communication ending flag
xBusy	Busy flag	BOOL	-	-	Busy state
xError	Error flag	BOOL	-	-	Error state
dwErrorID	Error code	DWORD	-	-	Error code
xActive	Communication success flag	BOOL	-	-	Flag of successful UDP communication creation
hpeer	Communication handle	_XWORD	-	-	UDP communication handle

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
xEnable	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
strIpAddrLocal	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√
uiPortLocal	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-
xDone	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
xBusy	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
xError	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
dwErrorID	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
xActive	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
hpeer	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

6.2. 3 UDP_Receive

This instruction receives UDP communication data.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
UDP_Receive	Receive UDP data	FB		UDP_Receive(xEnable:= , hPeer:= , uiSize:= , pbyData:= , xDone=> , xBusy=> , xError=> , dwErrorID=> , xReady=> , strIpFrom=> , uiPortFrom=> , uiCount=>);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
xEnable	Function block enable	BOOL	-	-	Input high level to enable the function block
hPeer	Communication handle	_XWORD	-	-	UDP communication handle
uiSize	Data size	UINT	-	-	Size of the area for receiving data
PbyData	Pointer to the receiving area	POINTER_ TO_BYTE	-	-	Pointer to the first byte of the receiving area

Output variables

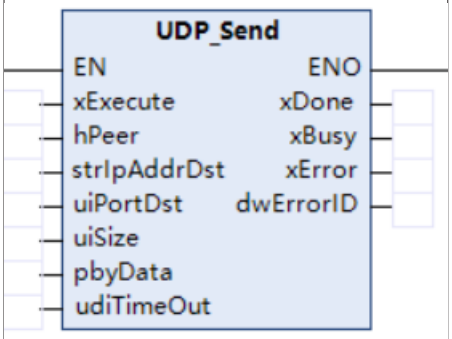
Output Variable	Name	Data Type	Value Range	Initial Value	Description
xDone	Ending signal	BOOL	-	-	Data receiving ending signal
xBusy	Busy flag	BOOL	-	-	Busy state
xError	Error flag	BOOL	-	-	Error state
dwErrorID	Error code	DWORD	-	-	Error code
xReady	Receiving success flag	BOOL	-	-	Flag of setting a scan period when data is read from the buffer
strIpFrom	Source IP address of the current data packet	STRING(80)	-	-	Source IP address of the current data packet
uiPortFrom	Source port of the current data packet	UINT	-	-	Source port of the current data packet
uiCount	Data volume	_UXINT	-	-	Volume of data read from the buffer

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
xEnable	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
hPeer	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√
uiSize	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-
PbyData	POINTER_TO_BYTE																			
xDone	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
xBusy	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
xError	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
dwErrorID	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
xReady	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
strIpFrom	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√
uiPortFrom	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-
uiCount	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-

6.2. 4 UDP_Send

This instruction sends UDP communication data.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
UDP_Send	Send UDP data	FB		UDP_Send(xExecute:= , hPeer:= , strIpAddrDst:= , uiPortDst:= , uiSize:= , pbyData:= , udiTimeOut:= , xDone=> , xBusy=> , xError=> , dwErrorID=>);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
xExecute	Function block enable	BOOL	-	-	Input high level to enable the function block
hPeer	Communication handle	_XWORD	-	-	UDP communication handle
strIpAddrDst	Destination IP address	STRING (80)	-	-	Destination IP address to which data is sent

Input Variable	Name	Data Type	Value Range	Initial Value	Description
uiPortDst	Destination port	UINT	-	-	Destination port to which data is sent
uiSize	Size of the sending area	UINT	-	-	Size of the sending area, in the unit of bytes
PbyData	Pointer to the sending area	POINTER_TO_BYTE	-	-	Head address of the pointer to the sending area
UdiTimeOut	Timeout period	UDINT	-	-	Timeout period for data sending The unit is us and the default value is 500 ms.

Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
xDone	Ending signal	BOOL	-	-	Done signal
xBusy	Busy flag	BOOL	-	-	Busy state
xError	Error flag	BOOL	-	-	Error state
dwErrorID	Error code	DWORD	-	-	Error code

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
xExecute	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
hPeer	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
strIpAddrDst	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√
uiPortDst	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-
uiSize	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-
PbyData	POINTER_TO_BYTE																			
UdiTimeOut	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-
xDone	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
xBusy	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
xError	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
dwErrorID	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

6.2. 5 UDP Communication Instruction Examples

UDP communication for AM600 and the commissioning assistant

For example, IP address of the commissioning assistant is 192.168.1.123 and the port number is 888.

ST

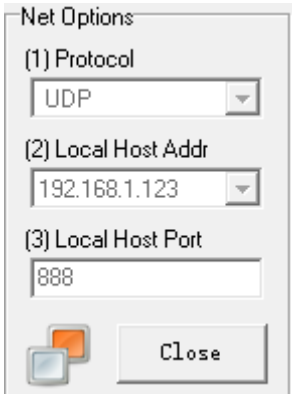
To establish a connection, trigger xEnable, set the strIpAddrLocal pin of the UDP_Peer instruction to the IP address of the local PLC communication network port, and set uiPortLocal to the local port number.

Trigger connection to the commissioning assistant. Ensure that the IP address and port number are

consistent. For example, the port number is 888.

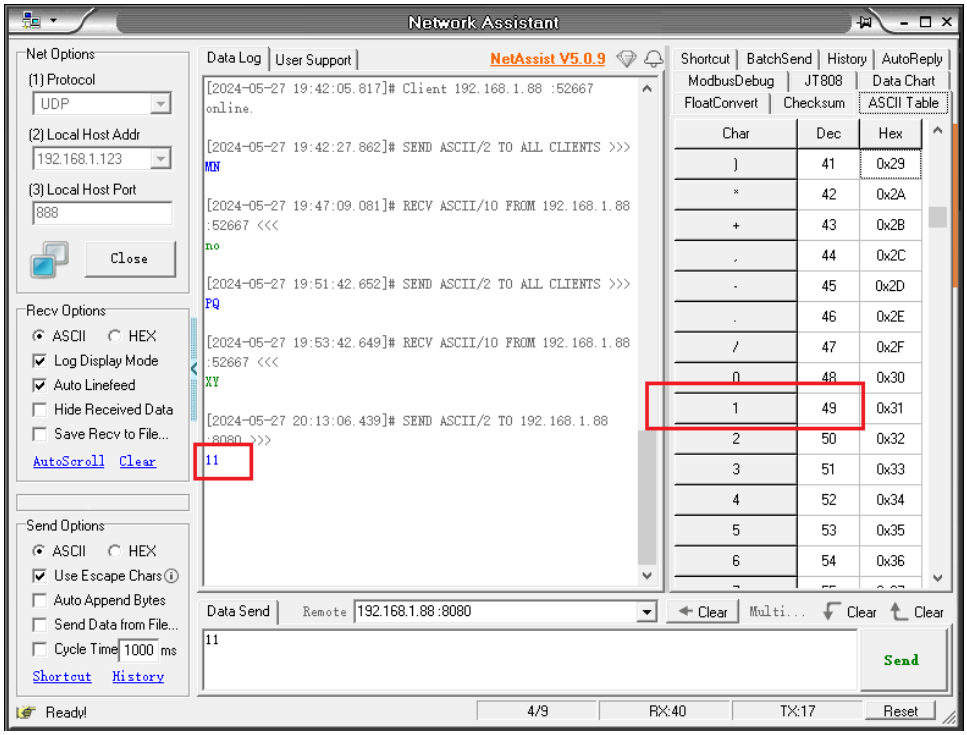
Expression	Type	Value
UDP_Peer_0	UDP_Peer	
UDP_Receive_0	UDP_Receive	
UDP_Send_0	UDP_Send	
Data	ARRAY [0..10] OF BYTE	
Data_0	ARRAY [0..10] OF BYTE	

```
1  UDP_Peer_0(
2      xEnable:= ,
3      strIpAddrLocal[192.168.1.]:='192.168.1.123',
4      uiPortLocal[888]:=888 ,
5      xDone=> ,
6      xBusy=> ,
7      xError=> ,
8      dwErrorID=> ,
9      xActive=> ,
10     hPeer=> );
```



Receive data

Use the commissioning assistant to send data "1" and "1".



The corresponding decimal values are "49" and "49".

Trigger UDP_Receive and ensure that the received data is in the address pointed to by the pbyData pointer. As shown in the following figure, the two values in the array pointed to by the pointer are respectively "49" and "49", indicating successful receiving.

Expression	Type	Value	Prepared Value
Data	ARRAY [0..10] OF BYTE		
Data[0]	BYTE	49	
Data[1]	BYTE	49	
Data[2]	BYTE	0	
Data[3]	BYTE	0	
Data[4]	BYTE	0	
Data[5]	BYTE	0	

```
12  UDP_Receive_0(
13      xEnable:= ,
14      hPeer 3033830728 :=UDP_Peer_0.hPeer 3033830728 ,
15      uiSize 10 :=10 ,
16      pbyData 16#B4D33A68 :=ADR(Data) ,
17      xDone=> ,
18      xBusy=> ,
19      xError=> ,
20      dwErrorID=> ,
21      xReady=> ,
22      strIpFrom=> ,
23      uiPortFrom=> ,
24      uiCount=> );
```

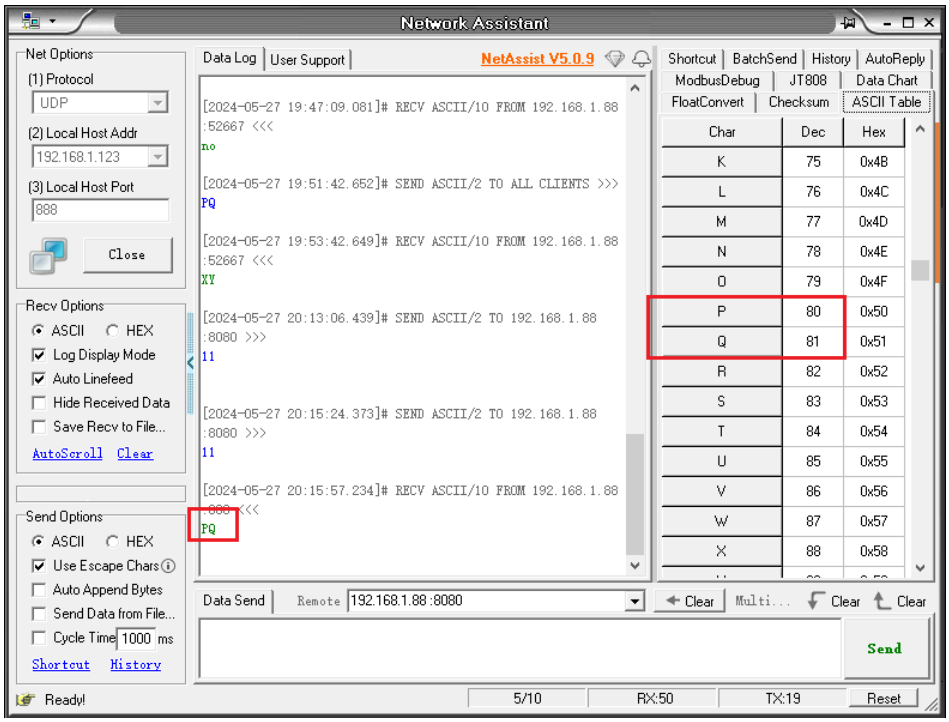
Send data

Trigger UDP_Send and ensure that the sent data is in the address pointed to by the pbyData pointer. As shown in the following figure, the two values in the array pointed to by the pointer are respectively "80" and "81".

Data_0	ARRAY [0..10] OF BYTE	
Data_0[0]	BYTE	80
Data_0[1]	BYTE	81
Data_0[2]	BYTE	0
Data_0[3]	BYTE	0
Data_0[4]	BYTE	0

```
12  UDP_Receive_0(
13      xEnable:= ,
14      hPeer 3033830728 :=UDP_Peer_0.hPeer 3033830728 ,
15      uiSize 10 :=10 ,
16      pbyData 16#B4D33A68 :=ADR(Data) ,
17      xDone=> ,
18      xBusy=> ,
19      xError=> ,
20      dwErrorID=> ,
21      xReady=> ,
22      strIpFrom=> ,
23      uiPortFrom=> ,
24      uiCount=> );
25
26  UDP_Send_0(
27      xExecute:= ,
28      hPeer 3033830728 :=UDP_Peer_0.hPeer 3033830728 ,
29      strIpAddrDst '192.168.1. ' :='192.168.1.123' ,
30      uiPortDst 888 :=888 ,
31      uiSize 10 :=10 ,
32      pbyData 16#B4D33A73 :=ADR(Data_0) ,
33      udiTimeOut 10000 :=10000 ,
34      xDone=> ,
35      xBusy=> ,
36      xError=> ,
37      dwErrorID=> );RETURN
```

Check the commissioning assistant. The received data "P" and "Q" corresponds to "80" and "81" respectively, indicating successful sending.

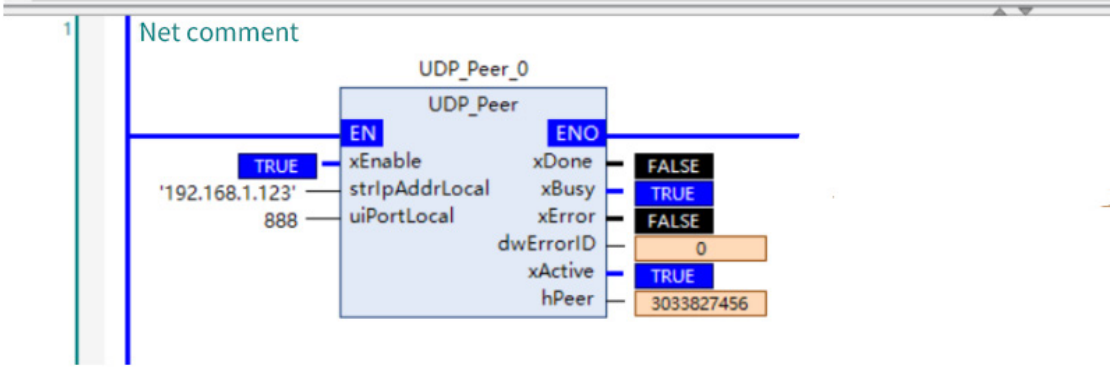


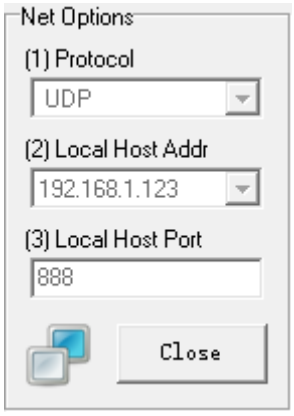
LD

To establish a connection, trigger xEnable, set the strIpAddrLocal pin of the UDP_Peer instruction to the IP address of the local PLC communication network port, and set uiPortLocal to the local port number.

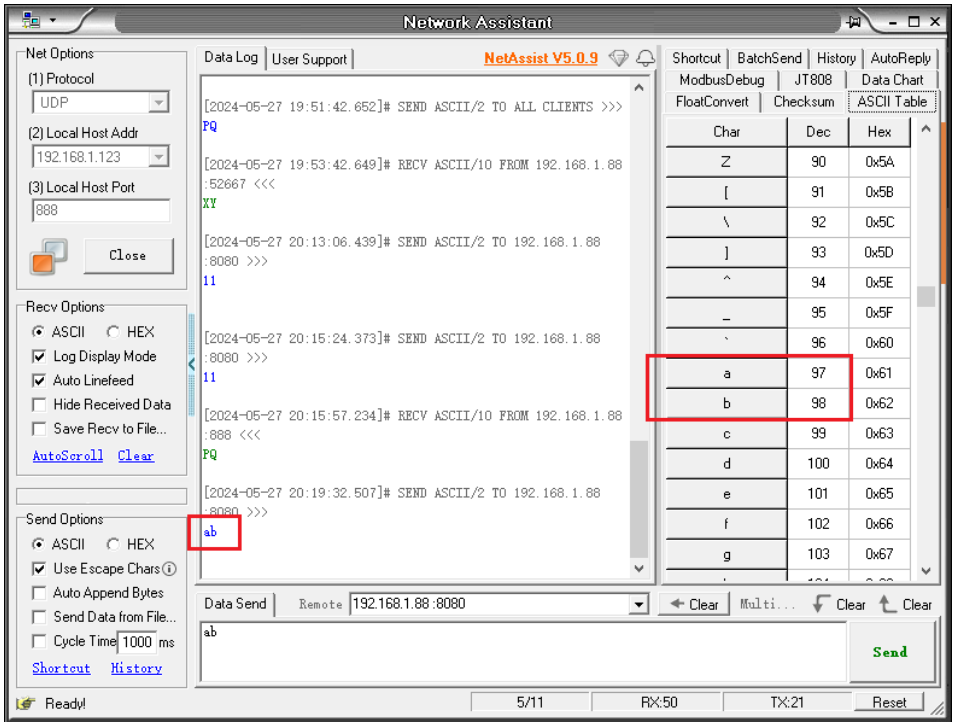
Trigger connection to the commissioning assistant. Ensure that the IP address and port number are consistent. For example, the port number is 888.

Expression	Type
UDP_Send_0	UDP_Send
UDP_Receive_0	UDP_Receive
UDP_Peer_0	UDP_Peer
Data	ARRAY [0..10] OF BYTE
Data_0	ARRAY [0..10] OF BYTE



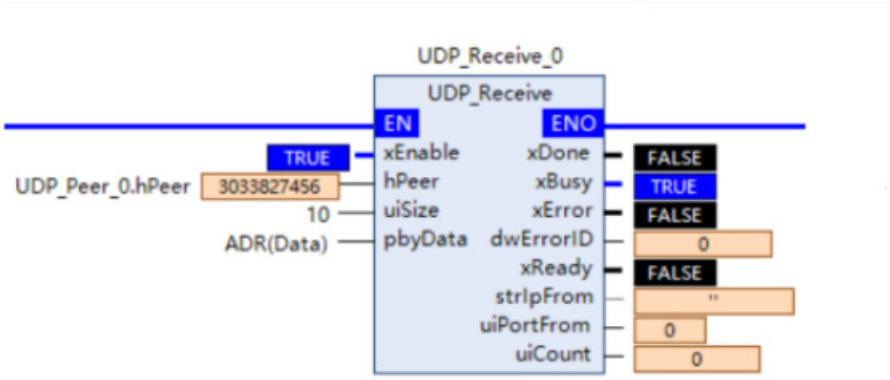


Receive data and use the commissioning assistant to send "a" and "b".



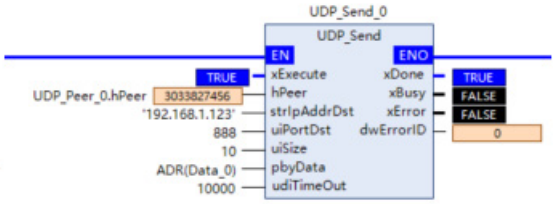
The two values in the array pointed to by the pbyData pin pointer of UDP_Receive instruction are respectively "97" and "98", indicating successful receiving.



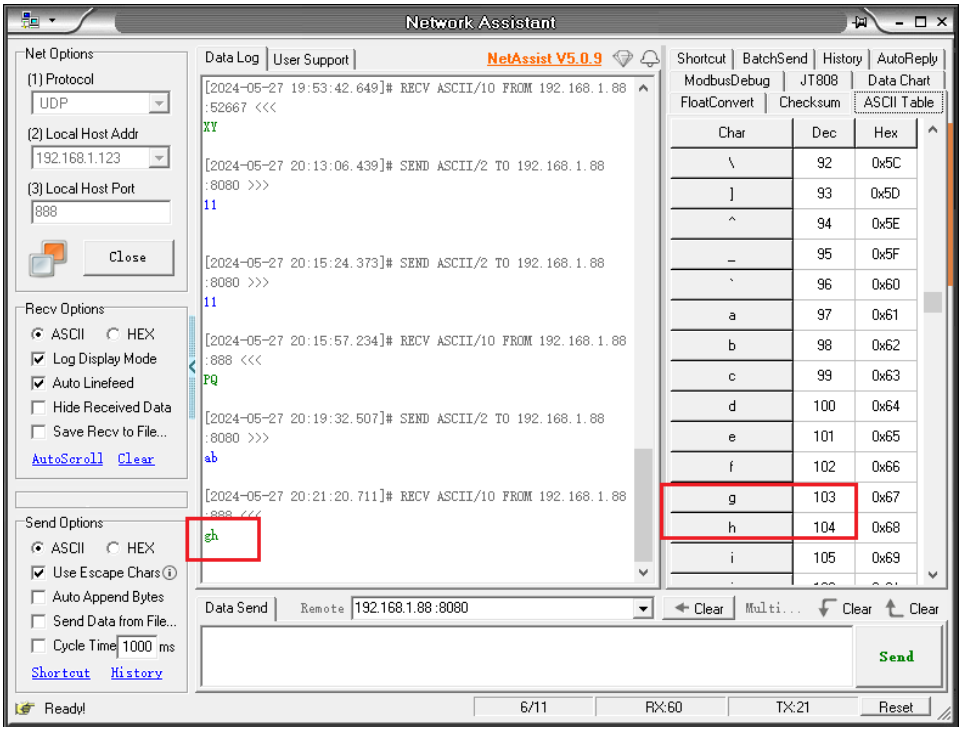


Send data, write "103" and "104" to the first two array elements in the sending buffer of UDP_Send, and set uiSize for the sent bytes.

Expression	Type	Value	Prepared Value	Address	Comment
Data	ARRAY [0..10] OF BYTE				
Data_0	ARRAY [0..10] OF BYTE				
Data_0[0]	BYTE	103			
Data_0[1]	BYTE	104			
Data_0[2]	BYTE	0			
Data_0[3]	BYTE	0			



Check the commissioning assistant. The values obtained are "g" and "h", which respectively correspond to values "103" and "104", indicating successful sending.



■ Precautions

Error code

Error ID	Error code	Description
NO_ERROR	0	Proper running without error
FIRST_ERROR	6000	-
TIME_OUT	6001	Timeout
INVALID_ADDR	6002	Invalid IP address
INVALID_HANDLE	6003	Invalid handle
INVALID_DATAPOINTER	6004	Invalid data pointer
INVALID_DATASIZE	6005	Invalid data length
UDP_RECEIVE_ERROR	6006	UDP receiving error
UDP_SEND_ERROR	6007	UDP sending error
UDP_SEND_NOT_COMPLETE	6008	UDP sending not completed
UDP_OPEN_ERROR	6009	UDP opening error
UDP_CLOSE_ERROR	6010	UDP closing error
TCP_SEND_ERROR	6011	TCP sending error
TCP_RECEIVE_ERROR	6012	TCP receiving error
TCP_OPEN_ERROR	6013	TCP opening error
TCP_CONNECT_ERROR	6014	TCP connection error
TCP_CLOSE_ERROR	6015	TCP closing error
TCP_SERVER_ERROR	6016	TCP server error
WRONG_PARAMETER	6017	Incorrect parameter
ERROR_UNKNOWN	6018	Unknown error
TCP_NO_CONNECTION	6019	TCP not connected
IOCTL_ERROR	6020	IOCTL
FIRST_MF	6050	-
LAST_ERROR	6099	Last error

6. 3 CANopen Communication Instructions

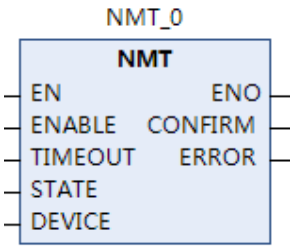
6.3. 1 Instruction List

Instruction Category	Name	FB/FC	Function
CANopen communication instructions	NMT	FB	NMT services
	RECV_EMCY	FB	Receive emergency object from any device
	RECV_EMCY_DEV	FB	Receive emergency object from input device
	GET_LOCAL_NODE_ID	FB	Get the CANopen NodeID of the local device
	GET_CANOPEN_KERNEL_STATE	FB	Get the current state of the CANopen kernel
	GET_STATE	FB	Get the CANopen state of device
	SDO_READ	FB	Read specific object from object dictionary of device
	SDO_READ4	FB	Read specific object up to 4 bytes from object dictionary of device
	SDO_WRITE	FB	Write specific object in object dictionary of device
	SDO_WRITE4	FB	Write specific object up to 4 bytes in object dictionary of device
	CANOPEN_COUNT	FB	CANopen count
	GET_MST_STATISTICS	FB	Get MST statistics

6.3. 2 NMT

This instruction manages the device network status.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
NMT	NMT services	FB		NMT_0(ENABLE :=, TIMEOUT :=, STATE :=, DEVICE :=, CONFIRM =>, ERROR =>,);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
ENABLE	Enable	BOOL	Depends on data types	-	(Triggered by level) Execution of the function block switch
TIMEOUT	Maximum execution time	UDINT	Depends on data types	-	Maximum execution time, in the unit of ms If the execution times out before the response is received, the execution aborts due to timeout. 0 (default): Timeout disabled 1 to 65535: Timeout value
STATE	NMT status to request	TRANSITION_STATE	Depends on data types	-	NMT status conversion to request
DEVICE	Destination device node ID	DEVICE	Depends on data types	-	CANopen destination device node ID 0: All NMT slave devices 1 to 127: Destination device node ID

Output variables

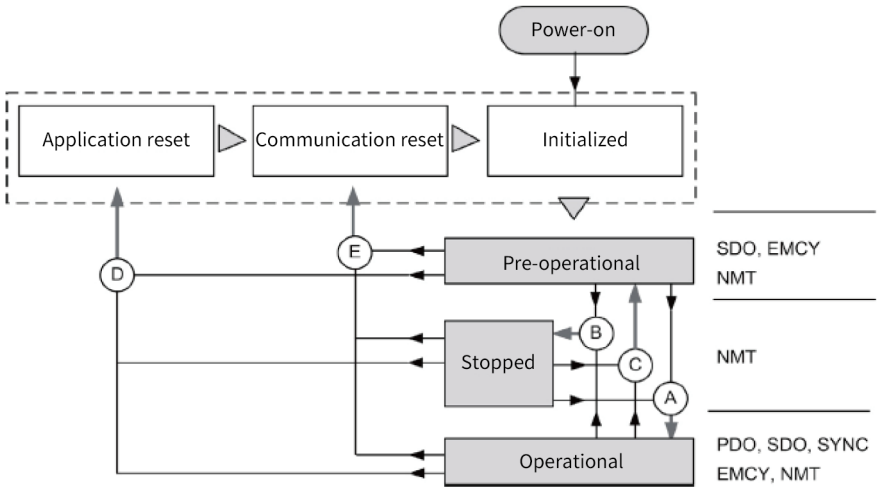
Output Variable	Name	Data Type	Value Range	Initial Value	Description
CONFIRM	Completion flag of the function block	BOOL	Depends on data types	-	Completion flag of the function block
ERROR	Error information of function block	CANOPEN_KERNEL_ERROR	Depends on data types	-	Error information of function block

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
ENABLE	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
TIMEOUT	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-
STATE	TRANSITION_STATE																			
DEVICE	DEVICE																			
CONFIRM	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
ERROR	CANOPEN_KERNEL_ERROR																			

■ Function

The NMT state machine defines the initialization and status of the NMT slave in main operations.

The following figure shows the NMT status, associated available communication objects (PDO, SDO, SYNC, EMCY, and NMT), and five types of status conversions (A to E).

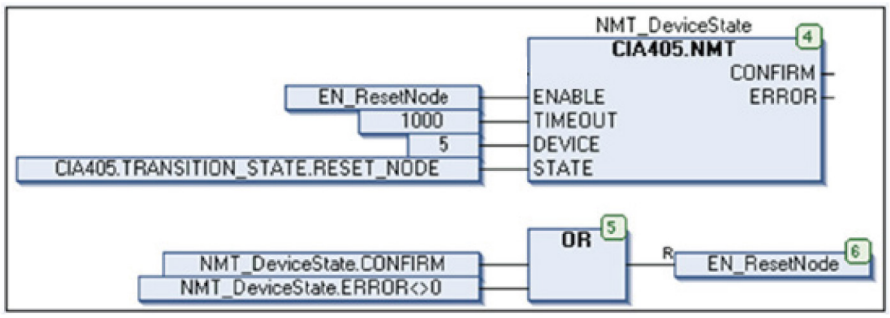


The data type of STATE is enumerated type TRANSITION_STATE. The meanings of the enumerators are as follows.

Enumerator	Meaning
STOP_REMOTE_NODE	Switches to the stop state. (Conversion B)
START_REMOTE_NODE	Switches to the normal operating state. (Conversion A)
RESET_NODE	Switches to the application reset state. In this state, the saved data of the device configuration file is loaded, and the communication reset state automatically switches to the pre-operational state. (Conversion D)
RESET_COMMUNICATION	Switches to the communication reset state. In this state, the saved data of the communication configuration file is loaded, and the state automatically switches to the pre-operational state. (Conversion E)
ENTER_PRE_OPERATIONAL	Switches to the pre-operational state. (Conversion C)
ALL_EXCEPT_NMT_AND_SENDER	Not implemented (invalid parameter)

■ Program example

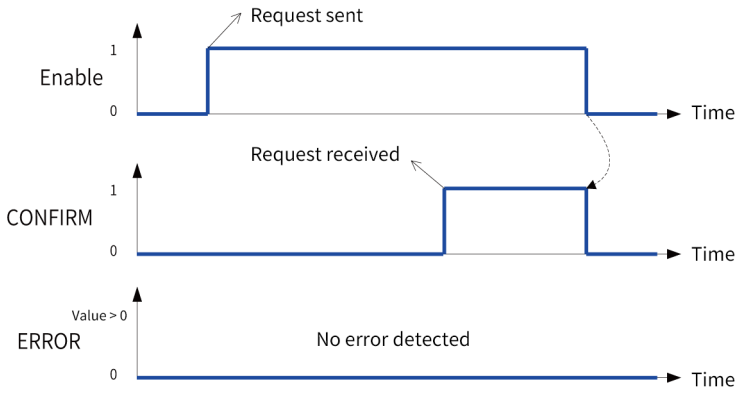
The following example demonstrates how to send a "reset node" command to CANopen node 5 that is connected to the first CAN bus interface, with a timeout period of 1s (1000 ms). The command is sent when the boolean variable EN_ResetNode is set to TRUE, either by the user online or by the application. Once the execution is successfully completed (CONFIRM is TRUE) or an error is detected (ERROR is not 0), the EN_ResetNode command is reset to FALSE.



■ Timing diagram

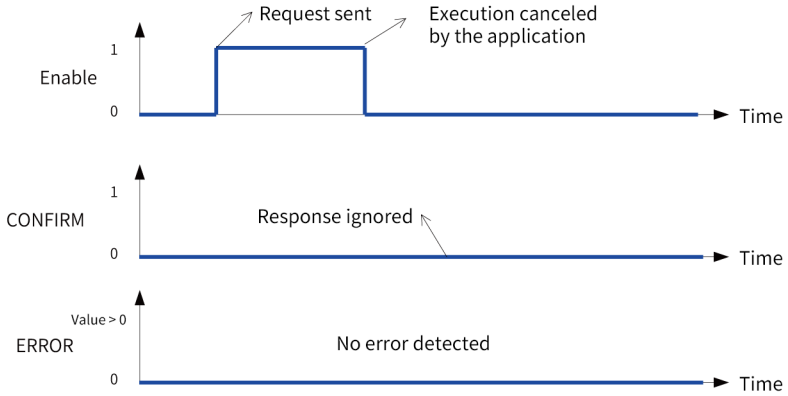
Once a response is generated for the current request without detecting any errors, CONFIRM is

immediately set to TRUE and remains TRUE (provided that the function block is used when ENABLE is set to TRUE). If the function block is used when ENABLE is reset to FALSE, CONFIRM is reset to FALSE, and the function block can start a new execution.



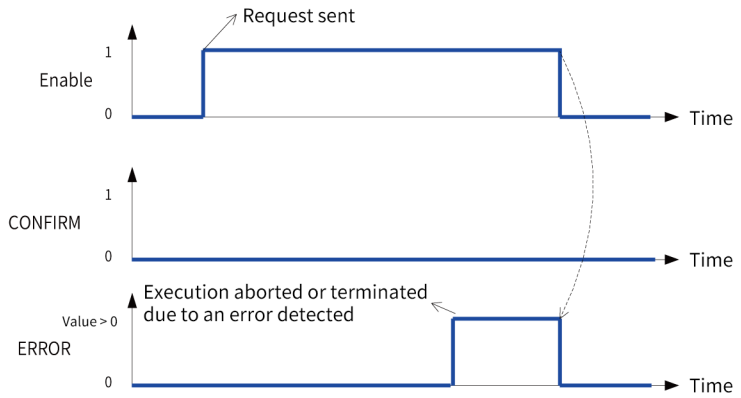
Execution is canceled by the application.

Before the current execution is completed, if the function block is used when ENABLE is reset to FALSE, the execution of the function block is canceled. It is possible that responses have been generated for canceled requests or will arrive later, but these responses will be ignored at this point.



Execution is aborted or terminated due to an error detected.

Once the current execution is aborted due to receiving an SDO abort message or terminated due to an error detected, ERROR is set to a value other than 0. For more information about the detected error codes, see the following table. If the function block is used when ENABLE is reset to FALSE, ERROR is reset to 0, and the function block can start a new execution.



■ Error description

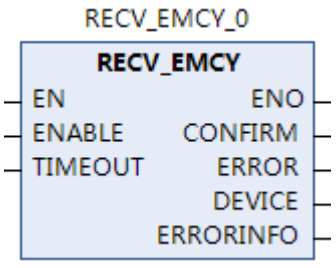
The data type of ERROR is enumerated type CANOPEN_KERNEL_ERROR. The meanings of the enumerators are as follows.

Error Code	Error	Description
0	CANOPEN_KERNEL_NO_ERROR	CANopen kernel does not detect any errors.
1	CANOPEN_KERNEL_OTHER_ERROR	If the value of ERROR is 01 in hexadecimal format, the error is detected. If the function block involves any other error codes, the output contains more detailed information. Example: CIA405.SDO_READ and CIA405.SDO_WRITE function blocks: The output of ERRORINFO contains the content of the SDO abort message. CIA405.RECV_EMCY and CIA405.RECV_EMCY_DEV function blocks: The output of ERRORINFO contains the content of the received EMCY message. If the output of ERRORINFO contains the EMCY message, the output of ERROR is 1 and the output of CONFIRM is 0. CIA405.GET_CANOPEN_KERNEL_STATE function block: The value 01 of the hexadecimal format will not be provided because no other error codes are output.
2	CANOPEN_KERNEL_DATA_OVERFLOW	The sending buffer or reception butter of the CANopen object overflows.
3	CANOPEN_KERNEL_TIMEOUT	Execution of the function block times out.
16	CANOPEN_KERNEL_CANBUS_OFF	The CANopen node is disconnected from the CAN bus.
17	CANOPEN_KERNEL_CAN_ERROR_PASSIVE	The CANopen node is in the error passive state: The node communicates properly but cannot send active error flags when an error is detected.
33	CANOPEN_INTERNAL_FB_ERROR	Manufacturer-specific error code. Note: Values 21 (hexadecimal) to FF (hexadecimal) are manufacturer-specific values.

6.3. 3 RECV_EMCY

This instruction scans EMCY messages.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
RECV_EMCY	Receive emergency object from any device	FB		RECV_EMCY_0(ENABLE :=, TIMEOUT :=, CONFIRM =>, ERROR =>, DEVICE =>, ERRORINFO =>,);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
ENABLE	Enable	BOOL	Depends on data types	-	(Triggered by level) Execution of the function block switch
TIMEOUT	Maximum execution time	UDINT	Depends on data types	-	Maximum execution time, in the unit of ms If the execution times out before the response is received, the execution aborts due to timeout. 0 (default): Timeout disabled 1 to 65535: Timeout value

Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
CONFIRM	Completion flag of the function block	BOOL	Depends on data types	-	Completion flag of the function block
ERROR	Error information of function block	CANOPEN_KERNEL_ERROR	Depends on data types	-	Error information of function block
DEVICE	Device node ID	DEVICE	Depends on data types	-	Device node ID When the EMCY message is received, the value is the node ID of the sending device. When no EMCY message is received, the value is 0.
ERRORINFO	EMCY information	HC_tagEmcyError	Depends on data types	-	EMCY information

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
ENABLE	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
TIMEOUT	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-
CONFIRM	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
ERROR	CANOPEN_KERNEL_ERROR																			
DEVICE	DEVICE																			
ERRORINFO	HC_tagEmcyError																			

■ Function

This function block scans the EMCY messages in all existing CANopen devices in a loop and returns the found EMCY messages.

After the function block is enabled, EMCY message scan starts from the previous scan stop point:

If an EMCY message is found, the scan ends and the function block returns the EMCY message and the associated device node ID.

If an EMCY message without errors is found but the message is not the previous no-error message, the scan ends and the function block returns an EMCY message with the value of 0 and the associated device node ID.

If there are no EMCY messages and no new no-error EMCY messages are found during the complete scan process, the function block returns an EMCY message with the value of 0 and a device node ID of 0.

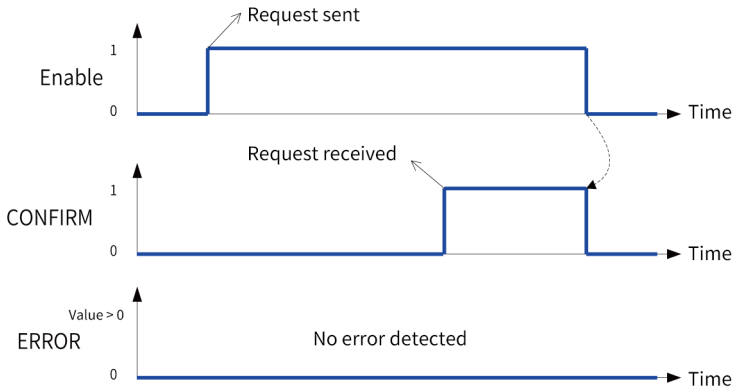
The data type of ERRORINFO is a structure HC_tagEmcyError. The meanings of the structure members are as follows.

Member	Data Type	Meaning
EMCY_ERROR_CODE	WORD	Error code of the EMCY message
ERROR_REGISTER	BYTE	Error register of the EMCY message (bit field)
ERROR_FIELD	ARRAY[0..4] OF BYTE	Manufacturer-specific error field of the EMCY message

■ Timing diagram

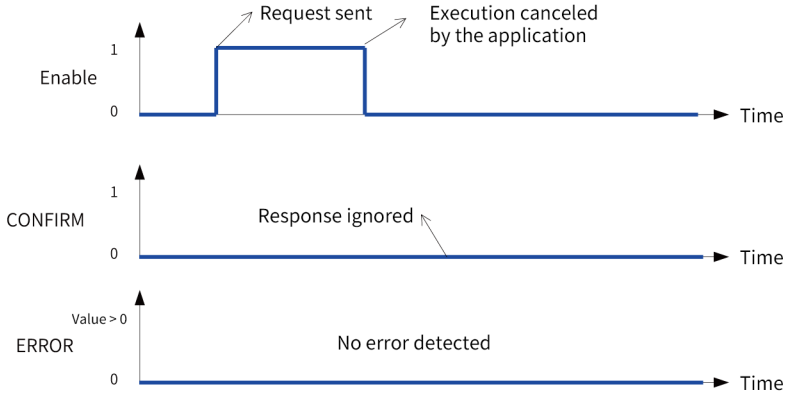
Once a response is generated for the current request without detecting any errors, CONFIRM is

immediately set to TRUE and remains TRUE (provided that the function block is used when ENABLE is set to TRUE). If the function block is used when ENABLE is reset to FALSE, CONFIRM is reset to FALSE, and the function block can start a new execution.



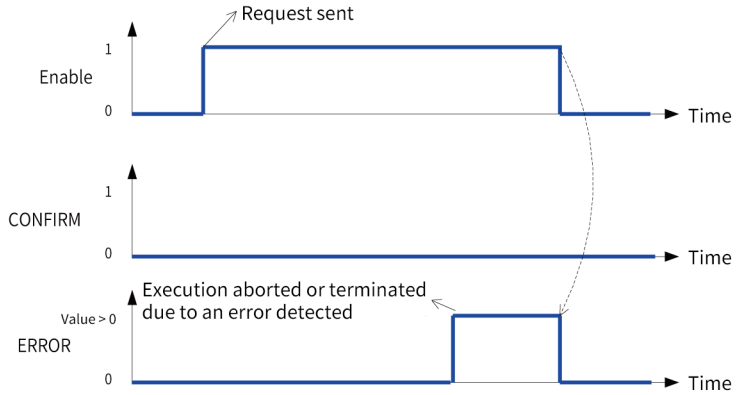
Execution is canceled by the application.

Before the current execution is completed, if the function block is used when ENABLE is reset to FALSE, the execution of the function block is canceled. It is possible that responses have been generated for canceled requests or will arrive later, but these responses will be ignored at this point.



Execution is aborted or terminated due to an error detected.

Once the current execution is aborted due to receiving an SDO abort message or terminated due to an error detected, ERROR is set to a value other than 0. For more information about the detected error codes, see the following table. If the function block is used when ENABLE is reset to FALSE, ERROR is reset to 0, and the function block can start a new execution.



■ Error description

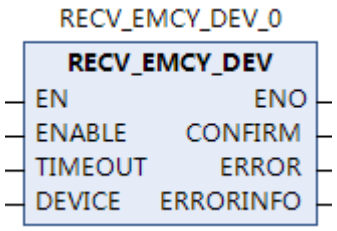
The data type of ERROR is enumerated type CANOPEN_KERNEL_ERROR. The meanings of the enumerators are as follows.

Error Code	Error	Description
0	CANOPEN_KERNEL_NO_ERROR	CANopen kernel does not detect any errors.
1	CANOPEN_KERNEL_OTHER_ERROR	If the value of ERROR is 01 in hexadecimal format, the error is detected. If the function block involves any other error codes, the output contains more detailed information. Example: CIA405.SDO_READ and CIA405.SDO_WRITE function blocks: The output of ERRORINFO contains the content of the SDO abort message. CIA405.RECV_EMCY and CIA405.RECV_EMCY_DEV function blocks: The output of ERRORINFO contains the content of the received EMCY message. If the output of ERRORINFO contains the EMCY message, the output of ERROR is 1 and the output of CONFIRM is 0. CIA405.GET_CANOPEN_KERNEL_STATE function block: The value 01 of the hexadecimal format will not be provided because no other error codes are output.
2	CANOPEN_KERNEL_DATA_ OVERFLOW	The sending buffer or reception butter of the CANopen object overflows.
3	CANOPEN_KERNEL_TIMEOUT	Execution of the function block times out.
16	CANOPEN_KERNEL_CANBUS_OFF	The CANopen node is disconnected from the CAN bus.
17	CANOPEN_KERNEL_CAN_ERROR_ PASSIVE	The CANopen node is in the error passive state: The node communicates properly but cannot send active error flags when an error is detected.
33	CANOPEN_INTERNAL_FB_ERROR	Manufacturer-specific error code. Note: Values 21 (hexadecimal) to FF (hexadecimal) are manufacturer-specific values.

6.3. 4 RECV_EMCY_DEV

This instruction receives the EMCY message of the device.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
RECV_EMCY_DEV	Receive emergency object from input device	FB		RECV_EMCY_DEV_0(ENABLE :=, TIMEOUT :=, DEVICE :=, CONFIRM =>, ERROR =>, ERRORINFO =>,);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
ENABLE	Enable	BOOL	Depends on data types	-	(Triggered by level) Execution of the function block switch
TIMEOUT	Maximum execution time	UDINT	Depends on data types	-	Maximum execution time, in the unit of ms If the execution times out before the response is received, the execution aborts due to timeout. 0 (default): Timeout disabled 1 to 65535: Timeout value

Input Variable	Name	Data Type	Value Range	Initial Value	Description
DEVICE	Destination device node ID	DEVICE	Depends on data types	-	CANopen destination device node ID 0: All NMT slave devices 1 to 127: Destination device node ID

Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
CONFIRM	Completion flag of the function block	BOOL	Depends on data types	-	Completion flag of the function block
ERROR	Error information of function block	CANOPEN_KERNEL_ERROR	Depends on data types	-	Error information of function block
ERRORINFO	EMCY information	HC_tagEmcyError	Depends on data types	-	EMCY information

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
ENABLE	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
TIMEOUT	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-
DEVICE	DEVICE																			
CONFIRM	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
ERROR	CANOPEN_KERNEL_ERROR																			
ERRORINFO	HC_tagEmcyError																			

■ Function

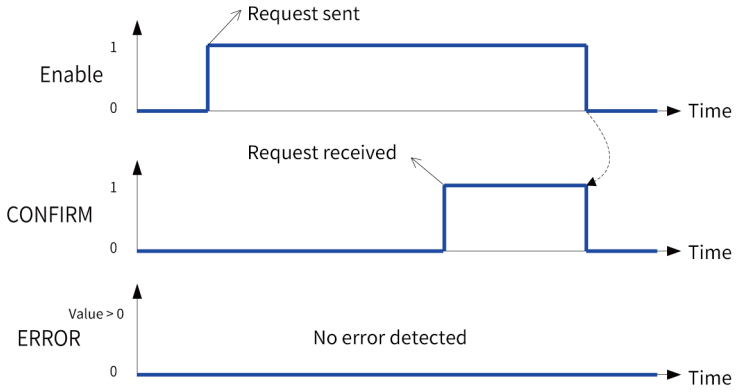
This function block returns the last EMCY message received from the specified CANopen device.

The data type of ERRORINFO is a structure HC_tagEmcyError. The meanings of the structure members are as follows.

Member	Data Type	Meaning
EMCY_ERROR_CODE	WORD	Error code of the EMCY message
ERROR_REGISTER	BYTE	Error register of the EMCY message (bit field)
ERROR_FIELD	ARRAY[0..4] OF BYTE	Manufacturer-specific error field of the EMCY message

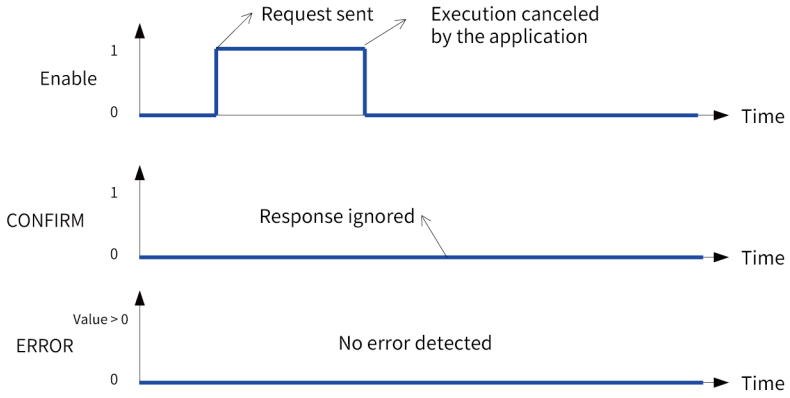
■ Timing diagram

Once a response is generated for the current request without detecting any errors, CONFIRM is immediately set to TRUE and remains TRUE (provided that the function block is used when ENABLE is set to TRUE). If the function block is used when ENABLE is reset to FALSE, CONFIRM is reset to FALSE, and the function block can start a new execution.



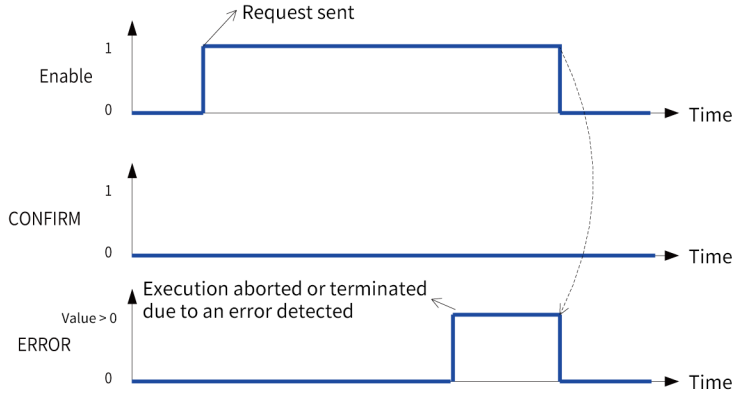
Execution is canceled by the application.

Before the current execution is completed, if the function block is used when ENABLE is reset to FALSE, the execution of the function block is canceled. It is possible that responses have been generated for canceled requests or will arrive later, but these responses will be ignored at this point.



Execution is aborted or terminated due to an error detected.

Once the current execution is aborted due to receiving an SDO abort message or terminated due to an error detected, ERROR is set to a value other than 0. For more information about the detected error codes, see the following table. If the function block is used when ENABLE is reset to FALSE, ERROR is reset to 0, and the function block can start a new execution.



■ Error description

The data type of ERROR is enumerated type CANOPEN_KERNEL_ERROR. The meanings of the enumerators are as follows.

Error Code	Error	Description
0	CANOPEN_KERNEL_NO_ERROR	CANopen kernel does not detect any errors.
1	CANOPEN_KERNEL_OTHER_ERROR	If the value of ERROR is 01 in hexadecimal format, the error is detected. If the function block involves any other error codes, the output contains more detailed information. Example: CIA405.SDO_READ and CIA405.SDO_WRITE function blocks: The output of ERRORINFO contains the content of the SDO abort message. CIA405.RECV_EMCY and CIA405.RECV_EMCY_DEV function blocks: The output of ERRORINFO contains the content of the received EMCY message. If the output of ERRORINFO contains the EMCY message, the output of ERROR is 1 and the output of CONFIRM is 0. CIA405.GET_CANOPEN_KERNEL_STATE function block: The value 01 of the hexadecimal format will not be provided because no other error codes are output.
2	CANOPEN_KERNEL_DATA_OVERFLOW	The sending buffer or reception butter of the CANopen object overflows.
3	CANOPEN_KERNEL_TIMEOUT	Execution of the function block times out.
16	CANOPEN_KERNEL_CANBUS_OFF	The CANopen node is disconnected from the CAN bus.
17	CANOPEN_KERNEL_CAN_ERROR_PASSIVE	The CANopen node is in the error passive state: The node communicates properly but cannot send active error flags when an error is detected.
33	CANOPEN_INTERNAL_FB_ERROR	Manufacturer-specific error code. Note: Values 21 (hexadecimal) to FF (hexadecimal) are manufacturer-specific values.

6.3. 5 GET_LOCAL_NODE_ID

This instruction gets the CANopen node ID of the controller.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
GET_LOCAL_NODE_ID	Get the CANopen NodeID of the local device	FB	GET_LOCAL_NODE_ID_0 GET_LOCAL_NODE_ID EN ENO ENABLE CONFIRM TIMEOUT ERROR DEVICE	GET_LOCAL_NODE_ID_0(ENABLE :=, TIMEOUT :=, CONFIRM =>, ERROR =>, DEVICE =>,);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
ENABLE	Enable	BOOL	Depends on data types	-	(Triggered by level) Execution of the function block switch
TIMEOUT	Maximum execution time	UDINT	Depends on data types	-	Maximum execution time, in the unit of ms If the execution times out before the response is received, the execution aborts due to timeout. 0 (default): Timeout disabled 1 to 65535: Timeout value

Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
CONFIRM	Completion flag of the function block	BOOL	Depends on data types	-	Completion flag of the function block
ERROR	Error information of function block	CANOPEN_KERNEL_ERROR	Depends on data types	-	Error information of function block
DEVICE	CANopen node ID of the controller	DEVICE	Depends on data types	-	CANopen node ID of the controller 0: Invalid 1 to 127: Controller node ID

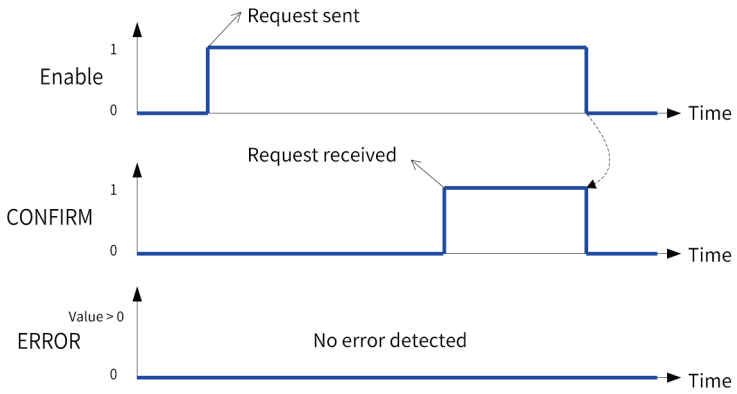
	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
ENABLE	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
TIMEOUT	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-
CONFIRM	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
ERROR	CANOPEN_KERNEL_ERROR																			
DEVICE	DEVICE																			

■ Function

This function block returns the CANopen node ID of the controller on the specified CAN bus interface.

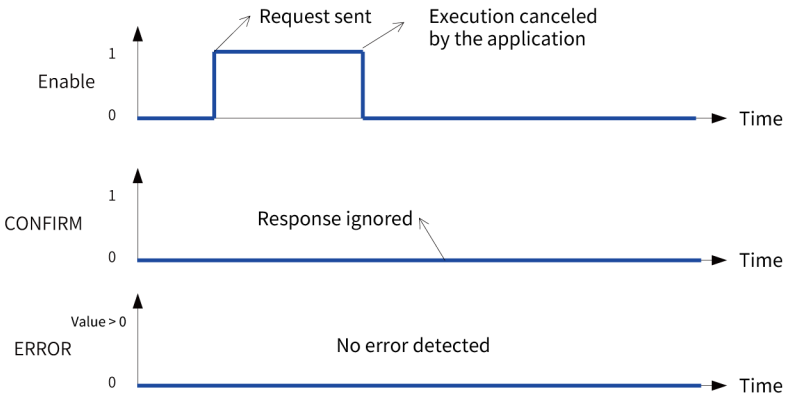
■ Timing diagram

Once a response is generated for the current request without detecting any errors, CONFIRM is immediately set to TRUE and remains TRUE (provided that the function block is used when ENABLE is set to TRUE). If the function block is used when ENABLE is reset to FALSE, CONFIRM is reset to FALSE, and the function block can start a new execution.



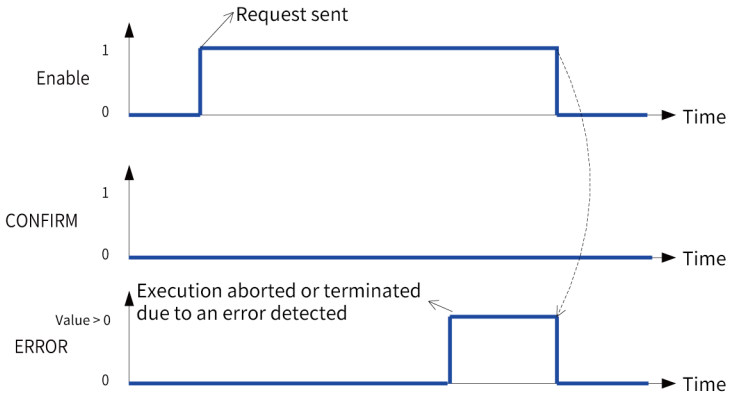
Execution is canceled by the application.

Before the current execution is completed, if the function block is used when ENABLE is reset to FALSE, the execution of the function block is canceled. It is possible that responses have been generated for canceled requests or will arrive later, but these responses will be ignored at this point.



Execution is aborted or terminated due to an error detected.

Once the current execution is aborted due to receiving an SDO abort message or terminated due to an error detected, ERROR is set to a value other than 0. For more information about the detected error codes, see the following table. If the function block is used when ENABLE is reset to FALSE, ERROR is reset to 0, and the function block can start a new execution.



■ Error description

The data type of ERROR is enumerated type CANOPEN_KERNEL_ERROR. The meanings of the enumerators are as follows.

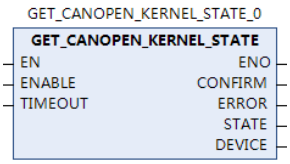
Error Code	Error	Description
0	CANOPEN_KERNEL_NO_ERROR	CANopen kernel does not detect any errors.
1	CANOPEN_KERNEL_OTHER_ERROR	If the value of ERROR is 01 in hexadecimal format, the error is detected. If the function block involves any other error codes, the output contains more detailed information. Example: CIA405.SDO_READ and CIA405.SDO_WRITE function blocks: The output of ERRORINFO contains the content of the SDO abort message. CIA405.RECV_EMCY and CIA405.RECV_EMCY_DEV function blocks: The output of ERRORINFO contains the content of the received EMCY message. If the output of ERRORINFO contains the EMCY message, the output of ERROR is 1 and the output of CONFIRM is 0. CIA405.GET_CANOPEN_KERNEL_STATE function block: The value 01 of the hexadecimal format will not be provided because no other error codes are output.
2	CANOPEN_KERNEL_DATA_OVERFLOW	The sending buffer or reception butter of the CANopen object overflows.
3	CANOPEN_KERNEL_TIMEOUT	Execution of the function block times out.
16	CANOPEN_KERNEL_CANBUS_OFF	The CANopen node is disconnected from the CAN bus.

Error Code	Error	Description
17	CANOPEN_KERNEL_CAN_ERROR_PASSIVE	The CANopen node is in the error passive state: The node communicates properly but cannot send active error flags when an error is detected.
33	CANOPEN_INTERNAL_FB_ERROR	Manufacturer-specific error code. Note: Values 21 (hexadecimal) to FF (hexadecimal) are manufacturer-specific values.

6.3. 6 GET_CANOPEN_KERNEL_STATE

This instruction gets the CANopen kernel state.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
GET_CANOPEN_KERNEL_STATE	Get the current state of the CANopen kernel	FB		GET_CANOPEN_KERNEL_STATE_0(ENABLE :=, TIMEOUT :=, CONFIRM =>, ERROR =>, STATE =>, DEVICE =>,);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
ENABLE	Enable	BOOL	Depends on data types	-	(Triggered by level) Execution of the function block switch
TIMEOUT	Maximum execution time	UDINT	Depends on data types	-	Maximum execution time, in the unit of ms If the execution times out before the response is received, the execution aborts due to timeout. 0 (default): Timeout disabled 1 to 65535: Timeout value

Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
CONFIRM	Completion flag of the function block	BOOL	Depends on data types	-	Completion flag of the function block
ERROR	Error information of function block	CANOPEN_KERNEL_ERROR	Depends on data types	-	Error information of function block
STATE	Current state of the controller CANopen kernel	CANOPEN_KERNEL_ERROR	Depends on data types	-	Current state of the controller CANopen kernel

DEVICE	CANopen node ID of the controller	DEVICE	Depends on data types	-	CANopen node ID of the controller 0: Invalid 1 to 127: Controller node ID

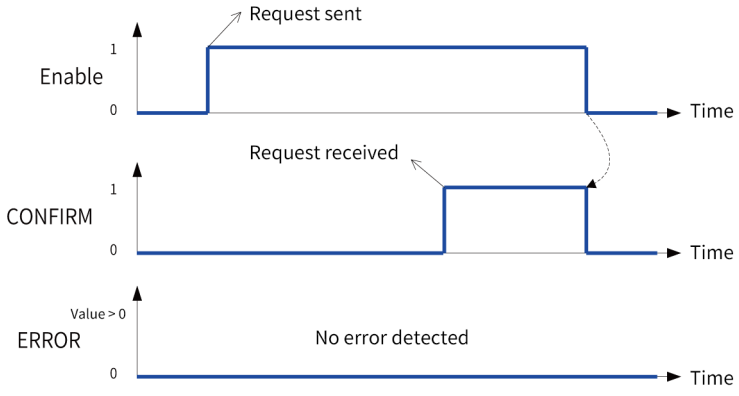
	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
ENABLE	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
TIMEOUT	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-
CONFIRM	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
ERROR	CANOPEN_KERNEL_ERROR																			
STATE	CANOPEN_KERNEL_ERROR																			
DEVICE	DEVICE																			

■ Function

This function block returns the current state of the controller CANopen kernel.

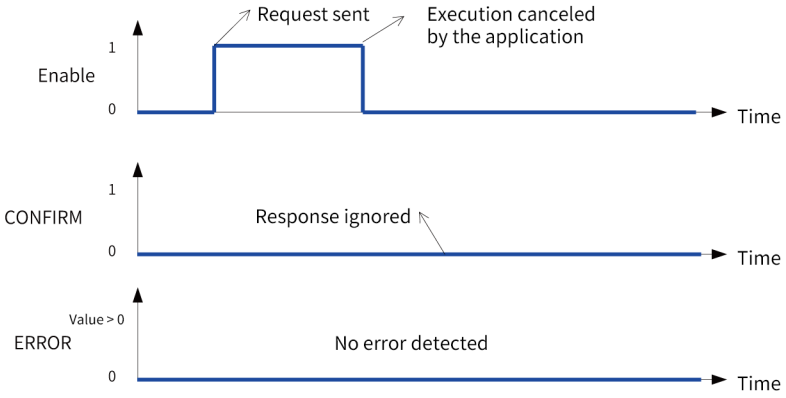
■ Timing diagram

Once a response is generated for the current request without detecting any errors, CONFIRM is immediately set to TRUE and remains TRUE (provided that the function block is used when ENABLE is set to TRUE). If the function block is used when ENABLE is reset to FALSE, CONFIRM is reset to FALSE, and the function block can start a new execution.



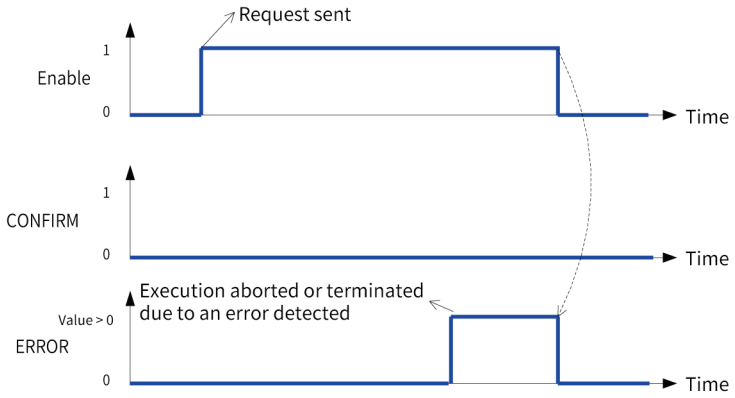
Execution is canceled by the application.

Before the current execution is completed, if the function block is used when ENABLE is reset to FALSE, the execution of the function block is canceled. It is possible that responses have been generated for canceled requests or will arrive later, but these responses will be ignored at this point.



Execution is aborted or terminated due to an error detected.

Once the current execution is aborted due to receiving an SDO abort message or terminated due to an error detected, ERROR is set to a value other than 0. For more information about the detected error codes, see the following table. If the function block is used when ENABLE is reset to FALSE, ERROR is reset to 0, and the function block can start a new execution.



■ Error description

The data type of ERROR is enumerated type CANOPEN_KERNEL_ERROR. The meanings of the enumerators are as follows.

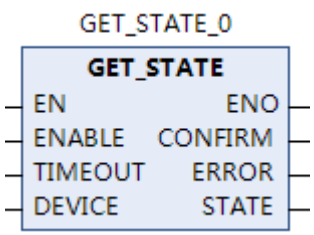
Error Code	Error	Description
0	CANOPEN_KERNEL_NO_ERROR	CANopen kernel does not detect any errors.
1	CANOPEN_KERNEL_OTHER_ERROR	If the value of ERROR is 01 in hexadecimal format, the error is detected. If the function block involves any other error codes, the output contains more detailed information. Example: CIA405.SDO_READ and CIA405.SDO_WRITE function blocks: The output of ERRORINFO contains the content of the SDO abort message. CIA405.RECV_EMCY and CIA405.RECV_EMCY_DEV function blocks: The output of ERRORINFO contains the content of the received EMCY message. If the output of ERRORINFO contains the EMCY message, the output of ERROR is 1 and the output of CONFIRM is 0. CIA405.GET_CANOPEN_KERNEL_STATE function block: The value 01 of the hexadecimal format will not be provided because no other error codes are output.
2	CANOPEN_KERNEL_DATA_OVERFLOW	The sending buffer or reception butter of the CANopen object overflows.
3	CANOPEN_KERNEL_TIMEOUT	Execution of the function block times out.
16	CANOPEN_KERNEL_CANBUS_OFF	The CANopen node is disconnected from the CAN bus.

Error Code	Error	Description
17	CANOPEN_KERNEL_CAN_ERROR_PASSIVE	The CANopen node is in the error passive state: The node communicates properly but cannot send active error flags when an error is detected.
33	CANOPEN_INTERNAL_FB_ERROR	Manufacturer-specific error code. Note: Values 21 (hexadecimal) to FF (hexadecimal) are manufacturer-specific values.

