# **XS** series PLCopen controller

## User manual [Basic Instruction] (XS Studio)

Wuxi XINJE Electric Co., Ltd.

Data No. PS05 20250307EN 1.0

## Basic Description

- Thank you for purchasing the Xinje XS series PLCopen standard controller.
- This manual mainly introduces the relevant usage of XS series PLCopen standard controller instructions.
- Before using the product, please read this manual carefully and program it on the premise of fully understanding the content of the manual.
- Please deliver this manual to the end user.

## User Information

- Only operators with a certain level of electrical knowledge can perform wiring and other operations on the product. If there are any areas where the use is unknown, please
- Consult our company's technical department.
- The examples listed in manuals and other technical materials are for user understanding and reference only, and do not guarantee certain actions.
- When using this product in combination with other products, please confirm whether it meets relevant specifications, principles, etc.
- When using this product, please confirm whether it meets the requirements and is safe.
- Please set up backup and safety functions on your own to avoid potential machine malfunctions or losses caused by the malfunction of this product.

## Declaration of Responsibility

- Although the content in the manual has been carefully checked, errors are inevitable and we cannot guarantee complete consistency.
- We will regularly check the content in the manual and make corrections in subsequent versions. We welcome your valuable feedback.
- The content described in the manual is subject to change without prior notice.

## Related manuals

Please refer to the following manual for hardware related and software applications of the XS series PLC.

- User Manual for XS Series PLCopen Standard Controller [Hardware]
- User Manual for XS Series PLCopen Standard Controller [Software]
- User Manual for XS Series PLCopen Standard Controller [Motion control]

WUXI XINJE ELECTRIC CO., LTD. Copy Rights Reserved

Without explicit written permission, this material and its contents shall not be copied, reproduced or used, and violators shall be liable for any losses caused. All rights provided in the patent license and registration, including utility modules or designs, are reserved.

Jan. 2023

# Catalog

**1. BASIC INSTRUCTIONS ..... 1**

1-1. BIT LOGIC INSTRUCTIONS ..... 1

*1-1-1. Basic bit logic instructions ..... 1*

1-1-1-1. Bit AND ..... 1

1-1-1-2. Boolean AND ..... 3

1-1-1-3. Bit OR ..... 4

1-1-1-4. Boolean OR ..... 5

1-1-1-5. Bit NOT ..... 6

1-1-1-6. Boolean NOT ..... 7

1-1-1-7. Bit XOR ..... 8

1-1-1-8. Boolean XOR ..... 9

*1-1-2. Bit logic operation instruction ..... 10*

1-1-2-1. Rising edge output PLS ..... 10

1-1-2-2. Falling edge output PLF ..... 11

1-1-2-3. Alternating output ALT ..... 12

1-1-2-4. Bit data output BOUT ..... 14

1-1-2-5. Set on bit BSET ..... 15

1-1-2-6. Reset bit BRST ..... 16

1-1-2-7. Logical AND/Logical OR/XOR/XNOR AryAnd/AryOr/AryXor/AryXorN ..... 18

*1-1-3. Set priority and reset priority trigger instructions ..... 21*

1-1-3-1. Set priority trigger SR ..... 21

1-1-3-2. Reset priority trigger RS ..... 22

*1-1-4. Edge detection instruction ..... 24*

1-1-4-1. Rising edge detection R_TRIG ..... 24

1-1-4-2. Falling edge detection F_TRIG ..... 25

1-2. TIMER ..... 27

*1-2-1. Pulse timer TP ..... 27*

*1-2-2. Power on delay timer TON ..... 28*

*1-2-3. Power off delay timer TOF ..... 30*

*1-2-4. RTC ..... 31*

*1-2-5. Accumulation timer ..... 33*

*1-2-6. 100ms Timer ..... 35*

1-3. COUNTER ..... 37

*1-3-1. Up counter CTU ..... 37*

*1-3-2. Down counter CTD ..... 39*

*1-3-3. Up/down counter CTUD ..... 40*

*1-3-4. Subtract counter CTD_XXX ..... 42*

*1-3-5. Up/Down Counter CTUD_XXX ..... 45*

*1-3-6. Up counter CTU_XXX ..... 49*

1-4. DATA PROCESSING INSTRUCTIONS ..... 52

*1-4-1. Selection commands ..... 52*

1-4-1-1. Binary choice instruction SEL ..... 52

1-4-1-2. Take the maximum value MAX ..... 53

1-4-1-3. Take the minimum value MIN ..... 54

1-4-1-4. Limit value LIMIT ..... 55
1-4-1-5. Multiple choice MUX ..... 57
*1-4-2. Compare Instructions* ..... *58*
1-4-2-1. Basic compare instruction EQ/NE/GT/GE/LT/LE ..... 58
1-4-2-2. Maximum/minimum value retrieval of arrays AryMax/AryMin ..... 60
1-4-2-3. Array retrieval ArySearch ..... 63
1-4-2-4. Batch comparison of arrays EQ AryCmpEQ ..... 66
1-4-2-5. Comparison of array elements EQV AryCmpEQV ..... 68
1-4-2-6. Batch comparison of arrays GE AryCmpGE ..... 70
1-4-2-7. Comparison of array elements GEV AryCmpGEV ..... 72
1-4-2-8. Batch comparison of arrays GT AryCmpGT ..... 74
1-4-2-9. Comparison of array elements GTV AryCmpGTV ..... 76
1-4-2-10. Batch comparison of arrays LE AryCmpLE ..... 78
1-4-2-11. Comparison of array elements LEV AryCmpLEV ..... 80
1-4-2-12. Batch comparison of arrays LT AryCmpLT ..... 82
1-4-2-13. Comparison of array elements LTV AryCmpLTV ..... 84
1-4-2-14. Batch comparison of arrays NE AryCmpNE ..... 86
1-4-2-15. Comparison of array elements NEV AryCmpNEV ..... 88
1-4-2-16. Table comparison TableCmp ..... 90
1-4-2-17. Zone comparison ZoneCmp ..... 93
*1-4-3. Shift instruction* ..... *95*
1-4-3-1. Shift left SHL ..... 95
1-4-3-2. Shift right SHR ..... 96
1-4-3-3. Rotate left ROL ..... 98
1-4-3-4. Rotate right ROR ..... 99
1-4-3-5. Shift register AryShiftReg ..... 101
1-4-3-6. Left-right shift register AryShiftRegLR ..... 104
1-4-3-7. Loop left/right shift with carry RCL/RCR ..... 107
1-4-3-8. Shift N elements of an array left/right ArySHL/ArySHR ..... 111
1-4-3-9. Shift N elements of an array left/right NSHLC/NSHRC ..... 114
1-4-3-10. Copy bit data to the left/right SFTL/SFTR ..... 116
1-4-3-11. Copy word data to the left/right WSFL/WSFR ..... 125
1-4-3-12. First in, first out data read/write SFRD/SFWR ..... 128
*1-4-4. Move instructions* ..... *133*
1-4-4-1. Bulk transfer BMOV ..... 133
1-4-4-2. Obtain data specified bit status BON ..... 135
1-4-4-3. Data combination of byte unit BTOW ..... 136
1-4-4-4. One to many data transfer FMOV ..... 138
1-4-4-5. ON bit SUM ..... 140
1-4-4-6. Array transfer AryMove ..... 141
1-4-4-7. Digit transfer MoveDigit ..... 143
1-4-4-8. Multiple bits transfer TransBits ..... 145
1-4-4-9. Set block SetBlock ..... 147
1-4-4-10. Bit decoder Decoder ..... 149
1-4-4-11. Bit encoder Encoder ..... 151
1-4-4-12. Bit count BitCnt ..... 153

1-4-4-13. High low byte exchange SWAP ..... 155
1-4-4-14. Data exchange XCH ..... 156
1-4-4-15. Array data exchange AryExchange ..... 158
1-4-4-16. Data exchange Exchange ..... 161
*1-4-5. Data separation and combination instructions ..... 163*
1-4-5-1. Byte unit data separation Dispart8Bit ..... 163
1-4-5-2. 4-bit separate DispartDigit ..... 165
1-4-5-3. Mantissa of real numbers, exponential separation DispartReal ..... 167
1-4-5-4. Byte unit data combination group Unite8Bit_XXX ..... 168
1-4-5-5. 4-digit combination group UniteDigit_XXX ..... 170
1-4-5-6. Combine mantissa and exponents into real numbers UniteReal ..... 172
1-4-5-7. Byte unit data separation WTOB ..... 173
*1-4-6. Data read and write instructions ..... 175*
1-4-6-1. N bits write to group WriteNbit_XXX ..... 175
1-4-6-2. N bits read group ReadNbit_XXX ..... 177
*1-4-7. Data conversion instructions ..... 179*
1-4-7-1. BCD code and integer data conversion ..... 179
1-4-7-2. BOOL_TO_<TYPE> boolean type conversion data ..... 182
1-4-7-3. BYTE_TO_<TYPE> byte type conversion data ..... 184
1-4-7-4. < Integer data > _TO_<TYPE> integer type conversion instruction ..... 185
1-4-7-5. REAL_TO_<TYPE> Real type conversion instruction ..... 187
1-4-7-6. TIME_TO_<TYPE> time type conversion instruction ..... 189
1-4-7-7. DATE_TO_<TYPE> Date type conversion instruction ..... 190
1-4-7-8. DT_TO_<TYPE> Date time type conversion instruction ..... 192
1-4-7-9. TOD_TO_<TYPE> Time type conversion instruction ..... 193
1-4-7-10. STRING_TO_<TYPE> Character type conversion instruction ..... 194
1-4-7-11. Rounding TRUNC ..... 195
1-4-7-12. Array conver to string AryToString ..... 196
1-4-7-13. Date convert to string DateToString ..... 198
1-4-7-14. Date time conver to string DtToString ..... 199
1-4-7-15. Convert hexadecimal strings to integers HexStringToNum_XXX ..... 200
1-4-7-16. Convert integers to fixed length decimal/hexadecimal strings NumToDecString/NumToHexString ..... 202
1-4-7-17. String conver to array StringToAry ..... 205
1-4-7-18. Time convert to string TodToString ..... 207
1-4-7-19. Convert from byte array AryByteTo ..... 208
1-4-7-20. Batch conversion of arrays to BCD AryToBCD ..... 213
1-4-7-21. Batch convert arrays to bit strings AryToBin ..... 215
1-4-7-22. Bit string conver to Gray code BinToGray_XXX ..... 217
1-4-7-23. Bit train/bit line conver to array ColmToLine_XXX ..... 219
1-4-7-24. Copy bit mode CopyByteToNum/CopyWordToNum/CopyDwordToNum/ CopyLwordToNum ..... 221
1-4-7-25. Copy bit mode CopyDwordToReal/CopyLwordToLreal ..... 223
1-4-7-26. Copy bit mode CopyRealToDWord/CopyLrealToLword ..... 224
1-4-7-27. Copy bit mode CopyNumToXXX ..... 225
1-4-7-28. Copy bit mode CopyNumToReal/CopyNumToLreal ..... 227
1-4-7-29. Copy bit mode CopyRealToNum/CopyLrealToNum ..... 228
1-4-7-30. Gray code convert to bit string GrayToBin_XXX ..... 229

1-4-7-31. Gray code convert to angle GRAY ....231
1-4-7-32. Convert fixed decimal point number to decimal strings FixNumToString....235
1-4-7-33. String convert to fixed decimal point number StringToFixNum....236
1-4-7-34. Row/column conversion LineToColm ....239
1-4-7-35. Convert to byte array ToAryByte....241
*1-4-8. Initialization instruction* .... *244*
1-4-8-1. Initialization Clear ....244
*1-4-9. Regional control instructions*.... *246*
1-4-9-1. Dead zone control command BZAND_TAB ....246
1-4-9-2. Calculation of average data value MEAN_TAB....248
1-4-9-3. Slope command RAMP_TAB....250
1-4-9-4. Table coordinate acquisition instruction SCL_TAB....253
1-4-9-5. Table data sorting instruction SORT_TAB....255
1-4-9-6. Data calculation total instruction WSUM_TAB....257
1-4-9-7. Zone control instruction ZONE_TAB....259
1-4-9-8. Batch data reset instruction ZRST_TAB....261
*1-4-10. Stack and Table*.... *263*
1-4-10-1. First in, first out queue function block FIFO....263
1-4-10-2. First in, first out instruction StackFIFO ....268
1-4-10-3. Insert stacked data StackIns....270
1-4-10-4. Last in, first out instruction StackLIFO ....273
1-4-10-5. Save stacked data StackPush....275
1-4-10-6. Delete stacked data instruction StackDel....278
*1-4-11. Filter command*.... *280*
1-4-11-1. Limiting filter LimitingFilter ....280
1-4-11-2. Median filter MedianFilter....284
1-4-11-3. Arithmetic average filter ArithmeticAverageFilter....288
1-4-11-4. Recursive average filter RecursiveAverageFilter ....291
1-4-11-5. Median average filter MedianAverageFilter....294
1-4-11-6. Limiting average filter LimitingAverageFilter ....297
1-4-11-7. First-order lag filter FirstOrderLagFilter....301
1-4-11-8. Weighted recursive average filter WeightRecursiveAverageFilter ....304
1-4-11-9. Debounce filter DebounceFilter....307
1-4-11-10. Limiting and debounce filter LimitingDebounceFilter....310
1-4-11-11. UTIL_ERROR data introduction ....314
1-5. OPERATION INSTRUCTIONS.... 315
*1-5-1. Assignment instruction*.... *315*
1-5-1-1. Assignment instruction MOVE....315
*1-5-2. Arithmetic operation instructions* .... *316*
1-5-2-1. Addition operation ADD....316
1-5-2-2. Subtraction operation SUB ....318
1-5-2-3. Multiplication operation MUL....319
1-5-2-4. Division operation DIV....320
1-5-2-5. Remainder operation MOD....322
*1-5-3. Mathematical operation instructions*.... *323*
1-5-3-1. Absolute value ABS....323

1-5-3-2. Square root SQRT ..... 324
1-5-3-3. Exponent EXP ..... 325
1-5-3-4. Natural logarithm LN ..... 326
1-5-3-5. Logarithm with a base of 10 LOG ..... 327
1-5-3-6. Sine function SIN ..... 328
1-5-3-7. Cosine function COS ..... 329
1-5-3-8. Arccosine function ACOS ..... 330
1-5-3-9. Arcsin function ASIN ..... 331
1-5-3-10. Tangent function TAN ..... 332
1-5-3-11. Arctangent function ATAN ..... 333
1-5-3-12. Floating point angle/radian conversion RAD/DEG ..... 334
1-5-3-13. Array element addition AryAddV ..... 336
1-5-3-14. Calculation of array elements average value AryMean ..... 338
1-5-3-15. Standard deviation of array elements ArySD ..... 340
1-5-3-16. Subtraction of array elements ArySubV ..... 342
1-5-3-17. Real number check CheckReal ..... 344
1-5-3-18. Increase/decrease Inc/Dec ..... 346
1-5-3-19. Real remainder ModReal ..... 347
1-5-3-20. Long real remainder ModReal_LR ..... 349
1-5-3-21. Moving average MovingAverage ..... 351
1-5-3-22. Real carry RoundUp ..... 353
1-5-3-23. Generate random number RAND ..... 354
*1-5-4. Address operation instruction ..... 356*
1-5-4-1. Data type size SIZEOF ..... 356
1-5-4-2. Address operator ADR ..... 357
1-5-4-3. Bit address operator BITADR ..... 359
1-6. STRING COMMAND ..... 361
*1-6-1. Sum calculation StringSum ..... 361*
*1-6-2. String clear Clearstring ..... 363*
*1-6-3. Get the number of bytes in the string GetByteLen ..... 364*
*1-6-4. Real string read delimiter delete SubDelimiter ..... 366*
*1-6-5. Long real string read delimiter delete SubDelimiter_LR ..... 370*
*1-6-6. String letter conversion ToUCase/ToLCase ..... 374*
*1-6-7. String left/right adjustment TrimL/TrimR ..... 376*
1-7. RECIPE INSTRUCTIONS ..... 378
*1-7-1. Create new recipe CreateRecipe ..... 378*
*1-7-2. Create new recipe CreateRecipeNoSave ..... 382*
*1-7-3. Save recipe SaveRecipe ..... 386*
*1-7-4. Delete recipe DeleteRecipe ..... 387*
*1-7-5. Delete recipe file DeleteRecipeFile ..... 388*
*1-7-6. Obtain the quantity of recipe GetRecipeCount ..... 390*
*1-7-7. Get recipe name GetRecipeNames ..... 392*
*1-7-8. Read values and save as recipe file ReadAndSaveAs ..... 394*
*1-7-9. Read values and save in the recipe file ReadAndSaveRecipe ..... 398*
*1-7-10. Read values and save ReadAndSaveRecipeAs ..... 400*
*1-7-11. Read recipe ReadRecipe ..... 402*

*1-7-12. Write recipe WriteRecipe* ..... *405*
*1-7-13. Recipe instruction return values* ..... *407*

**2. SPECIAL INSTRUCTIONS** ..... **409**

2-1. MODEL AND LIBRARY CORRESPONDENCE TABLE ..... 409
2-2. XJ_HSIO HIGH SPEED COUNT LIBRARY ..... 409
*2-2-1. Enable the high speed counter【XJ_Counter_Enable】* ..... *409*
*2-2-2. Compare consistent output【XJ_Counter_Compare】* ..... *412*
*2-2-3. Preset value write in【XJ_Counter_PresetValue】* ..... *414*
*2-2-4. Probe【XJ_TouchProbe】* ..... *416*
*2-2-5. Read the pulse width measurement value of the counter【XJ_MeasurePulseWidth】* ..... *418*
*2-2-6. Counter sample【XJ_Counter_Sample】* ..... *419*
*2-2-7. Multiple segments compare【XJ_Counter_CompareArray】* ..... *420*
*2-2-8. Ring counting【XJ_Counter_SetRing】* ..... *422*
*2-2-9. Reset port of compare consistent output【XJ_ResetCmpOutput】* ..... *423*
*2-2-10. Clear the error【XJ_Counter_Reset】* ..... *424*
*2-2-11. Enable external interrupt and compare consistent interrupt XJ_EnableInterrupt* ..... *425*
*2-2-12. Interrupt parameter write XJ_WriteInterruptParameter* ..... *428*
*2-2-13. Interrupt application* ..... *430*
2-3. XINJECNT HIGH SPEED COUNT LIBRARY ..... 432
*2-3-1. High speed counter【XJ_Counter】* ..... *432*
*2-3-2. Read high speed counter【XJ_CounterGetValue】* ..... *433*
*2-3-3. Write high speed counter【XJ_CounterSetValue】* ..... *434*
*2-3-4. Application example* ..... *437*

**3. PID INSTRUCTIONS** ..... **438**

3-1. COMMAND FORMAT ..... 438
3-2. RELATED VARIABLES ..... 439
3-3. FUNCTION DESCRIPTION ..... 441
3-4. APPLICATION EXAMPLE ..... 444

**4. SYSTEM LIBRARY** ..... **452**

4-1. FUNCTION OVERVIEW ..... 452
4-2. FUNCTION BLOCK INTRODUCTION ..... 452
*4-2-1. CPU dominant frequency【XJ_GetCPUFrequency】* ..... *452*
*4-2-2. CPU temperature【XJ_GetCPUTemperature】* ..... *452*
*4-2-3. CPU occupancy【XJ_GetCPUUsage】* ..... *453*
*4-2-4. Memory size【XJ_GetMemSize】* ..... *453*
*4-2-5. Memory occupancy【XJ_GetMemUsage】* ..... *454*
*4-2-6. On time【XJ_GetBootTime】* ..... *454*
*4-2-7. PLC name【XJ_GetPLCName】* ..... *455*
*4-2-8. PLCID【XJ_GetPLCID】* ..... *456*
*4-2-9. Firmware version【XJ_GetPLCVersion】* ..... *457*
*4-2-10. Obtain network port information【XJ_GetNetInfo】* ..... *458*
*4-2-11. Set network port information【XJ_SetNetInfo】* ..... *460*
*4-2-12. Runtime version【XJ_GetRuntimeVersion】* ..... *461*
*4-2-13. BOOT version【XJ_GetBootVersion】* ..... *462*

*4-2-14. Obtain date and time【XJ_GetTime】* ..... *462*
*4-2-15. Set date and time【XJ_SetTime】* ..... *463*
*4-2-16. Network port bridging instruction* ..... *465*
4-3. PARAMETER CONFIGURATION ..... 467
**5. REMOTE IO MODULE READ AND WRITE ..... 469**
5-1. FUNCTION OVERVIEW ..... 469
5-2. FUNCTION BLOCK INTRODUCTION ..... 469
*5-2-1. Remote IO read【XJ_ECATFromTo.XJ_EC_FROM】* ..... *469*
*5-2-2. Remote IO write【XJ_ECATFromTo.XJ_EC_TO】* ..... *470*
5-3. PARAMETER CONFIGURATION ..... 471
5-4. APPLICATION ..... 473
**6. DATE TIME ..... 474**
6-1. OBTAIN THE CURRENT SYSTEM CLOCK GETSYSTEMDATE ..... 474
6-2. OBTAIN SYSTEM RUNTIME GETSYSTEMTIME ..... 476
6-3. SET THE CURRENT SYSTEM CLOCK SETSYSTEMDATE ..... 477
6-4. ADD DATE AND TIME ADD_DT_TIME ..... 479
6-5. TIME ADDITION ADD_TOD_TIM ..... 480
6-6. LEAP YEAR JUDGMENT CHKLEAPYEAR ..... 482
6-7. DATE AND TIME COMBINATION DATESTRUCTTODT ..... 483
6-8. DATE SECOND CONVERSION DATETOSEC ..... 485
6-9. DAYS AND MONTHS CONVERSION DAYSTOMONTH ..... 486
6-10. TIME DIVISION DIVTIME ..... 487
6-11. DATE TIME DECOMPOSITION DTTODATESTRUCT ..... 489
6-12. DATE, TIME, SECOND CONVERSION DTTOSEC ..... 491
6-13. GET WEEK INFORMATION GETDAYOFWEEK ..... 492
6-14. OBTAIN THE NUMBER OF DAYS IN THE TARGET MONTH GETDAYSOFMONTH ..... 494
6-15. OBTAIN THE CURRENT SYSTEM TIME STRUCTURE GETSYSTEMDATE_SDT ..... 495
6-16. GET WEEK INFORMATION GETWEEKOFYEAR ..... 496
6-17. MULTIPLYING TIME MULTIME ..... 498
6-18. NANOSECOND TIME CONVERSION NANOSECTOTIME ..... 499
6-19. SECOND DATE CONVERSION SECTODATE ..... 500
6-20. SECOND DATE TIME CONVERSION SECTODT ..... 502
6-21. SECOND TIME CONVERSION SECTOTIME ..... 503
6-22. SECOND TIME CONVERSION SECTOTOD ..... 504
6-23. DATE SUBTRACTION SUB_DATE_DATE ..... 505
6-24. SUBTRACTING DATE AND TIME SUB_DT_DT ..... 506
6-25. SUBTRACTING DATE, TIME, AND TIME SUB_DT_TIME ..... 509
6-26. SUBTRACTING TIME FROM TIME SUB_TOD_TIME ..... 510
6-27. TIME SUBTRACTION SUB_TOD_TOD ..... 511
6-28. TIME NANOSECOND CONVERSION TIMETONANOSEC ..... 513
6-29. TIME SECOND CONVERSION TIMETOSEC ..... 514
6-30. TIME SECOND CONVERSION TODTOSEC ..... 515
6-31. DATE AND TIME ROUNDING OFF TRUNCDT ..... 516
6-32. TIME ROUNDING OFF TRUNCTIME ..... 518
6-33. ROUNDING OFF TIME TRUNCTOD ..... 519

**7. COMMUNICATION INSTRUCTIONS ..... 521**

7-1. TCP COMMUNICATION ..... 521
7-1-1. TCP COMMUNICATION DATA RECEIVE TCP_RECEIVE ..... 521
*7-1-2. TCP communication data send TCP_Send ..... 523*
*7-1-3. Create TCP connection TCP_Connect ..... 524*
*7-1-4. Create TCP server-side communication service TCP_Server ..... 525*
*7-1-5. Create TCP client communication service TCP_Client ..... 526*
*7-1-6. Example of TCP communication instruction program ..... 527*
7-2. UDP COMMUNICATION ..... 535
*7-2-1. Create UDP communication connecting UDP_Peer ..... 535*
*7-2-2. UDP communication data reception UDP_Receive ..... 536*
*7-2-3. UDP communication data send UDP_Send ..... 537*
*7-2-4. Example of UDP communication ..... 538*
7-3. TCP/UDP COMMUNICATION ERROR CODE ..... 542
7-4. CAN FREE FORMAT COMMUNICATION ..... 543
*7-4-1. CAN free format communication specification ..... 543*
*7-4-2. CAN free format send FreeCan_Send_CB ..... 543*
*7-4-3. CAN free format receive FreeCan_Recv_CB ..... 545*
*7-4-4. CAN free format communication example ..... 547*

**8. FILE PROCESSING ..... 549**

8-1. CREATE DIRECTORY DIRCREATE ..... 549
8-2. DELETE DIRECTORY DIRREMOVE ..... 553
8-3. OPEN FILE FILEOPEN ..... 555
8-4. CLOSE THE FILE FILECLOSE ..... 558
8-5. FIND THE FILE FILESEEK ..... 560
8-6. WRITE IN THE FILE FILEWRITE ..... 563
8-7. READ THE FILE FILEREAD ..... 566
8-8. WRITE STRING FILEPUTS ..... 569
8-9. READ STRING FILEGETS ..... 571
8-10. WRITE VARIABLE FILE FILEWRITEVAR ..... 574
8-11. READ VARIABLE FILE FILEREADVAR ..... 577
8-12. COPY FILE FILECOPY ..... 580
8-13. DELETE FILE FILEREMOVE ..... 583
8-14. RENAME FILE FILERENAME ..... 585
8-15. COMMAND ERROR CODE ..... 587

**9. FTP COMMUNICATION ..... 590**

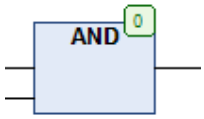
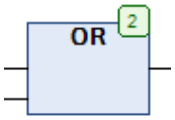
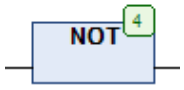
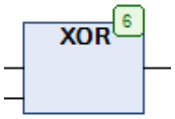
9-1. XSF AS A SERVER INSTRUCTION ..... 590
*9-1-1. Server startupXJ_FTP_server_initiate ..... 590*
*9-1-2. Server current state XJ_FTP_current_state_of_the_server ..... 592*
9-2. XSF AS A CLIENT INSTRUCTION ..... 593
*9-2-1. Connect the server XJ_FTP_login_Server ..... 593*
*9-2-2. Client read file XJ_FTP_deliver_ServerToClient ..... 594*
*9-2-3. Client transfer files XJ_FTP_deliver_ClientToServer ..... 596*
*9-2-4. Delete server files XJ_FTP_deleteFile_Server ..... 597*

9-3. EXAMPLE OF FTP COMMUNICATION COMMAND PROGRAM ..... 598
*9-3-1. Two XSF5 devices establish FTP communication..... 598*
*9-3-2. Establishing FTP Communication between XSF5 and FileZilla Software ..... 605*
9-4. FTP COMMUNICATION ERROR CODE ..... 631

# 1. Basic instructions

## 1-1. Bit logic instructions

### 1-1-1. Basic bit logic instructions

| Instruction | Instruction icon | Function |
|---|---|---|
| AND |  | Operator AND |
| OR |  | Operator OR |
| NOT |  | Operator NOT |
| XOR |  | Operator XOR |

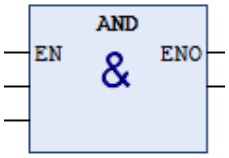
Basic bit logic instructions include AND, OR, NOT, and XOR. In XS Studio, functions can be divided into bit logic operation and Boolean logic operation.

- Bit logic operation: perform Boolean logic operation on corresponding bits of two integer data one by one, and return compatible integer results.
- Boolean logic operation: perform logical operation on two Boolean type data.

#### 1-1-1-1. Bit AND

Function: The bit AND operation instruction compares the corresponding bits of two integers. When the corresponding bits of two numbers are both 1, the corresponding result bit returned is "1". When the corresponding bits of two integers are both "0" or one of the bits is "0", the corresponding result bit is returned as "0".

| Bit AND [AND] | | | |
|---|---|---|---|
| Execution condition | Edge triggering | Suitable model | XS, XSA |
| Firmware | - | Software | - |
| Library | - | | |

| Instruction | Name | Graph | ST |
|---|---|---|---|
| AND | Bit AND |  | AND |

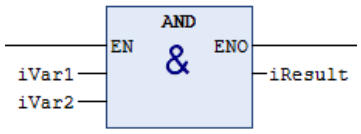
AND logic relationship table

| Input 1 | Input 2 | Result |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

<table>
<tr><th>DEMO</th><th>LD</th><th>ST</th></tr>
<tr><td>Define variables</td><td colspan="2">

```
VAR
    iVar1   :UINT:=1;
    iVar2   :UINT:=85;
    iResult:UINT;
END_VAR
```

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




</td><td>

```
1   iResult:=iVar1 and iVar2;
```

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

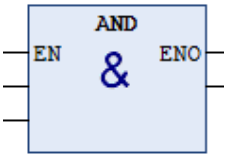
| 表达式 | 类型 | 值 |
|---|---|---|
| iVar1 | UINT | 1 |
| iVar2 | UINT | 85 |
| iResult | UINT | 1 |

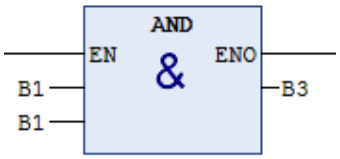
</td></tr>
<tr><td>Result analysis</td><td colspan="2">The decimal number 1 corresponds to a binary number of 0000 0001, and the decimal number 85 corresponds to a binary number of 0101 0101. According to the definition of bit AND operation, each independent bit is subjected to an AND operation one by one, resulting in the final result of 0000 0001, which is a decimal value of 1.</td></tr>
</table>

1-1-1-2. Boolean AND

Function: Boolean AND operation is used to calculate the AND results of two Boolean expression. When the result of two Boolean expression is true, it is returned as true. If one of them is false, it is returned as false.
Example: Create a POU and use the Boolean AND operation to determine the return value of the operation. The code is as follows:

| Boolean AND [AND] | | | |
|---|---|---|---|
| Execution condition | Edge triggering | Suitable model | XS, XSA |
| Firmware | - | Software | - |
| Library | - | | |

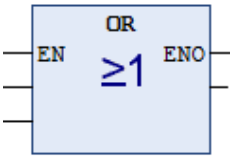
| Instruction | Name | Graph | ST |
|---|---|---|---|
| AND | Boolean AND |  | AND |

| DEMO | LD | ST |
|---|---|---|
| Define variables | VAR<br>B1:BOOL;<br>B2:BOOL;<br>B3:BOOL;<br>END_VAR | |
| Sample Program |  | 1 B3:=B1 AND B2; |
| Result analysis | When both B1 and B2 are true, the running result of the program is B3 to be true. | |

1-1-1-3. Bit OR

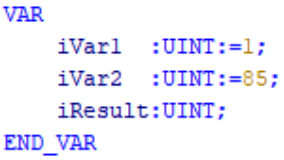
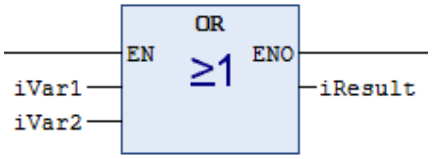
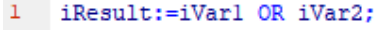
Function: The bit OR operation instruction compares the corresponding bits of two integers. When one of the corresponding bits of two numbers is "1" or both are "1", the corresponding result bit is returned as "1". When the corresponding bits of two integers are both "0", the corresponding result is returned as "0".

| Bit OR [OR] | | | |
|---|---|---|---|
| Execution condition | Edge triggering | Suitable model | XS, XSA |
| Firmware | - | Software | - |
| Library | - | | |

| Instruction | Name | Graph | ST |
|---|---|---|---|
| OR | Bit OR |  | OR |

OR logic relationship table

| Input 1 | Input 2 | Result |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |

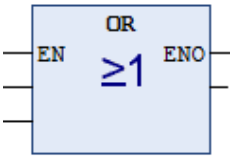
| DEMO | LD | ST |
|---|---|---|
| Define variables |  | |
| Sample Program |  |  |
| Operation result | | |
| Result analysis | The final running result of the program is 85. | |

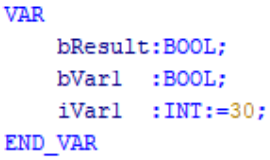
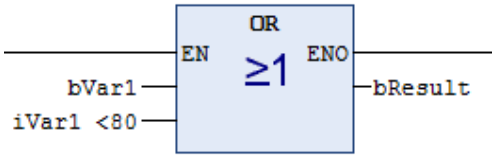
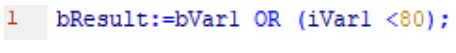
1-1-1-4. Boolean OR

Function: Boolean OR operation instruction is used to calculate the OR result of two Boolean expression. When one of the two Boolean expression returns true, the result is true. When the result of two Boolean expression is false, the result is false.

Example: Create a POU and use the Boolean OR operation to determine the return value of the operation. The code is as follows:

| Boolean OR [OR] | | | |
|---|---|---|---|
| Execution condition | Edge triggering | Suitable model | XS, XSA |
| Firmware | - | Software | - |
| Library | - | | |

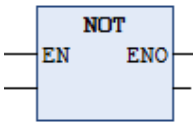
| Instruction | Name | Graph | ST |
|---|---|---|---|
| OR | Boolean OR |  | OR |

| DEMO | LD | ST |
|---|---|---|
| Define variables |  | |
| Sample Program |  |  |
| Operation result | | |
| Result analysis | Due to the initial value of iVar1 being 30, the condition of iVar1<80 is true, while the initial value of bVar1 is "0", so it is false. The logical result of one true and one false OR is true. Therefore, the condition on the right side of the equation is true, and the running result of the program is bResult being true. | |

1-1-1-5. Bit NOT

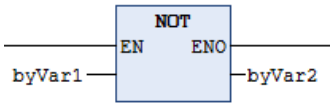
Function: Negate the logical string, changing the current value from "0" to "1" or from "1" to "0". The bit 'not' operation instruction is to take variables or constants one by one as not.

| Bit NOT [NOT] | | | |
|---|---|---|---|
| Execution condition | Edge triggering | Suitable model | XS, XSA |
| Firmware | - | Software | - |
| Library | - | | |

| Instruction | Name | Graph | ST |
|---|---|---|---|
| NOT | Bit NOT |  | NOT |

NOT logic relationship table

| Input | Result |
|---|---|
| 0 | 1 |
| 1 | 0 |

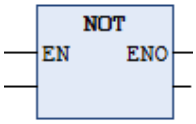
| DEMO | LD | ST |
|---|---|---|
| Define variables | | VAR<br>byVar1:BYTE:=1;<br>byVar2:BYTE;<br>END_VAR |
| Sample Program |  | 1 byVar2:=NOT byVar1; |
| Operation result | | |
| Result analysis | Due to the value of byVar1 being 1, converting it to binary yields 0000 0001. After performing bit negation, the result is 1111 1110. The final output result is 254. | |

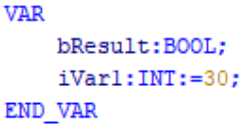
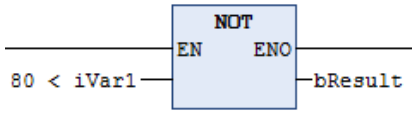
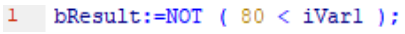
1-1-1-6. Boolean NOT

Function: Boolean "Not" operation instruction is used to calculate the result of a single Boolean expression. When the input is true, the result is false. When the input is false, the result is true.

Example: Create a POU and use the Boolean 'not' operation to determine the return value of the operation. The specific code is as follows:

| Boolean NOT [NOT] | | | |
|---|---|---|---|
| Execution condition | Edge triggering | Suitable model | XS, XSA |
| Firmware | - | Software | - |
| Library | - | | |

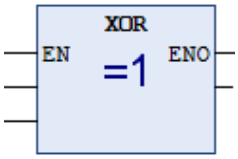
| Instruction | Name | Graph | ST |
|---|---|---|---|
| NOT | Boolean NOT |  | NOT |

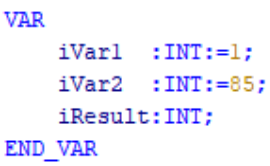
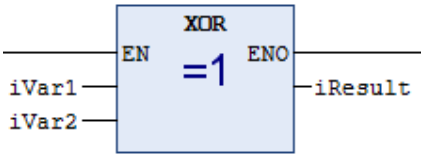
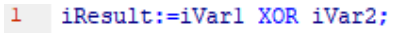
| DEMO | LD | ST |
|---|---|---|
| Define variables |  | |
| Sample Program |  |  |
| Operation result | | |
| Result analysis | 80<30 this proposition is false. After using the NOT instruction to negate the Boolean expression, the result is true, so the final bResult is true. | |

1-1-1-7. Bit XOR

Function: Compare the corresponding bits of two integers using the bit XOR operation instruction. When the corresponding bit of two integers is one "1" and the other is "0", the corresponding result bit returned is "1". When the corresponding bits of two integers are both "1" or "0", the corresponding result bit is returned as "0".
Example: Create a POU, perform bit XOR on variables iVar1 and iVar2, and output the results. The specific implementation code is as follows:

| Bit XOR [XOR] | | | |
|---|---|---|---|
| Execution condition | Edge triggering | Suitable model | XS, XSA |
| Firmware | - | Software | - |
| Library | - | | |

| Instruction | Name | Graph | ST |
|---|---|---|---|
| XOR | Bit XOR |  | XOR |

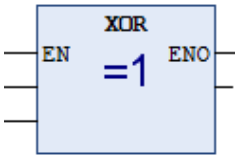
| DEMO | LD | ST |
|---|---|---|
| Define variables |  | |
| Sample Program |  |  |
| Operation result | | |
| Result analysis | The decimal number 1 corresponds to a binary number of 0000 0001, and the decimal number 85 corresponds to a binary number of 0101 0101. According to the definition of the bit XOR operation instruction, the result is 84.<br>Bit XOR operation. Only when the input state of one contact is "1" and the input state of the other contact is "0", the output is "1". If both contact states are "1" or "0" at the same time, the output is "0". | |

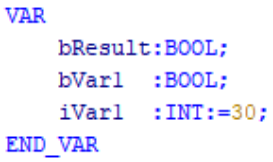
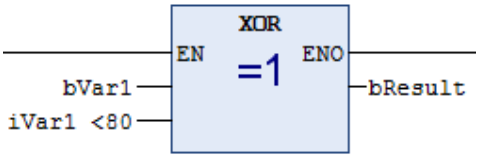
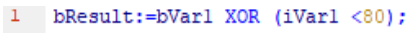
1-1-1-8. Boolean XOR

Function: Boolean XOR operation instruction is used to calculate the results of two Boolean expression. Only when one expression is true and the other is false, the result returned by the expression is true. When both expressions evaluate to true or false, the returned result is false.

Example: Create a POU and use the Boolean XOR operation instruction to determine whether the return value is TRUE or FALSE. The specific implementation code is as follows:

| Boolean XOR [XOR] | | | |
|---|---|---|---|
| Execution condition | Edge triggering | Suitable model | XS, XSA |
| Firmware | - | Software | - |
| Library | - | | |

| Instruction | Name | Graph | ST |
|---|---|---|---|
| XOR | Boolean XOR |  | XOR |

| DEMO | LD | ST |
|---|---|---|
| Define variables |  | |
| Sample Program |  |  |
| Operation result | | |
| Result analysis | The running result of the program is TRUE. | |

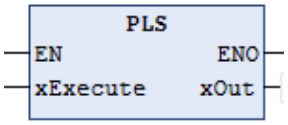
## 1-1-2. Bit logic operation instruction

### 1-1-2-1. Rising edge output PLS

(1) Instruction overview

When the PLS instruction xExecute detects a rising edge, xOut will be set to a true state, which will be set to a false state after one scan cycle.

| Rising edge output [PLS] | | | |
|---|---|---|---|
| Execution condition | Edge triggering | Suitable model | XS, XSA |
| Firmware | - | Software | XS StudioV2.2.0 and above |
| Library | XJ_DataUtils | | |

| Instruction | Name | Graph | ST |
|---|---|---|---|
| PLS | Rising edge output |  | PLS(xExecute:= , xOut=> ); |

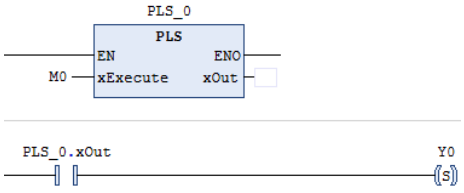
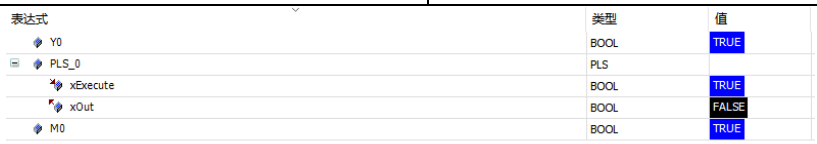
(2) Related variables

| VAR_INPUT | Name | Data type | Valid range | Initial value | Description |
|---|---|---|---|---|---|
| xExecute | Input signal | BOOL | FALSE,TRUE | FALSE | The BOOL value to be detected |
| VAR_OUTPUT | Name | Data type | Valid range | Initial value | Description |
| Out | Output signal | BOOL | FALSE,TRUE | FALSE | Output result |

(3) Function

- Output a scan cycle command when the rising edge is reached.

(4) Program example

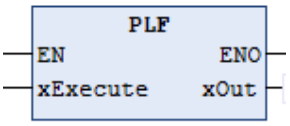
| DEMO | LD | ST |
|---|---|---|
| Define the variable | | VAR<br>PLS_0: PLS;<br>M0: BOOL;<br>Y0: BOOL;<br>END_VAR |
| Program example |  | PLS_0(xExecute:=M0, xOut=> );<br>IF PLS_0.xOut THEN<br>Y0:=TRUE;<br>END_IF |
| Operation result |  | |

1-1-2-2. Falling edge output PLF

(1) Instruction overview

When the PLF instruction xExecute detects a falling edge, xOut will be set to a true state, which will be set to a false state after one scan cycle.

| Falling edge output [PLF] | | | |
|---|---|---|---|
| Execution condition | Edge triggering | Suitable model | XS, XSA |
| Firmware | - | Software | XS StudioV2.2.0 and above |
| Library | XJ_DataUtils | | |

| Instruction | Name | Graph | ST |
|---|---|---|---|
| PLF | Falling edge output |  | PLF_0(xExecute:=, xOut=> ); |

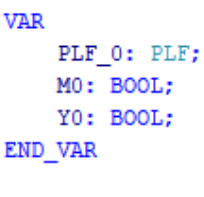
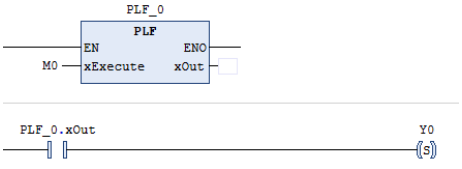
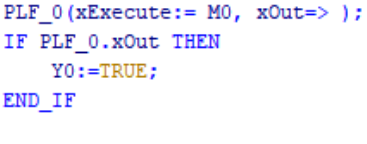
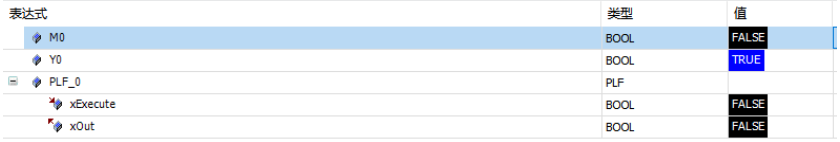
(2) Related variables

| VAR_INPUT | Name | Data type | Valid range | Initial value | Description |
|---|---|---|---|---|---|
| xExecute | Input signal | BOOL | FALSE,TRUE | FALSE | The BOOL value to be detected |
| VAR_OUTPUT | Name | Data type | Valid range | Initial value | Description |
| xOut | Output signal | BOOL | FALSE,TRUE | FALSE | Output result |

(3) Function

- Output a scanning cycle command at the falling edge

(4) Program example

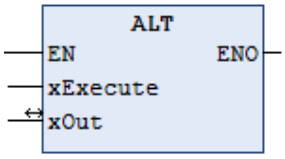
| DEMO | LD | ST |
|---|---|---|
| Define the variable | |  |
| Program example |  |  |
| Operation result |  | |

1-1-2-3. Alternating output ALT

(1) Instruction overview

When the ALT instruction xExecute detects the rising edge, xOut will reverse its state.

| Alternating output [ALT] | | | |
|---|---|---|---|
| Execution condition | Edge triggering | Suitable model | XS, XSA |
| Firmware | - | Software | XS StudioV2.2.0 and above |
| Library | XJ_DataUtils | | |

| Instruction | Name | Graph | ST |
|---|---|---|---|
| ALT | Alternating output |  | ALT_0(xExecute:= , xOut:= ); |

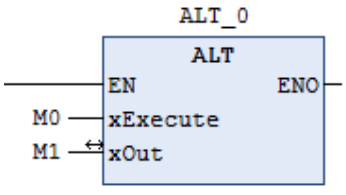
(2) Related variables

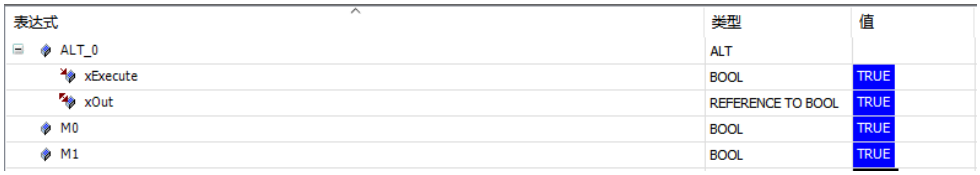
| VAR_INPUT | Name | Data type | Valid range | Initial value | Description |
|---|---|---|---|---|---|
| xExecute | Input signal | BOOL | FALSE,TRUE | FALSE | The BOOL value to be detected |
| VAR_OUTPUT | Name | Data type | Valid range | Initial value | Description |
| xOut | Output signal | BOOL | FALSE,TRUE | FALSE | If the rising edge of xExecute is detected and its state is reversed |

(3) Function

- After detecting the rising edge of xExecute, the output signal state will reverse; Convert from original FALSE to True, or from True to FALSE

(4) Program example

| DEMO | LD | ST |
|---|---|---|
| Define the variable | | VAR<br>M0: BOOL;<br>M1: BOOL;<br>ALT_0: ALT;<br>END_VAR |
| Program example |  | ALT_0(xExecute:=M0 , xOut:=M1 ); |

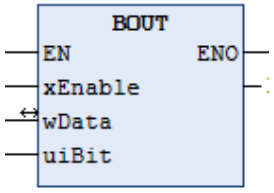
| | |
|---|---|
| Operation result |  |

1-1-2-4. Bit data output BOUT

(1) Instruction overview

Set the bit status.

| Bit data output [BOUT] | | | |
|---|---|---|---|
| Execution condition | Edge triggering | Suitable model | XS, XSA |
| Firmware | - | Software | XS StudioV2.2.0 and above |
| Library | XJ_DataUtils | | |

| Instruction | Name | Graph | ST |
|---|---|---|---|
| BOUT | Bit vlaue output |  | BOUT(xEnable:= ,<br>wData:= ,<br>uiBit:=); |

(2) Related variable

| VAR_INPUT | Name | Data type | Valid range | Initial value | Description |
|---|---|---|---|---|---|
| xExecute | Input signal | BOOL | FALSE,TRUE | FALSE | The BOOL value to be detected |
| uiBit | Input signal | UINT | As the data type | 0 | The bit to be set |
| VAR_IN_OUT | Name | Data type | Valid range | Initial value | Description |
| wData | Output signal | Word | As the data type | 0 | The value to be set |

(3) Function

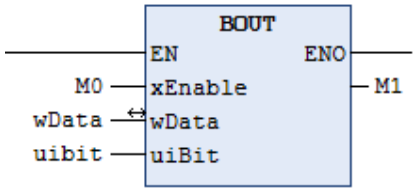
- Detect the rising edge of xExecute and set on the bit set by the uiBit corresponding to wData data.