6.3. 7 GET_STATE

This instruction gets the CANopen device state.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
GET_STATE	Get the CANopen state of device	FB		GET_STATE_0(ENABLE :=, TIMEOUT :=, DEVICE :=, CONFIRM =>, ERROR =>, STATE =>,);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
ENABLE	Enable	BOOL	Depends on data types	-	(Triggered by level) Execution of the function block switch
TIMEOUT	Maximum execution time	UDINT	Depends on data types	-	Maximum execution time, in the unit of ms If the execution times out before the response is received, the execution aborts due to timeout. 0 (default): Timeout disabled 1 to 65535: Timeout value
DEVICE	Node ID of the CANopen device	DEVICE	Depends on data types	-	Node ID of the CANopen device 0 (default): Local device (controller) 1 to 127: Device node ID

Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
CONFIRM	Completion flag of the function block	BOOL	Depends on data types	-	Completion flag of the function block
ERROR	Error information of function block	CANOPEN_KERNEL_ERROR	Depends on data types	-	Error information of function block
STATE	Current state of the controller CANopen kernel	DEVICE_STATE	Depends on data types	-	Current state of the controller CANopen kernel

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
ENABLE	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
TIMEOUT	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-
DEVICE	DEVICE																			
CONFIRM	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
ERROR	CANOPEN_KERNEL_ERROR																			
STATE	DEVICE_STATE																			

■ Function

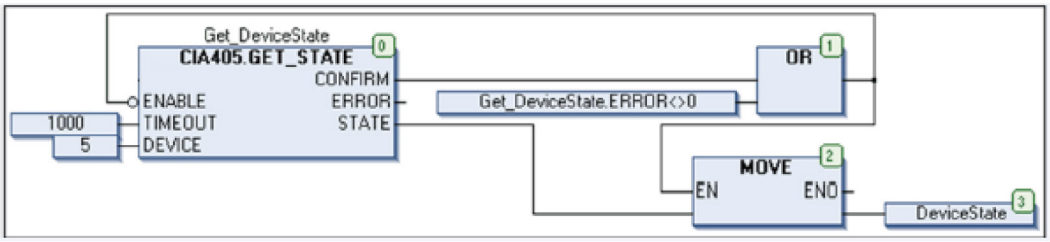
This function block returns the current NMT state of the CANopen device when the heartbeat or node protection is in the active state.

The data type of STATE is enumerated type DEVICE_STATE. The meanings of the enumerators are as follows.

Enumerator	Meaning
INIT	Initializing
RESET_COMM	Communication reset state
RESET_APP	Application reset state
PRE_OPERATIONAL	Pre-operation state
STOPPED	Stop state
OPERATIONAL	Normal operating state
UNKNOWN	Unknown NMT state The node protection or heartbeat of the selected device is not in the active state, or the controller is not the heartbeat consumer.
NOT_AVAIL	NMT state unavailable The node protection or heartbeat of the selected device is in the active state, but the device does not correctly report its NMT state before timeout.

■ Program example

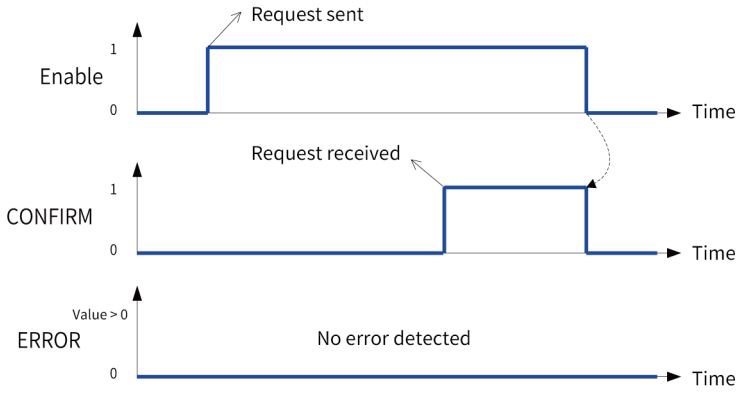
The following example demonstrates how to get the state of CANopen node 5 that is connected to the first CAN bus interface, with a timeout period of 1s (1000 ms). The CIA405.GET_STATE function is automatically executed to continuously read the state. The NMT state of the device is copied to the DeviceState variable of the CIA405.DEVICE_STATE type.



■ Timing diagram

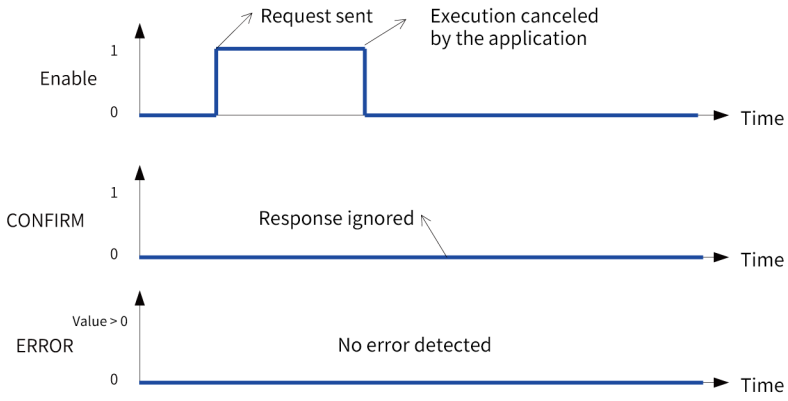
Once a response is generated for the current request without detecting any errors, CONFIRM is immediately set to TRUE and remains TRUE (provided that the function block is used when ENABLE is set to TRUE). If the function block is used when ENABLE is reset to FALSE, CONFIRM is reset to FALSE, and the

function block can start a new execution.



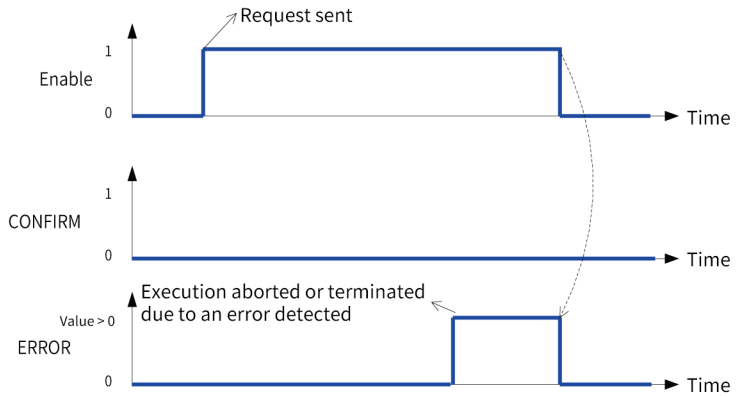
Execution is canceled by the application.

Before the current execution is completed, if the function block is used when ENABLE is reset to FALSE, the execution of the function block is canceled. It is possible that responses have been generated for canceled requests or will arrive later, but these responses will be ignored at this point.



Execution is aborted or terminated due to an error detected.

Once the current execution is aborted due to receiving an SDO abort message or terminated due to an error detected, ERROR is set to a value other than 0. For more information about the detected error codes, see the following table. If the function block is used when ENABLE is reset to FALSE, ERROR is reset to 0, and the function block can start a new execution.



■ Error description

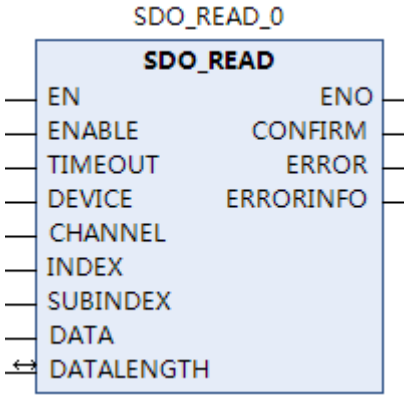
The data type of ERROR is enumerated type CANOPEN_KERNEL_ERROR. The meanings of the enumerators are as follows.

Error Code	Error	Description
0	CANOPEN_KERNEL_NO_ERROR	CANopen kernel does not detect any errors.
1	CANOPEN_KERNEL_OTHER_ERROR	If the value of ERROR is 01 in hexadecimal format, the error is detected. If the function block involves any other error codes, the output contains more detailed information. Example: CIA405.SDO_READ and CIA405.SDO_WRITE function blocks: The output of ERRORINFO contains the content of the SDO abort message. CIA405.RECV_EMCY and CIA405.RECV_EMCY_DEV function blocks: The output of ERRORINFO contains the content of the received EMCY message. If the output of ERRORINFO contains the EMCY message, the output of ERROR is 1 and the output of CONFIRM is 0. CIA405.GET_CANOPEN_KERNEL_STATE function block: The value 01 of the hexadecimal format will not be provided because no other error codes are output.
2	CANOPEN_KERNEL_DATA_ OVERFLOW	The sending buffer or reception butter of the CANopen object overflows.
3	CANOPEN_KERNEL_TIMEOUT	Execution of the function block times out.
16	CANOPEN_KERNEL_CANBUS_OFF	The CANopen node is disconnected from the CAN bus.
17	CANOPEN_KERNEL_CAN_ERROR_ PASSIVE	The CANopen node is in the error passive state: The node communicates properly but cannot send active error flags when an error is detected.
33	CANOPEN_INTERNAL_FB_ERROR	Manufacturer-specific error code. Note: Values 21 (hexadecimal) to FF (hexadecimal) are manufacturer-specific values.

6.3. 8 SDO_READ

This instruction reads CANopen objects of any size.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
SDO_READ	Read specific object from object dictionary of device	FB		SDO_READ_0(ENABLE :=, TIMEOUT :=, DEVICE :=, CHANNEL :=, INDEX := , SUBINDEX := , DATA := , DATALENGTH :=, CONFIRM =>, ERROR =>, ERRORINFO =>,);

■ Variables

In-out variables

In-Out Variable	Name	Data Type	Value Range	Initial Value	Description
DATALENGTH	Data buffer size	UINT	Depends on data types	-	Input: Data butter size (in the unit of bytes) Note: To correctly initialize the input of DATALENGTH (when the function block execution starts when ENABLE is set to TRUE) as the size of the data buffer, use the SIZEOF standard function. Output: Size of a read object (in the unit of bytes)

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
ENABLE	Enable	BOOL	Depends on data types	-	(Triggered by level) Execution of the function block switch
TIMEOUT	Maximum execution time	UDINT	Depends on data types	-	Maximum execution time, in the unit of ms If the execution times out before the response is received, the execution aborts due to timeout. 0 (default): Timeout disabled 1 to 65535: Timeout value
DEVICE	Node ID of the CANopen device	DEVICE	Depends on data types	-	Node ID of the CANopen device 0 (default): Local device (controller) 1 to 127: Device node ID
CHANNEL	SDO channel number	USINT	Depends on data types	-	SDO channel number Default value: 1
INDEX	Object index	WORD	Depends on data types	-	Object index The value range is from 0000 (hexadecimal) to FFFF (hexadecimal).
SUBINDEX	Object sub-index	BYTE	Depends on data types	-	Object sub-index The value range is from 00 (hexadecimal) to FF (hexadecimal).
DATA	Address of the data buffer to read data	POINTER TO BYTE	Depends on data types	-	Address of the data buffer to receive data read from the device object The ADR standard function must be used to define the associated pointer.

Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
CONFIRM	Completion flag of the function block	BOOL	Depends on data types	-	Completion flag of the function block
ERROR	Error information of function block	CANOPEN_KERNEL_ERROR	Depends on data types	-	Error information of function block
ERRORINFO	SDO abort message content	SDO_ERROR	Depends on data types	-	When ERROR is 1, the SDO abort message content (four bytes) is returned.

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
ENABLE	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
TIMEOUT	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-
DEVICE	DEVICE																			
CHANNEL	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-
INDEX	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
SUBINDEX	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
DATA	POINTER TO BYTE																			
DATALENGTH	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-
CONFIRM	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
ERROR	CANOPEN_KERNEL_ERROR																			
ERRORINFO	SDO_ERROR																			

■ Function

This function block reads a device-specific CANopen object of any size from the SDO message.

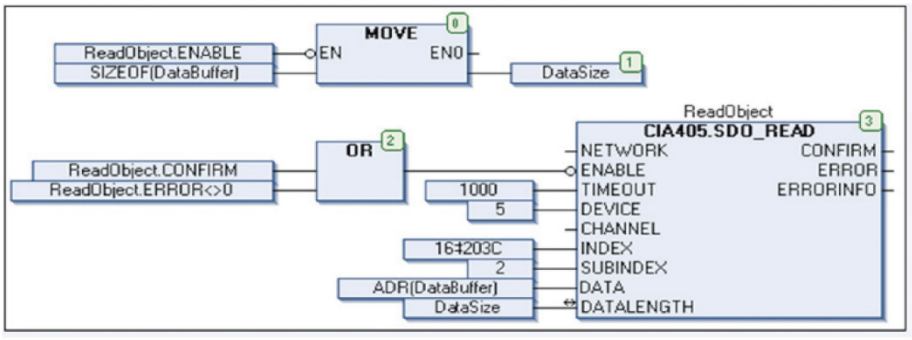
■ Program example

The following example demonstrates how to read the object index 203C (hexadecimal) and sub-index 02 (hexadecimal) of CANopen node 5 that is connected to the first CAN bus interface, with a timeout period of 1s (1000 ms). The ReadObject instance of the CIA405.SDO_READ function block is automatically executed to continuously read data.

Variable DataSize (UINT type):

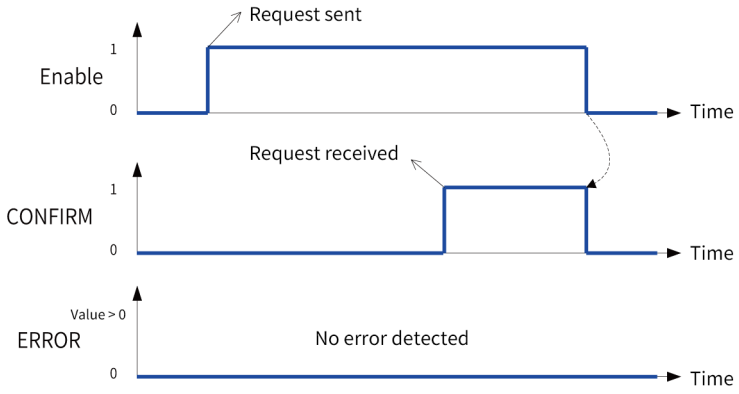
Initialized to the size of the data buffer (DataBuffer: N-byte array) (when ENABLE is set to FALSE for the function block and before the next execution starts)

Including the size of data read when CONFIRM is set to TRUE for the function block (This example does not demonstrate how to extract values from the data buffer or how to manage error detection.)



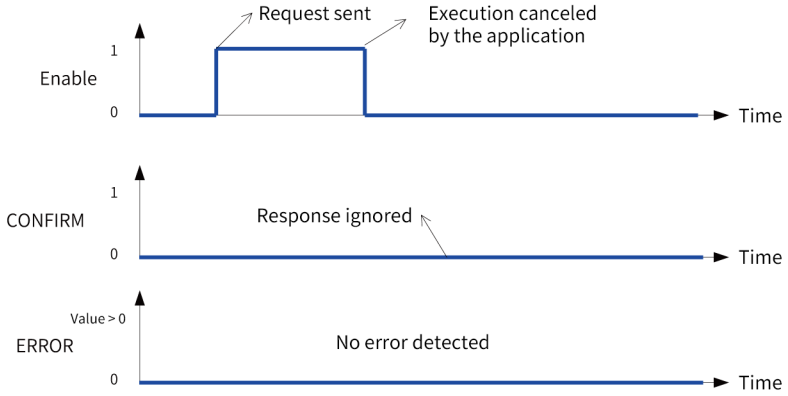
■ Timing diagram

Once a response is generated for the current request without detecting any errors, CONFIRM is immediately set to TRUE and remains TRUE (provided that the function block is used when ENABLE is set to TRUE). If the function block is used when ENABLE is reset to FALSE, CONFIRM is reset to FALSE, and the function block can start a new execution.



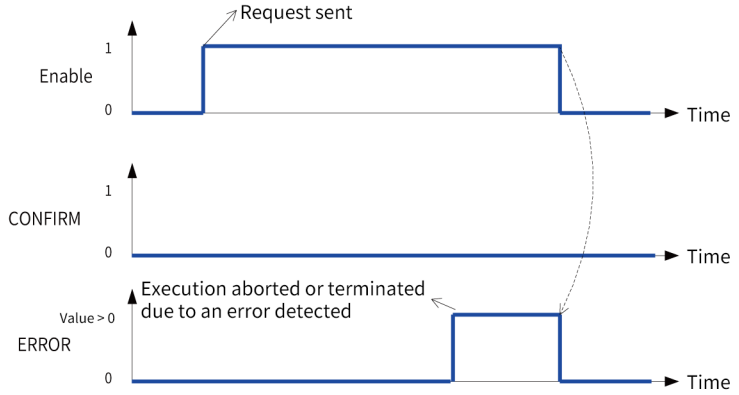
Execution is canceled by the application.

Before the current execution is completed, if the function block is used when ENABLE is reset to FALSE, the execution of the function block is canceled. It is possible that responses have been generated for canceled requests or will arrive later, but these responses will be ignored at this point.



Execution is aborted or terminated due to an error detected.

Once the current execution is aborted due to receiving an SDO abort message or terminated due to an error detected, ERROR is set to a value other than 0. For more information about the detected error codes, see the following table. If the function block is used when ENABLE is reset to FALSE, ERROR is reset to 0, and the function block can start a new execution.



■ Precautions

The following specific parameters must be transmitted to the function block:

1) Device node ID
2) SDO client/server channel (By default, only one channel is defined.)
3) CANopen object index/sub-index

4) Pointer of the data buffer for storing object values

5) Data buffer size

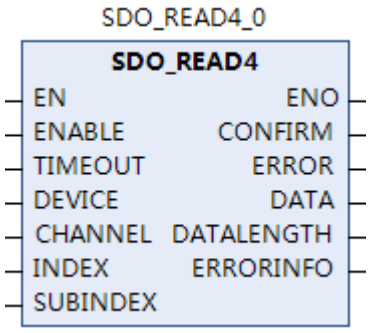
If data is read successfully, the function block returns the size of the read object. The data is provided in the data buffer.

If the object to read contains four bytes or less, the CIA405.SDO_READ4 function block is recommended.

6.3. 9 SDO_READ4

This instruction reads a CANopen object of up to four bytes.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
SDO_READ4	Read specific object up to 4 bytes from object dictionary of device	FB		SDO_READ4_0(ENABLE :=, TIMEOUT :=, DEVICE :=, CHANNEL :=, INDEX := , SUBINDEX := , CONFIRM =>, ERROR =>, DATA => , DATALENGTH=>, ERRORINFO =>,);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
ENABLE	Enable	BOOL	Depends on data types	-	(Triggered by level) Execution of the function block switch
TIMEOUT	Maximum execution time	UDINT	Depends on data types	-	Maximum execution time, in the unit of ms If the execution times out before the response is received, the execution aborts due to timeout. 0 (default): Timeout disabled 1 to 65535: Timeout value
DEVICE	Node ID of the CANopen device	DEVICE	Depends on data types	-	Node ID of the CANopen device 0 (default): Local device (controller) 1 to 127: Device node ID
CHANNEL	SDO channel number	USINT	Depends on data types	-	SDO channel number Default value: 1
INDEX	Object index	WORD	Depends on data types	-	Object index The value range is from 0000 (hexadecimal) to FFFF (hexadecimal).
SUBINDEX	Object sub-index	BYTE	Depends on data types	-	Object sub-index The value range is from 00 (hexadecimal) to FF (hexadecimal).

Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
CONFIRM	Completion flag of the function block	BOOL	Depends on data types	-	Completion flag of the function block
ERROR	Error information of function block	CANOPEN_KERNEL_ERROR	Depends on data types	-	Error information of function block
DATA	Receive data	ARRAY[1...4] OF BYTE	Depends on data types	-	Array to receive data read from the device object
DATALENGTH	Size of the read object	USINT	Depends on data types	-	Size of a read object (in the unit of bytes)
ERRORINFO	SDO abort message content	SDO_ERROR	Depends on data types	-	When ERROR is 1, the SDO abort message content (four bytes) is returned.

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
ENABLE	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
TIMEOUT	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-
DEVICE	DEVICE																			
CHANNEL	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-
INDEX	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
SUBINDEX	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
CONFIRM	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
ERROR	CANOPEN_KERNEL_ERROR																			
DATA	ARRAY[1...4] OF BYTE																			
DATALENGTH	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-
ERRORINFO	SDO_ERROR																			

■ Function

This function block reads a device-specific CANopen object of up to four bytes from the SDO message.

The following table lists the object size and corresponding DATA array content.

Object Size Example	DATALENGTH	DATA(1)	DATA(2)	DATA(3)	DATA(4)
1 byte 01 (hexadecimal)	1	01 (hexadecimal)	Disable	Disable	Disable
2 bytes 01 23 (hexadecimal)	2	LSB 23 (hexadecimal)	MSB 01 (hexadecimal)	Disable	Disable

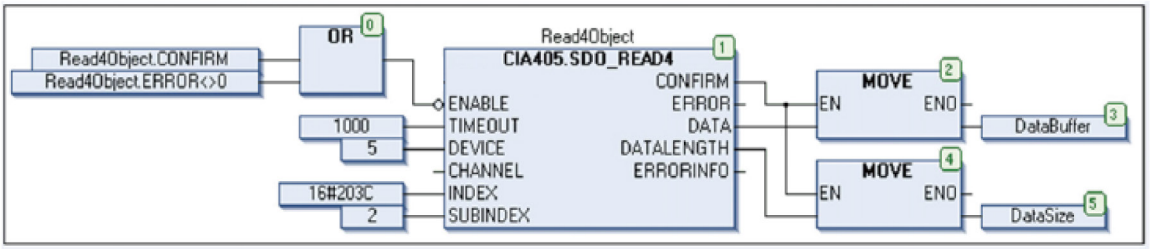
Object Size Example	DATALENGTH	DATA(1)	DATA(2)	DATA(3)	DATA(4)
3 bytes 01 23 45 (hexadecimal)	3	LSB 45 (hexadecimal)	23 (hexadecimal)	MSB 01 (hexadecimal)	Disable
4 bytes 01 23 45 67 (hexadecimal)	4	LSB 67 (hexadecimal)	45 (hexadecimal)	23 (hexadecimal)	MSB 01 (hexadecimal)

LSB: Least significant byte

MSB: Most significant byte

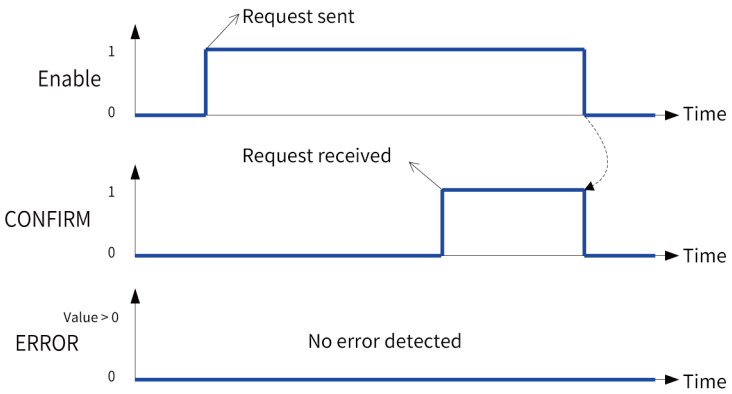
■ Program example

The following example demonstrates how to read the object index 203C (hexadecimal) and sub-index 02 (hexadecimal) of CANopen node 5 that is connected to the first CAN bus interface, with a timeout period of 1s (1000 ms). The Read4Object instance of the CIA405.SDO_READ4 function block is automatically executed to continuously read data. The DataBuffer variable (four-byte array) contains the value of the last read data. The DataSize variable (USINT type) contains the size of the last read data (up to four bytes). This example does not demonstrate how to management error detection.



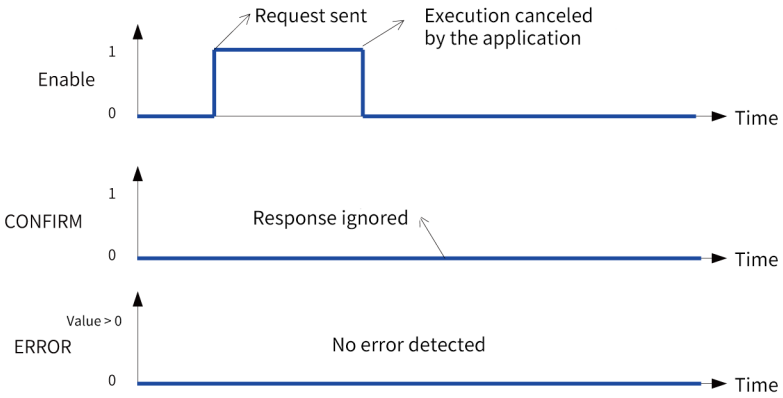
■ Timing diagram

Once a response is generated for the current request without detecting any errors, CONFIRM is immediately set to TRUE and remains TRUE (provided that the function block is used when ENABLE is set to TRUE). If the function block is used when ENABLE is reset to FALSE, CONFIRM is reset to FALSE, and the function block can start a new execution.



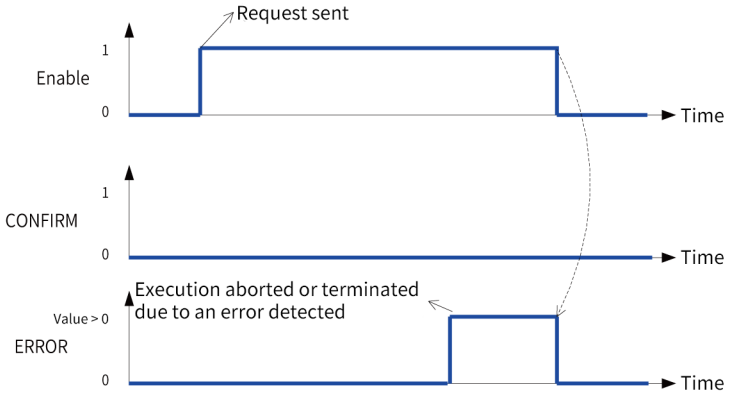
Execution is canceled by the application.

Before the current execution is completed, if the function block is used when ENABLE is reset to FALSE, the execution of the function block is canceled. It is possible that responses have been generated for canceled requests or will arrive later, but these responses will be ignored at this point.



Execution is aborted or terminated due to an error detected.

Once the current execution is aborted due to receiving an SDO abort message or terminated due to an error detected, ERROR is set to a value other than 0. For more information about the detected error codes, see the following table. If the function block is used when ENABLE is reset to FALSE, ERROR is reset to 0, and the function block can start a new execution.



■ Precautions

The following specific parameters must be transmitted to the function block:

1) Device node ID

2) SDO client/server channel (By default, only one channel is defined.)

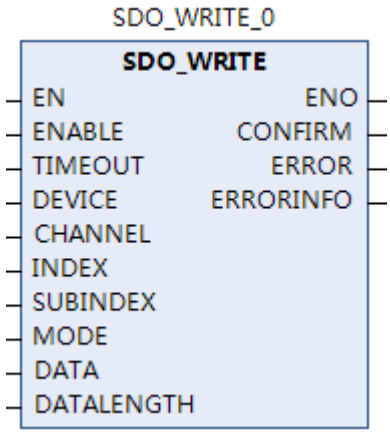
3) CANopen object index/sub-index

If data is read successfully, the function block returns the size of the read object. The data is provided in a four-byte array.

6.3. 10 SDO_WRITE

This instruction writes a CANopen object of any size.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
SDO_WRITE	Write specific object in object dictionary of device	FB		SDO_WRITE_0(ENABLE :=, TIMEOUT :=, DEVICE :=, CHANNEL :=, INDEX := , SUBINDEX := , MODE :=, DATA := , DATALENGTH :=, CONFIRM =>, ERROR =>, ERRORINFO =>,);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
ENABLE	Enable	BOOL	Depends on data types	-	(Triggered by level) Execution of the function block switch
TIMEOUT	Maximum execution time	UDINT	Depends on data types	-	Maximum execution time, in the unit of ms If the execution times out before the response is received, the execution aborts due to timeout. 0 (default): Timeout disabled 1 to 65535: Timeout value
DEVICE	Node ID of the CANopen device	DEVICE	Depends on data types	-	Node ID of the CANopen device 0 (default): Local device (controller) 1 to 127: Device node ID
CHANNEL	SDO channel number	USINT	Depends on data types	-	SDO channel number Default value: 1
INDEX	Object index	WORD	Depends on data types	-	Object index The value range is from 0000 (hexadecimal) to FFFF (hexadecimal).
SUBINDEX	Object sub-index	BYTE	Depends on data types	-	Object sub-index The value range is from 00 (hexadecimal) to FF (hexadecimal).
MODE	Data transmission mode	SDO_MODE	Depends on data types	-	Data transmission mode 0 (default): AUTO
DATA	Address of the data buffer to read data	POINTER TO BYTE	Depends on data types	-	Address of the data buffer to receive data read from the device object The ADR standard function must be used to define the associated pointer.

Input Variable	Name	Data Type	Value Range	Initial Value	Description
DATALENGTH	Size of the object to write	UINT	Depends on data types	-	Size of the object to write, in the unit of bytes

Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
CONFIRM	Completion flag of the function block	BOOL	Depends on data types	-	Completion flag of the function block
ERROR	Error information of function block	CANOPEN_KERNEL_ERROR	Depends on data types	-	Error information of function block
ERRORINFO	SDO abort message content	SDO_ERROR	Depends on data types	-	When ERROR is 1, the SDO abort message content (four bytes) is returned.

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
ENABLE	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
TIMEOUT	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-
DEVICE	DEVICE																			
CHANNEL	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-
INDEX	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
SUBINDEX	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
MODE	SDO_MODE																			
DATA	POINTER TO BYTE																			
DATALENGTH	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-
CONFIRM	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
ERROR	CANOPEN_KERNEL_ERROR																			
ERRORINFO	SDO_ERROR																			

■ Function

This function block writes a device-specific CANopen object of any size into the SDO message.

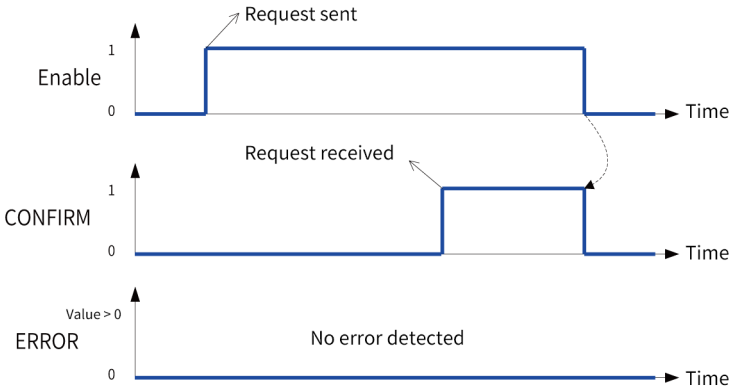
The data type of MODE is enumerated type SDO_MODE. The meanings of the enumerators are as follows.

Enumerator	Meaning
AUTO	Auto mode
EXPEDITED	SDO acceleration mode for data of up to four bytes The data is sent in an SDO request.
SEGMENTED	SDO segment mode for data of more than four bytes The data is divided into seven-byte segments, which are then sent in continuous SDO confirm requests.

Enumerator	Meaning
BLOCK	SDO block mode for data of more than four bytes The data sent in continuous frames is divided into seven-byte data blocks. These blocks will not be confirmed. After all blocks are received, the receiver sends a confirm message. Note: Transmission in this way is fast. However, your device may not support this mode because the mode is a recent addition item for CANopen configuration.

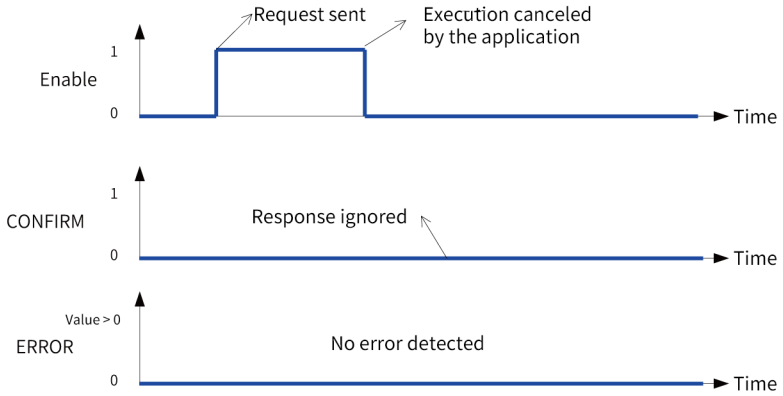
■ Timing diagram

Once a response is generated for the current request without detecting any errors, CONFIRM is immediately set to TRUE and remains TRUE (provided that the function block is used when ENABLE is set to TRUE). If the function block is used when ENABLE is reset to FALSE, CONFIRM is reset to FALSE, and the function block can start a new execution.



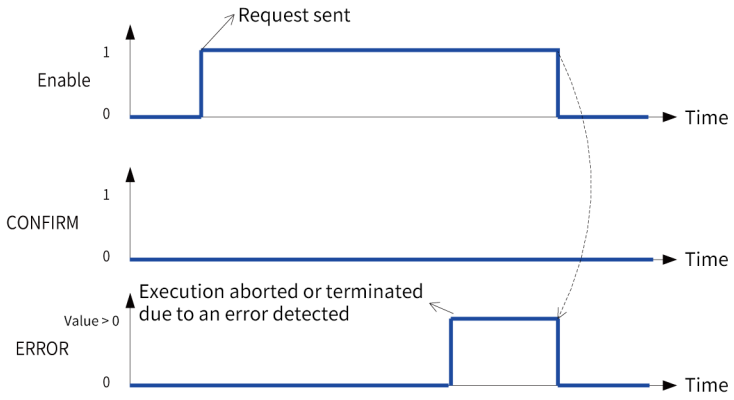
Execution is canceled by the application.

Before the current execution is completed, if the function block is used when ENABLE is reset to FALSE, the execution of the function block is canceled. It is possible that responses have been generated for canceled requests or will arrive later, but these responses will be ignored at this point.



Execution is aborted or terminated due to an error detected.

Once the current execution is aborted due to receiving an SDO abort message or terminated due to an error detected, ERROR is set to a value other than 0. For more information about the detected error codes, see the following table. If the function block is used when ENABLE is reset to FALSE, ERROR is reset to 0, and the function block can start a new execution.



■ Precautions

The following specific parameters must be transmitted to the function block:

1) Data buffer size
2) Device node ID
3) SDO client/server channel (By default, only one channel is defined.)
4) CANopen object index/sub-index
5) SDO mode
6) Pointer of the data buffer for storing object values to write
7) Number of bytes to write

If the object to write contains four bytes or less, the CIA405.SDO_WRITE4 function block is recommended.

6.3. 11 SDO_WRITE4

This instruction writes a CANopen object of up to four bytes.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
SDO_WRITE4	Write specific object up to 4 bytes in object dictionary of device	FB	SDO_WRITE4_0 SDO_WRITE4 EN ENO ENABLE CONFIRM TIMEOUT ERROR DEVICE ERRORINFO CHANNEL INDEX SUBINDEX DATA DATALENGTH	SDO_WRITE4_0(ENABLE :=, TIMEOUT :=, DEVICE :=, CHANNEL :=, INDEX := , SUBINDEX := , DATA := , DATALENGTH :=, CONFIRM =>, ERROR =>, ERRORINFO =>,);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
ENABLE	Enable	BOOL	Depends on data types	-	(Triggered by level) Execution of the function block switch

TIMEOUT	Maximum execution time	UDINT	Depends on data types	-	Maximum execution time, in the unit of ms If the execution times out before the response is received, the execution aborts due to timeout. 0 (default): Timeout disabled 1 to 65535: Timeout value
DEVICE	Node ID of the CANopen device	DEVICE	Depends on data types	-	Node ID of the CANopen device 0 (default): Local device (controller) 1 to 127: Device node ID
CHANNEL	SDO channel number	USINT	Depends on data types	-	SDO channel number Default value: 1
INDEX	Object index	WORD	Depends on data types	-	Object index The value range is from 0000 (hexadecimal) to FFFF (hexadecimal).
SUBINDEX	Object sub-index	BYTE	Depends on data types	-	Object sub-index The value range is from 00 (hexadecimal) to FF (hexadecimal).
DATA	Data to receive	ARRAY[1...4] OF BYTE	Depends on data types	-	Array for storing object values to write
DATALENGTH	Size of the object to write	UINT	Depends on data types	-	Size of the object to write, in the unit of bytes

Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
CONFIRM	Completion flag of the function block	BOOL	Depends on data types	-	Completion flag of the function block
ERROR	Error information of function block	CANOPEN_KERNEL_ERROR	Depends on data types	-	Error information of function block
ERRORINFO	SDO abort message content	SDO_ERROR	Depends on data types	-	When ERROR is 1, the SDO abort message content (four bytes) is returned.

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
ENABLE	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
TIMEOUT	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-
DEVICE	DEVICE																			
CHANNEL	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-
INDEX	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
SUBINDEX	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

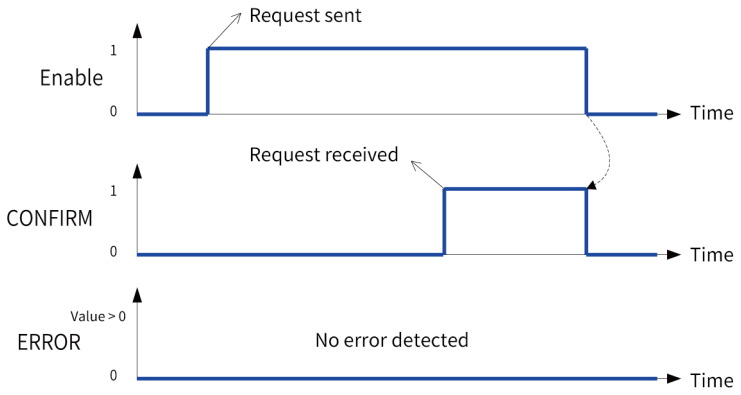
	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
DATA	ARRAY[1...4] OF BYTE																			
DATALENGTH	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-
CONFIRM	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
ERROR	CANOPEN_KERNEL_ERROR																			
ERRORINFO	SDO_ERROR																			

■ Function

This function block writes a device-specific CANopen object of up to four bytes from the SDO message.

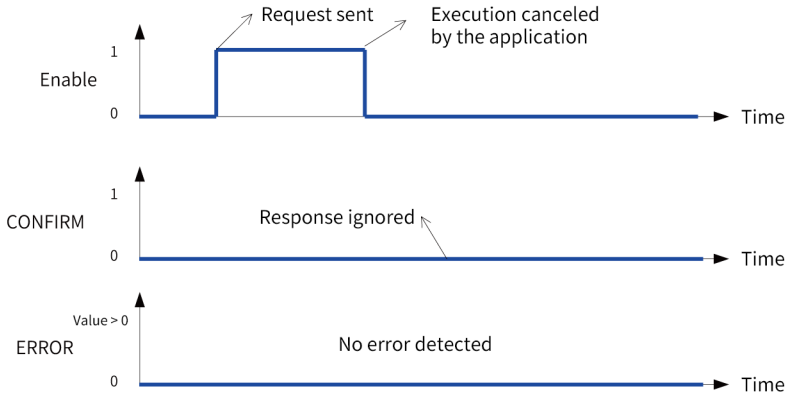
■ Timing diagram

Once a response is generated for the current request without detecting any errors, CONFIRM is immediately set to TRUE and remains TRUE (provided that the function block is used when ENABLE is set to TRUE). If the function block is used when ENABLE is reset to FALSE, CONFIRM is reset to FALSE, and the function block can start a new execution.



Execution is canceled by the application.

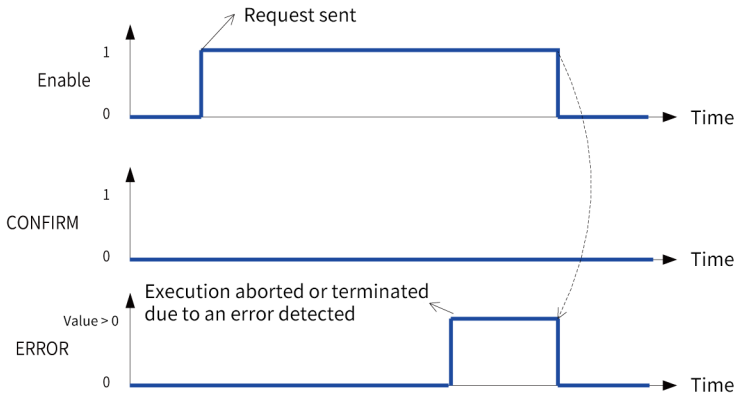
Before the current execution is completed, if the function block is used when ENABLE is reset to FALSE, the execution of the function block is canceled. It is possible that responses have been generated for canceled requests or will arrive later, but these responses will be ignored at this point.



Execution is aborted or terminated due to an error detected.

Once the current execution is aborted due to receiving an SDO abort message or terminated due to an error detected, ERROR is set to a value other than 0. For more information about the detected error codes, see the following table. If the function block is used when ENABLE is reset to FALSE, ERROR is reset

to 0, and the function block can start a new execution.



■ Precautions

The following specific parameters must be transmitted to the function block:

1) Device node ID
2) SDO client/server channel (By default, only one channel is defined.)
3) CANopen object index/sub-index
4) Value to write
5) Number of bytes to write

6.3. 12 CANOPEN_COUNT

This instruction implements CANopen count.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
CANOPEN_ COUNT	CANopen count	FB	CANOPEN_COUNT_0 CANOPEN_COUNT EN ENABLE CLEAR COBID1FORCOUNT COBID2FORCOUNT COBID3FORCOUNT COBID4FORCOUNT ENO CONFIRM TASKPERIOD RXBUFFERCOUNT TXBUFFERCOUNT RXCOUNT TXCOUNT RXTXCOUNT RXCOUNTPERSECOND TXCOUNTPERSECOND RXTXCOUNTPERSECOND REPEATCOUNT COBID1COUNT COBID2COUNT COBID3COUNT COBID4COUNT	CANOPEN_COUNT_0(ENABLE :=, CLEAR :=, COBID1FORCOUNT :=, COBID2FORCOUNT :=, COBID3FORCOUNT :=, COBID4FORCOUNT :=, CONFIRM => TASKPERIOD => RXBUFFERCOUNT => TXBUFFERCOUNT => RXCOUNT => TXCOUNT => RXTXCOUNT => RXCOUNTPERSECOND => TXCOUNTPERSECOND => RXTXCOUNTPERSECOND=> REPEATCOUNT => COBID1COUNT => COBID2COUNT => COBID3COUNT => COBID4COUNT =>);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
ENABLE	Enable	BOOL	Depends on data types	-	(Triggered by level) Execution of the function block switch
CLEAR	Clearing of the current output parameter value	BOOL	Depends on data types	-	Clearing of the current output parameter value
COBID1FORCOUNT	Data frame of ID 1	DWORD	Depends on data types	-	Data frame of ID 1, which is not supported
COBID2FORCOUNT	Data frame of ID 2	DWORD	Depends on data types	-	Data frame of ID 2, which is not supported
COBID3FORCOUNT	Data frame of ID 3	DWORD	Depends on data types	-	Data frame of ID 3, which is not supported
COBID4FORCOUNT	Data frame of ID 4	DWORD	Depends on data types	-	Data frame of ID 4, which is not supported

Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
CONFIRM	Completion flag of the function block	BOOL	Depends on data types	-	TRUE: Successful execution
TASKPERIOD	Network load rate	DWORD	Depends on data types	-	Network load rate
RXBUFFERCOUNT	CAN controller error	DWORD	Depends on data types	-	CAN controller error
TXBUFFERCOUNT	CAN controller error, which is not supported	DWORD	Depends on data types	-	CAN controller error, which is not supported
RXCOUNT	Number of sent frames	DWORD	Depends on data types	-	Number of sent frames
TXCOUNT	Number of received frames	DWORD	Depends on data types	-	Number of received frames
RXTXCOUNT	Number of transceived frames	DWORD	Depends on data types	-	Number of transceived frames
RXCOUNTPERSECOND	Frame receiving rate, which is not supported	DWORD	Depends on data types	-	Frame receiving rate, which is not supported
TXCOUNTPERSECOND	Frame sending rate, which is not supported	DWORD	Depends on data types	-	Frame sending rate, which is not supported
RXTXCOUNTPERSECOND	Frame receiving rate, which is not supported	DWORD	Depends on data types	-	Frame receiving rate, which is not supported
REPEATCOUNT	Frame resending count, which is not supported	DWORD	Depends on data types	-	Frame resending count, which is not supported

Output Variable	Name	Data Type	Value Range	Initial Value	Description
COBID1COUNT	Number of data frames of ID 1, which is not supported	DWORD	Depends on data types	-	Number of data frames of ID 1, which is not supported
COBID2COUNT	Number of data frames of ID 2, which is not supported	DWORD	Depends on data types	-	Number of data frames of ID 2, which is not supported
COBID3COUNT	Number of data frames of ID 3, which is not supported	DWORD	Depends on data types	-	Number of data frames of ID 3, which is not supported
COBID4COUNT	Number of data frames of ID 4, which is not supported	DWORD	Depends on data types	-	Number of data frames of ID 4, which is not supported

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
ENABLE	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
CLEAR	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
COBID1FORCOUNT	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
COBID2FORCOUNT	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
COBID3FORCOUNT	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
COBID4FORCOUNT	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
CONFIRM	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
TASKPERIOD	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
RXBUFFERCOUNT	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
TXBUFFERCOUNT	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
RXCOUNT	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
TXCOUNT	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
RXTXCOUNT	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
RXCOUNTPERSECOND	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
TXCOUNTPERSECOND	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
RXTXCOUNTPERSECOND	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
REPEATCOUNT	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
COBID1COUNT	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
COBID2COUNT	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
COBID3COUNT	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
COBID4COUNT	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

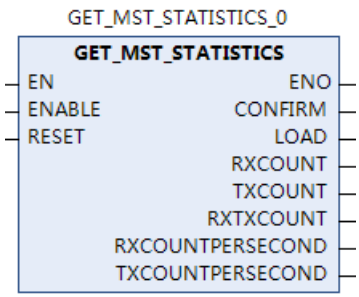
■ Function

This instruction implements CANopen count.

6.3. 13 GET_MST_STATISTICS

This instruction gets MST statistics.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
GET_MST_STATISTICS	Get MST statistics	FB		GET_MST_STATISTICS_0(ENABLE :=, RESET :=, CONFIRM =>, LOAD =>, RXCOUNT =>, TXCOUNT =>, RXTXCOUNT =>, RXCOUNTPERSECOND=>, TXCOUNTPERSECOND=>,);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
ENABLE	Enable	BOOL	Depends on data types	-	(Triggered by level) Execution of the function block switch
Reset	Clearing of the current output parameter value	BOOL	Depends on data types	-	Clearing of the current output parameter value

Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
CONFIRM	Completion flag of the function block	BOOL	Depends on data types	-	TRUE: Successful execution
LOAD	Network load rate	BYTE	Depends on data types	-	Network load rate
RXCOUNT	Number of sent frames	DWORD	Depends on data types	-	Number of sent frames
TXCOUNT	Number of received frames	DWORD	Depends on data types	-	Number of received frames
RXTXCOUNT	Number of transceived frames	DWORD	Depends on data types	-	Number of transceived frames
RXCOUNTPERSECOND	Frame receiving rate, which is not supported	DWORD	Depends on data types	-	Frame receiving rate, which is not supported
TXCOUNTPERSECOND	Frame sending rate, which is not supported	DWORD	Depends on data types	-	Frame sending rate, which is not supported

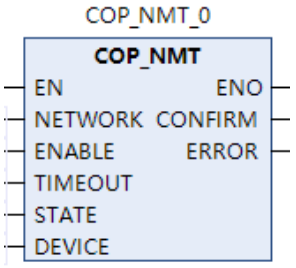
■ Function

This instruction gets MST statistics.

6.3. 14 COP_NMT

This instruction manages the device network status.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
COP_NMT	Device NMT state management	FB		COP_NMT_0(NETWORK:=, ENABLE :=, TIMEOUT :=, STATE :=, DEVICE :=, CONFIRM =>, ERROR =>,);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
NETWORK	CAN network number	USINT	Depends on data types	-	CAN port executed by the function block, corresponding to the CAN port number of the host
ENABLE	Enable	BOOL	Depends on data types	-	(Triggered by level) Execution of the function block switch
TIMEOUT	Maximum execution time	UDINT	Depends on data types	-	Maximum execution time, in the unit of ms If the execution times out before the response is received, the execution aborts due to timeout. 0 (default): Timeout disabled 1 to 65535: Timeout value This variable is not available.
STATE	NMT status to request	TRANSITION_ STATE	Depends on data types	-	Request NMT state transition
DEVICE	Destination device node ID	DEVICE	Depends on data types	-	Target CANopen device node ID. 0: All NMT slave devices 1 to 127: Destination device node ID

Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
CONFIRM	Completion flag of the function block	BOOL	Depends on data types	-	Completion flag of the function block
ERROR	Error information of function block	CANOPEN_KERNEL_ERROR	Depends on data types	-	Error information of function block

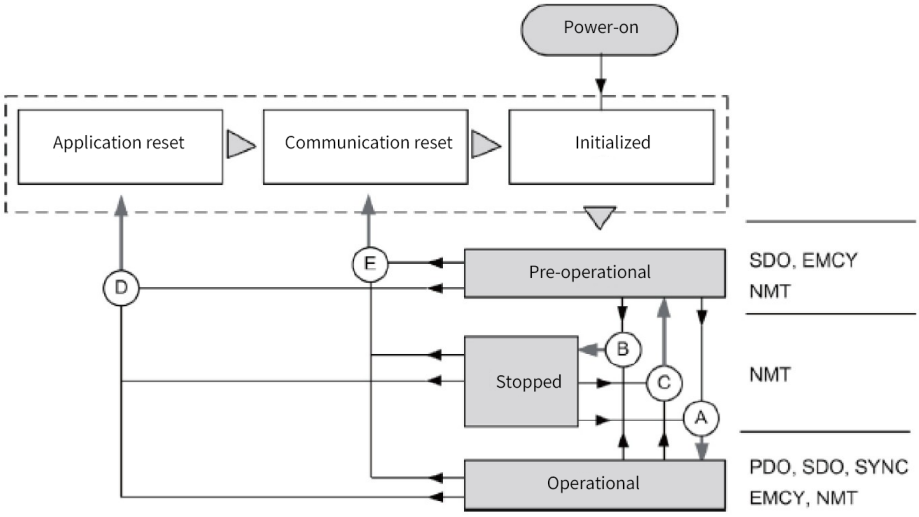
	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
NETWORK	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-
ENABLE	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
TIMEOUT	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
STATE	TRANSITION_STATE																			
DEVICE	DEVICE																			
CONFIRM	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
ERROR	CANOPEN_KERNEL_ERROR																			

■ Function

The NMT state machine describes initialization and state of NMT slaves.

The following figure shows the NMT state, associated communication objects available (PDO, SDO, SYNC, EMCY, and NMT), and transition of five states (A to E).

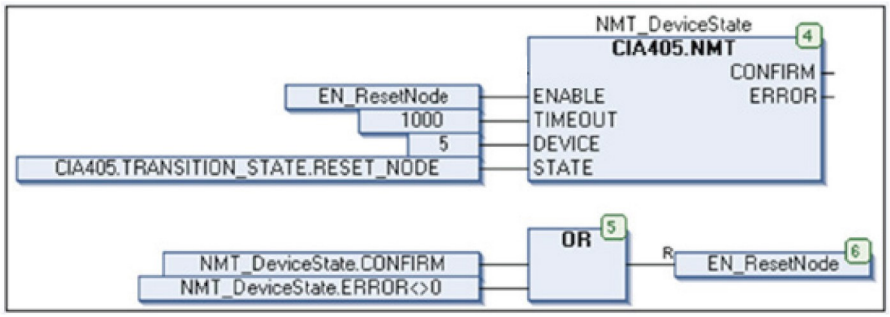


The data type of STATE is enumerated type TRANSITION_STATE. The meanings of the enumerators are as follows.

Enumerator	Meaning
STOP_REMOTE_NODE	Switch to the Stopped state. (Transition B)
START_REMOTE_NODE	Switch to the normal operational state. (Transition A)
RESET_NODE	Switch to the application program reset state. Load saved data of the device configuration file and automatically switch from the communication reset state to pre-operational state. (Transition D)
RESET_COMMUNICATION	Switch to the communication reset state. Load stored data of the communication configuration file and automatically switch from the communication reset state to pre-operational state. (Transition E)
ENTER_PRE_OPERATIONAL	Switch to the pre-operational state. (Transition C)
ALL_EXCEPT_NMT_AND_SENDER	Not implemented (invalid parameter)

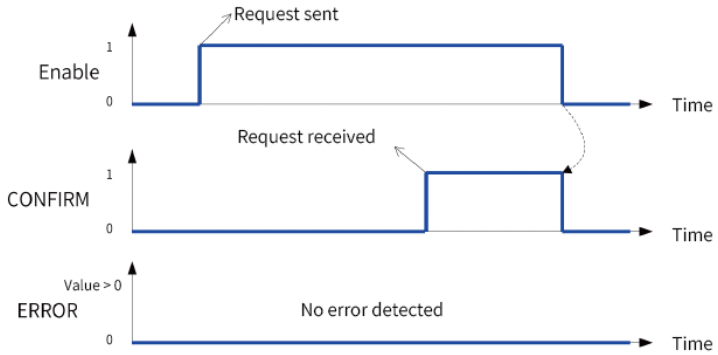
■ Program example

The following example demonstrates how to send a “reset node” command to the CANopen node 5 connected to the first CAN bus interface when the timeout interval is 1s (1,000 ms). The command is sent when the Boolean variable EN_ResetNode is set to TRUE (set by users online or by application programs). When the execution succeeds (CONFIRM = TRUE) or errors are detected (ERROR <> 0), the EN_ResetNode command is reset to FALSE.



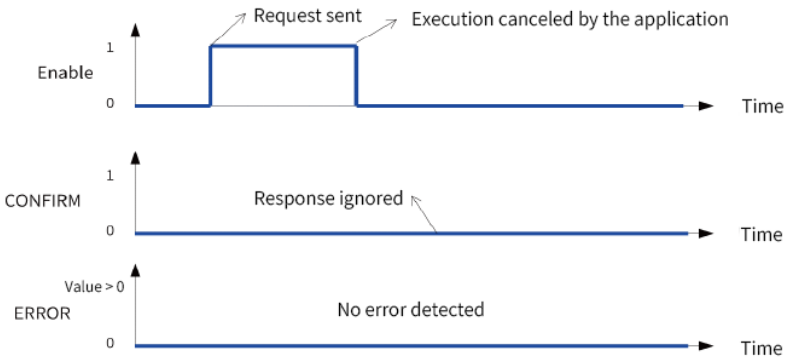
■ Timing diagram

When the system responds to the current request and no errors are detected, CONFIRM is set to TRUE, which is reserved (provided that the function block is called when ENABLE is set to TRUE). If the function block is called when ENABLE is reset to FALSE, CONFIRM is reset to FALSE, and the function block can be executed again.



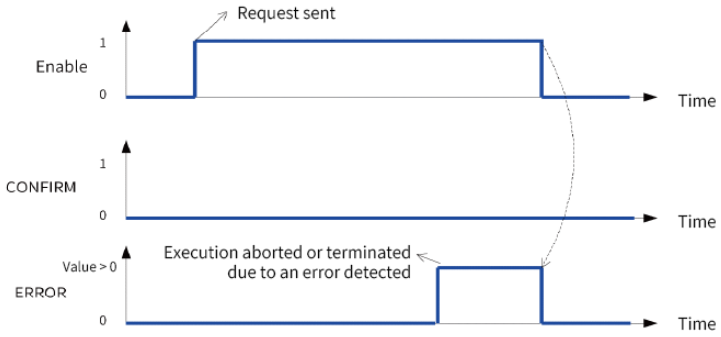
The execution is canceled by application programs.

Before the current execution ends, if the function block is called when ENABLE is reset to FALSE, the function block execution is canceled. The system may have responded to the canceled request or the response will be received later. However, the response is ignored.



Execution is aborted or terminated due to an error detected.

Once the current execution is aborted due to receiving an SDO abort message or terminated due to an error detected, ERROR is set to a value other than 0. For more information about the detected error codes, see the following table. If the function block is called when ENABLE is reset to FALSE, ERROR is reset to 0, and the function block can be executed again.



■ Error description

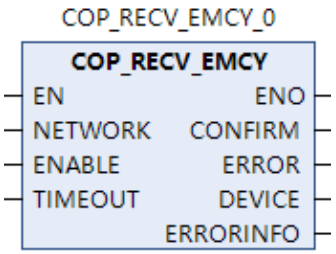
The data type of ERROR is enumerated type CANOPEN_KERNEL_ERROR. The meanings of the enumerators are as follows.

Error Code	Meaning	Description
0	CANOPEN_KERNEL_NO_ERROR	No error codes are detected by the CANopen kernel.
1	CANOPEN_KERNEL_OTHER_ERROR	If the value of ERROR is 01 in hexadecimal format, the error is detected. If the function block involves any other error codes, the output contains more detailed information. Example: CIA405.SDO_READ and CIA405.SDO_WRITE function blocks: The output of ERRORINFO contains the content of the SDO abort message. CIA405.RECV_EMCY and CIA405.RECV_EMCY_DEV function blocks: The output of ERRORINFO contains the content of the received EMCY message. If the ERRORINFO output includes emergency messages, ERROR is 1 and CONFIRM is 0. CIA405.GET_CANOPEN_KERNEL_STATE function block: No hexadecimal 01 value is provided because no other error codes are output.
2	CANOPEN_KERNEL_DATA_OVERFLOW	The CANopen object sending buffer or receiving buffer overflows.
3	CANOPEN_KERNEL_TIMEOUT	The function block execution timed out.
16	CANOPEN_KERNEL_CANBUS_OFF	CANopen nodes are disconnected from the CAN bus.
17	CANOPEN_KERNEL_CAN_ERROR_PASSIVE	CANopen nodes in passive state: Nodes can communicate but are not allowed to send motion error flags when errors are detected.
33	CANOPEN_INTERNAL_FB_ERROR	Specific manufacturer error codes. Note: 21 (hexadecimal) to FF (hexadecimal) are dedicated for device manufacturers.

6.3. 15 COP_RECV_EMCY

This instruction scans EMCY messages.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
COP_RECV_EMCY	EMCY message scan	FB		COP_RECV_EMCY_0(NETWORK:=, ENABLE :=, TIMEOUT :=, CONFIRM =>, ERROR =>, DEVICE =>, ERRORINFO =>,);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
NETWORK	CAN network number	USINT	Depends on data types	-	CAN port executed by the function block, corresponding to the CAN port number of the host
ENABLE	Enable	BOOL	Depends on data types	-	(Triggered by level) Execution of the function block switch
TIMEOUT	Maximum execution time	UDINT	Depends on data types	-	Maximum execution time, in the unit of ms If the execution times out before the response is received, the execution aborts due to timeout. 0 (default): Timeout disabled 1 to 65535: Timeout value This variable is not available.

Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
CONFIRM	Completion flag of the function block	BOOL	Depends on data types	-	Completion flag of the function block
ERROR	Error information of function block	CANOPEN_KERNEL_ERROR	Depends on data types	-	Error information of function block
DEVICE	Device node ID	DEVICE	Depends on data types	-	Device node ID When the EMCY message is received, the value is the node ID of the sending device. When no EMCY message is received, the value is 0.
ERRORINFO	EMCY information	HC_tagEmcyError	Depends on data types	-	EMCY information

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
NETWORK	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-
ENABLE	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
TIMEOUT	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-

CONFIRM	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
ERROR	CANOPEN_KERNEL_ERROR																			
DEVICE	DEVICE																			
ERRORINFO	HC_tagEmcyError																			

■ Function

This function block scans the EMCY messages in all existing CANopen devices in a loop and returns the found EMCY messages. After the function block is enabled, EMCY storage scan starts from the last scan stop point.

If an EMCY message without errors is found, scan ends at this point, and the function block returns the EMCY message and the node ID of an associated device.

If an EMCY message indicating no error is found but the message was not a “No Error” message, scan ends at this point, and the function block returns an EMCY message whose value is 0 and the node ID of an associated device.

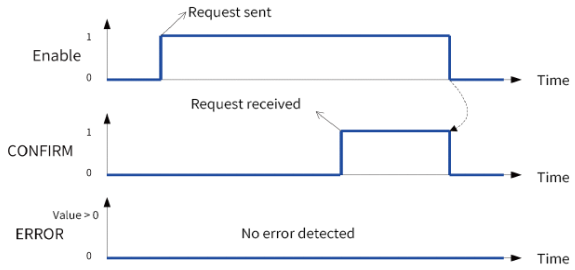
If there are no EMCY messages and no new no-error EMCY messages are found during the complete scan process, the function block returns an EMCY message with the value of 0 and a device node ID of 0.

The data type of ERRORINFO is a structure HC_tagEmcyError. The meanings of the structure members are as follows.

Member	Data Type	Meaning
EMCY_ERROR_CODE	WORD	Error code of an EMCY message
ERROR_REGISTER	BYTE	Error register of an EMCY message (bit field)
ERROR_FIELD	ARRAY[0..4] OF BYTE	Specific device manufacturer error field of an EMCY message

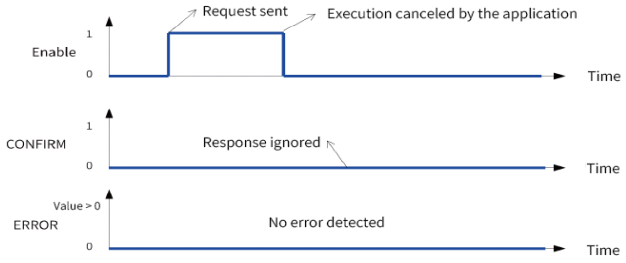
■ Timing diagram

When the system responds to the current request and no errors are detected, CONFIRM is set to TRUE, which is reserved (provided that the function block is called when ENABLE is set to TRUE). If the function block is called when ENABLE is reset to FALSE, CONFIRM is reset to FALSE, and the function block can be executed again.



The execution is canceled by application programs.

Before the current execution ends, if the function block is called when ENABLE is reset to FALSE, the function block execution is canceled. The system may have responded to the canceled request or the response will be received later. However, the response is ignored.



Execution is aborted or terminated due to an error detected.

Once the current execution is aborted due to receiving an SDO abort message or terminated due to an error detected, ERROR is set to a value other than 0. For more information about the detected error codes, see the following table. If the function block is called when ENABLE is reset to FALSE, ERROR is reset to 0, and the function block can be executed again.



■ Error description

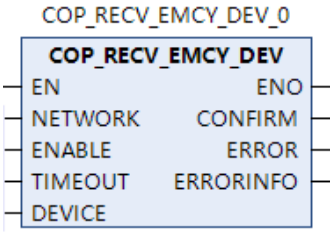
The data type of ERROR is enumerated type CANOPEN_KERNEL_ERROR. The meanings of the enumerators are as follows.

Error Code	Meaning	Description
0	CANOPEN_KERNEL_NO_ERROR	No error codes are detected by the CANopen kernel.
1	CANOPEN_KERNEL_OTHER_ERROR	If the value of ERROR is 01 in hexadecimal format, the error is detected. If the function block involves any other error codes, the output contains more detailed information. Example: CIA405.SDO_READ and CIA405.SDO_WRITE function blocks: The output of ERRORINFO contains the content of the SDO abort message. CIA405.RECV_EMCY and CIA405.RECV_EMCY_DEV function blocks: The output of ERRORINFO contains the content of the received EMCY message. If the output of ERRORINFO contains the EMCY message, the output of ERROR is 1 and the output of CONFIRM is 0. CIA405.GET_CANOPEN_KERNEL_STATE function block: No hexadecimal 01 value is provided because no other error codes are output.
2	CANOPEN_KERNEL_DATA_ OVERFLOW	The CANopen object sending buffer or receiving buffer overflows.
3	CANOPEN_KERNEL_TIMEOUT	The function block execution timed out.
16	CANOPEN_KERNEL_CANBUS_OFF	CANopen nodes are disconnected from the CAN bus.
17	CANOPEN_KERNEL_CAN_ERROR_ PASSIVE	CANopen nodes in passive state: Nodes can communicate but are not allowed to send motion error flags when errors are detected.
33	CANOPEN_INTERNAL_FB_ERROR	Specific manufacturer error codes. Note: 21 (hexadecimal) to FF (hexadecimal) are dedicated for device manufacturers.

6.3. 16 COP_RECV_EMCY_DEV

This instruction receives the EMCY message of the device.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
COP_RECV_EMCY_DEV	Obtain device EMCY messages	FB		COP_RECV_EMCY_DEV_0(NETWORK:=, ENABLE :=, TIMEOUT :=, DEVICE :=, CONFIRM =>, ERROR =>, ERRORINFO =>,);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
NETWORK	CAN network number	USINT	Depends on data types	-	CAN port executed by the function block, corresponding to the CAN port number of the host
ENABLE	Enable	BOOL	Depends on data types	-	(Triggered by level) Execution of the function block switch
TIMEOUT	Maximum execution time	UDINT	Depends on data types	-	Maximum execution time, in the unit of ms If the execution times out before the response is received, the execution aborts due to timeout. 0 (default): Timeout disabled 1 to 65535: Timeout value This variable is not available.
DEVICE	Destination device node ID	DEVICE	Depends on data types	-	Target CANopen device node ID. 0: All NMT slave devices 1 to 127: Destination device node ID

Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
CONFIRM	Completion flag of the function block	BOOL	Depends on data types	-	Completion flag of the function block
ERROR	Error information of function block	CANOPEN_KERNEL_ERROR	Depends on data types	-	Error information of function block
ERRORINFO	EMCY information	HC_tagEmcyError	Depends on data types	-	EMCY information

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
NETWORK	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
ENABLE	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
TIMEOUT	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-
DEVICE	DEVICE																			
CONFIRM	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
ERROR	CANOPEN_KERNEL_ERROR																			
ERRORINFO	HC_tagEmcyError																			

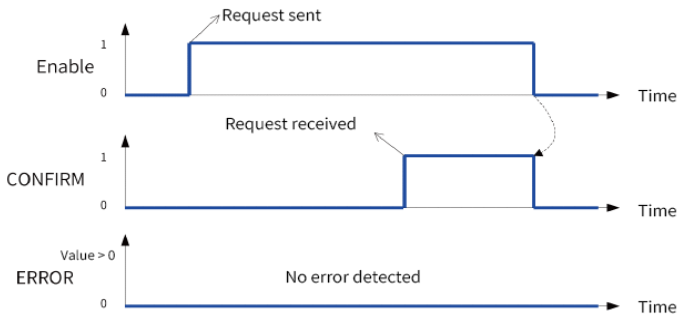
■ Function

This function block returns the last EMCY message received from the specified CANopen device. The data type of ERRORINFO is a structure HC_tagEmcyError. The meanings of the structure members are as follows.

Member	Data Type	Meaning
EMCY_ERROR_CODE	WORD	Error code of an EMCY message
ERROR_REGISTER	BYTE	Error register of an EMCY message (bit field)
ERROR_FIELD	ARRAY[0..4] OF BYTE	Specific device manufacturer error field of an EMCY message

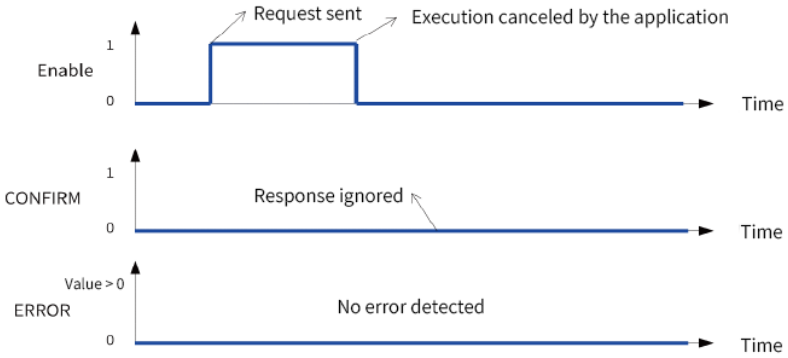
■ Timing diagram

When the system responds to the current request and no errors are detected, CONFIRM is set to TRUE, which is reserved (provided that the function block is called when ENABLE is set to TRUE). If the function block is called when ENABLE is reset to FALSE, CONFIRM is reset to FALSE, and the function block can be executed again.



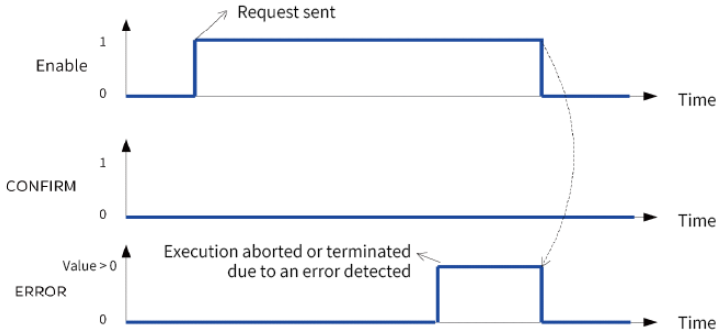
The execution is canceled by application programs.

Before the current execution ends, if the function block is called when ENABLE is reset to FALSE, the function block execution is canceled. The system may have responded to the canceled request or the response will be received later. However, the response is ignored.



Execution is aborted or terminated due to an error detected.

Once the current execution is aborted due to receiving an SDO abort message or terminated due to an error detected, ERROR is set to a value other than 0. For more information about the detected error codes, see the following table. If the function block is called when ENABLE is reset to FALSE, ERROR is reset to 0, and the function block can be executed again.



■ Error description

The data type of ERROR is enumerated type CANOPEN_KERNEL_ERROR. The meanings of the enumerators are as follows.

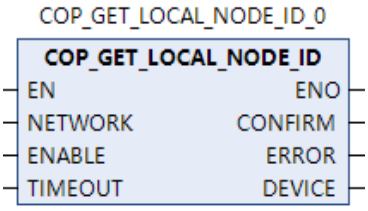
Error Code	Meaning	Description
0	CANOPEN_KERNEL_NO_ERROR	No error codes are detected by the CANopen kernel.
1	CANOPEN_KERNEL_OTHER_ERROR	If the value of ERROR is 01 in hexadecimal format, the error is detected. If the function block involves any other error codes, the output contains more detailed information. Example: CIA405.SDO_READ and CIA405.SDO_WRITE function blocks: The output of ERRORINFO contains the content of the SDO abort message. CIA405.RECV_EMCY and CIA405.RECV_EMCY_DEV function blocks: The output of ERRORINFO contains the content of the received EMCY message. If the output of ERRORINFO contains the EMCY message, the output of ERROR is 1 and the output of CONFIRM is 0. CIA405.GET_CANOPEN_KERNEL_STATE function block: No hexadecimal 01 value is provided because no other error codes are output.
2	CANOPEN_KERNEL_DATA_OVERFLOW	The CANopen object sending buffer or receiving buffer overflows.
3	CANOPEN_KERNEL_TIMEOUT	The function block execution timed out.
16	CANOPEN_KERNEL_CANBUS_OFF	CANopen nodes are disconnected from the CAN bus.
17	CANOPEN_KERNEL_CAN_ERROR_PASSIVE	CANopen nodes in passive state: Nodes can communicate but are not allowed to send motion error flags when errors are detected.

Error Code	Meaning	Description
33	CANOPEN_INTERNAL_FB_ERROR	Specific manufacturer error codes. Note: 21 (hexadecimal) to FF (hexadecimal) are dedicated for device manufacturers.

6.3. 17 COP_GET_LOCAL_NODE_ID

This instruction gets the CANopen node ID of the PLC.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
COP_GET_LOCAL_NODE_ID	Get the CANopen node ID of the PLC	FB		COP_GET_LOCAL_NODE_ID_0(NETWORK:=, ENABLE :=, TIMEOUT :=, CONFIRM =>, ERROR =>, DEVICE =>,);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
NETWORK	CAN network number	USINT	Depends on data types	-	CAN port executed by the function block, corresponding to the CAN port number of the host
ENABLE	Enable	BOOL	Depends on data types	-	(Triggered by level) Execution of the function block switch
TIMEOUT	Maximum execution time	UDINT	Depends on data types	-	Maximum execution time, in the unit of ms If the execution times out before the response is received, the execution aborts due to timeout. 0 (default): Timeout disabled 1 to 65535: Timeout value This variable is not available.

Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
CONFIRM	Completion flag of the function block	BOOL	Depends on data types	-	Completion flag of the function block
ERROR	Error information of function block	CANOPEN_KERNEL_ERROR	Depends on data types	-	Error information of function block
DEVICE	PLC CANopen node ID	DEVICE	Depends on data types	-	PLC CANopen node ID 0: invalid 1 to 127: PLC Node ID

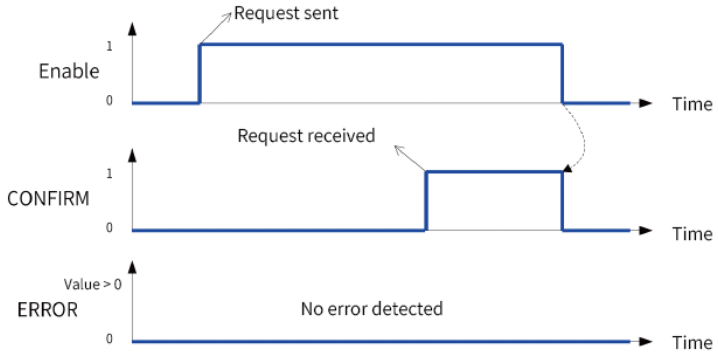
	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
NETWORK	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-
ENABLE	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
TIMEOUT	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-
CONFIRM	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
ERROR	CANOPEN_KERNEL_ERROR																			
DEVICE	DEVICE																			

■ Function

This function block returns the PLC CANopen node ID on the specified CAN bus interface.

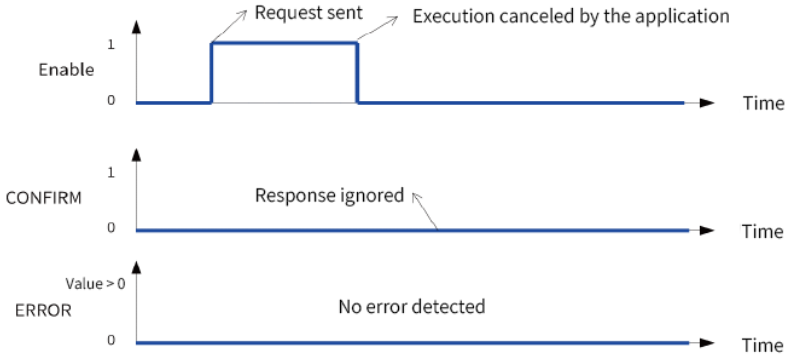
■ Timing diagram

When the system responds to the current request and no errors are detected, CONFIRM is set to TRUE, which is reserved (provided that the function block is called when ENABLE is set to TRUE). If the function block is called when ENABLE is reset to FALSE, CONFIRM is reset to FALSE, and the function block can be executed again.



The execution is canceled by application programs.

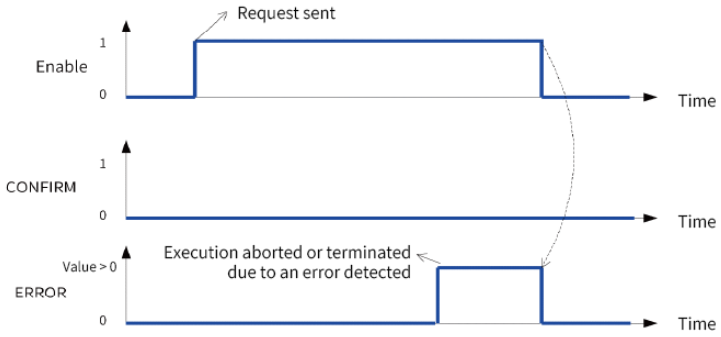
Before the current execution ends, if the function block is called when ENABLE is reset to FALSE, the function block execution is canceled. The system may have responded to the canceled request or the response will be received later. However, the response is ignored.



Execution is aborted or terminated due to an error detected.

Once the current execution is aborted due to receiving an SDO abort message or terminated due to an error detected, ERROR is set to a value other than 0. For more information about the detected error codes, see the following table. If the function block is called when ENABLE is reset to FALSE, ERROR is

reset to 0, and the function block can be executed again.



■ Error description

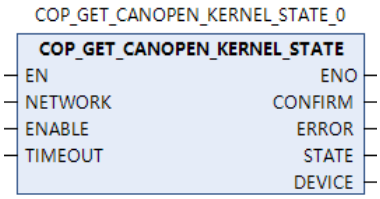
The data type of ERROR is enumerated type CANOPEN_KERNEL_ERROR. The meanings of the enumerators are as follows.

Error Code	Meaning	Description
0	CANOPEN_KERNEL_NO_ERROR	No error codes are detected by the CANopen kernel.
1	CANOPEN_KERNEL_OTHER_ERROR	If the value of ERROR is 01 in hexadecimal format, the error is detected. If the function block involves any other error codes, the output contains more detailed information. Example: CIA405.SDO_READ and CIA405.SDO_WRITE function blocks: The output of ERRORINFO contains the content of the SDO abort message. CIA405.RECV_EMCY and CIA405.RECV_EMCY_DEV function blocks: The output of ERRORINFO contains the content of the received EMCY message. If the ERRORINFO output includes emergency messages, ERROR is 1 and CONFIRM is 0. CIA405.GET_CANOPEN_KERNEL_STATE function block: No hexadecimal 01 value is provided because no other error codes are output.
2	CANOPEN_KERNEL_DATA_OVERFLOW	The CANopen object sending buffer or receiving buffer overflows.
3	CANOPEN_KERNEL_TIMEOUT	The function block execution timed out.
16	CANOPEN_KERNEL_CANBUS_OFF	CANopen nodes are disconnected from the CAN bus.
17	CANOPEN_KERNEL_CAN_ERROR_PASSIVE	CANopen nodes in passive state: Nodes can communicate but are not allowed to send motion error flags when errors are detected.
33	CANOPEN_INTERNAL_FB_ERROR	Specific manufacturer error codes. Note: 21 (hexadecimal) to FF (hexadecimal) are dedicated for device manufacturers.

6.3. 18 COP_GET_CANOPEN_KERNEL_STATE

This instruction gets the CANopen kernel state. (Functions of this function block are not implemented.)

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
COP_GET_CANOPEN_ KERNEL_STATE	Get the CANopen kernel state	FB		COP_GET_CANOPEN_ KERNEL_ STATE_0(NETWORK:=, ENABLE :=, TIMEOUT :=, CONFIRM =>, ERROR =>, STATE =>, DEVICE =>,);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
NETWORK	CAN network number	USINT	Depends on data types	-	CAN port executed by the function block, corresponding to the CAN port number of the host
ENABLE	Enable	BOOL	Depends on data types	-	(Triggered by level) Execution of the function block switch
TIMEOUT	Maximum execution time	UDINT	Depends on data types	-	Maximum execution time, in the unit of ms If the execution times out before the response is received, the execution aborts due to timeout. 0 (default): Timeout disabled 1 to 65535: Timeout value

Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
CONFIRM	Completion flag of the function block	BOOL	Depends on data types	-	Completion flag of the function block
ERROR	Error information of function block	CANOPEN_KERNEL_ ERROR	Depends on data types	-	Error information of function block
STATE	Current state of the PLC CANopen kernel	CANOPEN_KERNEL_ ERROR	Depends on data types	-	Current state of the PLC CANopen kernel
DEVICE	PLC CANopen node ID	DEVICE	Depends on data types	-	PLC CANopen node ID 0: invalid 1 to 127: PLC Node ID

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
NETWORK	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-
ENABLE	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

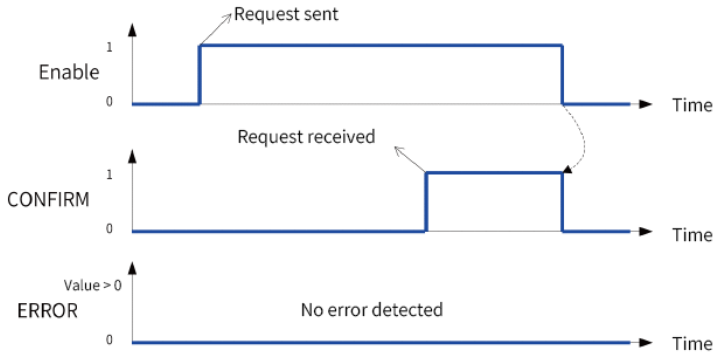
	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
TIMEOUT	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-
CONFIRM	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
ERROR	CANOPEN_KERNEL_ERROR																			
STATE	CANOPEN_KERNEL_ERROR																			
DEVICE	DEVICE																			

■ Function

This function block returns the current state of the PLC CANopen kernel.

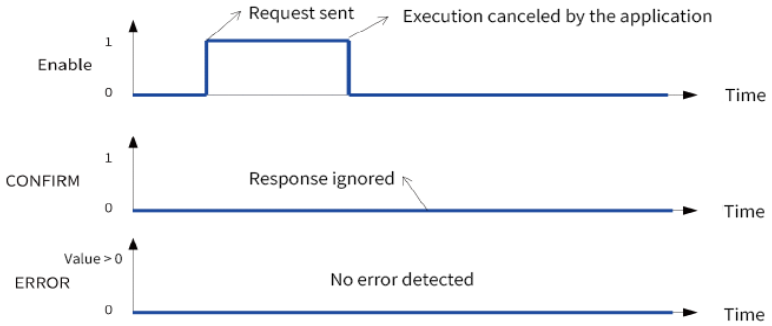
■ Timing diagram

When the system responds to the current request and no errors are detected, CONFIRM is set to TRUE, which is reserved (provided that the function block is called when ENABLE is set to TRUE). If the function block is called when ENABLE is reset to FALSE, CONFIRM is reset to FALSE, and the function block can be executed again.



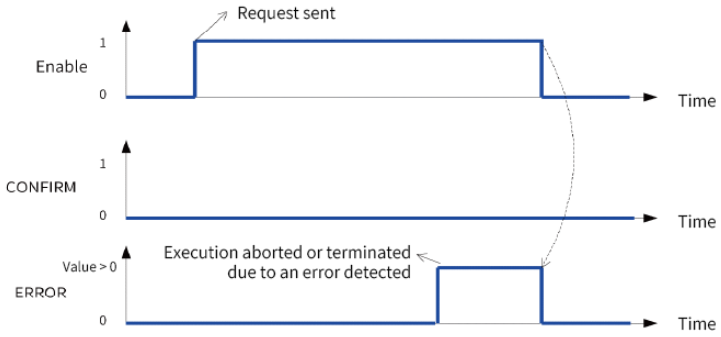
The execution is canceled by application programs.

Before the current execution ends, if the function block is called when ENABLE is reset to FALSE, the function block execution is canceled. The system may have responded to the canceled request or the response will be received later. However, the response is ignored.



Execution is aborted or terminated due to an error detected.

Once the current execution is aborted due to receiving an SDO abort message or terminated due to an error detected, ERROR is set to a value other than 0. For more information about the detected error codes, see the following table. If the function block is called when ENABLE is reset to FALSE, ERROR is reset to 0, and the function block can be executed again.



■ Error description

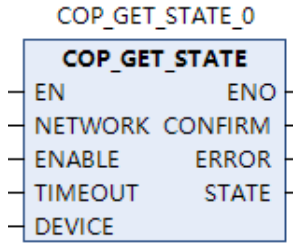
The data type of ERROR is enumerated type CANOPEN_KERNEL_ERROR. The meanings of the enumerators are as follows.

Error Code	Meaning	Description
0	CANOPEN_KERNEL_NO_ERROR	No error codes are detected by the CANopen kernel.
1	CANOPEN_KERNEL_OTHER_ERROR	If the value of ERROR is 01 in hexadecimal format, the error is detected. If the function block involves any other error codes, the output contains more detailed information. Example: CIA405.SDO_READ and CIA405.SDO_WRITE function blocks: The output of ERRORINFO contains the content of the SDO abort message. CIA405.RECV_EMCY and CIA405.RECV_EMCY_DEV function blocks: The output of ERRORINFO contains the content of the received EMCY message. IF If the ERRORINFO output includes emergency messages, ERROR is 1 and CONFIRM is 0. CIA405.GET_CANOPEN_KERNEL_STATE function block: No hexadecimal 01 value is provided because no other error codes are output.
2	CANOPEN_KERNEL_DATA_OVERFLOW	The CANopen object sending buffer or receiving buffer overflows.
3	CANOPEN_KERNEL_TIMEOUT	The function block execution timed out.
16	CANOPEN_KERNEL_CANBUS_OFF	CANopen nodes are disconnected from the CAN bus.
17	CANOPEN_KERNEL_CAN_ERROR_PASSIVE	CANopen nodes in passive state: Nodes can communicate but are not allowed to send motion error flags when errors are detected.
33	CANOPEN_INTERNAL_FB_ERROR	Specific manufacturer error codes. Note: 21 (hexadecimal) to FF (hexadecimal) are dedicated for device manufacturers.

6.3. 19 COP_GET_STATE

This instruction gets the CANopen device state.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
COP_GET_STATE	Get the CANopen device state	FB		COP_GET_STATE_0(NETWORK:=, ENABLE :=, TIMEOUT :=, DEVICE :=, CONFIRM =>, ERROR =>, STATE =>,);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
NETWORK	CAN network number	USINT	Depends on data types	-	CAN port executed by the function block, corresponding to the CAN port number of the host
ENABLE	Enable	BOOL	Depends on data types	-	(Triggered by level) Execution of the function block switch
TIMEOUT	Maximum execution time	UDINT	Depends on data types	-	Maximum execution time, in the unit of ms If the execution times out before the response is received, the execution aborts due to timeout. 0 (default): Timeout disabled 1 to 65535: Timeout value This variable is not available.
DEVICE	Node ID of the CANopen device	DEVICE	Depends on data types	-	Node ID of the CANopen device 0 (default): Local device (PLC) 1 to 127: Device node ID

Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
CONFIRM	Completion flag of the function block	BOOL	Depends on data types	-	Completion flag of the function block
ERROR	Error information of function block	CANOPEN_KERNEL_ERROR	Depends on data types	-	Error information of function block
STATE	Current state of the PLC CANopen kernel	DEVICE_STATE	Depends on data types	-	Current state of the PLC CANopen kernel

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
NETWORK	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-
ENABLE	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
TIMEOUT	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
DEVICE	DEVICE																			
CONFIRM	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
ERROR	CANOPEN_KERNEL_ERROR																			
STATE	DEVICE_STATE																			

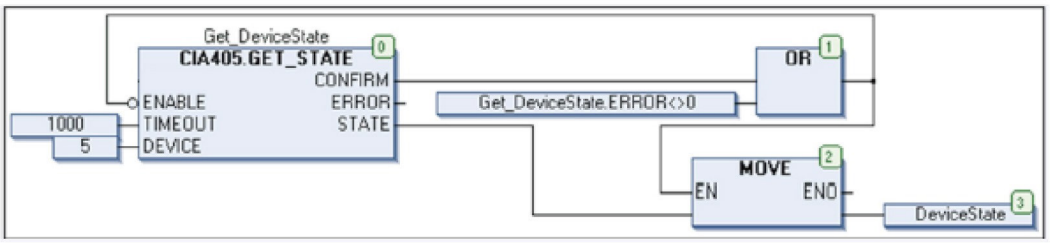
■ Function

This function block returns the current NMT state of the CANopen device when the heartbeat or node protection is in the active state. The data type of STATE is enumerated type DEVICE_STATE. The meanings of the enumerators are as follows.

Enumerator	Meaning
INIT	Initializing state
RESET_COMM	Communication reset state
RESET_APP	Application program reset state
PRE_OPERATIONAL	Pre-operational state
STOPPED	Stopped state
OPERATIONAL	Normal operational state
UNKNOWN	Unknown NMT state The node protection or heartbeat function of the selected device is not activated, or the PLC is not a heartbeat consumer.
NOT_AVAIL	NMT state not available The node protection or heartbeat function of the selected device is activated, but the device fails to report its NMT state correctly before timeout.

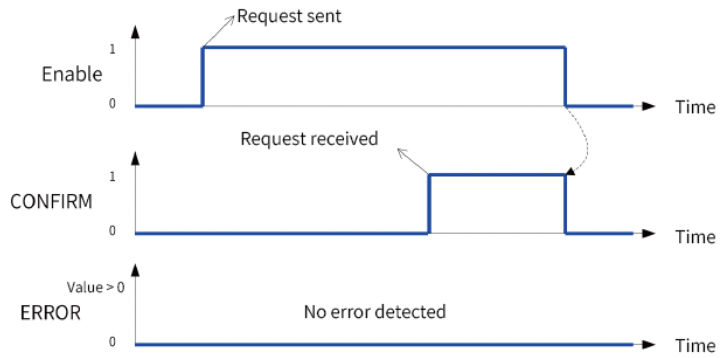
■ Program example

The following example describes how to obtain the state of the CANopen node 5 connected to the first CAN bus interface when the timeout interval is 1s (1,000 ms). The CIA405.GET_STATE function is automatically executed to read the state continuously. The device NMT state is copied to DeviceState variables of the CIA405.DEVICE_STATE type.



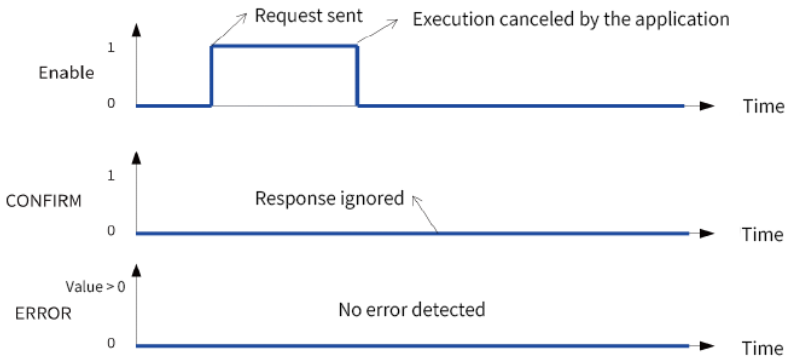
■ Timing diagram

When the system responds to the current request and no errors are detected, CONFIRM is set to TRUE, which is reserved (provided that the function block is called when ENABLE is set to TRUE). If the function block is called when ENABLE is reset to FALSE, CONFIRM is reset to FALSE, and the function block can be executed again.



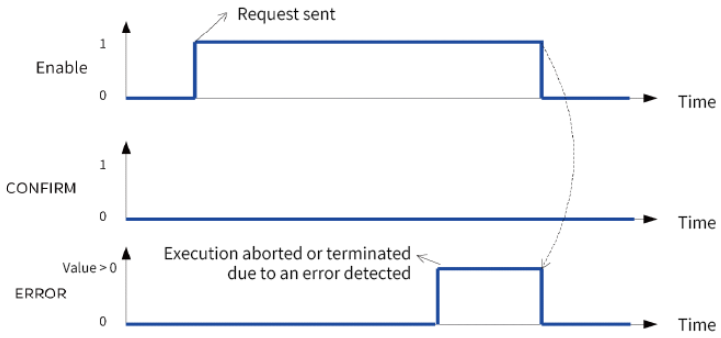
The execution is canceled by application programs.

Before the current execution ends, if the function block is called when ENABLE is reset to FALSE, the function block execution is canceled. The system may have responded to the canceled request or the response will be received later. However, the response is ignored.



The execution is canceled by application programs.

Before the current execution ends, if the function block is called when ENABLE is reset to FALSE, the function block execution is canceled. The system may have responded to the canceled request or the response will be received later. However, the response is ignored.



■ Error description

The data type of ERROR is enumerated type CANOPEN_KERNEL_ERROR. The meanings of the enumerators are as follows.

Error Code	Meaning	Description
0	CANOPEN_KERNEL_NO_ERROR	No error codes are detected by the CANopen kernel.

Error Code	Meaning	Description
1	CANOPEN_KERNEL_OTHER_ERROR	If the value of ERROR is 01 in hexadecimal format, the error is detected. If the function block involves any other error codes, the output contains more detailed information. Example: CIA405.SDO_READ and CIA405.SDO_WRITE function blocks: The output of ERRORINFO contains the content of the SDO abort message. CIA405.RECV_EMCY and CIA405.RECV_EMCY_DEV function blocks: The output of ERRORINFO contains the content of the received EMCY message. If the output of ERRORINFO contains the EMCY message, the output of ERROR is 1 and the output of CONFIRM is 0. CIA405.GET_CANOPEN_KERNEL_STATE function block: No hexadecimal 01 value is provided because no other error codes are output.
2	CANOPEN_KERNEL_DATA_ OVERFLOW	The CANopen object sending buffer or receiving buffer overflows.
3	CANOPEN_KERNEL_TIMEOUT	The function block execution timed out.
16	CANOPEN_KERNEL_CANBUS_OFF	CANopen nodes are disconnected from the CAN bus.
17	CANOPEN_KERNEL_CAN_ERROR_ PASSIVE	CANopen nodes in passive state: Nodes can communicate but are not allowed to send motion error flags when errors are detected.
33	CANOPEN_INTERNAL_FB_ERROR	Specific manufacturer error codes. Note: 21 (hexadecimal) to FF (hexadecimal) are dedicated for device manufacturers.

6.3. 20 COP_SDO_READ

This instruction reads CANopen objects of any size.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
COP_SDO_READ	Read CANopen objects of any size	FB	COP_SDO_READ_0 COP_SDO_READ EN ENO NETWORK CONFIRM ENABLE ERROR TIMEOUT ERRORINFO DEVICE CHANNEL INDEX SUBINDEX DATA DATALENGTH	COP_SDO_READ_0(NETWORK:=, ENABLE :=, TIMEOUT :=, DEVICE :=, CHANNEL :=, INDEX := , SUBINDEX := , DATA := , DATALENGTH :=, CONFIRM =>, ERROR =>, ERRORINFO =>,);

■ Variables

In-out variables

In-Out Variable	Name	Data Type	Value Range	Initial Value	Description
DATALENGTH	Data buffer size	UINT	Depends on data types	-	Input: Data buffer size (in the unit of bytes) Note: Use the SIZEOF standard function to initialize the DATALENGTH input correctly (when the function block execution is started at the rising edge through which ENABLE is input) to match the data buffer size. Output: Size of a read object (in the unit of bytes)

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
NETWORK	CAN network number	USINT	Depends on data types	-	CAN port executed by the function block, corresponding to the CAN port number of the host
ENABLE	Enable	BOOL	Depends on data types	-	(Triggered by level) Execution of the function block switch
TIMEOUT	Maximum execution time	UDINT	Depends on data types	-	Maximum execution time, in the unit of ms If the execution times out before the response is received, the execution aborts due to timeout. 0 (default): Timeout disabled 1 to 65535: Timeout value This variable is not available.
DEVICE	Node ID of the CANopen device	DEVICE	Depends on data types	-	Node ID of the CANopen device 0 (default): Local device (PLC) 1 to 127: Device node ID
CHANNEL	SDO channel number	USINT	Depends on data types	-	SDO channel number The number is 1 by default.
INDEX	Object index	WORD	Depends on data types	-	Object index The value ranges from 0000 (hexadecimal) to FFFF (hexadecimal).
SUBINDEX	Object sub-index	BYTE	Depends on data types	-	Object sub-index The value ranges from 00 (hexadecimal) to FF (hexadecimal).
DATA	Address of the data buffer to read data	POINTER TO BYTE	Depends on data types	-	Address of the data buffer to receive data read from the device object The ADR standard function must be used to define the associated pointer.

Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
CONFIRM	Completion flag of the function block	BOOL	Depends on data types	-	Completion flag of the function block
ERROR	Error information of function block	CANOPEN_KERNEL_ERROR	Depends on data types	-	Error information of function block

Output Variable	Name	Data Type	Value Range	Initial Value	Description
ERRORINFO	SDO abort message content	SDO_ERROR	Depends on data types	-	When ERROR is 1, the SDO abort message content (four bytes) is returned.

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
NETWORK	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-
ENABLE	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
TIMEOUT	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-
DEVICE	DEVICE																			
CHANNEL	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-
INDEX	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
SUBINDEX	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
DATA	POINTER TO BYTE																			
DATALENGTH	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-
CONFIRM	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
ERROR	CANOPEN_KERNEL_ERROR																			
ERRORINFO	SDO_ERROR																			

■ Function

This function block reads a device-specific CANopen object of any size from the SDO message.

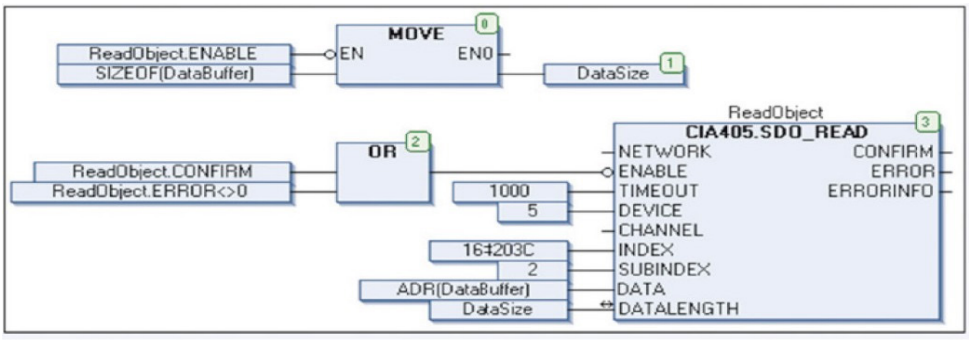
■ Program example

The following example describes how to read the object index 203C (hexadecimal)/sub-index 02 (hexadecimal) of the CANopen node 5 connected to the first CAN bus interface when the timeout interval is 1s (1,000 ms). The CIA405.SDO_READ function block instance (ReadObject) is automatically executed for continuous reading.

Variable DataSize (UINT type):

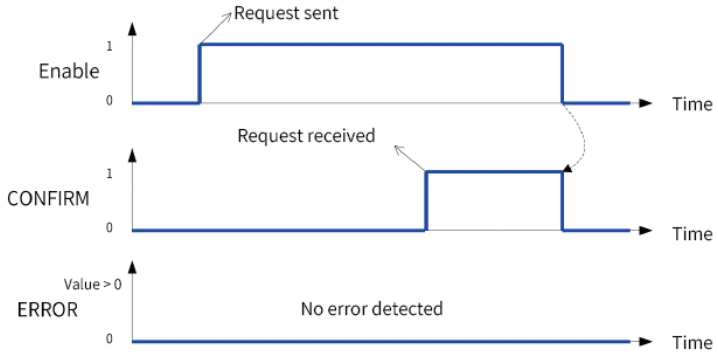
It is initialized to the data buffer size (DataBuffer: N-byte array) (when ENABLE is FALSE and before the next execution is started).

The size of data read at the rising edge through which CONFIRM is output (in the unit of bytes) (the example does not demonstrate how to extract values from the data buffer or how to manage error detection) is included.



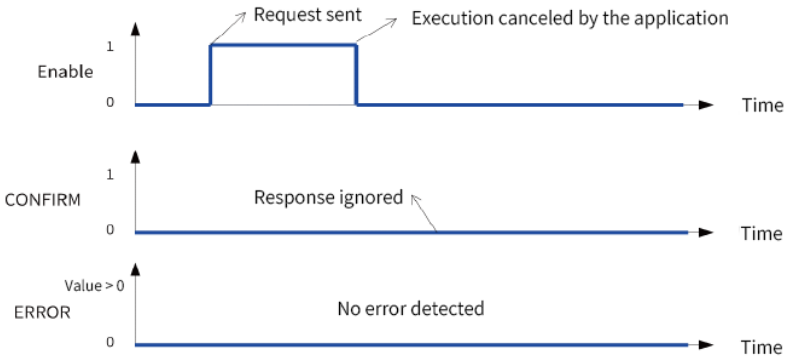
■ Timing diagram

When the system responds to the current request and no errors are detected, CONFIRM is set to TRUE, which is reserved (provided that the function block is called when ENABLE is set to TRUE). If the function block is called when ENABLE is reset to FALSE, CONFIRM is reset to FALSE, and the function block can be executed again.



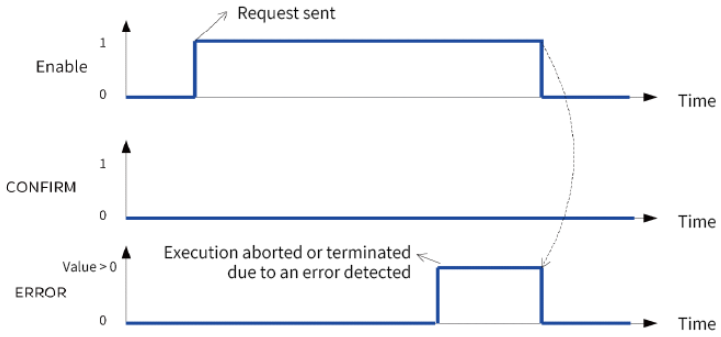
The execution is canceled by application programs.

Before the current execution ends, if the function block is called when ENABLE is reset to FALSE, the function block execution is canceled. The system may have responded to the canceled request or the response will be received later. However, the response is ignored.



Execution is aborted or terminated due to an error detected.

Once the current execution is aborted due to receiving an SDO abort message or terminated due to an error detected, ERROR is set to a value other than 0. For more information about the detected error codes, see the following table. If the function block is called when ENABLE is reset to FALSE, ERROR is reset to 0, and the function block can be executed again.



■ Precautions

The following specific parameters must be transferred to this function block.

1) Device node ID

2) SDO client/server channel (only one channel is defined by default)

3) CANopen object index/sub-index

4) Pointer of the data buffer used to store the object value

5) Data buffer size

If the read operation is successful, the function block returns the size of the object read. The data is provided in the data buffer. If the size of the object to be read is less than or equal to 4 bytes, it is recommended to use the CIA405.SDO_READ4 function block.

6.3. 21 COP_SDO_READ4

This instruction reads a CANopen object of up to four bytes.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
COP_SDO_READ4	Read CANopen objects of no more than four bytes	FB	COP_SDO_READ4_0 COP_SDO_READ4 EN ENO NETWORK CONFIRM ENABLE ERROR TIMEOUT DATA DEVICE DATALENGTH CHANNEL ERRORINFO INDEX SUBINDEX	COP_SDO_READ4_0(NETWORK:=, ENABLE :=, TIMEOUT :=, DEVICE :=, CHANNEL :=, INDEX := , SUBINDEX := , CONFIRM =>, ERROR =>, DATA => , DATALENGTH=>, ERRORINFO =>,);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
NETWORK	CAN network number	USINT	Depends on data types	-	CAN port executed by the function block, corresponding to the CAN port number of the host

Input Variable	Name	Data Type	Value Range	Initial Value	Description
ENABLE	Enable	BOOL	Depends on data types	-	(Triggered by level) Execution of the function block switch
TIMEOUT	Maximum execution time	UDINT	Depends on data types	-	Maximum execution time, in the unit of ms If the execution times out before the response is received, the execution aborts due to timeout. 0 (default): Timeout disabled 1 to 65535: Timeout value This variable is not available.
DEVICE	Node ID of the CANopen device	DEVICE	Depends on data types	-	Node ID of the CANopen device 0 (default): Local device (PLC) 1 to 127: Device node ID
CHANNEL	SDO channel number	USINT	Depends on data types	-	SDO channel number The number is 1 by default.
INDEX	Object index	WORD	Depends on data types	-	Object index The value ranges from 0000 (hexadecimal) to FFFF (hexadecimal).
SUBINDEX	Object sub-index	BYTE	Depends on data types	-	Object sub-index The value ranges from 00 (hexadecimal) to FF (hexadecimal).

Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
CONFIRM	Completion flag of the function block	BOOL	Depends on data types	-	Completion flag of the function block
ERROR	Error information of function block	CANOPEN_KERNEL_ERROR	Depends on data types	-	Error information of function block
DATA	Receive data	ARRAY[1...4] OF BYTE	Depends on data types	-	Array to receive data read from the device object
DATALENGTH	Size of the read object	USINT	Depends on data types	-	Size of a read object (in the unit of bytes)
ERRORINFO	SDO abort message content	SDO_ERROR	Depends on data types	-	When ERROR is 1, the SDO abort message content (four bytes) is returned.

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
NETWORK	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-
ENABLE	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
TIMEOUT	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-
DEVICE	DEVICE																			
CHANNEL	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-
INDEX	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
SUBINDEX	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
CONFIRM	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
ERROR	CANOPEN_KERNEL_ERROR																			
DATA	ARRAY[1...4] OF BYTE																			
DATALENGTH	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-
ERRORINFO	SDO_ERROR																			

■ Function

This function block reads a device-specific CANopen object of up to four bytes from the SDO message. The following table lists object sizes and corresponding DATA arrays.

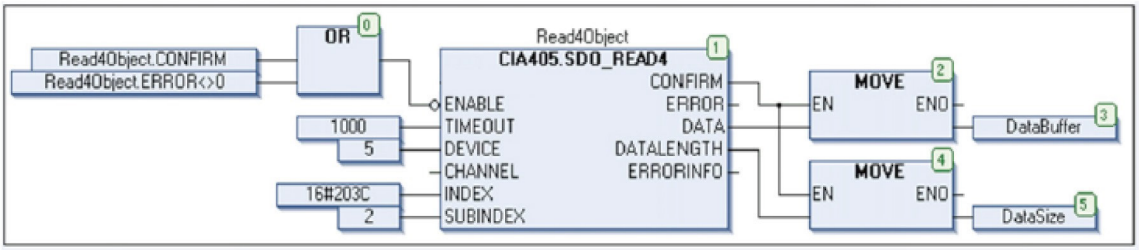
Object Size Example	DATALENGTH	DATA(1)	DATA(2)	DATA(3)	DATA(4)
1-byte 01 (hexadecimal)	1	01 (hexadecimal)	Invalid	Invalid	Invalid
2-byte 01 23 (hexadecimal)	2	LSB 23 (hexadecimal)	MSB 01 (hexadecimal)	Invalid	Invalid
3-byte 01 23 45 (hexadecimal)	3	LSB 45 (hexadecimal)	23 (hexadecimal)	MSB 01 (hexadecimal)	Invalid
4-byte 01 23 45 67 (hexadecimal)	4	LSB 67 (hexadecimal)	45 (hexadecimal)	23 (hexadecimal)	MSB 01 (hexadecimal)

LSB = least significant bit

MSB = most significant bit

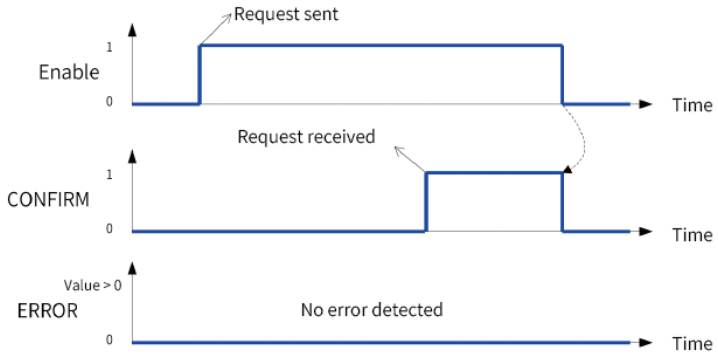
■ Program example

The following example describes how to read the object index 203C (hexadecimal)/sub-index 02 (hexadecimal) of the CANopen node 5 connected to the first CAN bus interface when the timeout interval is 1s (1,000 ms). The CIA405.SDO_READ4 function block instance (Read4Object) is automatically executed for continuous reading. The variable DataBuffer (4-byte array) includes the value of data last read. The variable DataSize (USINT type) includes the size of data last read (maximum: 4 bytes). The example does not demonstrate how to manage error detection.



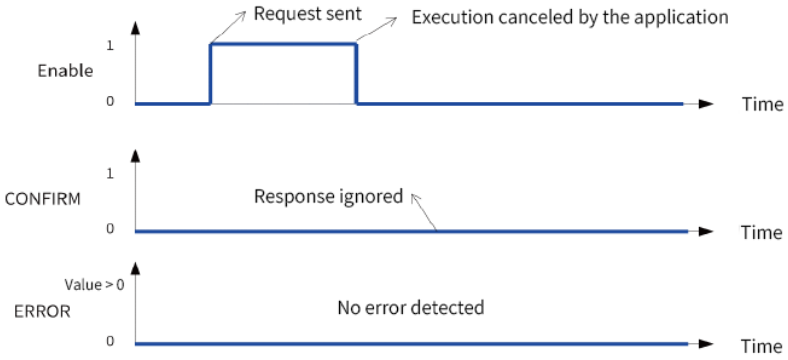
■ Timing diagram

When the system responds to the current request and no errors are detected, CONFIRM is set to TRUE, which is reserved (provided that the function block is called when ENABLE is set to TRUE). If the function block is called when ENABLE is reset to FALSE, CONFIRM is reset to FALSE, and the function block can be executed again.



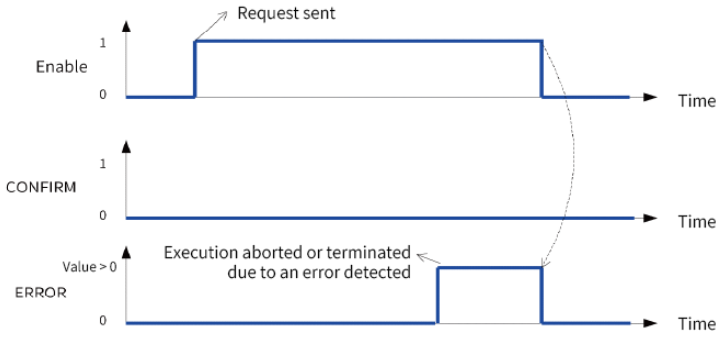
The execution is canceled by application programs.

Before the current execution ends, if the function block is called when ENABLE is reset to FALSE, the function block execution is canceled. The system may have responded to the canceled request or the response will be received later. However, the response is ignored.



Execution is aborted or terminated due to an error detected.

Once the current execution is aborted due to receiving an SDO abort message or terminated due to an error detected, ERROR is set to a value other than 0. For more information about the detected error codes, see the following table. If the function block is called when ENABLE is reset to FALSE, ERROR is reset to 0, and the function block can be executed again.



■ Precautions

The following specific parameters must be transferred to this function block.

1) Device node ID

2) SDO client/server channel (only one channel is defined by default)

3) CANopen object index/sub-index

If data is read successfully, the function block returns the size of the read object. The data is provided in a four-byte array.

6.3. 22 COP_SDO_WRITE

This instruction writes a CANopen object of any size.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
COP_SDO_WRITE	Write CANopen objects of any size	FB	COP_SDO_WRITE_0 COP_SDO_WRITE EN ENO NETWORK CONFIRM ENABLE ERROR TIMEOUT ERRORINFO DEVICE CHANNEL INDEX SUBINDEX MODE DATA DATALENGTH	SDO_WRITE_0(NETWORK:=, ENABLE :=, TIMEOUT :=, DEVICE :=, CHANNEL :=, INDEX := , SUBINDEX := , MODE :=, DATA := , DATALENGTH :=, CONFIRM =>, ERROR =>, ERRORINFO =>,);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
NETWORK	CAN network number	USINT	Depends on data types	-	CAN port executed by the function block, corresponding to the CAN port number of the host
ENABLE	Enable	BOOL	Depends on data types	-	(Triggered by level) Execution of the function block switch

Input Variable	Name	Data Type	Value Range	Initial Value	Description
TIMEOUT	Maximum execution time	UDINT	Depends on data types	-	Maximum execution time, in the unit of ms If the execution times out before the response is received, the execution aborts due to timeout. 0 (default): Timeout disabled 1 to 65535: Timeout value This variable is not available.
DEVICE	Node ID of the CANopen device	DEVICE	Depends on data types	-	Node ID of the CANopen device 0 (default): Local device (PLC) 1 to 127: Device node ID
CHANNEL	SDO channel number	USINT	Depends on data types	-	SDO channel number The number is 1 by default.
INDEX	Object index	WORD	Depends on data types	-	Object index The value ranges from 0000 (hexadecimal) to FFFF (hexadecimal).
SUBINDEX	Object sub-index	BYTE	Depends on data types	-	Object sub-index The value ranges from 00 (hexadecimal) to FF (hexadecimal).
MODE	Data transmission mode	SDO_MODE	Depends on data types	-	Data transmission mode 0 (default) = AUTO (automatic mode selection)
DATA	Address of the data buffer to read data	POINTER TO BYTE	Depends on data types	-	Address of the data buffer to receive data read from the device object The ADR standard function must be used to define the associated pointer.
DATALENGTH	Size of the object to write	UINT	Depends on data types	-	Size of the object to write, in the unit of bytes

Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
CONFIRM	Completion flag of the function block	BOOL	Depends on data types	-	Completion flag of the function block
ERROR	Error information of function block	CANOPEN_KERNEL_ERROR	Depends on data types	-	Error information of function block
ERRORINFO	SDO abort message content	SDO_ERROR	Depends on data types	-	When ERROR is 1, the SDO abort message content (four bytes) is returned.

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
NETWORK	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-
ENABLE	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
TIMEOUT	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-
DEVICE	DEVICE																			
CHANNEL	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-
INDEX	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
SUBINDEX	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
MODE	SDO_MODE																			
DATA	POINTER TO BYTE																			
DATALENGTH	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-
CONFIRM	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
ERROR	CANOPEN_KERNEL_ERROR																			
ERRORINFO	SDO_ERROR																			

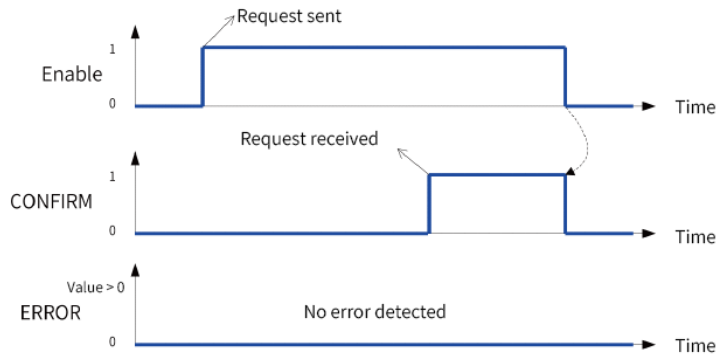
■ Function

This function block writes a device-specific CANopen object of any size into the SDO message. The data type of MODE is enumerated type SDO_MODE. The meanings of the enumerators are as follows.

Enumerator	Meaning
AUTO	Automatic mode selection
EXPEDITED	SDO acceleration mode for data of no more than 4 bytes Data is sent in response to an SDO request.
SEGMENTED	SDO segmentation mode for data of more than 4 bytes Data is segmented into 7-byte segments and sent through continuous SDO confirmation requests.
BLOCK	SDO block mode for data of more than 4 bytes Data sent through continuous frames is segmented into 7-byte data blocks. These blocks are not confirmed. Upon receipt of all blocks, the recipient sends confirmation. Note: This mode allows quick transmission, but your device may not support the mode. Therefore, the mode is a CANopen configuration item recently added.

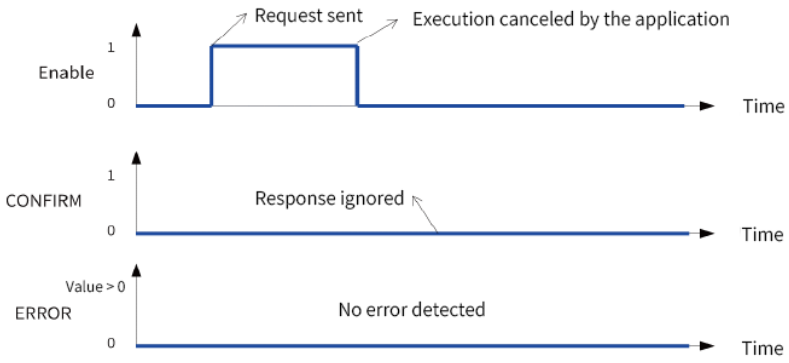
■ Timing diagram

When the system responds to the current request and no errors are detected, CONFIRM is set to TRUE, which is reserved (provided that the function block is called when ENABLE is set to TRUE). If the function block is called when ENABLE is reset to FALSE, CONFIRM is reset to FALSE, and the function block can be executed again.



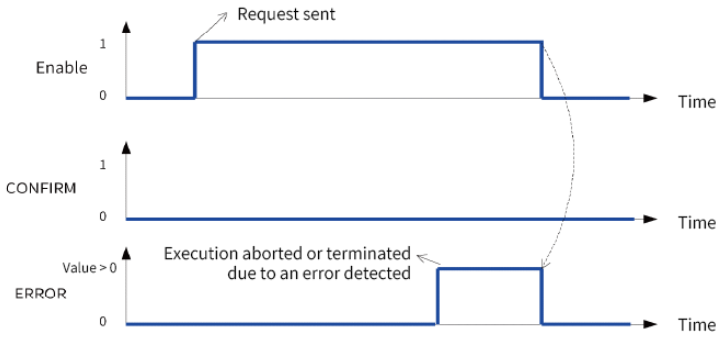
The execution is canceled by application programs.

Before the current execution ends, if the function block is called when ENABLE is reset to FALSE, the function block execution is canceled. The system may have responded to the canceled request or the response will be received later. However, the response is ignored.



Execution is aborted or terminated due to an error detected.

Once the current execution is aborted due to receiving an SDO abort message or terminated due to an error detected, ERROR is set to a value other than 0. For more information about the detected error codes, see the following table. If the function block is called when ENABLE is reset to FALSE, ERROR is reset to 0, and the function block can be executed again.



■ Precautions

The following specific parameters must be transferred to this function block.

1) Data buffer size
2) Device node ID
3) SDO client/server channel (only one channel is defined by default)
4) CANopen object index/sub-index
5) SDO mode
6) Pointer of the data buffer used to store the value of the object to be written

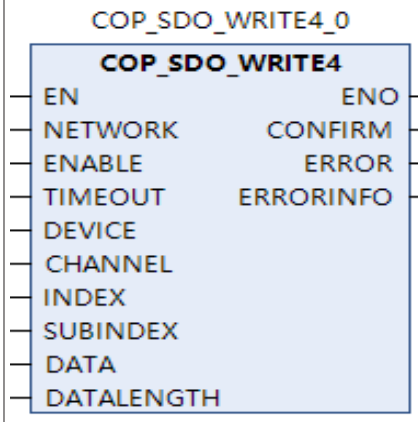
7) Number of bytes to be written

If the size of the object to be written is less than or equal to 4 bytes, use the CIA405.SDO_WRITE4 function block.

6.3. 23 COP_SDO_WRITE4

Write CANopen objects of no more than four bytes

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
COP_SDO_WRITE4	Write CANopen objects of no more than four bytes	FB		COP_SDO_WRITE4_0(ENABLE :=, TIMEOUT :=, DEVICE :=, CHANNEL :=, INDEX := , SUBINDEX := , DATA := , DATALENGTH :=, CONFIRM =>, ERROR =>, ERRORINFO =>,);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
NETWORK	CAN network number	USINT	Depends on data types	-	CAN port executed by the function block, corresponding to the CAN port number of the host
ENABLE	Enable	BOOL	Depends on data types	-	(Triggered by level) Execution of the function block switch
TIMEOUT	Maximum execution time	UDINT	Depends on data types	-	Maximum execution time, in the unit of ms If the execution times out before the response is received, the execution aborts due to timeout. 0 (default): Timeout disabled 1 to 65535: Timeout value This variable is not available.
DEVICE	Node ID of the CANopen device	DEVICE	Depends on data types	-	Node ID of the CANopen device 0 (default): Local device (PLC) 1 to 127: Device node ID
CHANNEL	SDO channel number	USINT	Depends on data types	-	SDO channel number The number is 1 by default.
INDEX	Object index	WORD	Depends on data types	-	Object index The value ranges from 0000 (hexadecimal) to FFFF (hexadecimal).

Input Variable	Name	Data Type	Value Range	Initial Value	Description
SUBINDEX	Object sub-index	BYTE	Depends on data types	-	Object sub-index The value ranges from 00 (hexadecimal) to FF (hexadecimal).
DATA	Data to receive	ARRAY[1...4] OF BYTE	Depends on data types	-	Array for storing object values to write
DATALENGTH	Size of the object to write	UINT	Depends on data types	-	Size of the object to write, in the unit of bytes

Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
CONFIRM	Completion flag of the function block	BOOL	Depends on data types	-	Completion flag of the function block
ERROR	Error information of function block	CANOPEN_KERNEL_ ERROR	Depends on data types	-	Error information of function block
ERRORINFO	SDO abort message content	SDO_ERROR	Depends on data types	-	When ERROR is 1, the SDO abort message content (four bytes) is returned.

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
NETWORK	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-
ENABLE	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
TIMEOUT	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-
DEVICE	DEVICE																			
CHANNEL	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-
INDEX	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
SUBINDEX	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
DATA	ARRAY[1...4] OF BYTE																			
DATALENGTH	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-
CONFIRM	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
ERROR	CANOPEN_KERNEL_ERROR																			
ERRORINFO	SDO_ERROR																			

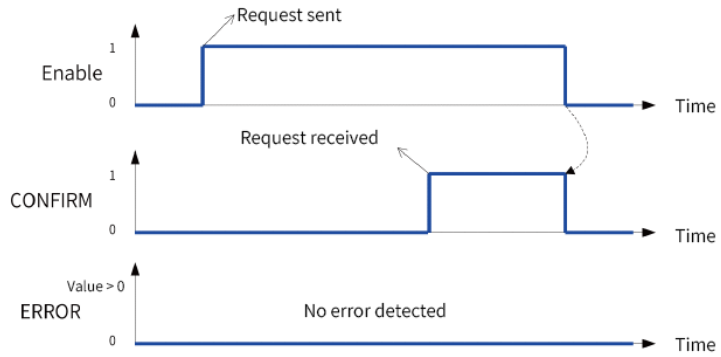
■ Function

This function block writes a device-specific CANopen object of up to four bytes from the SDO message.

■ Timing diagram

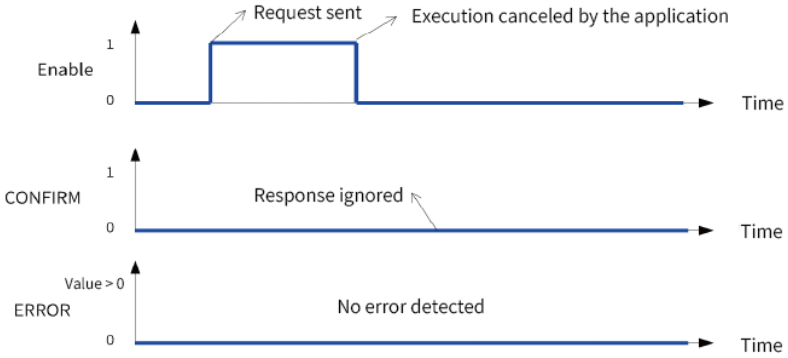
When the system responds to the current request and no errors are detected, CONFIRM is set to TRUE,

which is reserved (provided that the function block is called when ENABLE is set to TRUE). If the function block is called when ENABLE is reset to FALSE, CONFIRM is reset to FALSE, and the function block can be executed again.



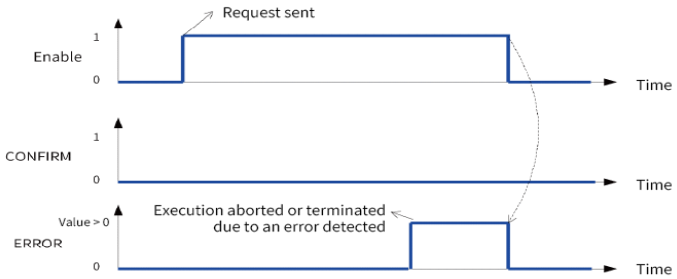
The execution is canceled by application programs.

Before the current execution ends, if the function block is called when ENABLE is reset to FALSE, the function block execution is canceled. The system may have responded to the canceled request or the response will be received later. However, the response is ignored.



Execution is aborted or terminated due to an error detected.

Once the current execution is aborted due to receiving an SDO abort message or terminated due to an error detected, ERROR is set to a value other than 0. For more information about the detected error codes, see the following table. If the function block is called when ENABLE is reset to FALSE, ERROR is reset to 0, and the function block can be executed again.



■ Precautions

The following specific parameters must be transferred to this function block.

1) Device node ID

2) SDO client/server channel (only one channel is defined by default)

3) CANopen object index/sub-index

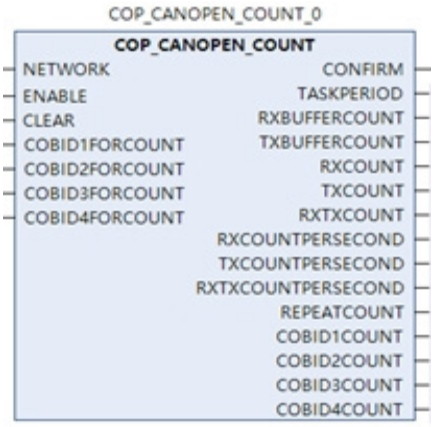
4) Value to be written

5) Number of bytes to be written

6.3. 24 COP_CANOPEN_COUNT

This instruction implements CANopen count.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
COP_CANOPEN_COUNT	CANopen count	FB		COP_CANOPEN_COUNT_0(NETWORK:=, ENABLE :=, CLEAR :=, COBID1FORCOUNT :=, COBID2FORCOUNT :=, COBID3FORCOUNT :=, COBID4FORCOUNT :=, CONFIRM => TASKPERIOD => RXBUFFERCOUNT => TXBUFFERCOUNT => RXCOUNT => TXCOUNT => RXTXCOUNT => RXCOUNTPERSECOND => TXCOUNTPERSECOND => RXTXCOUNTPERSECOND=> REPEATCOUNT => COBID1COUNT => COBID2COUNT => COBID3COUNT => COBID4COUNT =>);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
NETWORK	CAN network number	USINT	Depends on data types	-	CAN port executed by the function block, corresponding to the CAN port number of the host
ENABLE	Enable	BOOL	Depends on data types	-	(Triggered by level) Execution of the function block switch
CLEAR	Clears the value of the current output parameter.	BOOL	Depends on data types	-	Clears the value of the current output parameter.
COBID1FORCOUNT	Data frame of ID 1	DWORD	Depends on data types	-	Data frame whose ID is 1. Not supported currently
COBID2FORCOUNT	Data frame of ID 2	DWORD	Depends on data types	-	Data frame whose ID is 2. Not supported currently
COBID3FORCOUNT	Data frame of ID 3	DWORD	Depends on data types	-	Data frame whose ID is 3. Not supported currently
COBID4FORCOUNT	Data frame of ID 4	DWORD	Depends on data types	-	Data frame whose ID is 4. Not supported currently

Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
CONFIRM	Completion flag of the function block	BOOL	Depends on data types	-	TRUE: Successful execution
TASKPERIOD	Network load rate	DWORD	Depends on data types	-	Network load rate

Output Variable	Name	Data Type	Value Range	Initial Value	Description
RXBUFFERCOUNT	CAN PLC error	DWORD	Depends on data types	-	CAN PLC error
TXBUFFERCOUNT	CAN PLC error, which is not supported	DWORD	Depends on data types	-	CAN PLC error, which is not supported
RXCOUNT	Count of sent frames	DWORD	Depends on data types	-	Count of sent frames
TXCOUNT	Count of received frames	DWORD	Depends on data types	-	Count of received frames
RXTXCOUNT	Count of sent and received frames	DWORD	Depends on data types	-	Count of sent and received frames
RXCOUNTPERSECOND	Frame receiving speed. Not supported currently	DWORD	Depends on data types	-	Frame receiving speed. Not supported currently
TXCOUNTPERSECOND	Frame sending speed. Not supported currently	DWORD	Depends on data types	-	Frame sending speed. Not supported currently
RXTXCOUNTPERSECOND	Frame sending/receiving speed. Not supported currently	DWORD	Depends on data types	-	Frame sending/ receiving speed. Not supported currently
REPEATCOUNT	Number of repeat frames. Not supported currently	DWORD	Depends on data types	-	Number of repeat frames. Not supported currently
COBID1COUNT	Number of data frames whose ID is 1. Not supported currently	DWORD	Depends on data types	-	Number of data frames whose ID is 1. Not supported currently
COBID2COUNT	Number of data frames whose ID is 2. Not supported currently	DWORD	Depends on data types	-	Number of data frames whose ID is 2. Not supported currently
COBID3COUNT	Number of data frames whose ID is 3. Not supported currently	DWORD	Depends on data types	-	Number of data frames whose ID is 3. Not supported currently
COBID4COUNT	Number of data frames whose ID is 4. Not supported currently	DWORD	Depends on data types	-	Number of data frames whose ID is 4. Not supported currently

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
NETWORK	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-
ENABLE	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
CLEAR	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
COBID1FORCOUNT	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
COBID2FORCOUNT	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
COBID3FORCOUNT	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
COBID4FORCOUNT	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
CONFIRM	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
TASKPERIOD	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
RXBUFFERCOUNT	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
TXBUFFERCOUNT	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
RXCOUNT	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
TXCOUNT	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
RXTXCOUNT	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
RXCOUNTPERSECOND	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
TXCOUNTPERSECOND	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
RXTXCOUNTPERSECOND	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
REPEATCOUNT	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
COBID1COUNT	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
COBID2COUNT	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
COBID3COUNT	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
COBID4COUNT	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

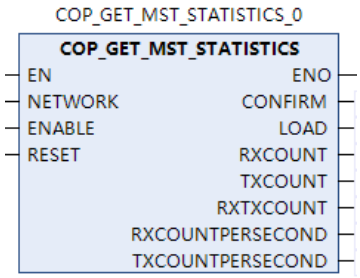
■ Function

This instruction implements CANopen count.