(4) Note

- WDate is the input and output data, which needs to be set in the variable table and converted to binary. Confirm the conversion bit.

(5) Program example

| DEMO | LD | ST |
|---|---|---|
| Define the variable | | VAR<br><br>M0:BOOL;<br>wData: WORD;<br>uibit: UINT;<br>M1: BOOL;<br><br>END_VAR |

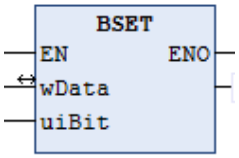
| | | |
|---|---|---|
| Program example |  | BOUT(xEnable:= M0,<br>wData:=wData,<br>uiBit:=uibit ); |
| Operation result | | |
| Result analysis | After setting the initial data of wDate to 22 (binary: 10110), uiBit equals 3 and M0 is set on, the bit3 of wDate data is set on, the wDate value changed to 30 (binary: 11110), and the return signal M1 is set to True. | |

1-1-2-5. Set on bit BSET

(1) Instruction overview

Set on the bit.

| Set on bit [BSET] | | | |
|---|---|---|---|
| Execution condition | Edge triggering | Suitable model | XS, XSA |
| Firmware | - | Software | XS StudioV2.2.0 and above |
| Library | XJ_DataUtils | | |

| Instruction | Name | Graph | ST |
|---|---|---|---|
| BSET | Set on the bit |  | BSET(wData:=,<br>uiBit:=); |

(2) Related variables

| VAR_INPUT | Name | Data type | Valid range | Initial value | Description |
|---|---|---|---|---|---|
| uiBit | Input signal | UINT | As the data type | 0 | The bit to be set |
| VAR_IN_OUT | Name | Data type | Valid range | Initial value | Description |
| wData | Output signal | Word | As the data type | 0 | The value to be set |

(3) Function

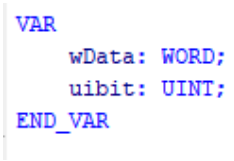
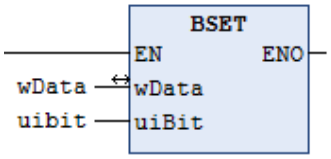
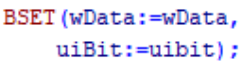
- Set on the bit set by uiBit corresponding to wData.

(4) Note

- WDate is the input and output data, which needs to be set in the variable table and converted to binary. Confirm the conversion bit.
- Regardless of whether the BSET command is used later or not, the ON status will always be maintained, and

the BRST command can be used to set this bit to OFF.

(5) Program example

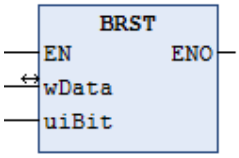
| DEMO | LD | ST |
|---|---|---|
| Define the variable |  | |
| Program example |  |  |
| Operation result | | |
| Result analysis | The initial data of wDate is set to 22 (binary: 10110), and uiBit is equal to 3; The bit3 of wDate data is set on, the wDate value changed to 30 (binary: 11110). | |

1-1-2-6. Reset bit BRST

(1) Instruction overview

Reset the bit.

| Reset bit [BRST] | | | |
|---|---|---|---|
| Execution condition | Edge triggering | Suitable model | XS, XSA |
| Firmware | - | Software | XS StudioV2.2.0 and above |
| Library | XJ_DataUtils | | |

| Instruction | Name | Graph | ST |
|---|---|---|---|
| BRST | Reset the bit |  | BRST(wData:=,<br>uiBit:=); |

(2) Related variables

| VAR_INPUT | Name | Data type | Valid range | Initial value | Description |
|---|---|---|---|---|---|
| uiBit | Input signal | UINT | As the data type | 0 | The bit to be set |
| VAR_IN_OUT | Name | Data type | Valid range | Initial value | Description |
| wData | Output signal | Word | As the data type | 0 | The value to be set |

(3) Function

- Reset the bit set by uiBit corresponding to wData

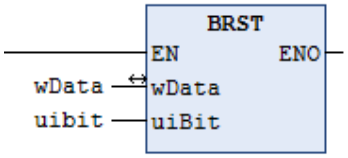
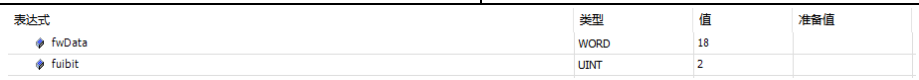
(4) Note

- WDate is the input and output data that needs to be set in the variable table, and the data needs to be converted to binary. Confirm the conversion bit
- When XN_DataUtils library version 1.0.0.6 and below, wData only supports WORD type; When XN_DataUtils library version 1.0.0.6 or above, wData supports BYTE/WORLD/DWORD/LWORD types.

(5) Program example

<table>
<tr><th>DEMO</th><th>LD</th><th>ST</th></tr>
<tr><td>Define the variable</td><td colspan="2"><pre>VAR

    fwData: WORD;
    fuibit: UINT;

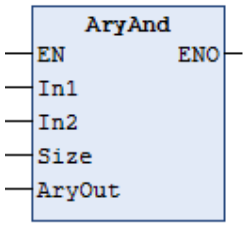
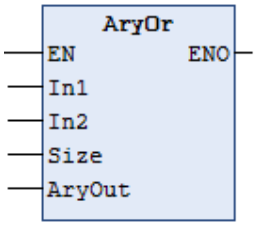
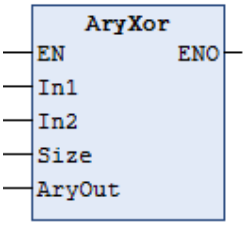
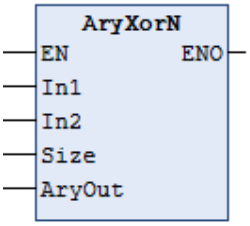
END_VAR</pre></td></tr>
<tr><td>Program example</td><td> 
</td><td><pre>BRST(wData:=fwData,
    uiBit:=fuibit);</pre></td></tr>
<tr><td>Operation result</td><td colspan="2"> 
</td></tr>
<tr><td>Result analysis</td><td colspan="2">The initial data of wDate is set to 22 (binary: 10110), uiBit is equal to 2, and bit3 of wDate data is reset, wDate changed to 18 (binary: 10010).</td></tr>
</table>

1-1-2-7. Logical AND/Logical OR/XOR/XNOR AryAnd/AryOr/AryXor/AryXorN

(1) Instruction overview

Perform operations on each bit of the Boolean and bit string of the elements between arrays.

| Logical AND/Logical OR/XOR/XNOR [AryAnd/AryOr/AryXor/AryXorN] | | | |
|---|---|---|---|
| Execution condition | Edge triggering | Suitable model | XS, XSA |
| Firmware | - | Software | XS StudioV2.2.0 and above |
| Library | XJ_DataUtils | | |

| Instruction | Name | Graph | ST |
|---|---|---|---|
| AryAnd | Logical and |  | AryAnd(In1:= ,<br>In2:= ,<br>Size:= ,<br>AryOut:= ); |
| AryOr | Logical or |  | AryOr(In1:=,<br>In2:= ,<br>Size:= ,<br>AryOut:=] ); |
| AryXor | Exclusive or |  | AryXor(In1:= ,<br>In2:= ,<br>Size:= ,<br>AryOut:= ); |
| AryXorN | Same or |  | AryXorN(In1:=,<br>In2:=,<br>Size:=,<br>AryOut:=); |

(2) Related variables

| VAR_INPUT | Name | Data type | Valid range | Initial value | Description |
|---|---|---|---|---|---|
| In1[], In2[] array | Arithmetic object array | BOOL/BYTE/ WORD/DWORD/ LWORD | As the data type | - | Arithmetic object array |
| Size | Data | UINT | As the data | - | Number of operation |

| | elements | | type | | object data elements |
|---|---|---|---|---|---|
| VAR_IN_OUT | Name | Data type | Valid range | Initial value | Description |
| Aryout[] | Operation result array | BOOL/BYTE/ WORD/DWORD /LWORD | As the data type | - | Operation result array |

(3) Function

- Perform logical operations on the corresponding bits of each of the "Size" elements starting with In1[] and In2[] in the operation object array. The operation result is saved in the corresponding element of AryOut[]. Therefore, the data types of In1[], In2[], and AryOut[] must be the same.
- The input-output relationship of the operation is as follows:

**AryAnd**

If one of the two elements is FALSE, the result of the operation is FALSE; Only when both elements are true, the result of the operation is true.

| In1[] element bit | In1[] element bit | AryOut[] element result bit |
|---|---|---|
| FALSE | FALSE | FALSE |
| FALSE | TRUE | FALSE |
| TRUE | FALSE | FALSE |
| TRUE | TRUE | TRUE |

**AryoR**

If one of the two elements is true, the result of the operation is true; Only when both elements are FALSE, the result of the operation is FALSE.

| In1[] element bit | In1[] element bit | AryOut[] element result bit |
|---|---|---|
| FALSE | FALSE | FALSE |
| FALSE | TRUE | TRUE |
| TRUE | FALSE | TRUE |
| TRUE | TRUE | TRUE |

**AryXor**

If two elements are the same, the result of the operation is FALSE; If two elements are different, the result of the operation is true.

| In1[] element bit | In1[] element bit | AryOut[] element result bit |
|---|---|---|
| FALSE | FALSE | FALSE |
| FALSE | TRUE | TRUE |
| TRUE | FALSE | TRUE |
| TRUE | TRUE | FALSE |

**AryXorN**

If two elements are the same, the result of the operation is True; If two elements are different, the result of the operation is FALSE.

| In1[] element bit | In1[] element bit | AryOut[] element result bit |
|---|---|---|
| FALSE | FALSE | TRUE |
| FALSE | TRUE | FALSE |
| TRUE | FALSE | FALSE |
| TRUE | TRUE | TRUE |

(4) Note

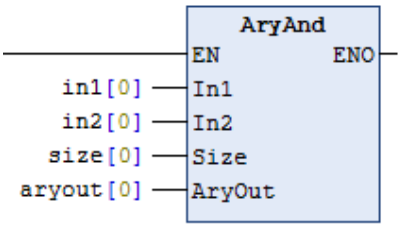
- The data types of In1[], In2[], and AryOut[] should all be set to the same. If different, the function returns a value of FALSE, and the value of "AryOut" remains unchanged.
- The value of "Size" should not exceed any array area in In1[], In2[], or AryOut[] to avoid PLC exceptions.
- When the value of "Size" is 0, the value of AryOut[] remains unchanged, and the function returns a value of True.
- In1[] and In2[] require Boolean and bitstring types, otherwise the program will compile an error at runtime.

(5) Program example

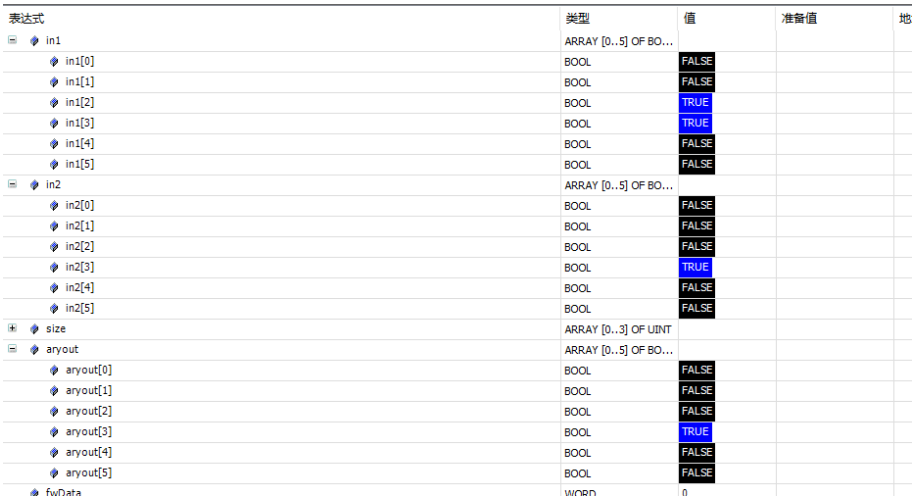
<table>
<tr><th>DEMO</th><th>LD</th><th>ST</th></tr>
<tr><td>Define the variable</td><td colspan="2">

```
VAR
    in1:ARRAY[0..5] OF BOOL;
    in2:ARRAY[0..5] OF BOOL;
    size:ARRAY[0..3] OF UINT;
    aryout:ARRAY[0..5] OF BOOL;

END_VAR
```

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

```
AryAnd(In1:=in1[0] ,
    In2:=in2[0] ,
    Size:=size[0] ,
    AryOut:=aryout[0] );
```

</td></tr>
<tr><td>Operation result</td><td colspan="2"> 
</td></tr>
<tr><td>Result analysis</td><td colspan="2">When in1[0] is FALSE, in2[0] is FALSE, aryout[0] result is FALSE;<br>When in1[0] is FALSE, in2[0] is TRUE, aryout[0] result is FALSE;<br>When in1[0] is TRUE, in2[0] is FALSE, aryout[0] result is FALSE;<br>When in1[0] is TRUE, in2[0] is TRUE, aryout[0] result is TRUE.</td></tr>
</table>

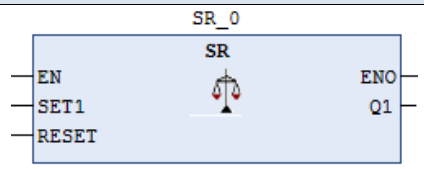
## 1-1-3. Set priority and reset priority trigger instructions

In a relay system, several pairs of contacts of a relay act simultaneously. In PLC, instructions are executed one by one, and the execution of instructions is sequential, without any "simultaneous" instructions. So the setting and reset commands for coil format have priority. The set and reset inputs of SR trigger and RS trigger are in the same instruction, and the set and reset inputs are executed by the one below the instruction input. The SR trigger is a "set priority" type trigger. When the set signal (SET1) and reset signal (RESET) are both 1, the trigger ultimately enters the set state. The RS trigger is a "reset priority" type trigger. When the set signal (SET) and reset signal (RESET1) are both 1, the trigger ultimately enters the reset state.

### 1-1-3-1. Set priority trigger SR

When both the set input signal and the reset input signal are valid. Set input signal priority.

| Set priority trigger [SR] | | | |
|---|---|---|---|
| Execution condition | Edge triggering | Suitable model | XS, XSA |
| Firmware | - | Software | - |
| Library | Standard | | |

| Instruction | Instruction icon | Function |
|---|---|---|
| SR |  | SR_0(SET1:= ,<br>RESET:= ,<br>Q1=> ); |

Related variables

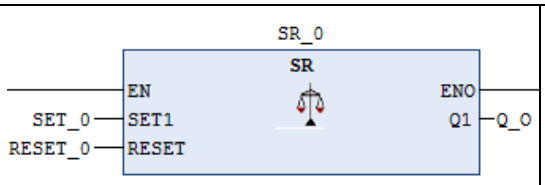
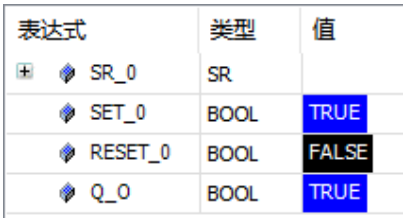
| VAR_INPUT | Name | Data type | Effective range | Initial value | Description |
|---|---|---|---|---|---|
| SET1 | Set signal | BOOL | [FALSE,TRUE] | FALSE | - |
| RESET | Reset siganl | BOOL | [FALSE,TRUE] | FALSE | - |
| VAR_OUTPUT | Name | Data type | Effective range | Initial value | Description |
| Q1 | Output status | BOOL | [FALSE,TRUE] | FALSE | Set priority output |

Function: Set bistable trigger, with set priority. Logical relationship: Q1=(NOT RESET AND Q1) OR SET1, where SET1 is the set signal and RESET is the reset signal.
Syntax: When SET1 is "1", regardless of whether RESET is "1" or not, Q1 output is "1". When SET1 is "0", if Q1 output is "1", once RESET is "1", Q1 output will immediately reset to "0". If Q1 output is "0", regardless of whether RESET is "1" or "0", Q1 output remains "0".

SR status table

| SET1 | RESET | Q1 output |
|---|---|---|
| 0 | 0 | Maintain the original state |
| 1 | 0 | 1 |
| 0 | 1 | 0 |

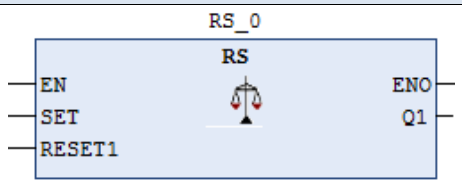
| | | |
|---|---|---|
| 1 | 1 | 1 |

| DEMO | LD | ST |
|---|---|---|
| Define variables | | VAR<br>SR_0 : SR;<br>SET_0 :BOOL:=TRUE;<br>RESET_0:BOOL;<br>Q_0 :BOOL;<br>END_VAR |
| Sample Program |  | 1 SR_0(SET1:= SET_0,<br>2 RESET:= RESET_0,<br>3 Q1=> Q_0); |
| Operation result |  | |

1-1-3-2. Reset priority trigger RS

When both the set input signal and the reset input signal are valid. Reset input signal priority.

| Reset priority trigger [RS] | | | |
|---|---|---|---|
| Execution condition | Edge triggering | Suitable model | XS, XSA |
| Firmware | - | Software | - |
| Library | Standard | | |

| Instruction | Instruction icon | Function |
|---|---|---|
| RS |  | RS_0(SET:= ,<br>RESET1:= ,<br>Q1=> ); |

Related variables

| VAR_INPUT | Name | Data type | Effective range | Initial value | Description |
|---|---|---|---|---|---|
| SET1 | Set signal | BOOL | [FALSE,TRUE] | FALSE | - |
| RESET1 | Reset siganl | BOOL | [FALSE,TRUE] | FALSE | - |
| VAR_OUTPUT | Name | Data type | Effective range | Initial value | Description |

| | | | | | |
|---|---|---|---|---|---|
| Q1 | Output status | BOOL | [FALSE,TRUE] | FALSE | Reset priority output |

Function: Reset the bistable trigger, with reset priority. Logical relationship: Q1=NOT RESET1 AND (Q1 OR SET), where SET is the set signal and RESET1 is the reset signal.

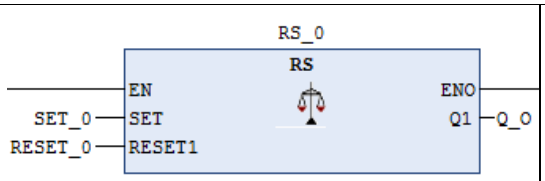
Syntax: When RESET1 is "1", regardless of whether SET is "1" or not, Q1 output is "0". When RESET1 is "0", if Q1 output is "0", once SET is "1", Q1 output is immediately set to "1". If Q1 output is "1", regardless of whether SET is "1" or "0", Q1 output remains "1".

RS status table

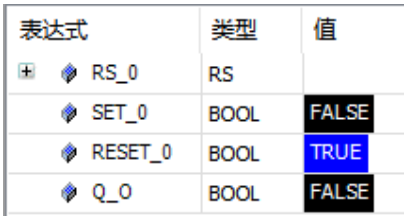
| SET | RESET1 | Q1 output |
|---|---|---|
| 0 | 0 | Maintain the original state |
| 1 | 0 | 1 |
| 0 | 1 | 0 |
| 1 | 1 | 0 |

| DEMO | LD | ST |
|---|---|---|
| Define variables | | |
| Sample Program | | |
| Operation result | | |

```
VAR
    RS_0: RS;
    SET_0  :BOOL;
    RESET_0:BOOL:=TRUE;
    Q_0    :BOOL;
END_VAR
```




```
1  RS_0(SET:= SET_0,
2       RESET1:= RESET_0,
3       Q1=> Q_0);
```



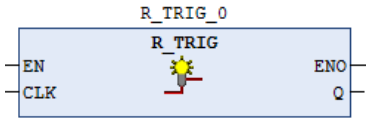
## 1-1-4. Edge detection instruction

The edge detection instruction is used to detect changes in the rising edge (signal from 0-->1) and falling edge (signal from 1-->0) of the BOOL signal. In each scanning cycle, the signal state is compared with its state in the previous scanning cycle. If different, it indicates a jump edge. Therefore, the signal state from the previous cycle must be stored in order to be compared with the new signal state.

### 1-1-4-1. Rising edge detection R_TRIG

For detecting the rising edge.

| Rising edge detection [R_TRIG] | | | |
|---|---|---|---|
| Execution condition | Edge triggering | Suitable model | XS, XSA |
| Firmware | - | Software | - |
| Library | Standard | | |

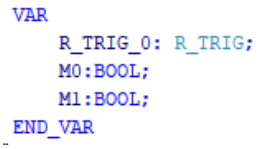
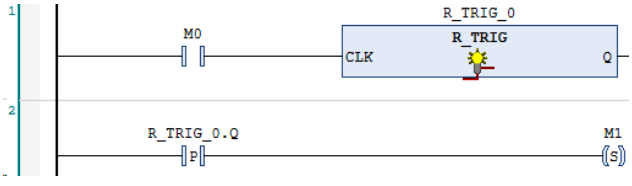
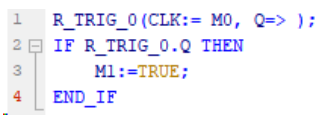
| Instruction | Instruction icon | Function |
|---|---|---|
| R_TRIG |  | R_TRIG_0(CLK:= , Q=> ); |

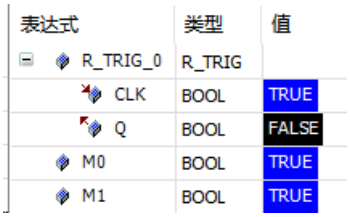
Related variables

| VAR_INPUT | Name | Data type | Effective range | Initial value | Description |
|---|---|---|---|---|---|
| CLK | Input signal | BOOL | [FALSE,TRUE] | FALSE | The rising edge of the signal (from 0 to 1) is effective |
| VAR_OUTPUT | Name | Data type | Effective range | Initial value | Description |
| Q | Output signal | BOOL | [FALSE,TRUE] | FALSE | Detect the rising edge |

Function: Used to detect the rising edge.

Syntax: When CLK changes from "0" to "1", the rising edge detector starts, and the Q output first changes from "1" to "0", lasting for one PLC operation cycle. If CLK remains at "1" or "0" continuously, Q output remains at "0".

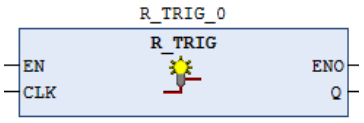
| DEMO | LD | ST |
|---|---|---|
| Define variables |  | |
| Sample Program |  |  |

| | |
|---|---|
| Operation result |  |

1-1-4-2. Falling edge detection F_TRIG

For detecting the falling edge.

| Falling edge detection [F_TRIG] | | | |
|---|---|---|---|
| Execution condition | Edge triggering | Suitable model | XS, XSA |
| Firmware | - | Software | - |
| Library | Standard | | |

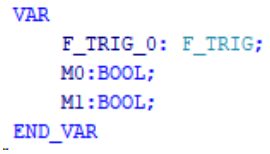
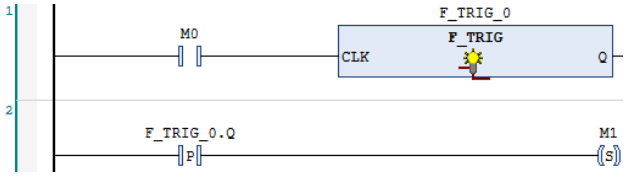
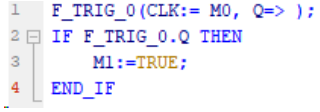
| Instruction | Instruction icon | Function |
|---|---|---|
| F_TRIG |  | F_TRIG_0(CLK:= , Q=> ); |

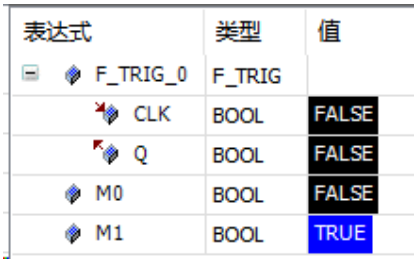
Related variables

| VAR_INPUT | Name | Data type | Effective range | Initial value | Description |
|---|---|---|---|---|---|
| CLK | Input signal | BOOL | [FALSE,TRUE] | FALSE | The falling edge of the signal (from 1 to 0) is effective |
| VAR_OUTPUT | Name | Data type | Effective range | Initial value | Description |
| Q | Output signal | BOOL | [FALSE,TRUE] | FALSE | Detect the falling edge |

Function: Used to detect the falling edge.

Syntax: When CLK changes from "1" to "0", the falling edge detector starts, and the Q output first changes from "1" to "0", lasting for one PLC operation cycle. If CLK remains at "1" or "0" continuously, Q output remains at "0".

| DEMO | LD | ST |
|---|---|---|
| Define variables |  | |
| Sample Program |  |  |

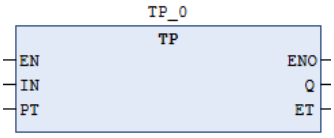
| | |
|---|---|
| Operation result |  |

# 1-2. Timer

## 1-2-1. Pulse timer TP

Pulse timer.

| Pulse timer [TP] | | | |
|---|---|---|---|
| Execution condition | Normally ON/OFF triggering | Suitable model | XS, XSA |
| Firmware | - | Software | - |
| Library | Standard | | |

| Command | Command icon | ST |
|---|---|---|
| TP |  | TP_0(IN:= ,<br>PT:= ,<br>Q=> ,<br>ET=> ); |

Related variables

| VAR_INPUT | Name | Data type | Effective range | Initial value | Description |
|---|---|---|---|---|---|
| IN | Input signal | BOOL | [FALSE,TRUE] | FALSE | FALSE → TRUE start timing |
| PT | Setting time | TIME | - | T#0ms | - |
| VAR_OUTPUT | Name | Data type | Effective range | Initial value | Description |
| Q | Timing completed | BOOL | [FALSE,TRUE] | FALSE | TRUE → FALSE timing completed |
| ET | Current timing | TIME | - | T#0ms | - |

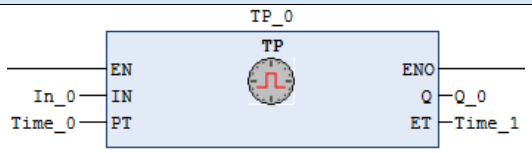
Function: Pulse timing.

Once IN becomes "True", Q becomes "True", and time will start counting in milliseconds in ET until its value equals PT and Q becomes "FALSE".

When the PT time expires, ET will maintain a timed time until IN becomes "FALSE" and returns to T#0ms. If the input IN has already changed to "FALSE" before reaching the PT timing time, the ET will immediately reset to T # 0ms after timing to the set PT time.

When IN is set to FALSE and PT=T#0ms, the timer can be reset.

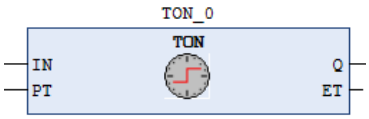
| DEMO | LD | ST |
|---|---|---|
| Define variables | | VAR<br>TP_0 : TP;<br>In_0 :BOOL;<br>Time_0:TIME;<br>Q_0 :BOOL;<br>Time_1:TIME;<br>END_VAR |

| DEMO | LD | ST |
|---|---|---|
| Sample Program |  | 1 TP_0(IN:= In_0,<br>2 PT:= Time_0,<br>3 Q=> Q_0,<br>4 ET=> Time_1); |
| Operation result | | |

## 1-2-2. Power on delay timer TON

After a delay in power on for a period of time, output a timing completion signal.

| Power on delay timer [TON] | | | |
|---|---|---|---|
| Execution condition | Normally ON/OFF triggering | Suitable model | XS, XSA |
| Firmware | - | Software | - |
| Library | Standard | | |

| Command | Command icon | Function |
|---|---|---|
| TON |  | TON_0(IN:= ,<br>PT:= ,<br>Q=> ,<br>ET=> ); |

Related variables

| VAR_INPUT | Name | Data type | Effective range | Initial value | Description |
|---|---|---|---|---|---|
| IN | Input signal | BOOL | [FALSE,TRUE] | FALSE | FALSE→TRUE start timing, keep it true during the timing process |
| PT | Setting time | TIME | - | T#0ms | - |
| VAR_OUTPUT | Name | Data type | Effective range | Initial value | Description |
| Q | Timing completed | BOOL | [FALSE,TRUE] | FALSE | After the timing is completed, FALSE becomes True |
| ET | Current timing | TIME | - | T#0ms | - |

Function: Power on delay timing.

When the input IN changes from "FALSE" to "True", the timer starts. When the signal IN at the input end is always maintained at "True" and ET reaches the timed time PT, its output end Q is "True".

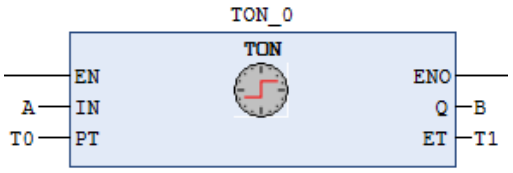
The output end ET provides a timing time, which starts from T#0s and ends at the set PT time. When ET reaches PT, it will maintain a timed time until IN becomes "FALSE".

If the input IN becomes "FALSE" and the output ET immediately becomes T#0s before ET reaches the PT timing time. So in order to restart the timer, IN can be set to "FALSE".

<table>
<tr><th>DEMO</th><th>LD</th><th>ST</th></tr>
<tr><td>Define variables</td><td colspan="2">

```
VAR
    TON_0: TON;
    A:BOOL;
    T0:TIME;
    B:BOOL;
    T1:TIME;
END_VAR
```

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




</td><td>

```
1  TON_0(IN:= A,
2       PT:= T0,
3       Q=> B,
4       ET=> T1);
```

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

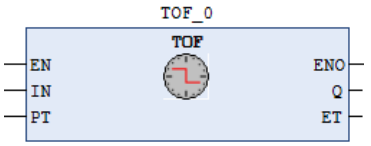
| 表达式 | 类型 | 值 |
|---|---|---|
| TON_0 | TON | |
| IN | BOOL | TRUE |
| PT | TIME | T#10s |
| Q | BOOL | TRUE |
| ET | TIME | T#10s |
| A | BOOL | TRUE |
| T0 | TIME | T#10s |
| B | BOOL | TRUE |
| T1 | TIME | T#10s |

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

## 1-2-3. Power off delay timer TOF

After a power outage delay for a period of time, output a timing completion signal.

| Power off delay timer [TOF] | | | |
|---|---|---|---|
| Execution condition | Normally ON/OFF triggering | Suitable model | XS, XSA |
| Firmware | - | Software | - |
| Library | Standard | | |

| Command | Command icon | Function |
|---|---|---|
| TOF |  | TOF_0(IN:= ,<br>PT:= ,<br>Q=> ,<br>ET=> ); |

Related variables

| VAR_INPUT | Name | Data type | Effective range | Initial value | Description |
|---|---|---|---|---|---|
| IN | Input signal | BOOL | [FALSE,TRUE] | FALSE | TRUE → FALSE start timing, keep it as FALSE during the timing process |
| PT | Setting time | TIME | - | T#0ms | - |
| VAR_OUTPUT | Name | Data type | Effective range | Initial value | Description |
| Q | Timing completed | BOOL | [FALSE,TRUE] | FALSE | After the timing is completed, TRUE becomes FALSE |
| ET | Current timing | TIME | - | T#0ms | - |

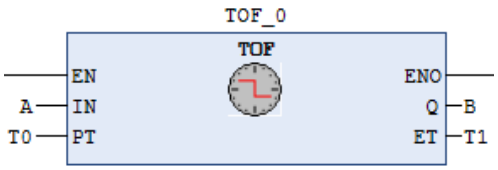
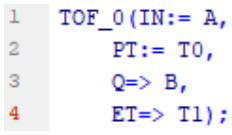
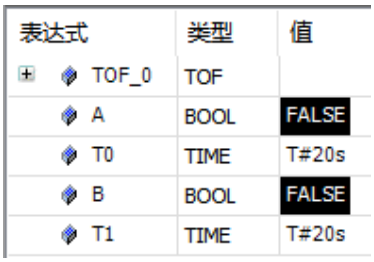
Function: Power off delay timing.

When the input terminal IN changes from "FALSE" to "True", the Q output signal of the timer is "True". When the input terminal IN starts to change to "FALSE", the timer starts. During the timer timing process, its output Q remains "True". When the timing time PT is reached, the output terminal Q resets.

The output end ET provides a timing time, which starts from T#0s and ends at the set PT time. When ET reaches PT, it will maintain a timed time until IN becomes "True".

If the input IN becomes "True" and the output ET immediately becomes T#0s before ET reaches the PT timing time. So in order to restart the timer, IN can be set to "True".

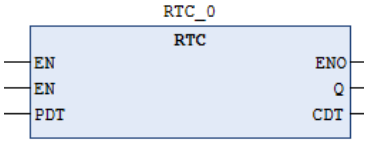
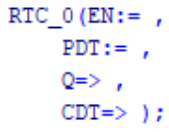
| DEMO | LD | ST |
|---|---|---|
| Define variables | | VAR<br>TOF_0: TOF;<br>A :BOOL;<br>T0 :TIME;<br>B :BOOL;<br>T1 :TIME;<br>END_VAR |

| | | |
|---|---|---|
| Sample Program |  |  |
| Operation result |  | |

## 1-2-4. RTC

Start timing from the set time.

| RTC [RTC] | | | |
|---|---|---|---|
| Execution condition | Normally ON/OFF triggering | Suitable model | XS, XSA |
| Firmware | - | Software | - |
| Library | Standard | | |

| Command | Command icon | Function |
|---|---|---|
| RTC |  |  |

Related variable

| VAR_INPUT | Name | Data type | Effective range | Initial value | Description |
|---|---|---|---|---|---|
| EN | Enable | BOOL | [FALSE,TRUE] | FALSE | Rising edge: CDT set to PDT, start rising at CDT<br>Falling edge: CDT set to DT#1970-01-01 00:00:00 |
| PDT | Set the time and date when it will be launched | DATE_AND_TIME | (DT#1970-1-1-0:0:0)-(2106-2-7-6:28:15) | DT#1970-1-1-0:0:0 | Preset date and time |
| VAR_OUTPUT | Name | Data type | Effective range | Initial value | Description |
| Q | Status output | BOOL | [FALSE,TRUE] | FALSE | CDT is TRUE when counting |

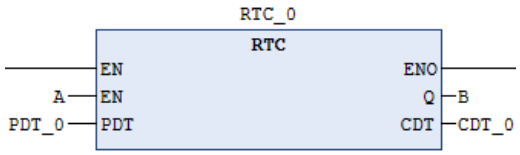
| | | | | | |
|---|---|---|---|---|---|
| CDT | Current counting time and date status | DATE_AND_TIME | (DT#1970-1-1-0:0:0)-(2106-2-7-6:28:15) | DT#1970-1-1-0:0:0 | Current counting time and date status |

Once EN is "True", CDT will be set to the time given by PDT and will be counted in seconds. Once EN is reset to "FALSE" and the output variable Q is "FALSE", CDT will be reset to the initial value DT #1970-01-00:00:00. PDT time is only effective on the rising edge.

<table>
<tr><th>DEMO</th><th>LD</th><th>ST</th></tr>
<tr><td>Define variables</td><td colspan="2">

```
VAR
    RTC_0: RTC;
    A    :BOOL;
    PDT_0:DATE_AND_TIME;
    B    :BOOL;
    CDT_0:DATE_AND_TIME;
END_VAR
```

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




</td><td>

```
1  RTC_0(EN:= A,
2       PDT:= PDT_0,
3       Q=> B,
4       CDT=> CDT_0);
```

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

| 表达式 | 类型 | 值 |
|---|---|---|
| RTC_0 | RTC | |
| A | BOOL | TRUE |
| PDT_0 | DATE_... | DT#1990-1-2-2:0:0 |
| B | BOOL | TRUE |
| CDT_0 | DATE_... | DT#1990-1-2-2:0:2 |

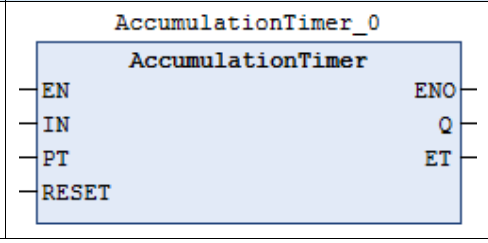
</td></tr>
</table>

## 1-2-5. Accumulation timer

(1) Instruction overview

Accumulation timer.

| Accumulation timer [AccumulationTimer] | | | |
|---|---|---|---|
| Execution condition | Normally ON/OFF triggering | Suitable model | XS, XSA |
| Firmware | - | Software | XS StudioV2.2.0 and above |
| Library | XJ_FunctionUtils | | |

| Instruction | Name | Graph | ST |
|---|---|---|---|
| AccumulationTimer | Accumulated timer |  | AccumulationTimer_0(IN:= ,<br>PT:= ,<br>RESET:= ,<br>Q=> ,<br>ET=> ); |

(2) Related variables

| VAR_INPUT | Name | Data type | Effective range | Initial value | Description |
|---|---|---|---|---|---|
| In | Timer input | BOOL | FALSE/TRUE | FALSE | FALSE: timer stop<br>TRUE: timer working |
| PT | Set time | TIME | As the data type | 0 | Maximum value of timing |
| Reset | Reset | BOOL | FALSE/TRUE | FALSE | FALSE: timer not reset<br>TRUE: timer reset |
| VAR_OUTPUT | Name | Data type | Effective range | Initial value | Description |
| Q | Timer output | BOOL | FASLE/TRUE | - | TRUE: “ET” reached “PT”<br>FALSE: “ET” not reach “PT” |
| ET | Accumulated time | TIME | As the data type | - | Accumulated time |

(3) Function

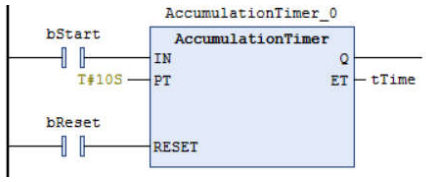
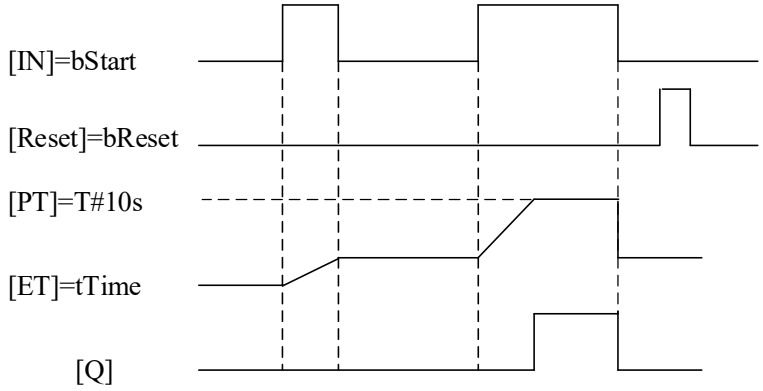
- A timer that accumulates the time when the input "In" is true. Set the time unit to ms and the timing accuracy to 100ns.
- When resetting "Reset" to FALSE and "In" changes from FALSE to True, the timer starts and the cumulative time "ET" increases simultaneously with the time.
- When "In" changes to FALSE, the timer stops. But maintain the "ET" value at that moment.
- When "In" changes to true again, the timer starts again. "ET" increases from the holding value.
- When "ET" reaches the set time "PT", the timer output "Q" changes to True. At this point, "ET" stops increasing.
- When "Reset" becomes true, the timer resets. The value of "ET" becomes 0, and "Q" becomes FALSE.

(4) Note

- "ET" and "Q" are updated during the execution of this command. Therefore, strictly speaking, the condition for "Q" to become true is not the moment when the cumulative time of timer operation equals "PT". But rather the moment when the cumulative time of the timer's operation reaches "PT" and this instruction is first executed.

Therefore, a delay of up to one task cycle will occur.

- PT and ET are set in ms, with a timing accuracy of 100ns.
- When both "In" and "Reset" become true, "Reset" takes priority. The value of "ET" becomes 0, and "Q" becomes FALSE.
- If the value of "In" is already true at the beginning of the run, timing will start from that moment.
- When T#0ms is set for "PT", the value of "In" becomes true, and "Q" becomes true.
- Before the value of "ET" reaches the value of "PT", the value of "PT" can be changed, but the cumulative timing "ET" is still calculated according to the original "PT".

<table>
<tr><th>DEMO</th><th>LD</th><th>ST</th></tr>
<tr><td>Define variables</td><td colspan="2"><pre>VAR
    AccumulationTimer_0: AccumulationTimer;
    bStart: BOOL;
    bReset: BOOL;
    tTime: TIME;
    XDONE: BOOL;
END_VAR</pre></td></tr>
<tr><td>Sample Program</td><td> 
</td><td><pre>AccumulationTimer_0(
    IN:= bStart,
    PT:= T#10S,
    RESET:= bReset,
    Q=> XDONE,
    ET=> tTime);</pre></td></tr>
<tr><td>Operation result</td><td colspan="2"> 

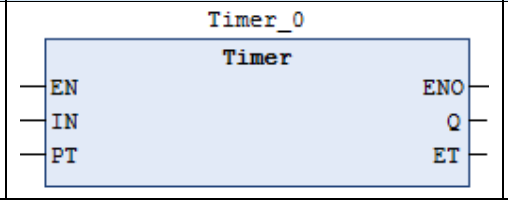
When the cumulative time for variable bStart to be true reaches 10 seconds, the value of variable XDONE becomes true.</td></tr>
</table>

## 1-2-6. 100ms Timer

(1) Instruction overview

100ms timer.

| 100ms timer [Timer] | | | |
|---|---|---|---|
| Execution condition | Normally ON/OFF triggering | Suitable model | XS, XSA |
| Firmware | - | Software | XS StudioV2.2.0 and above |
| Library | XJ_FunctionUtils | | |

| Instruction | Name | Graph | ST |
|---|---|---|---|
| Timer | 100ms timer |  | Timer_0(IN:= ,<br>PT:= ,<br>Q=> ,<br>ET=> ); |

(2) Related variables

| VAR_INPUT | Name | Data type | Effective range | Initial value | Description |
|---|---|---|---|---|---|
| In | Timer input | BOOL | FALSE/TRUE | FALSE | FALSE: timer stop<br>TRUE: timer working |
| PT | Set time | UINT | As the data type | 0 | The maximum value of timing, unit is 100ms |
| VAR_OUTPUT | Name | Data type | Effective range | Initial value | Description |
| Q | Timer output | BOOL | FASLE/TRUE | - | TRUE: “ET” reached “PT”<br>FALSE: “ET” not reach “PT” |
| ET | Remaining time | UINT | As the data type | - | Remaining time |

(3) Function

- After a set time has passed since startup, a timer outputs true. Set the time in 100ms units.
- If the value of "In" input to the timer becomes FALSE, the timer will reset. The remaining time "ET" will be set to the set time "PT", and the value of the timer output "Q" will become FALSE.
- If "In" changes from FALSE to True, start the timer. The value of "ET" decreases over time.
- After "ET" reaches 0, the timer output "Q" becomes True. At this point, the value of "ET" stops decreasing.
- After the timer starts, even if "ET" does not reach 0, if "In" becomes FALSE, the timer will reset.

(4) Note

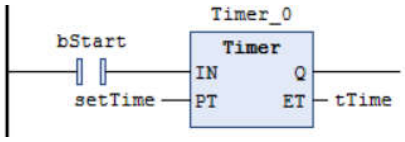
- Calling multiple TIMER within the same POU may cause exceptions in the program.
- The timing error required for "PT" and "Q" to become true is 1 task cycle. The timer determines whether the timing of "ET" for each task cycle has reached "PT". If the timing of "ET" reaches "PT" after judgment, it will delay by 1 task cycle.
- At the beginning of the run, if the value of "In" is already true, start timing immediately.
- When the value of "PT" is changed, it needs to be applied during the next timer reset; It does not take effect during the timing process.
- Manually modifying the values of internal parameter variables (ET, Q) cannot take effect.

(5) Program example

<table>
<tr><th>DEMO</th><th>LD</th><th>ST</th></tr>
<tr><td>Define variables</td><td colspan="2">

```
VAR
   Timer_0: Timer;
   bStart: BOOL;
   tTime: UINT;
   setTime: UINT:= 100;
   XDONE: BOOL;
END_VAR
```

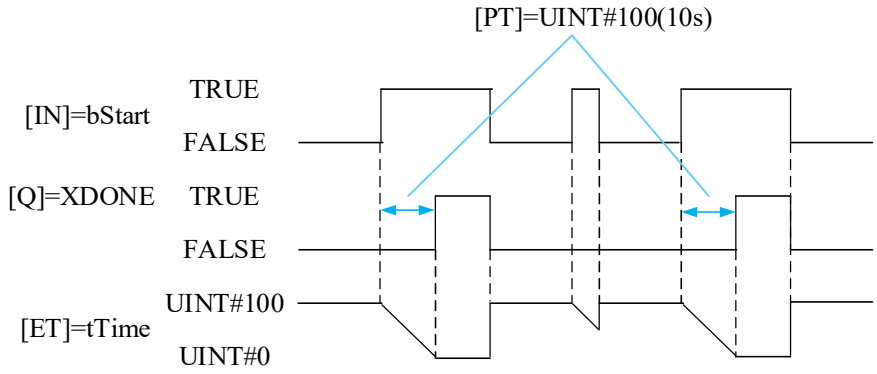
</td></tr>
<tr><td>Program example</td><td>




</td><td>

```
Timer_0(
    IN:= bStart,
    PT:= setTime,
    Q=> XDONE,
    ET=> tTime);
```

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




After variable bStart becomes True for 100*100ms (10s), the value of variable XDONE becomes True.