6.3. 25 COP_GET_MST_STATISTICS

This instruction gets MST statistics.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
COP_GET_MST_STATISTICS	Get MST statistics	FB		COP_GET_MST_STATISTICS_0(NETWORK:= ENABLE :=, RESET :=, CONFIRM =>, LOAD =>, RXCOUNT =>, TXCOUNT =>, RXTXCOUNT =>, RXCOUNTPERSECOND=>, TXCOUNTPERSECOND=>,);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
NETWORK	CAN network number	USINT	Depends on data types	-	CAN port executed by the function block, corresponding to the CAN port number of the host
ENABLE	Enable	BOOL	Depends on data types	-	(Triggered by level) Execution of the function block switch
RESET	Clears the value of the current output parameter.	BOOL	Depends on data types	-	Clears the value of the current output parameter.

Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
CONFIRM	Completion flag of the function block	BOOL	Depends on data types	-	TRUE: Successful execution

Output Variable	Name	Data Type	Value Range	Initial Value	Description
LOAD	Network load rate	BYTE	Depends on data types	-	Network load rate
RXCOUNT	Count of sent frames	DWORD	Depends on data types	-	Count of sent frames
TXCOUNT	Count of received frames	DWORD	Depends on data types	-	Count of received frames
RXTXCOUNT	Count of sent and received frames	DWORD	Depends on data types	-	Count of sent and received frames
RXCOUNTPERSECOND	Frame receiving speed. Not supported currently	DWORD	Depends on data types	-	Frame receiving speed. Not supported currently
TXCOUNTPERSECOND	Frame sending speed. Not supported currently	DWORD	Depends on data types	-	Frame sending speed. Not supported currently

■ Function

This instruction gets MST statistics.

6. 4 CAN Free Protocol

6.4. 1 Instruction List

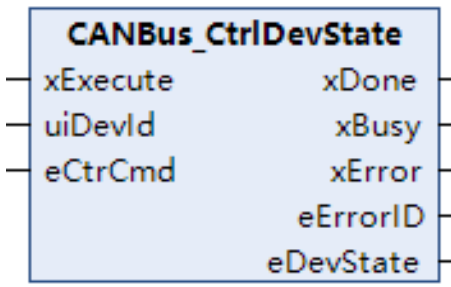
Instruction Category	Name	FB/FC	Function
CAN Free Protocol	CANBus_CtrlDevState	FB	Control the device state
	CANBus_DevRxMsg	FB	Allow the device to receive information
	CANBus_DevRxMultiMsg	FB	Allow the device to receive information about multiple frames
	CANBus_DevTxMsg	FB	Allow the device to send information
	CANBus_GetDevDiagInfo	FB	Get the device diagnosis information
	CANBus_GetIoDrvVer	FB	Get the version of the CANBus component
	CANBus_SetDevBaud	FB	Set the device baud rate

6.4. 2 CANBus_CtrlDevState

This instruction controls the device state.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression

CANBus_Ctrl DevState	Control the device state	FB		CANBus_CtrlDevState_0(xExecute:= , uiDevId:= , eCtrCmd:= , xDone=> , xBusy=> , xError=> , eErrorID=> , eDevState=>);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
xExecute	Function block enable	BOOL	Depends on data types	(*)	Function block enable
uiDevId	Network ID of the CANBus device	UINT	Depends on data types	(*)	Device ID, which must correspond to the network ID of the CANBus device
eCtrlCmd	Command to control the device state switchover	CANBus_ CtrlCmdType_ Enum	Depends on data types	(*)	Command to control the device state switchover

Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
xDone	Completion flag	BOOL	Depends on data types	FALSE	Completion flag
xBusy	Operation flag	BOOL	Depends on data types	FALSE	Operation flag
xError	Error flag	BOOL	Depends on data types	FALSE	Error flag
eErrorID	Error code	CANBus_ FBErrorType_Enum	Depends on data types	(*)	Error code
eDevState	Equipment status	CANBus_StateType_ Enum	Depends on data types	(*)	Equipment status

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
xExecute	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
uiDevId	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-
eCtrlCmd	CANBus_CtrlCmdType_Enum																			
xDone	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
xBusy	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
xError	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
eErrorID	CANBus_FBErrorType_Enum																			
eDevState	CANBus_StateType_Enum																			

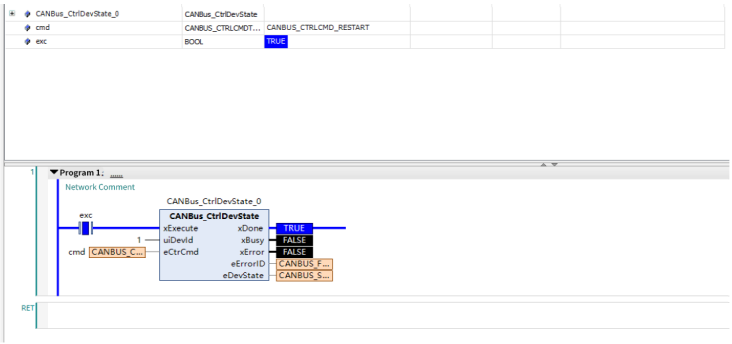
■ Function

The CANBus_CtrlDevState function block controls the device state based on the enumeration type CANBus_CtrlCmdType_Enum. The meanings of the enumerators are as follows.

Enumerator	Meaning
CANBUS_CTRLCMD_INVALID	Invalid command
CANBUS_CTRLCMD_RESTART	Restart CANBus
CANBUS_CTRLCMD_START	Start CANBus
CANBUS_CTRLCMD_STOP	Stop CANBus
CANBUS_CTRLCMD_CLEAR	Clear CANBus parameters

■ Program example

Example: Restart the CAN port of which the network number is 1.



■ Precautions

· If the command input of the function block is CANBUS_CTRLCMD_CLEAR (Clear CANBus parameters), the following information will be cleared after the function block is triggered: (1) Current load rate, (2) Receive and transmit buffers in the protocol stack, (3) Receive function block.

Data in items (1) and (2) will be automatically overwritten by the running result. For item (3), the receive function block needs to be restarted.

■ Error description

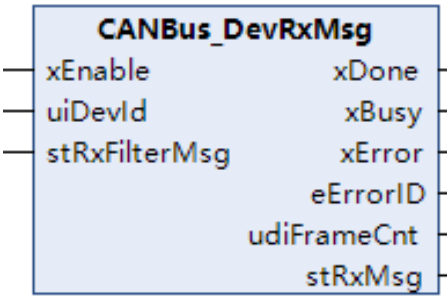
When an error occurs, locate the error specified by eErrorID according to the help document, and find the reason for the error.

Function Block Error Type	Error Code	Meaning	Troubleshooting
CANBUS_FBERR_NO_ERROR	0x0000	The function block runs normally or has no error.	The function block runs normally.
CANBUS_FBERR_EXE_FAILED	0x0001	The function block failed to run.	The input DevID is invalid.
CANBUS_FBERR_INVAILD_DEVICEID	0x0002	The device ID is invalid because this ID is not enabled.	The input DevID is invalid.
CANBUS_FBERR_DEVICE_FAULT	0x0003	The device is faulty.	The device is being started.
CANBUS_FBERR_WRONG_PARAMETER	0x0004	The input parameters of the function block are incorrect.	The input CtrlCmd is invalid.
CANBUS_FBERR_BUSY_ERROR	0x0005	The function block is busy.	-
CANBUS_FBERR_ABORT_ERROR	0x0006	An error occurs when the function block is aborted.	-
CANBUS_FBERR_TIME_OUT_ERROR	0x0007	The function block times out.	-
CANBUS_FBERR_POINTER_NULL	0x0008	The function block pointer is null.	-
CANBUS_FBERR_BUFF_FULL	0x0009	The function block buffer is used up.	-
CANBUS_FBERR_UNKNOWN_ERROR	0x00FF	The function block has an unknown error.	-

6.4. 3 CANBus_DevRxMsg

This instruction allows the device to receive information.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
CANBus_ DevRxMsg	Allow the device to receive information	FB		CANBus_DevRxMsg_0(xEnable:= , uiDevId:= , stRxFilterMsg:= , xDone=> , xBusy=> , xError=> , eErrorID=> , udiFrameCnt=> , stRxMsg=>);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
xEnable	Function block enable	BOOL	Depends on data types	FLASE	Function block enable: Enabled at high level, and aborted or reset at low level
uiDevId	Network ID of the CANBus device	UINT	Depends on data types	(*)	Device ID, which must correspond to the network ID of the CANBus device
stRxFilterMsg	Setting of information reception and filtering	CANBus_ RxFilter_t	Depends on data types	(*)	Setting of receive parameters and filter parameters

Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
xDone	Completion flag	BOOL	Depends on data types	FALSE	Completion flag
xBusy	Operation flag	BOOL	Depends on data types	FALSE	Operation flag
xError	Error flag	BOOL	Depends on data types	FALSE	Error flag
eErrorID	Error code	CANBus_FBErrorType_ Enum	Depends on data types	(*)	Error code
udiFrameCnt	Frame counter	UDINT	Depends on data types	(*)	Frame counter, indicating the number of received frames that meet the conditions
stRxMsg	Information reception	CANBus_FrameMsg_t	Depends on data types	(*)	Saving of information about the received frame that meets the conditions (only the last frame of the received CAN message is cached)

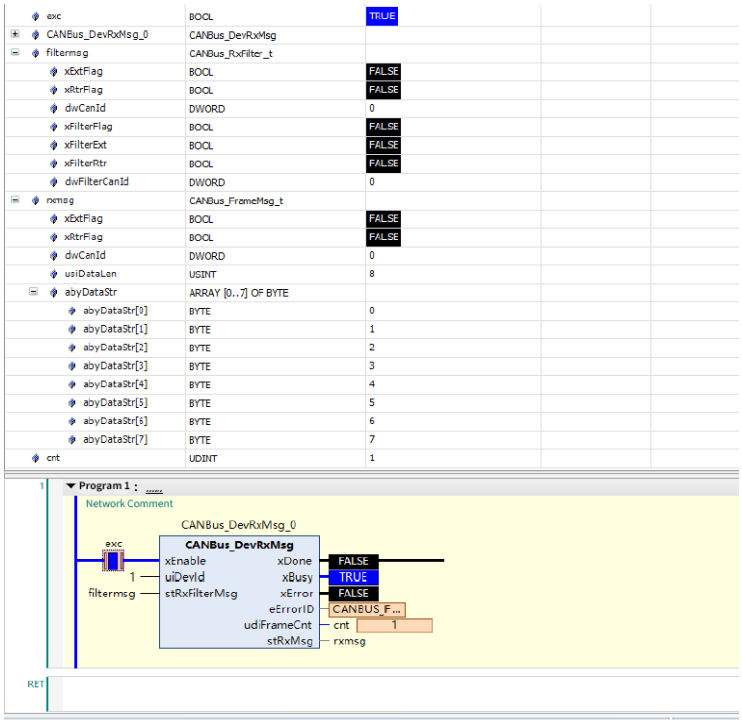
	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
xEnable	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
uiDevId	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-
stRxFilterMsg	CANBus_RxFilter_t																			
xDone	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
xBusy	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
xError	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
eErrorID	CANBus_FBErrorType_Enum																			
udiFrameCnt	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-
stRxMsg	CANBus_FrameMsg_t																			

■ Function

During the enabling process, the function block receives CAN data from the network number corresponding to uiDevId. The filtering information is specified by stRxFilterMsg. The received CAN frames that meet the conditions are counted by udiFrameCnt, and information about the latest CAN frame that meets the conditions is saved in stRxMsg.

■ Program example

Example: Receive data of the CAN port of which the network number is 1, without filtering. The received data is stored in rxmsg.



■ Precautions

Up to 128 instantiated receive function blocks can be enabled at a time, including CANBus_DevRxMsg and CANBus_DevRxMultiMsg.

■ Error description

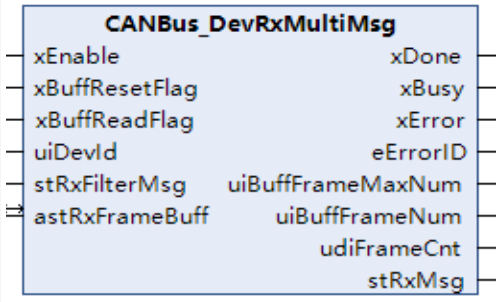
When an error occurs, locate the error specified by eErrorID according to the help document, and find the reason for the error.

Function Block Error Type	Error Code	Meaning	Troubleshooting
CANBUS_FBERR_NO_ERROR	0x0000	The function block runs normally or has no error.	The function block runs normally.
CANBUS_FBERR_EXE_FAILED	0x0001	The function block failed to run.	The parameter address is modified.
CANBUS_FBERR_INVAILD_DEVICEID	0x0002	The device ID is invalid because this ID is not enabled.	The input DevID is invalid.
CANBUS_FBERR_DEVICE_FAULT	0x0003	The device is faulty.	The device is being started.
CANBUS_FBERR_WRONG_PARAMETER	0x0004	The input parameters of the function block are incorrect.	The input CtrlCmd is invalid.
CANBUS_FBERR_BUSY_ERROR	0x0005	The function block is busy.	-
CANBUS_FBERR_ABORT_ERROR	0x0006	An error occurs when the function block is aborted.	-
CANBUS_FBERR_TIME_OUT_ERROR	0x0007	The function block times out.	-
CANBUS_FBERR_POINTER_NULL	0x0008	The function block pointer is null.	-
CANBUS_FBERR_BUFF_FULL	0x0009	The function block buffer is used up.	-
CANBUS_FBERR_UNKNOWN_ERROR	0x00FF	The function block has an unknown error.	-

6.4. 4 CANBus_DevRxMultiMsg

This instruction allows the device to receive information about multiple frames.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
CANBus_ DevRxMultiMsg	Allow the device to receive information about multiple frames	FB		CANBus_DevRxMultiMsg_0(xEnable:= , xBuffResetFlag:= , xBuffReadFlag:= , uiDevId:= , stRxFilterMsg:= , xDone=> , xBusy=> , xError=> , eErrorID=> , uiBuffFrameMaxNum=> , uiBuffFrameNum=> , udiFrameCnt=> , stRxMsg=> , astRxFrameBuff:=);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
xEnable	Function block enable	BOOL	Depends on data types	FLASE	Function block enable: Enabled at high level, and aborted or reset at low level
xBuffResetFlag	Reset buffer flag	BOOL	Depends on data types	FLASE	TRUE: Clearing the buffer. The buffer is filled again and the function block is automatically cleared.
xBuffReadFlag	Read buffer flag	BOOL	Depends on data types	FLASE	TRUE: The application layer reads data from the buffer. This state can prevent the function block from writing data to the buffer.
uiDevId	Network ID of the CANBus device	UINT	Depends on data types	(*)	Device ID, which must correspond to the network ID of the CANBus device
stRxFilterMsg	Setting of information reception and filtering	CANBus_ RxFilter_t	Depends on data types	(*)	Setting of receive parameters and filter parameters

Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
xDone	Completion flag	BOOL	BOOL	FALSE	Completion flag
xBusy	Operation flag	BOOL	BOOL	FALSE	Operation flag
xError	Error flag	BOOL	BOOL	FALSE	Error flag
eErrorID	Error code	CANBus_FBErrorType_ Enum	CANBus_ FBErrorType_ Enum	(*)	Error code
udiFrameCnt	Frame counter	UDINT	UDINT	-	Frame counter, indicating the number of received frames that meet the conditions
stRxMsg	Information reception	CANBus_FrameMsg_t	CANBus_ FrameMsg_t	(*)	Saving of information about the received frame that meets the conditions (only the last frame of the received CAN message is cached)

In-out variables

In-Out Variable	Name	Data Type	Value Range	Initial Value	Description
astRxFrameBuff	Received frame buffer	POINT TO CANBus_ FrameMsg_t	Depends on data types	FALSE	Received frame buffer. It is a one-dimensional length-variable structure array. The number of array elements cannot exceed 65535.

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
xEnable	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
xBuffResetFlag	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

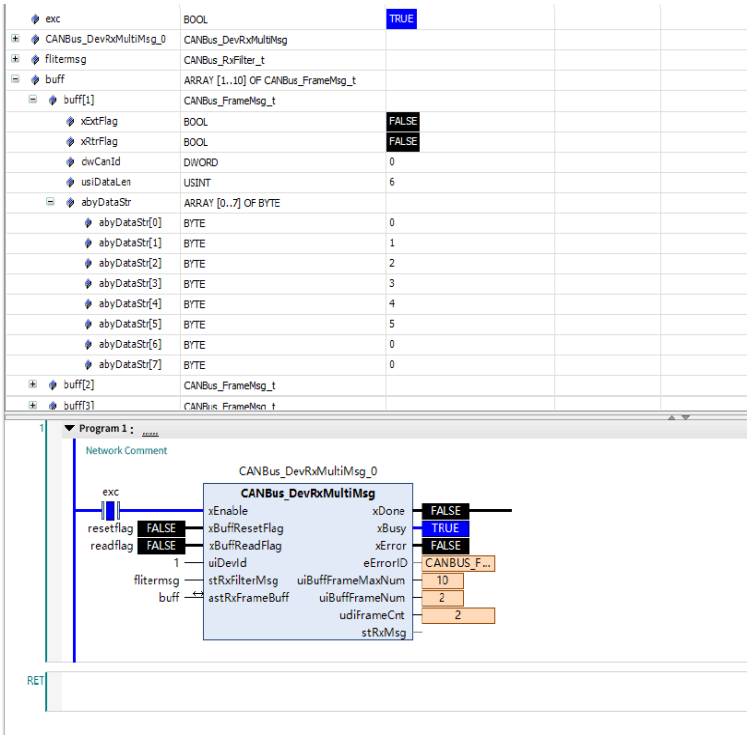
	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
xBuffReadFlag	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
uiDevId	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-
stRxFilterMsg	CANBus_RxFilter_t																			
xDone	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
xBusy	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
xError	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
eErrorID	CANBus_FBErrorType_Enum																			
udiFrameCnt	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-
stRxMsg	CANBus_FrameMsg_t																			
astRxFrameBuff	POINT TO CANBus_FrameMsg_t																			

■ Function

During the enabling process, the function block receives CAN data from the network number corresponding to uiDevId. The filtering information is specified by stRxFilterMsg. The received CAN frames that meet the conditions are counted in udiFrameCnt, and information about the latest CAN frame that meets the conditions is saved in stRxMsg.

■ Program example

Example: Receive data of the CAN port of which the network number is 1, without filtering. The received data is stored in the buff array.



■ Precautions

· Up to 128 instantiated receive function blocks can be enabled at a time, including CANBus_DevRxMsg and CANBus_DevRxMultiMsg.

· During buffer reading, you can reset xBuffReadFlag to prevent new data from overwriting existing data in the buffer. Note that during buffer reading, the internal register only saves the last received message. Therefore, to avoid frame loss, reset this flag timely at the possible smallest internal.

■ Error description

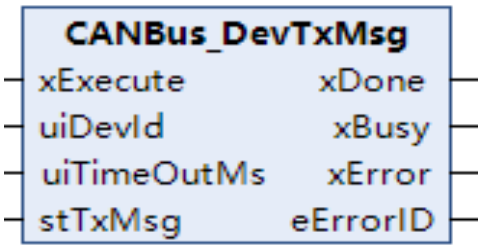
When an error occurs, locate the error specified by eErrorID according to the help document, and find the reason for the error.

Function Block Error Type	Error Code	Meaning	Troubleshooting
CANBUS_FBERR_NO_ERROR	0x0000	The function block runs normally or has no error.	The function block runs normally.
CANBUS_FBERR_EXE_FAILED	0x0001	The function block failed to run.	The parameter address is modified.
CANBUS_FBERR_INVAILD_DEVICEID	0x0002	The device ID is invalid because this ID is not enabled.	The input DevID is invalid.
CANBUS_FBERR_DEVICE_FAULT	0x0003	The device is faulty.	-
CANBUS_FBERR_WRONG_PARAMETER	0x0004	The input parameters of the function block are incorrect.	The maximum number of buffers is 0.
CANBUS_FBERR_BUSY_ERROR	0x0005	The function block is busy.	The instantiated function block is out of the limit.
CANBUS_FBERR_ABORT_ERROR	0x0006	An error occurs when the function block is aborted.	-
CANBUS_FBERR_TIME_OUT_ERROR	0x0007	The function block times out.	-
CANBUS_FBERR_POINTER_NULL	0x0008	The function block pointer is null.	The input/output parameters or the buffer address cannot be specified.
CANBUS_FBERR_BUFF_FULL	0x0009	The function block buffer is used up.	The receive buffer is used up.
CANBUS_FBERR_UNKNOWN_ERROR	0x00FF	The function block has an unknown error.	-

6.4. 5 CANBus_DevTxMsg

This instruction allows the device to send information.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
CANBus_ DevTxMsg	Allow the device to send information	FB		CANBus_DevTxMsg_0(xExecute:= , uiDevId:= , uiTimeOutMs:= , stTxMsg:= , xDone=> , xBbusy=> , xError=> , eErrorID=>);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
xExecute	Function block trigger flag	BOOL	Depends on data types	(*)	Function block trigger flag: Triggered at rising edge, and aborted or reset at low level
uiDevId	Network ID of the CANBus device	UINT	Depends on data types	(*)	Device ID, which must correspond to the network ID of the CANBus device
uiTimeOutMs	Timeout	UINT	Depends on data types	(*)	Send timeout time, in ms
stTxMsg	Sent information	CANBus_ FrameMsg_t	Depends on data types	(*)	Sent information, including the expansion frame flag, remote frame flag, CANID, data length, and data

Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
xDone	Completion flag	BOOL	Depends on data types	FALSE	Completion flag
xBusy	Operation flag	BOOL	Depends on data types	FALSE	Operation flag
xError	Error flag	BOOL	Depends on data types	FALSE	Error flag
eErrorID	Error code	CANBus_FBErrorType_ Enum	Depends on data types	(*)	Error code

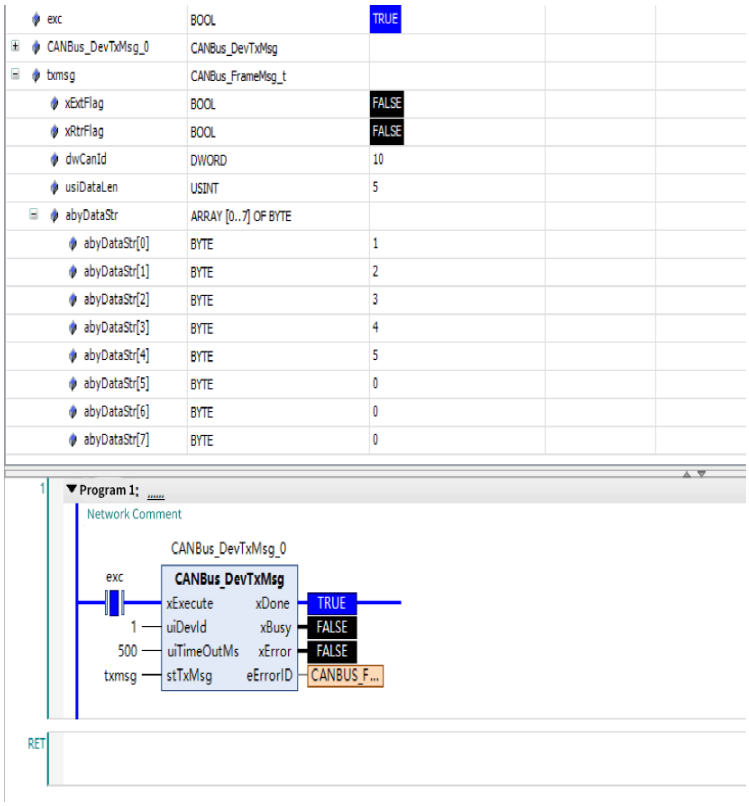
	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
xExecute	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
uiDevId	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-
uiTimeOutMs	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-
stTxMsg	CANBus_FrameMsg_t																			
xDone	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
xBusy	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
xError	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
eErrorID	CANBus_FBErrorType_Enum																			

■ Function

When triggered, the function block sends a frame of CAN data to the network number corresponding to uiDevId. The sent information is specified by stTxMsg.

■ Program example

Example: A standard data frame is sent to the CAN port of which the network number is 1. For this frame, CANID is 0x0a, data length is 5, and data is 0x01, 0x02, 0x03, 0x04, and 0x05.



■ Precautions

If the timeout time is set to 0, the send timeout is detected at an interval of 50 ms by default during function block running.

■ Error description

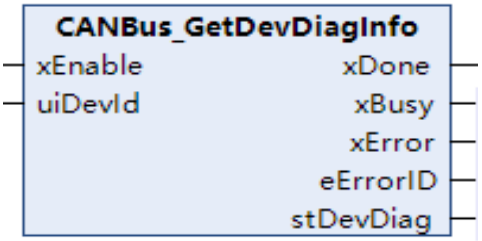
When an error occurs, locate the error specified by eErrorID according to the help document, and find the reason for the error.

Function Block Error Type	Error Code	Meaning	Troubleshooting
CANBUS_FBERR_NO_ERROR	0x0000	The function block runs normally or has no error.	The function block runs normally.
CANBUS_FBERR_EXE_FAILED	0x0001	The function block failed to run.	The input DevID is invalid.
CANBUS_FBERR_INVAILD_DEVICEID	0x0002	The device ID is invalid because this ID is not enabled.	The input DevID is invalid.
CANBUS_FBERR_DEVICE_FAULT	0x0003	The device is faulty.	The device is not running or faulty.
CANBUS_FBERR_WRONG_PARAMETER	0x0004	The input parameters of the function block are incorrect.	The CANID is invalid and the data length exceeds the limit (8 for data frames and 0 for remote frames).
CANBUS_FBERR_BUSY_ERROR	0x0005	The function block is busy.	The send requests are conflicting.
CANBUS_FBERR_ABORT_ERROR	0x0006	An error occurs when the function block is aborted.	-
CANBUS_FBERR_TIME_OUT_ERROR	0x0007	The function block times out.	The data sending times out and is not successful.
CANBUS_FBERR_POINTER_NULL	0x0008	The function block pointer is null.	-
CANBUS_FBERR_BUFF_FULL	0x0009	The function block buffer is used up.	-
CANBUS_FBERR_UNKNOWN_ERROR	0x00FF	The function block has an unknown error.	-

6.4. 6 CANBus_GetDevDiagInfo

This instruction gets the device diagnosis information.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
CANBus_ GetDevDiagInfo	Get the device diagnosis information	FB		CANBus_GetDevDiagInfo_0(xEnable:= , uiDevId:= , xDone=> , xBusy=> , xError=> , eErrorID=> , stDevDiag=>);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
xEnable	Function block enable	BOOL	Depends on data types	FLASE	Function block enable: Enabled at high level, and aborted or reset at low level
uiDevId	Network ID of the CANBus device	UINT	Depends on data types	(*)	Device ID, which must correspond to the network ID of the CANBus device

Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
xDone	Completion flag	BOOL	Depends on data types	FALSE	Completion flag
xBusy	Operation flag	BOOL	Depends on data types	FALSE	Operation flag
xError	Error flag	BOOL	Depends on data types	FALSE	Error flag
eErrorID	Error code	CANBus_FBErrorType_ Enum	Depends on data types	(*)	Error code
stDevDiag	Device diagnosis information	CANBus_DiagInfo_t	Depends on data types	(*)	Device diagnosis information

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
xExecute	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
uiDevId	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-
xDone	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
xBusy	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
xError	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

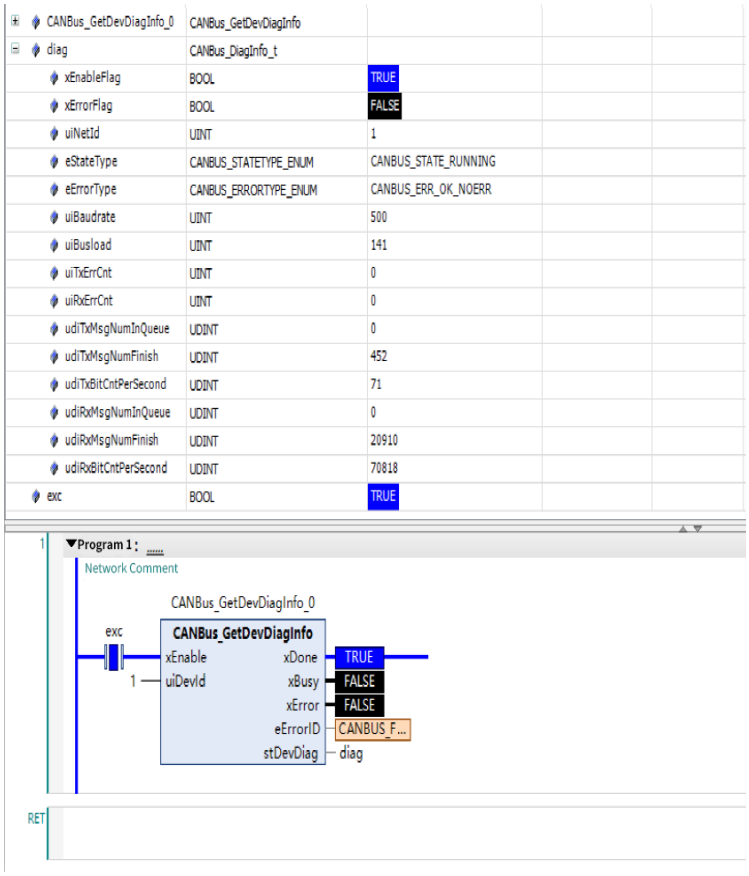
	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
eErrorID	CANBus_FBErrorType_Enum																			
stDevDiag	CANBus_DiagInfo_t																			

■ Function

During the enabling process, the function block obtains the diagnosis information of the network number corresponding to uiDevId, and the diagnosis information is stored in stDevDiag.

■ Program example

Example: Obtain the diagnosis information of the CAN port of which the network number is 1.



■ Precautions

The diagnosis information is updated in real time, including but not limited to the operation status, error state, DevID, running baud rate, bus load, and send/receive counter.

■ Error description

When an error occurs, locate the error specified by eErrorID according to the help document, and find the reason for the error.

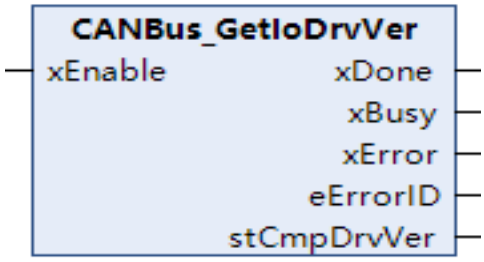
Function Block Error Type	Error Code	Meaning	Troubleshooting
CANBUS_FBERR_NO_ERROR	0x0000	The function block runs normally or has no error.	The function block runs normally.
CANBUS_FBERR_EXE_FAILED	0x0001	The function block failed to run.	-
CANBUS_FBERR_INVAILD_DEVICEID	0x0002	The device ID is invalid because this ID is not enabled.	The input DevID is invalid.

Function Block Error Type	Error Code	Meaning	Troubleshooting
CANBUS_FBERR_DEVICE_FAULT	0x0003	The device is faulty.	-
CANBUS_FBERR_WRONG_PARAMETER	0x0004	The input parameters of the function block are incorrect.	-
CANBUS_FBERR_BUSY_ERROR	0x0005	The function block is busy.	-
CANBUS_FBERR_ABORT_ERROR	0x0006	An error occurs when the function block is aborted.	-
CANBUS_FBERR_TIME_OUT_ERROR	0x0007	The function block times out.	-
CANBUS_FBERR_POINTER_NULL	0x0008	The function block pointer is null.	-
CANBUS_FBERR_BUFF_FULL	0x0009	The function block buffer is used up.	-
CANBUS_FBERR_UNKNOWN_ERROR	0x00FF	The function block has an unknown error.	-

6.4. 7 CANBus_CetIoDrvVer

This instruction gets the version of the CANBus component.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
CANBus_ GetIoDrvVer	Get the version of the CANBus component	FB		CANBus_GetIoDrvVer_0(xEnable:= , xDone=> , xBusy=> , xError=> , eErrorID=>, stCmpDrvVer=>);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
xEnable	Function block enable	BOOL	Depends on data types	FLASE	Function block enable: Enabled at high level, and aborted or reset at low level

Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
xDone	Completion flag	BOOL	Depends on data types	FALSE	Completion flag
xBusy	Operation flag	BOOL	Depends on data types	FALSE	Operation flag
xError	Error flag	BOOL	Depends on data types	FALSE	Error flag
eErrorID	Error code	CANBus_FBErrorType_ Enum	Depends on data types	(*)	Error code

Output Variable	Name	Data Type	Value Range	Initial Value	Description
stCmpDrvVer	Version of the CANBus component	CANBus_CmpDrvVer_t	Depends on data types	(*)	Version of the CANBus component

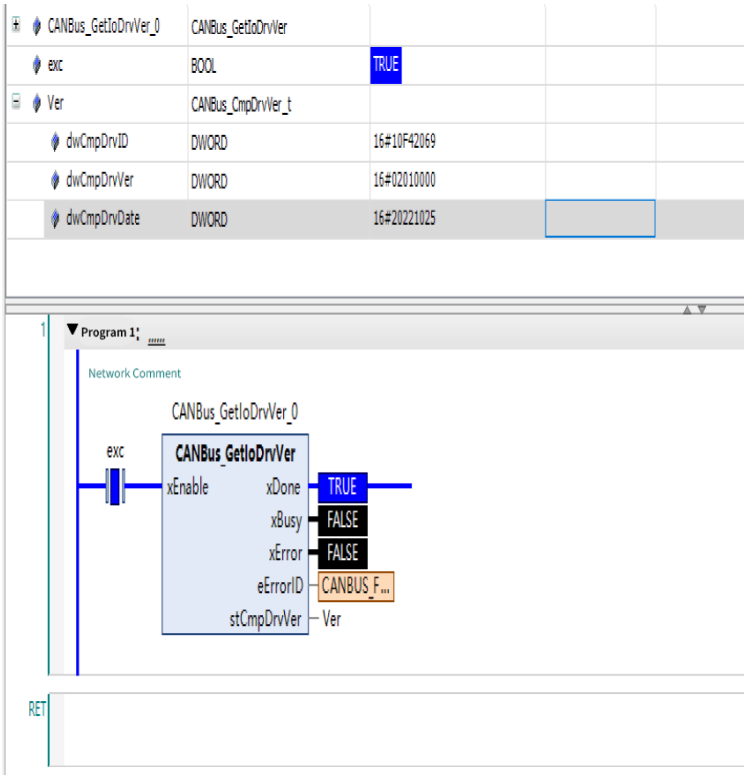
	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
xEnable	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
xDone	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
xBusy	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
xError	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
eErrorID	CANBus_FBErrorType_Enum																			
stCmpDrvVer	CANBus_CmpDrvVer_t																			

■ Function

When enabled, the function block obtains the version of the CANBus component.

■ Program example

Example: Get the version of the CANBus component.



■ Precautions

Parameters of the stCmpDrvVer structure are all in the hexadecimal format.

■ Error description

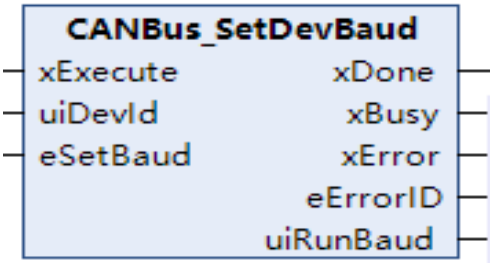
When an error occurs, locate the error specified by eErrorID according to the help document, and find the reason for the error.

Function Block Error Type	Error Code	Meaning	Troubleshooting
CANBUS_FBERR_NO_ERROR	0x0000	The function block runs normally or has no error.	The function block runs normally.
CANBUS_FBERR_EXE_FAILED	0x0001	The function block failed to run.	-
CANBUS_FBERR_INVAILD_DEVICEID	0x0002	The device ID is invalid because this ID is not enabled.	-
CANBUS_FBERR_DEVICE_FAULT	0x0003	The device is faulty.	-
CANBUS_FBERR_WRONG_PARAMETER	0x0004	The input parameters of the function block are incorrect.	-
CANBUS_FBERR_BUSY_ERROR	0x0005	The function block is busy.	-
CANBUS_FBERR_ABORT_ERROR	0x0006	An error occurs when the function block is aborted.	-
CANBUS_FBERR_TIME_OUT_ERROR	0x0007	The function block times out.	-
CANBUS_FBERR_POINTER_NULL	0x0008	The function block pointer is null.	-
CANBUS_FBERR_BUFF_FULL	0x0009	The function block buffer is used up.	-
CANBUS_FBERR_UNKNOWN_ERROR	0x00FF	The function block has an unknown error.	-

6.4. 8 CANBus_SetDevBaud

This instruction sets the device baud rate.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
CANBus_ SetDevBaud	Set the device baud rate	FB		CANBus_SetDevBaud_0(xExecute:= , uiDevId:= , eSetBaud:= , xDone=> , xBusy=> , xError=> , eErrorID=> , uiRunBaud=>);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
xExecute	Function block trigger flag	BOOL	Depends on data types	(*)	Function block trigger flag: Triggered at rising edge, and aborted or reset at low level
uiDevId	Network ID of the CANBus device	UINT	Depends on data types	(*)	Device ID, which must correspond to the network ID of the CANBus device

Input Variable	Name	Data Type	Value Range	Initial Value	Description
eSetBaud	CAN bus setting baud rate	CANBus_ BaudrateType_ Enum	Depends on data types	(*)	CAN bus setting baud rate, in kbps

Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
xDone	Completion flag	BOOL	Depends on data types	FALSE	Completion flag
xBusy	Operation flag	BOOL	Depends on data types	FALSE	Operation flag
xError	Error flag	BOOL	Depends on data types	FALSE	Error flag
eErrorID	Error code	CANBus_FBErrorType_ Enum	Depends on data types	(*)	Error code
uiRunBaud	CAN bus running baud rate	UINT	Depends on data types	(*)	CAN bus running baud rate

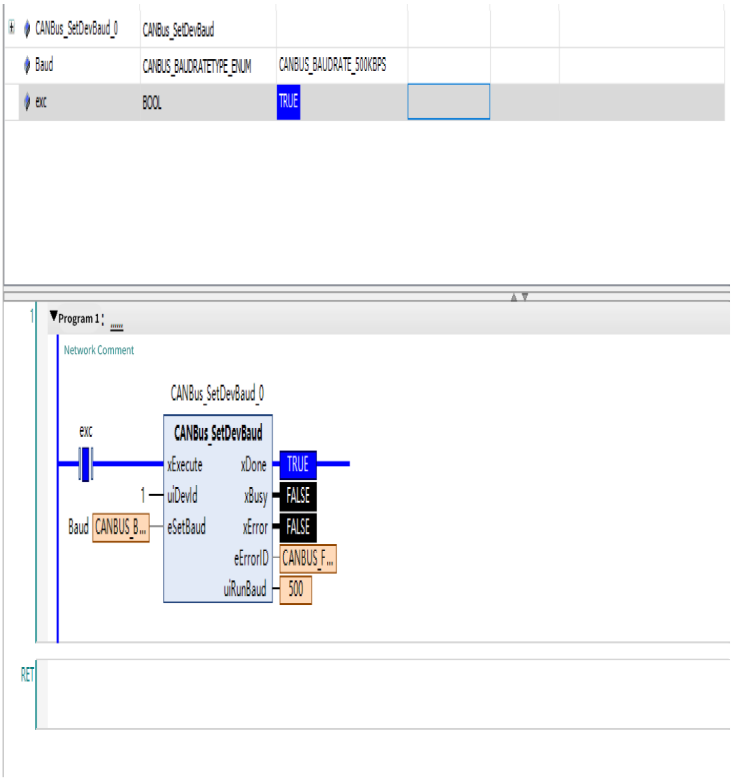
	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
xExecute	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
uiDevId	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
eSetBaud	CANBus_BaudrateType_Enum																			
stTxMsg	CANBus_FrameMsg_t																			
xDone	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
xBusy	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
xError	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
eErrorID	CANBus_FBErrorType_Enum																			

■ Function

When enabled, the function block changes the baud rate of the CAN port with the network number corresponding to uiDevId.

■ Program example

Example: Change the baud rate of the CAN port of which the network number is 1 to 500 kbps.



■ Precautions

· The function block of changing the baud rate only allows to change the baud rate of the CAN port of the PLC or expansion card, not allowing to change the baud rate of the port extended by the CAN module.

· eSetBaud specifies the CAN bus baud rate of an enumeration type, which can prevent the input of baud rates that the device does not support.

■ Error description

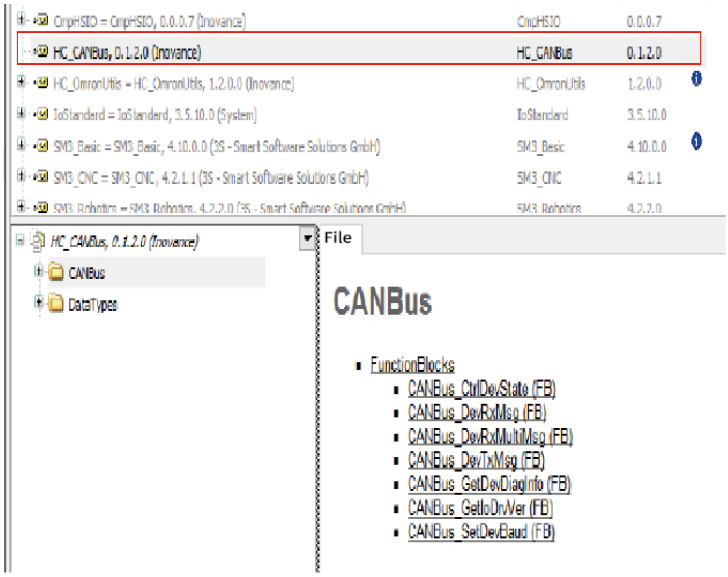
When an error occurs, locate the error specified by eErrorID according to the help document, and find the reason for the error.

Function Block Error Type	Error Code	Meaning	Troubleshooting
CANBUS_FBERR_NO_ERROR	0x0000	The function block runs normally or has no error.	The function block runs normally.
CANBUS_FBERR_EXE_FAILED	0x0001	The function block failed to run.	Modification of the baud rate failed.
CANBUS_FBERR_INVAILD_DEVICEID	0x0002	The device ID is invalid because this ID is not enabled.	The input DevID is invalid.
CANBUS_FBERR_DEVICE_FAULT	0x0003	The device is faulty.	-
CANBUS_FBERR_WRONG_PARAMETER	0x0004	The input parameters of the function block are incorrect.	The input baud rate is incorrect.
CANBUS_FBERR_BUSY_ERROR	0x0005	The function block is busy.	-
CANBUS_FBERR_ABORT_ERROR	0x0006	An error occurs when the function block is aborted.	-
CANBUS_FBERR_TIME_OUT_ERROR	0x0007	The function block times out.	-
CANBUS_FBERR_POINTER_NULL	0x0008	The function block pointer is null.	-
CANBUS_FBERR_BUFF_FULL	0x0009	The function block buffer is used up.	-
CANBUS_FBERR_UNKNOWN_ERROR	0x00FF	The function block has an unknown error.	-

6.4. 9 Example of Using Function Blocks

■ Enabling the CANBus library

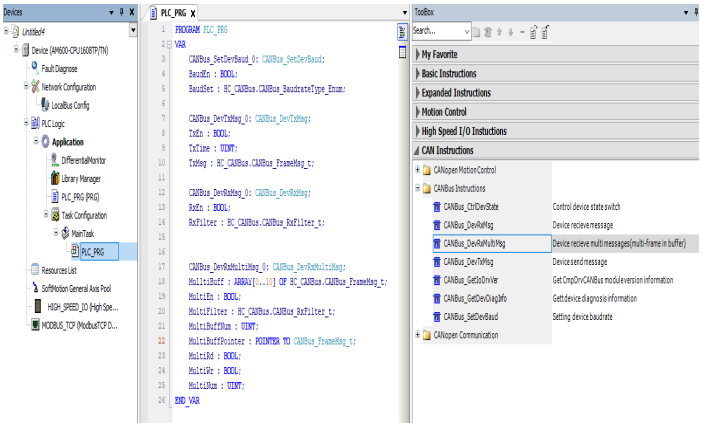
In the project, start “Library Manager” and add the HC_CANBus compilation library. The SP0T4 patch package is added by default.



◆ Note: Pay attention to the release status for the effective version of the compilation library.

■ Function block shortcut tools

Find the CANBus-related function blocks by expanding the CAN command in the toolbox.



■ Instantiating CANBus function blocks

The function block can be initialized by using the ST language or the ladder diagram.

ST language

```
18
19  CANBus_DevTxMsg0: CANBus_DevTxMsg;   Instantiated function block
20  txen: BOOL;
21  txtime: UINT;
22  txmsg: HC_CANBus.CANBus_FrameMsg_t;
23
24  CANBus_DevRxMsg0: CANBus_DevRxMsg;
25  rxen: BOOL;
26  rxfilter: HC_CANBus.CANBus_RxFilter_t;
27
28  CANBus_DevRxMultiMsg0: CANBus_DevRxMultiMsg;
                                                  100 %
31
32  CANBus_DevTxMsg0(                            Execution of the
33      xExecute:= txen,                         called function block
34      uiDevId:= devid,
35      uiTimeOutMs:= txtime,
36      stTxMsg:= txmsg,
37      xDone=> ,
38      xBusy=> ,
39      xError=> ,
40      eErrorID=> );
41
42  CANBus_DevRxMsg0(
43      xEnable:= rxen,
44      uiDevId:= devid,
45      stRxFilterMsg:= rxfilter,
46      xDone=> ,
47      xBusy=> ,
48      xError=> ,
49      eErrorID=> ,
50      udiFrameCnt=> ,
51      stRxMsg=> );
```

Ladder diagram

```
1  PROGRAM demo
2  VAR
3      CANBus_DevTxMsg_1: CANBus_DevTxMsg;   Instantiated function block
4      txmsg1: CANBus_FrameMsg_t;
5      txen1: BOOL;
6  END_VAR
7
                                                  100 %
1  Program1: .....
   Net comment                                    Called function block
                                                  in the ladder diagram
                  CANBus_DevTxMsg_1
   txen1          CANBus_DevTxMsg
                  xExecute      xDone
   60             uiDevId       xBusy
   10             uiTimeOutMs   xError
   txmsg1         stTxMsg       eErrorID
```

6. 5 EtherCAT Communication Instructions

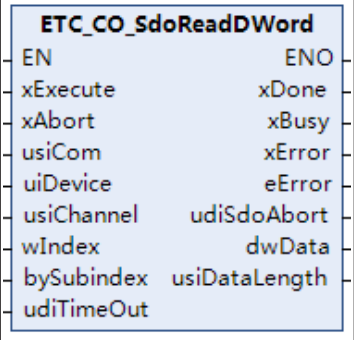
6.5. 1 Instruction List

Instruction Category	Name	FB/FC	Function
EtherCAT communication instructions	ETC_CO_SdoReadDWord	FB	Read EtherCAT slave parameters as DWORD
	ETC_CO_SdoRead4	FB	Read EtherCAT slave parameters that are no longer than 4 bytes
	ETC_CO_SdoRead	FB	Read EtherCAT slave parameters
	ETC_CO_SdoWriteDWord	FB	Write EtherCAT slave parameters as DWORD
	ETC_CO_SdoWrite4	FB	Write EtherCAT slave parameters that are no longer than 4 bytes
	ETC_CO_SdoWrite	FB	Write EtherCAT slave parameters
	IoDrvEtherCAT	FB	Implicit instance of EtherCAT master
	ETCSlave	FB	Implicit instance of EtherCAT slave
	ETC_ GetMasterInfo	FB	EtherCAT master station information
	ETC_ GetSlaveInfo	FB	EtherCAT slave station information

6.5. 2 ETC_CO_SdoReadDWord

This instruction enables the SDO to read up to four bytes from the object dictionary of the EtherCAT slave.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
ETC_CO_SdoReadDWord	Read EtherCAT slave parameters as DWORD	FB		ETC_CO_SdoRead DWord(xExecute:= , xAbort:= , usiCom:= , uiDevice:= , usiChannel:= , wIndex:= , bySubindex:= , udiTimeOut:= , xDone=> , xBusy=> , xError=> , eError=> , udiSdoAbort=> , dwData=> , usiDataLength=>);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
xExecute	Function block execution	BOOL	TRUE/FALSE	FALSE	TRUE: Stating to read parameters of the slave
xAbort	Function block execution abort	BOOL	TRUE/FALSE	FALSE	TRUE: The current reading process is terminated
usiCom	EtherCAT master count	BYTE	1 to 2	1	When only one EtherCAT master is used, the value is 1. When multiple masters are used, the value is 1 for the first master, 2 for the second master, and the like.
uiDevice	Slave physical address	UINT	1 to 65535	0	Number of a slave, such as 1000 and 1001
usiChannel	Reserved	BYTE	1	1	Reserved
wIndex	Object dictionary index	WORD	-	0	Example: Object dictionary index 16#6061 of the servo running mode
bySubindex	Object dictionary sub-index	BYTE	-	0	Example: Object dictionary sub-index 16#00 of the servo homing mode
udiTimeOut	Timeout period (ms)	UDINT	1 to 65535	0	During the period, if the request to read parameters of the slave is not replied, an error message is displayed.

Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
xDone	Parameter read completion	BOOL	TRUE/FALSE	FALSE	TRUE: Parameter read is completed
xBusy	Parameter read incompletion	BOOL	TRUE/FALSE	FALSE	TRUE: Parameter read is not completed
xError	Parameter read error	BOOL	TRUE/FALSE	FALSE	TRUE: An error occurs during parameter read. The eError parameter also displays the reason for the error.
eError	Error ID	ETC_CO_ERROR	-	0	Error ID
udiSdoAbort	More information about an error detected for the device	UDINT	-	0	-
dwData	Read data value	DWORD	-	0	-
usiDataLength	Read data length	BYTE	0 to 255	0	-

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	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	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

ENUM ETC_CO_ERROR

ETC_CO_NO_ERROR	0	No error
ETC_CO_FIRST_ERROR	5750	Error reason stored in the output of udiSdoAbort
ETC_CO_OTHER_ERROR	5751	Master not found
ETC_CO_DATA_OVERFLOW	5752	ETC_CO_Expedited, of which the length is larger than 4.
ETC_CO_TIME_OUT	5753	Timeout
ETC_CO_FIRST_MF	5770	Reserved
ETC_CO_LAST_ERROR	5779	Reserved

Note:

usiCom: EtherCAT master selection

When only one EtherCAT master is used for a device, such as Inovance AM600 or AM400 series, the value of usiCom is 1.

When two EtherCAT masters are used for a device:

If the value of usiCom is 1, the slave device is under the first EtherCAT master, such as Inovance AC800 series using the EtherCAT bus USB-C port.

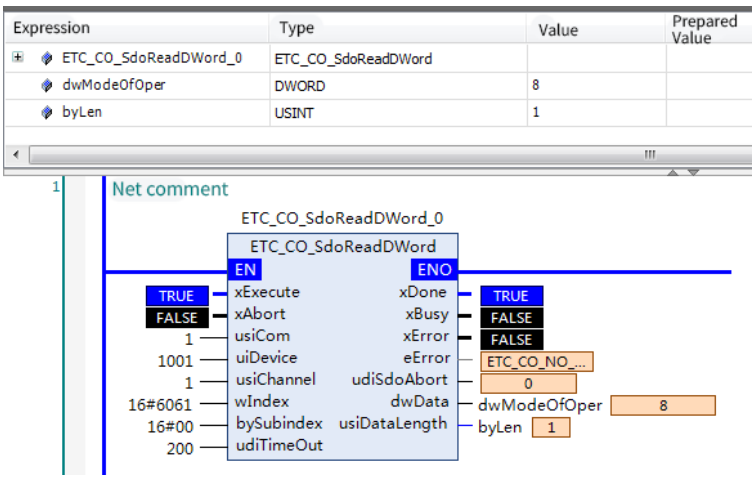
If the value of usiCom is 2, the slave device is under the second EtherCAT master, such as Inovance AC800 series using the EtherCAT D-sub bus connector.

■ Program example

Example 1

Read the object dictionary index 16#6061 and sub-index 16#00 of the servo slave running mode.

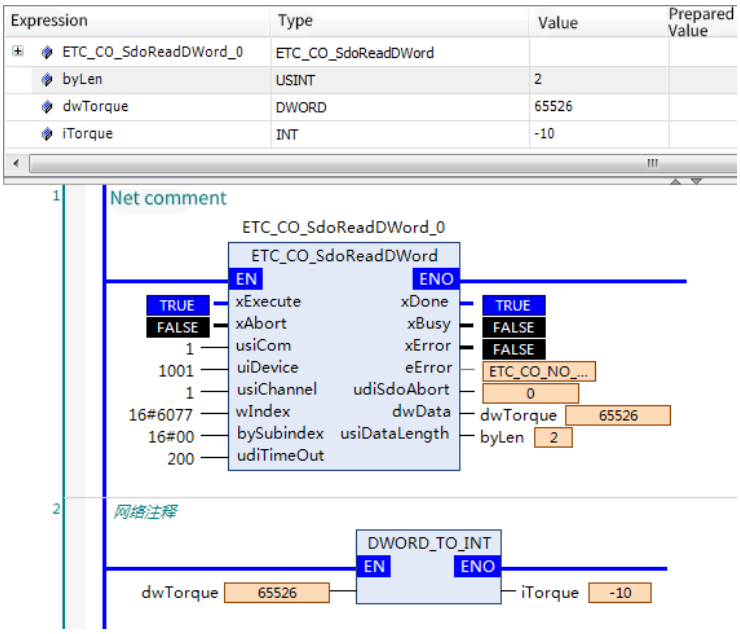
LD



Example 2

Read the object dictionary index 16#6077 and sub-index 16#00 of the real-time torque value of the servo slave.

LD

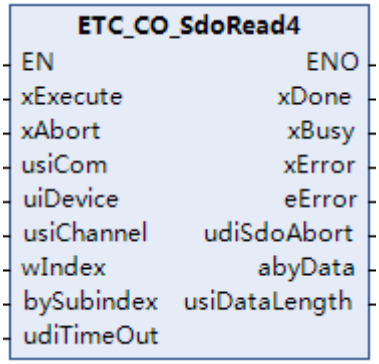


Note: The torque feedback 16#6077 should be converted to the INT type by DWORD_TO_INT.

6.5. 3 ETC_CO_SdoRead4

This instruction enables the SDO to read up to four bytes from the object dictionary of the EtherCAT slave. Different from the ETC_CO_SdoReadDWord function block, this instruction allows to store the read values to byte arrays.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
ETC_CO_SdoRead4	Read EtherCAT slave parameters that are no longer than 4 bytes	FB		ETC_CO_SdoRead DWord(xExecute:= , xAbort:= , usiCom:= , uiDevice:= , usiChannel:= , wIndex:= , bySubindex:= , udiTimeOut:= , xDone=> , xBusy=> , xError=> , eError=> , udiSdoAbort=> , abyData=> , usiDataLength=>);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
xExecute	Function block execution	BOOL	TRUE/FALSE	FALSE	TRUE: Stating to read parameters of the slave
xAbort	Function block execution abort	BOOL	[TRUE, FALSE]	FALSE	TRUE: The current reading process is terminated
usiCom	EtherCAT master count	BYTE	1 to 2	1	When only one EtherCAT master is used, the value is 1. When multiple masters are used, the value is 1 for the first master, 2 for the second master, and the like.
uiDevice	Slave physical address	UINT	1 to 65535	0	Number of a slave, such as 1000 and 1001
usiChannel	Reserved	BYTE	1	1	Reserved
wIndex	Object dictionary index	WORD	0 to 65535	0	Example: Object dictionary index 16#6061 of the servo running mode
bySubindex	Object dictionary sub-index	BYTE	0 to 255	0	Example: Object dictionary sub-index 16#00 of the servo homing mode
udiTimeOut	Timeout period (ms)	UDINT	0 to 65535	0	During the period, if the request to read parameters of the slave is not replied, an error message is displayed.

Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
xDone	Parameter read completion	BOOL	TRUE/FALSE	FALSE	TRUE: Parameter read is completed
xBusy	Parameter read incompletion	BOOL	TRUE/FALSE	FALSE	TRUE: Parameter read is not completed
xError	Parameter read error	BOOL	TRUE/FALSE	FALSE	TRUE: An error occurs during parameter read. The eError parameter also displays the reason for the error.
eError	Error ID	ETC_CO_ ERROR	-	0	-

Output Variable	Name	Data Type	Value Range	Initial Value	Description
udiSdoAbort	More information about an error detected for the device	UDINT	-	0	-
abyData	Read data value	ARRAY[1..4] OF BYTE	-	0	The read data is stored by bytes, with the values filled in the order from array index 1 to array index 4.
usiDataLength	Read data length	BYTE	0 to 255	0	-

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	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	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

ENUM ETC_CO_ERROR

ETC_CO_NO_ERROR	0	No error
ETC_CO_FIRST_ERROR	5750	Error reason stored in the output of udiSdoAbort
ETC_CO_OTHER_ERROR	5751	Master not found
ETC_CO_DATA_OVERFLOW	5752	ETC_CO_Expedited, of which the length is larger than 4.
ETC_CO_TIME_OUT	5753	Timeout
ETC_CO_FIRST_MF	5770	Reserved
ETC_CO_LAST_ERROR	5779	Reserved

Note:

usiCom: EtherCAT master selection

When only one EtherCAT master is used for a device, such as Inovance AM600 or AM400 series, the value of usiCom is 1.

When two EtherCAT masters are used for a device:

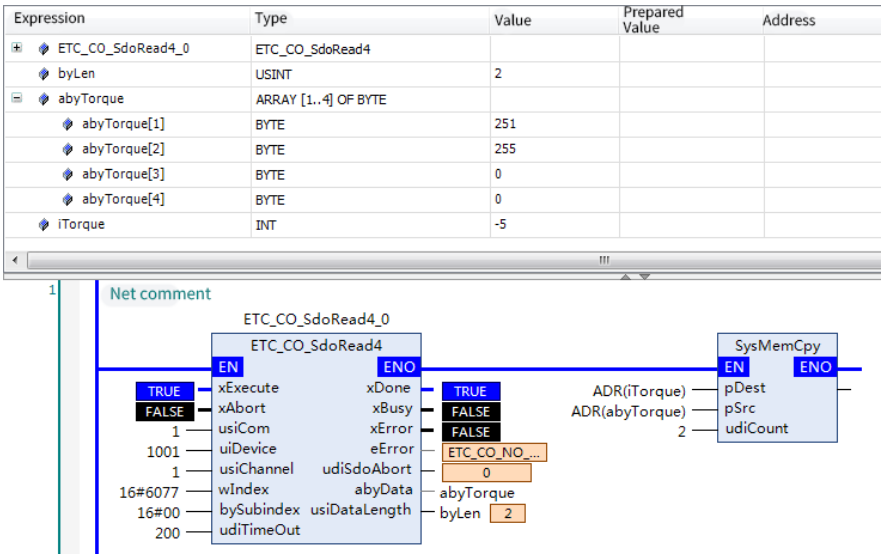
If the value of usiCom is 1, the slave device is under the first EtherCAT master, such as Inovance AC800 series using the EtherCAT bus USB-C port.

If the value of usiCom is 2, the slave device is under the second EtherCAT master, such as Inovance AC800 series using the EtherCAT D-sub bus connector.

■ Program example

Read the object dictionary index 16#6077 and sub-index 16#00 of the real-time torque value of the servo slave.

LD

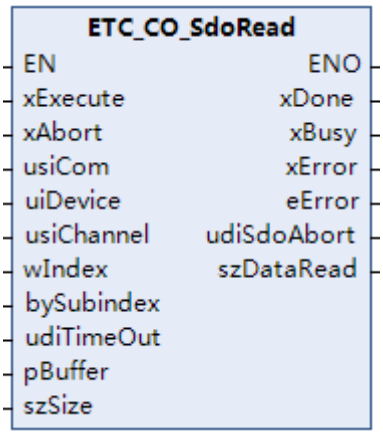


Note: The read data is stored in byte arrays. Use SysmemCpy to copy specific-length data to the corresponding variable.

6.5. 4 ETC_CO_SdoRead

This instruction enables the SDO to read data from the object dictionary of the EtherCAT slave. Different from the ETC_CO_SdoReadDWord function block, this instruction allows to read more than four bytes.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
ETC_CO_SdoRead	Read EtherCAT slave parameters	FB		ETC_CO_SdoRead(xExecute:= , xAbort:= , usiCom:= , uiDevice:= , usiChannel:= , wIndex:= , bySubindex:= , udiTimeOut:= , pBuffer:= , szSize:= , xDone=> , xBusy=> , xError=> , eError=> , udiSdoAbort=> szDataRead=>);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
xExecute	Function block execution	BOOL	TRUE/FALSE	FALSE	TRUE: Stating to read parameters of the slave
xAbort	Function block execution abort	BOOL	TRUE/FALSE	FALSE	TRUE: The current reading process is terminated
usiCom	EtherCAT master count	BYTE	1 to 2	1	When only one EtherCAT master is used, the value is 1. When multiple masters are used, the value is 1 for the first master, 2 for the second master, and the like.
uiDevice	Slave physical address	UINT	1 to 65535	0	Number of a slave, such as 1000 and 1001
usiChannel	Reserved	BYTE	1	1	Reserved
wIndex	Object dictionary index	WORD	0 to 65535	0	Example: Object dictionary index 16#6061 of the servo running mode
bySubindex	Object dictionary sub-index	BYTE	0 to 255	0	Example: Object dictionary sub-index 16#00 of the servo homing mode
pBuffer	Data buffer pointer	POINTER TO BYTE	-	Empty	Area for storing data after successful parameter transmission
szSize	Data buffer size	BYTE	0 to 255	0	Size of the data buffer (pBuffer)
udiTimeOut	Timeout period (ms)	UDINT	0 to 65535	0	During the period, if the request to read parameters of the slave is not replied, an error message is displayed.

Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
xDone	Parameter read completion	BOOL	TRUE/FALSE	FALSE	TRUE: Parameter read is completed
xBusy	Parameter read incompletion	BOOL	TRUE/FALSE	FALSE	TRUE: Parameter read is not completed
xError	Parameter read error	BOOL	TRUE/FALSE	FALSE	TRUE: An error occurs during parameter read. The eError parameter also displays the reason for the error.
eError	Error ID	ETC_CO_ERROR	-	0	-
udiSdoAbort	More information about an error detected for the device	UDINT	-	0	-
szDataRead	Read data length	BYTE	0 to 255	0	Length of read data, in the unit of bytes

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
xExecute	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
xAbort	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
usiCom	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
uiDevice	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-
usiChannel	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
wIndex	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
bySubindex	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
pBuffer	POINTER TO BYTE																			
szSize	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
udiTimeOut	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-
xDone	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
xBusy	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
xError	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
eError	ETC_CO_ERROR																			
udiSdoAbort	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-
szDataRead	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

ENUM ETC_CO_ERROR

ETC_CO_NO_ERROR	0	No error
ETC_CO_FIRST_ERROR	5750	Error reason stored in the output of udiSdoAbort
ETC_CO_OTHER_ERROR	5751	Master not found
ETC_CO_DATA_OVERFLOW	5752	ETC_CO_Expedited, of which the length is larger than 4.
ETC_CO_TIME_OUT	5753	Timeout
ETC_CO_FIRST_MF	5770	Reserved
ETC_CO_LAST_ERROR	5779	Reserved

Note:

usiCom: EtherCAT master selection

When only one EtherCAT master is used for a device, such as Inovance AM600 or AM400 series, the value of usiCom is 1.

When two EtherCAT masters are used for a device:

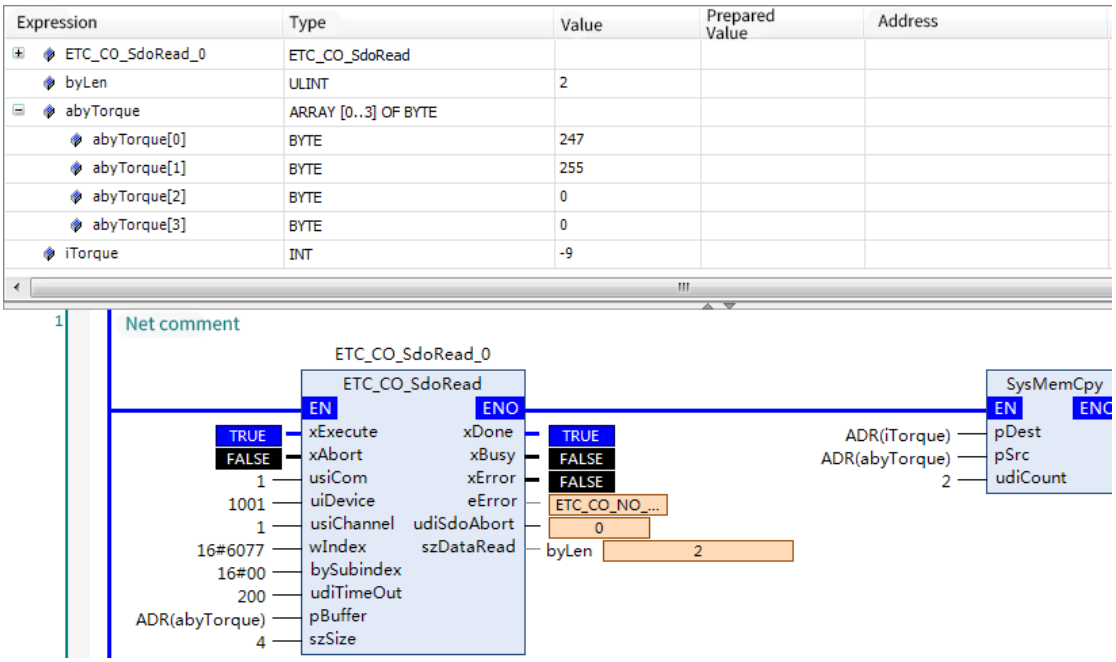
If the value of usiCom is 1, the slave device is under the first EtherCAT master, such as Inovance AC800 series using the EtherCAT bus USB-C port.

If the value of usiCom is 2, the slave device is under the second EtherCAT master, such as Inovance AC800 series using the EtherCAT D-sub bus connector.

■ Program example

Read the object dictionary index 16#6077 and sub-index 16#00 of the real-time torque value of the servo slave.

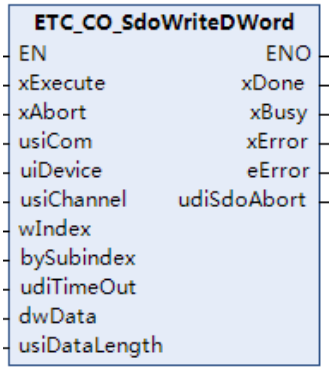
LD



6.5. 5 ETC_CO_SdoWriteDWord

This instruction enables the SDO to write up to four bytes to the object dictionary of the EtherCAT slave.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
ETC_CO_SdoWriteDWord	Write EtherCAT slave parameters as DWORD	FB		ETC_CO_SdoWriteDWord(xExecute:= , xAbort:= , usiCom:= , uiDevice:= , usiChannel:= , wIndex:= , bySubindex:= , udiTimeOut:= , dwData:= , usiDataLength:=, xDone=> , xBusy=> , xError=> , eError=> , udiSdoAbort=>);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
xExecute	Function block execution	BOOL	TRUE/FALSE	FALSE	TRUE: Stating to read parameters of the slave
xAbort	Function block execution abort	BOOL	TRUE/FALSE	FALSE	TRUE: The current reading process is terminated

Input Variable	Name	Data Type	Value Range	Initial Value	Description
usiCom	EtherCAT master count	BYTE	1 to 2	1	When only one EtherCAT master is used, the value is 1. When multiple masters are used, the value is 1 for the first master, 2 for the second master, and the like.
uiDevice	Slave physical address	UINT	1 to 65535	0	Number of a slave, such as 1000 and 1001
usiChannel	Reserved	BYTE	1	1	Reserved
wIndex	Object dictionary index	WORD	0 to 65535	0	Example: Object dictionary index 16#6061 of the servo running mode
bySubindex	Object dictionary sub-index	BYTE	0 to 255	0	Example: Object dictionary sub-index 16#00 of the servo homing mode
udiTimeOut	Timeout period (ms)	UDINT	0 to 65535	0	During the period, if the request to read parameters of the slave is not replied, an error message is displayed.
dwData	Data value to write	DWORD	-	0	Data value to write to the object dictionary
usiDataLength	Write length	USINT	0 to 255	0	Length of data to write

Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
xDone	Parameter read completion	BOOL	TRUE/FALSE	FALSE	TRUE: Parameter read is completed
xBusy	Parameter read incompletion	BOOL	TRUE/FALSE	FALSE	TRUE: Parameter read is not completed
xError	Parameter read error	BOOL	TRUE/FALSE	FALSE	TRUE: An error occurs during parameter read. The eError parameter also displays the reason for the error.
eError	Error ID	ETC_CO_ERROR	-	0	-
udiSdoAbort	More information about an error detected for the device	UDINT	-	0	-

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	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	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-

ENUM ETC_CO_ERROR

ETC_CO_NO_ERROR	0	No error
ETC_CO_FIRST_ERROR	5750	Error reason stored in the output of udiSdoAbort
ETC_CO_OTHER_ERROR	5751	Master not found
ETC_CO_DATA_OVERFLOW	5752	ETC_CO_Expedited, of which the length is larger than 4.
ETC_CO_TIME_OUT	5753	Timeout
ETC_CO_FIRST_MF	5770	Reserved
ETC_CO_LAST_ERROR	5779	Reserved

Note:

usiCom: EtherCAT master selection

When only one EtherCAT master is used for a device, such as Inovance AM600 or AM400 series, the value of usiCom is 1.

When two EtherCAT masters are used for a device:

If the value of usiCom is 1, the slave device is under the first EtherCAT master, such as Inovance AC800 series using the EtherCAT bus USB-C port.

If the value of usiCom is 2, the slave device is under the second EtherCAT master, such as Inovance AC800 series using the EtherCAT D-sub bus connector.

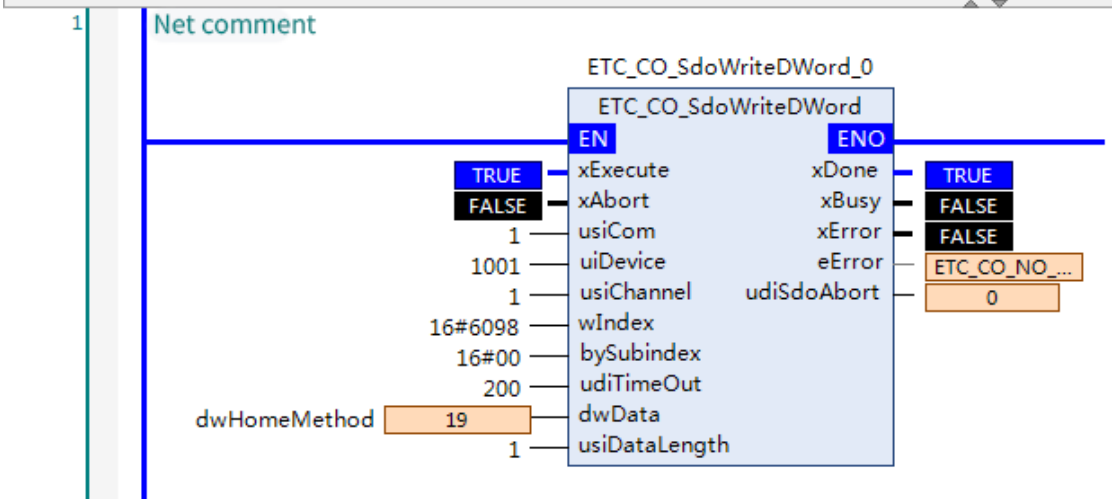
■ Program example

Example 1

LD

Write the object dictionary index 16#6098 and sub-index 16#00 of the servo slave homing mode.

Expression	Type	Value	Prepared Value
byLen	ULINT	0	
iTorque	INT	0	
ETC_CO_SdoWriteDWord_0	ETC_CO_SdoWriteDWord		
dwHomeMethod	DWORD	19	



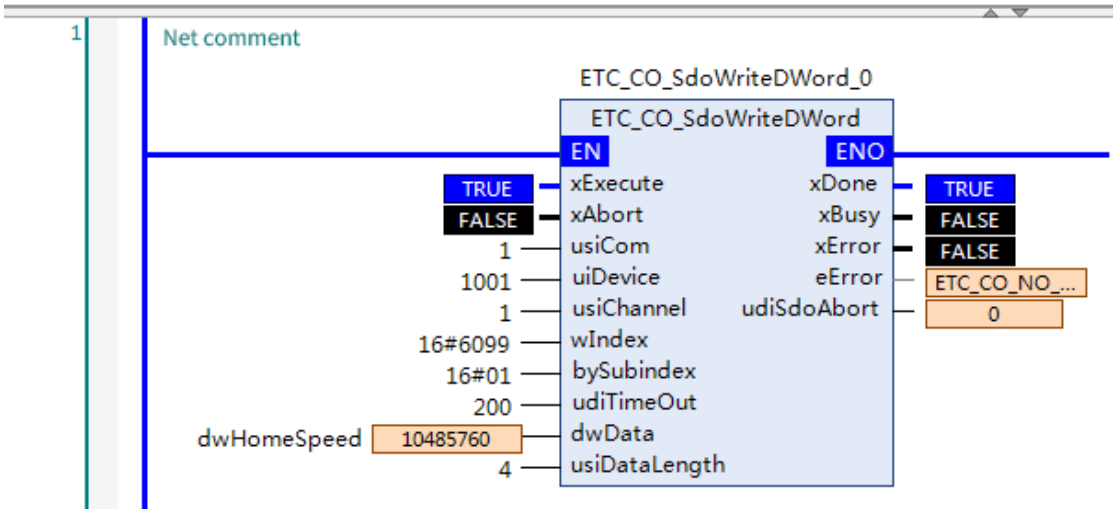
Note: Change the value of 16#6098 of the homing mode to 19, with the written data length of 1 byte.

Example 2

LD

Write the object dictionary index 16#6099 and sub-index 16#01 of the servo slave homing speed.

Expression	Type	Value	Prepared Value
byLen	ULINT	0	
iTorque	INT	0	
ETC_CO_SdoWriteDWord_0	ETC_CO_SdoWriteDWord		
dwHomeSpeed	DWORD	10485760	

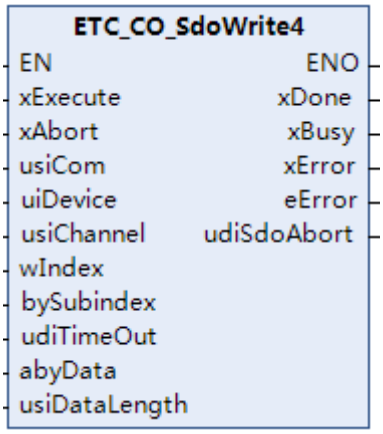


Note: Change the homing speed (index 16#6099 and sub-index 16#01) to 10485760 (pulse unit), with the data length of four bytes.

6.5. 6 ETC_CO_SdoWrite4

This instruction enables the SDO to write up to four bytes to the object dictionary of the EtherCAT slave. Different from the ETC_CO_SdoWriteDWord function block, this instruction allows to write values to byte arrays.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
ETC_CO_SdoWrite4	Write EtherCAT slave parameters that are no longer than 4 bytes	FB		ETC_CO_SdoWrite4(xExecute:= , xAbort:= , usiCom:= , uiDevice:= , usiChannel:= , wIndex:= , bySubindex:= , udiTimeOut:= , abyData:= , usiDataLength:= , xDone=> , xBusy=> , xError=> , eError=> , udiSdoAbort=>);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
xExecute	Function block execution	BOOL	TRUE/FALSE	FALSE	TRUE: Stating to read parameters of the slave
xAbort	Function block execution abort	BOOL	TRUE/FALSE	FALSE	TRUE: The current reading process is terminated
usiCom	EtherCAT master count	BYTE	1 to 2	1	When only one EtherCAT master is used, the value is 1. When multiple masters are used, the value is 1 for the first master, 2 for the second master, and the like.
uiDevice	Slave physical address	UINT	0 to 65535	0	Number of a slave, such as 1000 and 1001
usiChannel	Reserved	BYTE	1	1	Reserved
wIndex	Object dictionary index	WORD	0 to 65535	0	Example: Object dictionary index 16#6061 of the servo running mode
bySubindex	Object dictionary sub-index	BYTE	0 to 255	0	Example: Object dictionary sub-index 16#00 of the servo homing mode
udiTimeOut	Timeout period (ms)	UDINT	0 to 65535	0	During the period, if the request to read parameters of the slave is not replied, an error message is displayed.
abyData	Data value to write	ARRAY[1..4] OF BYTE	-	0	The data is written by bytes, with the values filled in the order from array index 1 to array index 4.
usiDataLength	Write length	USINT	0 to 255	0	Length of data to write

Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
xDone	Parameter read completion	BOOL	TRUE/FALSE	FALSE	TRUE: Parameter read is completed
xBusy	Parameter read incompletion	BOOL	TRUE/FALSE	FALSE	TRUE: Parameter read is not completed
xError	Parameter read error	BOOL	TRUE/FALSE	FALSE	TRUE: An error occurs during parameter read. The eError parameter also displays the reason for the error.
eError	Error ID	ETC_CO_ERROR	-	0	-
udiSdoAbort	More information about an error detected for the device	UDINT	-	0	-

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	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	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-
abyData	ARRAY[1..4] OF BYTE																			
usiDataLength	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-
xDone	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
xBusy	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
xError	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
eError	ETC_CO_ERROR																			
udiSdoAbort	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-

ENUM ETC_CO_ERROR

ETC_CO_NO_ERROR	0	No error
ETC_CO_FIRST_ERROR	5750	Error reason stored in the output of udiSdoAbort
ETC_CO_OTHER_ERROR	5751	Master not found
ETC_CO_DATA_OVERFLOW	5752	ETC_CO_Expedited, of which the length is larger than 4.
ETC_CO_TIME_OUT	5753	Timeout
ETC_CO_FIRST_MF	5770	Reserved
ETC_CO_LAST_ERROR	5779	Reserved

Note:

usiCom: EtherCAT master selection

When only one EtherCAT master is used for a device, such as Inovance AM600 or AM400 series, the value

of usiCom is 1.

When two EtherCAT masters are used for a device:

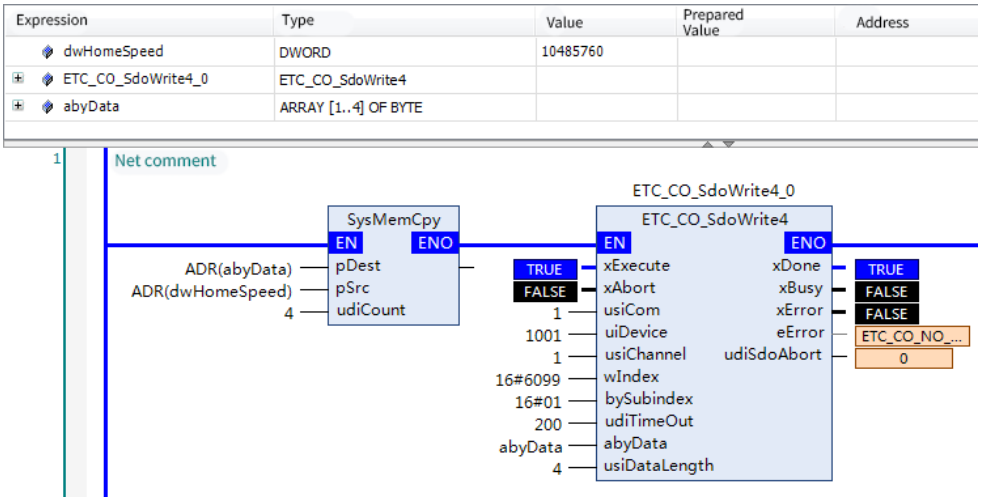
If the value of usiCom is 1, the slave device is under the first EtherCAT master, such as Inovance AC800 series using the EtherCAT bus USB-C port.

If the value of usiCom is 2, the slave device is under the second EtherCAT master, such as Inovance AC800 series using the EtherCAT D-sub bus connector.

■ Program example

Write the object dictionary index 16#6099 and sub-index 16#01 of the servo slave homing speed.

LD

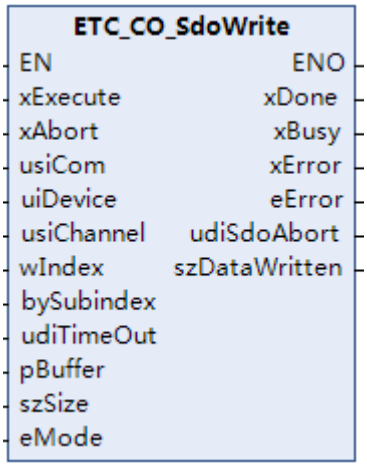


Note: Change the homing speed (index 16#6099 and sub-index 16#01) to 10485760 (pulse unit).

6.5. 7 ETC_CO_SdoWrite

This instruction enables the SDO to write bytes to the object dictionary of the EtherCAT slave. This instruction allows to write more than four bytes.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
ETC_CO_SdoWrite	Write EtherCAT slave parameters	FB		ETC_CO_SdoWrite(xExecute:= , xAbort:= , usiCom:= , uiDevice:= , usiChannel:= , wIndex:= , bySubindex:= , udiTimeOut:= , pBuffer:= , szSize:= , eMode:= , xDone=> , xBusy=> , xError=> , eError=> , udiSdoAbort=> , szDataWritten=>);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
xExecute	Function block execution	BOOL	TRUE/FALSE	FALSE	TRUE: Stating to read parameters of the slave
xAbort	Function block execution abort	BOOL	TRUE/FALSE	FALSE	TRUE: The current reading process is terminated
usiCom	EtherCAT master count	BYTE	1 to 2	1	When only one EtherCAT master is used, the value is 1. When multiple masters are used, the value is 1 for the first master, 2 for the second master, and the like.
uiDevice	Slave physical address	UINT	0 to 65535	0	Number of a slave, such as 1000 and 1001
usiChannel	Reserved	BYTE	1	1	Reserved
wIndex	Object dictionary index	WORD	0 to 65535	0	Example: Object dictionary index 16#6061 of the servo running mode
bySubindex	Object dictionary sub-index	BYTE	0 to 255	0	Example: Object dictionary sub-index 16#00 of the servo homing mode
udiTimeOut	Timeout period (ms)	UDINT	0 to 65535	0	During the period, if the request to read parameters of the slave is not replied, an error message is displayed.
pBuffer	Pointer head address of data to write	POINTER TO BYTE	-	-	Pointer head address of data to write
szSize	Write length	USINT	0 to 255	0	Length of data to write
eMode	Write mode	ETC_CO_MODE	-	0	-

Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
xDone	Parameter read completion	BOOL	TRUE/FALSE	FALSE	TRUE: Parameter read is completed
xBusy	Parameter read incompletion	BOOL	TRUE/FALSE	FALSE	TRUE: Parameter read is not completed
xError	Parameter read error	BOOL	TRUE/FALSE	FALSE	TRUE: An error occurs during parameter read. The eError parameter also displays the reason for the error.
eError	Error ID	ETC_CO_ERROR	-	0	-
udiSdoAbort	More information about an error detected for the device	UDINT	-	0	-
szDataWritten	Length of written data	BYTE	0 to 255	0	Length of written data

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	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	POINTER TO BYTE																			
szSize	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-
eMode	ETC_CO_MODE																			
xDone	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
xBusy	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
xError	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
eError	ETC_CO_ERROR																			
udiSdoAbort	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-
szDataWritten	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

ENUM ETC_CO_ERROR

ETC_CO_NO_ERROR	0	No error
ETC_CO_FIRST_ERROR	5750	Error reason stored in the output of udiSdoAbort
ETC_CO_OTHER_ERROR	5751	Master not found
ETC_CO_DATA_OVERFLOW	5752	ETC_CO_Expedited, of which the length is larger than 4.
ETC_CO_TIME_OUT	5753	Timeout
ETC_CO_FIRST_MF	5770	Reserved
ETC_CO_LAST_ERROR	5779	Reserved

ENUM ETC_CO_MODE

AUTO	0	The customer selects the automatic mode.
EXPEDITED	1	The customer uses the acceleration protocol.
SEGMENTED	2	The customer uses the segmented transmission protocol.

Note:

usiCom: EtherCAT master selection

When only one EtherCAT master is used for a device, such as Inovance AM600 or AM400 series, the value of usiCom is 1.

When two EtherCAT masters are used for a device:

If the value of usiCom is 1, the slave device is under the first EtherCAT master, such as Inovance AC800 series using the EtherCAT bus USB-C port.

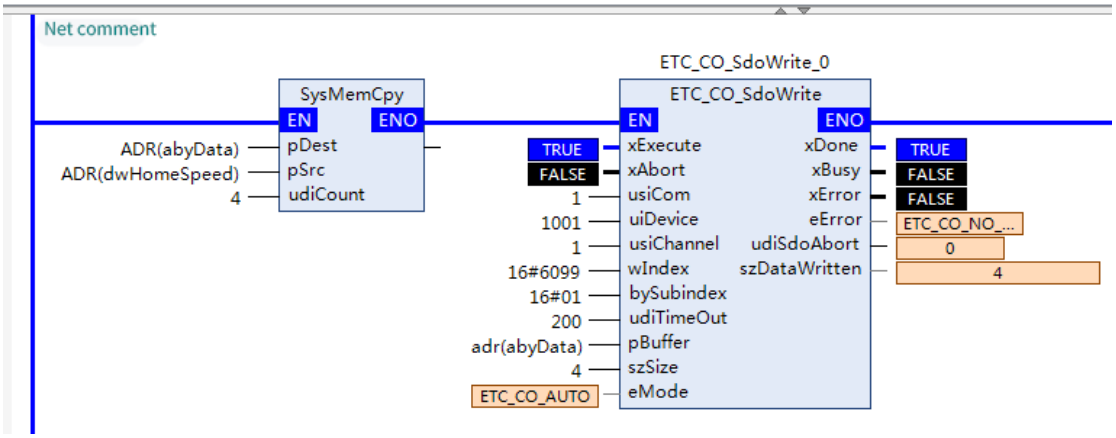
If the value of usiCom is 2, the slave device is under the second EtherCAT master, such as Inovance AC800 series using the EtherCAT D-sub bus connector.

■ Program example

Write the object dictionary index 16#6099 and sub-index 16#01 of the servo slave homing speed.

LD

Expression	Type	Value	Prepared Value	Address
dwHomeSpeed	DWORD	10485760		
ETC_CO_SdoWrite_0	ETC_CO_SdoWrite			
abyData	ARRAY [1..4] OF BYTE			



Note: Change the homing speed (index 16#6099 and sub-index 16#01) to 10485760 (pulse unit).

6.5. 8 IoDrvEtherCAT

This instruction restarts the EtherCAT master using an EtherCAT master instance.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
IoDrvEtherCAT	Implicit instance of EtherCAT master	FB	IoDrvEtherCAT EN ENO xRestart xConfigFinished xStopBus xDistributedClockInSync xError xSyncInWindow	EtherCAT(xRestart:= , xStopBus:= , xConfigFinished=> , xDistributedClockInSync=> , xError=> , xSyncInWindow=>);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
xRestart	EtherCAT master restart after reset	BOOL	TRUE/FALSE	FALSE	Restart of EtherCAT master upon rising edge trigger

Input Variable	Name	Data Type	Value Range	Initial Value	Description
xStopBus	Stop of EtherCAT master	BOOL	TRUE/FALSE	FALSE	Stop of EtherCAT master

Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
xDistributedClockInSync	DC sync signal	BOOL	TRUE/FALSE	FALSE	TRUE: Synchronization is completed in DC mode
xConfigFinished	EtherCAT master configuration completion	BOOL	TRUE/FALSE	FALSE	TRUE: EtherCAT configuration is completed and communication is ongoing
xError	A parameter error occurs.	BOOL	TRUE/FALSE	FALSE	TRUE: An error occurs, for example, the slave communication is interrupted

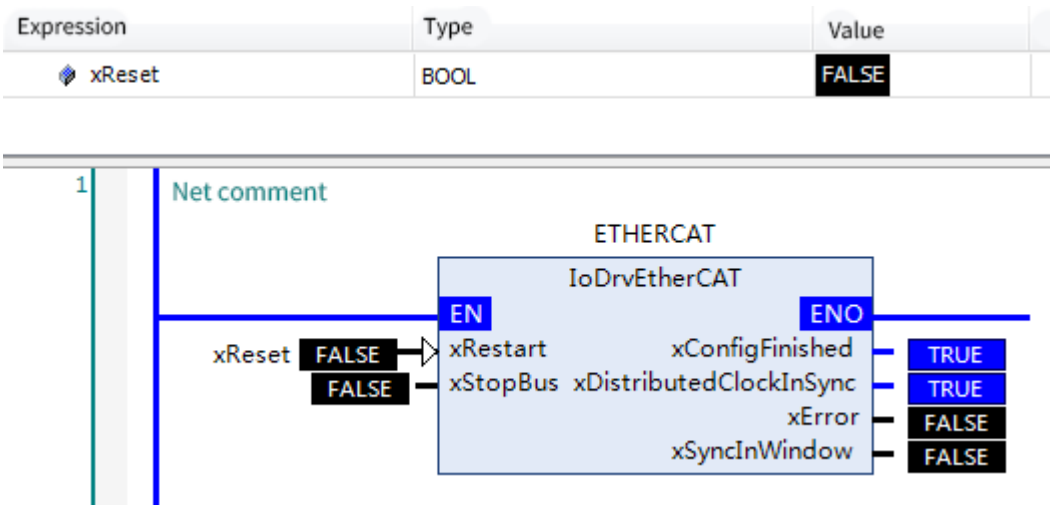
	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
xExecute	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
xAbort	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
usiCom	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
uiDevice	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
usiChannel	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

■ Program example

Example 1

EtherCAT master restart after reset

LD



xRestart: Restart when activated at the rising edge

ST

```
PROGRAM POU
VAR
    xReset: BOOL;
    xReset_R: R_TRIG;
END_VAR

xReset_R(CLK:= xReset, Q=> );
EtherCAT(
    xRestart:=xReset_R.Q ,
    xStopBus:= ,
    xConfigFinished=> ,
    xDistributedClockInSync=> ,
    xError=> ,
    xSyncInWindow=> );
```

Note:

FB instance

When only one EtherCAT master is used for a device, such as Inovance AM600 or AM400 series, the instance name should be ETHERCAT.

When two EtherCAT masters are used for a device, such as Inovance AC800 series, the instance name should be ETHERCAT_C or ETHERCAT_D.

Example 2

Determination on the EtherCAT master communication state

ST

```
PROGRAM PLC_PRG
VAR
    xETC_StateOK: BOOL; //EtherCAT master communication state
END_VAR
```

```
// EtherCAT master communication state
xETC_StateOK:=ETHERCAT.xConfigFinished AND ETHERCAT.xDistributedClockInSync AND NOT ETHERCAT.xError ;
```

6.5. 9 ETCSlave

This instruction obtains, switches, and checks the slave state using EtherCAT slave instances.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
ETCSlave	Implicit instance of EtherCAT slave	FB	ETCSlave EN ENO xSetOperational wState	ETCSlave(xSetOperational:= , wState=>);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
xSetOperational	Slave state obtaining	BOOL	TRUE/FALSE	FALSE	At the rising edge, the slave station communication state machine is attempted to be set as ETC_SLAVE_OPERATIONAL.

Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
wState	Slave state	WORD	0 to 8	0	Current state of the slave. The possible values include: 0: ETC_SLAVE_BOOT 1: ETC_SLAVE_INIT 2: ETC_SLAVE_PREOPERATIONAL 4: ETC_SLAVE_SAVEOPERATIONAL 8: ETC_SLAVE_OPERATIONAL

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
xSetOperational	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
wState	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

■ Program example

The ETC_SLAVE_OPERATIONAL state indicates that configuration is completed. If a configuration error occurs, the device may return to the previous state.

Example 1

Taking IS620N as an example, add the instance name IS620N. Definition: nSlaveState: ETC_SLAVE_STAT

ST Example	Description
IS620N(xSetOperational:= , wState=> nSlaveState);	By invoking the slave IS620N implicit instance, the slave state is output to the nSlaveState variable.

Example 2

Check all slave link tables.

A function block instance is created between each pair of EtherCAT master and EtherCAT slave in implicit way. The instance monitors the status of each slave. Therefore, this instance must be invoked in the application program. The slave state is read by using wState. To simplify the programming, all masters and slaves can be found in link tables.

Therefore, all slaves can be checked cyclically by a simple WHILE. The masters and slaves correspond to attributes NextInstance and LastInstance, respectively, returning the pointers to next and previous slaves. In addition, the FirstSlave attribute of the master is effective. It provides the pointer to the first slave.

Link table function example

Check all slave states. Definition: pSlave: POINTER TO ETCSlave;

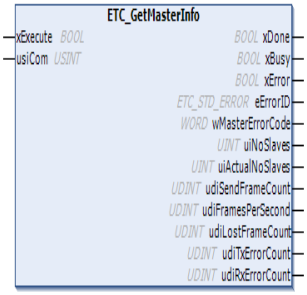
ST Example	Description

pSlave := EtherCAT_Master.FirstSlave;
WHILE pSlave <> 0 DO
pSlave^();
//TODO: User-added code, for example, to count the OP state of the slave.
pSlave := pSlave^.NextInstance;
END_WHILE
	The first slave of the master is found by using EtherCAT_Master.FirstSlave
	Instances are invoked in WHILE loop to determine wState.
	Then the status is checked.
	The pointer to the next slave is found by using pSlave^.NextInstance.
	The pointer at the end of table is null, and the loop is finished.

6.5. 10 ETC_GetMasterInfo

This instruction gets EtherCAT master information.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
ETC_ GetMasterInfo	Get EtherCAT master information	FB		ETC_GetMasterInfo(xExecute:=, usiCom:= , xDone=> , xBusy=> , xError=> , eErrorID=> , wMasterErrorCode=> , uiNoSlaves=> , uiActualNoSlaves=> , udiSendFrameCount=> , udiFramesPerSecond=> , udiLostFrameCount=> , udiTxErrorCount=> , udiRxErrorCount=>);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
xExecute	Function block execution	BOOL	TRUE/FALSE	FALSE	TRUE: Stating to read parameters of the slave
usiCom	EtherCAT master count	USINT	1 to 2	1	When only one EtherCAT master is used, the value is 1. When multiple masters are used, the value is 1 for the first master, 2 for the second master, and the like.

Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
xDone	Parameter read completion	BOOL	TRUE/FALSE	FALSE	TRUE: Parameter read is completed.
xBusy	Parameter read incompletion	BOOL	TRUE/FALSE	FALSE	TRUE: Parameter read is not completed.
xError	Parameter read error	BOOL	TRUE/FALSE	FALSE	The value is "TRUE" when an error occurs. The eError parameter also displays the reason for the error.
eErrorID	Error ID	ETC_STD_ ERROR	-	0	Error ID
wMasterError-Code	Master error code	WORD	-	0	-
uiNoSlaves	Number of slaves	UINT	-	0	-
uiActual-NoSlaves	Number of online slaves	UINT	-	0	-
udiSendFrame-Count	Number of frames sent by master	UDINT	-	0	-
udiFramesPer-Second	Frame sending frequency at master	UDINT	-	0	-

Output Variable	Name	Data Type	Value Range	Initial Value	Description
udiLostFrame-Count	Number of frames lost at master	UDINT	-	0	-
udiTxErrorCount	Number of error frames sent from master	UDINT	-	0	-
udiRxErrorCount	Number of error frames received by master	UDINT	-	0	-

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
xExecute	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
usiCom	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-
xDone	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
xBusy	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
xError	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
eErrorID	ETC_STD_ERROR																			
wMaster ErrorCode	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
uiNoSlaves	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-
uiActual NoSlaves	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-
udiSend FrameCount	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-
udiFrames PerSecond	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-
udiLost FrameCount	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-
udiTxError Count	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-
udiRxError Count	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-

ENUM ETC_STD_ERROR

ETC_NO_ERROR	0	No error
ETC_CFG_NOSLAVE	1	Slave configuration not completed
ETC_CFG_NOMASTER	2	Master index nonexistent
ETC_CFG_LOOPNETERROR	3	Network topology as a looped network
ETC_CO_GETSLAVEERROR	4	Current slave not found
ETC_OTHER_ERROR	5	Other error

Notes:

usiCom: EtherCAT master selection

When only one EtherCAT master is used for a device, such as Inovance AM600 or AM400 series, the value of usiCom is 1.

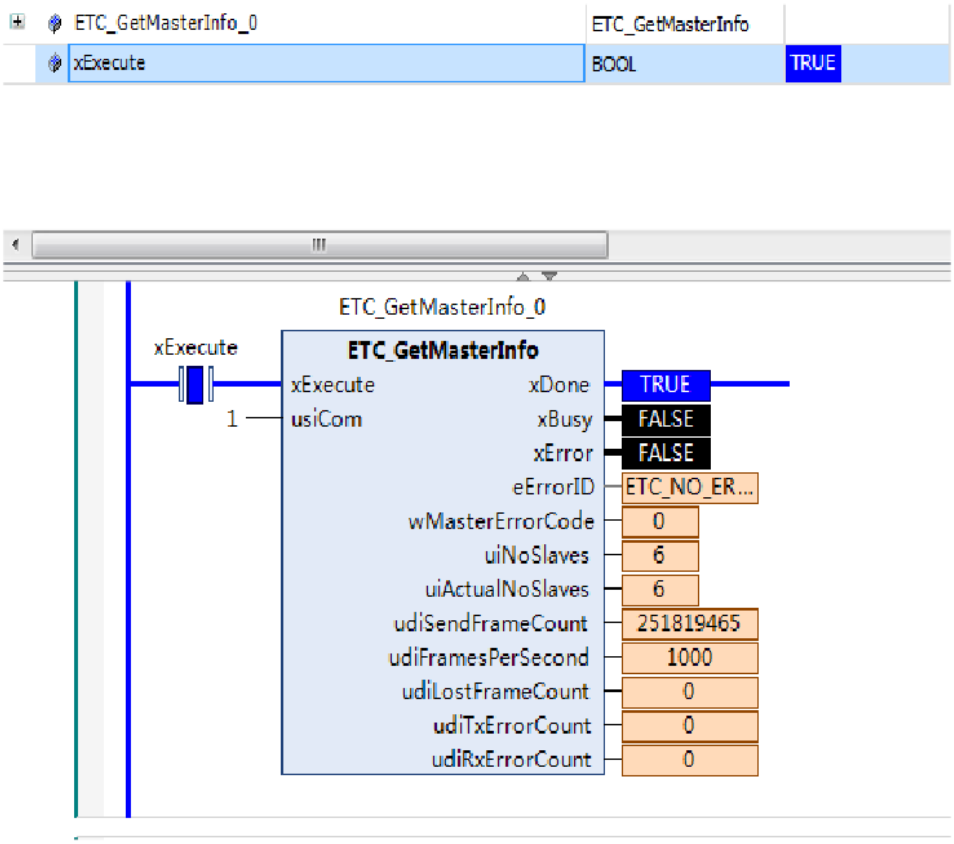
When two EtherCAT masters are used for a device:

- If the value of usiCom is 1, the slave device is under the first EtherCAT master, such as Inovance AC800 series using the EtherCAT bus USB-C port.
- If the value of usiCom is 2, the slave device is under the second EtherCAT master, such as Inovance AC800 series using the EtherCAT D-sub bus connector.

■ Program example

This instruction gets master information.

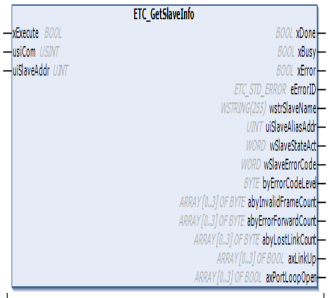
LD



6.5. 11 ETC_GetSlaveInfo

This instruction gets EtherCAT slave information.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
ETC_ GetSlaveInfo	Get EtherCAT slave information	FB		ETC_GetSlaveInfo_0(xExecute:= , usiCom:= , uiSlaveAddr:= , xDone=> , xBusy=> , xError=> , eErrorID=> , wstrSlaveName=> , uiSlaveAliasAddr=> , wSlaveStateAct=> , wSlaveErrorCode=> , byErrorCodeLevel=> , abyInvalidFrameCount=> , abyErrorForwardCount=> , abyLostLinkCount=> , axLinkUp=> , axPortLoopOpen=>);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
xExecute	Function block execution	BOOL	TRUE/FALSE	FALSE	TRUE: Stating to read parameters of the slave
usiCom	EtherCAT master index	USINT	1 to 2	1	When only one EtherCAT master is used, the value is 1. When multiple masters are used, the value is 1 for the first master, 2 for the second master, and the like.
uiSlaveAddr	Slave address	UINT	1 to 65535	0	Slave formal address

Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
xDone	Parameter read completion	BOOL	TRUE/FALSE	FALSE	TRUE: Parameter read is completed.
xBusy	Parameter read incompletion	BOOL	TRUE/FALSE	FALSE	TRUE: Parameter read is not completed.
xError	Parameter read error	BOOL	TRUE/FALSE	FALSE	The value is "TRUE" when an error occurs. The eError parameter also displays the reason for the error.
eErrorID	Error ID	ETC_STD_ERROR	-	0	Error ID
wstrSlaveName	Slave name	WSTRING(255)	-		Slave device name
uiSlaveAliasAddr	Slave alias address	UINT	-	0	Slave alias address, which is displayed only when the alias is used
wSlaveStateAct	Slave state	WORD	-	0	Slave state
wSlaveErrorCode	Slave error code	WORD	-	0	Slave error code, corresponding to slave fault information, which can be queried in the user guide

Output Variable	Name	Data Type	Value Range	Initial Value	Description
byErrorCodeLevel	Slave error code level	BYTE	-	0	Slave error code level
abyInvalidFrame-Count	Invalid frame count	ARRAY[0..3] OF BYTE	-	0	Slave CRC check error. A CRC check error but not any Ethernet frame error exists in the network.
abyErrorForward-Count	Count of forwarded frames	ARRAY[0..3] OF BYTE	-	0	Data forward error: The received data is incorrect and the additional error markings are added at the backend, indicating that it has been processed by other stations.
abyLostLinkCount	LinkLostCount	ARRAY[0..3] OF BYTE	-	0	Slave link loss count
axLinkUp	Port connection state	ARRAY[0..3] OF BOOL	-	FALSE	Master connection state
axPortLoopOpen	Port open state	ARRAY[0..3] OF BOOL	-	FALSE	Slave port state

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
xExecute	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
usiCom	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-
uiSlaveAddr	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-
xDone	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
xBusy	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
xError	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
eErrorID	ETC_STD_ERROR																			
wstrSlave Name	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√
uiSlave AliasAddr	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-
wSlave StateAct	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
wSlave ErrorCode	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
byError CodeLevel	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
abyInvalid FrameCount	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
abyError ForwardCount	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
abyLost LinkCount	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
axLinkUp	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
axPortLoop Open	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

ENUM ETC_STD_ERROR

ETC_NO_ERROR	0	No error
ETC_CFG_NOSLAVE	1	Slave configuration not completed
ETC_CFG_NOMASTER	2	Master index nonexistent
ETC_CFG_LOOPNETERROR	3	Network topology as a looped network
ETC_CO_GETSLAVEERROR	4	Current slave not found
ETC_OTHER_ERROR	5	Other error

Notes:

usiCom: EtherCAT master selection

When only one EtherCAT master is used for a device, such as Inovance AM600 or AM400 series, the value of usiCom is 1.

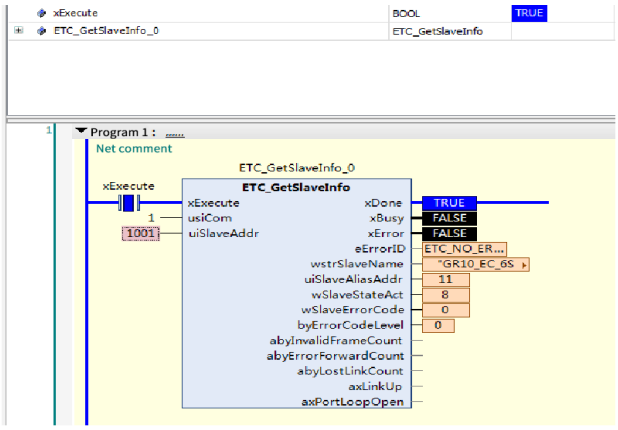
When two EtherCAT masters are used for a device:

- If the value of usiCom is 1, the slave device is under the first EtherCAT master, such as Inovance AC800 series using the EtherCAT bus USB-C port.
- If the value of usiCom is 2, the slave device is under the second EtherCAT master, such as Inovance AC800 series using the EtherCAT D-sub bus connector.

■ Program example

This instruction gets master information.

LD



6. 6 EtherNet/IP Communication Instructions

6.6. 1 Instruction List

Instruction Category	Name	FB/FC	Function
EtherNet/IP communication instructions	IoDrvEtherNetIP	FB	Implicit instance of EtherNet/IP
	RemoteAdapter	FB	Remote EtherNet/IP adapter
	Generic_Service	FB	Performs generic service at EtherNet/IP adapter
	Get_Attributes_All	FB	Get attributes of specific instance of all CIP object
	Get_Attribute_Single	FB	Get attribute of specific instance of single CIP object
	Set_Attributes_All	FB	Set attributes of specific instance of all CIP object
	Set_Attribute_Single	FB	Set attribute of specific instance of single CIP object

6.6. 2 IoDrvEtherNetIP

This instruction implements instances of the Ethernet/IP master.

■ Instruction format

Instruction	Name	FB/ FC	LD Expression	ST Expression
IoDrvEtherNetIP	Implicit instance of EtherNet/IP	FB		IoDrvEtherNetIP. IoDrvEtherNetIP(xReset:= , eState=> , eError=>);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
xReset	Reset	BOOL	-	-	Reset of the master through scanning upon rising edge trigger

Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
eState	State	ScannerState	-	-	EtherNet/IP status
eError	Error	ERROR	-	-	When the master is in the error state, an error occurs.

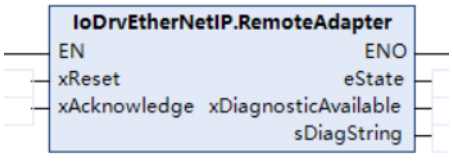
■ Program example

For details, see *"6.6. 9 Instances for Ethernet/IP Communication Reset Instructions" on page 894*.

6.6. 3 RemoteAdapter

This instruction implements instances of the Ethernet/IP slave.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
RemoteAdapter	Remote EtherNet/IP adapter	FB		IoDrvEtherNetIP.RemoteAdapter(xReset:= , xAcknowledge:= , eState=> , xDiagnosticAvailable=> , sDiagString=>);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
xReset	Reset	BOOL	-	-	Reset of slave instances upon rising edge trigger

Input Variable	Name	Data Type	Value Range	Initial Value	Description
xAcknowledge	Acknowledge information	BOOL	-	-	Acknowledge of diagnosis information The diagnosis information is available and the corresponding pin will be reset.

Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
eState	Current state	AdapterState	-	-	EtherNet/IP slave state
xDiagnosticAvailable	Diagnosis availability	BOOL	-	-	TRUE: The diagnosis information is available
sDiagString	Diagnosis information	String	-	-	Diagnosis information text string, which is displayed on the device status interface

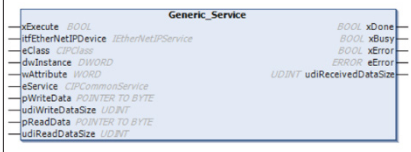
■ Program example

For details, see *"6.6. 9 Instances for Ethernet/IP Communication Reset Instructions" on page 894*.

6.6. 4 Generic_Service

This instruction provides access to the Universal Common Messaging (UCMM) interface of an EtherNet/IP adapter. It allows to send messages in UCMM format to display message requests.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
Generic_Service	Performs generic service at EtherNet/IP adapter	FB		genericService(xExecute:= , xDone=> , xBusy=> , xError=> , itfEtherNetIPDevice:= , eClass:= , dwInstance:= , eError=> , wAttribute:= , eService:= , pWriteData:= , udiWriteDataSize:= , pReadData:= , udiReadDataSize:= , udiReceivedDataSize=>);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
itfEtherNetIPDevice	Remote slave name	IEtherNetIPService	-	-	Instance name of the remote slave

Input Variable	Name	Data Type	Value Range	Initial Value	Description
eClass	Class	CIPClass	-	-	Class of an object that a service is directed towards, which can be a CIP standard target object or a third-party vendor-defined object
dwInstance	Instance	DWORD	-	-	Instance under a class, which should be greater than or equal to 1
wAttribute	Attribute	WORD	-	-	Attribute that a service is directed towards
eService	Common service	CIPCommonService	-	-	CIP common service
pWriteData	Write buffer	POINTER TO BYTE	-	-	Pointer directing towards the attribute data, which writes data to the destination slave
udiWriteDataSize	Write length	UDINT	-	-	Length of data to write
pReadData	Cache to receive	POINTER TO BYTE	-	-	Pointer directing towards the attribute data, which receives data from the destination slave
udiReadDataSize	Receive length	UDINT	-	-	Length of received data

Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
eError	Error Code	ERROR	-	-	Function block error ID
udiReceivedDataSize	Length of received data	UDINT	-	-	Length of received data

■ Program example

For details, see *"6.6. 9 Instances for Ethernet/IP Communication Reset Instructions" on page 894*.

6.6. 5 Get_Attributes_All

This instruction gets all attributes of a specific object instance.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
Get_Attributes_All	Get attributes of specific instance of all CIP object	FB		get_attributes_all(xExecute:= , xDone=> , xBusy=> , xError=> , itfEtherNetIPDevice:= , eClass:= , dwInstance:= , eError=> , pData:= , udiDataSize:= , udiReceivedDataSize=>);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
itfEtherNetIPDevice	Remote slave name		-	-	Instance name of the remote slave
eClass	Class	CIPClass	-	-	Class of an object that a service is directed towards, which can be a CIP standard target object or a third-party vendor-defined object
dwInstance	Instance	DWORD	-	-	Instance under a class, which should be greater than or equal to 1
pData	Cache to receive	POINTER TO BYTE	-	-	Pointer directing towards the attribute data, which receives data from the destination slave
udiDataSize	Receive length	UDINT	-	-	Length of received data

Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
eError	Error Code	ERROR	-	-	Function block error ID
udiReceivedDataSize	Length of received data	UDINT	-	-	Length of received data

■ Program example

For details, see *"6.6. 9 Instances for Ethernet/IP Communication Reset Instructions" on page 894*.

6.6. 6 Get_Attribute_Single

This instruction gets one attribute of a specific object instance.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
Get_Attribute_Single	Get attribute of specific instance of single CIP object	FB		Get_Attribute_Single_0(xExecute:= , xDone=> , xBusy=> , xError=> , itfEtherNetIPDevice:= , eClass:= , dwInstance:= , eError=> , wAttribute:= , pData:= , udiDataSize:= udiReceivedDataSize=>);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
itfEtherNetIPDevice	Remote slave name	IEtherNetIPService	-	-	Instance name of the remote slave
eClass	Class	CIPClass	-	-	Class of an object that a service is directed towards, which can be a CIP standard target object or a third-party vendor-defined object
dwInstance	Instance	DWORD	-	-	Instance under a class, which should be greater than or equal to 1
wAttribute	Attribute	WORD	-	-	Attribute that a service is directed towards
pData	Cache to receive	POINTER TO BYTE	-	-	Pointer directing towards the attribute data, which receives data from the destination slave
udiDataSize	Receive length	UDINT	-	-	Length of received data

Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
eError	Error Code	ERROR	-	-	Function block error ID
udiReceivedDataSize	Length of received data	UDINT	-	-	Length of received data

■ Program example

For details, see *"6.6. 10 Service Data Communication Examples for EtherNet/IP Master" on page 897*.

6.6. 7 Set_Attributes_All

This attribute sets all attribute values of a specific object instance.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
Set_Attributes_All	Set attributes of specific instance of all CIP object	FB		set_attributes_all(xExecute:= , xDone=> , xBusy=> , xError=> , itfEtherNetIPDevice:= , eClass:= , dwInstance:= , eError=> , pData:= , udiDataSize:=);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
itfEtherNetIPDevice	Remote slave name	IEtherNetIPService	-	-	Instance name of the remote slave
eClass	Class	CIPClass	-	-	Class of an object that a service is directed towards, which can be a CIP standard target object or a third-party vendor-defined object
dwInstance	Instance	DWORD	-	-	Instance under a class, which should be greater than or equal to 1
pData	Write buffer	POINTER TO BYTE	-	-	Pointer directing towards the attribute data, which sends the data to the destination slave
udiDataSize	Data length	UDINT	-	-	Length of data to write

Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
eError	Error Code	ERROR	-	-	Function block error ID
udiReceivedDataSize	Length of received data	UDINT	-	-	Length of received data

■ Program example

For details, see *"6.6. 10 Service Data Communication Examples for EtherNet/IP Master" on page 897*.

6.6. 8 Set_Attribute_Single

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
Set_Attribute_Single	Set attribute of specific instance of single CIP object	FB		set_attribute_single(xExecute:= , xDone=> , xBusy=> , xError=> , itfEtherNetIPDevice:= , eClass:= , dwInstance:= , eError=> , wAttribute:= , pData:= , udiDataSize:=);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
itfEtherNetIPDevice	Remote slave name	IEtherNetIPService	-	-	Instance name of the remote slave

Input Variable	Name	Data Type	Value Range	Initial Value	Description
eClass	Class	CIPClass	-	-	Class of an object that a service is directed towards, which can be a CIP standard target object or a third-party vendor-defined object
dwInstance	Instance	DWORD	-	-	Instance under a class, which should be greater than or equal to 1
wAttribute	Attribute	WORD	-	-	Attribute that a service is directed towards
pData	Write buffer	POINTER TO BYTE	-	-	Pointer directing towards the attribute data, which sends the data to the destination slave
udiDataSize	Data length	UDINT	-	-	Length of data to write

Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
eError	Error Code	ERROR	-	-	Function block error ID
udiReceivedDataSize	Length of received data	UDINT	-	-	Length of received data

■ Program example

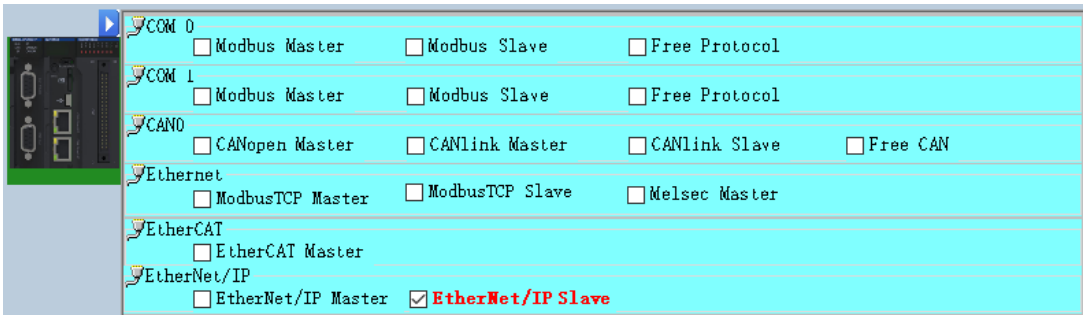
For details, see *"6.6. 10 Service Data Communication Examples for EtherNet/IP Master" on page 897*.

6.6. 9 Instances for Ethernet/IP Communication Reset Instructions

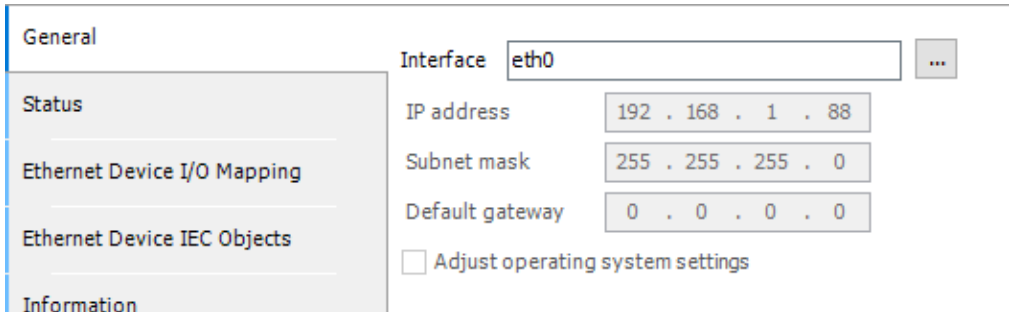
AC800 acts as the Ethernet/IP master, while AM600 acts as the Ethernet/IP slave.

Establish a connection.

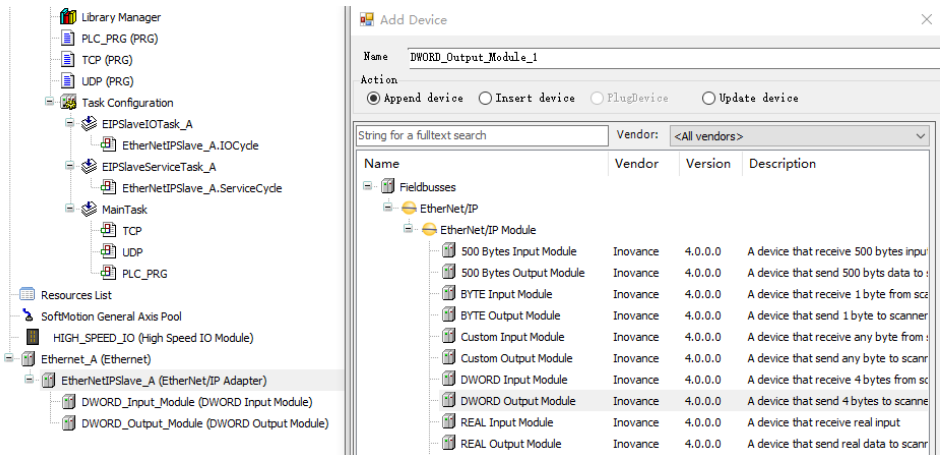
To configure the AM600 slave, select "EtherNet/IP Slave" in "AM600 NetWorkConfiguration".



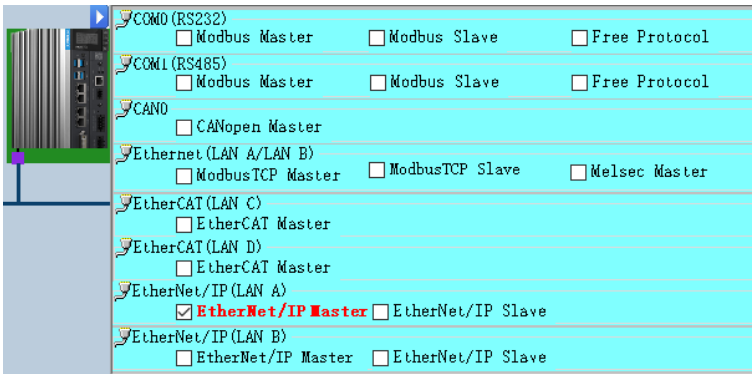
On the "General" page, set "Interface" to a network adapter (AM600 IP address: 10.44.52.188)



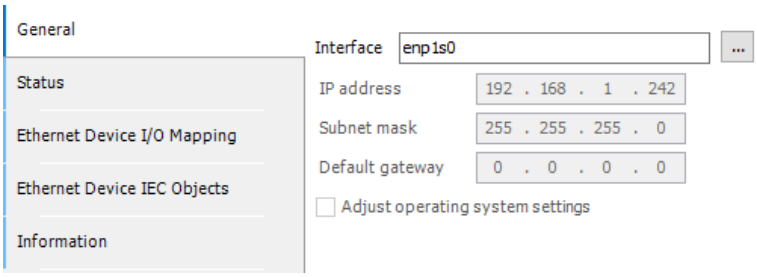
Select a slave. In the "Add Device" dialog box, add a device, for example, to receive four bytes from or send four bytes to the master.



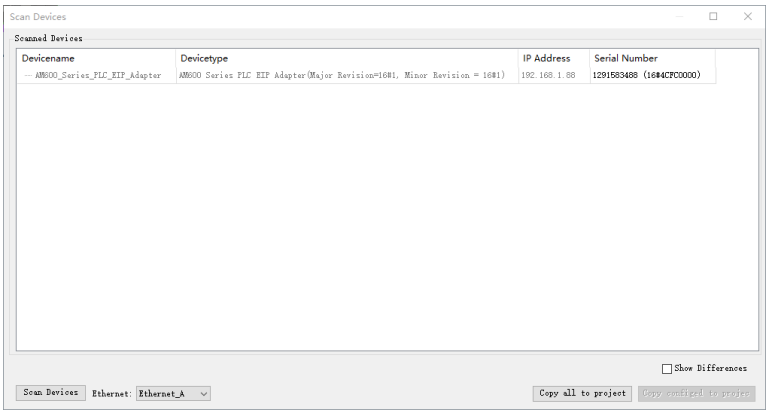
To configure the AC800 master, select "EtherNet/IP Master" in "NetWork Configuration" of AC800.



On the "General" page, set "Interface" to a network adapter (AC802 IP address: 10.44.52.15)

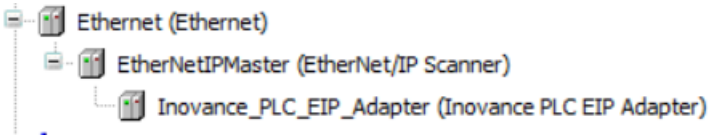


Use automatic scanning to add a slave device, or manually add a device using the toolbox. Scanning is recommended.



Add a function block without an instance, but change the function block name to match the actual configuration of the Ethernet/IP master-slave. The first line of the program lists the original name, and

the second line lists the name same as the actual configuration.



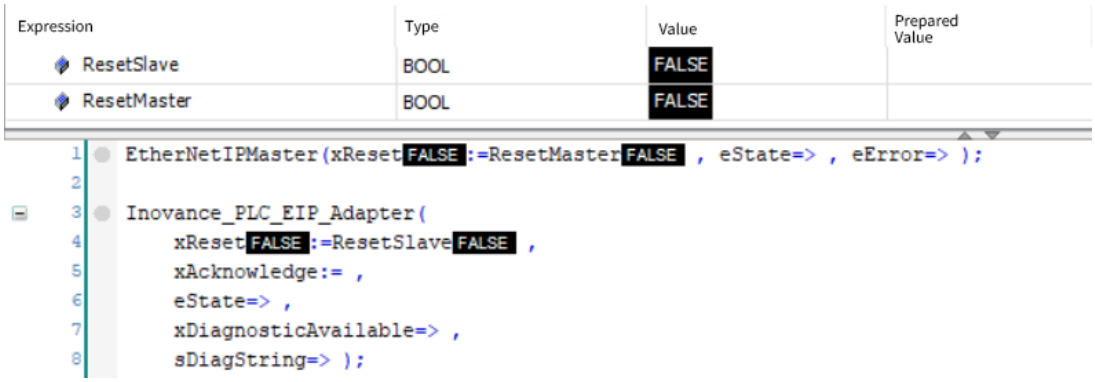

```
IoDrvEtherNetIP.IoDrvEtherNetIP(xReset:= , eState=> , eError=> );

EtherNetIPMaster(xReset:=ResetMaster , eState=>, eError=> );

IoDrvEtherNetIP.RemoteAdapter(
    xReset:= ,
    xAcknowledge:= ,
    eState=> ,
    xDiagnosticAvailable=> ,
    sDiagString=> );

Inovance_PLC_EIP_Adapter(
    xReset:=ResetSlave ,
    xAcknowledge:= ,
    eState=> ,
    xDiagnosticAvailable=> ,
    sDiagString=> );
```

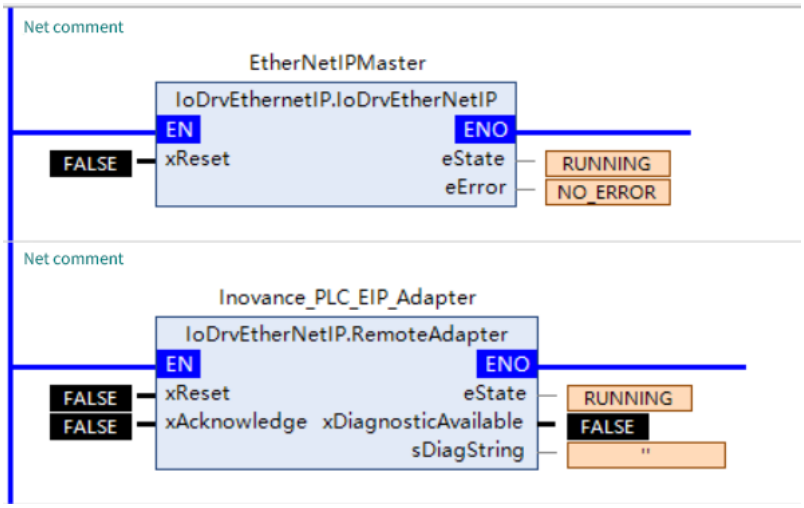
Define two BOOL variables for the master and slave respectively to enable reset of the master and slave.



LD

The process of establishing a connection is consistent with the ST language. Add a function block. Change the function block instance name to match the actual configuration of the Ethernet/IP master-slave.

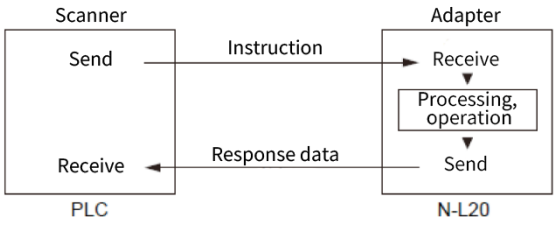
Use xReset to reset the master and slave, eState to check the master and slave states, and eERROR to check errors.



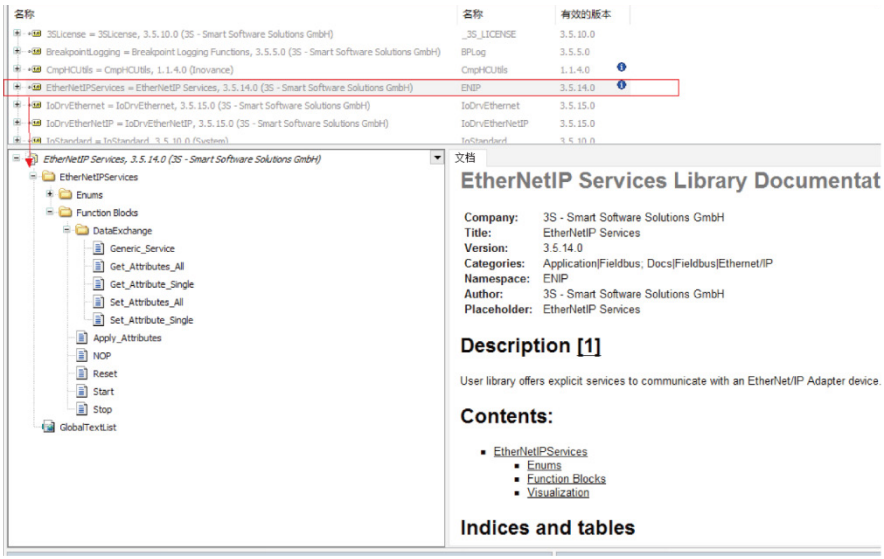
6.6. 10 Service Data Communication Examples for EtherNet/IP Master

Service data communication project example of the EtherNet/IP master

In service message communication, the timing is controlled by command/response, as shown in the following figure.



According to the naming format of service data communication sending command, it is usually necessary to specify "Service code", "Class ID", "Instance", "Attribute ID", and "Service data". This can be realized in the application program by using the EtherNet/IP service function block library EtherNetIPService, as shown in the library manager in the following figure.



■ CIP class definition: CIP class (ENUM)

InOut

Name	Initial Value
IdentityObject	16#1
MessageRouterObject	16#2
DeviceNetObject	16#3
AssemblyObject	16#4
ConnectionObject	16#5
ConnectionManagerObject	16#6
RegisterObject	16#7
DiscreteInputPointObject	16#8
DiscreteOutputPointObject	16#9
AnalogInputPointObject	16#A
AnalogOutputPointObject	16#B
PresenceSensingObject	16#E
ParameterObject	16#F
ParameterGroupObject	16#10
GroupObject	16#12
DiscreteInputGroupObject	16#1D
DiscreteOutputGroupObject	16#1E
DiscreteGroupObject	16#1F
AnalogInputGroupObject	16#20
AnalogOutputGroupObject	16#21
AnalogGroupObject	16#22
PositionSensorObject	16#23
PositionControllerSupervisorObject	16#24
PositionControllerObject	16#25
BlockSequencerObject	16#26
CommandBlockObject	16#27
MotorDataObject	16#28
ControlSupervisorObject	16#29
ACDCDriveObject	16#2A
AcknowledgeHandlerObject	16#2B
OverloadObject	16#2C
SoftstartObject	16#2D
SelectionObject	16#2E
S_DeviceSupervisorObject	16#30
S_AnalogSensorObject	16#31
S_AnalogActuatorObject	16#32
S_SingleStageControllerObject	16#33
S_GasCalibrationObject	16#34
TripPointObject	16#35
FileObject	16#37
S_PartialPressureObject	16#38
SafetySupervisorObject	16#39
SafetyValidatorObject	16#3A
SafetyDiscreteOutputPointObject	16#3B

Name	Initial Value
SafetyDiscreteOutputGroupObject	16#3C
SafetyDiscreteInputPointObject	16#3D
SafetyDiscreteInputGroupObject	16#3E
SafetyDualChannelOutputObject	16#3F
S_SensorCalibrationObject	16#40
EventLogObject	16#41
MotionDeviceAxisObject	16#42
TimeSyncObject	16#43
ModbusObject	16#44
OriginatorConnectionListObject	16#45
ModbusSerialLinkObject	16#46
DeviceLevelRingObject	16#47
QoSObject	16#48
SafetyAnalogInputPointObject	16#49
SafetyAnalogInputGroupObject	16#4A
SafetyDualChannelAnalogInputObject	16#4B
SERCOSIIILinkObject	16#4C
TargetConnectionListObject	16#4D
EnergyObject	16#4E
ElectricalEnergyObject	16#4F
Non_ElectricalEnergyObject	16#50
BaseSwitchObject	16#51
SNMPObject	16#52
PowerManagementObject	16#53
ControlNetObject	16#F0
ControlNetKeeperObject	16#F1
ControlNetSchedulingObject	16#F2
ConnectionConfigurationObject	16#F3
PortObject	16#F4
TCPIPInterfaceObject	16#F5
EthernetLinkObject	16#F6
CompoNetLink	16#F7
CompoNetRepeater	16#F8

To reference enum values, input the namespace ENIP first. For example, to use IdentityObject, input ENIP.IdentityObject.