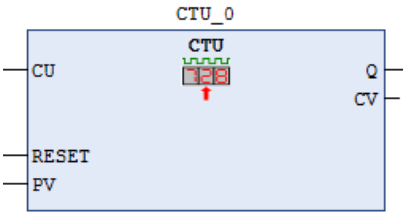
</td></tr>
</table>

# 1-3. Counter

## 1-3-1. Up counter CTU

When the rising edge signal is triggered once, the count value is increased by 1.

| Up counter [CTU] | | | |
|---|---|---|---|
| Execution condition | Edge triggering | Suitable model | XS, XSA |
| Firmware | - | Software | - |
| Library | Standard | | |

| Command name | Command icon | Function |
|---|---|---|
| CTU |  | CTU_0(CU:= ,<br>RESET:= ,<br>PV:= ,<br>Q=> ,<br>CV=> ); |

| VAR_INPUT | Name | Data type | Effective range | Initial value | Description |
|---|---|---|---|---|---|
| CU | Counting signal input | BOOL | [FALSE,TRUE] | FALSE | Rising edge: CV count value plus 1 |
| RESET | Reset input | BOOL | [FALSE,TRUE] | FALSE | TRUE: CV reset to 0 |
| PV | Set upper limit for counting | WORD | 0-65535 | 0 | Set upper limit for counting |
| VAR_OUTPUT | Name | Data type | Effective range | Initial value | Description |
| Q | Status output | BOOL | [FALSE,TRUE] | FALSE | When CV>=PV, it is TRUE |
| CV | Current counting value | WORD | 0-65535 | 0 | Current counting value |

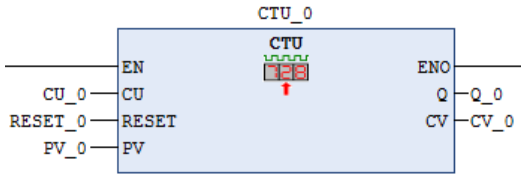
When the signal of CU changes from "FALSE" to "True", the current calculated value is increased by 1 and displayed through the output terminal CV. When the count reaches the upper limit of 65535, the counter will not increase again.

When the signal state of RESET is "True", the CV of the counter is initialized to 0 and Q is "FALSE", and the rising edge no longer has any effect on CU.

When the CV value is greater than or equal to PV, the output Q is "True". At this point, CV can still continue to accumulate, and the output Q continues to be output "True".

<table>
<tr><th>DEMO</th><th>LD</th><th>ST</th></tr>
<tr><td>Define variables</td><td colspan="2">

```
VAR
    CU_0    :BOOL;
    RESET_0:BOOL;
    PV_0    :WORD := 3;
    Q_0     :BOOL;
    CV_0    :WORD;
    CTU_0   : CTU;
END_VAR
```

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

```
1  CTU_0(CU:= CU_0,
2       RESET:= RESET_0,
3       PV:= PV_0,
4       Q=> Q_0,
5       CV=> CV_0);
```

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

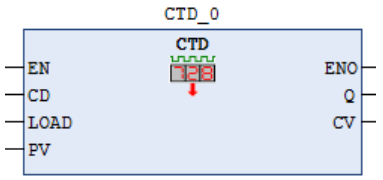
| 表达式 | 类型 | 值 |
|---|---|---|
| CU_0 | BOOL | TRUE |
| RESET_0 | BOOL | FALSE |
| PV_0 | WORD | 3 |
| Q_0 | BOOL | TRUE |
| CV_0 | WORD | 3 |
| ⊞ CTU_0 | CTU | |

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

## 1-3-2. Down counter CTD

When the rising edge signal is applied once, the count decreases by 1.

| Down counter [CTD] | | | |
|---|---|---|---|
| Execution condition | Edge triggering | Suitable model | XS, XSA |
| Firmware | - | Software | - |
| Library | Standard | | |

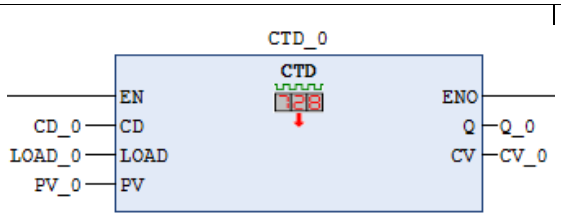
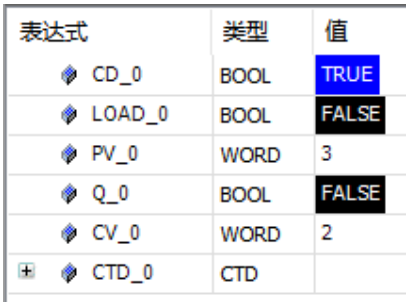
| Command name | Command icon | Function |
|---|---|---|
| CTD |  | CTD_0(CD:= ,<br>LOAD:= ,<br>PV:= ,<br>Q=> ,<br>CV=> ); |

| VAR_INPUT | Name | Data type | Effective range | Initial value | Description |
|---|---|---|---|---|---|
| CU | Counting signal input | BOOL | [FALSE,TRUE] | FALSE | Rising edge: CV count value minus 1 |
| LOAD | Load signal | BOOL | [FALSE,TRUE] | FALSE | True: Set CV as the initial value PV |
| PV | Decline starting value | WORD | 0-65535 | 0 | Decline starting value |
| VAR_OUTPUT | Name | Data type | Effective range | Initial value | Description |
| Q | Status output | BOOL | [FALSE,TRUE] | FALSE | CV=0, it is TRUE |
| CV | Current counting value | WORD | 0-65535 | 0 | Current counting value |

The load signal needs to be changed from "FALSE" to "True", CV will be set to the starting value given by PV, output Q will be changed from the initial "True" to "FALSE", and the load will be changed to the state "FALSE" before the function block can take effect.

When the CD signal at the input end of the counter changes from "FALSE" to the state "True", the current count value is subtracted by 1 and displayed on the output end CV. When the value of CV decreases to 0, Q will be set to True, and the count value will no longer decrease.

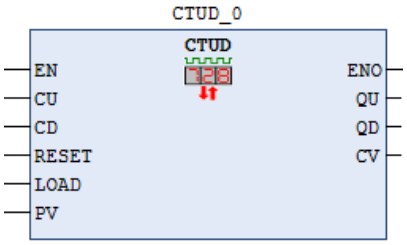
| DEMO | LD | ST |
|---|---|---|
| Define variables | VAR<br>CD_0 :BOOL;<br>LOAD_0 :BOOL;<br>PV_0 :WORD := 3;<br>Q_0 :BOOL;<br>CV_0 :WORD;<br>CTD_0 : CTD;<br>END_VAR | |

| | | |
|---|---|---|
| Program example |  | 1 CTD_0(CD:= CD_0,<br>2 LOAD:= LOAD_0,<br>3 PV:= PV_0,<br>4 Q=> Q_0,<br>5 CV=> CV_0); |
| Operation result |  | |
| Result analysis | After setting LOAD_0 to True, Q_0 changes from True to FALSE and CV_0 becomes 3. Reset LOAD_0 to FALSE again. At this point, CD_0 is conducting, and the value of CV_0 is reduced by 1. | |

## 1-3-3. Up/down counter CTUD

You can increase the count by 1, or decrease the count by 1.

| Up/Down counter [CTUD] | | | |
|---|---|---|---|
| Execution condition | Edge triggering | Suitable model | XS, XSA |
| Firmware | - | Software | - |
| Library | Standard | | |

| Command name | Command icon | Function |
|---|---|---|
| CTUD |  | CTUD_0(<br>CU:= ,<br>CD:= ,<br>RESET:= ,<br>LOAD:= ,<br>PV:= ,<br>QU=> ,<br>QD=> ,<br>CV=> ); |

| VAR_INPUT | Name | Data type | Effective range | Initial value | Description |
|---|---|---|---|---|---|
| CU | Count up input | BOOL | [FALSE,TRUE] | FALSE | Rising edge: CV count value plus 1 |
| CD | Count down input | BOOL | [FALSE,TRUE] | FALSE | Rising edge: CV count value minus 1 |
| RESET | Reset input | BOOL | [FALSE,TRUE] | FALSE | TRUE: CV reset to 0 |
| LOAD | Load signal | BOOL | [FALSE,TRUE] | FALSE | True: Set CV as the initial value PV |
| PV | Set upper limit for counting | WORD | 0-65535 | 0 | Set upper limit for counting |
| VAR_OUTPUT | Name | Data type | Effective range | Initial value | Description |
| QU | Increase count status output | BOOL | [FALSE,TRUE] | FALSE | CV>=PV, it is TRUE |
| QD | Countdown status output | BOOL | [FALSE,TRUE] | FALSE | CV=0, it is TRUE |
| CV | Current counting value | WORD | 0-65535 | 0 | Current counting value |

When the CU signal at the count input changes from "FALSE" to the state "True", the current count value is increased by 1 and displayed on the output CV.

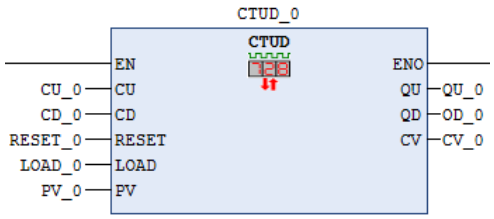
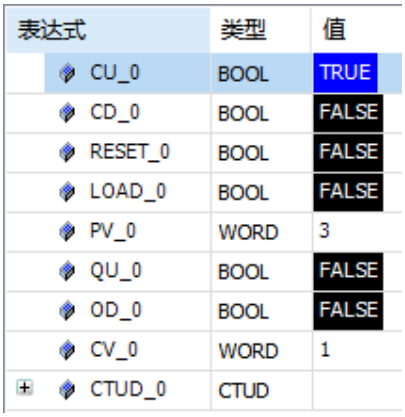
When the signal of the CD at the counting input end changes from "FALSE" to the state "True" (the loading signal load needs to be changed from "FALSE" to "True" and then to the state "FALSE" before the function block can take effect), the current count value is subtracted by 1 and displayed on the output end CV. If both inputs have rising edges, the current count value will remain unchanged.

When the counting value reaches the upper limit value of 65535, the rising edge of the counting input CU no longer works. Therefore, even if there is a rising edge on the counting input CU, its value will not increase. Similarly, when the count value reaches the lower limit value of 0, the counting input CD will not take effect. Therefore, even if the counting input CD shows an upward edge, the count value will not decrease.

When the CV value is greater than or equal to the PV value, the output QU is "True". When the CV value is equal to 0, the output QD is "True".

| DEMO | LD | ST |
|---|---|---|
| Define variables | | |

```
VAR
    CU_0    :BOOL;
    CD_0    :BOOL;
    RESET_0:BOOL;
    LOAD_0 :BOOL;
    PV_0    :WORD := 3;
    QU_0    :BOOL;
    OD_0    :BOOL;
    CV_0    :WORD;
    CTUD_0: CTUD;
END_VAR
```

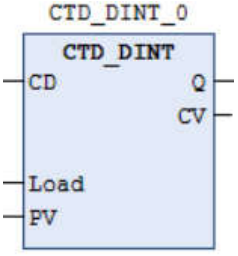
| | | |
|---|---|---|
| Program example |  | 1 CTUD_0(<br>2 CU:= CU_0,<br>3 CD:= CD_0,<br>4 RESET:= RESET_0,<br>5 LOAD:= LOAD_0,<br>6 PV:= PV_0,<br>7 QU=> QU_0,<br>8 QD=> OD_0,<br>9 CV=> CV_0); |
| Operation result |  | |

## 1-3-4. Subtract counter CTD_XXX

(1) Instruction overview

A counter that performs subtraction operations when inputing signal of the counter. The data types for preset values and count values are any of DINT, LINT, UDIND, or ULINT.

| Subtract counter [CTD_XXX] | | | |
|---|---|---|---|
| Execution condition | Edge triggering | Suitable model | XS, XSA |
| Firmware | - | Software | XS StudioV2.2.0 and above |
| Library | XJ_FunctionUtils | | |

| Instruction | Name | Graph | ST |
|---|---|---|---|
| CTD_XXX | Subtract counter |  | CTD_DINT_0(CD:= ,<br>Load:= ,<br>PV:= ,<br>Q=> ,<br>CV=> ); |

(2) Related variables

| VAR_INPUT | Name | Data type | Effective range | Initial value | Description |
|---|---|---|---|---|---|
| CD | Counter input | BOOL | FALSE/TRUE | FALSE | Counter input |
| Load | Load signal | BOOL | FALSE/TRUE | FALSE | When set TRUE, load PV to CV |
| PV | Preset value | DINT/LINT/ UDINT/ULINT | As the data type | 0 | Preset value of the counter |
| VAR_OUTPUT | Name | Data type | Effective range | Initial value | Description |
| Q | Counter output | BOOL | FALSE/TRUE | FALSE | TRUE: counter outputs ON<br>FALSE: counter outputs OFF |
| CV | Counting value | DINT/LINT/UDINT/ULINT | As the data type | - | Counter present value |

(3) Function

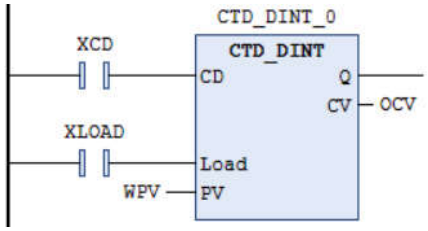
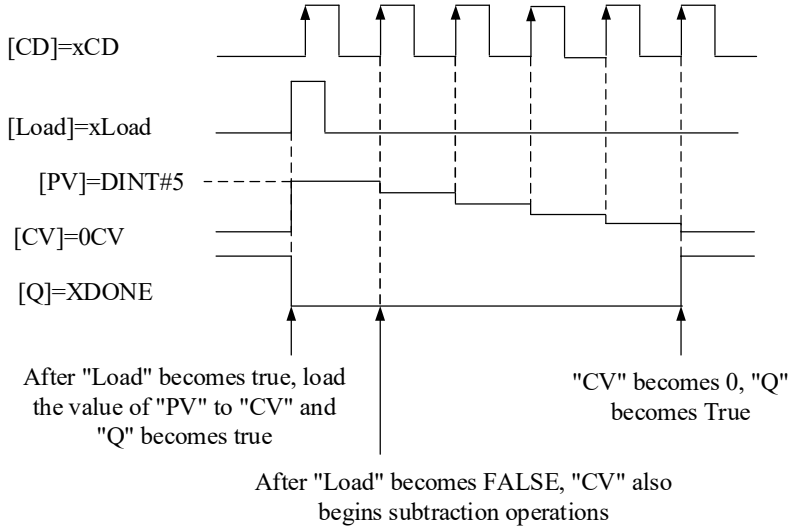
- Subtraction counter. The data types for preset values and count values are any of DINT, LINT, UDINT, or ULINT.
- The instruction name varies depending on the data type of "PV" and "CV". For example, when both are DINT type, the instruction name is CTD_ DINT.
- When the load signal "Load" is set to True, the preset value "PV" is loaded into the count value "CV", and the counter output "Q" becomes FALSE; When the load signal "Load" is set to True and the "PV" value is 0, the counter output "Q" becomes True.
- When the counter input signal "CD" is on the rising edge, "CV" will perform subtraction operation. When the value of "CV" is equal to 0, the value of "Q" becomes True. When the value of "CV" is manually modified to be less than 0, "Q" becomes True. In addition, the "Q" value cannot be modified during operation.
- When the value of "CV" is less than 0, even if "CD" is on the rising edge, "CV" does not perform subtraction operations.
- During the period when "Load" is true, "CV" cannot be modified and subtraction operations are not performed.

(4) Note

- When starting the counter again after the countdown is completed, please set the value of "Load" to True before setting it to FALSE.
- Please set the data types for "PV" and "CV" to be the same.
- When a negative number is set for "PV", the value of "PV" is loaded into "CV" when the value of "Load" becomes true. The value of "CV" is less than 0, so the value of "Q" will immediately become true. Afterwards, even if "CD" changes, "CV" does not perform subtraction operations.
- If the value of "PV" exceeds the range of values, "PV" will be implicitly converted to an error value, and the function will not have any exceptions.
- When the power is disconnected or the program controls the action mode in the state where the value of "CD" is FALSE, and the instruction is restarted, if the value of "CD" becomes true, "CV" performs one subtraction operation.
- When using this instruction through a ladder program, if an abnormality occurs in the front section of the circuit of this instruction, the value of "Q" will become FALSE.

(5) Program example

| DEMO | LD | ST |
|---|---|---|

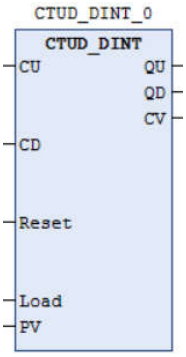
| | |
|---|---|
| Define variables | VAR<br>CTD_DINT_0: CTD_DINT;<br>XCD: BOOL;<br>XLOAD: BOOL;<br>WPV: DINT:= 5;<br>OCV: DINT;<br>XDONE: BOOL;<br>END_VAR |
| Sample program |  <br>CTD_DINT_0(<br>CD:= XCD,<br>Load:= XLOAD,<br>PV:= WPV,<br>Q=> XDONE,<br>CV=>OCV ); |
| Principle analysis |  |

## 1-3-5. Up/Down Counter CTUD_XXX

(1) Instruction overview

A counter that performs addition and subtraction operations based on the input of the addition counter and the subtraction counter. The data types for preset values and count values are any of DINT, LINT, UDIND, or ULINT.

| Up/Down counter [CTUD_XXX] | | | |
|---|---|---|---|
| Execution condition | Edge triggering | Suitable model | XS, XSA |
| Firmware | - | Software | XS StudioV2.2.0 and above |
| Library | XJ_FunctionUtils | | |

| Instruction | Name | Graph | ST |
|---|---|---|---|
| CTUD_XXX | Up/down counter |  | CTUD_DINT_0(<br>CU:= ,<br>CD:= ,<br>Reset:= ,<br>Load:= ,<br>PV:= ,<br>QU=> ,<br>QD=> ,<br>CV=> ); |

(2) Related variables

| VAR_INPUT | Name | Data type | Effective range | Initial value | Description |
|---|---|---|---|---|---|
| CU | Addition counter input | BOOL | FALSE/TRUE | FALSE | Addition counter input |
| CD | Subtraction counter input | BOOL | FALSE/TRUE | FALSE | Subtraction counter input |
| Reset | Reset signal | BOOL | FALSE/TRUE | FALSE | When set to true, reset 'CV' to 0 |
| Load | Load signal | BOOL | FALSE/TRUE | FALSE | When set to true, load 'PV' to 'CV' |
| PV | Preset value | DINT/LINT/ UDINT/ULINT | As the data type | 0 | The preset value of the counter |
| VAR_OUTPUT | Name | Data type | Effective range | Initial value | Description |
| QU | Addition counter output | BOOL | FALSE/TRUE | FALSE | TRUE: up counter output ON<br>FALSE: up counter output OFF |
| QD | Subtraction counter output | BOOL | FALSE/TRUE | FALSE | TRUE: down counter output ON<br>FALSE: down counter output OFF |
| CV | Count value | DINT/LINT/ UDINT/ULINT | As the data type | - | The current value of the counter |

(3) Function

- A counter that performs addition and subtraction operations based on the input signals of the addition counter

and the subtraction counter.

- It has the function of both an addition counter and a subtraction counter.
- The data types for preset values and count values are any of DINT, LINT, UDINT, or ULINT.
- The instruction name varies depending on the data type of "PV" and "CV". For example, when both are LINT type, the instruction name is CTUD_Lint.
- The function of the addition counter
  - When the reset signal "Reset" is set to True, the value of the count value "CV" becomes 0, and the output of the addition counter "QU" becomes FALSE.
  - When the input signal "CU" of the addition counter is on the rising edge, make "CV" perform the addition operation. When the value of "CV" is equal to the preset value of "PV", the value of "QU" becomes true. If the value of "CV" is manually modified to be greater than the value of "PV", the value of "QU" becomes true.
  - When the value of "CV" is greater than the value of "PV", even if the input signal "CU" is triggered again, "CV" will not change.
  - When "CV" is less than or equal to 0, "QD" becomes True.
- The function of the subtraction counter
  - When the load signal "Load" is set to True, the preset value "PV" is loaded into the count value "CV", and the subtraction counter output "QD" becomes FALSE.
  - When the input signal "CD" of the subtraction counter is on the rising edge, subtract "CV". When the value of "CV" is equal to 0, the value of "QD" becomes true. Manually modify the value of "CV" to be greater than or equal to the value of "PV", and the value of "QU" becomes true. Manually modify the value of "CV" to be less than 0, and the value of "QD" becomes true.
  - When the value of "CV" is less than 0, even if the input signal "CD" is triggered again, "CV" will not change.
- Common functions of addition and subtraction counters
  - During the period when "Load" or "Reset" is true, ignore "CU" and "CD". "CV" does not perform addition and subtraction operations.
  - When "CU" and "CD" are both on the rising edge, "CV" will not change.
  - When both "Reset" and "Load" are true, "Reset" takes priority and the value of "CV" becomes 0.
  - When "Reset" is set to true, the value of "CV" becomes 0, so the value of "QD" becomes true.
  - When "Load" is set to True, the value of "CV" is equal to "PV", so the value of "QU" becomes True.

| “Reset” | “Load” | “CV” | “QU” | “QD” | Action |
|---|---|---|---|---|---|
| FALSE | FALSE | Below 0 | FALSE | TRUE | Only perform addition counter actions<br>When "CU" is on the rising edge, "CV" performs addition operations<br>When CD is on the rising edge, no subtraction operation is performed |
| | | Between 0 and “PV” | FALSE | FALSE | Perform addition/subtraction counter actions<br>When "CU" is on the rising edge, "CV" performs addition operations<br>When CD is on the rising edge, CV performs subtraction operation |
| | | Larger than “PV” | TRUE | FALSE | Only perform subtraction counter action<br>When "CU" is on the rising edge, no addition operation is performed<br>When CD is on the rising edge, CV performs subtraction operation |
| TRUE | FALSE | 0 | FALSE | TRUE | |

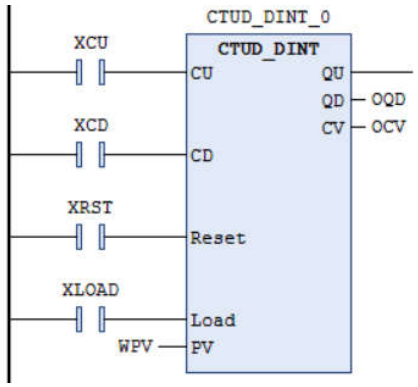
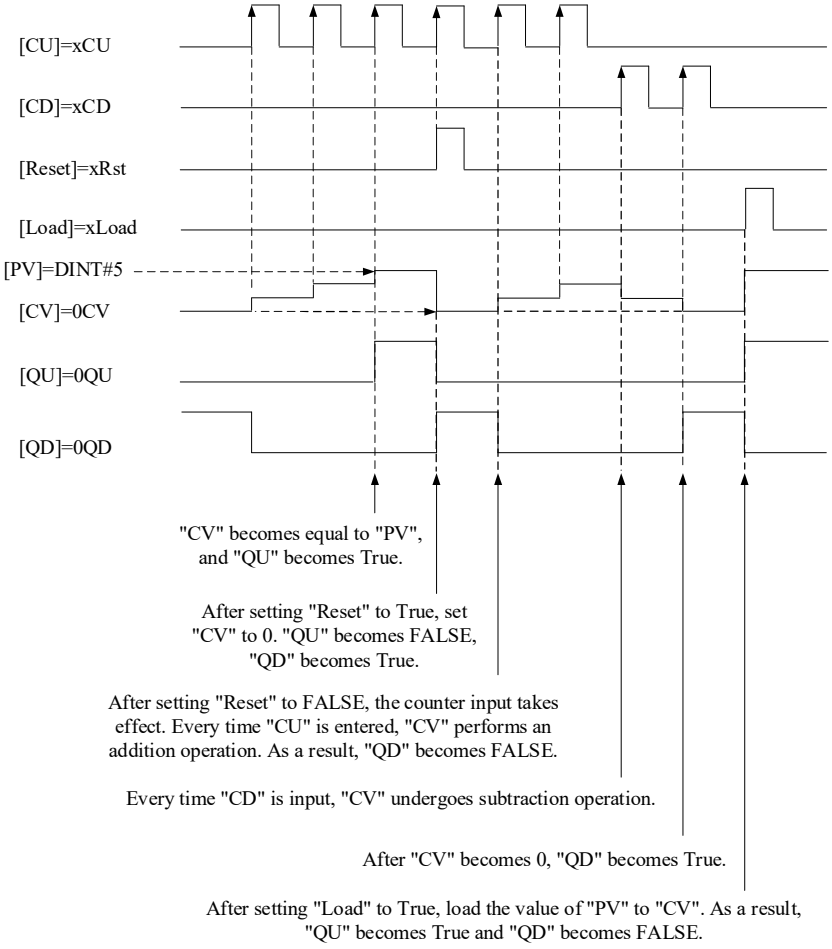
| | | | | | |
|---|---|---|---|---|---|
| FALSE | TRUE | “PV” | TRUE | FALSE | The reset state of the addition counter, with the value of "CV" set to 0<br>Reset state of subtraction counter, set the value of "CV" to "PV" |
| TRUE | TRUE | 0 | FALSE | TRUE | The reset status of the addition counter,<br>"Reset" takes priority over "Load", and the value of "CV" is set to 0 |

(4) Note

- Please note that to reset the addition counter, when the value of "Reset" is set to True, the value of "QU" becomes FALSE, and the value of "QD" becomes True.
- Please note that to reset the subtraction counter, when the value of "Load" is set to True, the value of "QD" becomes FALSE, and the value of "QU" becomes True.
- When a negative number is set for "PV", the value of "PV" is loaded into "CV" when the value of "Load" becomes true. The value of "CV" is less than 0, so the value of "QD" will immediately become true. Afterwards, even if "CD" changes, "CV" does not perform subtraction operations. The value of "Reset" becomes True, and the value of "CV" becomes 0. The value of "CV" is greater than the value of "PV", so the value of "QU" will immediately become true. Afterwards, even if "CU" changes, "CV" does not perform addition operations.
- If the value of "PV" exceeds the range of values, "PV" will be implicitly converted to an error value, and the function will not have any exceptions.
- Please set the data types for "PV" and "CV" to be the same.
- When the value of "CU" or "CD" is FALSE and the power is disconnected or the program controls the action mode, and the instruction is restarted, if the value of "CU" or "CD" becomes true, "CV" performs addition or subtraction operation once.
- The "Q" value cannot be modified during operation.
- The preset value of "PV" can be set to a negative value, and after triggering "CU"/"CD", it can be added/subtracted normally until it reaches the value of "PV".

(5) Program example

| DEMO | LD | ST |
|---|---|---|
| Define variables | | VAR<br>CTUD_DINT_0: CTUD_DINT;<br>XCU: BOOL;<br>XCD: BOOL;<br>XRST: BOOL;<br>XLOAD: BOOL;<br>WPV: DINT:= 5;<br>OQD: BOOL;<br>OCV: DINT;<br>OQU: BOOL;<br>END_VAR |

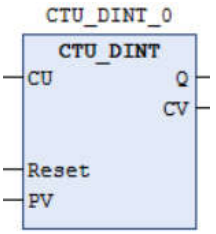
| | | |
|---|---|---|
| Sample program |  | CTUD_DINT_0(<br>CU:= XCU,<br>CD:= XCD,<br>Reset:= XRST,<br>Load:= XLOAD,<br>PV:= WPV,<br>QU=> OQU,<br>QD=> OQD,<br>CV=> OCV); |
| Principle analysis |  | |

## 1-3-6. Up counter CTU_XXX

(1) Instruction overview

A counter that performs addition operations when inputing counter input signal. The data types for preset values and count values are any of DINT, LINT, UDIND, or ULINT.

| Up counter [CTU_XXX] | | | |
|---|---|---|---|
| Execution condition | Edge triggering | Suitable model | XS, XSA |
| Firmware | - | Software | XS StudioV2.2.0 and above |
| Library | XJ_FunctionUtils | | |

| Instruction | Name | Graph | ST |
|---|---|---|---|
| CTU_XXX | Up counter |  | CTU_DINT_0(CU:= ,<br>Reset:= ,<br>PV:= ,<br>Q=> ,<br>CV=> ); |

(2) Related variables

| VAR_INPUT | Name | Data type | Effective range | Initial value | Description |
|---|---|---|---|---|---|
| CU | Counter input | BOOL | FALSE/TRUE | FALSE | Counter input |
| Reset | Load signal | BOOL | FALSE/TRUE | FALSE | When set to true, reset to 0 towards' CV ' |
| PV | Preset value | DINT/LINT/ UDINT/ULINT | As the data type | 0 | The counter preset value |
| VAR_OUTPUT | Name | Data type | Effective range | Initial value | Description |
| Q | Counter output | BOOL | FALSE/TRUE | FALSE | TRUE: counter output ON<br>FALSE: counter output OFF |
| CV | Count value | DINT/LINT/ UDINT/ULINT | As the data type | - | The counter current value |

(3) Function

- Addition counter. The data types for preset values and count values are any of DINT, LINT, UDINT, or ULINT.
- The instruction name varies depending on the data type of "PV" and "CV". For example, when both are DINT type, the instruction name is CTU_DINT.
- When the reset signal "Reset" is set to True, the value of the count value "CV" becomes 0, and the counter output "Q" becomes FALSE.
- When the counter input signal "CU" is on the rising edge, make "CV" perform addition operation. When the value of "CV" is equal to the preset value of "PV", the value of "Q" becomes true. When the value of "CV" is manually modified to be greater than "PV", the value of "Q" becomes true. In addition, during the execution process, the "Q" value cannot be modified.
- When the value of "CV" is greater than the value of "PV", even if a larger value of "CU" is entered, "CV" will not change.

- During the period when "Reset" is true, ignore "CU". CV does not perform addition operations.
- During the period when "Reset" is true, "CV" cannot be modified and does not perform addition operations.

(4) Note

- When starting the counter again after counting, please set the value of "Reset" to True then set to FALSE.
- When a negative number is set for "PV", when the value of "Reset" becomes true, the value of "CV" becomes 0. The value of "CV" is greater than the value of "PV", so the value of "Q" will immediately become true. Afterwards, even if "CU" changes, "CV" does not perform addition operations.
- If the value of "PV" exceeds the range of values, "PV" will be implicitly converted to an error value, and the function will not have any exceptions.
- Please set the data types for "PV" and "CV" to be the same.
- When the value of "Reset" is FALSE, if the value of "PV" changes, its action is shown in the following figure:

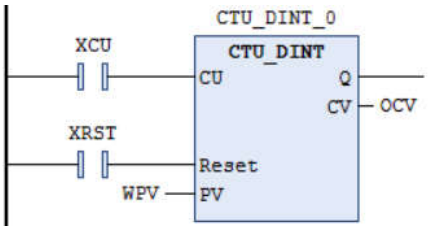
| “PV” value | Meaning |
|---|---|
| "CV" greater than that moment | Continue counting |
| "CV" less than that moment | Count ended. The value of "Q" becomes true, while the value of "CV" remains unchanged at that moment |

- When the value of "CU" is in the FALSE state and the power is disconnected or the program control action mode, when this instruction is restarted, if the value of "CU" becomes true, "CV" performs addition operation once.
- When using this instruction through a ladder program, if an abnormality occurs in the front section of the circuit of this instruction, the value of "Q" will become FALSE.

(5) Program example

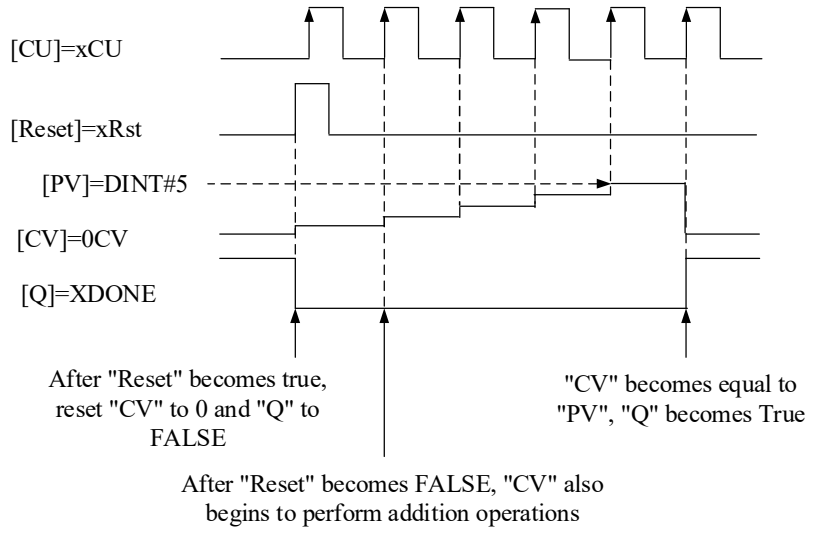
<table>
<tr><th>DEMO</th><th>LD</th><th>ST</th></tr>
<tr><td>Define variables</td><td colspan="2">

```
VAR
   CTU_DINT_0: CTU_DINT;
   XCU: BOOL;
   XRST: BOOL;
   WPV: DINT:= 5;
   OCV: DINT;
   XDONE: BOOL;
END_VAR
```

</td></tr>
<tr><td>Sample program</td><td> 
</td><td>

```
CTU_DINT_0(
    CU:= XCU,
    Reset:= XRST,
    PV:= WPV,
    Q=> XDONE,
    CV=> OCV);
```

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

| | |
|---|---|
| Principle analysis |  |

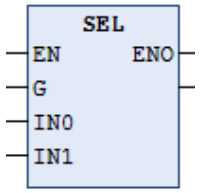
# 1-4. Data Processing Instructions

## 1-4-1. Selection commands

### 1-4-1-1. Binary choice instruction SEL

When the selection switch is FALSE, the output is the first input data, and when the selection switch is True, the output is the second data.

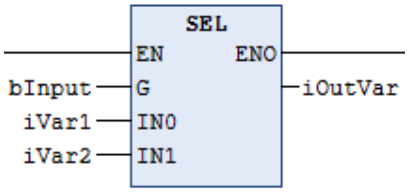
| Binary choice [SEL] | | | |
|---|---|---|---|
| Execution condition | Edge triggering | Suitable model | XS, XSA |
| Firmware | - | Software | - |
| Library | - | | |

| Command | Command icon | Function |
|---|---|---|
| SEL |  | SEL |

| VAR_INPUT | Name | Data type | Effective range | Initial value | Description |
|---|---|---|---|---|---|
| G | Input signal | BOOL | [FALSE,TRUE] | FALSE | FALSE: output IN0 data<br>TRUE: output IN1 data |
| IN0 | Input variable | Any type | As the data type | 0 | G is FALSE: output IN0 data |
| IN1 | Input variable | Any type | As the data type | 0 | G is TRUE: output IN1 data |
| VAR_OUTPUT | Name | Data type | Effective range | Initial value | Description |
| OUT | Output variable | Any type | As the data type | 0 | Output variable |

By selecting a switch, select one of the two input data as the output. If G is FALSE, the result of the return value is IN0; if G is True, the result of the return value is IN1.

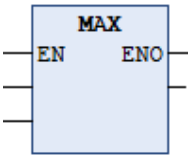
| DEMO | LD | ST |
|---|---|---|
| Define the variable | VAR<br>iVar1 :REAL:=3.3;<br>iVar2 :REAL:=3.4;<br>iOutVar:REAL;<br>bInput :BOOL;<br>END_VAR | |

| DEMO | LD | ST |
|---|---|---|
| Sample program |  | 1 iOutVar:=SEL(bInput,iVar1,iV |
| Operation result | | |
| Result analysis | When the input value bInput is FALSE, the output is 3.3. Conversely, when it is true, the output is 3.4. | |

1-4-1-2. Take the maximum value MAX

Choose the maximum value between two values to output.

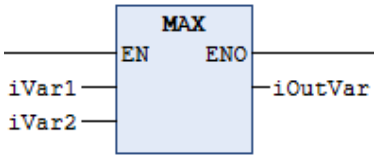
| Take the maximum value [MAX] | | | |
|---|---|---|---|
| Execution condition | Edge triggering | Suitable model | XS, XSA |
| Firmware | - | Software | - |
| Library | - | | |

| Command | Command icon | Function |
|---|---|---|
| MAX |  | MAX |

| VAR_INPUT | Name | Data type | Effective range | Initial value | Description |
|---|---|---|---|---|---|
| IN0 | Input variable | Any type | As the data type | 0 | Input data 0 |
| INn | Input variable | Any type | As the data type | 0 | Input data 1 |
| VAR_OUTPUT | Name | Data type | Effective range | Initial value | Description |

| | | | | | |
|---|---|---|---|---|---|
| OUT | Output variable | Any type | As the data type | 0 | Output variable |

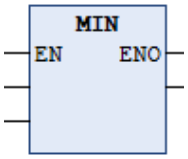
Given two values, take the maximum value between them.

| DEMO | LD | ST |
|---|---|---|
| Define the variable | VAR<br>iVar1 :INT:=30;<br>iVar2 :INT:=60;<br>iOutVar:INT;<br>END_VAR | |
| Sample program |  | 1 iOutVar:=MAX(iVar1,iVar2); |
| Operation result | | |

1-4-1-3. Take the minimum value MIN

Choose the minimum value between two values to output.

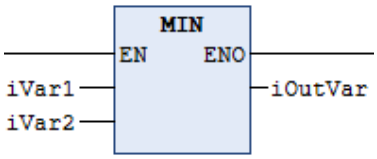
| Take the minimum value [MIN] | | | |
|---|---|---|---|
| Execution condition | Edge triggering | Suitable model | XS, XSA |
| Firmware | - | Software | - |
| Library | - | | |

| Command | Command icon | Function |
|---|---|---|
| MIN |  | MIN |

| VAR_INPUT | Name | Data type | Effective range | Initial value | Description |
|---|---|---|---|---|---|
| IN0 | Input variable | Any type | As the data type | 0 | Input data 0 |
| INn | Input variable | Any type | As the data type | 0 | Input data 1 |

| VAR_OUTPUT | Name | Data type | Effective range | Initial value | Description |
|---|---|---|---|---|---|
| OUT | Output variable | Any type | As the data type | 0 | Output variable |

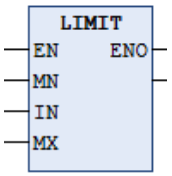
Given two values, take the minimum value between them.

| DEMO | LD | ST |
|---|---|---|
| Define the variable | VAR<br>iVar1 :INT:=30;<br>iVar2 :INT:=60;<br>iOutVar:INT;<br>END_VAR | |
| Sample program |  | 1 iOutVar:=MIN(iVar1,iVar2); |
| Operation result | | |

1-4-1-4. Limit value LIMIT

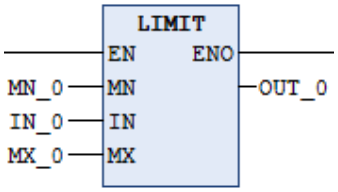
If the input data is between the upper and lower limit values, output the value. If it is below the lower limit value, output the lower limit value. If it is above the upper limit value, output the upper limit value.

| Limit value [LIMIT] | | | |
|---|---|---|---|
| Execution condition | Edge triggering | Suitable model | XS, XSA |
| Firmware | - | Software | - |
| Library | - | | |

| Command | Command icon | Function |
|---|---|---|
| LIMIT |  | LIMIT |

| VAR_INPUT | Name | Data type | Effective range | Initial value | Description |
|---|---|---|---|---|---|
| MN | Lower limit value | Any type | As the data type | 0 | Minimum value |
| IN | Input variable | Any type | As the data type | 0 | Input data |
| MX | Upper limit value | Any type | As the data type | 0 | Maximum value |
| VAR_OUTPUT | Name | Data type | Effective range | Initial value | Description |
| OUT | Output variable | Any type | As the data type | 0 | Output variable |

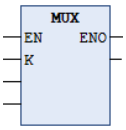
Limit value output. Determine if the input data is between the minimum and maximum values. If the input data is between the two, output it directly as output data. When the input data is greater than the maximum value, the maximum value is taken as the output value. When the input data is less than the minimum value, the minimum value is used as the output value.

| DEMO | LD | ST |
|---|---|---|
| Define the variable | VAR<br>MN_0 :INT:=20;<br>IN_0 :INT:=90;<br>MX_0 :INT:=60;<br>OUT_0:INT;<br>END_VAR | |
| Sample program |  | 1 OUT_0:=LIMIT(MN_0,IN_0,MX_0); |
| Operation result | | |

1-4-1-5. Multiple choice MUX

MUX selects the Kth value from a set of values. The first value is K=0. If K is greater than the number of other inputs (n), XS Studio passes the last value.

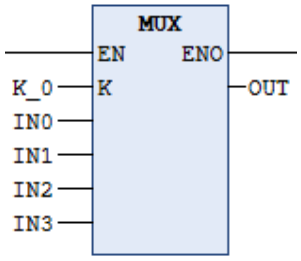
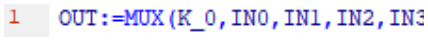
| Multiple choice [MUX] | | | |
|---|---|---|---|
| Execution condition | Edge triggering | Suitable model | XS, XSA |
| Firmware | - | Software | - |
| Library | - | | |

| Command | Command icon | Function |
|---|---|---|
| MUX |  | MUX |

| VAR_INPUT | Name | Data type | Effective range | Initial value | Description |
|---|---|---|---|---|---|
| K | Lower limit value | BYTE, WORD, DWORD, LWORD, SINT, USINT, INT, UINT, DINT, LINT, ULINT, UDINT | As the data type | 0 | Control number |
| IN0 | Input variable | Any type | As the data type | 0 | Input data 0 |
| INn | Upper limit value | Any type | As the data type | 0 | Input data n |
| VAR_OUTPUT | Name | Data type | Effective range | Initial value | Description |
| OUT | Output variable | Any type | As the data type | 0 | Output data |

MUX selects the Kth data output from the variable group.
When the value of K is less than 0 or greater than the number of input variables, output the last variable value.

| DEMO | LD | ST |
|---|---|---|
| Define the variable | | VAR<br>K_0:INT:=2;<br>IN0:INT:=10;<br>IN1:INT:=11;<br>IN2:INT:=20;<br>IN3:INT:=16;<br>OUT:INT;<br>END_VAR |

| | | |
|---|---|---|
| Sample program |  |  |
| Operation result | | |

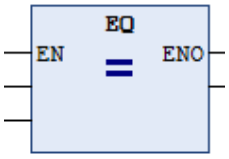
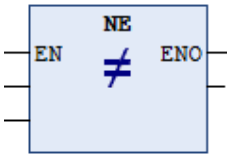
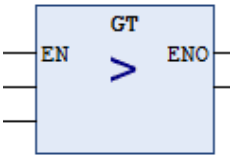
| 表达式 | 类型 | 值 |
|---|---|---|
| K_0 | INT | 2 |
| IN0 | INT | 10 |
| IN1 | INT | 11 |
| IN2 | INT | 20 |
| IN3 | INT | 16 |
| OUT | INT | 20 |

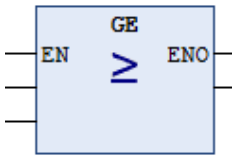
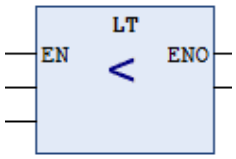
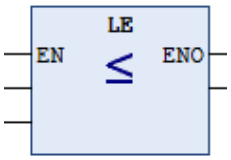
## 1-4-2. Compare Instructions

### 1-4-2-1. Basic compare instruction EQ/NE/GT/GE/LT/LE

Data size comparison instruction.

| Basic compare instruction [EQ/NE/GT/GE/LT/LE] | | | |
|---|---|---|---|
| Execution condition | Edge triggering | Suitable model | XS, XSA |
| Firmware | - | Software | - |
| Library | - | | |

| Command name | Command icon | Function |
|---|---|---|
| EQ |  | = |
| NE |  | <> |
| GT |  | > |

| Command name | Command icon | Function |
|---|---|---|
| GE |  | >= |
| LT |  | < |
| LE |  | <= |

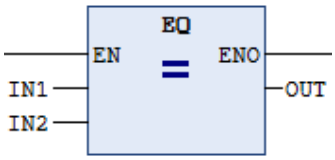
| VAR_INPUT | Name | Data type | Effective range | Initial value | Description |
|---|---|---|---|---|---|
| IN0 | Input variable | Any type | As the data type | 0 | Comparison value |
| IN1 | Input variable | Any type | As the data type | 0 | Comparison value |
| VAR_OUTPUT | Name | Data type | Effective range | Initial value | Description |
| OUT | Compare result | BOOL | [FALSE,TRUE] | 0 | Output data |

Basic comparison instructions comparison situation

| Instruction | Compare | Result |
|---|---|---|
| EQ | IN0=IN1 | TRUE |
| NE | IN0<>IN1 | TRUE |
| GT | IN0>IN1 | TRUE |
| GE | IN0>=IN1 | TRUE |
| LT | IN0<IN1 | TRUE |
| LE | IN0<=IN1 | TRUE |

| DEMO | LD | ST |
|---|---|---|
| Define the variable | | |

```
VAR
    IN1:INT:=10;
    IN2:INT:=10;
    OUT:BOOL;
END_VAR
```

| | | |
|---|---|---|
| Sample program |  | 1 OUT:=IN1=IN2; |
| Operation result | 表达式 / 类型 / 值<br>IN1 / INT / 10<br>IN2 / INT / 10<br>OUT / BOOL / TRUE | |

1-4-2-2. Maximum/minimum value retrieval of arrays AryMax/AryMin

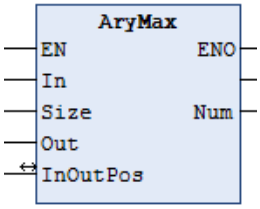
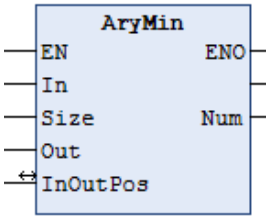
(1) Instruction overview

Retrieve the maximum/minimum values of a 1-dimensional array.

AryMax: Retrieves the maximum value of a 1-dimensional array element.