■ CIP service definition: CIPCommonService (ENUM)

InOut

Name	Initial Value
None	16#0
GET_ATTRIBUTES_ALL	16#1
SET_ATTRIBUTES_ALL	16#2
Reset	16#5
START	16#6

Name	Initial Value
STOP	16#7
APPLY_ATTRIBUTES	16#D
GET_ATTRIBUTE_SINGLE	16#E
SET_ATTRIBUTE_SINGLE	16#10
NO_OPERATION	16#17

To reference enum values, input the namespace ENIP first. For example, to use GET_ATTRIBUTES_ALL, input ENIP.GET_ATTRIBUTES_ALL.

This function block library provides most of common services of the CIP communication protocol (CIP Common Services). The service IDs and names are listed in the following table.

ID	Name
00	Reserved for future use
01	Get_Attributes_All
02	Set_Attributes_All Request
03	Get_Attribute_List
04	Set_Attribute_List
05	Reset
06	Start
07	Stop
08	Create
09	Delete
0A	Multiple Service Packet
0B to 0C	Reserved for future use
0D	Apply_Attributes
0E	Get_Attribute_Single
0F	Reserved for future use
10	Set_Attribute_Single
11	Find_Next_Object_Instance
12 to 13	Reserved for future use
14	Error Response (used by DeviceNet only)

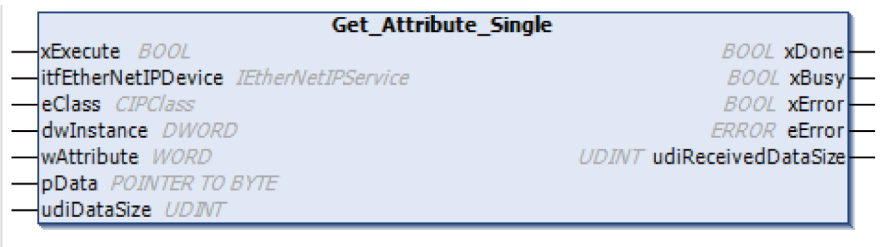
The Get_Attribute_Single and Generic_Service function blocks allow you to get attributes and service data from the user program.

■ Get_Attribute_Single

The KEYENCE NL-20 module guide provides special attribute IDs to get the current state of the reader, as shown in the table below.

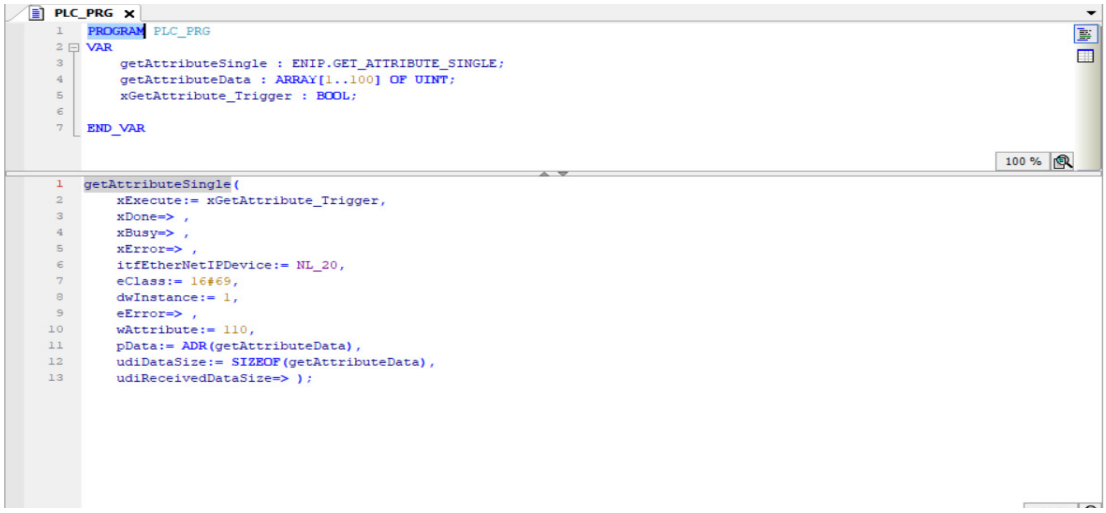
Instance ID	Attribute ID	Name	Parameter	
			Qty	Description
1 (0x01)	100 (0x64)	Read Status	UINT	bit0: Error bit1: Result Data Available bit2: Result Data Strobe bit3 to bit5: Reserved bit6: Buffer Overflow Error bit7: General Error bit8: BUSY bit9 to bit10: Reserved bit11: MODE BUSY bit12: ERR BUSY bit14 to bit15: Reserved
			UINT	Bit0: Read Complete
			UINT	Reserved
			UINT	Reserved
1 (0x01)	108 (0x6C)	IN/OUT Status	UINT	bit0: IN1 bit1: IN2 bit2 to bit3: Reserved bit4: OUT1 bit5: OUT2 bit6: OUT3 bit7: OUT4 bit8 to bit15: Reserved
	110 (0x6E)	Result Data Count	UINT	Result Data Ready Count
			UINT	General Error Code
	111 (0x6F)	General Error Code	UINT	General Error Code
	128 (0x80)	Result Data Ready Count	UINT	Result Data Ready Count
	129 (0x81)	Result Data Update Count	UINT	Result Data Update Count

The function block Get_Attribute_Single is used to realize corresponding parameter data in the above table, as shown below.



You can get the returned result count "Result Data Count" (attribute 0x6E) by instantiating the Get_Attribute_Single function block.

The example code is as follows: The AutoID Communication Unit Object = 16#69 is not an object in the EtherNet/IP standard, but an object developed by KEYENCE to make NL-20 more user-friendly.

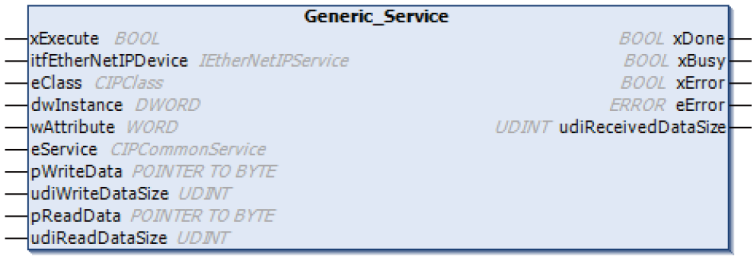


■ Generic_Service

In addition to the common services, there are also special types of services provided by some EtherNet/IP device vendors, such as the KEYENCE NL-20 module, which provides the following service codes for obtaining device-specific interaction data.

Instance ID	Service code	Service Data Data Type: Data	Name	Description
1 (0x01)	14 (0x0E)	-	Get_Attribute_Single	Get an attribute item.
	16 (0x10)	-	-	Get an attribute item.
	75 (0x4B)	UINT: Bank Number	Read Start	Start to read the data.
	76 (0x4C)	-	Read Stop	Stop reading the data.
	83 (0x53)	-	Error Clear	Clear the error.
	85 (0x55)	UINT: Result Data Size UINT: Offset	Get Result Data	Get the read data. Response data UINT: Size of the result data UINT: Size of the remaining data cached to NL-20 BYTE[]: Result data
	86 (0x56)	-	Sequence Reset	Clear the following information: Result Data Ready Count Result Data Update Count Main unit statistical information Buffering data Sequence bit
	90 (0x5A)	-	Read Status Clear	Clear the "Read Complete" and "Read Failed" notifications.

The above service data can be obtained through the function block Generic_Service (function block shown below). The interface data provided by the specified vendor can be obtained by providing the detailed Class ID, instance ID, and service code.



Read the data of the code scanner through the service code 85 (0x55), and the sample code is as follows.

```
VAR genericService : ENIP.Generic_Service;
VAR xGenericService_Trigger : BOOL;
VAR dUintWriteData : ARRAY[1..256] OF UINT;
VAR dUintReadData : ARRAY[1..256] OF UINT;
```

```
dUintWriteData[0] := SIZEOF(dUintReadData);
dUintWriteData[1] := 0;

genericService(
    xExecute:= xGenericService_Trigger,
    xDone=> ,
    xBusy=> ,
    xError=> ,
    itfEtherNetIPDevice:= NL_20,
    eClass:= 16#69,
    dwInstance:= 1,
    eError=> ,
    wAttribute:= 110,
    eService:= 16#55,                    //ENIP.CIPCommonService.SET_ATTRIBUTE_SING
    pWriteData:= ADR(dUintWriteData),
```

6. 7 Modbus RTU Communication Instructions

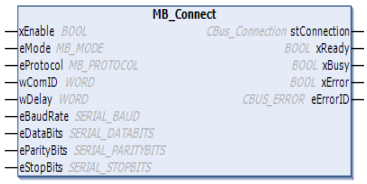
6.7. 1 Instruction List

Instruction Category	Name	FB/FC	Function
Modbus RTU communication instructions	ModbusRTUChannel	FB	Trigger channel of ModbusRTU master
	ModbusRTUSlave	FB	Modbus RTU slave disable
	sysHC_ModbusRtuDeviceDiagnose	FB	Get diagnostic information of ModbusRTU slave devices
	sysHC_ModbusRtuSlaveDiagnose	FB	Get diagnostic data of ModbusRTU master accessing slaves

6.7. 2 MB_Connect

This instruction configures the Modbus protocol.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
MB_Connect	Modbus protocol configuration	FB		MB_Connect(xEnable:= , eMode:= , eProtocol:= , wComID:= , wDelay:= , eBaudRate:= , eDataBits:= , eParityBits:= , eStopBits:= , stConnection=> , xReady=> , xBusy=> , xError=> , eErrorID=>);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
xEnable	Function block enable	BOOL	-	FALSE	Triggered by level
eMode	Modbus mode	ENUM	-	SLAVE	Communication mode: SLAVE: Modbus slave MASTER: Modbus master
eProtocol	Modbus protocol	ENUM	-	RTU	Communication protocol: RTU: Modbus RTU ASCII: Modbus ASCII
wComID	Slave IP address	WORD	-	0	COM port number
wDelay	Inter-frame interval (ms)	WORD	-	5	Modbus inter-frame interval
eBaudRate	Baud rate	ENUM	-	9600	Baud rate
eDateBits	Data bit	ENUM	-	8	Data bit
eParityBits	Parity bit	ENUM	-	0	Parity bit
eStopBits	Stop bit	ENUM	-	1	Stop bit

■ Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
stConnection	Connection handle	STRUCT	-	-	Connection handle
xReady	Instruction status	BOOL	-	-	Function block valid flag
xBusy	Instruction execution flag	BOOL	-	-	Function block execution flag

Output Variable	Name	Data Type	Value Range	Initial Value	Description
xError	Error flag	BOOL	-	-	Function block exception flag
eErrorID	Error code	ENUM	-	OK	Error code for the function block exception

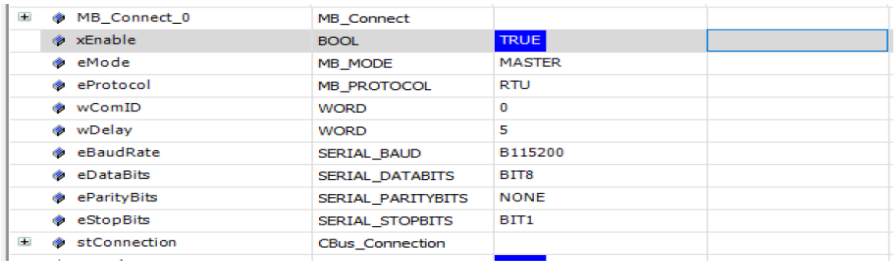
	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
xEnable	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
eMode	ENUM																			
eProtocol	ENUM																			
wComID	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
wDelay	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
eBaudRate	ENUM																			
eDateBits	ENUM																			
eParityBits	ENUM																			
eStopBits	ENUM																			
stConnection	STRUCT																			
xReady	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
xBusy	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
xError	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
eErrorID	ENUM																			

■ Function

The MB_Connect instruction sets the serial port and master parameters, such as the data transmission rate.

■ Program example

ST



```
MB_Connect_0(
    xEnable TRUE := xEnable TRUE,
    eMode MASTER := eMode MASTER,
    eProtocol RTU := eProtocol RTU,
    wComID 0 := wComID 0,
    wDelay 5 := wDelay 5,
    eBaudRate B115200 := eBaudRate B115200,
    eDataBits BIT8 := eDataBits BIT8,
    eParityBits NONE := eParityBits NONE,
    eStopBits BIT1 := eStopBits BIT1,
    stConnection=> stConnection,
    xReady TRUE => xReady TRUE,
    xBusy FALSE => xBusy FALSE,
    xError FALSE => xError FALSE,
    eErrorID NO_ERROR => eErrorID NO_ERROR);
```

■ Precautions

1) The connection handle of MB_Connect can only be used by either the MB_Master or MB_Slave instruction.

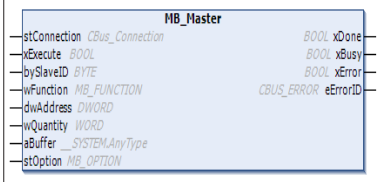
2) Ensure that multiple instructions with the same connection handle are used in the same user task.

3) Online parameter modifications do not take effect immediately. It is necessary to re-enable the instruction.

6.7. 3 MB_Master

This instruction enables Modbus master communication.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
MB_Master	Modbus master communication	FB		MB_Master(stConnection:= , xExecute:= , bySlaveID:= , wFunction:= , dwAddress:= , wQuantity:= , aBuffer:= , stOption:= , xDone=> , xBusy=> , xError=> , eErrorID=>);

■ Variables

In-out variables

In-Out Variable	Name	Data Type	Value Range	Initial Value	Description
stConnection	Connection handle	STRUCT	-	-	Connection handle

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
xExecute	Function block enable	BOOL	-	FALSE	Edge trigger
bySlaveID	Slave device ID	BYTE	0 to 247	1	ID of the slave requested to access
wFunction	Parameter	ENUM	-	0	Modbus parameter
dwAddress	Address of slave data	DWORD	1 to 65535	0	Head address of slave data requested to access
wQuantity	Number of accessed addresses	WORD	-	0	Maximum number of parameters: - 01/02: 2000 - 03/04: 125 - 15: 1968 - 16: 123
aBuffer	Read/Write buffering area	ANY	-	-	Modbus data buffer
stOption	Optional parameter	STRUCT	-	-	Operational parameter for the Modbus protocol

■ Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
xDone	Instruction completion flag	BOOL	-	FALSE	Function block execution completion flag
xBusy	Instruction execution flag	BOOL	-	FALSE	Function block being executed
xError	Error flag	BOOL	-	FALSE	Function block exception flag
eErrorID	Error code	ENUM	-	OK	Error code for the function block exception

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
stConnection	STRUCT																			
xExecute	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
bySlaveID	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
wFunction	ENUM																			
dwAddress	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
wQuantity	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
aBuffer	ANY																			
stOption	STRUCT																			
xDone	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
xBusy	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
xError	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
eErrorID	ENUM																			

■ Function

This instruction starts data transmission to send user-configured Modbus data frames by using the serial port bus.

■ Program example

ST

MB_Connect_0	MB_Connect		
xEnable	BOOL	TRUE	
eMode	MB_MODE	MASTER	
eProtocol	MB_PROTOCOL	RTU	
wComID	WORD	0	
wDelay	WORD	5	
eBaudRate	SERIAL_BAUD	B115200	
eDataBits	SERIAL_DATABITS	BIT8	
eParityBits	SERIAL_PARITYBITS	NONE	
eStopBits	SERIAL_STOPBITS	BIT1	
stConnection	CBus_Connection		
xReady	BOOL	TRUE	
xBusy	BOOL	FALSE	
xError	BOOL	FALSE	
eErrorID	CBUS_ERROR	NO_ERROR	
MB_Master_0	MB_Master		
xExecute	BOOL	TRUE	
bySlaveID	BYTE	1	
wFunction	MB_FUNCTION	READ_COILS	
dwAddress	DWORD	0	
wQuantity	WORD	50	
data	ARRAY [0..100] OF ...		
byRetryTimes	BYTE	1	
wTimeout	WORD	1000	
xErrorClear	BOOL	FALSE	
stOption	HC_CommBus.MB_O...		
xDone	BOOL	TRUE	
xBusy_0	BOOL	FALSE	
xError_0	BOOL	FALSE	
eErrorID_0	CBUS_ERROR	NO_ERROR	

```
//MB_Connect_0
MB_Connect_0(
    xEnable TRUE := xEnable TRUE,
    eMode MASTER := eMode MASTER,
    eProtocol RTU := eProtocol RTU,
    wComID 0 := wComID 0,
    wDelay 5 := wDelay 5,
    eBaudRate B115200 := eBaudRate B115200,
    eDataBits BIT8 := eDataBits BIT8,
    eParityBits NONE := eParityBits NONE,
    eStopBits BIT1 := eStopBits BIT1,
    stConnection=> stConnection,
    xReady TRUE => xReady TRUE,
    xBusy FALSE => xBusy FALSE,
    xError FALSE => xError FALSE,
    eErrorID NO_ERROR => eErrorID NO_ERROR);

//option
stOption.byRetryTimes 1 := byRetryTimes 1;
stOption.wTimeout 1000 := wTimeout 1000;
stOption.xErrorClear FALSE := xErrorClear FALSE;
```

```
MB_Master_0(
    stConnection:= stConnection,
    xExecute TRUE := xExecute TRUE ,
    bySlaveID 1 := bySlaveID 1 ,
    wFunction READ_COILS := wFunction READ_COILS ,
    dwAddress 0 := dwAddress 0 ,
    wQuantity 50 := wQuantity 50 ,
    aBuffer:= data,
    stOption:= stOption,
    xDone TRUE => xDone TRUE ,
    xBusy FALSE => xBusy_0 FALSE ,
    xError FALSE => xError_0 FALSE ,
    eErrorID NO_ERROR => eErrorID_0 NO_ERROR );RETURN
```

■ Precautions

1) It must be used with the MB_Connect instruction with eMode set to MASTER.

2) The connection handle configured by the MB_Connect instruction supports one or more MB_Master instances.

3) It is recommended that the MB_Connect and MB_Master instructions be processed within the same user task and the same program cycle. If they are in tasks with different priorities, operation anomalies may occur due to priority interruption.

4) When the MB_Master instruction is in the state specified by xBusy, the trigger action will be ignored and invalid in case of any repeated instructions until xDone or xError is set.

5) The MB_Master instruction cannot be executed in started, diagnostic, or safety mode, or in the case of time errors.

6) The bySlaveID slave address only supports standard station addresses 1 to 247. 0 is used for broadcasting.

7) The parameter and access address count are described as follows:

· Parameter: READ_COILS/READ_DISCRETE_INPUTS. The maximum number of supported access addresses is 2000.

· Parameter: WRITE_MULTIPLE_COILS. The maximum number of supported access addresses is 1968.

· Parameter: READ_HOLDING_REGISTERS/READ_INPUT_REGISTERS. The maximum number of supported access addresses is 125.

· Parameter: WRITE_MULTIPLE_REGISTERS. The maximum number of supported access addresses is 123.

8) Data types loaded by aBuffer are as follows:

· Supported types

One-dimensional array

Data types of the arrays/variables are as follows:

Access coil data: BYTE

Access register data: WORD, DWORD, INT, UINT, DINT, UDINT, and REAL

· Unsupported types

Multidimensional array

64-bit data types: LWORD, LINT, ULINT, and LREAL

Special types: STRING, WSTRING, TIME, TIME OF DAY(TOD), DATE, DATE OF TIME(DT), POINTER, and STRUCT

9) aBuffer cannot be directly set to I/Q/M direct variables. It needs to be mapped to the above supported variable/array data by using the AT method.

10) Online parameter modifications do not take effect immediately. It is necessary to re-enable the instruction.

6.7. 4 MB_Slave

This instruction enables Modbus slave communication.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
MB_Slave	Modbus slave communication	FB		MB_Slave(stConnection:= , xEnable:= , bySlaveID:= , aCoilsAddr:= , dwCoilsNum:= , aDiscretesInputAddr:= , dwDiscretesInputNum:= , aInputRegAddr:= , dwInputRegNum:= , aHoldingRegAddr:= , dwHoldingRegNum:= , xReady=> , xBusy=> , xError=> , eErrorID=>);

■ Variables

In-out variables

In-Out Variable	Name	Data Type	Value Range	Initial Value	Description
stConnection	Connection handle	STRUCT	-	-	Connection handle

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
xEnable	Function block enable	BOOL	-	FALSE	Triggered by level
bySlaveID	Slave device ID	BYTE	1 to 247	1	Modbus slave number
aCoilsAddr	Coil data area	ANY	-	0	Supported parameters: 0x01/0x05/0x0F
dwCoilsNum	Number of coils	DWORD	1 to 65535	0	Number of accessible coils
aDiscretes InputAddr	Discrete quantity data area	ANY	-	-	Supported parameters: 0x02
dwDiscretes InputNum	Input discrete quantity	DWORD	1 to 65536	0	Accessible discrete quantity
aInputRegAddr	Input register data area	ANY	-	-	Supported parameters: 0x04
dwInputRegNum	Number of input registers	DWORD	1 to 65536	0	Number of accessible input registers
aHoldingRegAddr	Holding register data area	ANY	-	-	Supported parameters: 0x03/0x06/0x10

Input Variable	Name	Data Type	Value Range	Initial Value	Description
dwHoldingRegNum	Number of holding registers	DWORD	1 to 65536	0	Number of accessible holding registers

■ Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
xReady	Instruction status	BOOL	-	FALSE	Function block valid flag
xBusy	Instruction execution flag	BOOL	-	FALSE	Function block execution flag
xError	Error flag	BOOL	-	FALSE	Function block exception flag
eErrorID	Error code	ENUM	-	OK	Error code for the function block exception

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
stConnection	STRUCT																			
xEnable	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
bySlaveID	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
aCoilsAddr	ANY																			
dwCoilsNum	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
aDiscretes InputAddr	ANY																			
dwDiscretes InputNum	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
aInputRegAddr																				
dwInput RegNum	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
aHolding RegAddr	ANY																			
dwHolding RegNum	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
xReady	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
xBusy	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
xError	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
eErrorID	ENUM																			

■ Function

This instruction allows any master to send Modbus requests and receive response frames.

■ Program example

ST

MB_Connect_0	MB_Connect	
xEnable	BOOL	TRUE
eMode	MB_MODE	MASTER
eProtocol	MB_PROTOCOL	RTU
wComID	WORD	0
wDelay	WORD	5
eBaudRate	SERIAL_BAUD	B115200
eDataBits	SERIAL_DATABITS	BIT8
eParityBits	SERIAL_PARITYBITS	NONE
eStopBits	SERIAL_STOPBITS	BIT1
stConnection	CBus_Connection	
xReady	BOOL	TRUE
xBusy	BOOL	FALSE
xError	BOOL	FALSE
eErrorID	CBUS_ERROR	NO_ERROR

MB_Slave_0	MB_Slave	
xEnable_1	BOOL	TRUE
SlaveID	BYTE	1
CoilsAddr	ARRAY [0..65536] ...	
CoilsNum	DWORD	65536
DisInAddr	ARRAY [0..65536] ...	
DisInNum	DWORD	65536
InRegAddr	ARRAY [0..65536] ...	
InRegNum	DWORD	65536
HoldRegAddr	ARRAY [0..65536] ...	
HoldRegNum	DWORD	65536
xReady_1	BOOL	TRUE
xBusy_1	BOOL	TRUE
xError_1	BOOL	FALSE
eErrorID_1	CBUS_ERROR	NO_ERROR

```
//MB_Connect_0
MB_Connect_0(
    xEnable TRUE := xEnable TRUE,
    eMode SLAVE := eMode SLAVE,
    eProtocol RTU := eProtocol RTU,
    wComID 0 := wComID 0,
    wDelay 5 := wDelay 5,
    eBaudRate B115200 := eBaudRate B115200,
    eDataBits BIT8 := eDataBits BIT8,
    eParityBits NONE := eParityBits NONE,
    eStopBits BIT1 := eStopBits BIT1,
    stConnection=> stConnection,
    xReady TRUE => xReady TRUE,
    xBusy FALSE => xBusy FALSE,
    xError FALSE => xError FALSE,
    eErrorID NO_ERROR => eErrorID NO_ERROR);

//MB_Slave_0
MB_Slave_0(
    stConnection:= stConnection,
    xEnable TRUE := xEnable_1 TRUE,
    bySlaveID 1 := SlaveID 1,
    aCoilsAddr:= CoilsAddr[CoilsAddrIdx 0] FALSE,
    dwCoilsNum 65536 := CoilsNum 65536,
    aDiscretesInputAddr:= DisInAddr[DisInAddrIdx 0] FALSE,
    dwDiscretesInputNum 65536 := DisInNum 65536,
    aInputRegAddr:= InRegAddr[InRegAddrIdx 0] 0,
    dwInputRegNum 65536 := InRegNum 65536,
    aHoldingRegAddr:= HoldRegAddr[HoldRegAddrIdx 0] 0,
    dwHoldingRegNum 65536 := HoldRegNum 65536,
    xReady TRUE => xReady_1 TRUE,
    xBusy TRUE => xBusy_1 TRUE,
    xError FALSE => xError_1 FALSE,
    eErrorID NO_ERROR => eErrorID_1 NO_ERROR);RETURN
```

■ Precautions

1) It must be used with the MB_Connect instruction with eMode set to SLAVE.

2) The connection handle configured by the MB_Connect instruction supports only one MB_Slave instance.

3) It is recommended that the MB_Connect and MB_Slave instructions be processed within the same user task and the same program cycle. If they are in tasks with different priorities, operation anomalies may occur due to priority interruption.

4) When the MB_Slave instruction is valid and accessible, xReady/xBusy will be set. If an exception occurs, xError will be set and eErrorID will display the error code.

5) The slave specified by MB_Slave is still accessible when the PLC dial is STOP.

6) The access areas are described as follows.

Data Area Variable	Area Name	Supported Parameter	Size of Accessible Area	Supported Data Type
aCoilsAddr	Coil data area	READ_COILS (01) WRITE_SINGLE_COIL (05) WRITE_MULTIPLE_COILS (15)	1 to 65536	BOOL, BYTE, and ARRAY
aDiscretesInputAddr	Discrete quantity data area	READ_DISCRETE_INPUTS (02)	1 to 65536	
aInputRegAddr	Input register data area	READ_INPUT_REGISTERS (04)	1 to 65536	WORD, DWORD, INT, UINT, DINT, UDINT, REAL, and ARRAY
aHoldingRegAddr	Holding register data area	READ_HOLDING_REGISTERS (03) WRITE_SIGNLE_REGISTER (06) WRITE_MULTIPLE_REGISTERS (16)	1 to 65536	

7) Unsupported types of access areas are as follows:

Multidimensional array

64-bit data types: LWORD, LINT, ULINT, and LREAL

Special types: STRING, WSTRING, TIME, TIME OF DAY(TOD), DATE, DATE OF TIME(DT), POINTER, and STRUCT

8) Access areas cannot be directly set to I/Q/M direct variables. They needs to be mapped to the above supported variable/array data by using the AT method.

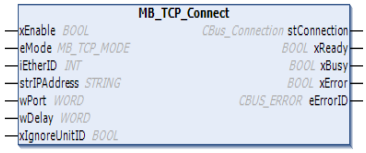
9) Access areas can be combined in free mode. Data in unnecessary access areas can be left empty, but not all the four access areas can be empty at the same time.

10) Online parameter modifications do not take effect immediately. It is necessary to re-enable the instruction.

6.7. 5 MB_TCP_Connect

This instruction configures Modbus TCP protocol parameters.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
MB_TCP_Connect	Modbus TCP protocol parameter configuration	FB		MB_TCP_Connect(xEnable:= , eMode:= , iEtherID:= , strIPAddress:= , wPort:= , wDelay:= , xIgnoreUnitID:= , stConnection=> , xReady=> , xBusy=> , xError=> , eErrorID=>);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
xEnable	Function block enable	BOOL	-	FALSE	FALSE: Network disconnected TRUE: Network connected
eMode	Modbus TCP mode	ENUM	-	SERVER	Communication mode: CLIENT: Modbus TCP client SERVER: Modbus TCP server
iEtherID	Network port ID	INT	-	-1	Network port configuration: -1: Automatic 0: Network port A It changes in incremental mode.
strIPAddress	IP address	STRING	-	``	IP address
wPort	Slave port	WORD	-	502	Slave port
wDelay	Inter-frame interval	WORD	-	1	Modbus TCP inter-frame interval
xIgnoreUnitID	Ignore uint_id	BOOL	-	FALSE	FALSE: unit_id filtering enabled TRUE: unit_id filtering disabled

■ Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
stConnection	Connection handle	STRUCT	-	-	Connection handle
xReady	Instruction status	BOOL	-	-	Function block valid flag
xBusy	Instruction execution flag	BOOL	-	-	Function block execution flag
xError	Error flag	BOOL	-	-	Function block exception flag
eErrorID	Error code	ENUM	-	OK	Error code for the function block exception

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
xEnable	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
eMode	ENUM																			
iEtherID	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-
strIPAddress	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√
wPort	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
wDelay	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
xIgnoreUnitID	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
stConnection	STRUCT																			

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
xReady	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
xBusy	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
xError	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
eErrorID	ENUM																			

■ Function

This instruction allows user to modify TCP parameters and inter-frame intervals during running.

■ Program example

ST

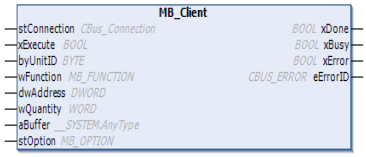
MB_TCP_Connect_0	MB_TCP_Connect	
xEnable	BOOL	TRUE
eMode	MB_TCP_MODE	SERVER
iEtherID	INT	-1
strIPAddress	STRING	''
wPort	WORD	503
wDelay	WORD	5
xIgnoreUnitID	BOOL	FALSE
stConnection	CBus_Connection	
xReady	BOOL	TRUE
xBusy	BOOL	FALSE
xError	BOOL	FALSE
eErrorID	CBUS_ERROR	NO_ERROR

```
MB_TCP_Connect_0(
    xEnable TRUE := xEnable TRUE,
    eMode SERVER := eMode SERVER,
    iEtherID 0 := iEtherID -1,
    strIPAddress '192.168.1. ▶ := strIPAddress '',
    wPort 511 := wPort 503,
    wDelay 0 := wDelay 5,
    xIgnoreUnitID FALSE := xIgnoreUnitID FALSE,
    stConnection=> stConnection,
    xReady TRUE => xReady TRUE,
    xBusy FALSE => xBusy FALSE,
    xError FALSE => xError FALSE,
    eErrorID NO_ERROR => eErrorID NO_ERROR);
```

6.7. 6 MB_Client

This instruction enables Modbus TCP client communication.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
MB_Client	Modbus TCP client communication	FB		MB_Client(stConnection:= , xExecute:= , byUnitID:= , wFunction:= , dwAddress:= , wQuantity:= , aBuffer:= , stOption:= , xDone=> , xBusy=> , xError=> , eErrorID=>);

■ Variables

In-out variables

In-Out Variable	Name	Data Type	Value Range	Initial Value	Description
stConnection	Connection handle	STRUCT	-	-	Connection handle

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
xExecute	Function block enable	BOOL	-	FALSE	Edge trigger
byUnitID	Slave device ID	BYTE	0 to 255	1	ID of the slave requested to access
wFunction	Parameter	ENUM	-	0	Modbus parameter
dwAddress	Address of slave data	DWORD	1 to 65535	0	Head address of slave data requested to access
wQuantity	Number of accessed addresses	WORD	1 to 65535	0	Maximum number of parameters: - 01/02: 2000 - 03/04: 125 - 15: 1968 - 16: 123
aBuffer	Read/Write buffering area	ANY	-	-	Modbus data buffer
stOption	Optional parameter	STRUCT	-	-	Operational parameter for the Modbus protocol

■ Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
xDone	Instruction completion flag	BOOL	-	FALSE	Function block execution completion flag
xBusy	Instruction execution flag	BOOL	-	FALSE	Function block being executed
xError	Error flag	BOOL	-	FALSE	Function block exception flag

Output Variable	Name	Data Type	Value Range	Initial Value	Description
eErrorID	Error code	ENUM	-	OK	Error code for the function block exception

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
stConnection	STRUCT																			
xExecute	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
bySlaveID	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
wFunction	ENUM																			
dwAddress	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
wQuantity	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
aBuffer	ANY																			
stOption	STRUCT																			
xDone	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
xBusy	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
xError	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
eErrorID	ENUM																			

■ Function

This instruction starts data transmission to send Modbus data frames by using TCP.

■ Program example

ST

MB_TCP_Connect_0	MB_TCP_Connect	
xEnable	BOOL	TRUE
eMode	MB_TCP_MODE	CLIENT
iEtherID	INT	-1
strIPAddress	STRING	'10.45.121.35'
wPort	WORD	502
wDelay	WORD	5
xIgnoreUnitID	BOOL	FALSE
stConnection	CBus_Connection	
xReady	BOOL	TRUE
xBusy	BOOL	FALSE
xError	BOOL	FALSE
eErrorID	CBUS_ERROR	NO_ERROR

MB_Client_0	MB_Client	
xExecute	BOOL	TRUE
bySlaveID	BYTE	1
wFunction	MB_FUNCTION	WRITE_MULTIPLE_COILS
dwAddress	DWORD	0
wQuantity	WORD	123
data	ARRAY [0..125] OF ...	
byRetryTimes	BYTE	1
wTimeout	WORD	1000
xErrorClear	BOOL	FALSE
stOption	HC_CommBus.MB_O...	
xDone	BOOL	TRUE
xBusy_0	BOOL	FALSE
xError_0	BOOL	FALSE
eErrorID_0	CBUS_ERROR	NO_ERROR

```
//MB_TCP_Connect_0
MB_TCP_Connect_0(
    xEnable TRUE := xEnable TRUE,
    eMode CLIENT := eMode CLIENT,
    iEtherID -1 := iEtherID -1,
    strIPAddress '10.45.121. ▶ := strIPAddress '10.45.121. ▶,
    wPort 502 := wPort 502,
    wDelay 5 := wDelay 5,
    xIgnoreUnitID FALSE := xIgnoreUnitID FALSE,
    stConnection=> stConnection,
    xReady TRUE => xReady TRUE,
    xBusy FALSE => xBusy FALSE,
    xError FALSE => xError FALSE,
    eErrorID NO_ERROR => eErrorID NO_ERROR );

//option
stOption.byRetryTimes 1 := byRetryTimes 1;
stOption.wTimeout 1000 := wTimeout 1000;
stOption.xErrorClear FALSE := xErrorClear FALSE;

MB_Client_0(
    stConnection:= stConnection,
    xExecute TRUE := xExecute TRUE,
    byUnitID 1 := bySlaveID 1,
    wFunction WRITE_MULT ▶ := wFunction WRITE_MULT ▶,
    dwAddress 0 := dwAddress 0,
    wQuantity 123 := wQuantity 123,
    aBuffer:= data,
    stOption:= stOption,
    xDone TRUE => xDone TRUE,
    xBusy FALSE => xBusy_0 FALSE,
    xError FALSE => xError_0 FALSE,
    eErrorID NO_ERROR => eErrorID_0 NO_ERROR );RETURN
```

■ Precautions

1) It must be used with the MB_TCP_Connect instruction with eMode set to CLIENT.

2) The connection handle configured by the MB_TCP_Connect instruction supports one or more MB_Client instances.

3) It is recommended that the MB_TCP_Connect and MB_Client instructions be processed within the same user task and the same program cycle. If they are in tasks with different priorities, operation anomalies may occur due to priority interruption.

4) When the MB_Client instruction is in the state specified by xBusy, the trigger action will be ignored and invalid in case of any repeated instructions until xDone or xError is set.

5) The MB_Client instruction cannot be executed in started, diagnostic, or safety mode, or in the case of time errors.

6) The parameter and access address count are described as follows:

Parameter: READ_COILS/READ_DISCRETE_INPUTS. The maximum number of supported access addresses is 2000.

Parameter: WRITE_MULTIPLE_COILS. The maximum number of supported access addresses is 1968.

Parameter: READ_HOLDING_REGISTERS/READ_INPUT_REGISTERS. The maximum number of supported access addresses is 125.

Parameter: WRITE_MULTIPLE_REGISTERS. The maximum number of supported access addresses is 123.

7) Data types loaded by aBuffer are as follows:

· Supported types

One-dimensional array

Data types of the arrays/variables are as follows:

Coil access: BYTE

Register access: WORD, DWORD, INT, UINT, DINT, UDINT, and REAL

· Unsupported types

Multidimensional array

64-bit data types: LWORD, LINT, ULINT, and LREAL

Special types: STRING, WSTRING, TIME, TIME OF DAY(TOD), DATE, DATE OF TIME(DT), POINTER, and STRUCT

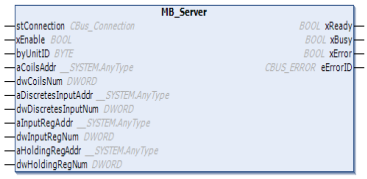
8) aBuffer cannot be directly set to I/Q/M direct variables. It needs to be mapped to the above supported variable/array data by using the AT method.

9) Online parameter modifications do not take effect immediately. It is necessary to re-enable the instruction.

6.7. 7 MB_Server

This instruction enables Modbus TCP server communication.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
MB_Server	Modbus TCP server communication	FB		MB_Server(stConnection:= , xEnable:= , byUnitID:= , aCoilsAddr:= , dwCoilsNum:= , aDiscretesInputAddr:= , dwDiscretesInputNum:= , aInputRegAddr:= , dwInputRegNum:= , aHoldingRegAddr:= , dwHoldingRegNum:= , xReady=> , xBusy=> , xError=> , eErrorID=>);

■ Variables

In-out variables

In-Out Variable	Name	Data Type	Value Range	Initial Value	Description
stConnection	Connection handle	STRUCT	-	-	Connection handle

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
xEnable	Function block enable	BOOL	-	FALSE	Triggered by level
byUnitID	Slave device ID	BYTE	0 to 255	1	Modbus slave number
aCoilsAddr	Coil data area	ANY	-	-	Supported parameters: 0x01/0x05/0x0F
dwCoilsNum	Number of coils	DWORD	1 to 65536	0	Number of accessible coils

Input Variable	Name	Data Type	Value Range	Initial Value	Description
aDiscretes InputAddr	Discrete quantity data area	ANY	-	-	Supported parameters: 0x02
dwDiscretes InputNum	Input discrete quantity	DWORD	1 to 65536	0	Accessible discrete quantity
aInputRegAddr	Input register data area	ANY	-	-	Supported parameters: 0x04
dwInput RegNum	Number of input registers	DWORD	1 to 65536	0	Number of accessible input registers
aHolding RegAddr	Holding register data area	ANY	-	-	Supported parameters: 0x03/0x06/0x10
dwHolding RegNum	Number of holding registers	DWORD	1 to 65536	0	Number of accessible holding registers

■ Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
xReady	Instruction status	BOOL	-	FALSE	Function block valid flag
xBusy	Instruction execution flag	BOOL	-	FALSE	Function block execution flag
xError	Error flag	BOOL	-	FALSE	Function block exception flag
eErrorID	Error code	ENUM	-	OK	Error code for the function block exception

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
stConnection	STRUCT																			
xEnable	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
bySlaveID	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
aCoilsAddr	ANY																			
dwCoilsNum	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
aDiscretes InputAddr	ANY																			
dwDiscretes InputNum	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
aInputRegAddr	ANY																			
dwInput RegNum	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
aHolding RegAddr	ANY																			
dwHolding RegNum	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
xReady	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
xBusy	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
xError	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
eErrorID	ENUM																			

■ Function

This instruction starts data transmission to send Modbus data frames by using TCP.

■ Program example

ST

MB_TCP_Connect_0	MB_TCP_Connect	
xEnable	BOOL	TRUE
eMode	MB_TCP_MODE	SERVER
iEtherID	INT	-1
strIPAddress	STRING	''
wPort	WORD	503
wDelay	WORD	5
xIgnoreUnitID	BOOL	FALSE
stConnection	CBus_Connection	
xReady	BOOL	TRUE
xBusy	BOOL	FALSE
xError	BOOL	FALSE
eErrorID	CBUS_ERROR	NO_ERROR

MB_Server_0	MB_Server	
xEnable_1	BOOL	TRUE
SlaveID	BYTE	1
CoilsAddr	ARRAY [0..65536] ...	
CoilsNum	DWORD	65536
DisInAddr	ARRAY [0..65536] ...	
DisInNum	DWORD	65536
InRegAddr	ARRAY [0..65536] ...	
InRegNum	DWORD	65536
HoldRegAddr	ARRAY [0..65536] ...	
HoldRegNum	DWORD	65536
xReady_1	BOOL	TRUE
xBusy_1	BOOL	TRUE
xError_1	BOOL	FALSE
eErrorID_1	CBUS_ERROR	NO_ERROR

```
//MB_TCP_Connect_0
MB_TCP_Connect_0(
    xEnable TRUE := xEnable TRUE,
    eMode SERVER := eMode SERVER,
    iEtherID -1 := iEtherID -1,
    strIPAddress '' := strIPAddress '',
    wPort 503 := wPort 503,
    wDelay 5 := wDelay 5,
    xIgnoreUnitID FALSE := xIgnoreUnitID FALSE,
    stConnection=> stConnection,
    xReady TRUE => xReady TRUE,
    xBusy FALSE => xBusy FALSE,
    xError FALSE => xError FALSE,
    eErrorID NO_ERROR => eErrorID NO_ERROR);

//MB_Server_0
MB_Server_0(
    stConnection:= stConnection,
    xEnable TRUE := xEnable_1 TRUE,
    byUnitID 1 := SlaveID 1,
    aCoilsAddr:= CoilsAddr[CoilsAddrIdx 0] TRUE,
    dwCoilsNum 65536 := CoilsNum 65536,
    aDiscretesInputAddr:= DisInAddr[DisInAddrIdx 0] FALSE,
    dwDiscretesInputNum 65536 := DisInNum 65536,
    aInputRegAddr:= InRegAddr[InRegAddrIdx 0] 0,
    dwInputRegNum 65536 := InRegNum 65536,
    aHoldingRegAddr:= HoldRegAddr[HoldRegAddrIdx 0] 55,
    dwHoldingRegNum 65536 := HoldRegNum 65536,
    xReady TRUE => xReady_1 TRUE,
    xBusy TRUE => xBusy_1 TRUE,
    xError FALSE => xError_1 FALSE,
    eErrorID NO_ERROR => eErrorID_1 NO_ERROR);RETURN
```

■ Precautions

1) It must be used with the MB_TCP_Connect instruction with eMode set to SERVER.

2) The connection handle configured by the MB_TCP_Connect instruction supports only one MB_Server instance.

3) It is recommended that the MB_TCP_Connect and MB_Server instructions be processed within the same user task and the same program cycle. If they are in tasks with different priorities, operation anomalies may occur due to priority interruption.

4) When the MB_Server instruction is valid and accessible, xReady/xBusy will be set. If an exception occurs, xError will be set and eErrorID will display the error code.

5) The slave specified by MB_Server is still accessible when the PLC dial is STOP.

6) If StrIPAddress of MB_TCP_Connect is empty, connection to any remote Modbus TCP client is allowed and the set data area is accessed.

If StrIPAddress is set to a valid IP address, the IP filter function is enabled, and only the remote Modbus TCP client with this IP address is allowed to access.

7) The access areas are described as follows.

Data Area Variable	Area Name	Supported Parameter	Size of Accessible Area	Supported Data Type
aCoilsAddr	Coil data area	READ_COILS (01) WRITE_SINGLE_COIL (05) WRITE_MULTIPLE_COILS (15)	1 to 65536	BOOL, BYTE, and ARRAY
aDiscretesInputAddr	Discrete quantity data area	READ_DISCRETE_INPUTS (02)	1 to 65536	

Data Area Variable	Area Name	Supported Parameter	Size of Accessible Area	Supported Data Type
aInputRegAddr	Input register data area	READ_INPUT_REGISTERS (04)	1 to 65536	WORD, DWORD, INT, UINT, DINT, UDINT, REAL, and ARRAY
aHoldingRegAddr	Holding register data area	READ_HOLDING_REGISTERS (03) WRITE_SIGNLE_REGISTER (06) WRITE_MULTIPLE_REGISTERS (16)	1 to 65536	

8) Unsupported types of access areas are as follows:

Multidimensional array

64-bit data types: LWORD, LINT, ULINT, and LREAL

Special types: STRING, WSTRING, TIME, TIME OF DAY(TOD), DATE, DATE OF TIME(DT), POINTER, and STRUCT

9) Access areas cannot be directly set to I/Q/M direct variables. They needs to be mapped to the above supported variable/array data by using the AT method.

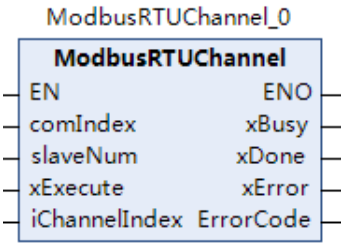
10) Access areas can be combined in free mode. Unnecessary access areas can be left empty, but not all the four access areas can be empty at the same time.

11) Online parameter modifications do not take effect immediately. It is necessary to re-enable the instruction.

6.7. 8 ModbusRTUChannel

This instruction implements triggering by channel for the Modbus RTU master.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
ModbusRTUChannel	Trigger channel of ModbusRTU master	FB		ModbusRTUChannel_0(comIndex :=, slaveNum :=, xExecute :=, iChannelIndex :=, xBusy =>, xDone =>, xError =>, ErrorCode =>,);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
comIndex	COM port number	INT	Depends on data types	-	COM port number 0: COM0
slaveNum	Slave number	INT	Depends on data types	-	Slave number 31: Slave 31
xExecute	Enable	BOOL	Depends on data types	-	(Triggered by edges) Execution of the function block switch

Input Variable	Name	Data Type	Value Range	Initial Value	Description
iChannelIndex	Channel number	INT	Depends on data types	-	Number of the communication channel 2: Channel 2

Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
xBusy	Function block execution flag	BOOL	Depends on data types	-	Function block execution flag
xDone	Function block completed state	BOOL	Depends on data types	-	Function block completed state
xError	Function block error flag	BOOL	Depends on data types	-	Function block error flag
ErrorCode	Error information of function block	Modbus_ERROR	Depends on data types	-	Error information of function block

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
comIndex	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-
slaveNum	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-
xExecute	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
iChannelIndex	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-
xBusy	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
xDone	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
xError	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
ErrorCode	See "Function" for the enumerators for the enumerated type Modbus_ERROR.																			

■ Function

This instruction supports triggering by channel for the Modbus RTU master.

Change the channel mode to cyclic execution before use. When the cycle time (ms) is set to 65535, the trigger mode is enabled and then the function block triggers the cyclic execution.

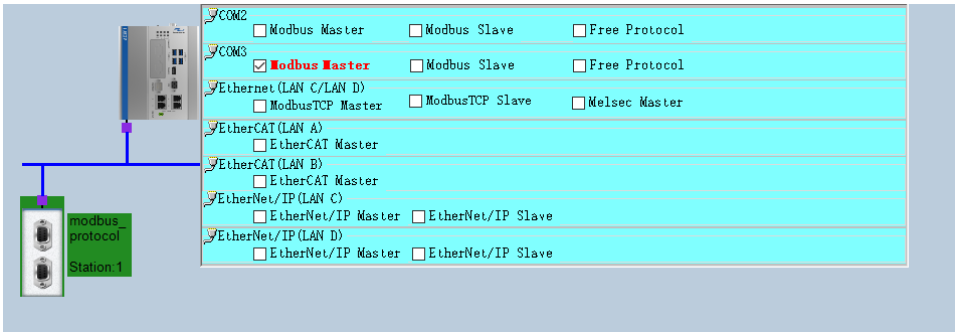
When xExecute is set to FALSE, trigger enable is cleared.

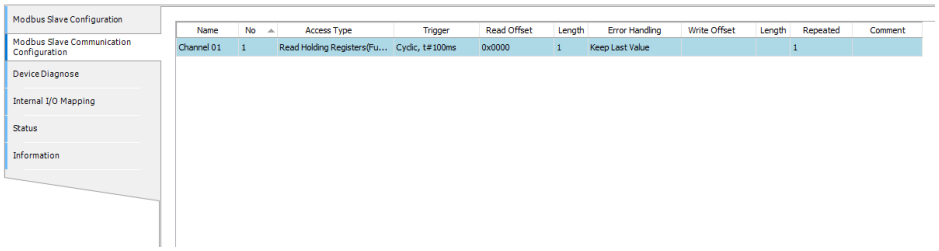
The data type of ErrorCode is enumerated type Modbus_ERROR. The meanings of the enumerators are as follows.

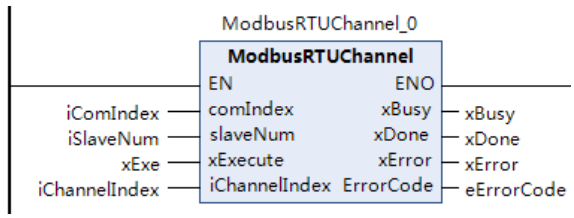
Error Code	Error	Description
0	Modbus_NO_ERROR	Modbus communication is normal.
1	Modbus_NO_COM_INDEX	The COM port configured based on Modbus RTU does not exist or cannot be accessed.
2	Modbus_NO_COM_SLAVE	The configuration of the Modbus RTU slave number is incorrect.
3	Modbus_NO_COM_CHANNEL	The configuration of the Modbus RTU channel number is incorrect.
4	Modbus_NO_TCP_SLAVE	The configuration of the Modbus TCP slave IP address is incorrect.
5	Modbus_NO_TCP_CHANNEL	The configuration of the Modbus TCP communication channel is incorrect, or the channel does not exist.

Error Code	Error	Description
6	Modbus_OTHER_FAILED	Other errors

■ Program example

<table>
<tr><th></th><th>LD</th><th>ST</th></tr>
<tr><td>Defined variable</td><td colspan="2"> 

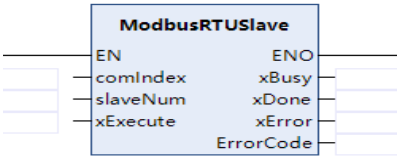


<pre>
VAR
    ModbusRTUChannel_0: ModbusRTUChannel;
    iComIndex : INT := 1;
    iSlaveNum : INT := 1;
    xExe : BOOL := TRUE;
    iChannelIndex : INT := 1;
    xBusy : BOOL;
    xDone : BOOL;
    xError : BOOL;
    eErrorCode : MODBUS_ERROR;
END_VAR
</pre></td></tr>
<tr><td>Program</td><td> 
</td><td>
<pre>
ModbusRTUChannel_0(
    comIndex:= iComIndex,
    slaveNum:= iSlaveNum,
    xExecute:= xExe,
    iChannelIndex:= iChannelIndex,
    xBusy=> xBusy,
    xDone=> xDone,
    xError=> xError,
    ErrorCode=> eErrorCode);
</pre></td></tr>
</table>

6.7. 9 ModbusRTUSlave

This instruction disables the Modbus RTU slave.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
ModbusRTUSlave	Modbus RTU slave disable	FB		ModbusRTUslave_0(comIndex:= , slaveNum:= , xExecute:= , xBusy=> , xDone=> , xError=> , ErrorCode=>);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
comIndex	COM port number	INT	Depends on data types	0	COM port number 0: COM0 1: COM1
slaveNum	Slave address	INT	Depends on data types	0	Slave address. 1 to 255 slave addresses are supported.
xExecute	Function block enable	BOOL	[FALSE, TRUE]	FALSE	Function block enable TRUE: The function block is executed and the slave is disabled FALSE: The function block is stopped and the slave is disabled

Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
xBusy	Execution flag	BOOL	[FALSE, TRUE]	-	Function block being executed
xDone	Completion flag	BOOL	[FALSE, TRUE]	-	Output valid flag of the function block
xError	Error flag	BOOL	[FALSE, TRUE]	-	TRUE: An internal error occurs in the function block
ErrorID	Error ID	Modbus_ERROR	Depends on data types	-	Error Code

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
comIndex	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-
slaveNum	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-
xExecute	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Done	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Busy	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Error	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

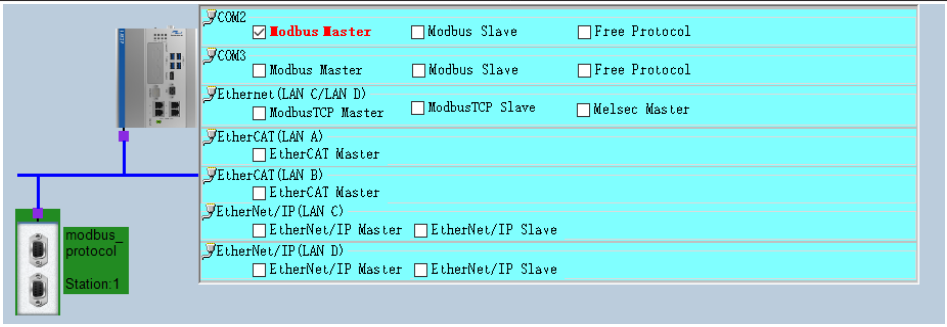
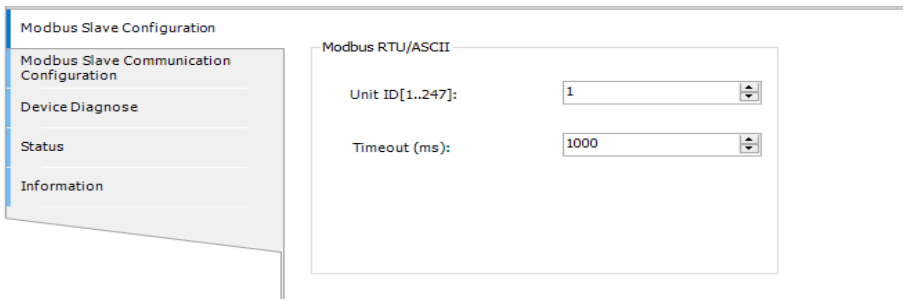
	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
ErrorID	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-

■ Function

This function block disables the slave in Modbus RTU master mode.

■ Program example

	LD	ST

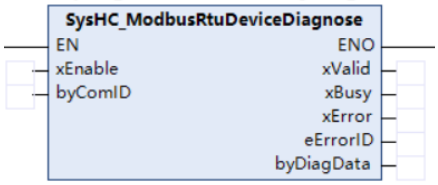
Defined variable	
	 
	VAR
	ModbusRTUSlave_0: ModbusRTUSlave;
	com: INT:=0;
	SlaveNum: INT:=1;
	en: BOOL:=TRUE;
	buy: BOOL;
	done: BOOL;
	error: BOOL;
	errorcode:MODBUS_ERROR;
	END_VAR	

Program	ModbusRTUSlave_0 ModbusRTUSlave EN → ENO com → comIndex SlaveNum → slaveNum en → xExecute xBusy → buy xDone → done xError → error ErrorCode → errorcode	ModbusRTUSlave_0(comIndex:= com, slaveNum:= SlaveNum, xExecute:= en, xBusy=> buy, xDone=> done, xError=> error, ErrorCode=> errorcode);

6.7. 10 sysHC_ModbusRtuDeviceDiagnose

This instruction gets diagnosis information of the Modbus RTU slave device by using the external library.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
sysHC_ ModbusRtuDeviceDiagnose	Get diagnostic information of ModbusRTU slave devices	FB		SysHC_ ModbusRtuDeviceDiagnose(xEnable:= , byComID:= , xValid=> , xBusy=> , xError=> , eErrorID=> , byDiagData=>);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
xEnable	Function block enable	BOOL	-	-	Input high level to enable the function block
byComID	Serial port number	BYTE	0 to 1	-	Serial port number used by Modbus

Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
xValid	Output active	BOOL	-	-	Output active
xBusy	Busy flag	BOOL	-	-	Executing the instruction
xError	Error flag	BOOL	-	-	Error flag
eErrorID	Error code	SYS_HC_ERROR	-	-	Error code
byDiagData	Diagnosis code	BYTE	-	-	Diagnosis code

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
xEnable	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
byComID	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
xValid	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
xBusy	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
xError	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
eErrorID	SYS_HC_ERROR																			
byDiagData	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

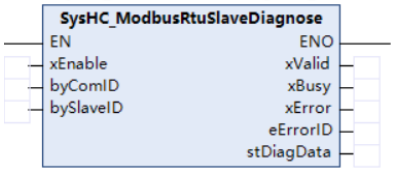
■ Program example

For details, see "6.7. 12 Examples of Modbus Communication Instructions" on page 930.

6.7. 11 sysHC_ModbusRtuSlaveDiagnose

This instruction gets diagnosis information of slave faults accessed from the Modbus RTU master by using the external library.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
sysHC_ ModbusRtuSlaveDiagnose	Get diagnostic data of ModbusRTU master accessing slaves	FB		SysHC_ ModbusRtuSlaveDiagnose(xEnable:= , byComID:= , bySlaveID:= , xValid=> , xBusy=> , xError=> , eErrorID=> , stDiagData=>);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
xEnable	Function block enable	BOOL	-	-	Input high level to enable the function block
byComID	Serial port number	BYTE	0 to 1	-	Serial port number used by Modbus
bySlaveID	Slave number	BYTE	-	-	Modbus slave number

Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
xValid	Valid flag	BOOL	-	-	Output active
xBusy	Busy flag	BOOL	-	-	Executing the instruction
xError	Error flag	BOOL	-	-	Error flag
eErrorID	Error code	SYS_HC_ERROR	-	-	Error code
stDiagData	Diagnostic data	Modbus_SLAVE_DIAGNOSE	-	-	Slave diagnostic data

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
xEnable	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
byComID	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
bySlaveID	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
xValid	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
xBusy	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
xError	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
eErrorID	SYS_HC_ERROR																			
stDiagData	Modbus_SLAVE_DIAGNOSE																			

■ Program example

For details, see *"6.7. 12 Examples of Modbus Communication Instructions" on page 930*.

6.7. 12 Examples of Modbus Communication Instructions

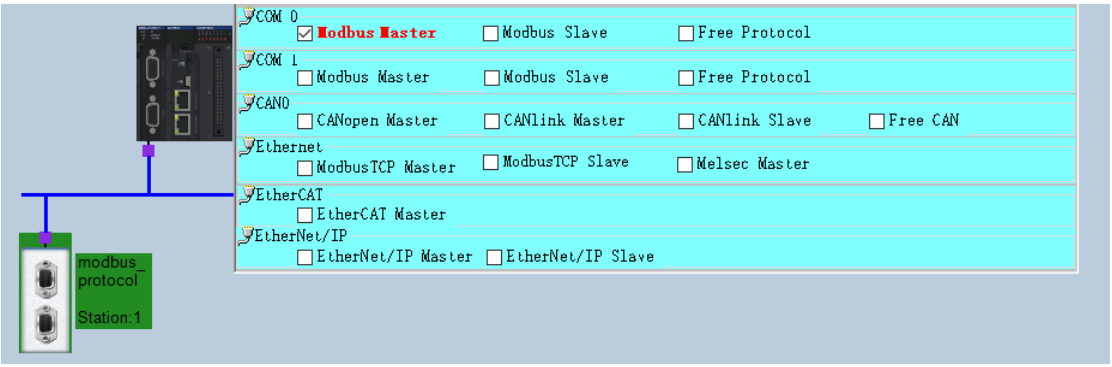
1. Modbus RTU

AM600 COM0 port acts as the Modbus RTU master, while H5U acts as the Modbus RTU slave.

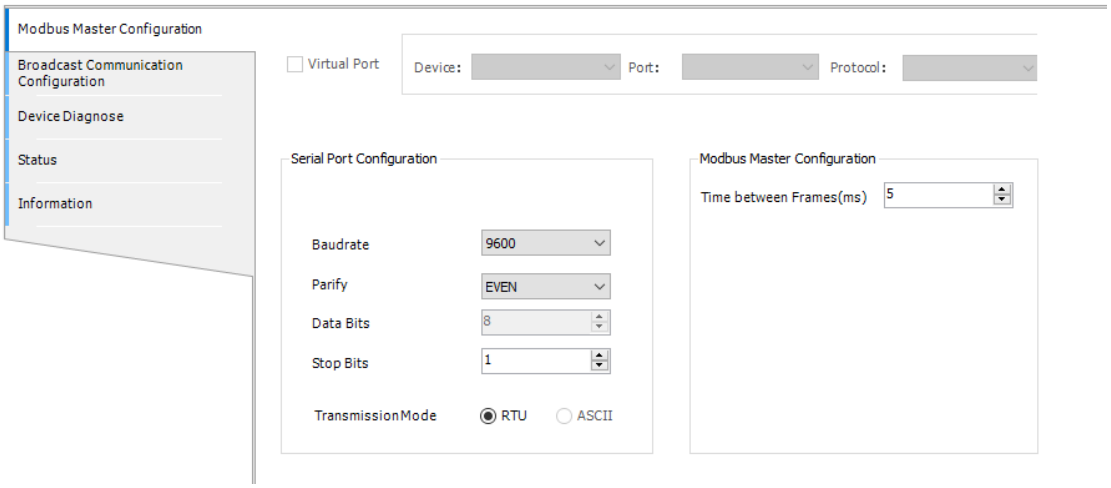
ST

Normal state

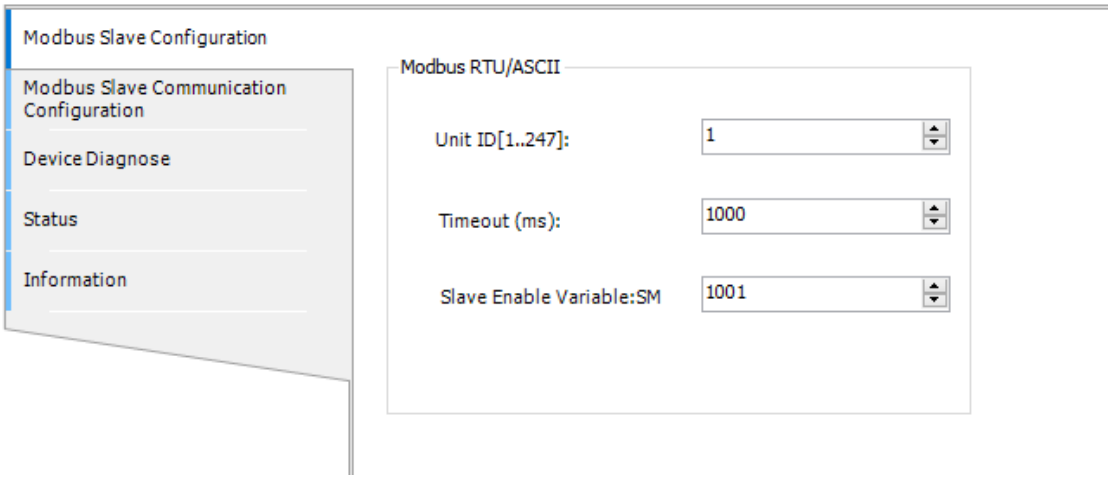
1) In "NetWork Configuration", select "Modbus Master" for "COM0" and add a slave, as shown in the following figure.