AryMin: Retrieve the minimum value of a 1-dimensional array element.

| Maximum/minimum value retrieval of arrays [AryMax/AryMin] | | | |
|---|---|---|---|
| Execution condition | Edge triggering | Suitable model | XS, XSA |
| Firmware | - | Software | XS StudioV2.2.0 and above |
| Library | XJ_DataUtils | | |

| Instruction | Name | Graph | ST |
|---|---|---|---|
| AryMax | Maximum value retrieval of arrays |  | Out:=AryMax(In:= ,<br>Size:= ,<br>Out:= ,<br>InOutPos:= ,<br>Num=> ); |
| AryMin | Minimum value retrieval of arrays |  | Out:=AryMin(In:= ,<br>Size:= ,<br>Out:= ,<br>InOutPos:= ,<br>Num=> ); |

(2) Related variables

| VAR_INPUT | Name | Data type | Effective range | Initial value | Description |
|---|---|---|---|---|---|
| In[] array | Retrieve Object Array | USINT/UINT/UDINT /ULINT/SINT/INT/DINT/ LINT/REAL/LREAL/TIME/ DATE/TOD/DT/STRING | As the data type | (*) | Retrieve Object Array |

| | | | | | |
|---|---|---|---|---|---|
| Size | Retrieve the number of elements in an object array | UINT | As the data type | 1 | In [], the number of elements in the retrieval object array |
| Out | Search results | Consistent with In[] element data type | As the data type | (*) | Search results |
| VAR_OUTPUT | Name | Data type | Effective range | Initial value | Description |
| Num | Number of retrieved values | UINT | As the data type | - | Number of retrieved values |
| VAR_IN_OUT | Name | Data type | Effective range | Initial value | Description |
| InOutPos | retrieve elements | UINT | As the data type | - | Array element numbers for retrieved values |

(3) Function

- Retrieve "Size" array elements starting from In[] set in the retrieval object array In.
- Assign the search result to "Out", the offset of the search element from the input element position to "InOutPos", and the number of search values to "Num". When Num is greater than 1, the value of InOutPos becomes the lowest element number in the retrieved value.
- The judgment of the size relationship between values of data types other than integers and real numbers is shown in the table below:

| Data type | Size relationship |
|---|---|
| TIME | The value with a larger value is judged as large |
| DATE, TOD, DT | The latter date or time is judged to be larger |
| STRING | Firstly, compare the character encoding of the first character of each string. If the character encoding is different, the size of the character encoding is the size of the string. If the character encoding of the first character is the same, continue comparing the second and third characters until a different character encoding is found. If the length of each string is different, a NULL character (16#00) will be added at the end of the shorter string based on the longer string, and then compared. |

(4) Note

- The value of "Size" should not exceed the array area of In[].
- In[] must be defined as a one-dimensional array.
- Please ensure that "Out" and the elements of the retrieval object array In[] are of the same data type.
- Retrieve the element number "InOutPos" as the offset of the input element position.
- When In[] is a real number, the expected result may not be obtained due to the influence of errors.
- When the value of "Size" is 0, the value of "Num" becomes 0, the values of "InOutPos" and "Out" remain unchanged, and the function returns a value of True.
- When In[] is an unsupported data type, it cannot be executed properly.
- When In[] and "Out" are STRING types, the area size of In[] should not be greater than that of "Out". Otherwise, the function returns a value of FALSE and "Out" remains unchanged.
- When In[] and "Out" are STRING types, the valid range supports a maximum of 1985 characters. Otherwise,

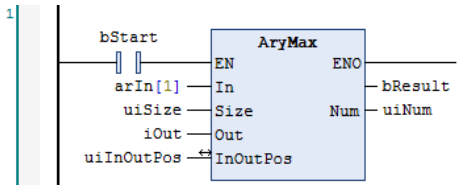
the return value returns FALSE.

- In[] requires integers, real numbers, and date types. Otherwise, the return value returns FALSE, while "InOutPos" and "Out" remain unchanged.

(5) Program example

| DEMO | LD | ST |
|---|---|---|
| Define variables | | |
| Sample program | | |
| Operation result | | |

```
VAR
    bStart:BOOL;
    arIn:ARRAY[0..7]OF INT:=[99,3,6,20,5,11,4,66];
    uiSize    :UINT:=6;
    iOut      :INT;
    uiInOutPos:UINT;
    uiNum     :UINT;
    bResult   :BOOL;
END_VAR
```




```
IF bStart THEN
    bResult:=AryMax(In:= arIn[1],
        Size:= uiSize,
        Out:= iOut,
        InOutPos:= uiInOutPos,
        Num=> uiNum);
    bStart:=FALSE;
END_IF
```

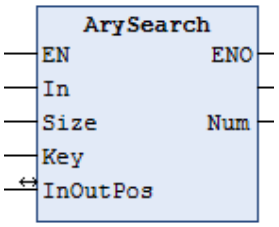
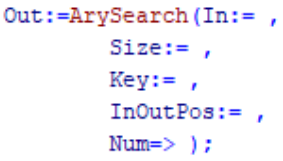
| 表达式 | 类型 | 值 |
|---|---|---|
| bStart | BOOL | FALSE |
| arIn | ARRAY [0..7] OF INT | |
| arIn[0] | INT | 99 |
| arIn[1] | INT | 3 |
| arIn[2] | INT | 6 |
| arIn[3] | INT | 20 |
| arIn[4] | INT | 5 |
| arIn[5] | INT | 11 |
| arIn[6] | INT | 4 |
| arIn[7] | INT | 66 |
| uiSize | UINT | 6 |
| iOut | INT | 20 |
| uiInOutPos | UINT | 2 |
| uiNum | UINT | 1 |
| bResult | BOOL | TRUE |

1-4-2-3. Array retrieval ArySearch

(1) Instruction overview

Retrieve the specified value in a one-dimensional array.

| Array retrieval [ArySearch] | | | |
|---|---|---|---|
| Execution condition | Edge triggering | Suitable model | XS, XSA |
| Firmware | - | Software | XS StudioV2.2.0 and above |
| Library | XJ_DataUtils | | |

| Instruction | Name | Graph | ST |
|---|---|---|---|
| ArySearch | Array retrieval |  |  |





(2) Related variables

| VAR_INPUT | Name | Data type | Effective range | Initial value | Description |
|---|---|---|---|---|---|
| In[] array | Retrieve Object Array | BOOL/BYTE/WORD/DWORD / LWORD/USINT/UINT/UDINT/ ULINT/SINT/INT/DINT/LINT/ REAL/LREAL/TIME/DATE/ TOD/DT/STRING | As the data type | (*) | Retrieve Object Array |
| Size | Retrieve the number of elements in an object array | UINT | As the data type | 1 | Retrieve the number of elements in an object array |
| Key | Search keywords | Consistent with In[] element data type | As the data type | (*) | Retrieve value |
| VAR_OUTPUT | Name | Data type | Effective range | Initial value | Description |
| Num | Number of retrieved values | UINT | As the data type | 0 | Number of retrieved values |
| VAR_IN_OUT | Name | Data type | Effective range | Initial value | Description |
| InOutPos | Retrieve element numbers | UINT | As the data type | - | Retrieved value is the offset value of the input element position |

(3) Function

- Retrieve elements with the same value as the retrieval keyword "Key" from the "Size" array elements starting from the In[] set in the 1-dimensional retrieval object array.
- The values for retrieving element number "InOutPos" and number of retrieved values "Num" are shown in the table below:

| The existence of elements with the same value as "Key" | “InOutPos” | “Num” |
|---|---|---|
| Exist | The offset of the input position of the lowest element in an element with the same value as "Key" | Number of elements with the same value as "Key" |
| Not exist | Not change | 0 |

- The judgment of the size relationship between values of data types other than integers and real numbers is shown in the table below:

| Data type | Size relationship |
|---|---|
| TIME | The one with a larger value is judged as large |
| DATE | The latter one is judged as large |
| STRING | Firstly, compare the character encoding of the first character of each string. If the character encoding is different, the size of the character encoding is the size of the string.<br>If the character encoding of the first character is the same, continue comparing the second and third characters until a different character encoding is found.<br>If the length of each string is different, a NULL character (16#00) will be added at the end of the shorter string based on the longer string, and then compared. |

(4) Note

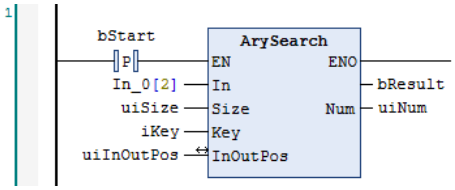
- The value of "Size" should not exceed the array area of In[].
- In[] must be defined as a one-dimensional array.
- Please ensure that "Key" and the elements of the retrieval object array In[] are of the same data type, otherwise the function returns a value of FALSE.
- Please ensure that the input parameter for "Key" is a variable.
- Retrieve the element number "InOutPos" as the offset of the input element position.
- When the value of "Size" is 0, the value of "Num" is initialized to 0, the value of "InOutPos" remains unchanged, and the function returns a value of True.
- When In[] and "Key" are STRING types, the valid range supports a maximum of 1985 characters. When it exceeds the valid range, the value returned by the function is FALSE, and the value of "Num" is initialized to 0.

(5) Program example

<table>
<tr><th>DEMO</th><th>LD</th><th>ST</th></tr>
<tr><td>Define variables</td><td colspan="2">

```
VAR
    bStart     :BOOL;
    In_0       :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;
    bResult    :BOOL;
END_VAR
```

</td></tr>
<tr><td>Sample program</td><td>



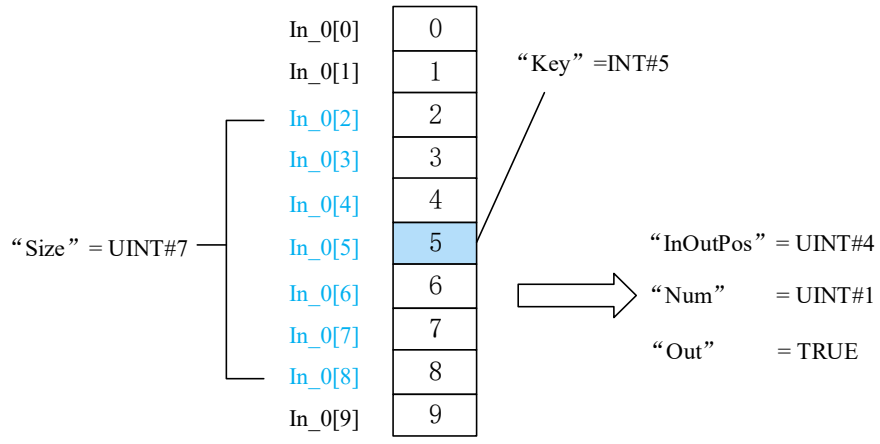

</td><td>

```
IF bStart THEN
    bResult:=ArySearch(In:= In_0[2],
        Size:= uiSize,
        Key:= iKey,
        InOutPos:= uiInOutPos,
        Num=> uiNum);
    bStart:=FALSE;
END_IF
```

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

| 表达式 | 类型 | 值 |
|---|---|---|
| bStart | BOOL | TRUE |
| In_0 | ARRAY [0..9] OF INT | |
| In_0[0] | INT | 0 |
| In_0[1] | INT | 1 |
| In_0[2] | INT | 2 |
| In_0[3] | INT | 3 |
| In_0[4] | INT | 4 |
| In_0[5] | INT | 5 |
| In_0[6] | INT | 6 |
| In_0[7] | INT | 7 |
| In_0[8] | INT | 8 |
| In_0[9] | INT | 9 |
| uiSize | UINT | 7 |
| iKey | INT | 5 |
| uiInOutPos | UINT | 3 |
| uiNum | UINT | 1 |
| bResult | BOOL | TRUE |

</td></tr>
<tr><td>Principle analysis</td><td colspan="2">



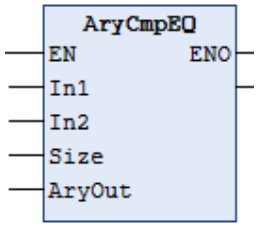

</td></tr>
</table>

1-4-2-4. Batch comparison of arrays EQ AryCmpEQ

(1) Instruction overview

Compare whether the elements of two arrays are equal.

| Batch comparison of arrays EQ [AryCmpEQ] | | | |
|---|---|---|---|
| Execution condition | Edge triggering | Suitable model | XS, XSA |
| Firmware | - | Software | XS StudioV2.2.0 and above |
| Library | XJ_DataUtils | | |

| Instruction | Name | Graph | ST |
|---|---|---|---|
| AryCmpEQ | Batch comparison of arrays EQ |  | bResult:=AryCmpEQ(In1:= ,<br>In2:= ,<br>Size:= ,<br>AryOut:= ); |

(2) Related variables

| VAR_INPUT | Name | Data type | Effective range | Initial value | Description |
|---|---|---|---|---|---|
| In1[] array | Compare array1 | BOOL/BYTE/WORD/DWORD/ LWORD/USINT/UINT/UDINT/ ULINT/SINT/INT/DINT/ LINT/REAL/LREAL | As the data type | - | Compare array1 |
| In2[] array | Compare array2 | An array with the same data type as In1[] | As the data type | - | Compare array2 |
| Size | the compare elements numbers | UINT | As the data type | 1 | the compare elements numbers |
| VAR_IN_OUT | Name | Data type | Effective range | Initial value | Description |
| AryOut[] array | Comparison result array | BOOL | As the data type | - | Comparison result array |

(3) Function

- Compare the Size elements starting from In1[] and the Size elements starting from In2[] of two arrays to see if they are equal. The comparison results are saved to the corresponding element numbers of the consecutive Size elements starting from AryOut[].
- When In1[i]=In2[i], AryOut[i] is True, otherwise it is FALSE.

(4) Note

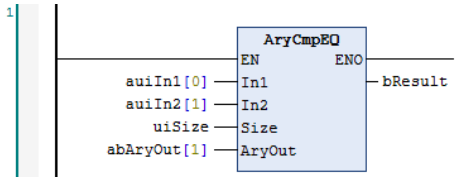
- The data types of In1[] and In2[] should be consistent, otherwise the return value will return FALSE and AryOut[] will remain unchanged.
- When In1[] and In2[] are real numbers and include inexhaustible division results, the processing results may differ from the expected ones due to rounding errors.
- When the value of "Size" is 0, the function returns a value of True and there is no change in AryOut[].
- The value of "Size" should not exceed any array area in In1[], In2[], or AryOut[]

(5) Program example

<table>
<tr><th>DEMO</th><th>LD</th><th>ST</th></tr>
<tr><td>Define variables</td><td colspan="2">

```
VAR
    auiIn1  :ARRAY[0..4] OF UINT:=[0,1,22,57,66];
    auiIn2  :ARRAY[0..4] OF UINT:=[0,99,1,22,33];
    abAryOut:ARRAY[0..4] OF BOOL;
    uiSize  :UINT:=4;
    bResult :BOOL;
END_VAR
```

</td></tr>
<tr><td>Sample program</td><td>




</td><td>

```
1  bResult:=AryCmpEQ(In1:= auiIn1[0],
2      In2:= auiIn2[1],
3      Size:= uiSize,
4      AryOut:= abAryOut[1]);
```

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

| 表达式 | 类型 | 值 |
|---|---|---|
| auiIn1 | ARRAY [0..4] OF UINT | |
| auiIn1[0] | UINT | 0 |
| auiIn1[1] | UINT | 1 |
| auiIn1[2] | UINT | 22 |
| auiIn1[3] | UINT | 57 |
| auiIn1[4] | UINT | 66 |
| auiIn2 | ARRAY [0..4] OF UINT | |
| auiIn2[0] | UINT | 0 |
| auiIn2[1] | UINT | 99 |
| auiIn2[2] | UINT | 1 |
| auiIn2[3] | UINT | 22 |
| auiIn2[4] | 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 | FALSE |
| uiSize | UINT | 4 |
| bResult | BOOL | TRUE |

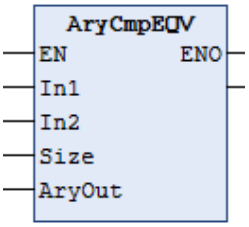
</td></tr>
</table>

1-4-2-5. Comparison of array elements EQV AryCmpEQV

(1) Instruction overview

Compare whether the elements of two arrays are equal.

| Comparison of array elements EQV [AryCmpEQV] | | | |
|---|---|---|---|
| Execution condition | Edge triggering | Suitable model | XS, XSA |
| Firmware | - | Software | XS StudioV2.2.0 and above |
| Library | XJ_DataUtils | | |

| Instruction | Name | Graph | ST |
|---|---|---|---|
| AryCmpEQV | Comparison of array elements EQV |  | bResult:=AryCmpEQV(In1:= ,<br>In2:= ,<br>Size:= ,<br>AryOut:= ); |

(2) Related variables

| VAR_INPUT | Name | Data type | Effective range | Initial value | Description |
|---|---|---|---|---|---|
| In1[] array | Compare array | BOOL/BYTE/WORD/DWORD/ LWORD/USINT/UINT/UDINT/ ULINT/SINT/INT/DINT/ LINT/REAL/LREAL | As the data type | - | Compare array |
| In2[] array | Compare variable | Same data type as In1[] element | As the data type | - | Compare variable |
| Size | Compare element numbers | UINT | As the data type | 1 | Compare element numbers |
| VAR_IN_OUT | Name | Data type | Effective range | Initial value | Description |
| AryOut[] array | Comparison result array | BOOL | As the data type | - | Comparison result array |

Note: In2 can also be defined as a variable.

(3) Function

- Compare the consecutive Size elements starting from array In1[] and variable "In2", whether they are equal (or compare the specified elements of array In1 and array In2). The comparison results are saved to the corresponding element numbers of the consecutive Size elements starting from AryOut[].
- When In1[i]=In2, AryOut[i] is True, otherwise it is FALSE.

(4) Note

- The data types of In1[] and In2 should be consistent, otherwise the return value will return FALSE and AryOut[] will remain unchanged.
- When In1[] and In2 are real numbers and include inexhaustible division results, the processing results may differ from the expected ones due to rounding errors.
- When the value of "Size" is 0, the function returns a value of True and there is no change in AryOut[].

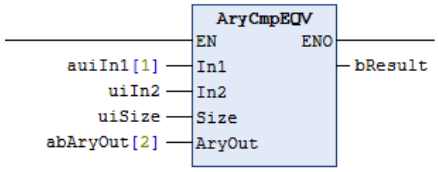
- The value of "Size" should not exceed any array area in In1[] or AryOut[].
- In1[] and In2[] need to use Boolean, bitstring, integer, or real types, otherwise the runtime function returns FALSE and AryOut[] remains unchanged.

(5) Program example

<table>
<tr><th>DEMO</th><th>LD</th><th>ST</th></tr>
<tr><td>Define variables</td><td colspan="2">

```
VAR
    auiIn1  :ARRAY[0..4] OF UINT:=[0,1,22,57,66];
    uiIn2   :UINT:=57;
    abAryOut:ARRAY[0..4] OF BOOL;
    uiSize  :UINT:=3;
    bResult :BOOL;
END_VAR
```

</td></tr>
<tr><td>Sample program</td><td>




</td><td>

```
1  bResult:=AryCmpEQV(In1:= auiIn1[1],
2      In2:= uiIn2,
3      Size:= uiSize,
4      AryOut:= abAryOut[2]);
```

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

| 表达式 | 类型 | 值 |
|---|---|---|
| auiIn1 | ARRAY [0..4] OF UINT | |
| auiIn1[0] | UINT | 0 |
| auiIn1[1] | UINT | 1 |
| auiIn1[2] | UINT | 22 |
| auiIn1[3] | UINT | 57 |
| auiIn1[4] | UINT | 66 |
| uiIn2 | UINT | 57 |
| abAryOut | ARRAY [0..4] OF BOOL | |
| abAryOut[0] | BOOL | FALSE |
| abAryOut[1] | BOOL | FALSE |
| abAryOut[2] | BOOL | FALSE |
| abAryOut[3] | BOOL | FALSE |
| abAryOut[4] | BOOL | TRUE |
| uiSize | UINT | 3 |
| bResult | BOOL | TRUE |

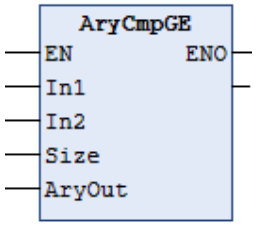
</td></tr>
</table>

1-4-2-6. Batch comparison of arrays GE AryCmpGE

(1) Instruction overview

Compare the size of each element of two arrays (greater than or equal to).

| Batch comparison of arrays GE [AryCmpGE] | | | |
|---|---|---|---|
| Execution condition | Edge triggering | Suitable model | XS, XSA |
| Firmware | - | Software | XS StudioV2.2.0 and above |
| Library | XJ_DataUtils | | |

| Instruction | Name | Graph | ST |
|---|---|---|---|
| AryCmpGE | Batch comparison of arrays GE |  | bResult:=AryCmpGE(In1:= ,<br>In2:= ,<br>Size:= ,<br>AryOut:= ); |

(2) Related variables

| VAR_INPUT | Name | Data type | Effective range | Initial value | Description |
|---|---|---|---|---|---|
| In1[] array | Compare array1 | USINT/UINT/UDINT/ULINT/ SINT/INT/DINT/LINT/REAL/ LREAL | As the data type | - | Compare array1 |
| In2[] array | Compare array2 | An array with the same data type as In1[] | As the data type | - | Compare array2 |
| Size | Compare element number | UINT | As the data type | 1 | Compare element number |
| VAR_IN_OUT | Name | Data type | Effective range | Initial value | Description |
| AryOut[] array | Comparison result array | BOOL | As the data type | - | Comparison result array |

(3) Function

- Compare the consecutive Size elements starting from In1[] and the consecutive Size elements starting from In2[] in two arrays. The comparison results are saved to the corresponding element numbers for the consecutive Size elements starting from the comparison result array AryOut[].
- When In1[i]>=In2[i], AryOut[i] is True, otherwise it is FALSE.

(4) Note

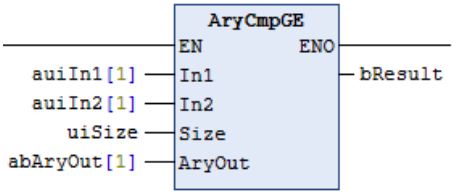
- The data types of In1[] and In2[] should be consistent, otherwise the return value will return FALSE and AryOut[] will remain unchanged.
- When In1[] and In2[] are real numbers and include inexhaustible division results, the processing results may differ from the expected ones due to rounding errors.
- When the value of "Size" is 0, the function returns a value of True and there is no change in AryOut[].
- The value of "Size" should not exceed any array area in In1[], In2[], or AryOut[].

(5) Program example

<table>
<tr><th>DEMO</th><th>LD</th><th>ST</th></tr>
<tr><td>Define variables</td><td colspan="2">

```
VAR
    auiIn1  :ARRAY[0..4] OF UINT:=[0,1,22,57,66];
    auiIn2  :ARRAY[0..4] OF UINT:=[0,99,1,22,33];
    abAryOut:ARRAY[0..4] OF BOOL;
    uiSize  :UINT:=4;
    bResult :BOOL;
END_VAR
```

</td></tr>
<tr><td>Sample program</td><td>




</td><td>

```
1  bResult:=AryCmpGE(In1:= auiIn1[1],
2      In2:= auiIn2[1],
3      Size:= uiSize,
4      AryOut:= abAryOut[1]);
```

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

| 表达式 | 类型 | 值 |
|---|---|---|
| auiIn1 | ARRAY [0..4] OF UINT | |
| auiIn1[0] | UINT | 0 |
| auiIn1[1] | UINT | 1 |
| auiIn1[2] | UINT | 22 |
| auiIn1[3] | UINT | 57 |
| auiIn1[4] | UINT | 66 |
| auiIn2 | ARRAY [0..4] OF UINT | |
| auiIn2[0] | UINT | 0 |
| auiIn2[1] | UINT | 99 |
| auiIn2[2] | UINT | 1 |
| auiIn2[3] | UINT | 22 |
| auiIn2[4] | 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 | 4 |
| bResult | BOOL | TRUE |

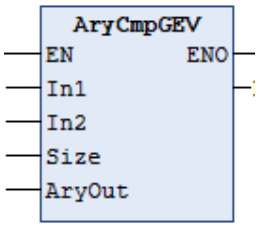
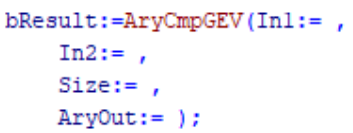
</td></tr>
</table>

1-4-2-7. Comparison of array elements GEV AryCmpGEV

(1) Instruction overview

Compare the size of each element and variable in the array (greater than or equal to).

| Comparison of array elements GEV [AryCmpGEV] | | | |
|---|---|---|---|
| Execution condition | Edge triggering | Suitable model | XS, XSA |
| Firmware | - | Software | XS StudioV2.2.0 and above |
| Library | XJ_DataUtils | | |

| Instruction | Name | Graph | ST |
|---|---|---|---|
| AryCmpGEV | Comparison of array elements GEV |  |  |



```
bResult:=AryCmpGEV(In1:= ,
    In2:= ,
    Size:= ,
    AryOut:= );
```

(2) Related variables

| VAR_INPUT | Name | Data type | Effective range | Initial value | Description |
|---|---|---|---|---|---|
| In1[] array | Compare array | USINT/UINT/UDINT/ ULINT/SINT/INT/DINT/ LINT/REAL/LREAL | As the data type | - | Compare array |
| In2[] array | Compare variable | Same data type as In1[] element | As the data type | - | Compare variable |
| Size | Compare element numbers | UINT | As the data type | 1 | Compare element numbers |
| VAR_IN_OUT | Name | Data type | Effective range | Initial value | Description |
| AryOut[] array | Comparison result array | BOOL | As the data type | - | Comparison result array |

Note: In2 can also be defined as a variable.

(3) Function

- Compare the consecutive Size elements starting from the two arrays In1[] and variable "In2" (greater than or equal to). The comparison results are saved to the corresponding element numbers for the consecutive Size elements starting from the comparison result arrangement of AryOut[].
- When In1[i]>=In2, AryOut[i] is True, otherwise it is FALSE.

(4) Note

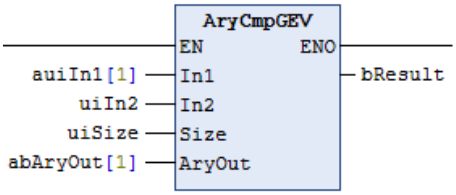
- The data types of In1[] and In2 should be consistent, otherwise the return value will return FALSE and AryOut[] will remain unchanged.
- When In1[] and In2 are real numbers and include inexhaustible division results, the processing results may differ from the expected ones due to rounding errors.
- When the value of "Size" is 0, the function returns a value of True and there is no change in AryOut[].
- The value of "Size" should not exceed any array area in In1[] or AryOut[].

(5) Program example

<table>
<tr><th>DEMO</th><th>LD</th><th>ST</th></tr>
<tr><td>Define variables</td><td colspan="2">

```
VAR
    auiIn1  :ARRAY[0..4] OF UINT:=[0,1,22,57,66];
    uiIn2   :UINT:=57;
    abAryOut:ARRAY[0..4] OF BOOL;
    uiSize  :UINT:=3;
    bResult :BOOL;
END_VAR
```

</td></tr>
<tr><td>Sample program</td><td>




</td><td>

```
1  bResult:=AryCmpGEV(In1:= auiIn1[1],
2      In2:= uiIn2,
3      Size:= uiSize,
4      AryOut:= abAryOut[1]);
```

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

| 表达式 | 类型 | 值 |
|---|---|---|
| auiIn1 | ARRAY [0..4] OF UINT | |
| auiIn1[0] | UINT | 0 |
| auiIn1[1] | UINT | 1 |
| auiIn1[2] | UINT | 22 |
| auiIn1[3] | UINT | 57 |
| auiIn1[4] | UINT | 66 |
| uiIn2 | UINT | 57 |
| 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 |
| bResult | BOOL | TRUE |

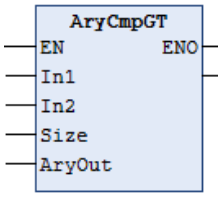
</td></tr>
</table>

1-4-2-8. Batch comparison of arrays GT AryCmpGT

(1) Instruction overview

Compare the size of each element of two arrays (greater than).

| Batch comparison of arrays GT [AryCmpGT] | | | |
|---|---|---|---|
| Execution condition | Edge triggering | Suitable model | XS, XSA |
| Firmware | - | Software | XS StudioV2.2.0 and above |
| Library | XJ_DataUtils | | |

| Instruction | Name | Graph | ST |
|---|---|---|---|
| AryCmpGT | Batch comparison of arrays GT |  | bResult:=AryCmpGT(In1:= ,<br>In2:= ,<br>Size:= ,<br>AryOut:= ); |

(2) Related variables

| VAR_INPUT | Name | Data type | Effective range | Initial value | Description |
|---|---|---|---|---|---|
| In1[] array | Compare array1 | USINT/UINT/UDINT/ ULINT/SINT/INT/DINT/ LINT/REAL/LREAL | As the data type | - | Compare array1 |
| In2[] array | Compare array2 | An array with the same data type as In1[] | As the data type | - | Compare array2 |
| Size | Compare element number | UINT | As the data type | 1 | Compare element number |
| VAR_IN_OUT | Name | Data type | Effective range | Initial value | Description |
| AryOut[] array | Comparison result array | BOOL | As the data type | - | Comparison result array |

(3) Function

- Compare the consecutive Size elements of In1[] and In2[] (greater than). The comparison results are saved to the corresponding element numbers for the consecutive Size elements starting from the comparison result array AryOut[].
- When In1 [i]>In2 [i], AryOut[i] is True, otherwise it is FALSE.

(4) Note

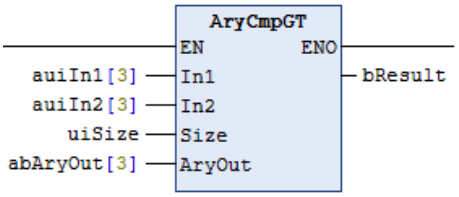
- The data types of In1[] and In2[] should be consistent, otherwise the return value will return FALSE and AryOut[] will remain unchanged.
- When In1[] and In2[] are real numbers and include inexhaustible division results, the processing results may differ from the expected ones due to rounding errors.
- When the value of "Size" is 0, the function returns a value of True and there is no change in AryOut[].
- The value of "Size" should not exceed any array area in In1[], In2[], or AryOut[].

(5) Program example

<table>
<tr><th>DEMO</th><th>LD</th><th>ST</th></tr>
<tr><td>Define variables</td><td colspan="2">

```
VAR
    auiIn1  :ARRAY[0..4] OF UINT:=[0,1,22,57,66];
    auiIn2  :ARRAY[0..4] OF UINT:=[0,99,1,22,33];
    abAryOut:ARRAY[0..4] OF BOOL;
    uiSize  :UINT:=2;
    bResult :BOOL;
END_VAR
```

</td></tr>
<tr><td>Sample program</td><td>




</td><td>

```
1  bResult:=AryCmpGT(In1:= auiIn1[3],
2      In2:= auiIn2[3],
3      Size:= uiSize,
4      AryOut:= abAryOut[3]);
```

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

| 表达式 | 类型 | 值 |
|---|---|---|
| auiIn1 | ARRAY [0..4] OF UINT | |
| auiIn1[0] | UINT | 0 |
| auiIn1[1] | UINT | 1 |
| auiIn1[2] | UINT | 22 |
| auiIn1[3] | UINT | 57 |
| auiIn1[4] | UINT | 66 |
| auiIn2 | ARRAY [0..4] OF UINT | |
| auiIn2[0] | UINT | 0 |
| auiIn2[1] | UINT | 99 |
| auiIn2[2] | UINT | 1 |
| auiIn2[3] | UINT | 22 |
| auiIn2[4] | 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 | 2 |
| bResult | BOOL | TRUE |

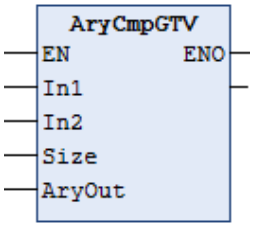
</td></tr>
</table>

1-4-2-9. Comparison of array elements GTV AryCmpGTV

(1) Instruction overview

Compare the size of each element and variable in the array (greater than).

| Comparison of array elements GTV [AryCmpGTV] | | | |
|---|---|---|---|
| Execution condition | Edge triggering | Suitable model | XS, XSA |
| Firmware | - | Software | XS StudioV2.2.0 and above |
| Library | XJ_DataUtils | | |

| Instruction | Name | Graph | ST |
|---|---|---|---|
| AryCmpGTV | Comparison of array elements GTV |  | bResult:=AryCmpGTV(In1:= ,<br>In2:= ,<br>Size:= ,<br>AryOut:= ); |

(2) Related variables

| VAR_INPUT | Name | Data type | Effective range | Initial value | Description |
|---|---|---|---|---|---|
| In1[] array | Compare array1 | USINT/UINT/UDINT/ ULINT/SINT/INT/DINT/ LINT/REAL/LREAL | As the data type | - | Compare array |
| In2[] array | Compare array2 | Same data type as In1[] element | As the data type | - | Compare variable |
| Size | Compare element number | UINT | As the data type | 1 | Compare element number |
| VAR_IN_OUT | Name | Data type | Effective range | Initial value | Description |
| AryOut[] array | Comparison result array | BOOL | As the data type | - | Comparison result array |

(3) Function

- Compare the consecutive Size elements starting from In1[] and variable "In2"(greater than). The comparison results are saved to the corresponding element numbers for the consecutive Size elements starting from the comparison result array AryOut[].
- When In1 [i]>In2, AryOut [i] is True, otherwise it is FALSE.

(4) Note

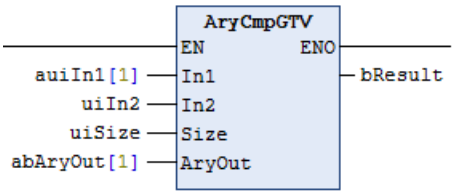
- The data types of In1[] and In2 should be consistent, otherwise the return value will return FALSE and AryOut[] will remain unchanged.
- When In1[] and In2 are real numbers and include inexhaustible division results, the processing results may differ from the expected ones due to rounding errors.
- When the value of "Size" is 0, the function returns a value of True and there is no change in AryOut[].
- The value of "Size" should not exceed any array area in In1[] or AryOut[].

(5) Program example

<table>
<tr><th>DEMO</th><th>LD</th><th>ST</th></tr>
<tr><td>Define variables</td><td colspan="2">

```
VAR
    auiIn1  :ARRAY[0..4] OF UINT:=[0,1,22,57,66];
    uiIn2   :UINT:=20;
    abAryOut:ARRAY[0..4] OF BOOL;
    uiSize  :UINT:=3;
    bResult :BOOL;
END_VAR
```

</td></tr>
<tr><td>Sample program</td><td>




</td><td>

```
bResult:=AryCmpGTV(In1:= auiIn1[1],
    In2:= uiIn2,
    Size:= uiSize,
    AryOut:= abAryOut[1]);
```

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

| 表达式 | 类型 | 值 |
|---|---|---|
| auiIn1 | ARRAY [0..4] OF UINT | |
| auiIn1[0] | UINT | 0 |
| auiIn1[1] | UINT | 1 |
| auiIn1[2] | UINT | 22 |
| auiIn1[3] | UINT | 57 |
| auiIn1[4] | UINT | 66 |
| uiIn2 | UINT | 20 |
| 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 | 3 |
| bResult | BOOL | TRUE |

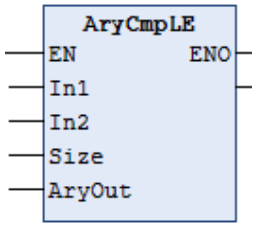
</td></tr>
</table>

1-4-2-10. Batch comparison of arrays LE AryCmpLE

(1) Instruction overview

Compare the size of each element of two arrays (less than or equal to).

| Batch comparison of arrays LE [AryCmpLE] | | | |
|---|---|---|---|
| Execution condition | Edge triggering | Suitable model | XS, XSA |
| Firmware | - | Software | XS StudioV2.2.0 and above |
| Library | XJ_DataUtils | | |

| Instruction | Name | Graph | ST |
|---|---|---|---|
| AryCmpLE | Batch comparison of arrays LE |  | bResult:=AryCmpLE(In1:= ,<br>In2:= ,<br>Size:= ,<br>AryOut:= ); |

(2) Related variables

| VAR_INPUT | Name | Data type | Effective range | Initial value | Description |
|---|---|---|---|---|---|
| In1[] array | Compare array1 | USINT/UINT/UDINT/ ULINT/SINT/INT/DINT/ LINT/REAL/LREAL | As the data type | - | Compare array1 |
| In2[] array | Compare array2 | An array with the same data type as In1[] | As the data type | - | Compare array2 |
| Size | Compare element number | UINT | As the data type | 1 | Compare element number |
| VAR_IN_OUT | Name | Data type | Effective range | Initial value | Description |
| AryOut[] array | Comparison result array | BOOL | As the data type | - | Comparison result array |

(3) Function

- Compare the continuous Size elements starting from In1[] and the continuous Size elements starting from In2[] (less than or equal to). The comparison results are saved to the corresponding element numbers for the consecutive Size elements starting from the comparison result array AryOut [].
- When In1[i]<=In2[i], AryOut[i] is True, otherwise it is FALSE.

(4) Note

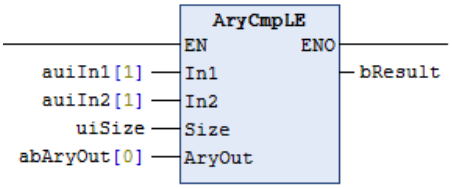
- The data types of In1[] and In2[] should be consistent, otherwise the return value will return FALSE and AryOut[] will remain unchanged.
- When In1[] and In2[] are real numbers and include inexhaustible division results, the processing results may differ from the expected ones due to rounding errors.
- When the value of "Size" is 0, the function returns a value of True and there is no change in AryOut[].
- The value of "Size" should not exceed any array area in In1[], In2[], or AryOut[].

(5) Program example

<table>
<tr><th>DEMO</th><th>LD</th><th>ST</th></tr>
<tr><td>Define variables</td><td colspan="2">

```
VAR
    auiIn1  :ARRAY[0..4] OF UINT:=[0,1,22,57,66];
    auiIn2  :ARRAY[0..4] OF UINT:=[0,99,1,22,33];
    abAryOut:ARRAY[0..4] OF BOOL;
    uiSize  :UINT:=4;
    bResult :BOOL;
END_VAR
```

</td></tr>
<tr><td>Sample program</td><td>




</td><td>

```
1  bResult:=AryCmpLE(In1:= auiIn1[1],
2      In2:= auiIn2[1],
3      Size:= uiSize,
4      AryOut:= abAryOut[0]);
```

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

| 表达式 | 类型 | 值 |
|---|---|---|
| auiIn1 | ARRAY [0..4] OF UINT | |
| auiIn1[0] | UINT | 0 |
| auiIn1[1] | UINT | 1 |
| auiIn1[2] | UINT | 22 |
| auiIn1[3] | UINT | 57 |
| auiIn1[4] | UINT | 66 |
| auiIn2 | ARRAY [0..4] OF UINT | |
| auiIn2[0] | UINT | 0 |
| auiIn2[1] | UINT | 99 |
| auiIn2[2] | UINT | 1 |
| auiIn2[3] | UINT | 22 |
| auiIn2[4] | UINT | 33 |
| abAryOut | ARRAY [0..4] OF BOOL | |
| abAryOut[0] | BOOL | TRUE |
| abAryOut[1] | BOOL | FALSE |
| abAryOut[2] | BOOL | FALSE |
| abAryOut[3] | BOOL | FALSE |
| abAryOut[4] | BOOL | FALSE |
| uiSize | UINT | 4 |
| bResult | BOOL | TRUE |

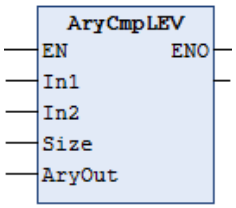
</td></tr>
</table>

1-4-2-11. Comparison of array elements LEV AryCmpLEV

(1) Instruction overview

Compare the size of each element and variable in an array (less than or equal to).

| Comparison of array elements LEV [AryCmpLEV] | | | |
|---|---|---|---|
| Execution condition | Edge triggering | Suitable model | XS, XSA |
| Firmware | - | Software | XS StudioV2.2.0 and above |
| Library | XJ_DataUtils | | |

| Instruction | Name | Graph | ST |
|---|---|---|---|
| AryCmpLEV | Comparison of array elements LEV |  | bResult:=AryCmpLEV(In1:= ,<br>In2:= ,<br>Size:= ,<br>AryOut:= ); |

(2) Related variables

| VAR_INPUT | Name | Data type | Effective range | Initial value | Description |
|---|---|---|---|---|---|
| In1[] array | Compare array | USINT/UINT/UDINT/ ULINT/SINT/INT/DINT/ LINT/REAL/LREAL | As the data type | - | Compare array |
| In2[] array | Compare variable | Same data type as In1[] element | As the data type | - | Compare variable |
| Size | Compare element number | UINT | As the data type | 1 | Compare element number |
| VAR_IN_OUT | Name | Data type | Effective range | Initial value | Description |
| AryOut[] array | Comparison result array | BOOL | As the data type | - | Comparison result array |

Note: In2 can also be defined as a variable.

(3) Function

- Compare the consecutive Size elements starting from the comparison array In1[] and variable "In2" (less than or equal to). The comparison results are saved to the corresponding element numbers for the consecutive Size elements starting from the comparison result array AryOut[].
- When In1[i]<=In2, AryOut[i] is True, otherwise it is FALSE.

(4) Note

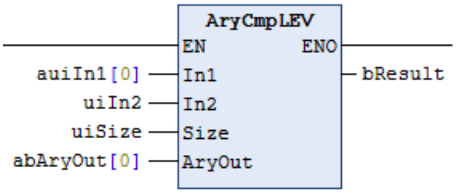
- The data types of In1[] and In2 should be consistent, otherwise the return value will return FALSE and AryOut[] will remain unchanged.
- When In1[] and In2 are real numbers and include inexhaustible division results, the processing results may differ from the expected ones due to rounding errors.
- When the value of "Size" is 0, the function returns a value of True and there is no change in AryOut[].
- The value of "Size" should not exceed any array area in In1[] or AryOut[].

(5) Program example

<table>
<tr><th>DEMO</th><th>LD</th><th>ST</th></tr>
<tr><td>Define variables</td><td colspan="2">

```
VAR
    auiIn1  :ARRAY[0..4] OF UINT:=[0,1,22,57,66];
    uiIn2   :UINT:=20;
    abAryOut:ARRAY[0..4] OF BOOL;
    uiSize  :UINT:=3;
    bResult :BOOL;
END_VAR
```

</td></tr>
<tr><td>Sample program</td><td>




</td><td>

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

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

| 表达式 | 类型 | 值 |
|---|---|---|
| auiIn1 | ARRAY [0..4] OF UINT | |
| auiIn1[0] | UINT | 0 |
| auiIn1[1] | UINT | 1 |
| auiIn1[2] | UINT | 22 |
| auiIn1[3] | UINT | 57 |
| auiIn1[4] | UINT | 66 |
| uiIn2 | UINT | 20 |
| 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 | 3 |
| bResult | BOOL | TRUE |

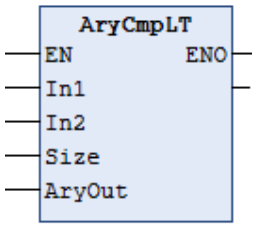
</td></tr>
</table>

1-4-2-12. Batch comparison of arrays LT AryCmpLT

(1) Instruction overview

Compare the size of each element of two arrays (less than).

| Batch comparison of arrays LT [AryCmpLT] | | | |
|---|---|---|---|
| Execution condition | Edge triggering | Suitable model | XS, XSA |
| Firmware | - | Software | XS StudioV2.2.0 and above |
| Library | XJ_DataUtils | | |

| Instruction | Name | Graph | ST |
|---|---|---|---|
| AryCmpLT | Batch comparison of arrays LT |  | bResult:=AryCmpLT(In1:= ,<br>In2:= ,<br>Size:= ,<br>AryOut:= ); |

(2) Related variables

| VAR_INPUT | Name | Data type | Effective range | Initial value | Description |
|---|---|---|---|---|---|
| In1[] array | Compare array1 | USINT/UINT/UDINT/ ULINT/SINT/INT/DINT/ LINT/REAL/LREAL | As the data type | - | Compare array1 |
| In2[] array | Compare array2 | An array with the same data type as In1[] | As the data type | - | Compare array2 |
| Size | Compare element number | UINT | As the data type | 1 | Compare element number |
| VAR_IN_OUT | Name | Data type | Effective range | Initial value | Description |
| AryOut[] array | Comparison result array | BOOL | As the data type | - | Comparison result array |

(3) Function

- Compare the consecutive Size elements starting from In1[] and consecutive Size elements starting from In2[](less than). The comparison results are saved to the corresponding element numbers for the consecutive Size elements starting from the comparison result array AryOut[].
- When In1[i]<In2[i], AryOut[i] is True, otherwise it is FALSE.

(4) Note

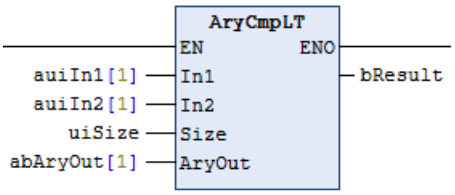
- The data types of In1[] and In2[] should be consistent, otherwise the return value will return FALSE and AryOut[] will remain unchanged.
- When In1[] and In2[] are real numbers and include inexhaustible division results, the processing results may differ from the expected ones due to rounding errors.
- When the value of "Size" is 0, the function returns a value of True and there is no change in AryOut[].
- The value of "Size" should not exceed any array area in In1[], In2[], or AryOut[].

(5) Program example

<table>
<tr><th>DEMO</th><th>LD</th><th>ST</th></tr>
<tr><td>Define variables</td><td colspan="2">

```
VAR
    auiIn1  :ARRAY[0..4] OF UINT:=[0,1,22,57,66];
    auiIn2  :ARRAY[0..4] OF UINT:=[0,99,1,22,33];
    abAryOut:ARRAY[0..4] OF BOOL;
    uiSize  :UINT:=4;
    bResult :BOOL;
END_VAR
```

</td></tr>
<tr><td>Sample program</td><td>




</td><td>

```
1  bResult:=AryCmpLT(In1:= auiIn1[1],
2      In2:= auiIn2[1],
3      Size:= uiSize,
4      AryOut:= abAryOut[1]);
```

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

| 表达式 | 类型 | 值 |
|---|---|---|
| auiIn1 | ARRAY [0..4] OF UINT | |
| auiIn1[0] | UINT | 0 |
| auiIn1[1] | UINT | 1 |
| auiIn1[2] | UINT | 22 |
| auiIn1[3] | UINT | 57 |
| auiIn1[4] | UINT | 66 |
| auiIn2 | ARRAY [0..4] OF UINT | |
| auiIn2[0] | UINT | 0 |
| auiIn2[1] | UINT | 99 |
| auiIn2[2] | UINT | 1 |
| auiIn2[3] | UINT | 22 |
| auiIn2[4] | UINT | 33 |
| 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 | FALSE |
| uiSize | UINT | 4 |
| bResult | BOOL | TRUE |

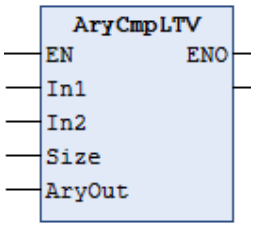
</td></tr>
</table>

1-4-2-13. Comparison of array elements LTV AryCmpLTV

(1) Instruction overview

Compare the size of each element and variable in the array (less than).

| Comparison of array elements LTV [AryCmpLTV] | | | |
|---|---|---|---|
| Execution condition | Edge triggering | Suitable model | XS, XSA |
| Firmware | - | Software | XS StudioV2.2.0 and above |
| Library | XJ_DataUtils | | |

| Instruction | Name | Graph | ST |
|---|---|---|---|
| AryCmpLTV | Comparison of array elements LT |  | bResult:=AryCmpLTV(In1:= ,<br>In2:= ,<br>Size:= ,<br>AryOut:= ); |

(2) Related variables

| VAR_INPUT | Name | Data type | Effective range | Initial value | Description |
|---|---|---|---|---|---|
| In1[] array | Compare array | USINT/UINT/UDINT/ ULINT/SINT/INT/DINT/ LINT/REAL/LREAL | As the data type | - | Compare array |
| In2[] array | Compare variable | Same data type as In1[] element | As the data type | - | Compare variable |
| Size | Compare element number | UINT | As the data type | 1 | Compare element number |
| VAR_IN_OUT | Name | Data type | Effective range | Initial value | Description |
| AryOut[] array | Comparison result array | BOOL | As the data type | - | Comparison result array |

Note: In2 can also be defined as a variable.

(3) Function

- Compare the consecutive Size elements starting from array In1[0] and the comparison variable "In2" (less than). The comparison results are saved to the corresponding element numbers for the consecutive Size elements starting from the comparison result array AryOut[].
- When In1[i]<In2, AryOut[i] is True, otherwise it is FALSE.

(4) Note

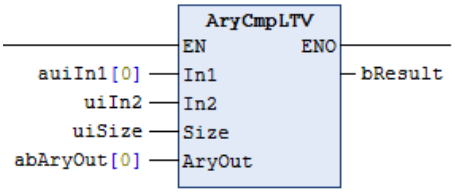
- The data types of In1[] and In2 should be consistent, otherwise the return value will return FALSE and AryOut[] will remain unchanged.
- When In1[] and In2 are real numbers and include inexhaustible division results, the processing results may differ from the expected ones due to rounding errors.
- When the value of "Size" is 0, the function returns a value of True and there is no change in AryOut[].

- The value of "Size" should not exceed any array area in In1[] or AryOut[].

(5) Program example

| DEMO | LD | ST |
|---|---|---|
| Define variables | | |
| Sample program | | |
| Operation result | | |

```
VAR
    auiIn1  :ARRAY[0..4] OF UINT:=[0,1,22,57,66];
    uiIn2   :UINT:=20;
    abAryOut:ARRAY[0..4] OF BOOL;
    uiSize  :UINT:=3;
    bResult :BOOL;
END_VAR
```




```
1  bResult:=AryCmpLTV(In1:= auiIn1[0],
2      In2:= uiIn2,
3      Size:= uiSize,
4      AryOut:= abAryOut[0]);
```

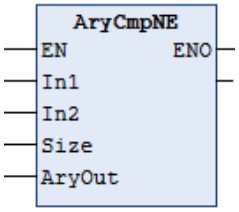
| 表达式 | 类型 | 值 |
|---|---|---|
| auiIn1 | ARRAY [0..4] OF UINT | |
| auiIn1[0] | UINT | 0 |
| auiIn1[1] | UINT | 1 |
| auiIn1[2] | UINT | 22 |
| auiIn1[3] | UINT | 57 |
| auiIn1[4] | UINT | 66 |
| uiIn2 | UINT | 20 |
| abAryOut | ARRAY [0..4] OF BOOL | |
| abAryOut[0] | BOOL | TRUE |
| abAryOut[1] | BOOL | TRUE |
| abAryOut[2] | BOOL | FALSE |
| abAryOut[3] | BOOL | FALSE |
| abAryOut[4] | BOOL | FALSE |
| uiSize | UINT | 3 |
| bResult | BOOL | TRUE |

1-4-2-14. Batch comparison of arrays NE AryCmpNE

(1) Instruction overview

Compare whether the elements of two arrays are different.

| Batch comparison of arrays NE [AryCmpNE] | | | |
|---|---|---|---|
| Execution condition | Edge triggering | Suitable model | XS, XSA |
| Firmware | - | Software | XS StudioV2.2.0 and above |
| Library | XJ_DataUtils | | |

| Instruction | Name | Graph | ST |
|---|---|---|---|
| AryCmpNE | Batch comparison of arrays NE |  | bResult:=AryCmpNE(In1:= ,<br>In2:= ,<br>Size:= ,<br>AryOut:= ); |

(2) Related variables

| VAR_INPUT | Name | Data type | Effective range | Initial value | Description |
|---|---|---|---|---|---|
| In1[] array | Compare array1 | BOOL/BYTE/WORD/ DWORD/LWORD/USINT/ UINT/UDINT/ULINT/SINT/ INT/DINT/LINT/REAL/LREAL | As the data type | - | Compare array1 |
| In2[] array | Compare array2 | An array with the same data type as In1[] | As the data type | - | Compare array2 |
| Size | Compare element number | UINT | As the data type | 1 | Compare element number |
| VAR_IN_OUT | Name | Data type | Effective range | Initial value | Description |
| AryOut[] array | Comparison result array | BOOL | As the data type | - | Comparison result array |

(3) Function

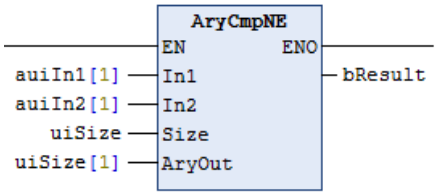
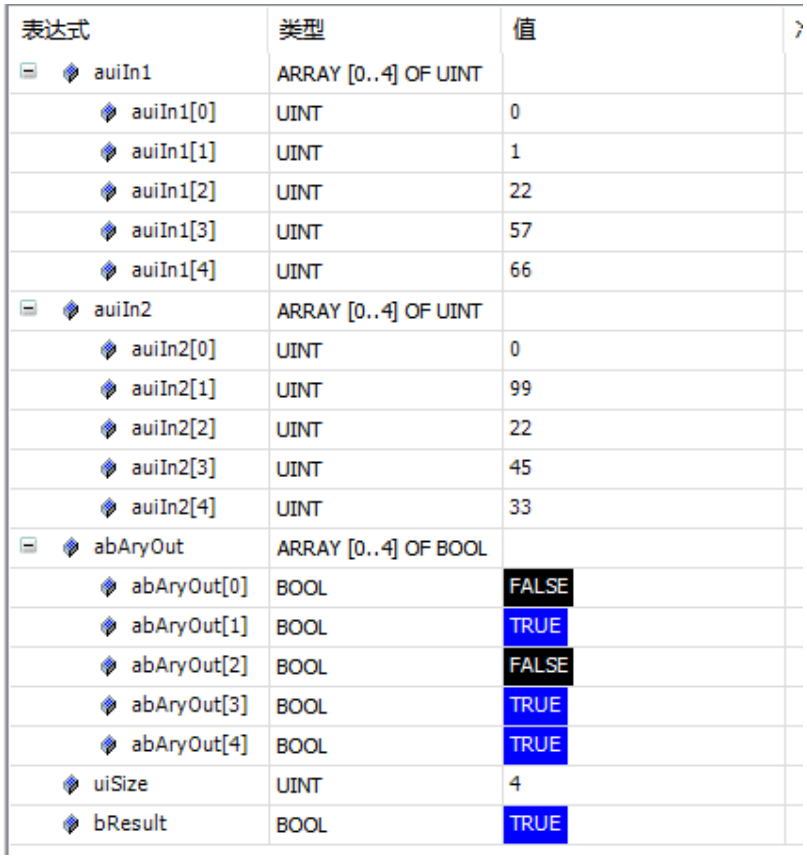
- Compare the consecutive Size elements starting from array In1[] and consecutive Size elements starting from array In2[]. The comparison results are saved to the corresponding element numbers for the consecutive Size elements starting from the comparison result array AryOut[].
- When In1[i]<>In2[i], AryOut[i] is True, otherwise it is FALSE.

(4) Note

- The data types of In1[] and In2[] should be consistent, otherwise the return value will return FALSE and AryOut[] will remain unchanged.
- In1[] and In2[] need to use Boolean, bitstring, integer, or real types, otherwise the return value returns FALSE, and AryOut[] remains unchanged.
- When In1[] and In2[] are real numbers and include inexhaustible division results, the processing results may differ from the expected ones due to rounding errors.

- When the value of "Size" is 0, the function returns a value of True and there is no change in AryOut[].
- The value of "Size" should not exceed any array area in In1[], In2[], or AryOut[].

(5) Program example

| DEMO | LD | ST |
|---|---|---|
| Define variables | VAR<br>auiIn1 :ARRAY[0..4] OF UINT:=[0,1,22,57,66];<br>auiIn2 :ARRAY[0..4] OF UINT:=[0,99,22,45,33];<br>abAryOut:ARRAY[0..4] OF BOOL;<br>uiSize :UINT:=4;<br>bResult :BOOL;<br>END_VAR | |
| Sample program |  | 1 bResult:=AryCmpNE(In1:= auiIn1[1],<br>2 In2:= auiIn2[1],<br>3 Size:= uiSize,<br>4 AryOut:= abAryOut[1]); |
| Operation result |  | |

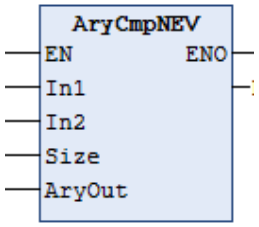
| 表达式 | 类型 | 值 |
|---|---|---|
| auiIn1 | ARRAY [0..4] OF UINT | |
| auiIn1[0] | UINT | 0 |
| auiIn1[1] | UINT | 1 |
| auiIn1[2] | UINT | 22 |
| auiIn1[3] | UINT | 57 |
| auiIn1[4] | UINT | 66 |
| auiIn2 | ARRAY [0..4] OF UINT | |
| auiIn2[0] | UINT | 0 |
| auiIn2[1] | UINT | 99 |
| auiIn2[2] | UINT | 22 |
| auiIn2[3] | UINT | 45 |
| auiIn2[4] | UINT | 33 |
| abAryOut | ARRAY [0..4] OF BOOL | |
| abAryOut[0] | BOOL | FALSE |
| abAryOut[1] | BOOL | TRUE |
| abAryOut[2] | BOOL | FALSE |
| abAryOut[3] | BOOL | TRUE |
| abAryOut[4] | BOOL | TRUE |
| uiSize | UINT | 4 |
| bResult | BOOL | TRUE |

1-4-2-15. Comparison of array elements NEV AryCmpNEV

(1) Instruction overview

Compare whether the elements of two arrays are different.

| Comparison of array elements NEV [AryCmpNEV] | | | |
|---|---|---|---|
| Execution condition | Edge triggering | Suitable model | XS, XSA |
| Firmware | - | Software | XS StudioV2.2.0 and above |
| Library | XJ_DataUtils | | |

| Instruction | Name | Graph | ST |
|---|---|---|---|
| AryCmpNEV | Comparison of array elements NEV |  | AryCmpNEV (<br>In1 :=,<br>In2 :=,<br>Size := ,<br>AryOut := ); |

(2) Related variables

| VAR_INPUT | Name | Data type | Effective range | Initial value | Description |
|---|---|---|---|---|---|
| In1[] array | Compare array | BOOL/BYTE/WORD/DWORD/ LWORD/USINT/UINT/UDINT/ ULINT/SINT/INT/DINT/LINT/ REAL/LREAL | As the data type | - | Compare array |
| In2[] array | Compare variable | Same data type as In1[] element | As the data type | - | Compare variable |
| Size | Compare element number | UINT | As the data type | 1 | Compare element number |
| VAR_IN_OUT | Name | Data type | Effective range | Initial value | Description |
| AryOut[] array | Comparison result array | BOOL | As the data type | - | Comparison result array |

Note: In2 can also be defined as a variable.

(3) Function

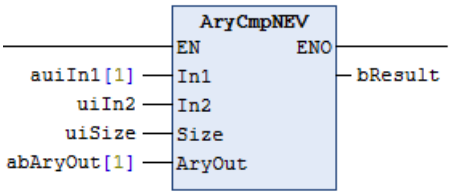
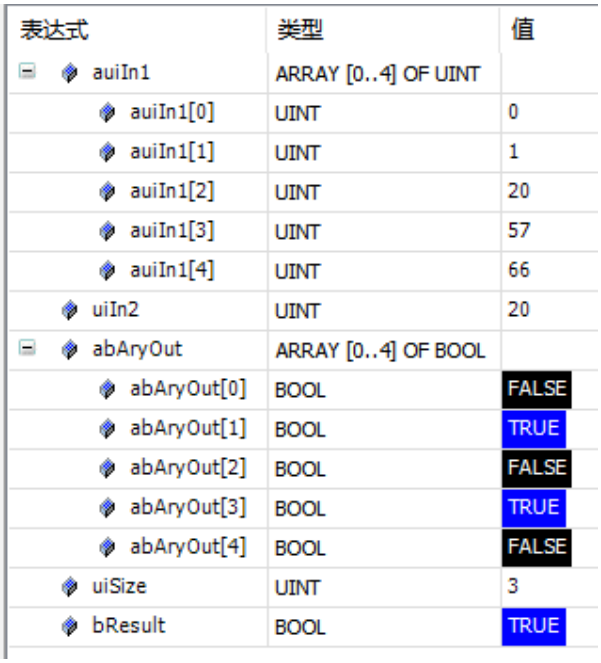
- Compare the continuous Size elements of compare array In1[] and comparison variable "In2" to see if they are different. The comparison results are saved to the corresponding element numbers for the consecutive Size elements starting from the comparison result array AryOut [].
- When In1[i]<>In2, AryOut[i] is True, otherwise it is FALSE.

(4) Note

- The data types of In1[] and In2 should be consistent, otherwise the return value will return FALSE and AryOut[] will remain unchanged.
- When In1[] and In2 are real numbers and include inexhaustible division results, the processing results may differ from the expected ones due to rounding errors.
- When the value of "Size" is 0, the function returns a value of True and there is no change in AryOut[].

- The value of "Size" should not exceed any array area in In1[] or AryOut[].
- In1[] and In2[] need to use Boolean, bitstring, integer, or real types, otherwise the return value returns FALSE, and AryOut[] remains unchanged.

(5) Program example

| DEMO | LD | ST |
|---|---|---|
| Define variables | VAR<br>auiIn1 :ARRAY[0..4] OF UINT:=[0,1,20,57,66];<br>uiIn2 :UINT:=20;<br>abAryOut:ARRAY[0..4] OF BOOL;<br>uiSize :UINT:=3;<br>bResult :BOOL;<br>END_VAR | |
| Sample program |  | 1 bResult:=AryCmpNEV(In1:= auiIn1[1],<br>2 In2:= uiIn2,<br>3 Size:= uiSize,<br>4 AryOut:= abAryOut[1]); |
| Operation result |  | |

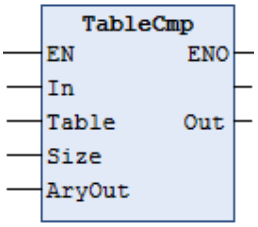
| 表达式 | 类型 | 值 |
|---|---|---|
| auiIn1 | ARRAY [0..4] OF UINT | |
| auiIn1[0] | UINT | 0 |
| auiIn1[1] | UINT | 1 |
| auiIn1[2] | UINT | 20 |
| auiIn1[3] | UINT | 57 |
| auiIn1[4] | UINT | 66 |
| uiIn2 | UINT | 20 |
| abAryOut | ARRAY [0..4] OF BOOL | |
| abAryOut[0] | BOOL | FALSE |
| abAryOut[1] | BOOL | TRUE |
| abAryOut[2] | BOOL | FALSE |
| abAryOut[3] | BOOL | TRUE |
| abAryOut[4] | BOOL | FALSE |
| uiSize | UINT | 3 |
| bResult | BOOL | TRUE |

1-4-2-16. Table comparison TableCmp

(1) Instruction overview

Compare multiple defined intervals specified in the comparison data and comparison table.

| Table comparison [TableCmp] | | | |
|---|---|---|---|
| Execution condition | Edge triggering | Suitable model | XS, XSA |
| Firmware | - | Software | XS StudioV2.2.0 and above |
| Library | XJ_DataUtils | | |

| Instruction | Name | Graph | ST |
|---|---|---|---|
| TableCmp | Table comparison |  | bResult:=TableCmp(In:= ,<br>Table:= ,<br>Size:= ,<br>AryOut:= ,<br>Out=> ); |

(2) Related variables

| VAR_INPUT | Name | Data type | Effective range | Initial value | Description |
|---|---|---|---|---|---|
| In | Compare data | USINT/UINT/UDINT/ ULINT/SINT/INT/DINT/ LINT/REAL/LREAL | As the data type | - | Value to be compared |
| Table[] 2D array | Compare table | A 2D array with elements of the same data type as In | As the data type | - | A 2D array with defined intervals as elements |
| Size | Compare size | UINT | As the data type | 1 | The number of elements in Table[] compared to "In" |
| AryOut[] array | Individual comparison result array | BOOL | [FALSE,TRUE] | FALSE | Each elements comparison results in Table[] True: Consistent FALSE: Inconsistent |
| VAR_OUTPUT | Name | Data type | Effective range | Initial value | Description |
| Out | Return value | BOOL | [FALSE,TRUE] | FALSE | Comparison result |

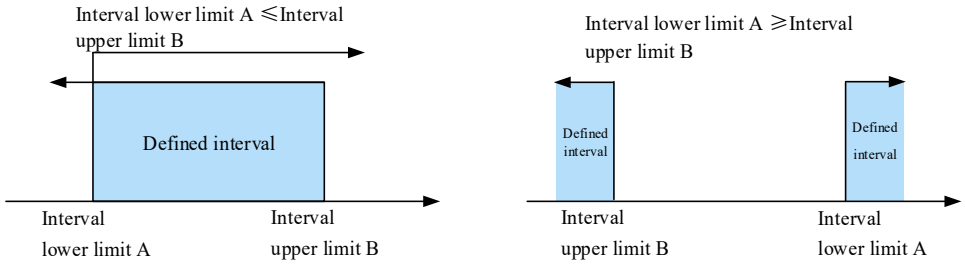
(3) Function

- Compare the defined group “Size” interval in the comparison data "In" and the comparison table[].
- Table[] is a 2-dimensional array, where the 1-dimensional element is the number of the defined interval, the 0th element in the 2-dimensional array represents the lower limit value A of the defined interval, and the 1st element represents the upper limit value B of the defined interval.

| Defined intervals | Interval lower limit value A | Interval upper limit value B |
|---|---|---|

| | | |
|---|---|---|
| Interval 0 | Table[0,0] | Table[0,1] |
| Interval 1 | Table[1,0] | Table[1,1] |
| ... | ... | ... |
| Interval “Size”-1 | Table[“Size”-1,0] | Table[“Size”-1,1] |

- The method of specifying the defined interval using the lower limit value A and the upper limit value B of the interval is as follows. The values of the lower limit value A and the upper limit value B of the interval are included in the defined interval.



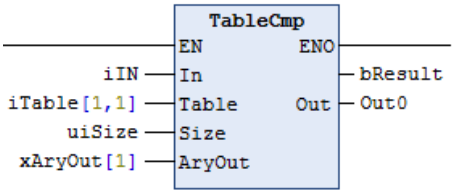

  - The comparison results of "In" and Table[] are saved in the individual comparison result array AryOut[]. If "In" is within the i-th defined interval, then the value of AryOut[i] is True, otherwise it is FALSE. If all the "Size" elements of AryOut[] are true, then the value of the comparison result "Out" is true, otherwise it is false.

(4) Note

- The data types of "In" and Table[] elements should be consistent, otherwise the function return value will be FALSE.
- Please make sure to use a 2-dimensional array for Table[] and the size of the 2nd dimensional array is 2, otherwise the output result will be unexpected.
- When the size of the AryOut[] array is greater than "Size", the comparison results will be saved to the corresponding element numbers of the consecutive Size at the beginning of AryOut[]. Other array elements remain unchanged.
- When comparing real numbers and including division results that cannot be fully divided, the processing results may differ from the expected ones due to rounding errors.
- When the value of "Size" is 0, the value of "Out" is FALSE, AryOut[] remains unchanged, and the function returns a value of True.
- When the value of "Size" exceeds the size of the AryOut[] array or the first dimensional array of Table[], it may cause program execution exceptions and even PLC crashes.

(5) Program example

| DEMO | LD | ST |
|---|---|---|
| Define variables | VAR<br>iIN :INT:=5;<br>iTable :ARRAY[1..3,1..2] OF INT:=[0,1,4,10,10,20];<br>uiSize :UINT:=3;<br>xAryOut:ARRAY[1..4] OF BOOL;<br>Out0 :BOOL;<br>bResult:BOOL;<br>END_VAR | |

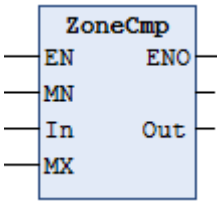
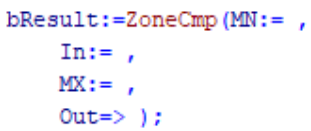
| | | |
|---|---|---|
| Sample program |  | ```<br>1 bResult:=TableCmp(In:= iIN,<br>2 Table:= iTable[1,1],<br>3 Size:= uiSize,<br>4 AryOut:= xAryOut[1],<br>5 Out=> Out0);<br>``` |
| Operation result | 表达式 / 类型 / 值<br>iIN — INT — 5<br>iTable — ARRAY [1..3, 1..2] O...<br>uiSize — UINT — 3<br>xAryOut — ARRAY [1..4] OF BOOL<br>xAryOut[1] — BOOL — FALSE<br>xAryOut[2] — BOOL — TRUE<br>xAryOut[3] — BOOL — FALSE<br>xAryOut[4] — BOOL — FALSE<br>Out0 — BOOL — FALSE<br>bResult — BOOL — TRUE | |

1-4-2-17. Zone comparison ZoneCmp

(1) Instruction overview

Determine whether the comparison data is between the specified upper and lower limits.

| Zone comparison [ZoneCmp] | | | |
|---|---|---|---|
| Execution condition | Edge triggering | Suitable model | XS, XSA |
| Firmware | - | Software | - |
| Library | XJ_DataUtils | | |

| Instruction | Name | Graph | ST |
|---|---|---|---|
| ZoneCmp | Zone comparison |  |  |



```
bResult:=ZoneCmp(MN:= ,
    In:= ,
    MX:= ,
    Out=> );
```

(2) Related variables

| VAR_INPUT | Name | Data type | Effective range | Initial value | Description |
|---|---|---|---|---|---|
| MN | Lower limit value | USINT/UINT/UDINT/ULINT/ SINT/INT/DINT/LINT/REAL/ LREAL/TIME/DATE/TOD/DT | As the data type | - | Lower limit value |
| In | Compare data | USINT/UINT/UDINT/ULINT/ SINT/INT/DINT/LINT/REAL/ LREAL/TIME/DATE/TOD/DT | As the data type | - | Value to be compared |
| MX | Upper limit value | USINT/UINT/UDINT/ULINT/ SINT/INT/DINT/LINT/REAL/ LREAL/TIME/DATE/TOD/DT | As the data type | 1 | Upper limit value |
| VAR_IN_OUT | Name | Data type | Effective range | Initial value | Description |
| Out | Comparison result | BOOL | [FALSE,TRUE] | FALSE | Comparison result |

(3) Function

- Determine whether the comparison data "In" is between the upper limit value "MX" and the lower limit value "MN".
- When "MX"≥"In"≥"MN", the value of "Out" is true, while others are false.
- The determination of the size relationship between values other than integers and real numbers is as follows:

| Data type | Size relationship |
|---|---|
| TIME | The value with a larger value is judged as large |
| DATA, TOD, DT | The date or time later is judged as large |

(4) Note

- When the data types of "In", "MX", and "MN" are different, the return value returns FALSE and Out remains unchanged.
- When "In", "MX", and "MN" are real numbers and include inexhaustible division results, the processing results

may differ from the expected ones due to rounding errors.

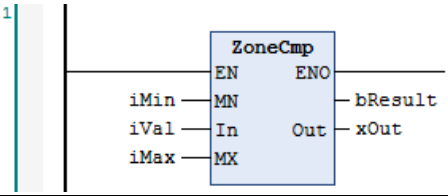
- Determine whether in the range of +∞ or -∞ are equal.
- When the value of "MN" is greater than the value of "MX", the program encounters an exception and the function returns a value of FALSE.
- "MX" and "MN" should not be set to non numerical values, otherwise the program will encounter an exception and the function will return a value of FALSE.
- "In", "MX", and "MN" need to use integers, real numbers, and dates. Otherwise, the return value will return FALSE and Out will remain unchanged.

(5) Program example

<table>
<tr><th>DEMO</th><th>LD</th><th>ST</th></tr>
<tr><td>Define variables</td><td colspan="2">

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

</td></tr>
<tr><td>Sample program</td><td>




</td><td>

```
1  bResult:=ZoneCmp(MN:= iMin,
2       In:= iVal,
3       MX:= iMax,
4       Out=> xOut);
```

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

| 表达式 | 类型 | 值 |
|---|---|---|
| iMin | INT | 10 |
| iVal | INT | 20 |
| iMax | INT | 30 |
| xOut | BOOL | TRUE |
| bResult | BOOL | TRUE |

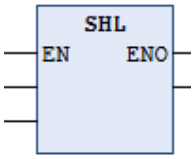
</td></tr>
</table>

## 1-4-3. Shift instruction

### 1-4-3-1. Shift left SHL

Shift the operand bit by bit to the left, leaving the left out bit unprocessed, and automatically filling the right empty bit with 0.

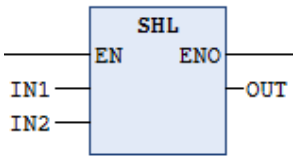
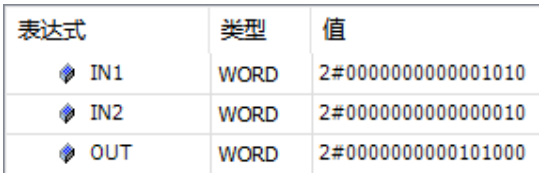
| Shift left [SHL] | | | |
|---|---|---|---|
| Execution condition | Edge triggering | Suitable model | XS, XSA |
| Firmware | - | Software | - |
| Library | - | | |

| Command name | Command icon | Function |
|---|---|---|
| SHL |  | SHL |

| VAR_INPUT | Name | Data type | Effective range | Initial value | Description |
|---|---|---|---|---|---|
| IN1 | Input data | BYTE/WORD/DWORD/LWORD/USINT/UINT/UDINT/ULINT/SINT/INT/DINT/LINT | As the data type | 0 | Input data |
| IN2 | Shift bits | BYTE/WORD/DWORD/LWORD/USINT/UINT/UDINT/ULINT/SINT/INT/DINT/LINT | As the data type | 0 | Shift bits |
| VAR_OUTPUT | Name | Data type | Effective range | Initial value | Description |
| OUT | Data after shifting | BYTE/WORD/DWORD/LWORD/USINT/UINT/UDINT/ULINT/SINT/INT/DINT/LINT | As the data type | 0 | Data after shifting |

The instruction can shift the data in input IN1 to the left by IN2 bits, and assign the output result to OUT. Shifting the binary number to the left by one bit is equivalent to multiplying the original number by 2. If the value of IN2 is greater than the width of the data type in IN1, the value of IN1 will be filled in as 0.

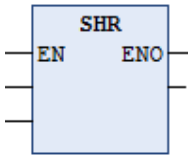
| DEMO | LD | ST |
|---|---|---|
| Define variables | | VAR<br>IN1:WORD:=10;<br>IN2:WORD:=2;<br>OUT:WORD;<br>END_VAR |

| DEMO | LD | ST |
|---|---|---|
| Sample program |  | 1 OUT:=SHL(IN1,IN2); |
| Operation result |  | |

1-4-3-2. Shift right SHR

Shift the operand bit by bit to the right, and do not process the right out bit.

| Shift right [SHR] | | | |
|---|---|---|---|
| Execution condition | Edge triggering | Suitable model | XS, XSA |
| Firmware | - | Software | - |
| Library | - | | |

| Command name | Command icon | Function |
|---|---|---|
| SHR |  | SHR |

| VAR_INPUT | Name | Data type | Effective range | Initial value | Description |
|---|---|---|---|---|---|
| IN1 | Input data | BYTE/WORD/DWORD/LWORD/USINT/UINT/UDINT/ULINT/SINT/INT/DINT/LINT | As the data type | 0 | Input data |
| IN2 | Shift bits | BYTE/WORD/DWORD/LWORD/USINT/UINT/UDINT/ULINT/SINT/INT/DINT/LINT | As the data type | 0 | Shift bits |
| VAR_OUTPUT | Name | Data type | Effective range | Initial value | Description |
| OUT | Data after shifting | BYTE/WORD/DWORD/LWORD/USINT/UINT/UDINT/ULINT/SINT/INT/DINT/LINT | As the data type | 0 | Data after shifting |

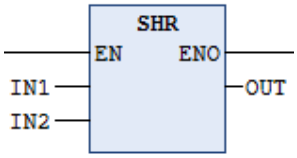
The instruction can shift the data in input IN1 to the right by IN2 bits, assign the output result to OUT, and shift the binary number to the right by one bit is equivalent to dividing the original number by 2. If the IN2 value is greater than the width of the data type in IN1, the IN1 value will be filled with 0.

After the unsigned data type is shifted to the right, since there is no sign bit, the high bit can be filled with 0. Encountering signed integer data, the high bit needs to be shifted to the right by filling in the sign bit.

<table>
<tr><th>DEMO</th><th>LD</th><th>ST</th></tr>
<tr><td>Define variables</td><td colspan="2">

```
VAR
    IN1:INT:=16#F100;
    IN2:WORD:=2;
    OUT:INT;
END_VAR
```

</td></tr>
<tr><td>Sample program</td><td>



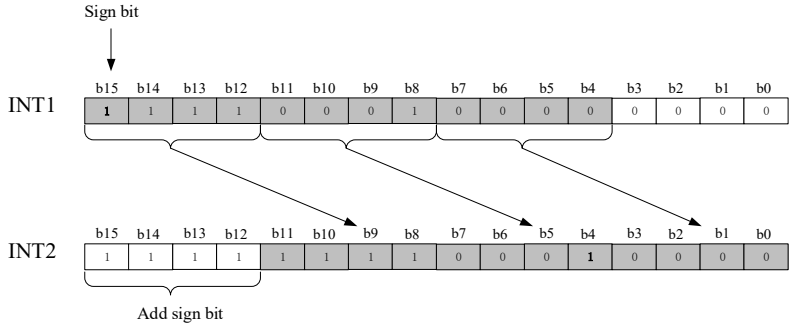

</td><td>

```
1  OUT:=SHR(IN1,IN2);
```

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

| 表达式 | 类型 | 值 |
|---|---|---|
| IN1 | INT | 2#1111000100000000 |
| IN2 | WORD | 2#0000000000000010 |
| OUT | INT | 2#1111110001000000 |

</td></tr>
<tr><td>Result analysis</td><td colspan="2">Due to INT is signed bit data with valid values ranging from -32768 to 32767, IN1 is hexadecimal signed data F100, and the highest bit b15 is the sign bit, after being moved 4 bits to the right, data needs to be added. As the source data sign bit is 1, the high 4 bits are added with four 1 (i.e. sign bits). Therefore, the final result of program operation is 16#FF10. The specific shift process is shown in the following figure:

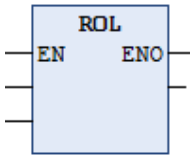



</td></tr>
</table>

1-4-3-3. Rotate left ROL

Rotate the operand in a bitwise loop to the left, and the bits removed from the left are directly added to the lowest bit on the right.

| Rotate left [ROL] | | | |
|---|---|---|---|
| Execution condition | Edge triggering | Suitable model | XS, XSA |
| Firmware | - | Software | - |
| Library | - | | |

| Command name | Command icon | Function |
|---|---|---|
| ROL |  | ROL |

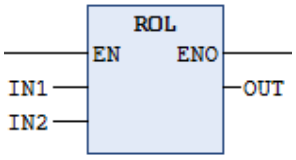
| VAR_INPUT | Name | Data type | Effective range | Initial value | Description |
|---|---|---|---|---|---|
| IN1 | Input data | BYTE/WORD/DWORD/LWORD/ USINT/UINT/UDINT/ULINT/ SINT/INT/DINT/LINT | As the data type | 0 | Input data |
| IN2 | Shift bits | BYTE/WORD/DWORD/LWORD/ USINT/UINT/UDINT/ULINT/ SINT/INT/DINT/LINT | As the data type | 0 | Shift bits |
| VAR_OUTPUT | Name | Data type | Effective range | Initial value | Description |
| OUT | Data after shifting | BYTE/WORD/DWORD/LWORD/ USINT/UINT/UDINT/ULINT/ SINT/INT/DINT/LINT | As the data type | 0 | Data after shifting |

This instruction can be used to cycle all the contents of input IN1 to the left bit by bit, and the empty bits are filled with the signal status of the shifted bits. The input parameter IN2 provides a numerical value representing the number of bits to rotate, and OUT is the result of the rotation operation.

After the unsigned data type is shifted to the right, since there is no sign bit, the high bit can be filled with 0. Encountering signed integer data, the high bit needs to be shifted to the right by filling in the sign bit.

| DEMO | LD | ST |
|---|---|---|
| Define variables | | |

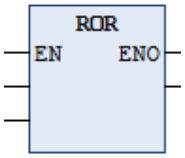
```
VAR
    IN1:WORD:=16#AA00;
    IN2:WORD:=4;
    OUT:WORD;
END_VAR
```

| | | |
|---|---|---|
| Sample program |  | 1 OUT:=ROL(IN1,IN2); |
| Operation result | | |
| Result analysis | The use of the output OUT after a right shift loop does not fill in the empty bit by 0, but rather fills in the 1010 from b12 to b15 in the input data IN1 to bits b0 to b3. | |

1-4-3-4. Rotate right ROR

Rotate the operands bit by bit to the right, and the bits removed from the right are directly added to the highest position on the left.

| Rotate right [ROR] | | | |
|---|---|---|---|
| Execution condition | Edge triggering | Suitable model | XS, XSA |
| Firmware | - | Software | - |
| Library | - | | |

| Command name | Command icon | Function |
|---|---|---|
| ROR |  | ROR |

| VAR_INPUT | Name | Data type | Effective range | Initial value | Description |
|---|---|---|---|---|---|
| IN1 | Input data | BYTE/WORD/DWORD/LWORD/ USINT/UINT/UDINT/ULINT/ SINT/INT/DINT/LINT | As the data type | 0 | Input data |
| IN2 | Shift bits | BYTE/WORD/DWORD/LWORD/ USINT/UINT/UDINT/ULINT/ SINT/INT/DINT/LINT | As the data type | 0 | Shift bits |
| VAR_OUTPUT | Name | Data type | Effective range | Initial value | Description |
| OUT | Data after shifting | BYTE/WORD/DWORD/LWORD/ USINT/UINT/UDINT/ULINT/SINT /INT/DINT/LINT | As the data type | 0 | Data after shifting |

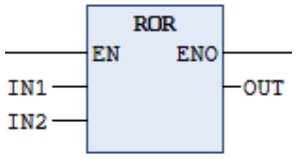
This instruction can be used to cycle all the contents of input IN1 to the left bit by bit, and the empty bits are filled with the signal status of the shifted bits. The input parameter IN2 provides a numerical value representing the number of bits to rotate, and OUT is the result of the rotation operation.

After the unsigned data type is shifted to the right, since there is no sign bit, the high bit can be filled with 0. Encountering signed integer data, the high bit needs to be shifted to the right by filling in the sign bit.

<table>
<tr><th>DEMO</th><th>LD</th><th>ST</th></tr>
<tr><td>Define variables</td><td colspan="2">

```
VAR
    IN1:WORD:=16#AA00;
    IN2:WORD:=4;
    OUT:WORD;
END_VAR
```

</td></tr>
<tr><td>Sample program</td><td>




</td><td>

```
1  OUT:=ROR(IN1,IN2);
```

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

| 表达式 | 类型 | 值 |
|---|---|---|
| IN1 | WORD | 16#AA00 |
| IN2 | WORD | 16#0004 |
| OUT | WORD | 16#0AA0 |

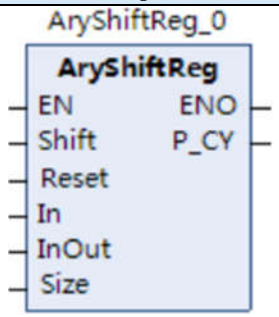
</td></tr>
</table>

1-4-3-5. Shift register AryShiftReg

(1) Instruction overview

Shift the entire array elements by 1 bit to the left, and insert the input value in the lowest position.

| Shift register [AryShiftReg] | | | |
|---|---|---|---|
| Execution condition | Edge triggering | Suitable model | XS, XSA |
| Firmware | - | Software | XS StudioV2.2.0 and above |
| Library | XJ_DataUtils | | |

| Instruction | Name | Graph | ST |
|---|---|---|---|
| AryShiftReg | shift register |  | AryShiftReg_0(<br>Shift:= ,<br>Reset:= ,<br>In:= ,<br>InOut:= ,<br>Size:= ,<br>P_CY=> ,<br>ENO=> ); |

(2) Related variables

| VAR_INPUT | Name | Data type | Effective range | Initial value | Description |
|---|---|---|---|---|---|
| Shift | Shift | BOOL | [FALSE,TRUE] | FALSE | When changing from FALSE to True, perform a shift |
| Reset | Reset | BOOL | [FALSE,TRUE] | FALSE | When changing from FALSE to True, perform a reset |
| In | Input value | BOOL | [FALSE,TRUE] | FALSE | The value inserted in the lowest position of InOut[] |
| Size | The number of elements in a positional array | UINT | As the data type | 0 | The number of elements used in the shift register of InOut[] |
| VAR_OUTPUT | Name | Data type | Effective range | Initial value | Description |
| P_CY | carry flag | BOOL | As the data type | - | The value saved in the carry flag |
| VAR_IN_OUT | Name | Data type | Effective range | Initial value | Description |
| InOut[] array | positional array | BOOL/BYTE/WORD /DWORD/LWORD | As the data type | - | positional array |

(3) Function

- When "Shift" changes from FALSE to True, shift the "Size" array elements starting from the InOut[] set on pin

InOut to the left (high bit direction) by 1 bit.

- Insert the input value "In" in the lowest position. The highest bit overflowing from the array will be output to the carry (CY) flag "P_CY".
- When "Reset" is true, set all bits of "Size" elements starting from the InOut[] set on pin InOut and carry (CY) flags to FALSE.

(4) Note

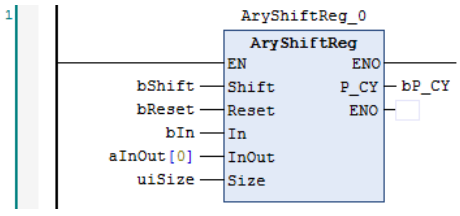
- When "Reset" is true, the array InOut[] cannot be assigned a value, and even if "Shift" changes from FALSE to true, no shift is performed.
- When the value of "Size" is 0, InOut[] remains unchanged, and the function returns a value of True.
- When the value of "Size" exceeds the array area of InOut[], the PLC will stop prompting an exception, as shown in the following figure:

- InOut[] requires the use of Boolean and bit string types, otherwise ENO will be false when the program runs.

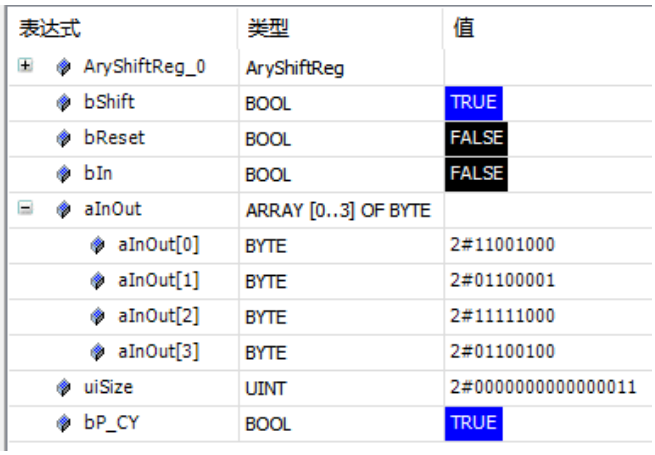
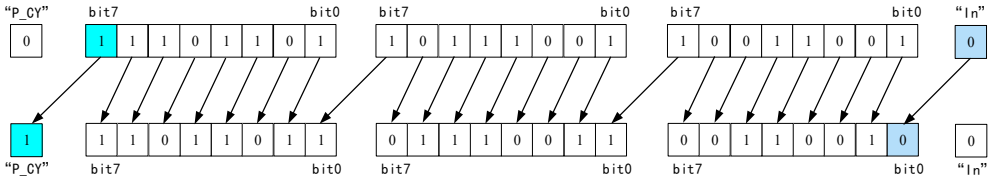
(5) Program example

| DEMO | LD | ST |
|---|---|---|
| Define the variable | | |
| Sample program | | |

```
PROGRAM AryShiftReg_ST
VAR
    AryShiftReg_0:AryShiftReg;
    bShift:BOOL;
    bReset:BOOL;
    bIn:BOOL;
    aInOut:ARRAY[0..3]OF BYTE:=[2#11100100,2#00110000,2#11111100,2#01100100];
    uiSize:UINT;
    bP_CY:BOOL;
END_VAR
```




```
AryShiftReg_0(
    Shift:= bShift,
    Reset:= bReset,
    In:= bIn,
    InOut:= aInOut[0],
    Size:= uiSize,
    P_CY=> bP_CY,
    ENO=> );
```

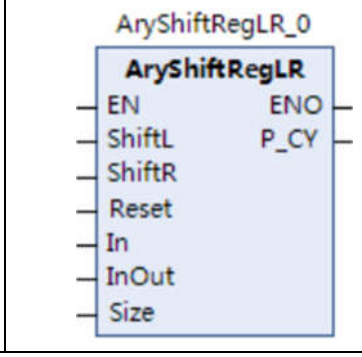
| | |
|---|---|
| Operation result |  <br> |
| Principle analysis |  <br> |

1-4-3-6. Left-right shift register AryShiftRegLR

(1) Instruction overview

Shift the entire array elements by 1 bit to the left (or right), and insert the input value at the lowest (or highest) position.

| Left-right shift register [AryShiftRegLR] | | | |
|---|---|---|---|
| Execution condition | Edge triggering | Suitable model | XS, XSA |
| Firmware | - | Software | XS StudioV2.2.0 and above |
| Library | XJ_DataUtils | | |

| Instruction | Name | Graph | ST |
|---|---|---|---|
| AryShiftRegLR | left-right shift register |  | AryShiftRegLR_0(<br>ShiftL:= ,<br>ShiftR:= ,<br>Reset:= ,<br>In:= ,<br>InOut:= ,<br>Size:= ,<br>P_CY=> ,<br>ENO=> ); |

(2) Related variables

| VAR_INPUT | Name | Data type | Effective range | Initial value | Description |
|---|---|---|---|---|---|
| ShiftL | Shift left | BOOL | [FALSE,TRUE] | FALSE | When changing from FALSE to True, perform a left shift |
| ShiftR | Shift right | BOOL | [FALSE,TRUE] | FALSE | When changing from FALSE to True, perform a right shift |
| Reset | Reset | BOOL | [FALSE,TRUE] | FALSE | When changing from FALSE to True, perform reset |
| In | Input value | BOOL | [FALSE,TRUE] | FALSE | The value inserted into the lowest or highest position of InOut[] |
| Size | The number of elements in a positional array | UINT | As the data type | 0 | The number of elements used in the shift register of InOut[] |
| VAR_OUTPUT | Name | Data type | Effective range | Initial value | Description |

| P_CY | carry flag | BOOL | As the data type | - | The value saved in the carry flag |
|---|---|---|---|---|---|
| VAR_IN_OUT | Name | Data type | Effective range | Initial value | Description |
| InOut[] array | positional array | BOOL/BYTE/WORD /DWORD/LWORD | As the data type | - | positional array |

(3) Function

- When "ShiftL" changes from FALSE to True, shift the "Size" array elements starting from the InOut[] set on pin InOut to the left (high bit direction) by 1 bit. Insert the input value "In" in the lowest position. The highest bit overflowing from the array will be output to the carry (CY) flag "P_CY".
- When "ShiftR" changes from FALSE to True, shift the "Size" array elements starting from the InOut[] set on pin InOut to the right (low bit direction) by 1 bit. Insert the input value "In" into the highest position. The lowest bit overflowing from the array will be output to the carry (CY) flag "P_CY".
- When "Reset" is true, set all bits of "Size" elements starting from the InOut[] set on pin InOut and carry (CY) flags to FALSE.

(4) Note

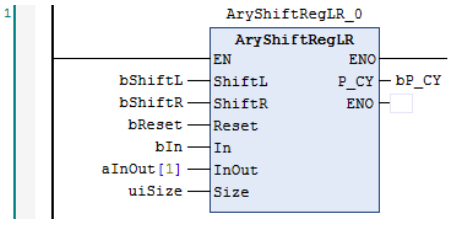
- When "Reset" is true, even if "ShiftL" or "ShiftR" changes from FALSE to true, no shift is performed.
- When "ShiftL" and "ShiftR" both change from FALSE to True, no shift is performed.
- When the value of "Size" is 0, InOut[] remains unchanged, and the function returns a value of True.
- When the value of "Size" exceeds the array area of InOut[], the PLC will stop prompting an exception, as shown in the following figure:

- InOut[] requires Boolean and bit string types, otherwise ENO will be false at runtime.