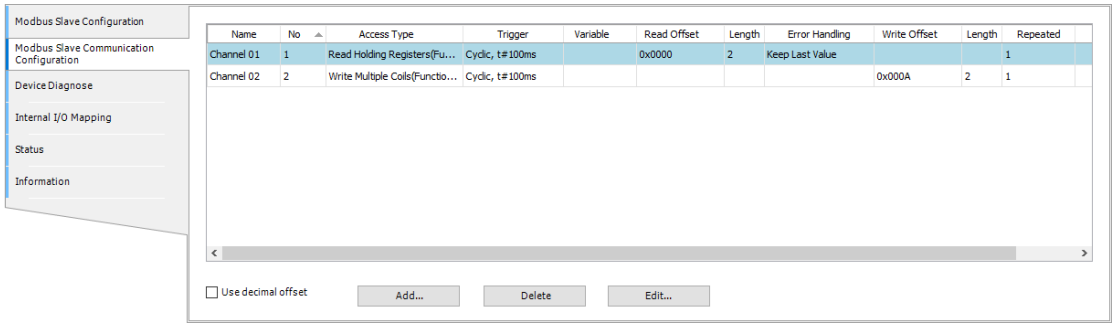
2) On the "Modbus Master Configuration" page, configure the serial port communication parameters, including "Baudrate", "Parify", "Data Bits", and "Stop Bits".



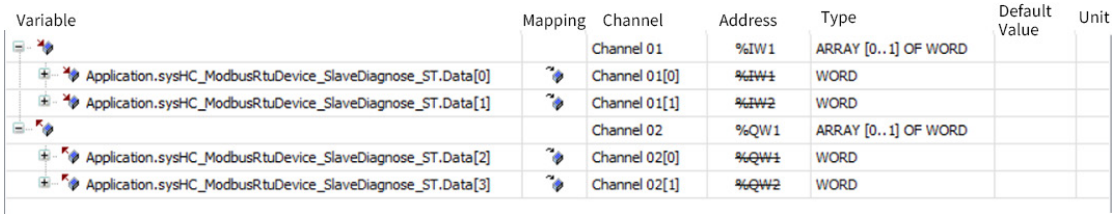
3) On the "Modbus Slave Configuration" page, set "Unit ID[1..247]" to 1.



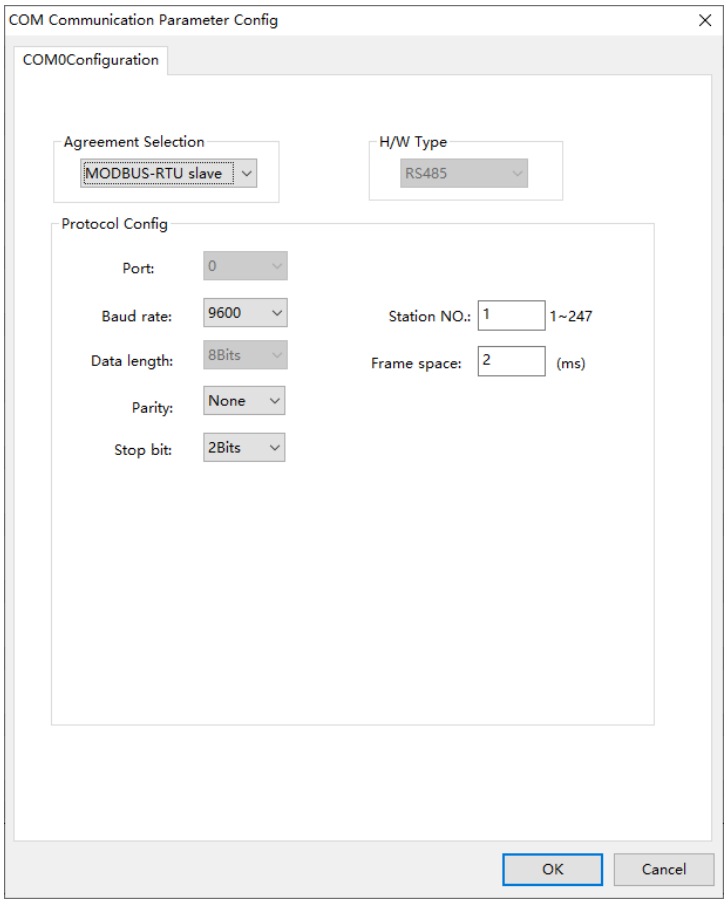
4) On the "Modbus Slave Communication Configuration" page, add one read configuration for the read slave address 0x0000 with the length of 2WORD, corresponding to H5U D0 and D1, and one write configuration for the write slave address 0x000A with the length of 2WORD, corresponding to H5U D10 and D11.



5) Add related mapping variables to facilitate monitoring and control.



6) Configure H5U COM port parameters, as shown in the following figure.



7) Enable the slave enable variable SM1001, and ensure successful read and write based on the H5U variables.

Expression	Type	Value
SysHC_ModbusRtuDeviceDiagnose_0	SysHC_ModbusRtuDevice...	
SysHC_ModbusRtuSlaveDiagnose_0	SysHC_ModbusRtuSlaveD...	
Data	ARRAY [0..3] OF INT	
Data[0]	INT	1
Data[1]	INT	2
Data[2]	INT	3
Data[3]	INT	4

```
SysHC_ModbusRtuDeviceDiagnose_0(
    xEnable TRUE :=TRUE ,
    byComID:= ,
    xValid=> ,
    xBusy=> ,
    xError=> ,
    eErrorID=> ,
    byDiagData=> );

SysHC_ModbusRtuSlaveDiagnose_0(
    xEnable TRUE :=TRUE ,
    byComID:= ,
    bySlaveID 1 :=1 ,
    xValid=> ,
    xBusy=> ,
    xError=> ,
    eErrorID=> ,
    stDiagData=> );

SM1001 TRUE :=1;RETURN
```

...	D0	INT	Decimal	1
...	D1	INT	Decimal	2
...	D10	INT	Decimal	3
...	D11	INT	Decimal	4

Error state

Change the slave number to 2 to mismatch the H5U slave number 1.

SysHC_ModbusRtuSlaveDiagnose_0	SysHC_ModbusRtuSlaveDiagnose	
xEnable	BOOL	TRUE
byComID	BYTE	0
bySlaveID	BYTE	1
xValid	BOOL	FALSE
xBusy	BOOL	FALSE
xError	BOOL	TRUE
eErrorID	SYS_HC_ERROR	SLAVE_NOT_EXIST
stDiagData	MODBUS_SLAVE_DIAGNOSE	
Data	ARRAY [0..3] OF INT	

```
1  SysHC_ModbusRtuDeviceDiagnose_0(
2      xEnable TRUE :=TRUE ,
3      byComID 0 :=0 ,
4      xValid=> ,
5      xBusy=> ,
6      xError=> ,
7      eErrorID=> ,
8      byDiagData=> );
9
10 SysHC_ModbusRtuSlaveDiagnose_0(
11     xEnable TRUE :=TRUE ,
12     byComID 0 :=0 ,
13     bySlaveID 1 :=1 ,
14     xValid=> ,
15     xBusy=> ,
16     xError=> ,
17     eErrorID=> ,
18     stDiagData=> );
19
20 SM1001 TRUE :=1;RETURN
```

The SLAVE_NOT_EXIST error is reported, indicating that the slave does not exist, which is consistent with the actual error.

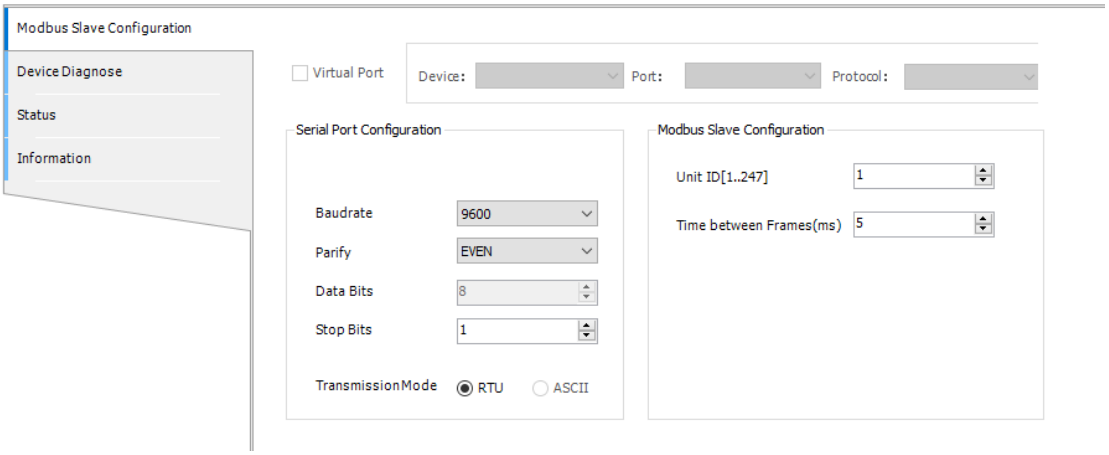
AM600 COM0 port acts as the Modbus RTU slave, while H5U acts as the Modbus RTU master.

Normal state

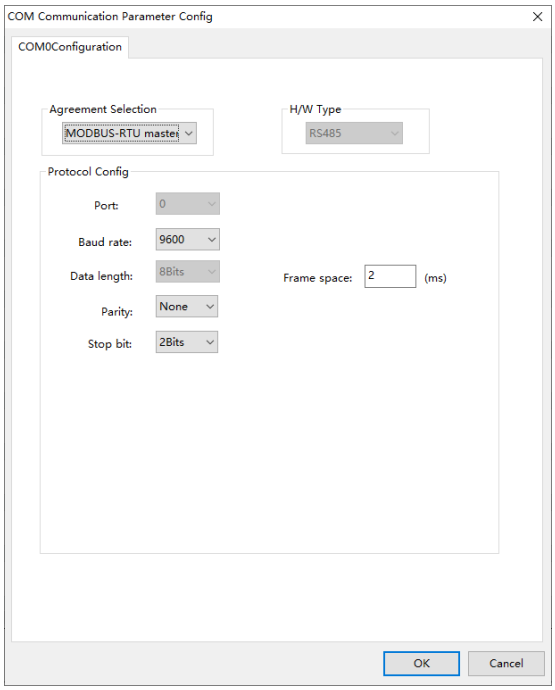
1) In "NetWork Configuration", select "Modbus Slave" for "COM0" and add a slave, as shown in the following figure.



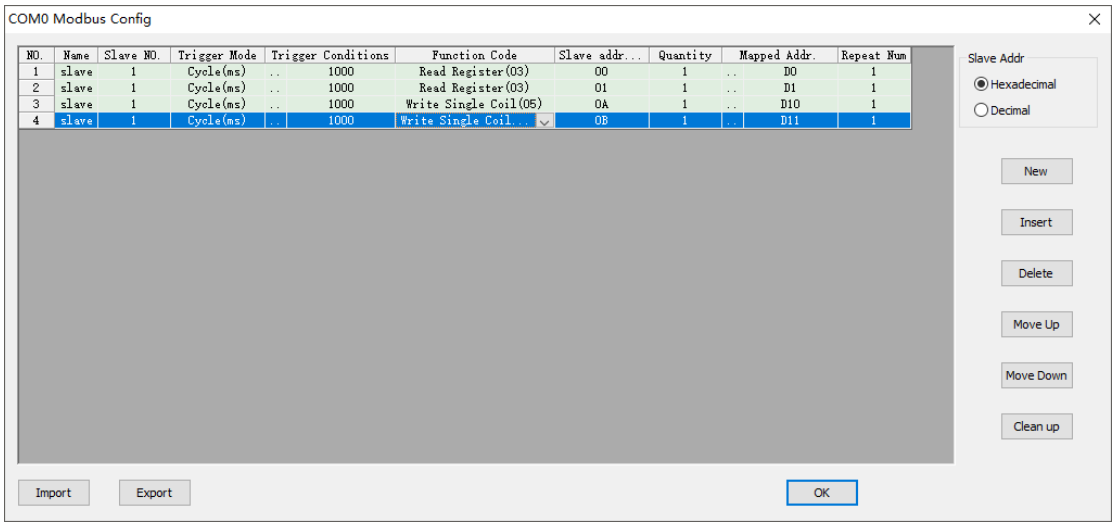
2) On the "Modbus Slave Configuration" page, configure the communication parameters, including "Baudrate", "Parify", "Data Bits", and "Stop Bits", and set "Unit ID[1..247]" to 1.



3) For the H5U COM port, select the Modbus RTU master protocol and set the communication parameters to be the same as those of the AM600 COM0 port.



4) Configure the communication configuration table for the COM port, including MW0 to MW1 for reading and MW10 to MW11 for writing.



The function block does not report an error and the parameters are consistent, indicating successful communication.

SysHC_ModbusRtuDeviceDiagnose_0	SysHC_ModbusRtuDeviceDiagnose				
xEnable	BOOL	TRUE			
byComID	BYTE	0			
xValid	BOOL	TRUE			
xBusy	BOOL	TRUE			
xError	BOOL	FALSE			
eErrorID	SYS_HC_ERROR	NO_ERROR			
byDiagData	BYTE	0			
SysHC_ModbusRtuSlaveDiagnose_0	SysHC_ModbusRtuSlaveDiagnose				
Data	ARRAY [0..1] OF INT				%MW0
Data[0]	INT	1			%MB0
Data[1]	INT	2			%MB2
Data_0	ARRAY [0..1] OF INT				%MW10
Data_0[0]	INT	3			%MB20
Data_0[1]	INT	4			%MB22

```
SysHC_ModbusRtuDeviceDiagnose_0(
    xEnable TRUE :=TRUE ,
    byComID 0 :=0 ,
    xValid=> ,
    xBusy=> ,
    xError=> ,
    eErrorID=> ,
    byDiagData=> );
```

...	D0	INT	Decimal	1
...	D1	INT	Decimal	2
...	D10	INT	Decimal	3
...	D11	INT	Decimal	4

Error state

Set comID to 10, which is incorrect. The WRONG_PARAMETER error is reported, indicating that the parameter settings are incorrect, which is consistent with the actual error.

SysHC_ModbusRtuDeviceDiagnose_0	SysHC_ModbusRtuDeviceDiagnose			
xEnable	BOOL	TRUE		
byComID	BYTE	10		
xValid	BOOL	FALSE		
xBusy	BOOL	FALSE		
xError	BOOL	TRUE		
eErrorID	SYS_HC_ERROR	WRONG_PARAMETER		
byDiagData	BYTE	0		
SysHC_ModbusRtuSlaveDiagnose_0	SysHC_ModbusRtuSlaveDiagnose			
Data	ARRAY [0..1] OF INT			%MW0
Data_0	ARRAY [0..1] OF INT			%MW10

```
SysHC_ModbusRtuDeviceDiagnose_0(
    xEnable TRUE :=TRUE ,
    byComID 10 :=10 ,
    xValid=> ,
    xBusy=> ,
    xError=> ,
    eErrorID=> ,
    byDiagData=> );
```

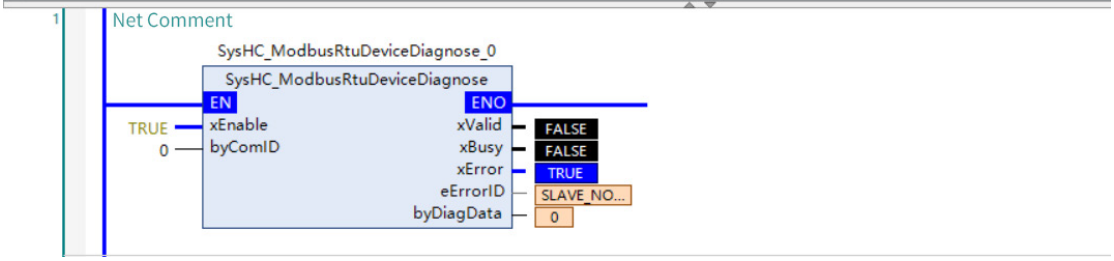
LD

AM600 COM0 port acts as the Modbus RTU master, while H5U acts as the Modbus RTU slave.

The process of establishing a connection is consistent with the ST language.

As shown in the following figure, the communication parameters "Baudrate", "Stop Bits", and "Parify" are set to mismatch the slave, reporting an error of no slave, which is consistent with the actual error.

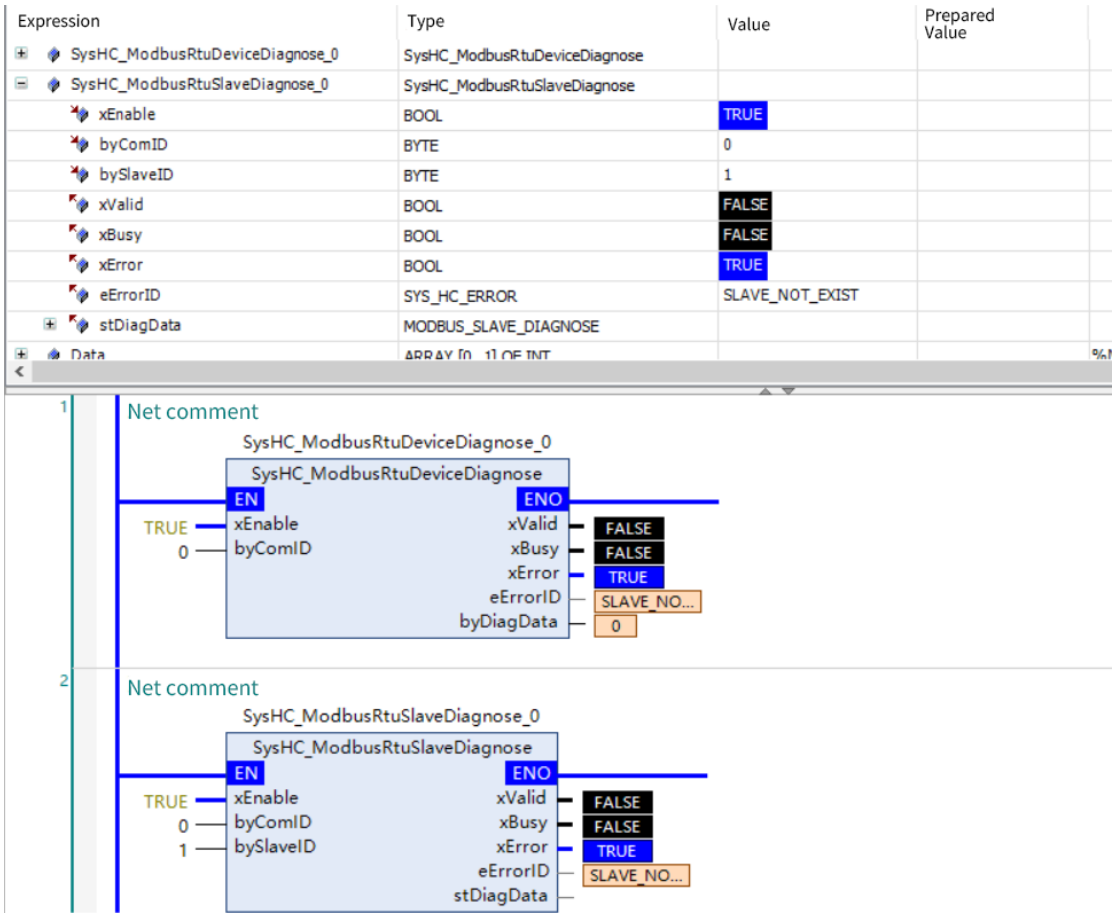
SysHC_ModbusRtuDeviceDiagnose_0	SysHC_ModbusRtuDeviceDiagnose			
xEnable	BOOL	TRUE		
byComID	BYTE	0		
xValid	BOOL	FALSE		
xBusy	BOOL	FALSE		
xError	BOOL	TRUE		
eErrorID	SYS_HC_ERROR	SLAVE_NOT_EXIST		
byDiagData	BYTE	0		
SysHC_ModbusRtuSlaveDiagnose_0	SysHC_ModbusRtuSlaveDiagnose			
Data	ARRAY [0..1] OF INT			%MW0
Data_0	ARRAY [0..1] OF INT			%MW10



AM600 COM0 port acts as the Modbus RTU slave, while H5U acts as the Modbus RTU master.

The process of establishing a connection is consistent with the ST language.

In "NetWork Configuration", deselect "Modbus Slave". The SLAVE_NOT_EXIST error is reported, indicating that the slave does not exist, which is consistent with the actual error.



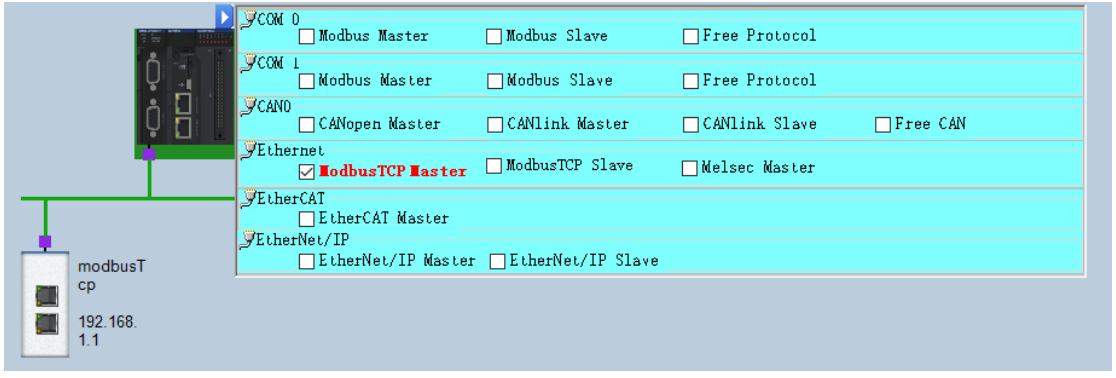
AM600 acts as the Modbus TCP master, while H5U acts as the Modbus TCP slave.

The AM600 IP address is 192.168.1.88, while the H5U IP address is 192.168.1.100.

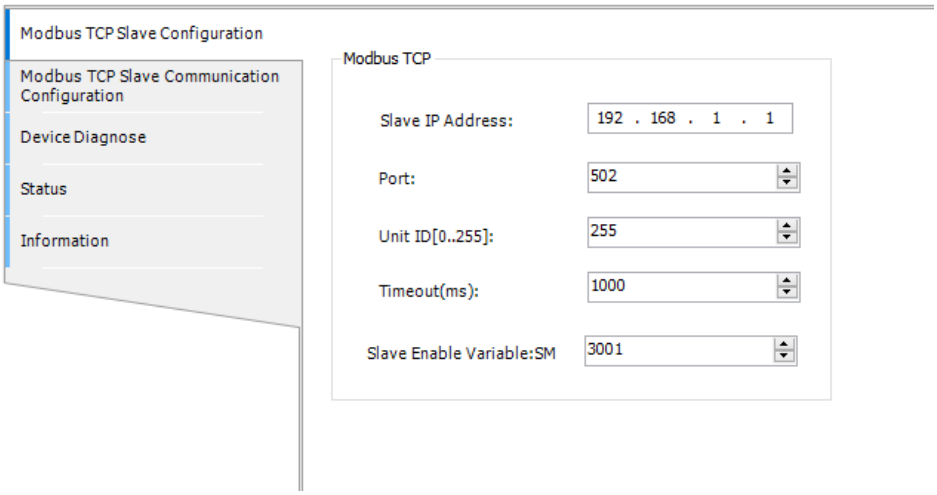
ST

Normal state

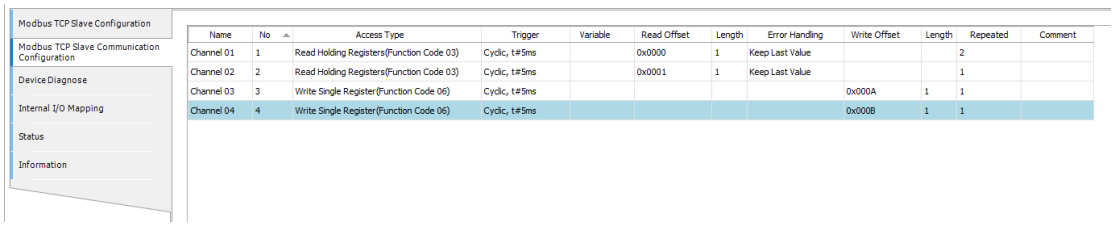
1) In "NetWork Configuration", select "ModbusTCP Master" and add a slave, as shown in the following figure.



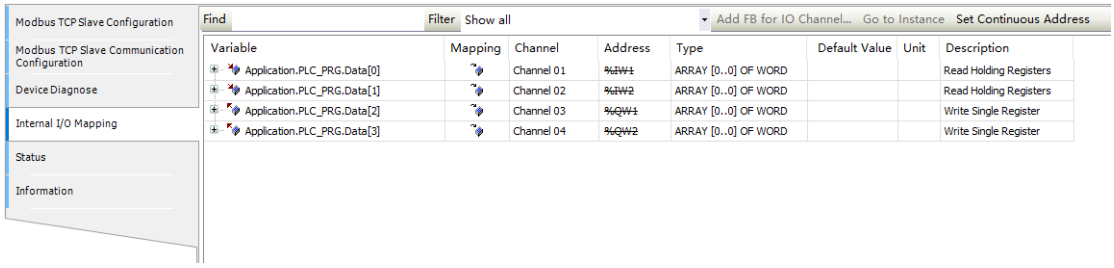
2) On the "Modbus TCP Slave Configuration" page, set "Slave IP Address" to "192.168.1.100" for H5U.



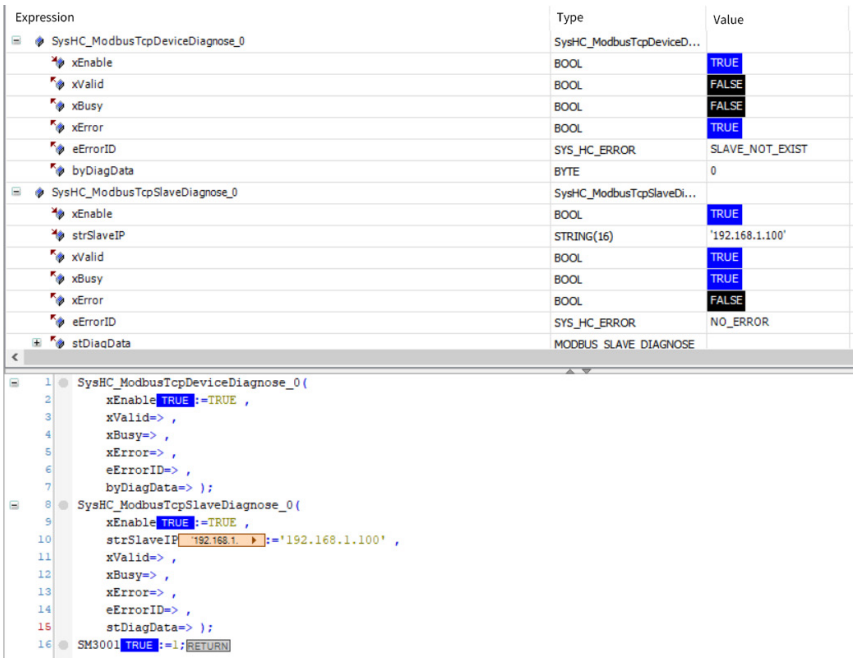
3) On the "Modbus TCP Slave Communication Configuration" page, read H5U D0 and D1 and write D10 and D11.



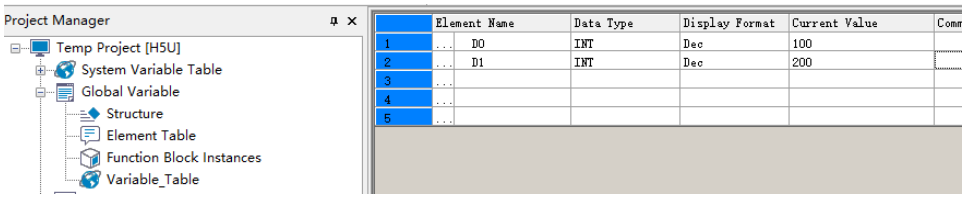
4) Add mapping elements for the four configurations to facilitate checking. For example, INT arrays of four elements are established.



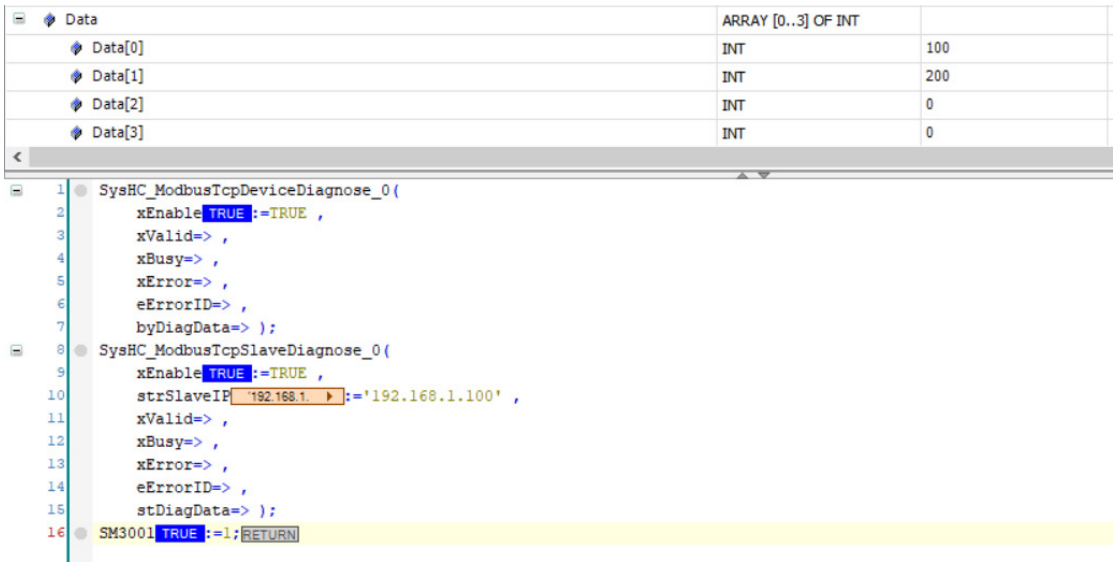
5) Set the slave enable variable SM3001 to TRUE. Then the communication is normal and NO_ERROR is reported.



6) Write 100 and 200 for H5U D0 and D1 respectively.



7) Check that the first two element values of Data are 100 and 200, indicating successful reading.

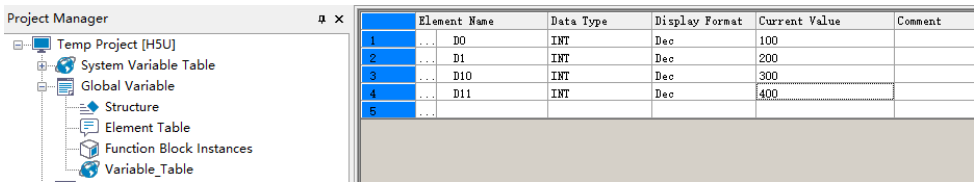


8) Write 300 and 400 for elements Data[2] and Data[3] in the AM600 Data array respectively.

Data	ARRAY [0..3] OF INT	
Data[0]	INT	100
Data[1]	INT	200
Data[2]	INT	300
Data[3]	INT	400

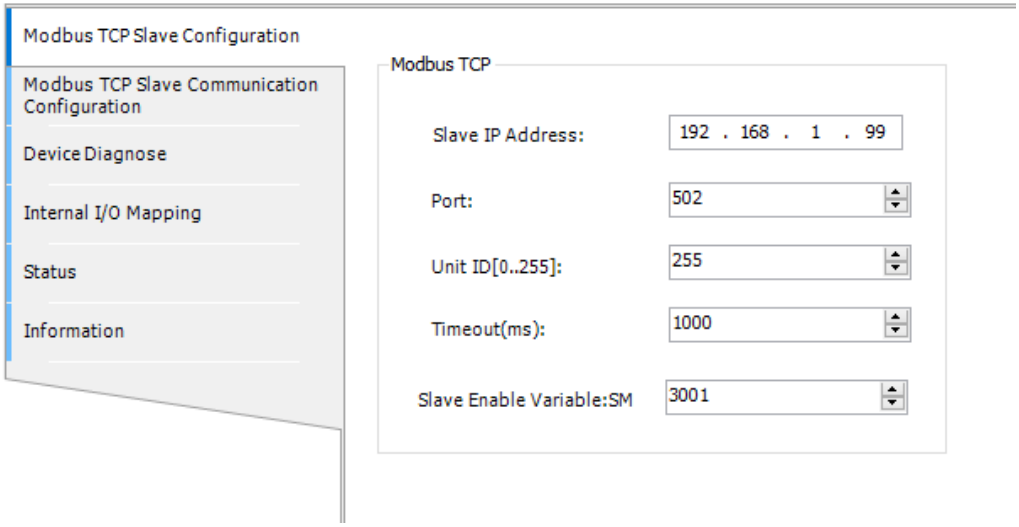
```
SysHC_ModbusTcpDeviceDiagnose_0(
    xEnable TRUE :=TRUE ,
    xValid=> ,
    xBusy=> ,
    xError=> ,
    eErrorID=> ,
    byDiagData=> );
SysHC_ModbusTcpSlaveDiagnose_0(
    xEnable TRUE :=TRUE ,
    strSlaveIP '192.168.1.' :='192.168.1.100' ,
    xValid=> ,
    xBusy=> ,
    xError=> ,
    eErrorID=> ,
    stDiagData=> );
SM3001 TRUE :=1;RETURN
```

9) Check that the values of H5U D10 and D11 are 300 and 400, indicating successful writing.



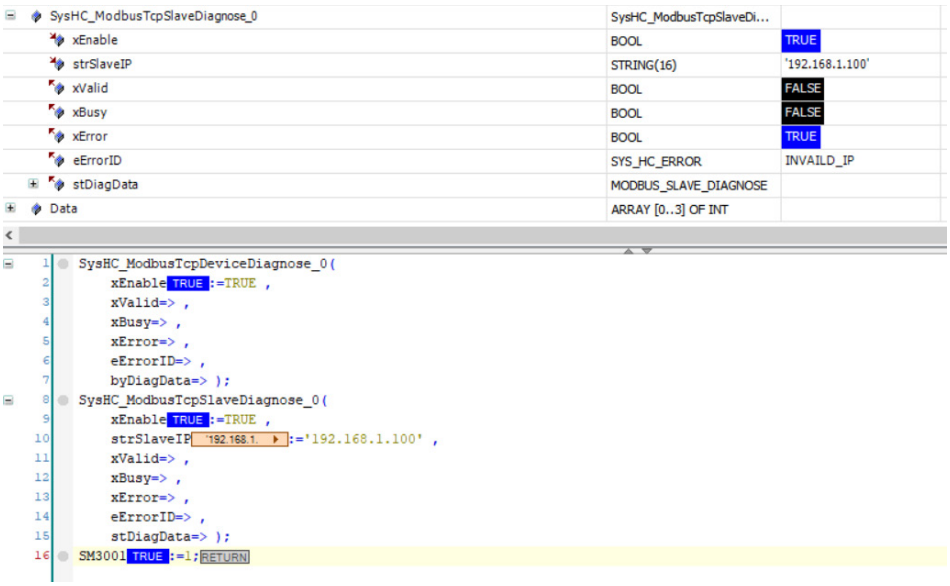
Error state

On the "Modbus TCP Slave Configuration" page, set "Slave IP Address" to "192.168.1.99".



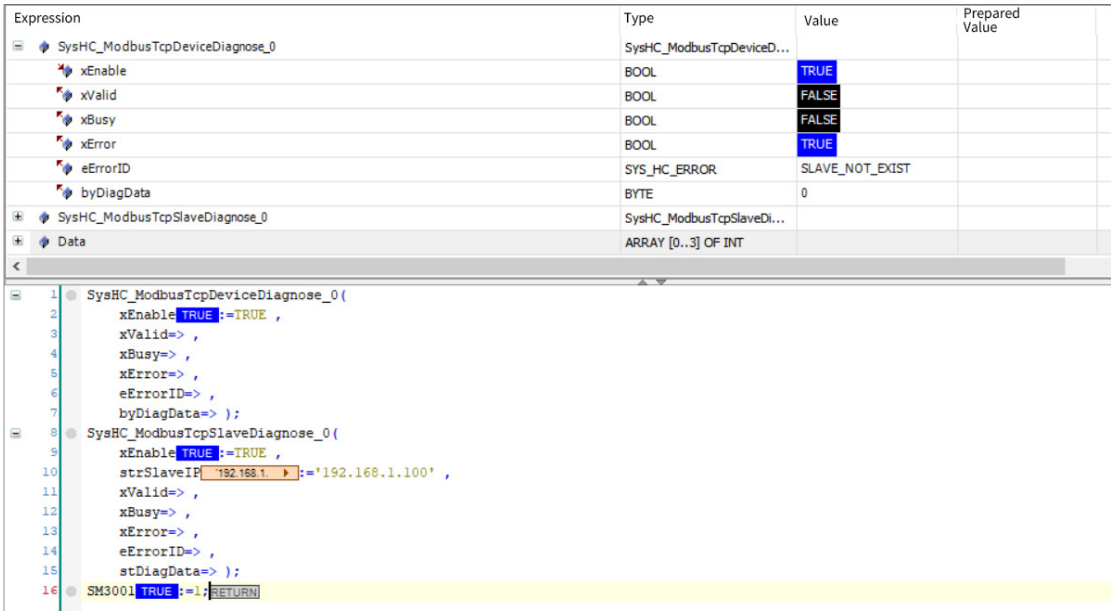
Check the function block state.

After execution of sysHC_ModbusTcpSlaveDiagnose, eError is reported as INVALID_IP, indicating that the IP address is invalid, which is consistent with the actual error.

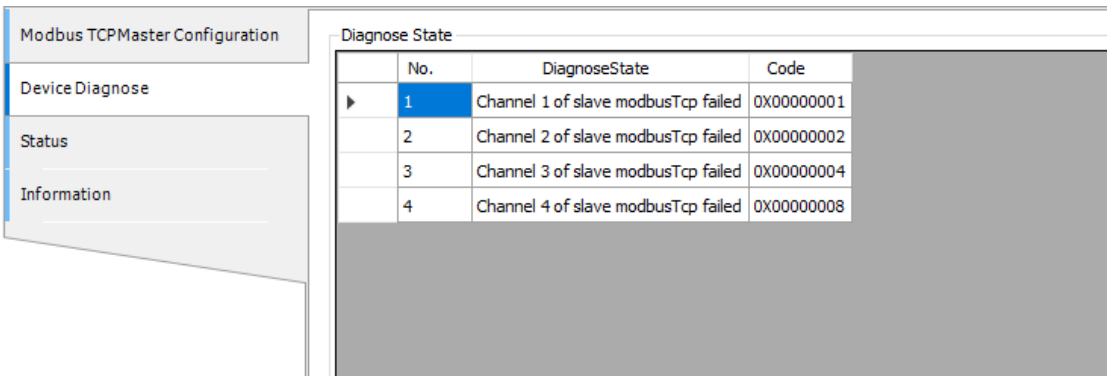


Write values for elements Data[2] and Data[3] in the AM600 Data array respectively.

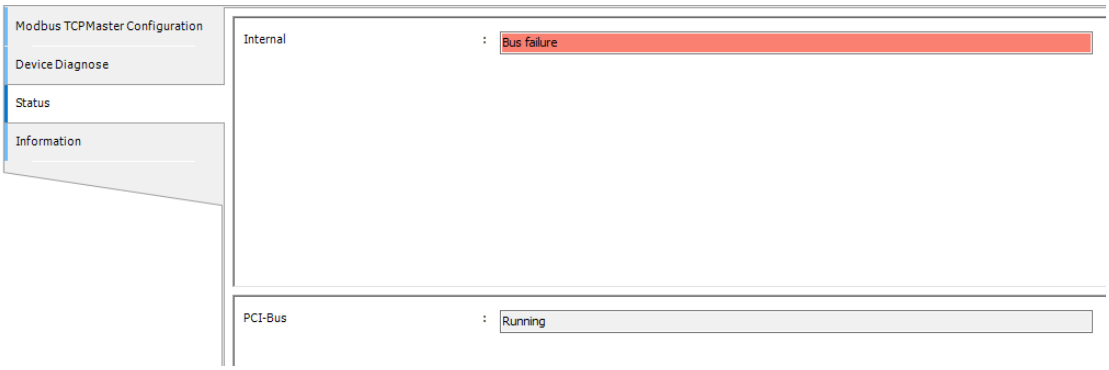
For execution of the sysHC_ModbusTcpDeviceDiagnose function block, eERROR_ID is SLAVE_NOT_EXIST, indicating that the slave does not exist, which is consistent with the actual error. In this case, AM600 has no slave attributes.



The configuration communication fails during Modbus TCP master device diagnosis.



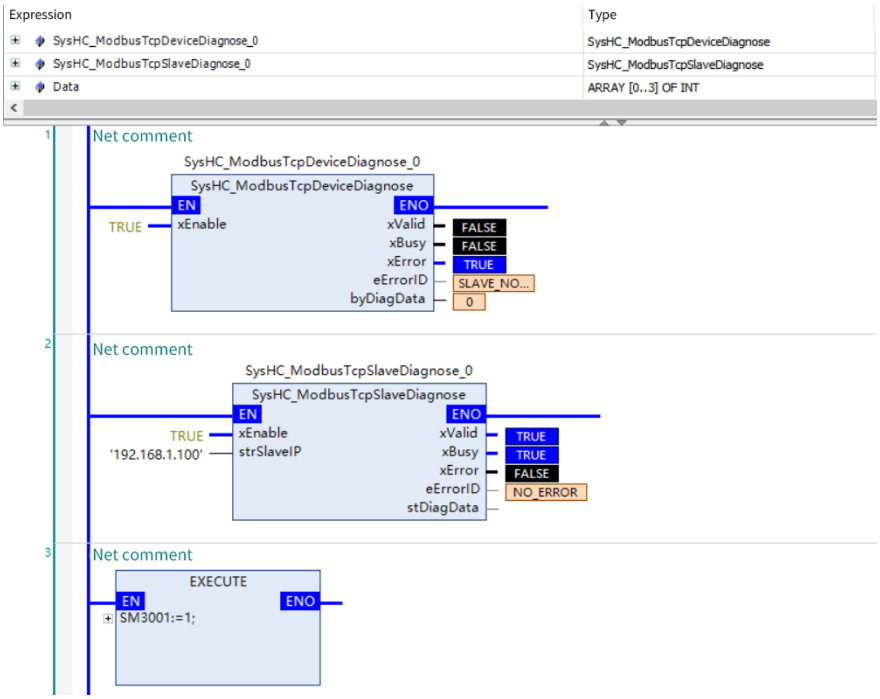
The state is failed.



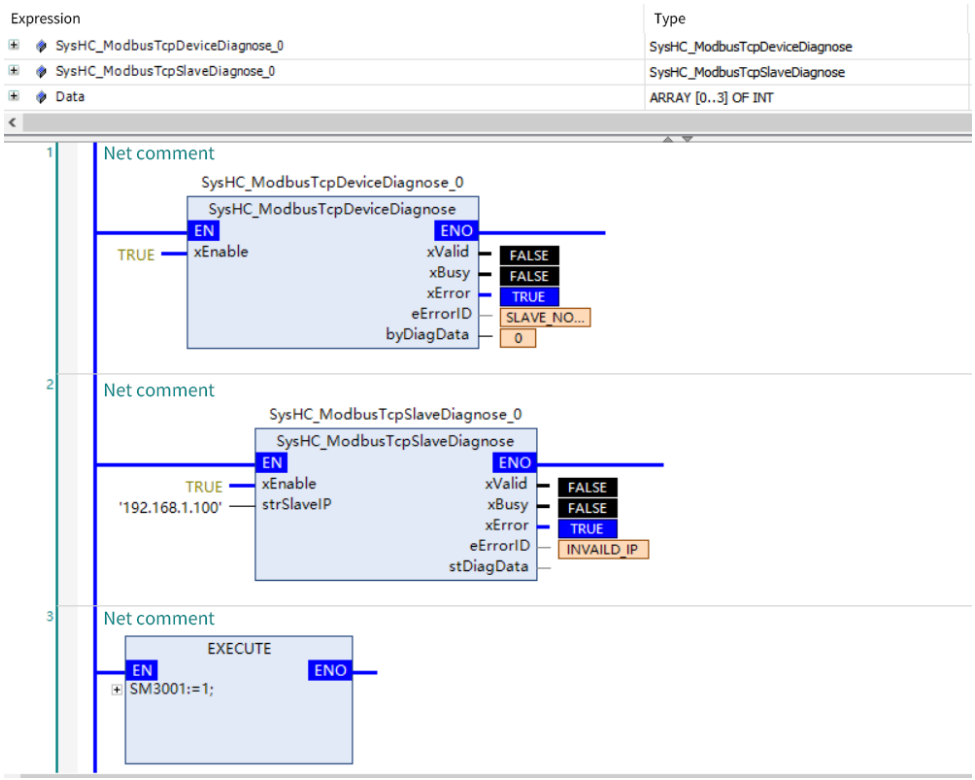
LD

The communication mode is consistent with the ST language.

The successful communication effect is shown in the following figure.



After the IP address is changed to an incorrect one, the error is reported, as shown in the following figure. In this case, AM600 has no slave attributes.



■ Precautions

Error code

Error Code	Description
0x70	Incorrect Modbus slave address
0x71	Incorrect data frame length, serial port 0 (COM0)
0x72	Illegal data address
0x73	CRC error
0x74	Unsupported instruction code
0x75	Reception timeout
0x76	Illegal data value
0x77	Buffer overflow
0x78	Frame error
0x79	Serial protocol error
0x7C	Incorrect address
0x7D	No data received
0x7E	Incorrect data returned by the slave
0x80	Incorrect Modbus slave address
0x81	Incorrect data frame length, serial port 1 (COM1)
0x82	Illegal data address
0x83	CRC error
0x84	Unsupported instruction code
0x85	Reception timeout
0x86	Illegal data value
0x87	Buffer overflow
0x88	Frame error

Error Code	Description
0x89	Serial protocol error
0x8C	Incorrect address
0x8D	No data received
0x8E	Incorrect data returned by the slave
0x8E	Incorrect data returned by the slave
0x90	Incorrect Modbus slave address
0x91	Incorrect data frame length, Ethernet (Modbus TCP)
0x92	Illegal data address
0x93 CRC	CRC error
0x94	Unsupported instruction code
0x95	Reception timeout
0x96	Illegal data value
0x97	Buffer overflow
0x98	Frame error
0x99	Serial protocol error
0x9A	Slave not connected
0x9B	Incorrect protocol identifier
0x9C	Incorrect address
0x9D	No data received
0x9E	Incorrect data returned by the slave
0x9F	Number of connected clients out of range
0xA0	Illegal data value

6.7.17 ModbusTCPSlave

This instruction disables the Modbus TCP slave in the same way as the ModbusTCPSlaveDisable. The ModbusTCPSlaveDisable new instruction can be used.

6. 8 Free Serial Communication Instructions

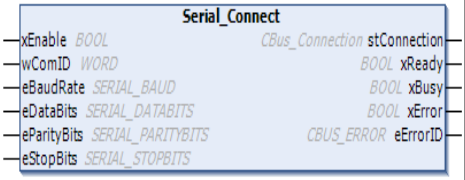
6.8. 1 Instruction List

Instruction Category	Name	FB/FC	Function
Free Serial Communication Instructions	Serial_Connect	FB	Serial port free protocol configuration
	Serial_Recv	FB	Serial port free protocol transmission
	Serial_Send	FB	Serial port free protocol reception

6.8. 2 Serial_Connect

This instruction connects serial ports by using free protocols.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
Serial_Connect	Serial port free protocol connection	FB		Serial_Connect(xEnable:= , wComID:= , eBaudRate:= , eDataBits:= , eParityBits:= , eStopBits:= , stConnection=> , xReady=> , xBusy=> , xError=> , eErrorID=>);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
xEnable	Function block enable	BOOL	-	FALSE	Triggered by level FALSE: Serial port disabled TRUE: Serial port enabled
wComID	Slave IP address	WORD	-	0	COM port number
eBaudRate	Baud rate	ENUM	-	9600	Baud rate
eDataBits	Data bit	ENUM	-	8	Data bit
eParityBits	Parity bit	ENUM	-	0	Parity bit
eStopBits	Stop bit	ENUM	-	1	Stop bit

■ Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
stConnection	Connection handle	STRUCT	-	-	Connection handle
xReady	Instruction status	BOOL	-	FALSE	Function block valid flag
xBusy	Instruction execution flag	BOOL	-	FALSE	Function block execution flag
xError	Error flag	BOOL	-	FALSE	Function block exception flag
eErrorID	Error code	ENUM		OK	Error code for the function block exception

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
xEnable	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

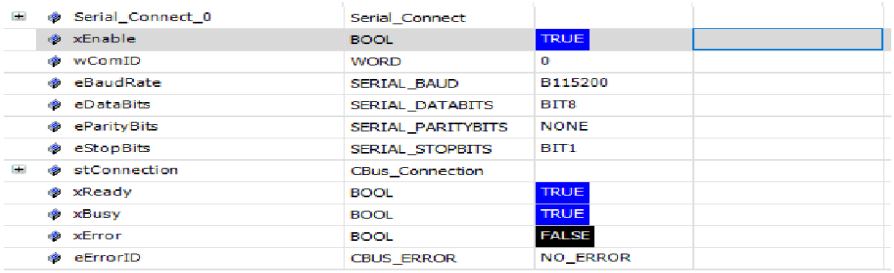
	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
wComID	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
eBaudRate	ENUM																			
eDataBits	ENUM																			
eParityBits	ENUM																			
eStopBits	ENUM																			
stConnection	STRUCT																			
xReady	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
xBusy	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
xError	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
eErrorID	ENUM																			

■ Function

This instruction allows users to modify parameters during running, such as the data transmission rate and parity bit.

■ Program example

ST




```
//Serial_Connect_0
Serial_Connect_0(
    xEnable:= xEnable,
    wComID:= wComID,
    eBaudRate:= eBaudRate,
    eDataBits:= eDataBits,
    eParityBits:= eParityBits,
    eStopBits:= eStopBits,
    stConnection=> stConnection,
    xReady=> xReady,
    xBusy=> xBusy,
    xError=> xError,
    eErrorID=> eErrorID);

RETURN
```

Precautions

1) The connection handle of MB_Connect can only be used by either the Serial_Recv or Serial_Send instruction.

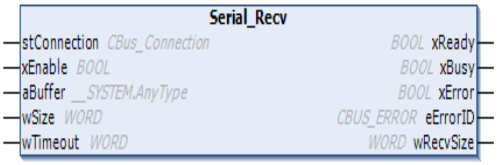
2) Ensure that multiple instructions with the same connection handle are used in the same user task.

3) Online parameter modifications do not take effect immediately. It is necessary to re-enable the instruction.

6.8. 3 Serial_Recv

This instruction receives data through serial ports by using free protocols.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
Serial_Recv	Serial port free protocol reception	FB		Serial_Recv(stConnection:= , xEnable:= , aBuffer:= , wSize:= , wTimeout:= , xReady=> , xBusy=> , xError=> , eErrorID=> , wRecvSize=>);

■ Variables

In-out variables

In-Out Variable	Name	Data Type	Value Range	Initial Value	Description
stConnection	Connection handle	STRUCT	-	-	Connection handle

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
xEnable	Function block enable	BOOL	-	FALSE	Triggered by level
aBuffer	Reception buffer	ANY	-	-	Reception buffer for serial port data
wSize	Number of bytes to receive	WORD	-	0	Number of data bytes to receive from the serial port
wTimeout	Timeout	WORD	-	0	Timeout interval of serial port data

■ Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
xReady	Instruction status	BOOL	-	FALSE	Function block valid flag
xBusy	Instruction execution flag	BOOL	-	FALSE	Function block execution flag
xError	Error flag	BOOL	-	FALSE	Function block exception flag
eErrorID	Error code	ENUM	-	OK	Error code for the function block exception
wRecvSize	Number of received bytes	WORD	-	0	Number of data bytes received from the serial port

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
stConnection	STRUCT																			
xEnable	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
aBuffer	ANY																			
wSize	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
wTimeout	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
xReady	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
xBusy	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
xError	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
wRecvSize	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
eErrorID	ENUM																			

■ Function

This instruction starts data receiving to save the serial port bus data to the reception buffer.

■ Program example

ST

⊞ Serial_Connect_0	Serial_Connect	
xEnable	BOOL	TRUE
wComID	WORD	0
eBaudRate	SERIAL_BAUD	B115200
eDataBits	SERIAL_DATABITS	BIT8
eParityBits	SERIAL_PARITYBITS	NONE
eStopBits	SERIAL_STOPBITS	BIT1
⊞ stConnection	CBus_Connection	
xReady	BOOL	TRUE
xBusy	BOOL	TRUE
xError	BOOL	FALSE
eErrorID	CBUS_ERROR	NO_ERROR
⊞ Serial_Recv_0	Serial_Recv	
xEnable_0	BOOL	TRUE
⊞ recv_data	ARRAY [1..100] OF ...	
wSize	WORD	1
wTimeout	WORD	0
xReady_0	BOOL	FALSE
xBusy_0	BOOL	TRUE
xError_0	BOOL	FALSE
eErrorID_0	CBUS_ERROR	NO_ERROR
wRecvSize	WORD	0

```
//Serial_Connect_0
Serial_Connect_0(
    xEnable TRUE := xEnable TRUE,
    wComID 0 := wComID 0,
    eBaudRate B115200 := eBaudRate B115200,
    eDataBits BIT8 := eDataBits BIT8,
    eParityBits NONE := eParityBits NONE,
    eStopBits BIT1 := eStopBits BIT1,
    stConnection=> stConnection,
    xReady TRUE => xReady TRUE,
    xBusy TRUE => xBusy TRUE,
    xError FALSE => xError FALSE,
    eErrorID NO_ERROR => eErrorID NO_ERROR );

//Serial_Recv_0
Serial_Recv_0(
    stConnection:= stConnection,
    xEnable TRUE := xEnable_0 TRUE,
    aBuffer:= recv_data,
    wSize 1 := wSize 1,
    wTimeout 0 := wTimeout 0,
    xReady FALSE => xReady_0 FALSE,
    xBusy TRUE => xBusy_0 TRUE,
    xError FALSE => xError_0 FALSE,
    eErrorID NO_ERROR => eErrorID_0 NO_ERROR,
    wRecvSize 0 => wRecvSize 0 );RETURN
```

■ Precautions

1) It must be used with the Serial_Connect instruction.

2) If the timeout interval parameter is set, the timeout interval begins to be calculated when serial port data fails to be received or there is no data in the serial port. If there is still no data when the timeout interval expires, an error will be reported for receiving timeout. If the serial port receives data when the timeout interval does not expire, the timeout interval will be reset.

3) When wRecvSize is set to wSize, the data reception is complete, xReady will be set to one cycle and reset in the next cycle, and wRecvSize will be cleared.

4) Data types loaded by aBuffer are as follows:

· Supported types

One-dimensional array

Data types of the arrays/variables are as follows: BYTE, WORD, DWORD, SINT, USINT, INT, UINT, DINT, UDINT, and REAL

· Unsupported types

Multidimensional array

· 64-bit data types: LWORD, LINT, ULINT, and LREAL

· Special types: STRING, WSTRING, TIME, TIME OF DAY(TOD), DATE, DATE OF TIME(DT), POINTER, and STRUCT

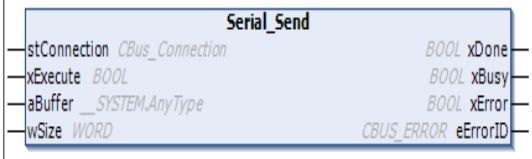
5) aBuffer cannot be directly set to I/Q/M direct variables. It needs to be mapped to the above supported variable/array data by using the AT method.

6) Online parameter modifications do not take effect immediately. It is necessary to re-enable the instruction.

6.8. 4 Serial_Send

This instruction sends data through serial ports by using free protocols.

■ Instruction format

Instruction	Name	FB/ FC	LD Expression	ST Expression
Serial_Send	Serial port free protocol transmission	FB		Serial_Send(stConnection:= , xExecute:= , aBuffer:= , wSize:= , xDone=> , xBusy=> , xError=> , eErrorID=>);

■ Variables

In-out variables

In-Out Variable	Name	Data Type	Value Range	Initial Value	Description
stConnection	Connection handle	STRUCT	-	-	Connection handle

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
xExecute	Function block enable	BOOL	-	FALSE	Edge trigger
aBuffer	Transmission buffer	ANY	-	-	Serial port transmission buffer
wSize	Number of bytes to transmit	WORD	-	0	Number of data bytes sent by serial port

Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
xDone	Completion flag	BOOL	-	FALSE	Function block execution completion flag
xBusy	Instruction execution flag	BOOL	-	FALSE	Function block being executed
xError	Error flag	BOOL	-	FALSE	Function block exception flag
eErrorID	Error code	ENUM	-	OK	Error code for the function block exception

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
stConnection	STRUCT																			
xExecute	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
aBuffer	ANY																			
wSize	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
xDone	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
xBusy	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

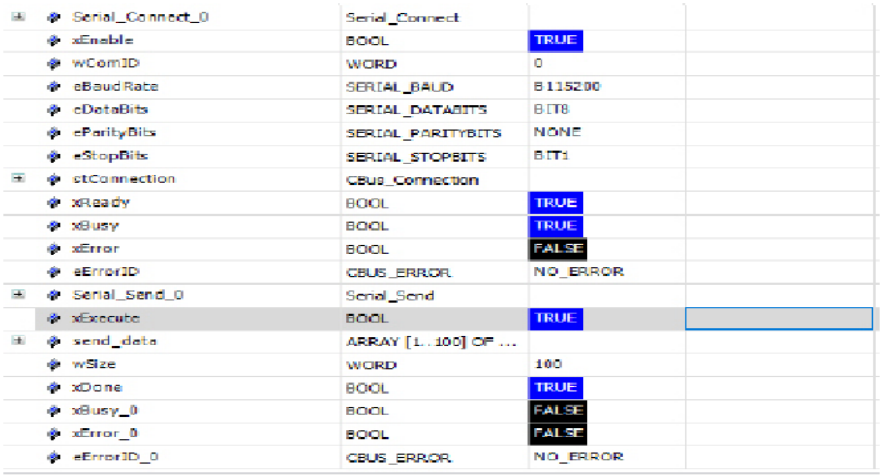
	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
xError	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
eErrorID	ENUM																			

■ Function

This instruction starts data transmission to send data in the transmission buffer by using the serial port bus.

■ Program example

ST




```
//Serial_Connect_0
Serial_Connect_0(
    xEnable:= xEnable,
    wComID:= wComID,
    eBaudRate:= eBaudRate,
    eDataBits:= eDataBits,
    eParityBits:= eParityBits,
    eStopBits:= eStopBits,
    stConnection=> stConnection,
    xReady=> xReady,
    xBusy=> xBusy,
    xError=> xError,
    eErrorID=> eErrorID);

Serial_Send_0(
    stConnection:= stConnection,
    xExecute:= xExecute,
    aBuffer:= send_data,
    wSize:= wSize,
    xDone=> xDone,
    xBusy=> xBusy_0,
    xError=> xError_0,
    eErrorID=> eErrorID_0):RETURN
```

■ Precautions

1) It must be used with the Serial_Connect instruction.

2) Online parameter modifications do not take effect immediately. It is necessary to re-enable the instruction.

3) Data types loaded by aBuffer are as follows:

· Supported types

One-dimensional array

Data types of the arrays/variables are as follows: BYTE, WORD, DWORD, SINT, USINT, INT, UINT, DINT, UDINT, and REAL

· Unsupported types

Multidimensional array

· 64-bit data types: LWORD, LINT, ULINT, and LREAL

· Special types: STRING, WSTRING, TIME, TIME OF DAY(TOD), DATE, DATE OF TIME(DT), POINTER, and STRUCT

4) aBuffer cannot be directly set to I/Q/M direct variables. It needs to be mapped to the above supported variable/array data by using the AT method.

6.8. 5 Error Codes

Name	Error Code	Description
NO_ERROR	0	No error
Error codes 0 to 199: Modbus diagnosis codes in the Medium-Sized PLC Programming Software User Guide		
CONNECT_FAILED	200	Connection failed
CONNECT_MODE_INVALID	201	Connection mode invalid
CONNECTION_INVALID	202	Connection handle invalid
CONNECTION_OCCUPY	203	Connection handle occupied
CONNECTION_MODIFY	204	Connection handle modified unexpectedly
MEMORY_ADDR_INVALID	205	Memory address invalid
MEMORY_OUT_OF_BOUNDS	206	Memory access out of bounds
DATA_DIM_INVALID	207	Data dimension invalid (only one-dimension supported)
DATA_ADDR_INVALID	208	Data address invalid
DATA_TYPE_INVALID	209	Data type invalid
DATA_VALUE_INVALID	210	Value invalid
SEND_FAILED	211	Data sending failed
RECV_FAILED	212	Data receiving failed
RECV_TIMEOUT	213	Data receiving timeout
Reserved		
SERIAL_COM_INVALID	300	COM port invalid
SERIAL_COM_UNPREADY	301	COM port not ready
SERIAL_COM_USED	302	COM used
SERIAL_BAUDRATE_ERR	303	COM baud rate configuration error
SERIAL_DATABITS_ERR	304	COM data bit configuration error
SERIAL_PARITY_ERR	305	COM parity bit configuration error
SERIAL_STOPBITS_ERR	306	COM stop bit configuration error
Reserved		
MB_CONFIG_ERR	400	Modbus configuration failed
MB_FUNC_CODE_INVALID	401	Modbus parameter invalid
MB_QUANTITY_INVALID	402	Modbus address quantity invalid
MB_RETRY_TIMES_INVALID	403	Modbus resending times invalid (0 to 15)
MB_SLAVE_ID_INVALID	404	Modbus slave ID invalid
MB_SLAVE_ADDR_ILLEGAL	405	Modbus slave address illegal
MB_SLAVE_DATA_ILLEGAL	406	Modbus slave data illegal
MB_MAPPING_MEM_ILLEGAL	407	Modbus slave mapping address illegal
MB_MAPPING_NUM_INVALID	408	Modbus slave mapping number illegal

Name	Error Code	Description
MB_RTU_DATABITS7_ILLEGAL	409	Modbus RTU 7-bit data bit configuration illegal
MB_ASC_DATABITS8_ILLEGAL	410	Modbus ASCII 8-bit data bit configuration illegal
Reserved		
MB_TCP_CONNECT_FAILED	500	Modbus TCP connection failed
MB_TCP_IP_ADDR_INVALID	501	Modbus TCP IP address invalid
MB_TCP_ETHER_ID_INVALID	502	Modbus TCP network port invalid
MB_TCP_DELAY_POLL_INVALID	503	Modbus TCP inter-frame interval invalid (1 to 1000)
Reserved		
UNKNOWN	65535	Unknown error

■ Description

The above error codes only support the MB_Connect, MB_Master, MB_Slave, MB_TCP_Connect, MB_Client, and MB_Server instructions mentioned in section 6.6, as well as the free serial port instructions in section 6.7.

6. 9 Modbus TCP Communication Instructions

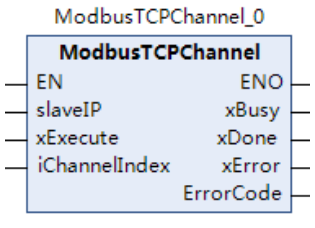
6.9. 1 Instruction List

Instruction Category	Name	FB/FC	Function
Modbus TCP communication instructions	ModbusTCPChannel	FB	Trigger channel of ModbusTcp master
	ModbusTCPServerConfig	FB	Modbus TCP slave configuration
	ModbusTCPSlaveSetIPAddr	FB	Modbus TCP slave IP address configuration
	ModbusTCPSlaveDisable	FB	Modbus TCP slave disable
	sysHC_ModbusTcpDeviceDiagnose	FB	Get diagnostic information of ModbusTcp slave devices
	sysHC_ModbusTcpSlaveDiagnose	FB	Get diagnostic data of ModbusTcp master accessing slaves

6.9. 2 ModbusTCPChannel

This instruction implements triggering by channel for the Modbus TCP master.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
ModbusTCPChannel	Trigger channel of ModbusTcp master	FB		ModbusTCPChannel_0(slaveIP :=, xExecute :=, iChannelIndex :=, xBusy =>, xDone =>, xError =>, ErrorCode =>,);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
slaveIP	Slave IP address	STRING	Depends on data types	-	IP address of the slave 192.168.1.15: The IP address that accesses the slave is 192.168.1.15.
xExecute	Enable	BOOL	Depends on data types	-	(Triggered by edges) Execution of the function block switch
iChannelIndex	Channel number	INT	Depends on data types	-	Number of the communication channel 2: Channel 2

Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
xBusy	Function block execution flag	BOOL	Depends on data types	-	Function block execution flag
xDone	Function block completed state	BOOL	Depends on data types	-	Function block completed state
xError	Function block error flag	BOOL	Depends on data types	-	Function block error flag
ErrorCode	Error information of function block	Modbus_ERROR	Depends on data types	-	Error information of function block

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
slaveIP	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√
xExecute	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
iChannelIndex	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-
xBusy	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
xDone	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
xError	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
ErrorCode	See "Function" for the enumerators for the enumerated type Modbus_ERROR.																			

■ Function

This instruction supports triggering by channel for the Modbus TCP master.