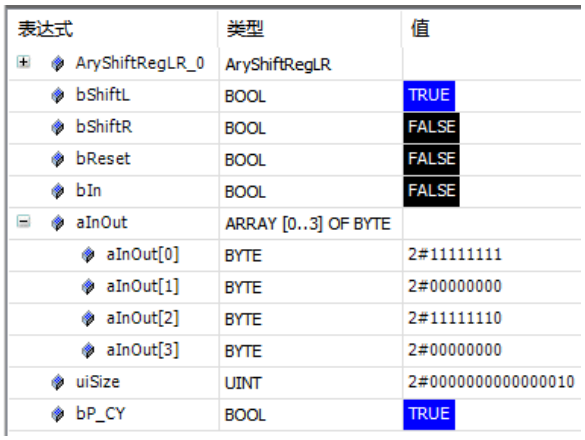
(5) Program example

| DEMO | LD | ST |
|---|---|---|
| Define the variable | | |
| Sample program | | |

```
VAR
    AryShiftRegLR_0:AryShiftRegLR;
    bShiftL:BOOL;
    bShiftR:BOOL;
    bReset:BOOL;
    bIn:BOOL;
    aInOut:ARRAY[0..3]OF BYTE:=[2#11111111,2#00000000,2#11111111,2#00000000];
    uiSize:UINT:=2;
    bP_CY:BOOL;
END_VAR
```




```
1  AryShiftRegLR_0(
2      ShiftL:= bShiftL,
3      ShiftR:= bShiftR,
4      Reset:= bReset,
5      In:= bIn,
6      InOut:= aInOut[1],
7      Size:= uiSize,
8      P_CY=> bP_CY,
9      ENO=> );
```

| | |
|---|---|
| Operation result |  |

| 表达式 | 类型 | 值 |
|---|---|---|
| AryShiftRegLR_0 | AryShiftRegLR | |
| bShiftL | BOOL | TRUE |
| bShiftR | BOOL | FALSE |
| bReset | BOOL | FALSE |
| bIn | BOOL | FALSE |
| aInOut | ARRAY [0..3] OF BYTE | |
| aInOut[0] | BYTE | 2#11111111 |
| aInOut[1] | BYTE | 2#00000000 |
| aInOut[2] | BYTE | 2#11111110 |
| aInOut[3] | BYTE | 2#00000000 |
| uiSize | UINT | 2#0000000000000010 |
| bP_CY | BOOL | TRUE |

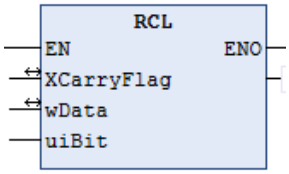
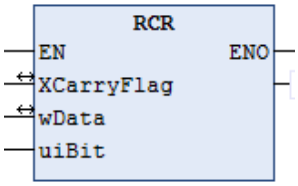
1-4-3-7. Loop left/right shift with carry RCL/RCR

(1) Instruction overview

RCL: After execution, the operand is cyclically shifted to the left with a carry flag, and the bit moved out on the left is directly filled to the carry flag. The original carry flag is shifted to the lowest position on the right.

RCR: After execution, the operand is cyclically shifted to the right with a carry flag, and the bit moved to the right is directly filled to the carry flag. The original carry flag is shifted to the highest position on the left.

| Loop left/right shift with carry [RCL/RCR] | | | |
|---|---|---|---|
| Execution condition | Edge triggering | Suitable model | XS, XSA |
| Firmware | - | Software | XS StudioV2.2.0 and above |
| Library | XJ_DataUtils | | |

| Instruction | Name | Graph | ST |
|---|---|---|---|
| RCL | Loop left shift with carry |  | RCL(XCarryFlag:= ,<br>wData:= ,<br>uiBit:= ); |
| RCR | Loop right shift with carry |  | RCR(XCarryFlag:= ,<br>wData:= ,<br>uiBit:= ); |

(2) Related variables

| VAR_INPUT | Name | Data type | Effective range | Initial value | Description |
|---|---|---|---|---|---|
| uiBit | Loop shift number | UINT | 1-1024 | - | Loop shift number |
| VAR_IN_OUT | Name | Data type | Effective range | Initial value | Description |
| xCarryFlag | Data carry flag | BOOL | - | - | Data carry flag |
| wData | data sources | WORD | 1-1024 | - | data sources |

(3) Function

- Shift the WORD type data in "wData" to the left and right according to binary display.
- Specify the number of elements to be converted of "In[]" in "Size".

(4) Program example

**① RCL**

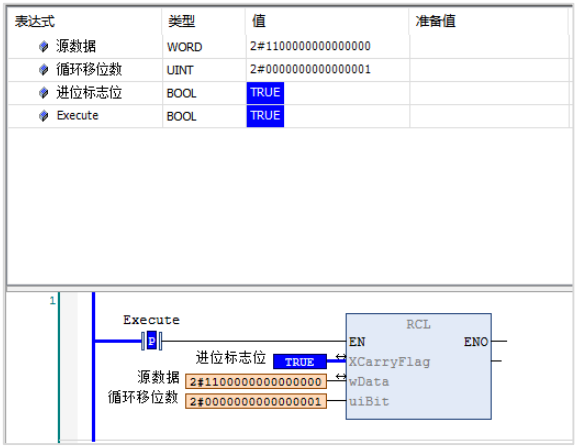
When the input signal "source data" is 2 # 111000000 and the "shift number" is 1:

**ST:**

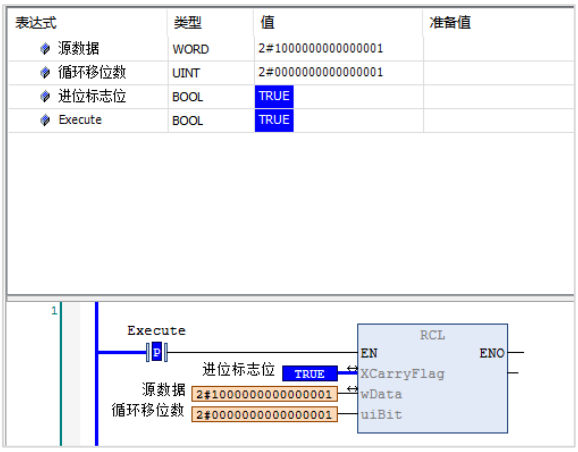
Triggering for the first time, the output signal "result data" is 2 # 11000000000. The valid data is shifted to the left by 1 bit to the carry flag, and the original carry flag is shifted to the lowest position on the right.

Triggering for the second time, the output signal "data result" is 2 # 1000000000001. The valid data is shifted to the left by 1 bit to the carry flag, and the original carry flag is shifted to the lowest position on the right.

**LD:**




Triggering for the first time, the output signal "result data" is 2 # 11000000000. The valid data is shifted to the left by 1 bit to the carry flag, and the original carry flag is shifted to the lowest position on the right.

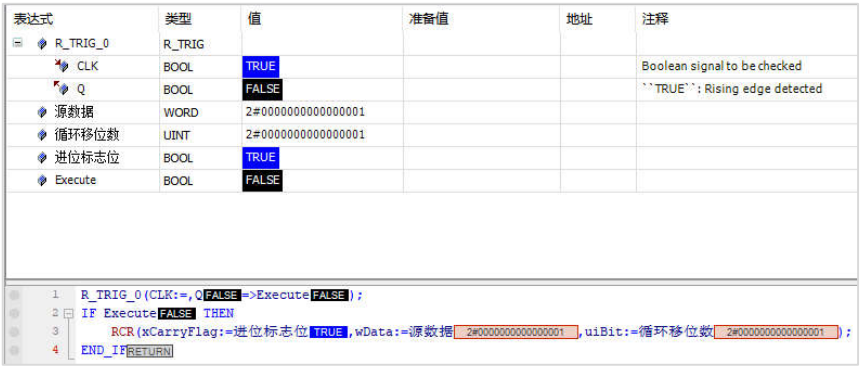



Triggering for the second time, the output signal "data result" is 2 # 1000000000001. The valid data is shifted to the left by 1 bit to the carry flag, and the original carry flag is shifted to the lowest position on the right.

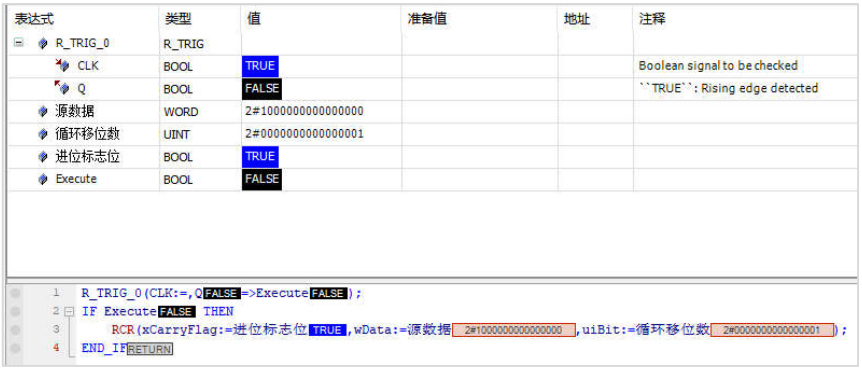
## ② RCR

When the input signal "source data" is 2 # 000000 0000011 and the "shift number" is 1:

**ST:**

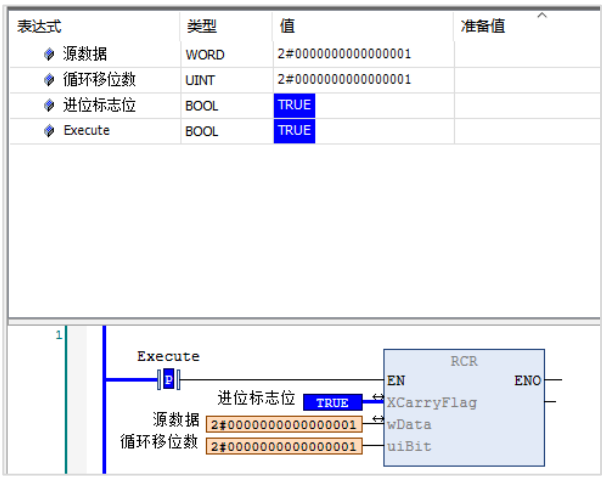



Triggering for the first time, the output signal "result data" is 2 # 1000000000001. The valid data is shifted 1 bit to the right to the carry flag, and the original carry flag is shifted to the highest position on the left.

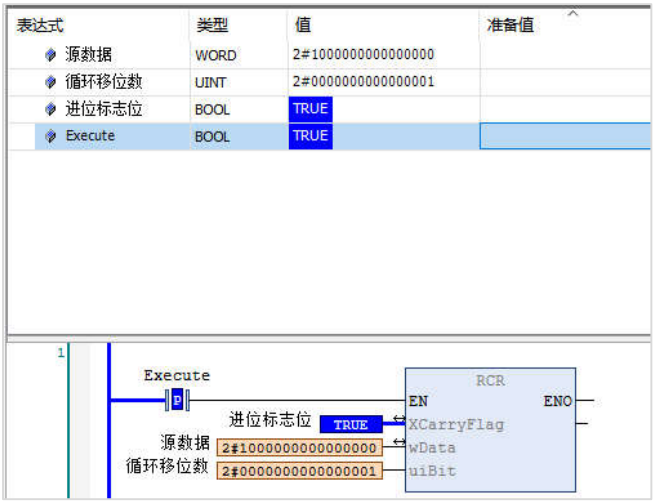



Triggering for the second time, the output signal "data result" is 2 # 100000000000. The valid data is shifted 1 bit to the right to the carry flag, and the original carry flag is shifted to the highest position on the left.

**LD:**




Triggering for the first time, the output signal "result data" is 2 # 1000000000001. The valid data is shifted 1 bit to the right to the carry flag, and the original carry flag is shifted to the highest position on the left.




Triggering for the second time, the output signal "data result" is 2 # 100000000000. The valid data is shifted 1 bit to the right to the carry flag, and the original carry flag is shifted to the highest position on the left.

1-4-3-8. Shift N elements of an array left/right ArySHL/ArySHR

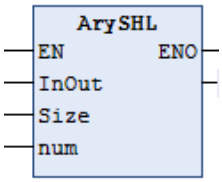
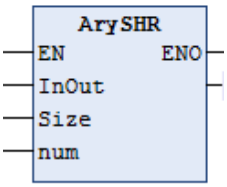
(1) Instruction overview

Shift multiple array elements.

ArySHL: Shift multiple elements in an array to the left (high order direction).

ArySHR: Shift multiple elements in an array to the right (low order direction).

| Shift N elements of an array left/right [ArySHL/ArySHR] | | | |
|---|---|---|---|
| Execution condition | Edge triggering | Suitable model | XS, XSA |
| Firmware | - | Software | XS StudioV2.2.0 and above |
| Library | XJ_DataUtils | | |

| Instruction | Name | Graph | ST |
|---|---|---|---|
| ArySHL | Left shift of N elements in an array |  | ArySHL(InOut:= ,<br>Size:= ,<br>num:= ); |
| ArySHR | Right shift of N elements in an array |  | ArySHR(InOut:= ,<br>Size:= ,<br>num:= ); |

(2) Related variables

| VAR_INPUT | Name | Data type | Effective range | Initial value | Description |
|---|---|---|---|---|---|
| Size | The number of elements in the shifted object array | UINT | As the data type | 1 | The number of elements in the shifted object array |
| Num | Number of shifted digits | UINT | As the data type | 1 | Number of shifted digits |
| VAR_IN_OUT | Name | Data type | Effective range | Initial value | Description |
| InOut[] array | Shift Object Array | BOOL/BYTE/WORD/DWORD/ LWORD/USINT/UINT/UDINT /ULINT/SINT/INT/DINT/ LINT/REAL/LREAL/TIME/ DATE/TOD/DT/STRING | As the data type | - | Shift Object Array |

(3) Function

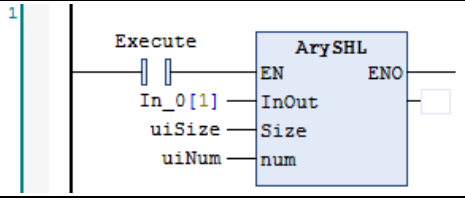
- Shift the "Num" bits left/right for the low/high "Size" elements of the shift object array InOut[].
- Elements are only shifted within "Size", and after shifting, overflow elements from the shifted object array are cleared.
- After shifting, initialize the empty elements of the object array InOut[], that is, assign the default initial value of the corresponding data type to the empty elements.
- The default initial values for each data type are shown in the table below:

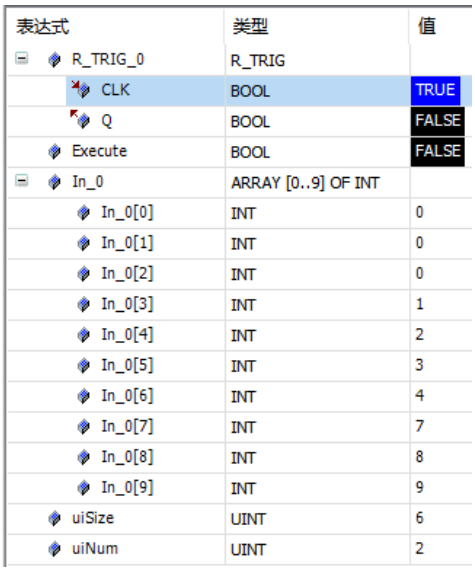
| Data type | Initial value |
|---|---|
| BOOL | FALSE |
| BYTE, WORD, DWORD, LWORD | 16#0 |
| USINT, UINT, UDINT, ULINT, SINT, INT, DINT, LINT, REAL, LREAL | 0 |
| TIME | T#0ms |
| DATE | D#1970-1-1 |
| TOD | TOD#0:0:0 |
| DT | DT#1970-1-1-0:0:0 |
| STRING | ‘’ |

(4) Note

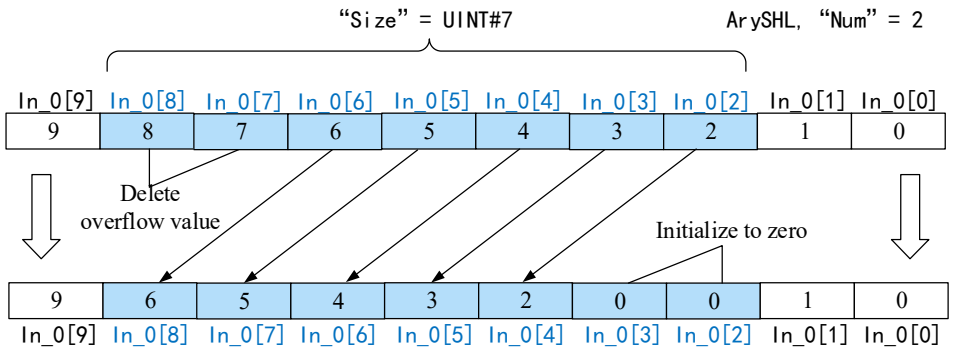
- The value of "Size" should not exceed the array area of InOut[].
- When the value of "Num" or "Size" is 0, no shift action is taken, and the value returned by the function is true, while InOut[] remains unchanged.
- When the value of "Num" is greater than "Size", consecutive Size values starting from InOut[] are initialized to zero.

(5) Program example

<table>
<tr><th>DEMO</th><th>LD</th><th>ST</th></tr>
<tr><td>Define the variable</td><td colspan="2">
<pre>VAR
    R_TRIG_0:R_TRIG;
    Execute :BOOL;
    In_0    :ARRAY[0..9] OF INT:=[0,1,2,3,4,5,6,7,8,9];
    uiSize  :UINT:=6;
    uiNum   :UINT:=2;
END_VAR</pre>
R_TRIG is not defined and used in LD
</td></tr>
<tr><td>Sample program</td><td>


</td><td>
<pre>1 R_TRIG_0(CLK:=,Q=>Execute);
2 IF Execute THEN
3     ArySHL(InOut:= In_0[1],
4         Size:= uiSize,
5         num:= uiNum);
6 END_IF</pre>
</td></tr>
<tr><td>Operation result</td><td colspan="2">


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

| | |
|---|---|
| Principle analysis |  |

1-4-3-9. Shift N elements of an array left/right NSHLC/NSHRC

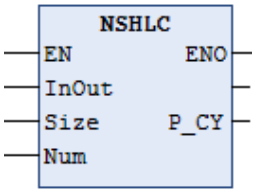
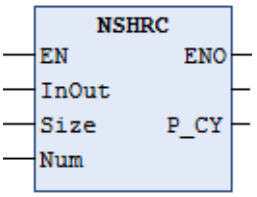
(1) Instruction overview

Move multiple bits along with the carry (CY) flag.

NSHLC: Shift to the left (high bit direction).

NSHRC: Shift to the right (low position direction).

| Shift N elements of an array left/right [NSHLC/NSHRC] | | | |
|---|---|---|---|
| Execution condition | Edge triggering | Suitable model | XS, XSA |
| Firmware | - | Software | XS StudioV2.2.0 and above |
| Library | XJ_DataUtils | | |

| Instruction | Name | Graph | ST |
|---|---|---|---|
| NSHLC | Left shift of N elements in an array |  | xResult:=NSHLC(InOut:= ,<br>Size:= ,<br>Num:= ,<br>P_CY=> ); |
| NSHRC | Right shift of N elements in an array |  | xResult:=NSHRC(InOut:= ,<br>Size:= ,<br>Num:= ,<br>P_CY=> ); |

(2) Related variable

| VAR_INPUT | Name | Data type | Effective range | Initial value | Description |
|---|---|---|---|---|---|
| Size | The number of bits of the shifted object | USINT | As the data type | 1 | The number of bits of the shifted object |
| Num | Shift bits | USINT | As the data type | 1 | Shift bits |
| VAR_OUTPUT | Name | Data type | Effective range | Initial value | Description |
| P_CY | Carry flag | BOOL | As the data type | - | Carry flag |
| VAR_IN_OUT | Name | Data type | Effective range | Initial value | Description |
| InOut[] array | Shift Object Array | BOOL/BYTE/WORD /DWORD/LWORD | As the data type | - | Shift Object Array |

(3) Function

- Move the "Size" array elements starting from the InOut [0] of the shifted object array InOut [] by "Num" bits. The final value of the overflow bit will be output to the carry (CY) flag, and the bit inserted on the opposite side will be filled with 0.
- NSHLC: Shift to the left (from low to high).

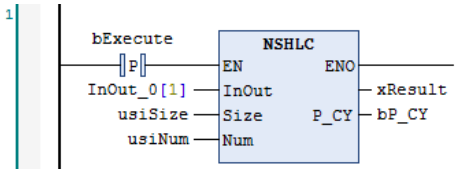
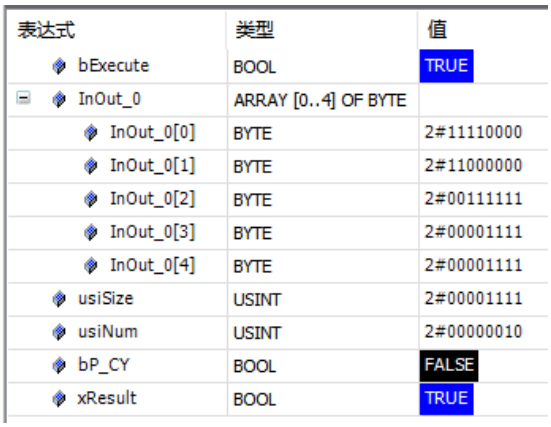
- NSHRC: Shift to the right (from high to low).

(4) Note

- When the value of "Num" is 0, no shift is performed, the function returns a value of True, InOut[] remains unchanged, and the value of "P_CY" is FALSE.
- When the value of "Num" is greater than "Size", the value from bit0 of InOut[] to the "Size" bits is FALSE. In addition, the value of the carry (CY) flag is also FALSE.
- When the value of "Size" exceeds the array area of InOut[], a PLC exception will occur as shown in the figure below. This incorrect usage should be avoided.

- "InOut[]" requires Boolean and bitstring types, otherwise the runtime function returns FALSE.

(5) Program sample

| DEMO | LD | ST |
|---|---|---|
| Define the variable | VAR<br>bExecute:BOOL;<br>InOut_0:ARRAY[0..4] OF BYTE:=[2(2#11110000),3(2#00001111)];<br>usiSize:USINT:=15;<br>usiNum :USINT:=2;<br>bP_CY :BOOL;<br>xResult:BOOL;<br>END_VAR | |
| Sample program |  | 1 R_TRIG_0(CLK:=,Q=>bExecute);<br>2 IF bExecute THEN<br>3 xResult:=NSHLC(InOut:= InOut_0[1],<br>4 Size:= usiSize,<br>5 Num:= usiNum,<br>6 P_CY=> bP_CY);<br>7 END_IF |
| Operation result |  | |

| 表达式 | 类型 | 值 |
|---|---|---|
| bExecute | BOOL | TRUE |
| InOut_0 | ARRAY [0..4] OF BYTE | |
| InOut_0[0] | BYTE | 2#11110000 |
| InOut_0[1] | BYTE | 2#11000000 |
| InOut_0[2] | BYTE | 2#00111111 |
| InOut_0[3] | BYTE | 2#00001111 |
| InOut_0[4] | BYTE | 2#00001111 |
| usiSize | USINT | 2#00001111 |
| usiNum | USINT | 2#00000010 |
| bP_CY | BOOL | FALSE |
| xResult | BOOL | TRUE |

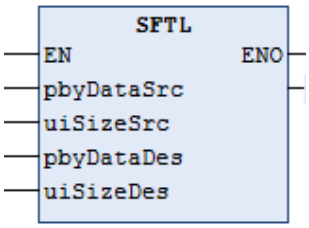
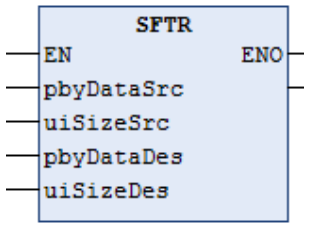
1-4-3-10. Copy bit data to the left/right SFTL/SFTR

(1) Instruction overview

SFTL: Copy the bit data of the specified length from the source array continuously to the left into the target array.

SFTR: Copy the bit data of the specified length from the source array continuously to the right into the target array.

| Copy bit data to the left/right [SFTL/SFTR] | | | |
|---|---|---|---|
| Execution condition | Edge triggering | Suitable model | XS, XSA |
| Firmware | - | Software | XS StudioV2.2.0 and above |
| Library | XJ_DataUtils | | |

| Instruction | Name | Graph | ST |
|---|---|---|---|
| SFTL | Copy bit data to the left |  | SFTL(pbyDataSrc:= ,<br>uiSizeSrc:= ,<br>pbyDataDes:= ,<br>uiSizeDes:= ); |
| SFTR | Copy bit data to the right |  | SFTR(pbyDataSrc:= ,<br>uiSizeSrc:= ,<br>pbyDataDes:= ,<br>uiSizeDes:= ); |

(2) Related variables

| VAR_INPUT | Name | Data type | Effective range | Initial value | Description |
|---|---|---|---|---|---|
| pbyDataSrc | Source data | POINTER_TO_BYTE | - | - | Source data |
| pbyDataDes | Target data | POINTER_TO_BYTE | - | - | Target data |
| uiSizeSrc | Data source numbers | BYTE | - | - | Data source numbers |
| uiSizeDes | Target data numbers | BYTE | - | - | Target data numbers |

(3) Function

- uiSizeSrc must larger than uiSizeDes, otherwise, the PLC will shut down abnormally.

(4) Program sample

**① SFTL**

Copy the consecutive uiSizeSrc (number of elements in a single copy) bits of data starting from the first address of pointer pbyDataSrc (source array) to the low range of the first address of pbyDataDes (target array). Before copying, all uiSizeDes (offset target) bits of data in pbyDataDes are moved to the high range by uiSizeSrc bits, and only within the range of uiSizeDes (offset target) bits of data in pbyDataDes (target array).

**ST：**

```
PROGRAM SFTL_ST
VAR
    R_TRIG_0:R_TRIG;
    uiSizeSrc:BYTE;
    uiSizeDes:BYTE;
    源数据:ARRAY[0..3]OF BOOL;
    目标数据:ARRAY[0..5]OF BOOL;
    Execute:BOOL;
END_VAR
```

```
R_TRIG_0(CLK:=,Q=>Execute);
IF Execute=1 THEN
    SFTL(pbyDataSrc:=ADR(源数据[0]),
        uiSizeSrc:=uiSizeSrc,
        pbyDataDes:=ADR(目标数据[0]),
        uiSizeDes:=uiSizeDes);
END_IF
```

Define variables and programming;

| 表达式 | 类型 | 值 |
|---|---|---|
| R_TRIG_0 | R_TRIG | |
| CLK | BOOL | FALSE |
| Q | BOOL | FALSE |
| uiSizeSrc | BYTE | 2 |
| uiSizeDes | BYTE | 5 |
| 源数据 | ARRAY [0..3] OF BO... | |
| 源数据[0] | BOOL | TRUE |
| 源数据[1] | BOOL | TRUE |
| 源数据[2] | BOOL | FALSE |
| 源数据[3] | BOOL | FALSE |
| 目标数据 | ARRAY [0..5] OF BO... | |
| 目标数据[0] | BOOL | FALSE |
| 目标数据[1] | BOOL | FALSE |
| 目标数据[2] | BOOL | FALSE |
| 目标数据[3] | BOOL | FALSE |
| 目标数据[4] | BOOL | FALSE |
| 目标数据[5] | BOOL | FALSE |
| Execute | BOOL | FALSE |

```
R_TRIG_0(CLK:=,Q FALSE =>Execute FALSE );
IF Execute FALSE =1 THEN
    SFTL(pbyDataSrc:=ADR(源数据[0] TRUE ),
        uiSizeSrc:=uiSizeSrc 2 ,
        pbyDataDes:=ADR(目标数据[0] FALSE ),
        uiSizeDes:=uiSizeDes 5 );
END_IF RETURN
```

Initial assignment;

| 表达式 | 类型 | 值 |
|---|---|---|
| R_TRIG_0 | R_TRIG | |
| CLK | BOOL | TRUE |
| Q | BOOL | FALSE |
| uiSizeSrc | BYTE | 2 |
| uiSizeDes | BYTE | 5 |
| 源数据 | ARRAY [0..3] OF BO... | |
| 源数据[0] | BOOL | TRUE |
| 源数据[1] | BOOL | TRUE |
| 源数据[2] | BOOL | FALSE |
| 源数据[3] | BOOL | FALSE |
| 目标数据 | ARRAY [0..5] OF BO... | |
| 目标数据[0] | BOOL | TRUE |
| 目标数据[1] | BOOL | TRUE |
| 目标数据[2] | BOOL | FALSE |
| 目标数据[3] | BOOL | FALSE |
| 目标数据[4] | BOOL | FALSE |
| 目标数据[5] | BOOL | FALSE |
| Execute | BOOL | FALSE |

```
1  R_TRIG_0(CLK:=,Q FALSE =>Execute FALSE );
2  IF Execute FALSE =1 THEN
3      SFTL(pbyDataSrc:=ADR(源数据[0] TRUE ),
4          uiSizeSrc:=uiSizeSrc 2 ,
5          pbyDataDes:=ADR(目标数据[0] TRUE ),
6          uiSizeDes:=uiSizeDes 5 );
7  END_IF RETURN
```

Triggering for the first time, as the target data is all FALSE, shifting 2 bits towards the high bit is still FALSE; Source data [0], source data [1] are passed to target data [0], and the target data [1] is true;

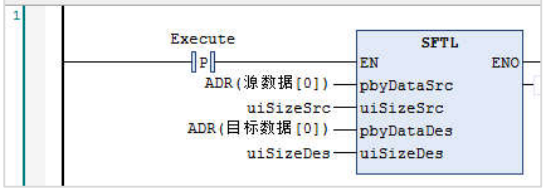
| 表达式 | 类型 | 值 |
|---|---|---|
| R_TRIG_0 | R_TRIG | |
| CLK | BOOL | TRUE |
| Q | BOOL | FALSE |
| uiSizeSrc | BYTE | 2 |
| uiSizeDes | BYTE | 5 |
| 源数据 | ARRAY [0..3] OF BO... | |
| 源数据[0] | BOOL | TRUE |
| 源数据[1] | BOOL | TRUE |
| 源数据[2] | BOOL | FALSE |
| 源数据[3] | BOOL | FALSE |
| 目标数据 | ARRAY [0..5] OF BO... | |
| 目标数据[0] | BOOL | TRUE |
| 目标数据[1] | BOOL | TRUE |
| 目标数据[2] | BOOL | TRUE |
| 目标数据[3] | BOOL | TRUE |
| 目标数据[4] | BOOL | FALSE |
| 目标数据[5] | BOOL | FALSE |
| Execute | BOOL | FALSE |

```
1  R_TRIG_0(CLK:=,Q FALSE =>Execute FALSE );
2  IF Execute FALSE =1 THEN
3      SFTL(pbyDataSrc:=ADR(源数据[0] TRUE ),
4          uiSizeSrc:=uiSizeSrc 2 ,
5          pbyDataDes:=ADR(目标数据[0] TRUE ),
6          uiSizeDes:=uiSizeDes 5 );
7  END_IF RETURN
```

Triggering the second time, the value of the original target data [0] - target data [4] is moved 2 bits to the high position, to the target data [2] - target data [4], and the overflow 2 bits are moved to the target data [0] - target data [1]; Next, the value of the source data [0] and source data [1] are passed to the target data [0] and target data [1].

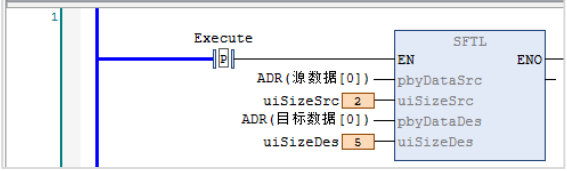
**LD:**

```
PROGRAM SFTL_LD
VAR
    uiSizeSrc:BYTE;
    uiSizeDes:BYTE;
    源数据:ARRAY[0..3]OF BOOL;
    目标数据:ARRAY[0..5]OF BOOL;
    Execute:BOOL;
END_VAR
```



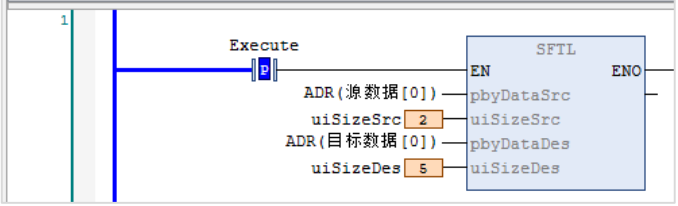

Defining variables and programming.

| 表达式 | 类型 | 值 |
|---|---|---|
| uiSizeSrc | BYTE | 2 |
| uiSizeDes | BYTE | 5 |
| 源数据 | ARRAY [0..3] OF BOOL | |
| 源数据[0] | BOOL | TRUE |
| 源数据[1] | BOOL | TRUE |
| 源数据[2] | BOOL | FALSE |
| 源数据[3] | BOOL | FALSE |
| 目标数据 | ARRAY [0..5] OF BOOL | |
| 目标数据[0] | BOOL | FALSE |
| 目标数据[1] | BOOL | FALSE |
| 目标数据[2] | BOOL | FALSE |
| 目标数据[3] | BOOL | FALSE |
| 目标数据[4] | BOOL | FALSE |
| 目标数据[5] | BOOL | FALSE |
| Execute | BOOL | FALSE |



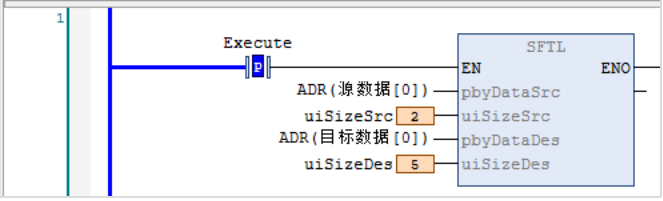

Initial assignment.

| 表达式 | 类型 | 值 |
|---|---|---|
| uiSizeSrc | BYTE | 2 |
| uiSizeDes | BYTE | 5 |
| 源数据 | ARRAY [0..3] OF BOOL | |
| 源数据[0] | BOOL | TRUE |
| 源数据[1] | BOOL | TRUE |
| 源数据[2] | BOOL | FALSE |
| 源数据[3] | BOOL | FALSE |
| 目标数据 | ARRAY [0..5] OF BOOL | |
| 目标数据[0] | BOOL | TRUE |
| 目标数据[1] | BOOL | TRUE |
| 目标数据[2] | BOOL | FALSE |
| 目标数据[3] | BOOL | FALSE |
| 目标数据[4] | BOOL | FALSE |
| 目标数据[5] | BOOL | FALSE |
| Execute | BOOL | TRUE |




Triggering for the first time, as the target data is all FALSE, shifting 2 bits towards the high bit is still FALSE; Source data [0], source data [1] are passed to target data [0], and the target data [1] is true.

| 表达式 | 类型 | 值 |
|---|---|---|
| uiSizeSrc | BYTE | 2 |
| uiSizeDes | BYTE | 5 |
| 源数据 | ARRAY [0..3] OF BOOL | |
| 源数据[0] | BOOL | TRUE |
| 源数据[1] | BOOL | TRUE |
| 源数据[2] | BOOL | FALSE |
| 源数据[3] | BOOL | FALSE |
| 目标数据 | ARRAY [0..5] OF BOOL | |
| 目标数据[0] | BOOL | TRUE |
| 目标数据[1] | BOOL | TRUE |
| 目标数据[2] | BOOL | TRUE |
| 目标数据[3] | BOOL | TRUE |
| 目标数据[4] | BOOL | FALSE |
| 目标数据[5] | BOOL | FALSE |
| Execute | BOOL | TRUE |




Triggering the second time, the value of the original target data [0] - target data [4] is moved 2 bits to the high position, to the target data [2] - target data [4], and the overflow 2 bits are moved to the target data [0] - target data [1]; Next, the value of the source data [0] and source data [1] is passed to the target data [0] and target data [1].

### ② SFTR

Copy the consecutive uiSizeSrc (number of array elements in a single copy) bit data starting from the first address of pointer pbyDataSrc (source array) to the high bit area of pbyDataDes (target array). Before copying, all uiSizeSrc (offset target) bit data of pbyDataDes will be shifted low by uiSizeSrc bits, and only within the range of uiSizeDes (offset target) data of pbyDataDes (target array).

**ST：**

```
PROGRAM SFTL_ST
VAR
    R_TRIG_0:R_TRIG;
    uiSizeSrc:BYTE;
    uiSizeDes:BYTE;
    源数据:ARRAY[0..3]OF BOOL;
    目标数据:ARRAY[0..5]OF BOOL;
    Execute:BOOL;
END_VAR
```

```
R_TRIG_0(CLK:=,Q=>Execute);
IF Execute=1 THEN
    SFTR(pbyDataSrc:=ADR(源数据[0]),
        uiSizeSrc:=uiSizeSrc,
        pbyDataDes:=ADR(目标数据[0]),
        uiSizeDes:=uiSizeDes);
END_IF
```

Defining variables and programming.

| 表达式 | 类型 | 值 |
| --- | --- | --- |
| R_TRIG_0 | R_TRIG | |
| CLK | BOOL | FALSE |
| Q | BOOL | FALSE |
| uiSizeSrc | BYTE | 2 |
| uiSizeDes | BYTE | 5 |
| 源数据 | ARRAY [0..3] OF BOOL | |
| 源数据[0] | BOOL | TRUE |
| 源数据[1] | BOOL | TRUE |
| 源数据[2] | BOOL | FALSE |
| 源数据[3] | BOOL | FALSE |
| 目标数据 | ARRAY [0..5] OF BOOL | |
| 目标数据[0] | BOOL | FALSE |
| 目标数据[1] | BOOL | FALSE |
| 目标数据[2] | BOOL | FALSE |
| 目标数据[3] | BOOL | FALSE |
| 目标数据[4] | BOOL | FALSE |
| 目标数据[5] | BOOL | FALSE |
| Execute | BOOL | FALSE |

```
1  R_TRIG_0(CLK:=,Q FALSE =>Execute FALSE );
2  IF Execute FALSE =1 THEN
3      SFTR(pbyDataSrc:=ADR(源数据[0] TRUE ),
4          uiSizeSrc:=uiSizeSrc 2 ,
5          pbyDataDes:=ADR(目标数据[0] FALSE ),
6          uiSizeDes:=uiSizeDes 5 );
7  END_IF RETURN
```

Initial assignment.

| 表达式 | 类型 | 值 |
| --- | --- | --- |
| R_TRIG_0 | R_TRIG | |
| CLK | BOOL | TRUE |
| Q | BOOL | FALSE |
| uiSizeSrc | BYTE | 2 |
| uiSizeDes | BYTE | 5 |
| 源数据 | ARRAY [0..3] OF BOOL | |
| 源数据[0] | BOOL | TRUE |
| 源数据[1] | BOOL | TRUE |
| 源数据[2] | BOOL | FALSE |
| 源数据[3] | BOOL | FALSE |
| 目标数据 | ARRAY [0..5] OF BOOL | |
| 目标数据[0] | BOOL | FALSE |
| 目标数据[1] | BOOL | FALSE |
| 目标数据[2] | BOOL | FALSE |
| 目标数据[3] | BOOL | TRUE |
| 目标数据[4] | BOOL | TRUE |
| 目标数据[5] | BOOL | FALSE |
| Execute | BOOL | FALSE |

```
1  R_TRIG_0(CLK:=,Q FALSE =>Execute FALSE );
2  IF Execute FALSE =1 THEN
3      SFTR(pbyDataSrc:=ADR(源数据[0] TRUE ),
4          uiSizeSrc:=uiSizeSrc 2 ,
5          pbyDataDes:=ADR(目标数据[0] FALSE ),
6          uiSizeDes:=uiSizeDes 5 );
7  END_IF RETURN
```

Triggering for the first time, as the target data is all FALSE, shifting 2 bits towards the low bit is still FALSE; Source data [0], source data [1] are passed to target data [3], and the target data [4] is true.

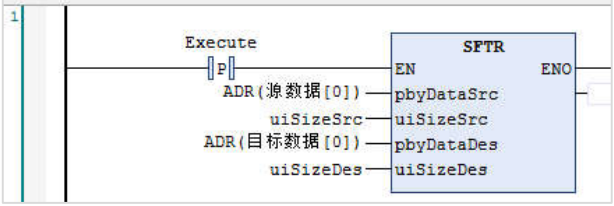
| 表达式 | 类型 | 值 |
| --- | --- | --- |
| R_TRIG_0 | R_TRIG | |
| CLK | BOOL | TRUE |
| Q | BOOL | FALSE |
| uiSizeSrc | BYTE | 2 |
| uiSizeDes | BYTE | 5 |
| 源数据 | ARRAY [0..3] OF BOOL | |
| 源数据[0] | BOOL | TRUE |
| 源数据[1] | BOOL | TRUE |
| 源数据[2] | BOOL | FALSE |
| 源数据[3] | BOOL | FALSE |
| 目标数据 | ARRAY [0..5] OF BOOL | |
| 目标数据[0] | BOOL | FALSE |
| 目标数据[1] | BOOL | TRUE |
| 目标数据[2] | BOOL | TRUE |
| 目标数据[3] | BOOL | TRUE |
| 目标数据[4] | BOOL | TRUE |
| 目标数据[5] | BOOL | FALSE |
| Execute | BOOL | FALSE |

```
1  R_TRIG_0(CLK:=,Q=>Execute);
2  IF Execute=1 THEN
3      SFTR(pbyDataSrc:=ADR(源数据[0]),
4          uiSizeSrc:=uiSizeSrc,
5          pbyDataDes:=ADR(目标数据[0]),
6          uiSizeDes:=uiSizeDes);
7  END_IF
```

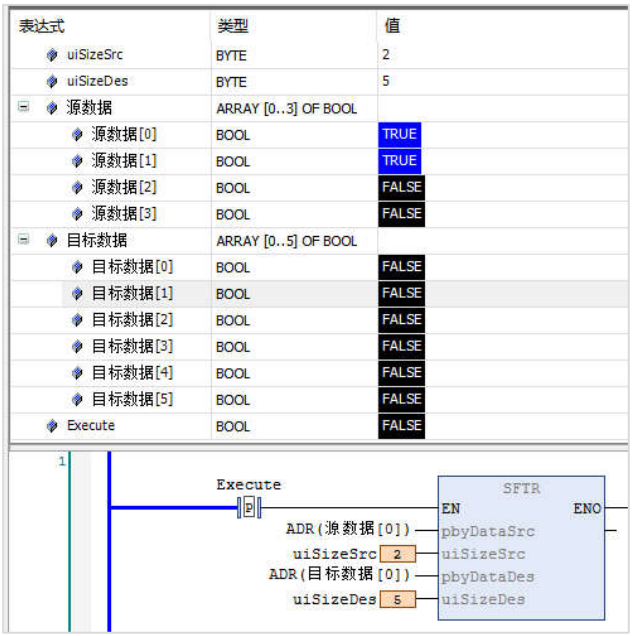
Triggering the second time, the value of the original target data [0] - target data [4] is moved 2 bits to the low position, to the target data [0] - target data [2], and the overflow 2 bits are moved to the target data [3] - target data [4]; Next, the value of the source data [0] and source data [1] is passed to the target data [3] and target data [4].

**LD:**

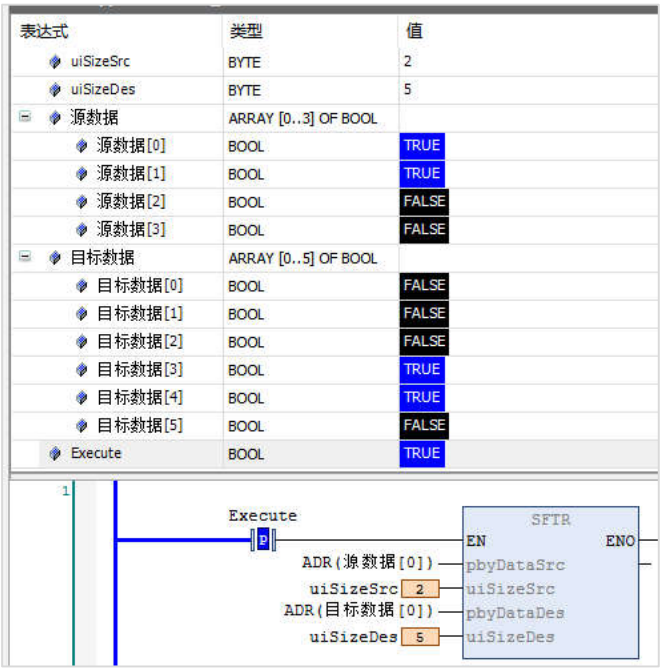
```
1  PROGRAM SFTL_LD
2  VAR
3      uiSizeSrc:BYTE;
4      uiSizeDes:BYTE;
5      源数据:ARRAY[0..3]OF BOOL;
6      目标数据:ARRAY[0..5]OF BOOL;
7      Execute:BOOL;
8  END_VAR
```




Defining variables and programming.

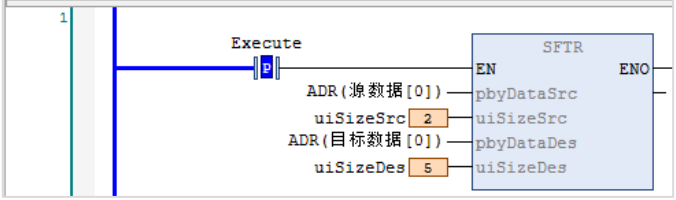



Initial assignment.




Triggering for the first time, as the target data is all FALSE, shifting 2 bits towards the low bit is still FALSE; Source data [0], source data [1] are passed to target data [3], and the target data [4] is true.

| 表达式 | 类型 | 值 |
| --- | --- | --- |
| uiSizeSrc | BYTE | 2 |
| uiSizeDes | BYTE | 5 |
| 源数据 | ARRAY [0..3] OF BOOL | |
| 源数据[0] | BOOL | TRUE |
| 源数据[1] | BOOL | TRUE |
| 源数据[2] | BOOL | FALSE |
| 源数据[3] | BOOL | FALSE |
| 目标数据 | ARRAY [0..5] OF BOOL | |
| 目标数据[0] | BOOL | FALSE |
| 目标数据[1] | BOOL | TRUE |
| 目标数据[2] | BOOL | TRUE |
| 目标数据[3] | BOOL | TRUE |
| 目标数据[4] | BOOL | TRUE |
| 目标数据[5] | BOOL | FALSE |
| Execute | BOOL | TRUE |




Triggering the second time, the value of the original target data [0] - target data [4] is moved 2 bits to the low position, to the target data [0] - target data [2], and the overflow 2 bits are moved to the target data [3] - target data [4]; Next, the value of the source data [0] and source data [1] is passed to the target data [3] and target data [4].

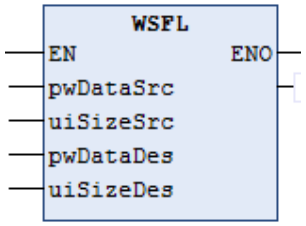
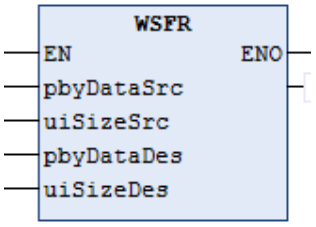
1-4-3-11. Copy word data to the left/right WSFL/WSFR

(1) Instruction overview

WSFL: Copy word data of the specified length from the source array continuously to the left into the target array.

WSFR: Copy word data of the specified length from the source array continuously to the right into the target array.

| Copy word data to the left/right [WSFL/WSFR] | | | |
|---|---|---|---|
| Execution condition | Edge triggering | Suitable model | XS, XSA |
| Firmware | - | Software | XS StudioV2.2.0 and above |
| Library | XJ_DataUtils | | |

| Instruction | Name | Graph | ST |
|---|---|---|---|
| WSFL | Copy word data to the left |  | WSFL(pwDataSrc:= ,uiSizeSrc:= , pwDataDes:= , uiSizeDes:= ); |
| WSFR | Copy word data to the right |  | WSFR(pwDataSrc:= , uiSizeSrc:= , pwDataDes:= , uiSizeDes:= ); |

(2) Related variable

| VAR_INPUT | Name | Data type | Effective range | Initial value |
|---|---|---|---|---|
| pbyDataSrc | Source data | POINTER_TO_WORD | - | - |
| pbyDataDes | Target data | POINTER_TO_WORD | - | - |
| uiSizeSrc | Source data numbers | UINT | 1-1024 | - |
| uiSizeDes | Target data numbers | UINT | 1-1024 | - |

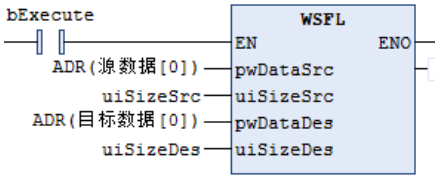
(3) Note

- UiSizeSrc must not be greater than uiSizeDes, otherwise the PLC will shut down abnormally.

(4) Sample program

| DEMO | LD | ST |
|---|---|---|
| Define variables | VAR<br>R_TRIG_0:R_TRIG;<br>bExecute: BOOL;<br>源数据:ARRAY[0..3] OF INT:=[10,20,30,40];<br>目标数据:ARRAY[0..9]OF INT:=[1,2,3,4,5,6,7,8,9,10];<br>uiSizeSrc:UINT;<br>uiSizeDes:UINT;<br>END_VAR | |

Sample program




```
R_TRIG_0(CLK:=,Q=>bExecute);
IF bExecute=1 THEN
    WSFL(pwDataSrc:= ADR(源数据[0]),
         uiSizeSrc:= uiSizeSrc,
         pwDataDes:= ADR(目标数据[0]),
         uiSizeDes:= uiSizeDes);
END_IF
```

Variable assignment

| 表达式 | 类型 | 值 |
|---|---|---|
| R_TRIG_0 | R_TRIG | |
| CLK | BOOL | FALSE |
| Q | BOOL | FALSE |
| bExecute | BOOL | FALSE |
| 源数据 | ARRAY [0..3] OF INT | |
| 源数据[0] | INT | 10 |
| 源数据[1] | INT | 20 |
| 源数据[2] | INT | 30 |
| 源数据[3] | INT | 40 |
| 目标数据 | ARRAY [0..9] OF INT | |
| 目标数据[0] | INT | 1 |
| 目标数据[1] | INT | 2 |
| 目标数据[2] | INT | 3 |
| 目标数据[3] | INT | 4 |
| 目标数据[4] | INT | 5 |
| 目标数据[5] | INT | 6 |
| 目标数据[6] | INT | 7 |
| 目标数据[7] | INT | 8 |
| 目标数据[8] | INT | 9 |
| 目标数据[9] | INT | 10 |
| uiSizeSrc | UINT | 2 |
| uiSizeDes | UINT | 6 |

| | |
|---|---|
| Operation twice | |

| 表达式 | 类型 | 值 |
|---|---|---|
| R_TRIG_0 | R_TRIG | |
| CLK | BOOL | TRUE |
| Q | BOOL | FALSE |
| bExecute | BOOL | FALSE |
| 源数据 | ARRAY [0..3] OF INT | |
| 源数据[0] | INT | 10 |
| 源数据[1] | INT | 20 |
| 源数据[2] | INT | 30 |
| 源数据[3] | INT | 40 |
| 目标数据 | ARRAY [0..9] OF INT | |
| 目标数据[0] | INT | 10 |
| 目标数据[1] | INT | 20 |
| 目标数据[2] | INT | 10 |
| 目标数据[3] | INT | 20 |
| 目标数据[4] | INT | 1 |
| 目标数据[5] | INT | 2 |
| 目标数据[6] | INT | 7 |
| 目标数据[7] | INT | 8 |
| 目标数据[8] | INT | 9 |
| 目标数据[9] | INT | 10 |
| uiSizeSrc | UINT | 2 |
| uiSizeDes | UINT | 6 |

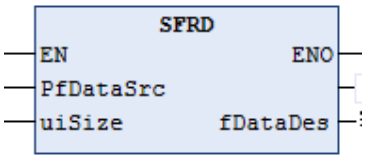
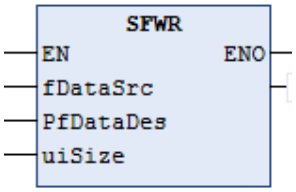
1-4-3-12. First in, first out data read/write SFRD/SFWR

(1) Instruction overview

SFRD: Read the content of a specified length of data after execution.

SFWR: Write the content of a specified length of data after execution.

| First in, first out data read/write [SFRD/SFWR] | | | |
|---|---|---|---|
| Execution condition | Edge triggering | Suitable model | XS, XSA |
| Firmware | - | Software | XS StudioV2.2.0 and above |
| Library | XJ_DataUtils | | |

| Instruction | Name | Graph | ST |
|---|---|---|---|
| SFRD | First in, first out data reading |  | SFRD(pfDataSrc:=, uiSize:=, fDataDes=>); |
| SFWR | First in, first out data writing |  | SFWR(fDataSrc:=, pfDataDes:=, uiSize:= ); |

(2) Related variables

| VAR_INPUT | Name | Data type | Effective range | Initial value |
|---|---|---|---|---|
| PfDataSrc | Source data | POINTER_TO_REAL | - | - |
| uiSize | The number of data floating point elements or the source data queue | UINT | - | - |
| fDataDes | Read data | REAL | - | - |
| VAR_OUTPUT | Name | Data type | Effective range | Initial value |
| fDataSrc | Source data | REAL | - | - |
| PfDataDes | Target data | POINTER_TO_REAL | - | - |
| uiSize | The number of data floating point elements or the source data queue | UINT | - | - |

(3) Note

- The first element of the source data needs to be assigned in advance, otherwise the execution will be abnormal.

(4) Sample program

**① SFRD**

Read the first item of the first in, first out queue pfDataSrc (pointer pointing to source data) into fDataDes (target data), and then move the queue pfDatasrc to the right one word word by word, decreasing the queue pointer. Using the first numbered device as the pointer, when the instruction is executed, the content value of the pointer is first reduced by 1, and then the content value of the device specified by pfDataSrc is written to the position specified by

the pointer in the first in, first out fDataDes data string. If the pointer is already 0, the instruction does not process the aforementioned operation.

**ST：**

| 表达式 | 类型 | 值 |
|---|---|---|
| 源数据 | ARRAY [0..9] OF REAL | |
| 源数据[0] | REAL | 1 |
| 源数据[1] | REAL | 1 |
| 源数据[2] | REAL | 2 |
| 源数据[3] | REAL | 3 |
| 源数据[4] | REAL | 4 |
| 源数据[5] | REAL | 5 |
| 源数据[6] | REAL | 6 |
| 源数据[7] | REAL | 7 |
| 源数据[8] | REAL | 8 |
| 源数据[9] | REAL | 9 |
| 数据长度 | UINT | 10 |
| 结果数据 | REAL | 0 |
| bExecute | BOOL | FALSE |
| R_TRIG_0 | R_TRIG | |
| CLK | BOOL | FALSE |
| Q | BOOL | FALSE |

```
1  源数据[0] 1 :=1;
2  R_TRIG_0(CLK:= , Q FALSE => bExecute FALSE );
3  IF bExecute FALSE =1 THEN
4      SFRD(PfDataSrc:= ADR(源数据),
5          uiSize:= 数据长度 10 ,
6          fDataDes=> 结果数据 0 );
7  END_IF RETURN
```

Initial status.

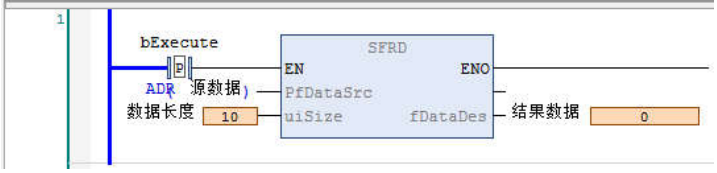
| 表达式 | 类型 | 值 |
|---|---|---|
| 源数据 | ARRAY [0..9] OF REAL | |
| 源数据[0] | REAL | 1 |
| 源数据[1] | REAL | 4 |
| 源数据[2] | REAL | 5 |
| 源数据[3] | REAL | 6 |
| 源数据[4] | REAL | 7 |
| 源数据[5] | REAL | 8 |
| 源数据[6] | REAL | 9 |
| 源数据[7] | REAL | 9 |
| 源数据[8] | REAL | 9 |
| 源数据[9] | REAL | 9 |
| 数据长度 | UINT | 10 |
| 结果数据 | REAL | 3 |
| bExecute | BOOL | FALSE |
| R_TRIG_0 | R_TRIG | |
| CLK | BOOL | TRUE |
| Q | BOOL | FALSE |

```
1  源数据[0] 1 :=1;
2  R_TRIG_0(CLK:= , Q FALSE => bExecute FALSE );
3  IF bExecute FALSE =1 THEN
4      SFRD(PfDataSrc:= ADR(源数据),
5          uiSize:= 数据长度 10 ,
6          fDataDes=> 结果数据 3 );
7  END_IF RETURN
```

After triggering three times.

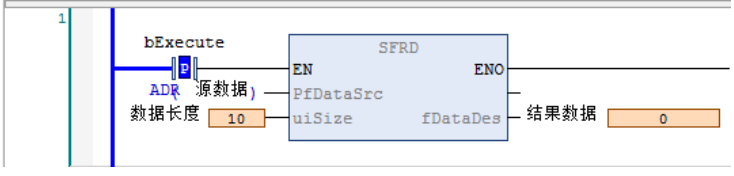
**LD：**

| 表达式 | 类型 | 值 | 准备值 |
|---|---|---|---|
| 源数据 | ARRAY [0..9] OF REAL | | |
| 源数据[0] | REAL | 1 | |
| 源数据[1] | REAL | 1 | |
| 源数据[2] | REAL | 2 | |
| 源数据[3] | REAL | 3 | |
| 源数据[4] | REAL | 4 | |
| 源数据[5] | REAL | 5 | |
| 源数据[6] | REAL | 6 | |
| 源数据[7] | REAL | 7 | |
| 源数据[8] | REAL | 8 | |
| 源数据[9] | REAL | 9 | |
| 数据长度 | UINT | 10 | |
| 结果数据 | REAL | 0 | |
| bExecute | BOOL | FALSE | |




Initial status.

| 表达式 | 类型 | 值 | 准备值 |
|---|---|---|---|
| 源数据 | ARRAY [0..9] OF REAL | | |
| 源数据[0] | REAL | 0 | |
| 源数据[1] | REAL | 2 | |
| 源数据[2] | REAL | 3 | |
| 源数据[3] | REAL | 4 | |
| 源数据[4] | REAL | 5 | |
| 源数据[5] | REAL | 6 | |
| 源数据[6] | REAL | 7 | |
| 源数据[7] | REAL | 8 | |
| 源数据[8] | REAL | 9 | |
| 源数据[9] | REAL | 9 | |
| 数据长度 | UINT | 10 | |
| 结果数据 | REAL | 0 | |
| bExecute | BOOL | TRUE | |




After triggering three times.

**② SFWR**

Write the value of fDataSrc (source data) into a "first in, first out" queue starting from the address of pfDataDes (target data) with a number of uiSize (data length), using the first numbered device as the pointer. When the instruction is executed, the pointer's content value first increases by 1, and then the content value of the device specified by fDataSrc is written to the position specified by the pointer in the first in, first out pfDataDes data string.

**ST：**

| 表达式 | 类型 | 值 |
| --- | --- | --- |
| R_TRIG_0 | R_TRIG | |
| CLK | BOOL | FALSE |
| Q | BOOL | FALSE |
| 源数据 | REAL | 1 |
| 数据长度 | UINT | 10 |
| 结果数据 | ARRAY [0..9] OF REAL | |
| 结果数据[0] | REAL | 0 |
| 结果数据[1] | REAL | 0 |
| 结果数据[2] | REAL | 0 |
| 结果数据[3] | REAL | 0 |
| 结果数据[4] | REAL | 0 |
| 结果数据[5] | REAL | 0 |
| 结果数据[6] | REAL | 0 |
| 结果数据[7] | REAL | 0 |
| 结果数据[8] | REAL | 0 |
| 结果数据[9] | REAL | 0 |
| bExecute | BOOL | FALSE |

```
1  R_TRIG_0(CLK:= , Q=> bExecute);
2  IF bExecute=1 THEN
3      SFWR(fDataSrc:= 源数据[1],
4          PfDataDes:= ADR(结果数据),
5          uiSize:= 数据长度[10]);
6  END_IF
```

Initial status.

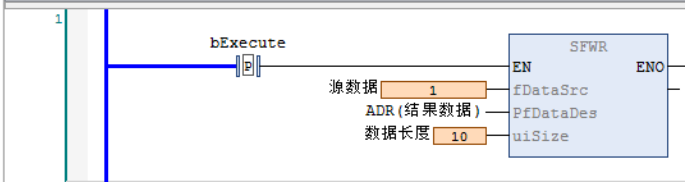
| 表达式 | 类型 | 值 |
| --- | --- | --- |
| R_TRIG_0 | R_TRIG | |
| CLK | BOOL | TRUE |
| Q | BOOL | FALSE |
| 源数据 | REAL | 1 |
| 数据长度 | UINT | 10 |
| 结果数据 | ARRAY [0..9] OF REAL | |
| 结果数据[0] | REAL | 3 |
| 结果数据[1] | REAL | 1 |
| 结果数据[2] | REAL | 1 |
| 结果数据[3] | REAL | 1 |
| 结果数据[4] | REAL | 0 |
| 结果数据[5] | REAL | 0 |
| 结果数据[6] | REAL | 0 |
| 结果数据[7] | REAL | 0 |
| 结果数据[8] | REAL | 0 |
| 结果数据[9] | REAL | 0 |
| bExecute | BOOL | FALSE |

```
1  R_TRIG_0(CLK:= , Q=> bExecute);
2  IF bExecute=1 THEN
3      SFWR(fDataSrc:= 源数据[1],
4          PfDataDes:= ADR(结果数据),
5          uiSize:= 数据长度[10]);
6  END_IF
```

After triggering three times.

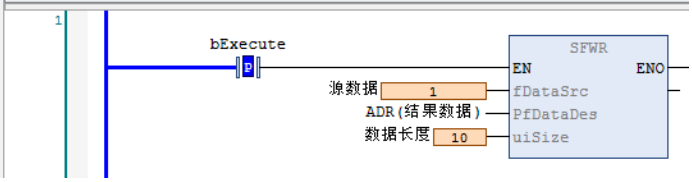
**LD:**

| 表达式 | 类型 | 值 | 准备值 |
|---|---|---|---|
| 源数据 | REAL | 1 | |
| 数据长度 | UINT | 10 | |
| 结果数据 | ARRAY [0..9] OF REAL | | |
| 结果数据[0] | REAL | 0 | |
| 结果数据[1] | REAL | 0 | |
| 结果数据[2] | REAL | 0 | |
| 结果数据[3] | REAL | 0 | |
| 结果数据[4] | REAL | 0 | |
| 结果数据[5] | REAL | 0 | |
| 结果数据[6] | REAL | 0 | |
| 结果数据[7] | REAL | 0 | |
| 结果数据[8] | REAL | 0 | |
| 结果数据[9] | REAL | 0 | |
| bExecute | BOOL | FALSE | |




Initial status.

| 表达式 | 类型 | 值 | 准备值 |
|---|---|---|---|
| 源数据 | REAL | 1 | |
| 数据长度 | UINT | 10 | |
| 结果数据 | ARRAY [0..9] OF REAL | | |
| 结果数据[0] | REAL | 3 | |
| 结果数据[1] | REAL | 1 | |
| 结果数据[2] | REAL | 1 | |
| 结果数据[3] | REAL | 1 | |
| 结果数据[4] | REAL | 0 | |
| 结果数据[5] | REAL | 0 | |
| 结果数据[6] | REAL | 0 | |
| 结果数据[7] | REAL | 0 | |
| 结果数据[8] | REAL | 0 | |
| 结果数据[9] | REAL | 0 | |
| bExecute | BOOL | TRUE | |




After triggering three times.

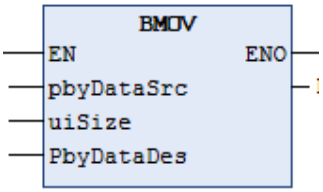
## 1-4-4. Move instructions

### 1-4-4-1. Bulk transfer BMOV

(1) Instruction overview

Copy the source data to the target data at the specified length.

| Bulk transfer [BMOV] | | | |
|---|---|---|---|
| Execution condition | Edge triggering | Suitable model | XS, XSA |
| Firmware | - | Software | XS StudioV2.2.0 and above |
| Library | XJ_DataUtils | | |

| Instruction | Name | Graph | ST |
|---|---|---|---|
| BMOV | Batch transmission of data |  | BMOV(pbyDataSrc:= ,<br>uiSize:= ,<br>PbyDataDes:=); |

(2) Related variables

| VAR_INPUT | Name | Data type | Effective range | Initial value |
|---|---|---|---|---|
| pbyDataSrc | Starting address for data replication | POINTER TO BYTE | - | - |
| uiSize | Number of variables | UINT | As the data type | 0 |
| VAR_IN_OUT | Name | Data type | Effective range | Initial value |
| PbyDataDes | Starting address for data storage | POINTER TO BYTE | - | - |

(3) Function

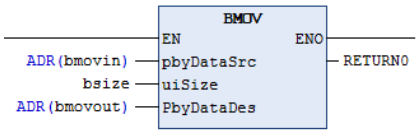
- This instruction is contact driven, copying the uiSize variable values of the specified starting address of pbyDataSrc to the uiSize units of the specified starting address of pbyDataDes. The range of uiSize values is 1-1024. Return value, True: Execution successful, FALSE: Execution error.

(4) Note

- Uisize is byte numbers.

(5) Sample program

| DEMO | LD | ST |
|---|---|---|
| Define variables | | VAR<br><br>bmovin:ARRAY[0..5] OF INT;<br>bmovout:ARRAY[0..5] OF INT;<br>bsize:UINT;<br>RETURN0: BOOL;<br><br>END_VAR |

<table>
<tr><td>Sample program</td><td> 

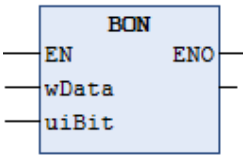
<pre>BMOV(pbyDataSrc:=ADR(bmovin) ,
    uiSize:=bsize ,
    PbyDataDes:=ADR(bmovout) );</pre></td></tr>
<tr><td>Operation result</td><td>
<table>
<tr><th>表达式</th><th>类型</th><th>值</th></tr>
<tr><td>bmovin</td><td>ARRAY [0..5] OF INT</td><td></td></tr>
<tr><td>bmovin[0]</td><td>INT</td><td>111</td></tr>
<tr><td>bmovin[1]</td><td>INT</td><td>222</td></tr>
<tr><td>bmovin[2]</td><td>INT</td><td>333</td></tr>
<tr><td>bmovin[3]</td><td>INT</td><td>444</td></tr>
<tr><td>bmovin[4]</td><td>INT</td><td>555</td></tr>
<tr><td>bmovin[5]</td><td>INT</td><td>666</td></tr>
<tr><td>bmovout</td><td>ARRAY [0..5] OF INT</td><td></td></tr>
<tr><td>bmovout[0]</td><td>INT</td><td>111</td></tr>
<tr><td>bmovout[1]</td><td>INT</td><td>222</td></tr>
<tr><td>bmovout[2]</td><td>INT</td><td>333</td></tr>
<tr><td>bmovout[3]</td><td>INT</td><td>444</td></tr>
<tr><td>bmovout[4]</td><td>INT</td><td>555</td></tr>
<tr><td>bmovout[5]</td><td>INT</td><td>0</td></tr>
<tr><td>bsize</td><td>UINT</td><td>10</td></tr>
<tr><td>RETURN0</td><td>BOOL</td><td>TRUE</td></tr>
</table>
</td></tr>
<tr><td>Result analysis</td><td>Uisize represents the number of bytes, with data set to 10; The array element is defined as INT, which is 2 bytes. Therefore, it transfers the data from bmovin[0] - bmovin[4] to bmovout[0] - bmovout[4].</td></tr>
</table>

1-4-4-2. Obtain data specified bit status BON

(1) Instruction overview

Determine the status of the uiBit bit of data wData, store the result in BON, and output it from the right side.

| Obtain data specified bit status [BON] | | | |
|---|---|---|---|
| Execution condition | Edge triggering | Suitable model | XS, XSA |
| Firmware | - | Software | XS StudioV2.2.0 and above |
| Library | XJ_DataUtils | | |

| Instruction | Name | Graph | ST |
|---|---|---|---|
| BON | Obtain data specified bit status |  | BON(wData:= ,<br>uiBit:= ); |

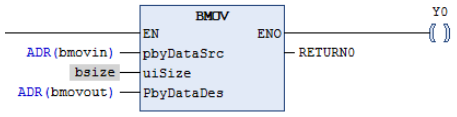
(2) Related variables

| VAR_INPUT | Name | Data type | Effective range | Initial value | Description |
|---|---|---|---|---|---|
| wData | Data that needs to be judged | WORD | As the data type | 0 | Data that needs to be judged (decimal) |
| uiBit | The number of data bits | UINT | As the data type | 0 | The number of data bits (binary bit numbers) |

(3) Function

- Determine the status of the uiBit bit of data wData, store the result in BON, and output it from the right side.

(4) Sample program

| DEMO | LD | ST |
|---|---|---|
| Define variables | | VAR<br>wData0: WORD;<br>uibit0: UINT;<br>Y0: BOOL;<br>RETURN0: BOOL;<br>END_VAR |
| Sample program |  | Y0:=BON(wData:=wData0,<br>uiBit:=uisize0); |
| Operation result | | |
| Result | Set the value of wData0 to 22 (binary display 10110), set uibit0 to 2, that is, bit2, with a value of | |

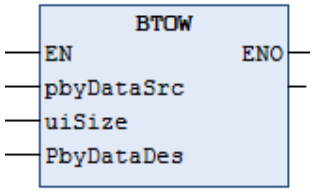
| | |
|---|---|
| analysis | 1, and output the result as True. |

1-4-4-3. Data combination of byte unit BTOW

(1) Instruction overview

An instruction that combines the low 8 bits (low bytes) of continuous 16 bits data together.

| Data combination of byte unit [BTOW] | | | |
|---|---|---|---|
| Execution condition | Edge triggering | Suitable model | XS, XSA |
| Firmware | - | Software | XS StudioV2.2.0 and above |
| Library | XJ_DataUtils | | |

| Instruction | Name | Graph | ST |
|---|---|---|---|
| BTOW | Data combination of byte unit |  | BTOW(pbyDataSrc:= ,<br>uiSize:= ,<br>PbyDataDes:= ); |

(2) Related variables

| VAR_INPUT | Name | Data type | Effective range | Initial value | Description |
|---|---|---|---|---|---|
| pbyDataSrc | Starting address of combined data | POINTER TO BYTE | - | - | Starting address of combined data |
| uiSize | Number of combined data | UINT | As the data type | 0 | Number of combined data |
| VAR_IN_OUT | Name | Data type | Effective range | Initial value | Description |
| PbyDataDes | Target address for storing combined data | POINTER TO BYTE | - | - | Target address for storing combined data |

(3) Function

- Enter the source data in byDataSrc and the number of combined data in uiSize. The instruction combines the low 8 bits of multiple 16 bits data in the input source data to form a new 16 bits data. The newly generated data is saved to the starting address of pwDataDes, and the high 8 bits of pbyDataSrc are ignored.

(4) Note

- When the SIZE value is 0, the function returns a value of True.

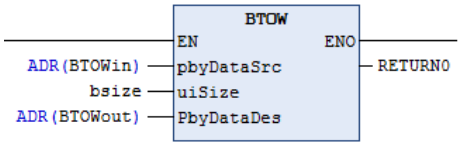
(5) Sample program

<table>
<tr><th>DEMO</th><th>LD</th><th>ST</th></tr>
<tr><td>Define variables</td><td colspan="2">

```
VAR

    bsize:UINT;
    BTOWin:ARRAY[0..5] OF INT;
    BTOWout:ARRAY[0..5] OF INT;
    RETURN0: BOOL;

END_VAR
```

</td></tr>
<tr><td>Sample program</td><td> 
</td><td>

```
BTOW(pbyDataSrc:=ADR(BTOWin) ,
    uiSize:=bsize ,
    PbyDataDes:=ADR(BTOWout) );
```

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

| 表达式 | 类型 | 值 |
|---|---|---|
| bsize | UINT | 6 |
| BTOWin | ARRAY [0..5] OF INT | |
| BTOWin[0] | INT | 1 |
| BTOWin[1] | INT | 1 |
| BTOWin[2] | INT | 1 |
| BTOWin[3] | INT | 1 |
| BTOWin[4] | INT | 1 |
| BTOWin[5] | INT | 1 |
| BTOWout | ARRAY [0..5] OF INT | |
| BTOWout[0] | INT | 257 |
| BTOWout[1] | INT | 257 |
| BTOWout[2] | INT | 257 |
| BTOWout[3] | INT | 0 |
| BTOWout[4] | INT | 0 |
| BTOWout[5] | INT | 0 |
| RETURN0 | BOOL | TRUE |

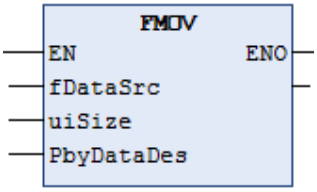
</td></tr>
<tr><td>Result analysis</td><td colspan="2">Bsize represents the number of bytes, and the data is set to 6; Convert the two low octets of BTOWin [0] - BTOWin [5] in order, i.e. 01 data, into a new 16 bits data (0101) and store it in BTOWout [0] - BTOWout [2] in sequence.</td></tr>
</table>

1-4-4-4. One to many data transfer FMOV

(1) Instruction overview

Copy the source data array specified bit to multiple bits of the target array, such as transferring the value of A1 [1] to A2 [0,1,2].

| One to many data transfer [FMOV] | | | |
|---|---|---|---|
| Execution condition | Edge triggering | Suitable model | XS, XSA |
| Firmware | - | Software | XS StudioV2.2.0 and above |
| Library | XJ_DataUtils | | |

| Instruction | Name | Graph | ST |
|---|---|---|---|
| FMOV | One to many data transfer |  | FMOV(fDataSrc:= ,<br>uiSize:= ,<br>byDataDes:= ); |

(2) Related variables

| VAR_INPUT | Name | Data type | Effective range | Initial value | Description |
|---|---|---|---|---|---|
| fDataSrc | Starting address for data replication | REAL | As the data type | 0 | Starting address for data replication |
| uiSize | Number of variables | UINT | As the data type | 0 | Number of variables |
| VAR_IN_OUT | Name | Data type | Effective range | Initial value | Description |
| PbyDataDes | Starting address for data storage | POINTER TO BYTE | - | - | Starting address for data storage |

(3) Function

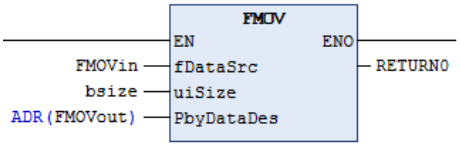
- Transfer the data from fDataSrc to uiSize units at the specified starting address of PbyDataDes, where the uiSize data range is from 0 to 1024.

(4) Note

- UiSize represents the number of transfers.

(5) Sample program

| DEMO | LD | ST |
|---|---|---|
| Define variables | VAR<br><br>FMOVin: REAL;<br>FMOVout:ARRAY[0..10] OF REAL;<br>bsize:UINT;<br>RETURN0: BOOL;<br><br>END_VAR | |

| | | |
|---|---|---|
| Sample program |  | FMOV(fDataSrc:=FMOVin ,<br>uiSize:=bsize ,<br>PbyDataDes:=ADR(FMOVout) ); |
| Operation result | | |

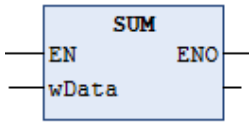
| 表达式 | 类型 | 值 |
|---|---|---|
| FMOVin | REAL | 22 |
| FMOVout | ARRAY [0..10] OF R... | |
| FMOVout[0] | REAL | 22 |
| FMOVout[1] | REAL | 22 |
| FMOVout[2] | REAL | 22 |
| FMOVout[3] | REAL | 22 |
| FMOVout[4] | REAL | 22 |
| FMOVout[5] | REAL | 22 |
| FMOVout[6] | REAL | 0 |
| FMOVout[7] | REAL | 0 |
| FMOVout[8] | REAL | 0 |
| FMOVout[9] | REAL | 0 |
| FMOVout[10] | REAL | 0 |
| bsize | UINT | 6 |
| RETURN0 | BOOL | TRUE |

1-4-4-5. ON bit SUM

(1) Instruction overview

Count the number of "1" in binary data, store the statistical results in SUM, and output them from the right side.

| ON bit sum [SUM] | | | |
|---|---|---|---|
| Execution condition | Edge triggering | Suitable model | XS, XSA |
| Firmware | - | Software | XS StudioV2.2.0 and above |
| Library | XJ_DataUtils | | |

| Instruction | Name | Graph | ST |
|---|---|---|---|
| SUM | Total ON bits |  | SUM(wData:= ); |

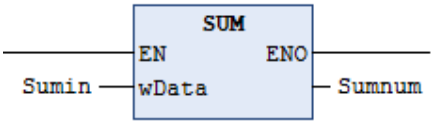
(2) Related variables

| VAR_INPUT | Name | Data type | Effective range | Initial value | Description |
|---|---|---|---|---|---|
| wData | Data that needs to be judged | WORD | As the data type | 0 | Data to be judged (decimal) |

(3) Function

- Count the number of "1" in binary data, and store the statistical results in SUM.

(4) Sample program

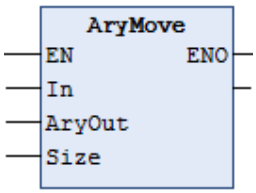
| DEMO | LD | ST |
|---|---|---|
| Define variables | | VAR<br>Sumin: WORD;<br>sumnum: INT;<br>END_VAR |
| Sample program |  | SUMnum:=SUM(wData:=SUMin ); |
| Operation result | | |
| Result analysis | Sumin data input 22 (binary display 10110), so Sumnum displays 3. | |

1-4-4-6. Array transfer AryMove

(1) Instruction overview

Transfer multiple array elements.

| Array transfer [AryMove] | | | |
|---|---|---|---|
| Execution condition | Edge triggering | Suitable model | XS, XSA |
| Firmware | - | Software | XS StudioV2.2.0 and above |
| Library | XJ_DataUtils | | |

| Instruction | Name | Graph | ST |
|---|---|---|---|
| AryMove | Array transfer |  | AryMove(In:= ,<br>AryOut:= ,<br>Size:= ); |

(2) Related variables

| VAR_INPUT | Name | Data type | Effective range | Initial value | Description |
|---|---|---|---|---|---|
| In | The array to be transferred | BOOL/BYTE/WORD /DWORD/LWORD/ USINT/UINT/UDINT /ULINT/SINT/INT/DINT /LINT/REAL/LREAL/TIME /DATE/TOD/DT/STRING | As the data type | - | The array to be transferred |
| VAR_IN_OUT | Name | Data type | Effective range | Initial value | Description |
| AryOut | Transfer result array | BOOL/BYTE/WORD /DWORD/LWORD/ USINT/UINT/UDINT /ULINT/SINT/INT/DINT /LINT/REAL/LREAL/TIME /DATE/TOD/DT/STRING | As the data type | - | Transfer result array |
| Size | Number of elements to be transmitted | UINT | As the data type | 0 | Number of elements to be transmitted |

(3) Function

- Transfer the "Size" elements starting from In[0] of the transfer object array In[] to the elements starting from AryOut[0] of the transfer result array AryOut[].

(4) Note

- The data types of In[] and AryOut[] elements should be consistent, otherwise the runtime function return value is FALSE and AryOut[] remains unchanged.
- When the value of "Size" is 0, the function returns a value of True and there is no change in AryOut[].
- The length of In[] should not be greater than the memory size of AryOut[] to avoid PLC exceptions.

- The value of "Size" should not exceed the array area of In[] and AryOut[] to avoid PLC exceptions.

(5) Sample program

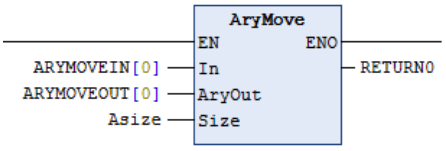
<table>
<tr><th>DEMO</th><th>LD</th><th>ST</th></tr>
<tr><td>Define variables</td><td colspan="2">

```
VAR

    ARYMOVEIN:ARRAY[0..5] OF INT;
    ARYMOVEOUT:ARRAY[0..5] OF INT;
    Asize: UINT;
    RETURN0: BOOL;

END_VAR
```

</td></tr>
<tr><td>Sample program</td><td>




</td><td>

```
AryMove(In:=ARYMOVEIN[0],
    AryOut:=ARYMOVEOUT[0] ,
    Size:=bsize );
```

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

| 表达式 | 类型 | 值 |
|---|---|---|
| ARYMOVEIN | ARRAY [0..5] OF INT | |
| ARYMOVEIN[0] | INT | 11 |
| ARYMOVEIN[1] | INT | 22 |
| ARYMOVEIN[2] | INT | 33 |
| ARYMOVEIN[3] | INT | 44 |
| ARYMOVEIN[4] | INT | 55 |
| ARYMOVEIN[5] | INT | 66 |
| ARYMOVEOUT | ARRAY [0..5] OF INT | |
| ARYMOVEOUT[0] | INT | 11 |
| ARYMOVEOUT[1] | INT | 22 |
| ARYMOVEOUT[2] | INT | 33 |
| ARYMOVEOUT[3] | INT | 44 |
| ARYMOVEOUT[4] | INT | 55 |
| ARYMOVEOUT[5] | INT | 66 |
| Asize | UINT | 6 |
| RETURN0 | BOOL | TRUE |

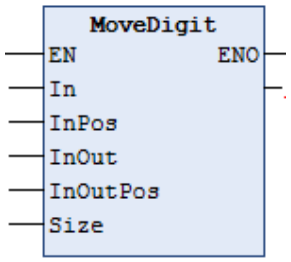
</td></tr>
</table>

1-4-4-7. Digit transfer MoveDigit

(1) Instruction overview

Transfer multiple digits in a sequence (1 digit is 4 bits).

| Digit transfer [MoveDigit] | | | |
|---|---|---|---|
| Execution condition | Edge triggering | Suitable model | XS, XSA |
| Firmware | - | Software | XS StudioV2.2.0 and above |
| Library | XJ_DataUtils | | |

| Instruction | Name | Graph | ST |
|---|---|---|---|
| MoveDigit | Digit transfer |  | MoveDigit(In:= ,<br>InPos:= ,<br>InOut:= ,<br>InOutPos:= ,<br>Size:= ); |

(2) Related variables

| VAR_INPUT | Name | Data type | Effective range | Initial value | Description |
|---|---|---|---|---|---|
| In | Sender | BYTE/WORD/ DWORD/LWORD | As the data type | - | Sender |
| Inpos | Number position of the sender | USINT | As the data type | 0 | The position of the number transmitted in "In" |
| InOut | Receiver | BYTE/WORD/ DWORD/LWORD | As the data type | - | Receiver |
| InOutPos | Receiver's numerical position | USINT | As the data type | 0 | The numerical position of the receiver in "Out" |
| Size | Number of digits to be transmitted | USINT | As the data type | 1 | Number of digits to be transmitted |

(3) Function

- Transfer the "Size" digits starting from the number position "InPos" of the sender "In" to the number position "InOutPos" of the receiver "InOut", where one digit is 4 bits.

(4) Note

- If the specified receiver digit position exceeds the highest position of "InOut", the excess data will be saved to the lowest position of "InOut".
- If the specified number position of the sender exceeds the highest position of "In", the excess data will be transmitted to the lowest position of "In".
- When the value of "Size" is 0, the function returns a value of True and "InOut" remains unchanged.
- An exception will occur in the following situations. The return value of the function becomes FALSE, and "InOut" remains unchanged.
  - When the value of "InPos" exceeds the valid range.

- When the value of "InOutPos" exceeds the valid range.
- When the value of "Size" exceeds the valid range.

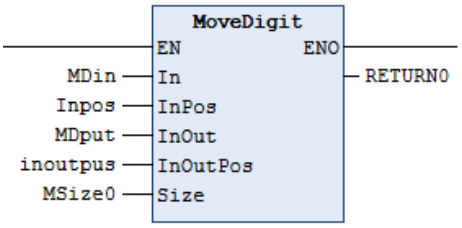
(5) Sample program

<table>
<tr><th>DEMO</th><th>LD</th><th>ST</th></tr>
<tr><td>Define variables</td><td colspan="2">

```
VAR
    MDin:WORD;
    MDput:WORD;
    Inpos: USINT;
    inoutpus: USINT;
    MSize0: USINT;
    RETURN0: BOOL;

END_VAR
```

</td></tr>
<tr><td>Sample program</td><td>




</td><td>

```
MoveDigit(In:= MDin,
    InPos:= Inpos,
    InOut:=MDput,
    InOutPos:= inoutpus,
    Size:=MSize0 );
```

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

| 表达式 | 类型 | 值 |
|---|---|---|
| MDin | WORD | 16#1234 |
| MDput | WORD | 16#9934 |
| Inpos | USINT | 16#00 |
| inoutpus | USINT | 16#00 |
| MSize0 | USINT | 16#02 |
| RETURN0 | BOOL | TRUE |

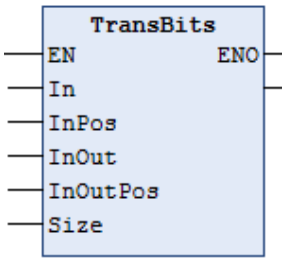
</td></tr>
</table>

1-4-4-8. Multiple bits transfer TransBits

(1) Instruction overview

Transfer multiple bits of a sequence.

| Multiple bits transfer [TransBits] | | | |
|---|---|---|---|
| Execution condition | Edge triggering | Suitable model | XS, XSA |
| Firmware | - | Software | XS StudioV2.2.0 and above |
| Library | XJ_DataUtils | | |

| Instruction | Name | Graph | ST |
|---|---|---|---|
| TransBits | Multiple bits transfer |  | TransBits(In:= ,<br>InPos:= ,<br>InOut:= ,<br>InOutPos:= ,<br>Size:= ); |

(2) Related variables

| VAR_INPUT | Name | Data type | Effective range | Initial value | Description |
|---|---|---|---|---|---|
| In | Sender | BYTE/WORD/ DWORD/LWORD | As the data type | - | Sender |
| Inpos | Bit position of the sender | USINT | 0~'in' bits -1 | - | The position of the transmitted bit in "In" |
| InOut | Receiver | BYTE/WORD/ DWORD/LWORD | As the data type | - | Receiver |
| InOutPos | Bit position of the receiver | USINT | 0~'inout' bits -1 | - | The position of receiver bit in "Out" |
| Size | Number of digits to be transmitted | USINT | 0~'in' and 'inout' bits -1 | - | Number of digits to be transmitted |

(3) Function

- Transfer the "Size" bits starting from bit position "InPos" of the sender "In" to bit position "InOutPos" of the receiver "InOut".

(4) Note

- When the value of "Size" is 0, it is not transmitted and the function returns a value of True.
- The bits in "InOut" that are not related to transmission remain unchanged.
- In the following situations, an exception will occur, with the function returning a value of FALSE and InOut remaining unchanged.
  - When the value of "InPos" exceeds the valid range.
  - When the value of "InOutPos" exceeds the valid range.
  - When the value of "Size" exceeds the valid range.

- When the specified values of "InPos" and "Size" exceed the number of digits of "In".
- When the specified values of "InOutPos" and "Size" exceed the number of digits of "InOut".

(5) Sample program

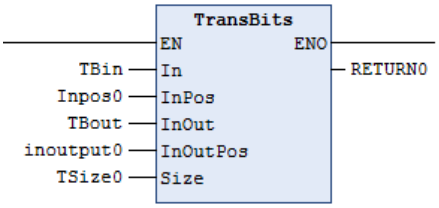
<table>
<tr><th>DEMO</th><th>LD</th><th>ST</th></tr>
<tr><td>Define variables</td><td colspan="2">

```
VAR

    RETURN0: BOOL;
    TBin:WORD;
    TBout:WORD;
    Inpos0: USINT;
    inoutput0: USINT;
    TSize0: USINT;

END_VAR
```

</td></tr>
<tr><td>Sample program</td><td>




</td><td>

```
TransBits(In:= TBin,
    InPos:= Inpos0,
    InOut:=TBout ,
    InOutPos:=inoutput0 ,
    Size:=TSize0 );
```

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

| 表达式 | 类型 | 值 |
|---|---|---|
| RETURN0 | BOOL | TRUE |
| TBin | WORD | 2#0000000000000000 |
| TBout | WORD | 2#0000000000011100 |
| Inpos0 | USINT | 2#00000000 |
| inoutput0 | USINT | 2#00000000 |
| TSize0 | USINT | 2#00000010 |

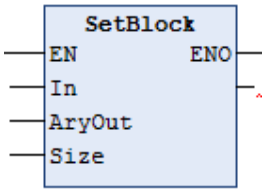
</td></tr>
<tr><td>Result analysis</td><td colspan="2">Set TBin (data set to 0), starting from input0 (data set to 0), two pieces of data are transferred to TBout (initial data bit 2#11111), starting from inoutpos0 (value 0), two pieces of data are transferred to TBout (TBout data is transferred to 2#11100), and the function returns a value of TRUE.</td></tr>
</table>

1-4-4-9. Set block SetBlock

(1) Instruction overview

Transfer the values of variables and constants to multiple array elements.

| Set block [SetBlock] | | | |
|---|---|---|---|
| Execution condition | Edge triggering | Suitable model | XS, XSA |
| Firmware | - | Software | XS StudioV2.2.0 and above |
| Library | XJ_DataUtils | | |

| Instruction | Name | Graph | ST |
|---|---|---|---|
| SetBlock | Set block |  | SetBlock(In:= ,<br>AryOut:= ,<br>Size:= ); |

(2) Related variables

| VAR_INPUT | Name | Data type | Effective range | Initial value | Description |
|---|---|---|---|---|---|
| In | Transmission source | BOOL/BYTE/WORD/DWORD/LWORD/USINT/UINT/UDINT/ULINT/SINT/INT/DINT/LINT/REAL/LREAL/TIME/DATE/TOD/DT/STRING | As the data type | - | The array to be transferred |
| Size | Number of elements to be transmitted | UINT | As the data type | 0 | Number of elements to be transmitted |
| VAR_IN_OUT | Name | Data type | Effective range | Initial value | Description |
| AryOut | Transfer target array | BOOL/BYTE/WORD/DWORD/LWORD/USINT/UINT/UDINT/ULINT/SINT/INT/DINT/LINT/REAL/LREAL/TIME/DATE/TOD/DT/STRING | As the data type | - | Transfer result array |

(3) Function

- Transfer the value of the transfer source "In" to "Size" positions starting from the AryOut[0] of the transfer target array AryOut[].

(4) Note

- When the data types of "In" and "AryOut" are different, the value returned by the function is FALSE, while "AryOut" remains unchanged.
- The area sizes of "In" and "AryOut" should be consistent to avoid PLC anomalies.
- When the value of "Size" is 0, the value returned by the function is true, and AryOut[] remains unchanged.
- The value of "Size" should not exceed the array area of AryOut[] to avoid PLC exceptions.

- When "In" and AryOut[] are unsupported data types, an exception will occur during compilation.

(5) Sample program

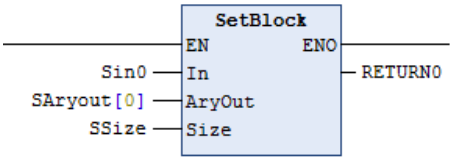
<table>
<tr><th>DEMO</th><th>LD</th><th>ST</th></tr>
<tr><td>Define variables</td><td colspan="2">

```
VAR

    RETURN0: BOOL;
    Sin0:INT;
    SAryout:ARRAY[0..5] OF INT;
    SSize:UINT;

END_VAR
```

</td></tr>
<tr><td>Sample program</td><td>




</td><td>

```
SetBlock(In:= Sin0,
    AryOut:= SAryout[0],
    Size:=SSize );
```

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

| 表达式 | 类型 | 值 |
|---|---|---|
| RETURN0 | BOOL | TRUE |
| Sin0 | INT | 11 |
| SAryout | ARRAY [0..5] OF INT | |
| SAryout[0] | INT | 11 |
| SAryout[1] | INT | 11 |
| SAryout[2] | INT | 11 |
| SAryout[3] | INT | 11 |
| SAryout[4] | INT | 0 |
| SAryout[5] | INT | 0 |
| SSize | UINT | 4 |

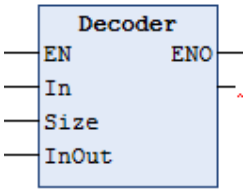
</td></tr>
</table>

1-4-4-10. Bit decoder Decoder

(1) Instruction overview

Set one of the arranged elements consisting of up to 256 bits as True and the other bits as FALSE.

| Bit decoder [Decoder] | | | |
|---|---|---|---|
| Execution condition | Edge triggering | Suitable model | XS, XSA |
| Firmware | - | Software | XS StudioV2.2.0 and above |
| Library | XJ_DataUtils | | |

| Instruction | Name | Graph | ST |
|---|---|---|---|
| Decoder | Bit decoder |  | Decoder(In:= ,<br>Size:= ,<br>InOut:= ); |

(2) Related variables

| VAR_INPUT | Name | Data type | Effective range | Initial value | Description |
|---|---|---|---|---|---|
| In | Conversion bit position | BYTE | As the data type | 0 | Conversion bit position |
| Size | Convert bits | USINT | 0~8 | 1 | Convert bits |
| VAR_IN_OUT | Name | Data type | Effective range | Initial value | Description |
| InOut | Convert Object Array | BOOL/BYTE/WORD /DWORD/LWORD | As the data type | - | Convert Object Array |

(3) Function

- Set the specified bit of the 2^"Size" bits starting from element InOut[0] in the transformation object array InOut[] to True, and set the other bits to FALSE.
- Among them, the bit position to be set as True is specified by the conversion bit position "In".
- The input parameters passed to InOut[] should be specified along with the element number, such as array[3].

(4) Note

- The value of "Size" should not exceed the number of digits of the "InOut" array element to avoid PLC exceptions.
- The value of "Size" should not exceed the valid range to avoid PLC anomalies.
- When the value of "Size" is 0, the function returns a value of True and InOut[] remains unchanged.
- InOut[] requires the use of Boolean and bitstring types, otherwise the program will compile an error at runtime.