Change the channel mode to cyclic execution before use. When the cycle time (ms) is set to 65535, the trigger mode is enabled and then the function block triggers the cyclic execution.

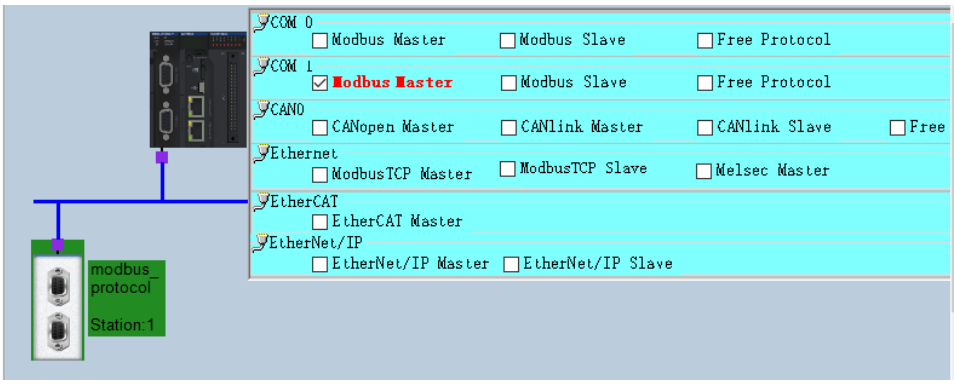
When xExecute is set to FALSE, trigger enable is cleared.

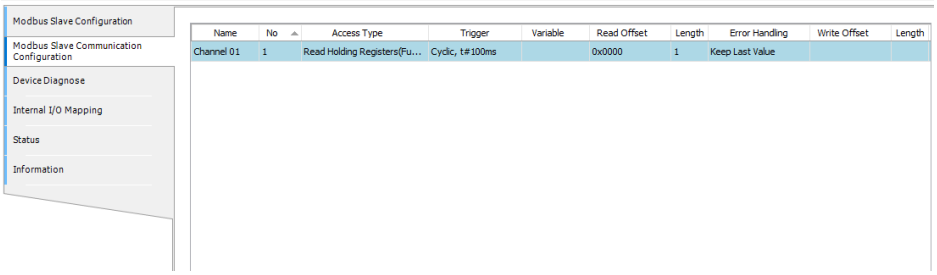
The data type of ErrorCode is enumerated type Modbus_ERROR. The meanings of the enumerators are as follows.

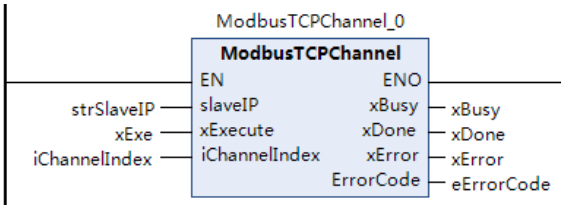
Error code	Error	Description
0	Modbus_NO_ERROR	Modbus communication is normal.
1	Modbus_NO_COM_INDEX	The COM port configured based on Modbus RTU does not exist or cannot be accessed.

Error code	Error	Description
2	Modbus_NO_COM_SLAVE	The configuration of the Modbus RTU slave number is incorrect.
3	Modbus_NO_COM_CHANNEL	The configuration of the Modbus RTU channel number is incorrect.
4	Modbus_NO_TCP_SLAVE	The configuration of the Modbus TCP slave IP address is incorrect.
5	Modbus_NO_TCP_CHANNEL	The configuration of the Modbus TCP communication channel is incorrect, or the channel does not exist.
6	Modbus_OTHER_FAILED	Other errors

■ Program example

<table>
<tr><th></th><th>LD</th><th>ST</th></tr>
<tr><td>Defined variable</td><td colspan="2">





```
VAR
    ModbusTCPChannel_0: ModbusTCPChannel;
    strSlaveIP : STRING := "192.168.1.15";
    xExe : BOOL := TRUE;
    iChannelIndex : INT := 1;
    xBusy : BOOL;
    xDone : BOOL;
    xError : BOOL;
    eErrorCode : MODBUS_ERROR;
END_VAR
```
</td></tr>
<tr><td>Program</td><td>


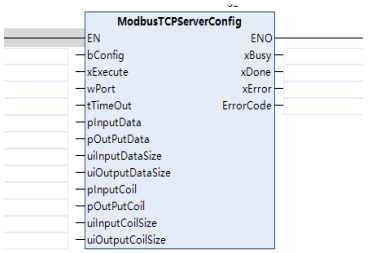
</td><td>

```
ModbusTCPChannel_0(
    slaveIP:= strSlaveIP,
    xExecute:= xExe,
    iChannelIndex:= iChannelIndex,
    xBusy=> xBusy,
    xDone=> xDone,
    xError=> xError,
    ErrorCode=> eErrorCode);
```
</td></tr>
</table>

6.9. 3 ModbusTCPServerConfig

This instruction configures Modbus TCP slave devices.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
ModbusTCPServerConfig	Modbus TCP slave configuration	FB		ModbusTCPServerConfig_0 (bConfig:= , xEnable:= , wPort:= , tTimeOut:= , pInputData:= , pOutPutData:= , uiInputDataSize:= , uiOutputDataSize:= , pInputCoil:= , pOutPutCoil:= , uiInputCoilSize:= , uiOutputCoilSize:= , xBusy=> , xDone=> , xError=> , ErrorCode=>);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
bConfig	Default configuration standard	BOOL	[FALSE, TRUE]		Indicates whether to the configuration takes effect preferentially (reserved)
xExecute	Function block enable	BOOL	[FALSE, TRUE]		Function block enable
wPort	Port number	WORD	502	502	Port number
tTimeOut	Execution timeout period	UDINT	-	-	Write timeout period of the function block (reserved)
pInputData	Input register head address	POINTER TO WORD	%MW0 to %MW65535	%MW0 address	Head address of the M area: %MWx (x = 0 to 65535) Supported parameter: 0x04
pOutputData	Output register head address	POINTER TO WORD	%MW0 to %MW65535	%MW0 address	Head address of the M area: %MWx (x = 0 to 65535) Supported parameters: 0x03, 0x06, and 0x10

Input Variable	Name	Data Type	Value Range	Initial Value	Description
uiInputDataSize	Input register size	UDINT	1 to 65536	1 to 65536	Address width The value range is from 1 to 65536 (set head address offset + set address width).
uiOutput DataSize	Output register size	UDINT	1 to 65536	1 to 65536	Address width The value range is from 1 to 65536 (set head address offset + set address width).
pInputCoil	Input coil head address	POINTER TO BYTE	%IB0 to %IB8191	%IB0 address	Head address of the I/M area: %IBx/%MBx (x = 0 to 8191) Supported parameter: 0x02
pOutputCoil	Output coil head address	POINTER TO BYTE	%QB0 to %QB8191	%QB0 address	Head address of the M area: %QBx/%MBx (x = 0 to 8191) Supported parameters: 0x01, 0x05, and 0x0f
uiInputCoilSize	Input coil size	UDINT	1 to 8192	8192	Address width The value range is from 1 to 8192 (set head address offset + set address width).
uiOutput CoilSize	Input coil size	UDINT	1 to 8192	8192	Address width The value range is from 1 to 8192 (set head address offset + set address width).

■ Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
xBusy	Execution flag	BOOL	[FALSE, TRUE]	-	Function block being executed
xDone	Completion flag	BOOL	[FALSE, TRUE]	-	Output valid flag of the function block
xError	Error flag	BOOL	[FALSE, TRUE]	-	TRUE: An internal error occurs in the function block
ErrorCode	Error ID	Modbus_ERROR	Depends on data types	-	Error code

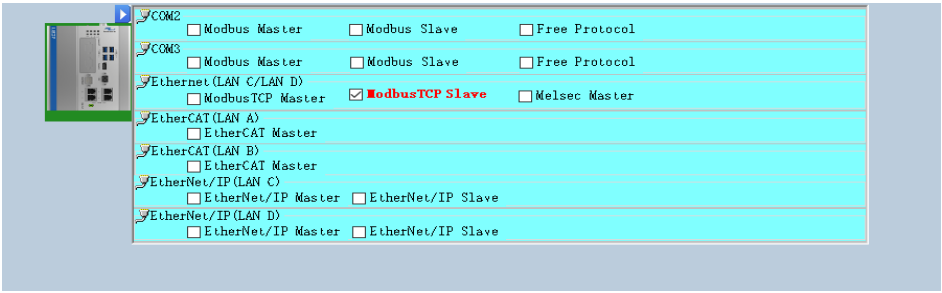
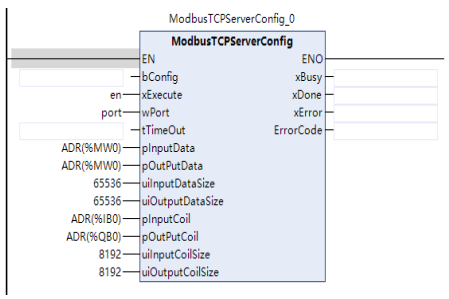
	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
bConfig	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
xExecute	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
wPort	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
tTimeOut	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-
pInputData	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
pOutputData	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
uiInputDataSize	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-
uiOutput DataSize	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
pInputCoil	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
pOutputCoil	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
uiInputCoilSize	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-
uiOutput CoilSize	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-
xBusy	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
xDone	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
xError	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
ErrorCode	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

■ Function

- This function block set parameters for slave devices in Modbus TCP slave device mode.
- The instruction is not supported for PLCs of the AM600 series.

■ Program example

	LD	ST
Defined variable	 VAR ModbusTCPServerConfig_0: ModbusTCPServerConfig; en: BOOL; port: WORD:=502; END_VAR	
Program		ModbusTCPServerConfig_0(bConfig:= , xExecute:= en, wPort:= port, tTimeOut:= , pInputData:= ADR(%MW0), pOutPutData:= ADR(%MW0), uiInputDataSize:= 65536, uiOutputDataSize:= 65536, pInputCoil:= ADR(%IB0), pOutPutCoil:= ADR(%QB0), uiInputCoilSize:= 8192, uiOutputCoilSize:= 8192, xBusy=> , xDone=> , xError=> , ErrorCode=>);

6.9. 4 ModbusTCPSlaveSetIPAddr

This instruction configures the Modbus TCP slave IP address.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
ModbusTCPSlaveSetIPAddr	Modbus TCP slave IP address configuration	FB		ModbusTCPSlaveSetIPAddr(xExecute:= , bySlaveNum:= , strSlaveIPAddr:= , wPort:= , xBusy=> , xDone=> , xError=> , ErrorCode=>);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
xExecute	Function block trigger execution	BOOL	[FALSE, TRUE]	-	Execution of trigger at the rising edge
bySlaveNum	Slave number	BYTE	0 to 255	0	Slave number Slave numbers are sorted by the slave sequence. The first number is 0, the second is 1, and the like, up to the number that can be configured by InoProShop.
strSlaveIPAddr	Slave IP address	STRING[17]	0.0.0.0 to 255.255.255.255	192.168.1.88	Slave IP address
wPort	Slave port number	WORD	0 to 65535	502	Port number

■ Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
xDone	Output active	BOOL	[FALSE, TRUE]	-	Output active
xBusy	Executing instruction	BOOL	[FALSE, TRUE]	-	Executing instruction
xError	Error	BOOL	[FALSE, TRUE]	-	Error
ErrorCode	Error ID	MODBUS_ERROR	Depends on data types	-	Error ID

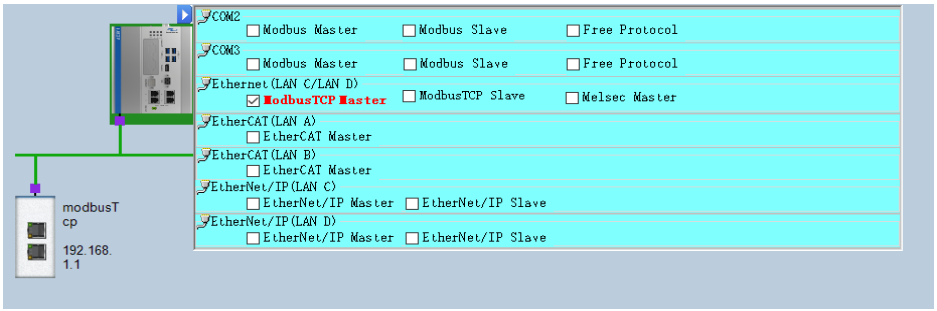
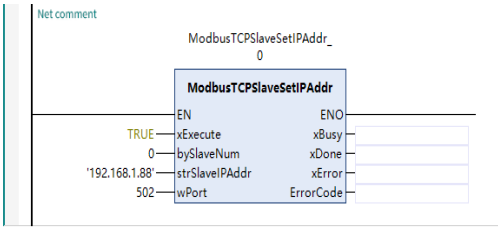
	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
xExecute	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
bySlaveNum	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
strSlaveIPAddr	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√
wPort	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
xDone	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
xBusy	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
xError	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
ErrorCode	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

■ Function

This function block configures the slave IP address in Modbus TCP master mode.

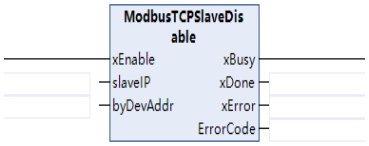
■ Program example

	LD	ST
Defined variable	 VAR ModbusTCPSlaveSetIPAddr_0: ModbusTCPSlaveSetIPAddr; END_VAR	
Program		ModbusTCPSlaveSetIPAddr_0(xExecute:= TRUE, bySlaveNum:= 0, strSlaveIPAddr:= '192.168.1.88', wPort:= 502, xBusy=> , xDone=> , xError=> , ErrorCode=>);

6.9. 5 ModbusTCPSlaveDisable

This instruction disables the Modbus TCP slave.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
ModbusTCPSlaveDisable	Modbus TCP slave disable	FB		ModbusTCPSlaveDisable_0(xEnable:= , slaveIP:= , byDevAddr:= , xBusy=> , xDone=> , xError=> , ErrorCode=>);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
xEnable	Function block enable	BOOL	[FALSE, TRUE]	FALSE	Function block enable
slaveIP	Slave IP address	STRING	0.0.0.0 to 255.255.255.255	192.168.1.88	Slave IP address
byDevAddr	Slave address	BYTE	1 to 255	255	Slave address

■ Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
xBusy	Execution flag	BOOL	[FALSE, TRUE]	-	Function block being executed
xDone	Completion flag	BOOL	[FALSE, TRUE]	-	Output valid flag of the function block
xError	Error flag	BOOL	[FALSE, TRUE]	-	TRUE: An internal error occurs in the function block
ErrorCode	Error ID	Modbus_ERROR	Depends on data types	-	Error code

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
xEnable	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
slaveIP	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√
byDevAddr	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
xBusy	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
xDone	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

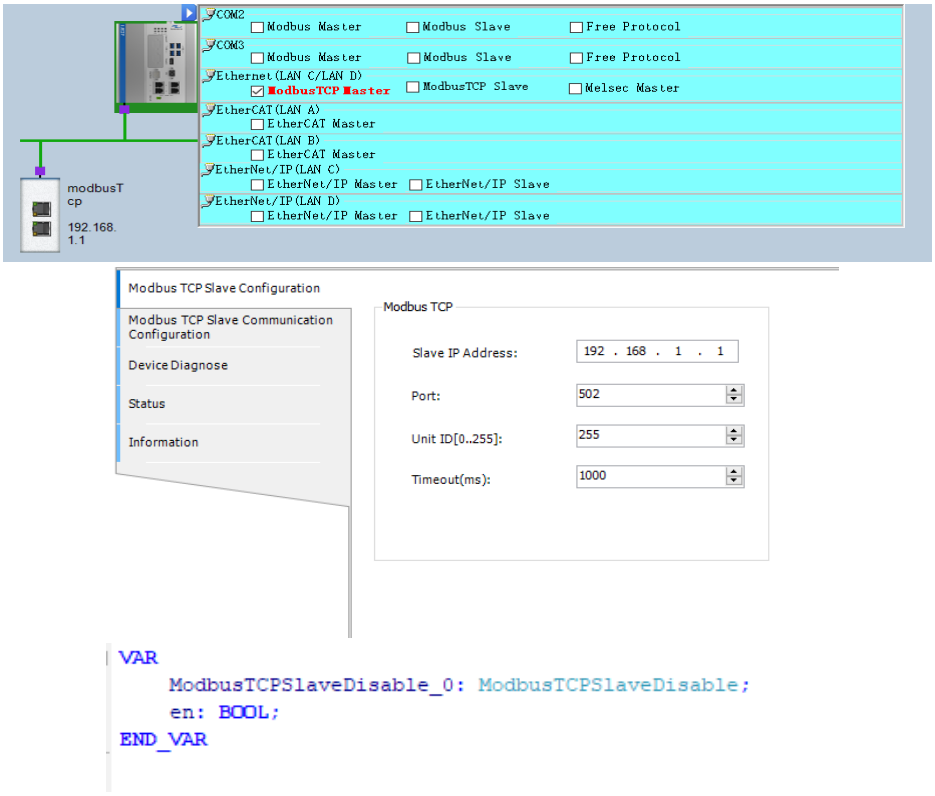
	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
xError	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
ErrorCode	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

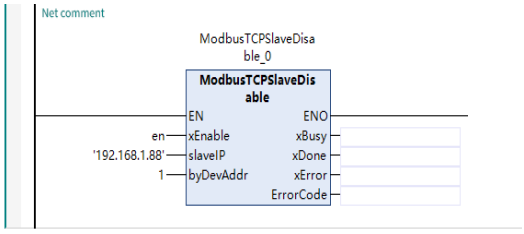
■ Function

- This function block disables the slave in Modbus TCP master mode.
- The instruction is not supported for PLCs of the AM600 series.

■ Program example

	LD	ST

Defined variable	
	`VAR`
	`  ModbusTCPSlaveDisable_0: ModbusTCPSlaveDisable;`
	`  en: BOOL;`
	`END_VAR`	

Program		`ModbusTCPSlaveDisable_0(`
		`  xEnable:= en,`
		`  slaveIP:= '192.168.1.88',`
		`  byDevAddr:= 1,`
		`  xBusy=> ,`
		`  xDone=> ,`
		`  xError=> ,`
		`  ErrorCode=> );`

6.9. 6 sysHC_ModbusTcpDeviceDiagnose

This instruction gets diagnosis information of Modbus TCP slave device faults by using the external library.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
sysHC_ ModbusTcpDeviceDiagnose	Get diagnostic information of ModbusTcp slave devices	FB	SysHC_ModbusTcpDeviceDiagnose_0 SysHC_ModbusTcpDeviceDiagnose EN — ENO xEnable — xValid xBusy xError eErrorID byDiagData	SysHC_ ModbusTcpDeviceDiagnose(xEnable:= , xValid=> , xBusy=> , xError=> , eErrorID=> , byDiagData=>);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
xEnable	Function block enable	BOOL	-	-	Input high level to enable the function block

Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
xValid	Valid flag	BOOL	-	-	Output active
xBusy	Busy flag	BOOL	-	-	Executing the instruction
xError	Error flag	BOOL	-	-	Error flag
eErrorID	Error code	SYS_HC_ERROR	-	-	Error code
byDiagData	Diagnostic data	BYTE	-	-	Diagnostic data

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
xEnable	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
xValid	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
xBusy	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
xError	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
eErrorID	SYS_HC_ERROR																			
byDiagData	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

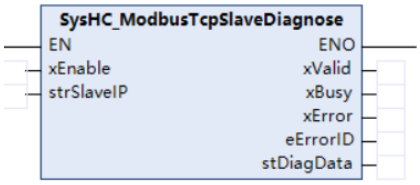
■ Program example

For details, see *"6.7. 12 Examples of Modbus Communication Instructions" on page 930*.

6.9. 7 sysHC_ModbusTcpSlaveDiagnose

This instruction gets diagnosis information of slave faults accessed from the Modbus TCP master by using the external library.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
sysHC_ ModbusTcpSlaveDiagnose	Get diagnostic data of ModbusTcp master accessing slaves	FB		SysHC_ ModbusTcpSlaveDiagnose(xEnable:= , strSlaveIP:= , xValid=> , xBusy=> , xError=> , eErrorID=> , stDiagData=>);

■ Variables

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
xEnable	Function block enable	BOOL	-	-	Input high level to enable the function block
strSlaveIP	Slave IP address	STRING[16]	-	-	Remote slave IP address

Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
xValid	Valid flag	BOOL	-	-	Output active
xBusy	Busy flag	BOOL	-	-	Executing the instruction
xError	Error flag	BOOL	-	-	Error flag
eErrorID	Error code	SYS_HC_ERROR	-	-	Error code
stDiagData	Diagnostic data	Modbus_SLAVE_DIAGNOSE	-	-	Slave diagnostic data

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
xEnable	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
strSlaveIP	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√
xValid	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
xBusy	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
xError	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
eErrorID	SYS_HC_ERROR																			
stDiagData	Modbus_SLAVE_DIAGNOSE																			

■ Program example

For details, see *"6.7. 12 Examples of Modbus Communication Instructions" on page 930*.

Chapter 7 CANopen Axis Control Instructions

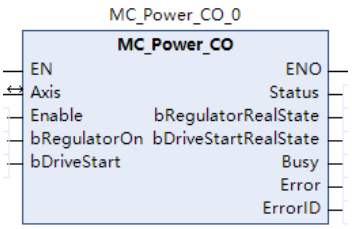
7. 1 Instruction List

Instruction Category	Name	FB/FC	Function
CANopen axis control instructions	MC_Power_CO	FB	Enable axis
	MC_MoveAbsolute_CO	FB	Move absolute position
	MC_MoveRelative_CO	FB	Move relative position
	MC_MoveVelocity_CO	FB	Move in specified speed
	MC_Home_CO	FB	Move homing
	MC_Stop_CO	FB	Stop motion
	MC_Halt_CO	FB	Stop motion and motion can be interrupted
	MC_Reset_CO	FB	Reset status
	MC_WriteParameter_CO	FB	Write specific axis parameters
	MC_ReadParameter_CO	FB	Read specific axis parameters
	MC_ReadStatus_CO	FB	Read motion status
	MC_JOG_CO	FB	Stop motion and motion can be interrupted

7. 2 MC_Power_CO

This instruction enables a motor.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
MC_Power_CO	Enable axis	FB		MC_Power_CO(Axis:= , Enable:= , bRegulatorOn:= , bDriveStart:= , Status=> , bRegulatorRealState=> , bDriveStartRealState=> , Busy=> , Error=> , ErrorID=>);

■ Variables

In-out variables

In-Out Variable	Name	Data Type	Value Range	Initial Value	Description
Axis	Axes	AXIS_REF_HC_CO	-	-	CANopen axis

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
Enable	Enable	BOOL	[FALSE, TRUE]	FALSE	Triggered by level TRUE: Enable the function block
bRegulatorOn	Use status	BOOL	[FALSE, TRUE]	FALSE	TRUE: Enable the axis
bDriveStart	Drive enable	BOOL	[FALSE, TRUE]	FALSE	TRUE: Emergency stop is not supported for the function block

Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
Status	Axis motion preparation status	BOOL	[FALSE, TRUE]	FALSE	Axis motion preparation status
bRegulatorRealState	Enable valid state	BOOL	[FALSE, TRUE]	FALSE	Indicates whether the drive is enabled TRUE: Enabled
bDriveStartRealState	Valid state of quick stop mechanism	BOOL	[FALSE, TRUE]	FALSE	Emergency stop not supported currently
Busy	Execution flag	BOOL	[FALSE, TRUE]	FALSE	TRUE: The function block is being executed
Error	Error flag	BOOL	[FALSE, TRUE]	FALSE	TRUE: An internal error occurs in the function block
ErrorID	Error ID	ERROR_CO	-	NO_ERROR	Error ID For detail, see ERROR_CO.

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Axis	AXIS_REF_HC_CO axis variables																			
Enable	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
bRegulatorOn	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
bDriveStart	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Status	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
bRegulatorRealState	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
bDriveStartRealState	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Busy	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Error	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
ErrorID	ERROR_CO enumerators																			

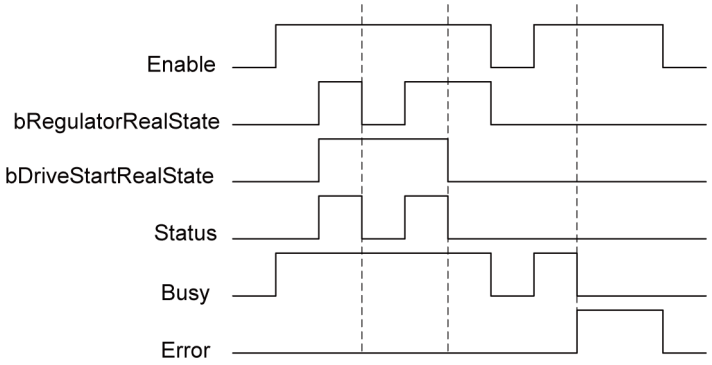
■ Function

This function block is mainly used for axis enabling/disabling and quick emergency stop. Use of the quick emergency stop function depends on whether the drive supports the function.

If bRegulatorRealState is TRUE, the drive is enabled but not necessarily controllable. Motion control can be implemented only when Status is TRUE.

The quick stop mode depends on the value of the object dictionary 16#605A. To change the stop mode, set the 16#605A parameter, depending on whether the drive supports the function.

■ Timing diagram



■ Error description

When an error occurs, locate the error specified by ErrorID in COUNTER_ERROR according to the help document, and find the reason for the error.

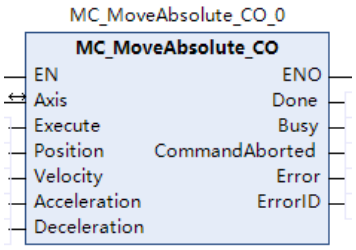
Error ID	Enumerator	Description
0	NO_ERROR	No error
1	DI_GENERAL_COMMUNICATION_ERROR	Axis communication error
2	DI_AXIS_ERROR	Drive axis error
21	WRONG_OPMODE	Incorrect operation mode
33	AXIS_IN_ERRORSTOP	Axis in ErrorStop state
34	AXIS_NOT_READY_FOR_MOTION	Axis not ready for motion
35	MA_MR_MODULO_ACT_POS_NOT_MAPPED	Reserved
36	MV_INVALID_VELACCDEC_VALUES	Invalid speed or acceleration/deceleration
80	RAG_ERROR_DURING_STARTUP	Reserved
81	RAG_ERROR_WRITING_COMSTATE	Reserved
82	RAG_ERROR_READING_COMSTATE	Reserved
90	CGR_ZERO_VALUES	Reserved
91	CGR_AXIS_POWERED	Reserved
93	CGR_MODULOPERIOD_NOT_INTEGRAL	Reserved
94	CGR_MOVEMENTTYPE_INVALID	Reserved
95	CGR_MODULOPERIOD_NON_POSITIVE	Reserved
96	CGR_MODULOPERIOD_TOO_SMALL	Reserved
97	CGR_MODULOPERIOD_TOO_LARGE	Reserved
120	R_NO_ERROR_TO_RESET	No error to reset
121	R_DRIVE_DOESNT_ANSWER	No answer
122	R_ERROR_NOT_RESETTABLE	Error not resettable
123	R_DRIVE_DOESNT_ANSWER_IN_TIME	Reserved
130	RP_PARAM_UNKNOWN	Parameter read error
131	RP_REQUESTING_ERROR	Communication request error
132	RP_RCV_PARAM_CONVERSION_ERROR	Reserved
133	RP_LOCAL_PARAM_NOT_DONE_IMMEDIATELY	Reserved

Error ID	Enumerator	Description
134	RP_CANNOT_SEND_MSG	Unable to send messages
140	WP_PARAM_INVALID	Parameter write error
141	WP_SENDING_ERROR	Sending error
142	WP_TMT_PARAM_CONVERSION_ERROR	Reserved
143	WP_LOCAL_PARAM_NOT_DONE_IMMEDIATELY	Reserved
144	WP_CANNOT_SEND_MSG	Unable to send messages
170	H_AXIS_WASNT_STANDSTILL	Axis in StandStill state for homing
183	MS_AXIS_IN_ERRORSTOP	ErrorStop state not allowed for the mc_stop function block
184	MS_AXIS_IN_STOPPING	Stopping state not allowed for the mc_stop function block
10000	TIMEOUT_CHANGING_OPMODE	Mode switching timeout
10001	INTERNAL_UNKNOWN_CMD	Reserved
10003	CANNOT_START_MOVEMENT	Reserved
10004	CANNOT_START_HOMING	Reserved
10005	STOP_ALREADY_ACTIVE	Reserved
10006	POWER_ALREADY_ACTIVE	Reserved
10007	SMC_DI_VOLTAGE_DISABLED	Axis motion process interrupt enabled

7.3 MC_MoveAbsolute_CO

This instruction moves an axis to an absolute position.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
MC_MoveAbsolute_CO	Move absolute position	FB		MC_MoveAbsolute_CO(Axis:= , Execute:= , Position:= , Velocity:= , Acceleration:= , Deceleration:= , Done=> , Busy=> , CommandAborted=> , Error=> , ErrorID=>);

■ Variables

In-out variables

In-Out Variable	Name	Data Type	Value Range	Initial Value	Description
Axis	Axes	AXIS_REF_HC_CO	-	-	CANopen axis

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
Execute	Execution input	BOOL	[FALSE, TRUE]	FALSE	(Triggered by rising edges) FALSE to TRUE: Execution starts
Position	Target position	LREAL	(0, 1.7E 308)	0.0	Unit: unit, only linear mode supported
Velocity	Target velocity	LREAL	(0, 1.7E 308)	0.0	Unit: unit/s, only linear mode supported
Acceleration	Target acceleration	LREAL	(0, 1.7E 308)	0.0	Unit: unit/s☒, only linear mode supported
Deceleration	Target acceleration	LREAL	(0, 1.7E 308)	0.0	Unit: unit/s☒, only linear mode supported

Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
Done	Completion flag	BOOL	[FALSE, TRUE]	FALSE	TRUE: The motion execution is completed, and positioning succeeds
Busy	Execution flag	BOOL	[FALSE, TRUE]	FALSE	TRUE: Motion is being executed
CommandAborted	Abort flag	BOOL	[FALSE, TRUE]	FALSE	TRUE: Motion is interrupted
Error	Error flag	BOOL	[FALSE, TRUE]	FALSE	TRUE: An error occurs during motion
ErrorID	Error ID	ERROR_CO	-	NO_ERROR	Error ID For detail, see ERROR_CO.

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Axis	AXIS_REF_HC_CO axis variables																			
Execute	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Position	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–
Velocity	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–
Acceleration	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–
Deceleration	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–
Done	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Busy	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
CommandAborted	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Error	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
ErrorID	ERROR_CO enumerators																			

■ Function

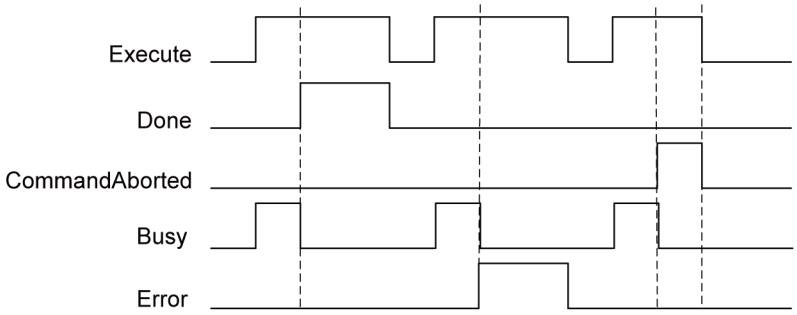
The controlled axis is moved to a specified absolute position through the absolute positioning function. The pulse count for absolute positioning ranges from －2147483648 to +2147483647. If the pulse count is out of range, an error occurs during positioning. The motion function block cannot run normally.

The values of Velocity, Acceleration, and Deceleration must be greater than 0.

If the positioning length is out of range (– 2147483648 to +2147483647), reduce the servo electronic gear ratio coefficient or lower the stepper drive.

When positioning is completed, Done is set to TRUE and .nAxisState of the controlled axis is StandStill.

■ Timing diagram



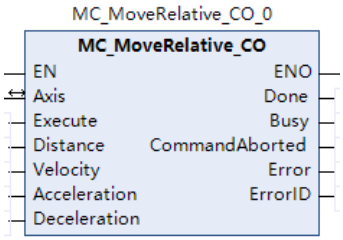
■ Error description

When an error occurs, locate the error specified by ErrorID in COUNTER_ERROR according to the help document, and find the reason for the error.

7.4 MC_MoveRelative_CO

This instruction moves an axis to a relative position.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
MC_MoveRelative_CO	Move relative position	FB		MC_MoveRelative_CO(Axis:= , Execute:= , Position:= , Velocity:= , Acceleration:= , Deceleration:= , Done=> , Busy=> , CommandAborted=> , Error=> , ErrorID=>);

■ Variables

In-out variables

In-Out Variable	Name	Data Type	Value Range	Initial Value	Description
Axis	Axes	AXIS_REF_HC_CO	-	-	CANopen axis

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
Execute	Execution input	BOOL	[FALSE, TRUE]	FALSE	(Triggered by rising edges) FALSE to TRUE: Execution starts
Position	Target position	LREAL	(0, 1.7E 308)	0.0	Unit: unit, only linear mode supported

Velocity	Target velocity	LREAL	(0, 1.7E 308)	0.0	Unit: unit/s, only linear mode supported
Acceleration	Target acceleration	LREAL	(0, 1.7E 308)	0.0	Unit: unit/s☒, only linear mode supported
Deceleration	Target acceleration	LREAL	(0, 1.7E 308)	0.0	Unit: unit/s☒, only linear mode supported

Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
Done	Completion flag	BOOL	[FALSE, TRUE]	FALSE	TRUE: The motion execution is completed, and positioning succeeds
Busy	Execution flag	BOOL	[FALSE, TRUE]	FALSE	TRUE: Motion is being executed
CommandAborted	Abort flag	BOOL	[FALSE, TRUE]	FALSE	TRUE: Motion is interrupted
Error	Error flag	BOOL	[FALSE, TRUE]	FALSE	TRUE: An error occurs during motion
ErrorID	Error ID	ERROR_CO	-	NO_ERROR	Error ID For detail, see ERROR_CO.

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Axis	AXIS_REF_HC_CO axis variables																			
Execute	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Position	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–
Velocity	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–
Acceleration	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–
Deceleration	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–
Done	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Busy	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
CommandAborted	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Error	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
ErrorID	ERROR_CO enumerators																			

■ Function

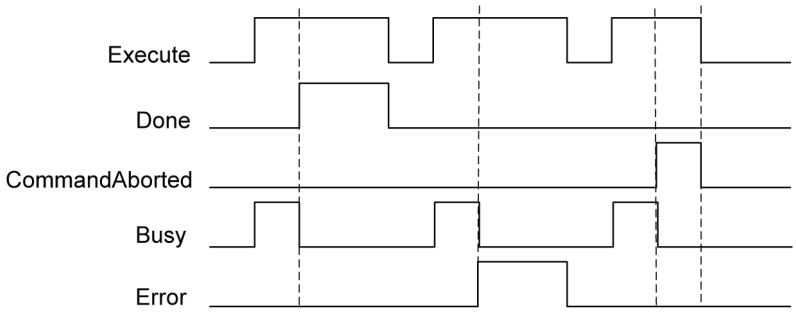
The relative positioning function block moves the controlled axis a distance (Motion target position = Standstill position when the function block is executed + Relative distance). If the relation positioning instruction is started while the axis is moving under the absolute positioning instruction, the motion target position is the absolutely defined target position plus relative distance.

If the positioning length is out of range (− 2147483648 to +2147483647), reduce the servo electronic gear ratio coefficient or lower the stepper drive.

When the device is used with Inovance IS620_CO, if the relative positioning command or absolute positioning command being executed is interrupted by the relative motion command, the absolute target position calculated through a relative command is the position the axis moving relatively or absolutely should reach plus the relative position for the motion instruction.

When positioning is completed, Done is set to TRUE and .nAxisState of the controlled axis is StandStill.

■ Timing diagram



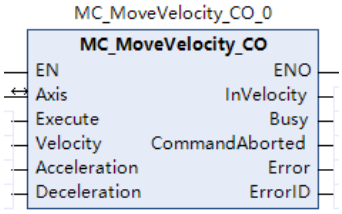
■ Error description

When an error occurs, locate the error specified by ErrorID in COUNTER_ERROR according to the help document, and find the reason for the error.

7.5 MC_MoveVelocity_CO

This instruction moves an axis in a specified speed.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
MC_MoveVelocity_CO	Move in specified speed	FB		MC_MoveRelative_CO(Axis:= , Execute:= , Velocity:= , Acceleration:= , Deceleration:= , InVelocity=> , Busy=> , CommandAborted=> , Error=> , ErrorID=>);

■ Variables

In-out variables

In-Out Variable	Name	Data Type	Value Range	Initial Value	Description
Axis	Axes	AXIS_REF_HC_CO	-	-	CANopen axis

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
Execute	Execution input	BOOL	[FALSE, TRUE]	FALSE	(Triggered by rising edges) FALSE to TRUE: Execution starts
Position	Target position	LREAL	(0, 1.7E 308)	0.0	Unit: unit, only linear mode supported
Velocity	Target velocity	LREAL	(0, 1.7E 308)	0.0	Unit: unit/s, only linear mode supported
Acceleration	Target acceleration	LREAL	(0, 1.7E 308)	0.0	Unit: unit/s☒, only linear mode supported

Deceleration	Target acceleration	LREAL	(0, 1.7E 308)	0.0	Unit: unit/s☒, only linear mode supported

Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
InVelocity	Velocity reached flag	BOOL	[FALSE, TRUE]	FALSE	TRUE: The axis speed reaches the preset value
Busy	Execution flag	BOOL	[FALSE, TRUE]	FALSE	TRUE: Motion is being executed
CommandAborted	Abort flag	BOOL	[FALSE, TRUE]	FALSE	TRUE: Motion is interrupted
Error	Error flag	BOOL	[FALSE, TRUE]	FALSE	TRUE: An error occurs during motion
ErrorID	Error ID	ERROR_CO	-	NO_ERROR	Error ID For detail, see ERROR_CO.

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Axis	AXIS_REF_HC_CO axis variables																			
Execute	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Velocity	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–
Acceleration	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–
Deceleration	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–
InVelocity	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Busy	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
CommandAborted	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Error	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
ErrorID	ERROR_CO enumerators																			

■ Function

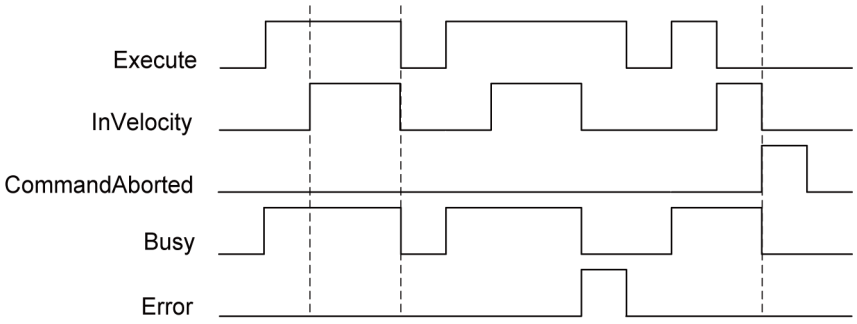
On the function block for motion in velocity mode, after the speed reaches the preset value, the controlled axis runs at the speed constantly. To stop the axis, interrupt and abort the current function block by using another command. When the function block is interrupted by another command, the output parameter InVelocity must be reset.

The position varies in the range of －2147483648 to +2147483647.

When the device is used with IS620N_CO, if the position overflows (the pulse position changes from +2147483647 to －2147483648 or from －2147483648 to +2147483647) in motion mode, return the axis to home, switch to the position mode, and then use the position motion command because the overflow count is not memorized in position mode (different from the synchronization cycle mode).

During normal motion, .nAxisState of the controlled axis is ContinuesMotion.

■ Timing diagram



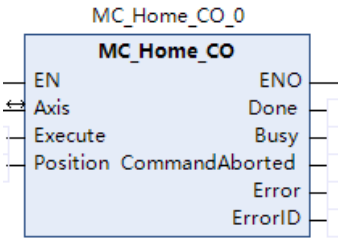
■ Error description

When an error occurs, locate the error specified by ErrorID in COUNTER_ERROR according to the help document, and find the reason for the error.

7.6 MC_Home_CO

This instruction moves an axis to home.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
MC_Home_CO	Move homing	FB		MC_Home_CO(Axis:= , Execute:= , Position:= , Done=> , Busy=> , CommandAborted=> , Error=> , ErrorID=>);

■ Variables

In-out variables

In-Out Variable	Name	Data Type	Value Range	Initial Value	Description
Axis	Axes	AXIS_REF_HC_CO	-	-	CANopen axis

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
Execute	Execution input	BOOL	[FALSE, TRUE]	FALSE	(Triggered by rising edges) FALSE to TRUE: Execution starts
Position	Home offset	LREAL	(0, 1.7E 308)	0.0	Unit: unit/s, only linear mode supported

Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
Done	Completion flag	BOOL	[FALSE, TRUE]	FALSE	TRUE: Homing is completed
Busy	Execution flag	BOOL	[FALSE, TRUE]	FALSE	TRUE: Motion is being executed

Output Variable	Name	Data Type	Value Range	Initial Value	Description
CommandAborted	Abort flag	BOOL	[FALSE, TRUE]	FALSE	TRUE: Motion is interrupted
Error	Error flag	BOOL	[FALSE, TRUE]	FALSE	TRUE: An error occurs during motion
ErrorID	Error ID	ERROR_CO	-	NO_ERROR	Error ID For detail, see ERROR_CO.

	Bool-ean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Axis	AXIS_REF_HC_CO axis variables																			
Execute	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Position	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–
Done	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Busy	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
CommandAborted	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Error	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
ErrorID	ERROR_CO enumerators																			

■ Function

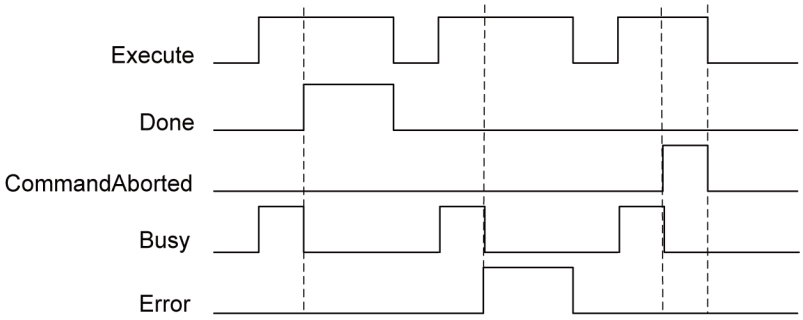
On the homing function block, the controlled axis returns to home in preset mode. The axis stops when reference signals are detected. The current absolute position is updated to the value of the input parameter Position.

Homing cannot be interrupted by the positioning command or speed command.

During normal motion, .nAxisState of the controlled axis is Homing.

The position home offset is the distance the axis deviates from the absolute position of home (unit: unit). When the hardware model is detected and homing stops, the current absolute position is set to Position, and .nAxisState of the controlled axis is StandStill.

■ Timing diagram



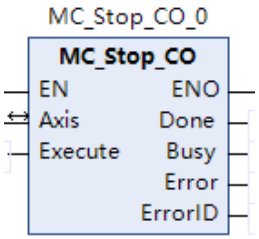
■ Error description

When an error occurs, locate the error specified by ErrorID in COUNTER_ERROR according to the help document, and find the reason for the error.

7.7 MC_Stop_CO

This instruction stops motion of an axis.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
MC_Stop_CO	Stop motion	FB		MC_Stop_CO(Axis:= , Execute:= , Done=> , Busy=> , Error=> , ErrorID=>);

■ Variables

In-out variables

In-Out Variable	Name	Data Type	Value Range	Initial Value	Description
Axis	Axes	AXIS_REF_HC_CO	-	-	CANopen axis

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
Execute	Execution input	BOOL	[FALSE, TRUE]	FALSE	(Triggered by rising edges) FALSE to TRUE: Execution starts
Position	Home offset	LREAL	(0, 1.7E 308)	0.0	Unit: unit/s, only linear mode supported

Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
Done	Completion flag	BOOL	[FALSE, TRUE]	FALSE	TRUE: The axis stops
Busy	Execution flag	BOOL	[FALSE, TRUE]	FALSE	TRUE: Motion is being executed
Error	Error flag	BOOL	[FALSE, TRUE]	FALSE	TRUE: An error occurs during motion
ErrorID	Error ID	ERROR_CO	-	NO_ERROR	Error ID For detail, see ERROR_CO.

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Axis	AXIS_REF_HC_CO axis variables																			
Execute	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Done	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Busy	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Error	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
ErrorID	ERROR_CO enumerators																			

■ Function

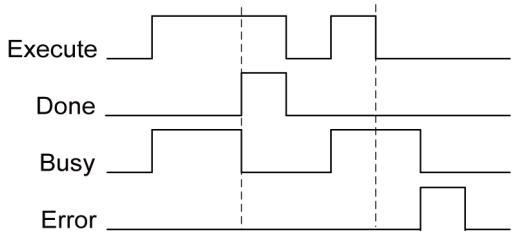
On the motion stop function block, after execution is triggered by rising edges, the controlled axis stops. If

Execute is TRUE or the axis is not stopped, no other motion commands can be sent, and the axis is in Stopping state. After the axis is stopped and Execute is reset, the axis is in the StandStill state.

As many manufacturers do not support execution stop through the control word bit 8, the stop function switches the mode to the velocity mode, and the target velocity is set to 0 to stop motion by running velocity commands.

After motion stops, reset the Execute state. Otherwise, other motion commands cannot be run.

Timing diagram



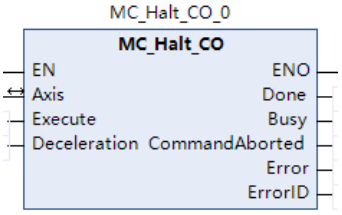
■ Error description

When an error occurs, locate the error specified by ErrorID in COUNTER_ERROR according to the help document, and find the reason for the error.

7.8 MC_Halt_CO

This instruction stops motion of an axis.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
MC_Halt_CO	Stop motion and motion can be interrupted	FB		MC_ Halt _CO(Axis:= , Execute:= , Deceleration:= , Done=> , Busy=> , CommandAborted=> , Error=> , ErrorID=>);

■ Variables

In-out variables

In-Out Variable	Name	Data Type	Value Range	Initial Value	Description
Axis	Axes	AXIS_REF_HC_CO	-	-	CANopen axis

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
Execute	Execution input	BOOL	[FALSE, TRUE]	FALSE	(Triggered by rising edges) FALSE to TRUE: Execution starts
Deceleration	Deceleration	LREAL	(0, 1.7E 308)	0.0	Unit: unit/s☒, only linear mode supported

Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
Done	Completion flag	BOOL	[FALSE, TRUE]	FALSE	TRUE: The axis stops
Busy	Execution flag	BOOL	[FALSE, TRUE]	FALSE	TRUE: Motion is being executed
CommandAborted	Abort flag	BOOL	[FALSE, TRUE]	FALSE	TRUE: Motion is interrupted
Error	Error flag	BOOL	[FALSE, TRUE]	FALSE	TRUE: An error occurs during motion
ErrorID	Error ID	ERROR_CO	-	NO_ERROR	Error ID For detail, see ERROR_CO.

	Bool-ean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Axis	AXIS_REF_HC_CO axis variables																			
Execute	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Deceleration	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–
Done	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Busy	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
CommandAborted	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Error	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
ErrorID	ERROR_CO enumerators																			

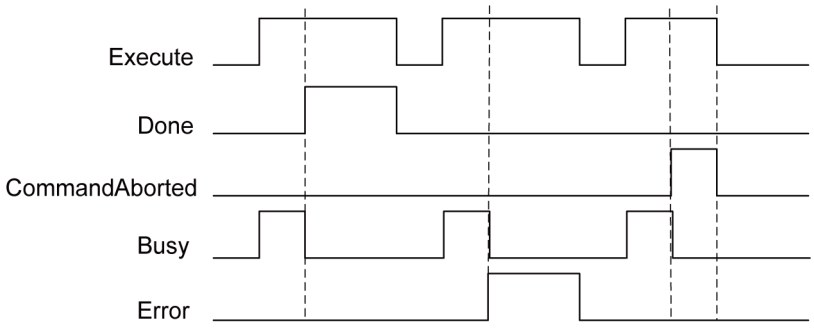
■ Function

When the axis is stopped through commands, the axis is in DiscreteMotion state until the velocity is zero. When Done is TRUE, the axis is in the StandStill state.

During the MC_Halt_CO execution stop, MC_Halt_CO can be interrupted by other motion commands.

After the MC_Halt_CO execution is completed, unlike MC_Stop_CO, other motion commands can be used without reset of the Execute port state.

■ Timing diagram



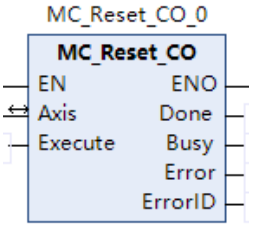
■ Error description

When an error occurs, locate the error specified by ErrorID in COUNTER_ERROR according to the help document, and find the reason for the error.

7.9 MC_Reset_CO

This instruction resets errors.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
MC_Reset_CO	Reset status	FB		MC_ Halt _CO(Axis:= , Execute:= , Done=> , Busy=> , Error=> , ErrorID=>);

■ Variables

In-out variables

In-Out Variable	Name	Data Type	Value Range	Initial Value	Description
Axis	Axes	AXIS_REF_HC_CO	-	-	CANopen axis

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
Execute	Execution input	BOOL	[FALSE, TRUE]	FALSE	(Triggered by rising edges) FALSE to TRUE: Execution starts

Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
Done	Completion flag	BOOL	[FALSE, TRUE]	FALSE	TRUE: The axis stops
Busy	Execution flag	BOOL	[FALSE, TRUE]	FALSE	TRUE: Motion is being executed
Error	Error flag	BOOL	[FALSE, TRUE]	FALSE	TRUE: An error occurs during motion
ErrorID	Error ID	ERROR_CO	-	NO_ERROR	Error ID For detail, see ERROR_CO.

	Bool-ean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Axis	AXIS_REF_HC_CO axis variables																			
Execute	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Done	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Busy	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Error	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
ErrorID	ERROR_CO enumerators																			

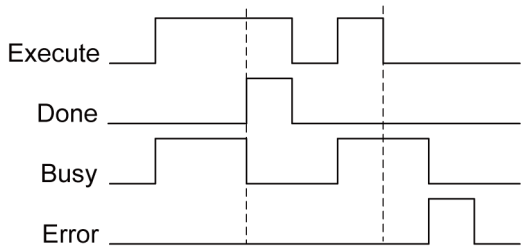
■ Function

Reset all errors relating to internal axes, and forcibly switch axes from the ErrorStop state to the StandStill or PowerOff state. If you perform the reset action when the controlled axis is not in the ErrorStop state, the

function block reports an error.

Motion errors and errors for which the servo supports the reset action can be reset.

■ Timing diagram



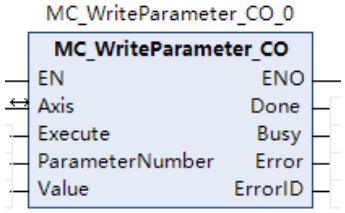
■ Error description

When an error occurs, locate the error specified by ErrorID in COUNTER_ERROR according to the help document, and find the reason for the error.

7. 10 MC_WriteParameter_CO

This instruction writes remote device parameters.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
MC_WriteParameter_CO	Write specific axis parameters	FB		MC_ WriteParameter _CO(Axis:= , Execute:= , ParameterNumber:= , Value:= , Done=> , Busy=> , Error=> , ErrorID=>);

■ Variables

In-out variables

In-Out Variable	Name	Data Type	Value Range	Initial Value	Description
Axis	Axes	AXIS_REF_HC_CO	-	-	CANopen axis

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
Execute	Execution input	BOOL	[FALSE, TRUE]	FALSE	(Triggered by rising edges) FALSE to TRUE: Execution starts
ParameterNumber	Specifications	DINT	[0, 2^32)	0	Parameter number, which should be operated in combined mode
Value	Value	LREAL	(− 1.7E 308, 1.7E 308)	0.0	Input value

Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
Done	Completion flag	BOOL	[FALSE, TRUE]	FALSE	TRUE: The axis stops
Busy	Execution flag	BOOL	[FALSE, TRUE]	FALSE	TRUE: Motion is being executed
Error	Error flag	BOOL	[FALSE, TRUE]	FALSE	TRUE: An error occurs during motion
ErrorID	Error ID	ERROR_CO	-	NO_ERROR	Error ID For detail, see ERROR_CO.

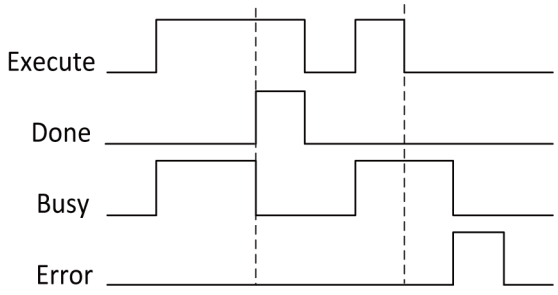
	Bool-ean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Axis	AXIS_REF_HC_CO axis variables																			
Execute	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
ParameterNumber	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–
Value	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–
Done	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Busy	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Error	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
ErrorID	ERROR_CO enumerators																			

■ Function

Write object dictionary data: Based on CANopen parameters, the specific axis parameter number consists of the object dictionary length 16#UV (1 byte), object dictionary index 16#ABCD (2 bytes), and sub-index 16#EF (1 byte), which form the -16#UVABCDEF mode.

Example: If the write object index value is 16#607C, the sub-index value is 16#00, and the data length value is 16#04, the parameter number is -16#04607C00.

■ Timing diagram



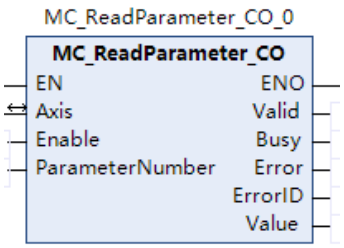
■ Error description

When an error occurs, locate the error specified by ErrorID in COUNTER_ERROR according to the help document, and find the reason for the error.

7. 11 MC_ReadParameter_CO

This instruction reads remote device parameters.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
MC_ReadParameter_CO	Read specific axis parameters	FB		MC_ReadParameter_CO(Axis:= , Execute:= , ParameterNumber:= , Valid=> , Busy=> , Error=> , ErrorID=> , Value =>);

■ Variables

In-out variables

In-Out Variable	Name	Data Type	Value Range	Initial Value	Description
Axis	Axes	AXIS_REF_HC_CO	-	-	CANopen axis

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
Execute	Execution input	BOOL	[FALSE, TRUE]	FALSE	(Triggered by rising edges) FALSE to TRUE: Execution starts
ParameterNumber	Specifications	DINT	[0, 2^32)	0	Parameter number, which should be operated in combined mode

Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
Done	Completion flag	BOOL	[FALSE, TRUE]	FALSE	TRUE: The axis stops
Busy	Execution flag	BOOL	[FALSE, TRUE]	FALSE	TRUE: Motion is being executed
Error	Error flag	BOOL	[FALSE, TRUE]	FALSE	TRUE: An error occurs during motion
ErrorID	Error ID	ERROR_CO	-	NO_ERROR	Error ID For detail, see ERROR_CO.
Value	Value	LREAL	(– 1.7E 308, 1.7E 308)	0.0	Read value

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Axis	AXIS_REF_HC_CO axis variables																			
Execute	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
ParameterNumber	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–
Value	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–

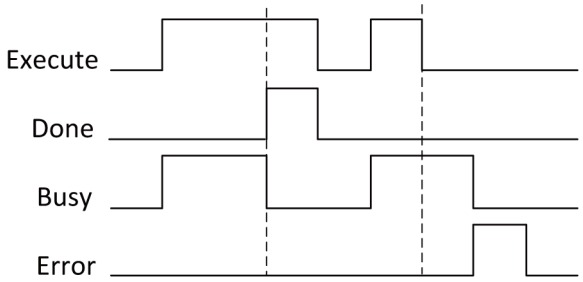
	Bool-ean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Done	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Busy	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Error	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
ErrorID	ERROR_CO enumerators																			

■ Function

Read object dictionary data: Based on CANopen parameters, the specific axis parameter number consists of the object dictionary length 16#UV (1 byte), object dictionary index 16#ABCD (2 bytes), and sub-index 16#EF (1 byte), which form the -16#UVABCDEF mode.

Example: If the read object index value is 16#607C, the sub-index value is 16#00, and the data length value is 16#04, the parameter number is -16#04607C00.

■ Timing diagram



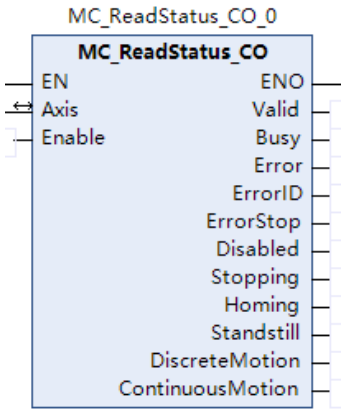
■ Error description

When an error occurs, locate the error specified by ErrorID in COUNTER_ERROR according to the help document, and find the reason for the error.

7.12 MC_ReadStatus_CO

This instruction reads the motion status of the current axis.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
MC_ReadStatus_CO	Read motion status	FB		MC_ReadStatus_CO(Axis:= , Enable:= , Valid=> , Busy=> , Error=> , ErrorID=> , ErrorStop=> , Disabled=> , Stopping=> , Homing=> , Standstill=> , DiscreteMotion=> , ContinuousMotion=>);

■ Variables

In-out variables

In-Out Variable	Name	Data Type	Value Range	Initial Value	Description
Axis	Axes	AXIS_REF_HC_CO	-	-	CANopen axis

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
Execute	Execution input	BOOL	[FALSE, TRUE]	FALSE	(Triggered by rising edges) FALSE to TRUE: Execution starts

Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
Valid	Reading state valid flag	BOOL	[FALSE, TRUE]	FALSE	TRUE: Reset is completed
Busy	Execution flag	BOOL	[FALSE, TRUE]	FALSE	TRUE: The command is being executed
Error	Error flag	BOOL	[FALSE, TRUE]	FALSE	TRUE: An error occurs on the function block
ErrorID	Error ID	ERROR_CO	-	NO_ERROR	Error ID For detail, see ERROR_CO.
ErrorStop	Error stop	BOOL	[FALSE, TRUE]	FALSE	TRUE: The axis fails
Disabled	Disabled	BOOL	[FALSE, TRUE]	FALSE	TRUE: The axis is disabled
Stopping	Stopping	BOOL	[FALSE, TRUE]	FALSE	TRUE: The stop command is being executed
Homing	Homing	BOOL	[FALSE, TRUE]	FALSE	TRUE: The axis is homing
StandStill	Motor enable	BOOL	[FALSE, TRUE]	FALSE	TRUE: The axis is enabled
DiscreteMotion	Discrete motion	BOOL	[FALSE, TRUE]	FALSE	TRUE: The axis is moving in point-to-point mode

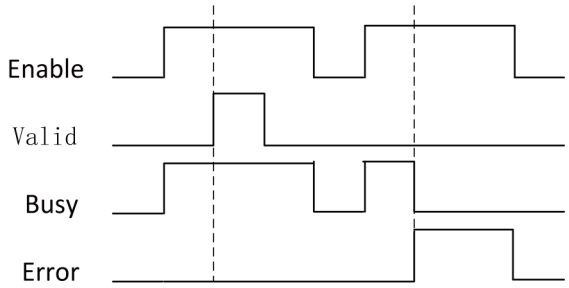
Output Variable	Name	Data Type	Value Range	Initial Value	Description
ContinuousMotion	Continuous motion	BOOL	[FALSE, TRUE]	FALSE	TRUE: The velocity command is being executed

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Axis	AXIS_REF_HC_CO axis variables																			
Execute	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Valid	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Busy	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Error	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
ErrorID	ERROR_CO enumerators																			
ErrorStop	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Disabled	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Stopping	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Homing	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
StandStill	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
DiscreteMotion	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
ContinuousMotion	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

■ Function

The motion status of the controlled axis is displayed as a Boolean variable based on the PLCopen state machine standard.

■ Timing diagram



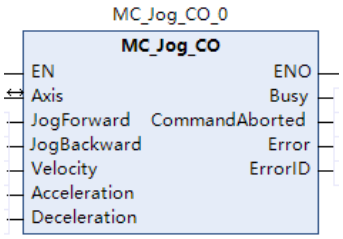
■ Error description

When an error occurs, locate the error specified by ErrorID in COUNTER_ERROR according to the help document, and find the reason for the error.

7.13 MC_Jog_CO

This instruction implements jogging motion of an axis.

■ Instruction format

Instruction	Name	FB/FC	LD Expression	ST Expression
MC_Halt_CO	Stop motion and motion can be interrupted	FB		MC_Jog_CO(Axis:= , JogForward:= , JogBackward:= , Velocity:= , Acceleration:= , Deceleration:= , Busy=> , CommandAborted=> , Error=> , ErrorID=>);

■ Variables

In-out variables

In-Out Variable	Name	Data Type	Value Range	Initial Value	Description
Axis	Axes	AXIS_REF_HC_CO	-	-	CANopen axis

Input variables

Input Variable	Name	Data Type	Value Range	Initial Value	Description
JogForward	Forward jog	BOOL	[FALSE, TRUE]	FALSE	Jog in the forward direction
JogBackward	Reverse jog	BOOL	[FALSE, TRUE]	FALSE	Jog in the reverse direction
Velocity	Speed	LREAL	(0, 1.7E 308)	0.0	Jog speed
Acceleration	Acceleration	LREAL	(0, 1.7E 308)	0.0	Acceleration
Deceleration	Deceleration	LREAL	(0, 1.7E 308)	0.0	Deceleration

Output variables

Output Variable	Name	Data Type	Value Range	Initial Value	Description
Busy	Execution flag	BOOL	[FALSE, TRUE]	FALSE	TRUE: Motion is being executed
CommandAborted	Abort flag	BOOL	[FALSE, TRUE]	FALSE	TRUE: Motion is interrupted
Error	Error flag	BOOL	[FALSE, TRUE]	FALSE	TRUE: An error occurs during motion
ErrorID	Error ID	ERROR_CO	-	NO_ERROR	Error ID For detail, see ERROR_CO.

	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Axis	AXIS_REF_HC_CO axis variables																			
Execute	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Deceleration	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–

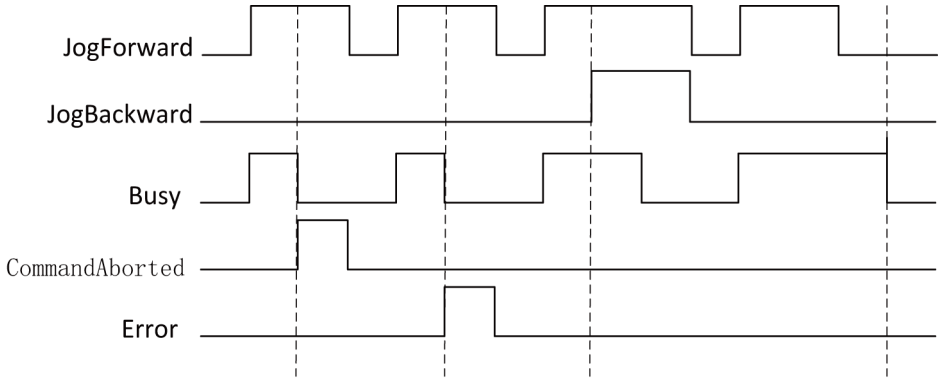
	Boolean	Bit String				Integer								Real Number		Time, Duration, Date, and Text String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Done	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Busy	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
CommandAborted	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Error	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
ErrorID	ERROR_CO enumerators																			

■ Function

The function block is used for axis jog control in both forward and reverse directions.

If JogBackward and JogForward are set to TRUE simultaneously, motion stops. The values of Velocity, Acceleration, and Deceleration must be greater than 0.

■ Timing diagram



■ Error description

When an error occurs, locate the error specified by ErrorID in COUNTER_ERROR according to the help document, and find the reason for the error.

Copyright © Shenzhen Inovance Technology Co., Ltd.

19012377A00

Shenzhen Inovance Technology Co., Ltd.

www.inovance.com

Add.: Inovance Headquarters Tower, High-tech Industrial Park,
Guanlan Street, Longhua New District, Shenzhen
Tel: (0755) 2979 9595 Fax: (0755) 2961 9897

Suzhou Inovance Technology Co., Ltd.

www.inovance.com

Add.: No. 16 Youxiang Road, Yuexi Town,
Wuzhong District, Suzhou 215104, P.R. China
Tel: (0512) 6637 6666 Fax: (0512) 6285 6720