(5) Sample program

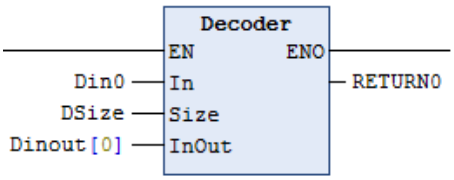
<table>
<tr><th>DEMO</th><th>LD</th><th>ST</th></tr>
<tr><td>Define variables</td><td colspan="2">

```
VAR

    Din0: BYTE;
    DSize: USINT;
    DAryout:ARRAY[0..5] OF BYTE;
    RETURN0: BOOL;

END_VAR
```

</td></tr>
<tr><td>Sample program</td><td>




</td><td>

```
Decoder(In:= Din0,
    Size:=DSize ,
    InOut:=Dinout[0] );
```

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

| 表达式 | 类型 | 值 |
|---|---|---|
| Din0 | BYTE | 2#00000010 |
| DSize | USINT | 2#00000100 |
| Dinout | ARRAY [0..5] OF BYTE | |
| Dinout[0] | BYTE | 2#00000100 |
| Dinout[1] | BYTE | 2#00000000 |
| Dinout[2] | BYTE | 2#11111111 |
| Dinout[3] | BYTE | 2#11111111 |
| Dinout[4] | BYTE | 2#11111111 |
| Dinout[5] | BYTE | 2#11111111 |
| RETURN0 | BOOL | TRUE |

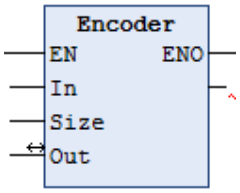
</td></tr>
<tr><td>Result analysis</td><td colspan="2">Set the Dinout array data to 2#11111111; Run the command to set the 2^4(DSize) bits starting from Dinout[0] data to 0, except bit2(Din0) data which is set to 1.</td></tr>
</table>

1-4-4-11. Bit encoder Encoder

(1) Instruction overview

Calculate the bit position with a value of True in a permutation element composed of up to 256 bits.

| Bit decoder [Encoder] | | | |
|---|---|---|---|
| Execution condition | Edge triggering | Suitable model | XS, XSA |
| Firmware | - | Software | XS StudioV2.2.0 and above |
| Library | XJ_DataUtils | | |

| Instruction | Name | Graph | ST |
|---|---|---|---|
| Encoder | Bit encoder |  | Encoder(In:= ,<br>Size:= ,<br>Out:= ); |

(2) Related variables

| VAR_INPUT | Name | Data type | Effective range | Initial value | Description |
|---|---|---|---|---|---|
| In | Convert object arrangement | BOOL/BYTE/WORD /DWORD/LWORD | As the data type | - | Convert object arrangement |
| Size | Convert bits | USINT | 0~8 | 1 | Convert bits |
| VAR_IN_OUT | Name | Data type | Effective range | Initial value | Description |
| AryOut | Convert result | BYTE | As the data type | 0 | Convert result |

(3) Function

- Calculate the bit position with a value of True within any range of the transformation object arrangement In[].
- The range for calculating bit positions is 2^"Size" bits starting from In[0]. The positions of the bits with a value of True within this range are represented in binary and saved in the conversion result "Out". Then, save FALSE in the remaining bits of "Out".
- If there are multiple bits with a value of True within the specified range, calculate the position of the highest bit among them.
- The input parameters passed to In[] should be specified together with the element number, such as Array[3].
- When "Size"=USINT#4 and In[] is a BYTE arrangement, since "Size"=USINT#4, the object range for calculating the TRUE bit position is 2^4=16 starting from In[0].

(4) Note

- When the value of "Size" is 0, the function returns true and "Out" remains unchanged.
- When "Size" exceeds the valid range, avoid PLC anomalies.
- When the value of "Size" exceeds the number of digits of In[] arranged elements, an incorrect value may be obtained.
- When all the values specified in "Size" in the In[] bit are FALSE, the program encounters an exception and the function returns a value of FALSE, while the value of "Out" remains unchanged.
- In[] requires the use of Boolean and bit string types, otherwise the program compilation will report an error at runtime.

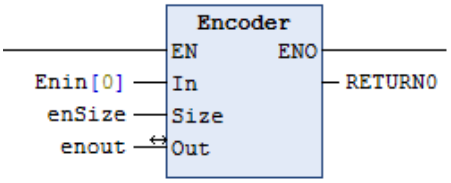
(5) Sample program

<table>
<tr><th>DEMO</th><th>LD</th><th>ST</th></tr>
<tr><td>Define variables</td><td colspan="2">

```
VAR

    Enin:ARRAY[0..5] OF BYTE;
    enSize: USINT;
    enout: BYTE;
    RETURN0: BOOL;

END_VAR
```

</td></tr>
<tr><td>Sample program</td><td> 
</td><td>

```
Encoder(In:=enin[0] ,
    Size:=enSize ,
    Out:=enout );
```

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

| 表达式 | 类型 | 值 |
|---|---|---|
| Enin | ARRAY [0..5] OF BYTE | |
| Enin[0] | BYTE | 2#11111111 |
| Enin[1] | BYTE | 2#11111111 |
| Enin[2] | BYTE | 2#11111111 |
| Enin[3] | BYTE | 2#11111111 |
| Enin[4] | BYTE | 2#11111111 |
| Enin[5] | BYTE | 2#11111111 |
| enSize | USINT | 2#00000100 |
| enout | BYTE | 2#00001111 |
| RETURN0 | BOOL | TRUE |

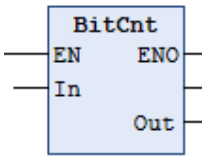
</td></tr>
</table>

1-4-4-12. Bit count BitCnt

(1) Instruction overview

Calculate the total number of bits in a string with a value of True.

| Bit count [BitCnt] | | | |
|---|---|---|---|
| Execution condition | Edge triggering | Suitable model | XS, XSA |
| Firmware | - | Software | XS StudioV2.2.0 and above |
| Library | XJ_DataUtils | | |

| Instruction | Name | Graph | ST |
|---|---|---|---|
| BitCnt | Bit count |  | `BitCnt(In:= , Out=> );` |

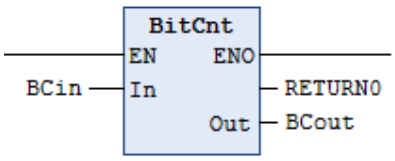
(2) Related variables

| VAR_INPUT | Name | Data type | Effective range | Initial value | Description |
|---|---|---|---|---|---|
| In | Count object | BYTE/WORD/ DWORD/LWORD | As the data type | - | Count object |
| VAR_IN_OUT | Name | Data type | Effective range | Initial value | Description |
| Out | Calculate the bits | USINT | As the data type | 0 | Count the number of bits in the object that are true |

(3) Function

- Calculate the total number of bits in the count object "In" where the value is true.

(4) Sample program

| DEMO | LD | ST |
|---|---|---|
| Define variables | | `VAR`<br>`BCin: WORD;`<br>`BCout: USINT;`<br>`RETURN0: BOOL;`<br>`END_VAR` |
| Sample program |  | `BitCnt(In:=BCin ,`<br>`Out=> BCout);` |

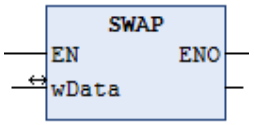
| | |
|---|---|
| Operation result | |

1-4-4-13. High low byte exchange SWAP

(1) Instruction overview

Swap the high and low 8 bits of 16-bit type data.

| High low byte exchange [SWAP] | | | |
|---|---|---|---|
| Execution condition | Edge triggering | Suitable model | XS, XSA |
| Firmware | - | Software | XS StudioV2.2.0 and above |
| Library | XJ_DataUtils | | |

| Instruction | Name | Graph | ST |
|---|---|---|---|
| SWAP | High low byte exchange |  | SWAP(wData:= ); |

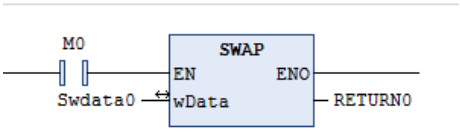
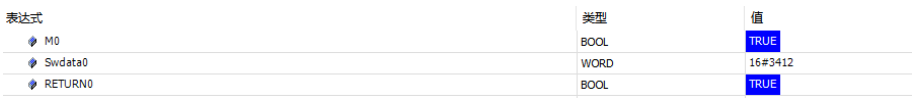
(2) Related variables

| VAR_IN_OUT | Name | Data type | Effective range | Initial value | Description |
|---|---|---|---|---|---|
| wData | Data that needs to be operated | Word | As the data type | - | Data that needs to be operated |

(3) Function

- Swap the high and low 8 bits of 16-bit type data.

(4) Sample program

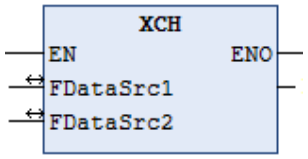
| DEMO | LD | ST |
|---|---|---|
| Define variables | | VAR<br>M0:BOOL;<br>Swdata0: WORD;<br>RETURN0: BOOL;<br>END_VAR |
| Sample program |  | IF M0 THEN<br>SWAP(wData:=Swdata0 );<br>END_IF |
| Operation result |  | |
| Result analysis | M0 is set to True, and the high and low bits of Swdata0 will be converted from 16#1234 to 16#3412; M0 is constantly triggered, and the high and low positions will continue to transition. | |

1-4-4-14. Data exchange XCH

(1) Instruction overview

Swap two floating-point data.

| Data exchange [XCH] | | | |
|---|---|---|---|
| Execution condition | Edge triggering | Suitable model | XS, XSA |
| Firmware | - | Software | XS StudioV2.2.0 and above |
| Library | XJ_DataUtils | | |

| Instruction | Name | Graph | ST |
|---|---|---|---|
| XCH | Data exchange |  | XCH(FDataSrc1:= , FDataSrc2:= ); |

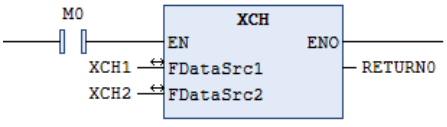
(2) Related variables

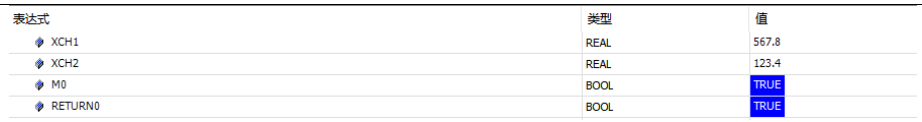
| VAR_IN_OUT | Name | Data type | Effective range | Initial value | Description |
|---|---|---|---|---|---|
| FDataSrc1 | Data1 to be exchanged | REAL | As the data type | 0 | Data to be exchanged |
| FDataSrc1 | Data2 to be exchanged | REAL | As the data type | 0 | Data to be exchanged |

(3) Function

- Swap two floating-point data.

(4) Sample program

| DEMO | LD | ST |
|---|---|---|
| Define variables | | VAR<br>XCH1: REAL;<br>XCH2: REAL;<br>M0:BOOL;<br>RETURN0: BOOL;<br>END_VAR |
| Sample program |  | IF M0 THEN<br>XCH(FDataSrc1:= XCH1,<br>FDataSrc2:=XCH2 );<br>END_IF |
| Operation result | Before operation<br> <br>After operation | |

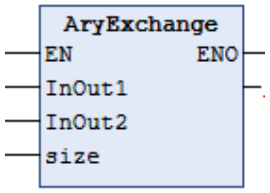
| | |
|---|---|
| |  |
| Result analysis | Set M0 to True, the data of XCH1 and XCH2 will be exchanged, with data 123.4 in XCH1 being exchanged for data 567.8 in XCH2; M0 is constantly triggered, and XCH1 and XCH2 will continue to exchange. |

1-4-4-15. Array data exchange AryExchange

(1) Instruction overview

Swap the elements of two arrays.

| Array data exchange [AryExchange] | | | |
|---|---|---|---|
| Execution condition | Edge triggering | Suitable model | XS, XSA |
| Firmware | - | Software | XS StudioV2.2.0 and above |
| Library | XJ_DataUtils | | |

| Instruction | Name | Graph | ST |
|---|---|---|---|
| AryExchange | Array data exchange |  | AryExchange(InOut1:= ,<br>InOut2:= ,<br>size:= ); |

(2) Related variables

| VAR_INPUT | Name | Data type | Effective range | Initial value | Description |
|---|---|---|---|---|---|
| In | The array to be transferred | BOOL/BYTE/WORD/ DWORD/LWORD/ USINT/UINT/UDINT/ ULINT/SINT/INT/DINT /LINT/REAL/LREAL TIME/DATE/TOD/DT/ STRING | As the data type | - | The array to be transferred |
| VAR_IN_OUT | Name | Data type | Effective range | Initial value | Description |
| AryOut | Transfer result array | BOOL/BYTE/WORD/ DWORD/LWORD/ USINT/UINT/UDINT/ ULINT/SINT/INT/DINT /LINT/REAL/LREAL TIME/DATE/TOD/DT/ STRING | As the data type | - | Transfer result array |
| Size | Number of elements to be transmitted | UINT | As the data type | 0 | Number of elements to be transmitted |

(3) Function

- Exchange the values of "InOut1" and "InOut2".
- Swap the "Size" elements starting from InOut1[0] in the swappable object array InOut1[] with the "Size" elements starting from InOut2[0] in InOut2[].

(4) Note

- The data types of InOut1[] and InOut2[] elements should be consistent, otherwise the function return value is FALSE, and InOut1[] and InOut2[] remain unchanged.
- When the value of "Size" is 0, the function returns a value of True, and InOut1[] and InOut2[] remain unchanged.
- The InOut1[] and InOut2[] elements are STRING type, and the string length of one should not be greater than the memory size of the other to avoid PLC exceptions.
- The value of "Size" should not exceed the array area of InOut1[] or InOut2[] to avoid PLC exceptions.

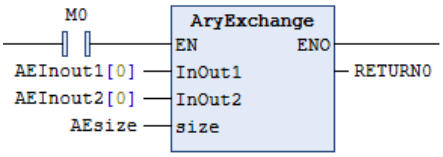
(5) Sample program

<table>
<tr><th>DEMO</th><th>LD</th><th>ST</th></tr>
<tr><td>Define variables</td><td colspan="2">

```
VAR

    AEInout1:ARRAY[0..5] OF BYTE;
    AEInout2:ARRAY[0..5] OF BYTE;
    AEsize: UINT;
    M0:BOOL;
    RETURN0: BOOL;

END_VAR
```

</td></tr>
<tr><td>Sample program</td><td>


</td><td>

```
IF M0 THEN
    AryExchange(InOut1:= AEInout1[0],
        InOut2:= AEInout2[0],
        size:=AEsize );
END_IF
```

</td></tr>
<tr><td>Operation result</td><td colspan="2">
Before operation

| 表达式 | 类型 | 值 |
|---|---|---|
| AEInout1 | ARRAY [0..5] OF BYTE | |
| AEInout1[0] | BYTE | 11 |
| AEInout1[1] | BYTE | 22 |
| AEInout1[2] | BYTE | 33 |
| AEInout1[3] | BYTE | 44 |
| AEInout1[4] | BYTE | 55 |
| AEInout1[5] | BYTE | 66 |
| AEInout2 | ARRAY [0..5] OF BYTE | |
| AEInout2[0] | BYTE | 77 |
| AEInout2[1] | BYTE | 88 |
| AEInout2[2] | BYTE | 99 |
| AEInout2[3] | BYTE | 123 |
| AEInout2[4] | BYTE | 234 |
| AEInout2[5] | BYTE | 245 |
| AEsize | UINT | 5 |
| M0 | BOOL | FALSE |

After operation

| 表达式 | 类型 | 值 |
|---|---|---|
| AEInout1 | ARRAY [0..5] OF BYTE | |
| AEInout1[0] | BYTE | 77 |
| AEInout1[1] | BYTE | 88 |
| AEInout1[2] | BYTE | 99 |
| AEInout1[3] | BYTE | 123 |
| AEInout1[4] | BYTE | 234 |
| AEInout1[5] | BYTE | 66 |
| AEInout2 | ARRAY [0..5] OF BYTE | |
| AEInout2[0] | BYTE | 11 |
| AEInout2[1] | BYTE | 22 |
| AEInout2[2] | BYTE | 33 |
| AEInout2[3] | BYTE | 44 |
| AEInout2[4] | BYTE | 55 |
| AEInout2[5] | BYTE | 245 |
| AEsize | UINT | 5 |
| M0 | BOOL | TRUE |
| RETURN0 | BOOL | TRUE |

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

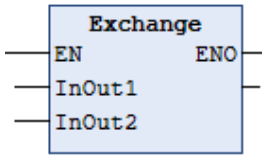
| | |
|---|---|
| Result analysis | Set M0 to True, and the AEsize data of AEInOut1 and AEinout2 will be exchanged; M0 is constantly triggered, and XCH1 and XCH2 will continue to exchange. |

1-4-4-16. Data exchange Exchange

(1) Instruction overview

Swap the values of two variables.

| Data exchange [Exchange] | | | |
|---|---|---|---|
| Execution condition | Edge triggering | Suitable model | XS, XSA |
| Firmware | - | Software | XS StudioV2.2.0 and above |
| Library | XJ_DataUtils | | |

| Instruction | Name | Graph | ST |
|---|---|---|---|
| Exchange | Data exchange |  | Exchange(InOut1:= , InOut2:= ); |

(2) Related variables

| VAR_IN_OUT | Name | Data type | Effective range | Initial value | Description |
|---|---|---|---|---|---|
| InOut1 | Exchange Object 1 | BOOL/BYTE/WORD/ DWORD/LWORD/USINT/UINT/ UDINT/ULINT/SINT/INT/DINT /LINT/REAL/LREAL/TIME/DATE/ TOD/DT/STRING | As the data type | 0 | Exchange Object 1 |
| InOut2 | Exchange Object 2 | BOOL/BYTE/WORD/DWORD/ LWORD/USINT/UINT/UDINT/ ULINT/SINT/INT/DINT/LINT/REAL/ LREAL/TIME/DATE/TOD/DT/STRING | As the data type | 0 | Exchange Object 2 |

(3) Function

- Exchange the values of "InOut1" and "InOut2".

(4) Note

- The data types of "InOut1" and "InOut2" should be consistent, otherwise the runtime function returns a value of False.
- When using this instruction in the ST program, the return value "Out" is not used.
- "InOut1" and "InOut2" are STRING types. If the length of any string does not match the size of the other party, it will not be swapped.

(5) Sample program

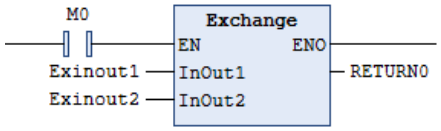
<table>
<tr><th>DEMO</th><th>LD</th><th>ST</th></tr>
<tr><td>Define variables</td><td colspan="2">

```
VAR

    Exinout1: WORD;
    Exinout2: WORD;
    M0:BOOL;
    RETURN0: BOOL;

END_VAR
```

</td></tr>
<tr><td>Sample program</td><td>




</td><td>

```
IF M0 THEN
    Exchange(InOut1:=Exinout1,
        InOut2:=Exinout2 );
END_IF
```

</td></tr>
<tr><td>Operation result</td><td colspan="2">
Before operation

| 表达式 | 类型 | 值 |
|---|---|---|
| Exinout1 | WORD | 123 |
| Exinout2 | WORD | 456 |
| M0 | BOOL | FALSE |
| RETURN0 | BOOL | FALSE |

After operation

| 表达式 | 类型 | 值 |
|---|---|---|
| Exinout1 | WORD | 456 |
| Exinout2 | WORD | 123 |
| M0 | BOOL | TRUE |
| RETURN0 | BOOL | TRUE |

</td></tr>
<tr><td>Result analysis</td><td colspan="2">Set M0 to True, and the data of ExInOut1 and Exinout2 will be exchanged; M0 is constantly triggered, and ExInOut1 and Exinout2 will always switch.</td></tr>
</table>

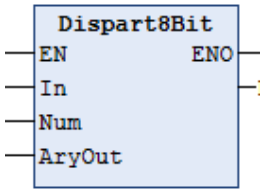
## 1-4-5. Data separation and combination instructions

### 1-4-5-1. Byte unit data separation Dispart8Bit

(1) Instruction overview

Separate bit strings in units of 1 byte.

| Byte unit data separation [Dispart8Bit] | | | |
|---|---|---|---|
| Execution condition | Edge triggering | Suitable model | XS, XSA |
| Firmware | - | Software | XS StudioV2.2.0 and above |
| Library | XJ_DataUtils | | |

| Instruction | Name | Graph | ST |
|---|---|---|---|
| Dispart8Bit | Byte unit data separation |  | bResult:=Dispart8Bit(In:= ,<br>Num:= ,<br>AryOut:= ); |

(2) Related variables

| VAR_INPUT | Name | Data type | Effective range | Initial value | Description |
|---|---|---|---|---|---|
| In | Separate objects | BYTE/WORD/ DWORD/LWORD | As the data type | - | Bit string of separate object |
| Num | Separate bytes | USINT | As the data type | 1 | Number of bytes to be separated |
| VAR_IN_OUT | Name | Data type | Effective range | Initial value | Description |
| AryOut[] array | Separate result array | BYTE | As the data type | - | Separate result array |

Note: The input and output variable AryOut requires specifying array elements.

(3) Function

- Separate the separated object "In" in units of 1 byte and save it in the separation result array AryOut[ ].
- Firstly, separate "In" in units of 1 byte. Then, save the lowest byte in the starting address of AryOut[] set on pin AryOut. Save the next byte in the AryOut[]+1 address. Perform the same processing on the number of separated bytes "Num".

(4) Note

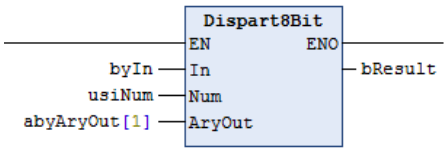
- When the value of "Num" exceeds the valid range, the return value of the function is FALSE, and AryOut[] remains unchanged.
- The value of "Num" should not exceed the array area of AryOut[].

(5) Sample program

<table>
<tr><th>DEMO</th><th>LD</th><th>ST</th></tr>
<tr><td>Define variables</td><td colspan="2">

```
VAR
    byIn      :LWORD:=16#FFFF00FF;
    usiNum    :USINT:=4;
    abyAryOut:ARRAY[0..4] OF BYTE:=[0,0,0,0,0];
    bResult   :BOOL;
END_VAR
```

</td></tr>
<tr><td>Sample program</td><td>




</td><td>

```
1  bResult:=Dispart8Bit(In:= byIn,
2      Num:= usiNum,
3      AryOut:= abyAryOut[1]);
```

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

| 表达式 | 类型 | 值 |
|---|---|---|
| byIn | LWORD | 16#00000000FFFF00FF |
| usiNum | USINT | 16#04 |
| abyAryOut | ARRAY [0..4] OF BYTE | |
| abyAryOut[0] | BYTE | 16#00 |
| abyAryOut[1] | BYTE | 16#FF |
| abyAryOut[2] | BYTE | 16#00 |
| abyAryOut[3] | BYTE | 16#FF |
| abyAryOut[4] | BYTE | 16#FF |
| bResult | BOOL | TRUE |

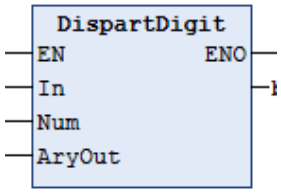
</td></tr>
</table>

1-4-5-2. 4-bit separate DispartDigit

(1) Instruction overview

Separate bit strings in units of 4 bits.

| 4-bit separate [DispartDigit] | | | |
|---|---|---|---|
| Execution condition | Edge triggering | Suitable model | XS, XSA |
| Firmware | - | Software | XS StudioV2.2.0 and above |
| Library | XJ_DataUtils | | |

| Instruction | Name | Graph | ST |
|---|---|---|---|
| DispartDigit | 4-bit separate |  | bResult:=DispartDigit(In:= ,<br>Num:= ,<br>AryOut:= ); |

(2) Related variables

| VAR_INPUT | Name | Data type | Effective range | Initial value | Description |
|---|---|---|---|---|---|
| In | Separate objects | BYTE/WORD/ DWORD/LWORD | As the data type | - | Bit strings of separate object |
| Num | Separate bits | USINT | As the data type | 1 | Number of digits to be separated |
| VAR_IN_OUT | Name | Data type | Effective range | Initial value | Description |
| AryOut[] array | Separate result array | BYTE | As the data type | - | Separate result array |

Note: The input and output variable AryOut requires specifying array elements.

(3) Function

- Separate the "In" of the separated object in units of 4 digits (digital units) and save it in the separation result array AryOut[].
- Firstly, separate "In" in units of 4 digits. Then, save the lowest 4 bits in the starting address of the AryOut[] set on pin AryOut. At this point, add 16#0 in bits 4 to 7 of the BYTE type AryOut[].
- Perform the same treatment on the number of separation digits "Num".

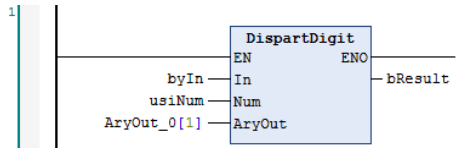
(4) Note

- When the value of "Num" is 0, the function returns a value of True, and the value of AryOut[] remains unchanged.
- When the value of "Num" exceeds the valid range, the return value of the function is FALSE, and AryOut[] remains unchanged.
- The value of "Num" should not exceed the array area of AryOut[].

(5) Sample program

<table>
<tr><th>DEMO</th><th>LD</th><th>ST</th></tr>
<tr><td>Define variables</td><td colspan="2">

```
VAR
    byIn:LWORD:=16#FF00FF;
    usiNum:USINT:=3;
    AryOut_0:ARRAY[0..5] OF BYTE:=[0,0,0,0,0,0];
    bResult:BOOL;
END_VAR
```

</td></tr>
<tr><td>Sample program</td><td> 
</td><td>

```
1  bResult:=DispartDigit(In:= byIn,
2      Num:= usiNum,
3      AryOut:= AryOut_0[1]);
```

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

| 表达式 | 类型 | 值 |
|---|---|---|
| byIn | LWORD | 16#0000000000FF00FF |
| usiNum | USINT | 16#03 |
| AryOut_0 | ARRAY [0..5] OF BYTE | |
| AryOut_0[0] | BYTE | 16#00 |
| AryOut_0[1] | BYTE | 16#0F |
| AryOut_0[2] | BYTE | 16#0F |
| AryOut_0[3] | BYTE | 16#00 |
| AryOut_0[4] | BYTE | 16#00 |
| AryOut_0[5] | BYTE | 16#00 |
| bResult | BOOL | TRUE |

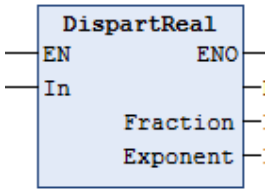
</td></tr>
</table>

1-4-5-3. Mantissa of real numbers, exponential separation DispartReal

(1) Instruction overview

Decompose real numbers into signed mantissa parts and exponential parts.

| Mantissa of real numbers, exponential separation [DispartReal] | | | |
|---|---|---|---|
| Execution condition | Edge triggering | Suitable model | XS, XSA |
| Firmware | - | Software | XS StudioV2.2.0 and above |
| Library | XJ_DataUtils | | |

| Instruction | Name | Graph | ST |
|---|---|---|---|
| DispartReal | Mantissa of real numbers, exponential separation |  | bResult:=DispartReal(In:= ,<br>Fraction=> ,<br>Exponent=> ); |

(2) Related variables

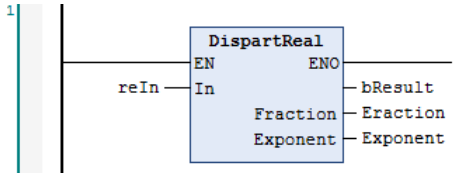
| VAR_INPUT | Name | Data type | Effective range | Initial value | Description |
|---|---|---|---|---|---|
| In | Real number | REAL/LREAL | As the data type | - | Real number to be separated |
| VAR_OUTPUT | Name | Data type | Effective range | Initial value | Description |
| Fraction | Signed mantissa part | LINT | As the data type | 0 | Signed mantissa part |
| Exponent | Exponential part | INT | As the data type | 0 | Exponential part |

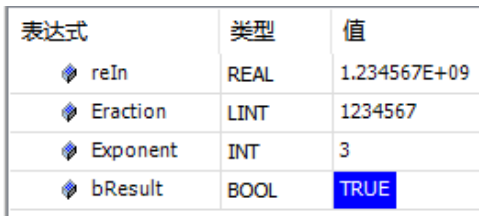
(3) Function

- Decompose the real number "In" into a signed mantissa part "Fraction" and an exponential part "Exponent".
- When "In" is a REAL type, "Fraction" is a 7-bit integer. When "In" is LREAL type, "Fraction" is a 15 bits integer. The effective range of "Fraction" varies depending on the data type of "In".

| “In” data type | “Fraction” effective range |
|---|---|
| REAL | -9999999～9999999 |
| LREAL | -999999999999999～999999999999999 |

(4) Sample program

| DEMO | LD | ST |
|---|---|---|
| Define variables | | VAR<br>reIn:REAL:=12.34567E+8;<br>Eraction:LINT;<br>Exponent:INT;<br>bResult:BOOL;<br>END_VAR |
| Sample program |  | 1 bResult:=DispartReal(In:= reIn,<br>2 Fraction=> Eraction,<br>3 Exponent=> Exponent); |

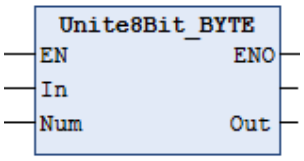
| | |
|---|---|
| Operation result |  |

1-4-5-4. Byte unit data combination group Unite8Bit_XXX

(1) Instruction overview

Combine data in units of 1 byte into a bit string.

| Byte unit data combination group [Unite8Bit_XXX] | | | |
|---|---|---|---|
| Execution condition | Edge triggering | Suitable model | XS, XSA |
| Firmware | - | Software | XS StudioV2.2.0 and above |
| Library | XJ_DataUtils | | |

| Instruction | Name | Graph | ST |
|---|---|---|---|
| Unite8Bit_XXX | Byte unit data combination group |  | bResult:=Unite8Bit_BYTE(In:= ,<br>Num:= ,<br>Out=> ); |

(2) Related variables

| VAR_INPUT | Name | Data type | Effective range | Initial value | Description |
|---|---|---|---|---|---|
| In[] array | Combining Object Array | BYTE | As the data type | - | Array to be combined |
| Num | Combining bytes | USINT | As the data type | 1 | Bytes to be combined |
| VAR_OUTPUT | Name | Data type | Effective range | Initial value | Description |
| Out | Combining result | BYTE/WORD/ DWORD/LWORD | As the data type | - | Bit string of combined results |

Note: The input variable In needs to pass in the specified element of the array.

(3) Function

- Combine the elements of the combination object array In[] to generate the bit string "Out" of the combination result.
- The number of array elements to be combined is specified by the combined byte count "Num". Firstly, combine In[] with consecutive "Num" bytes to generate a bit string of "Num" bytes. On its high bit, take the 16#00 bit string of the combination {(bytes of "Out") - "Num"} as "Out".
- The instruction name varies depending on the data type of "Out". For example, when "Out" is DWORD type, the instruction name is Unite8Bit_DWORD.

(4) Note

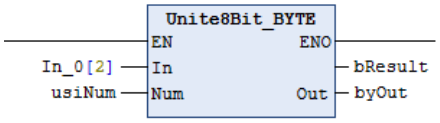
- When the value of "Num" is 0, the value of "Out" is initialized to 0, and the function returns a value of True.
- When the value of "Num" exceeds the valid range, the function returns a value of FALSE and initializes the value of "Out" to 0.
- The value of "Num" cannot exceed the array range of In[] and cannot exceed the range of "Out".
- The input needs to pass in the specified elements of the array, and the array length cannot be obtained. Otherwise, the function returns False.

(5) Sample program

<table>
<tr><th>DEMO</th><th>LD</th><th>ST</th></tr>
<tr><td>Define variables</td><td colspan="2">

```
VAR
    In_0     :ARRAY[1..10] OF BYTE:=[10(1)];
    usiNum   :USINT:=1;
    byOut    :BYTE;
    bResult  :BOOL;
END_VAR
```

</td></tr>
<tr><td>Sample program</td><td>




</td><td>

```
bResult:=Unite8Bit_BYTE(In:= In_0[2],
    Num:= usiNum,
    Out=> byOut);
```

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

| 表达式 | 类型 | 值 |
|---|---|---|
| In_0 | ARRAY [1..10] OF BYTE | |
| In_0[1] | BYTE | 16#01 |
| In_0[2] | BYTE | 16#01 |
| In_0[3] | BYTE | 16#01 |
| In_0[4] | BYTE | 16#01 |
| In_0[5] | BYTE | 16#01 |
| In_0[6] | BYTE | 16#01 |
| In_0[7] | BYTE | 16#01 |
| In_0[8] | BYTE | 16#01 |
| In_0[9] | BYTE | 16#01 |
| In_0[10] | BYTE | 16#01 |
| usiNum | USINT | 16#01 |
| byOut | BYTE | 16#01 |
| bResult | BOOL | TRUE |

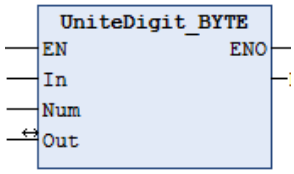
</td></tr>
</table>

1-4-5-5. 4-digit combination group UniteDigit_XXX

(1) Instruction overview

Combine 4-bit data into a bit string.

| 4-digit combination group [UniteDigit_XXX] | | | |
|---|---|---|---|
| Execution condition | Edge triggering | Suitable model | XS, XSA |
| Firmware | - | Software | XS StudioV2.2.0 and above |
| Library | XJ_DataUtils | | |

| Instruction | Name | Graph | ST |
|---|---|---|---|
| UniteDigit_XXX | 4-digit combination group |  | bResult:=UniteDigit_BYTE(In:= ,<br>Num:= ,<br>Out:= ); |

(2) Related variables

| VAR_INPUT | Name | Data type | Effective range | Initial value | Description |
|---|---|---|---|---|---|
| In[] array | Combining Object Array | BYTE | As the data type | - | Array to be combined |
| Num | Combining bits | USINT | As the data type | 1 | Bits to be combined |
| VAR_OUTPUT | Name | Data type | Effective range | Initial value | Description |
| Out | Combining result | BYTE/WORD/ DWORD/LWORD | As the data type | - | Bit string of combined results |

(3) Function

- Combine the elements of the combination object array In[] by the low 4 bits (1 digit=4 bits) to generate the bit string "Out" of the combination result.
- The number of array elements to be combined is specified by the combination digit "Num". Firstly, combine each low 4 bits of the consecutive "Num" starting from In[] to generate a bit string of "Num" bits. At its high position, take the 16#0 bit string of the combination {(number of "Out" bits) - "Num"} as "Out".
- The instruction name varies depending on the data type of "Out". For example, when "Out" is WORD type, the instruction name is UniteDigit_WORD.

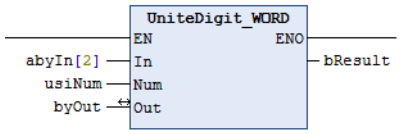
(4) Note

- When the value of "Num" is 0, the value of "Out" is 0, and the function returns a value of True.
- When the value of "Num" exceeds the valid range, the function returns a value of FALSE and "Out" remains unchanged.
- The value of "Num" cannot exceed the array range of In[] and cannot exceed the range of "Out".
- Input In requires passing in the specified elements of the array, and the array length cannot be obtained. Otherwise, the function returns False.

(5) Sample program

<table>
<tr><th>DEMO</th><th>LD</th><th>ST</th></tr>
<tr><td>Define variables</td><td colspan="2">

```
VAR
    abyIn:ARRAY[1..10] OF BYTE:=[10(1)];
    usiNum:USINT:=2;
    byOut:WORD;
    bResult:BOOL;
END_VAR
```

</td></tr>
<tr><td>Sample program</td><td> 
</td><td>

```
1  bResult:=UniteDigit_WORD(In:= abyIn[2],
2      Num:= usiNum,
3      Out:= byOut);
```

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

| 表达式 | 类型 | 值 |
|---|---|---|
| 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 | WORD | 16#0011 |
| bResult | BOOL | TRUE |

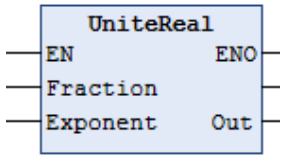
</td></tr>
</table>

1-4-5-6. Combine mantissa and exponents into real numbers UniteReal

(1) Instruction overview

Combine the signed mantissa and exponential parts to obtain real numbers.

| Combine mantissa and exponents into real numbers [UniteReal] | | | |
|---|---|---|---|
| Execution condition | Edge triggering | Suitable model | XS, XSA |
| Firmware | - | Software | XS StudioV2.2.0 and above |
| Library | XJ_DataUtils | | |

| Instruction | Name | Graph | ST |
|---|---|---|---|
| UniteReal | Combine mantissa and exponents into real numbers |  | bResult:=UniteReal(Fraction:= ,<br>Exponent:= ,<br>Out=> ); |

(2) Related variables

| VAR_INPUT | Name | Data type | Effective range | Initial value | Description |
|---|---|---|---|---|---|
| Fraction | Signed mantissa part | DINT/LINT | As the data type | - | Signed mantissa part |
| Exponent | Exponential part | INT | As the data type | 0 | Exponential part |
| VAR_OUTPUT | Name | Data type | Effective range | Initial value | Description |
| Out | Real number | LREAL | As the data type | 0 | Real numbers obtained by combining |

Note: When the data type of "Fraction" is DINT, the data type of "Out" is REAL; When the data type of "Fraction" is LINT, the data type of "Out" is LREAL.

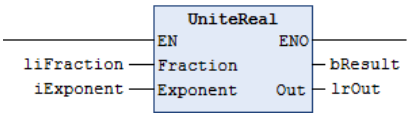
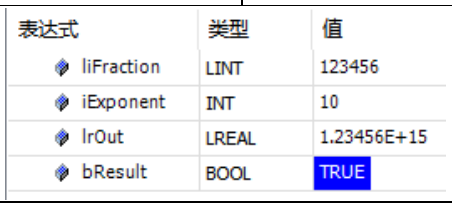
(3) Function

- Combine the signed mantissa part "Fraction" and the exponential part "Exponent" to obtain the real number "Out".

(4) Note

- Due to the different values of "Fraction" and "Exponent", there may be errors when converting integers to real numbers.
- When the combination result exceeds the valid range of "Out", if "Exponent" is a positive number, then the value of "Out" is infinite with the same sign as "Fraction". If "Exponent" is negative, then the value of "Out" is 0.
- "Fraction" requires the use of Dint and Lint types. When the type is determined to be outside the integer type, a compilation error is reported. Otherwise, the runtime function returns a value of FALSE, and the value of "Out" is 0.

(5) Sample program

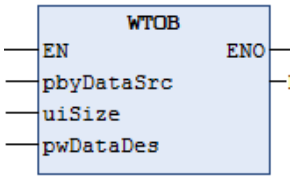
| DEMO | LD | ST |
|---|---|---|
| Define variables | | VAR<br>liFraction:LINT:=123456;<br>iExponent :INT:=10;<br>lrOut :LREAL;<br>bResult :BOOL;<br>END_VAR |
| Sample program |  | bResult:=UniteReal(Fraction:= liFraction,<br>Exponent:= iExponent,<br>Out=> lrOut); |
| Operation result |  | |

1-4-5-7. Byte unit data separation WTOB

(1) Instruction overview

Separate continuous 16 bits data in byte units.

| Byte unit data separation [WTOB] | | | |
|---|---|---|---|
| Execution condition | Edge triggering | Suitable model | XS, XSA |
| Firmware | - | Software | XS StudioV2.2.0 and above |
| Library | XJ_DataUtils | | |

| Instruction | Name | Graph | ST |
|---|---|---|---|
| WTOB | Byte unit data separation |  | bResult:=WTOB(pbyDataSrc:= ,<br>uiSize:= ,<br>pwDataDes:= ); |

(2) Related variable

| VAR_INPUT | Name | Data type | Effective range | Initial value | Description |
|---|---|---|---|---|---|
| pbyDataSrc | Starting address for separating data | POINTER_TO_BYTE | As the data type | - | Starting address for separating data |
| uiSize | Number of separated data | UINT | As the data type | 0 | Number of separated data |

| | | | | | |
|---|---|---|---|---|---|
| pwDataDes | The final storage destination address | POINTER_TO_WORD | As the data type | - | The final storage destination address of separating data |

(3) Function description

- Save the 16 bits data starting from the pbyDataSrc address, in bytes, to the low 8 bits of the uiSize*2 starting from the pwDataDes address, and the high 8 bits to 16#00.

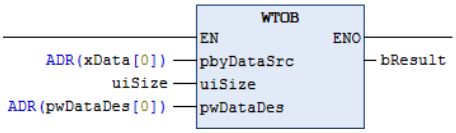
(4) Note

- When the SIZE value is 0, the function returns a value of True.

(5) Program example

| DEMO | LD | ST |
|---|---|---|
| Define variables | | |
| Sample program | | |
| Operation result | | |

```
VAR
    bResult:BOOL;
    xData:ARRAY[0..8] OF BYTE:=[9(16#11)];
    uiSize:UINT:=3;
    pwDataDes:ARRAY[0..10] OF WORD;
END_VAR
```




```
1  bResult:=WTOB(pbyDataSrc:= ADR(xData[0]),
2      uiSize:= uiSize,
3      pwDataDes:= ADR(pwDataDes[0]));
```

| 表达式 | 类型 | 值 |
|---|---|---|
| bResult | BOOL | TRUE |
| xData | ARRAY [0..8] OF BYTE | |
| xData[0] | BYTE | 16#11 |
| xData[1] | BYTE | 16#11 |
| xData[2] | BYTE | 16#11 |
| xData[3] | BYTE | 16#11 |
| xData[4] | BYTE | 16#11 |
| xData[5] | BYTE | 16#11 |
| xData[6] | BYTE | 16#11 |
| xData[7] | BYTE | 16#11 |
| xData[8] | BYTE | 16#11 |
| uiSize | UINT | 16#0003 |
| pwDataDes | ARRAY [0..10] OF WORD | |
| pwDataDes[0] | WORD | 16#0011 |
| pwDataDes[1] | WORD | 16#0011 |
| pwDataDes[2] | WORD | 16#0011 |
| pwDataDes[3] | WORD | 16#0011 |
| pwDataDes[4] | WORD | 16#0011 |
| pwDataDes[5] | WORD | 16#0011 |
| pwDataDes[6] | WORD | 16#0000 |
| pwDataDes[7] | WORD | 16#0000 |
| pwDataDes[8] | WORD | 16#0000 |
| pwDataDes[9] | WORD | 16#0000 |
| pwDataDes[10] | WORD | 16#0000 |

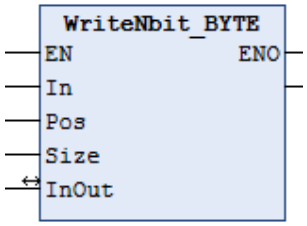
## 1-4-6. Data read and write instructions

### 1-4-6-1. N bits write to group WriteNbit_XXX

(1) Instruction overview

Write multiple bits into a bit train.

| N bits write to group [WriteNbit_XXX] | | | |
|---|---|---|---|
| Execution condition | Edge triggering | Suitable model | XS, XSA |
| Firmware | - | Software | XS StudioV2.2.0 and above |
| Library | XJ_DataUtils | | |

| Instruction | Name | Graph | ST |
|---|---|---|---|
| WriteNbit_XXX | N bits write to group |  | WriteNbit_BYTE(In:= ,<br>Pos:= ,<br>Size:= ,<br>InOut:= ); |

(2) Related variables

| VAR_INPUT | Name | Data type | Effective range | Initial value | Description |
|---|---|---|---|---|---|
| In | Read source | BYTE/WORD/ DWORD/LWORD | As the data type | - | This bit train is used to read the value to be written to "InOut" |
| Pos | Write in position | USINT | As the data type | 0 | The position of the bit to be written |
| Size | Write in size | USINT | As the data type | 0 | The number of bits to be written |
| VAR_IN_OUT | Name | Data type | Effective range | Initial value | Description |
| InOut | Write in object | Same as "In" type | As the data type | - | Write in the result |

(3) Function

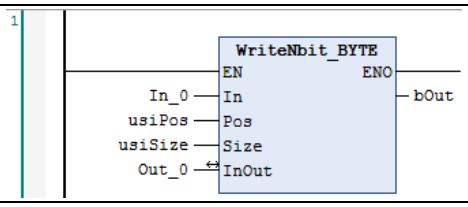
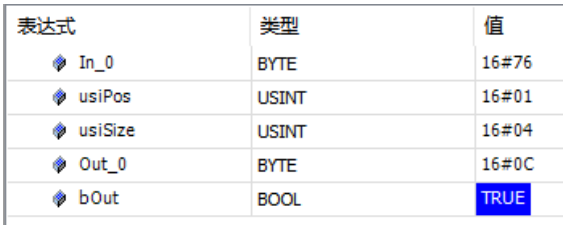
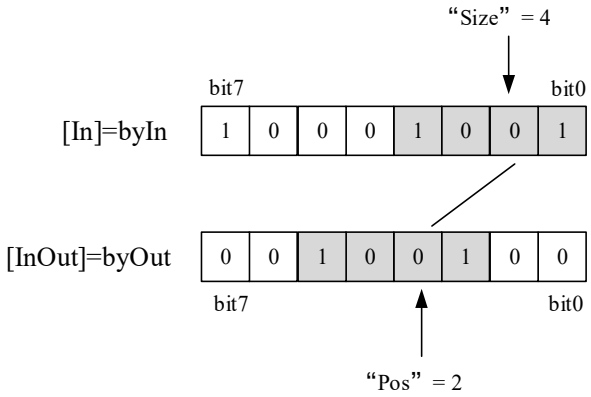
- Firstly, read out the low bit "Size" of the source "In". Then, write the read value to the write position "Pos" of the write object "InOut".
- The instruction name varies depending on the data type of "In" and "InOut". For example, when "In" and "InOut" are BYTE types, the instruction name type is WriteNbit_BYTE.

(4) Note

- Please set the data types of "In" and "InOut" to be the same.
- When the value of "Size" is 0, the value of "InOut" remains unchanged, and the function returns a value of True.
- When using this instruction in the ST program, the return value "Out" is not used.
- When "Size" and "Pos" exceed the valid range, the program encounters an exception, and the function returns a value of FALSE, while the value of "InOut" remains unchanged.

- When the number of bits in "InOut" is insufficient from the position shown in "Pos" to the position shown in "Size", the program encounters an exception, and the function returns a value of FALSE, while the value of "InOut" remains unchanged.

(5) Program example

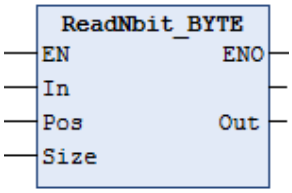
| DEMO | LD | ST |
|---|---|---|
| Define variables | VAR<br>In_0 :BYTE:=16#76;<br>usiPos :USINT:=1;<br>usiSize:USINT:=4;<br>Out_0 :BYTE;<br>bOut :BOOL;<br>END_VAR | |
| Sample program |  | 1 WriteNbit_BYTE(In:= In_0,<br>2 Pos:= usiPos,<br>3 Size:= usiSize,<br>4 InOut:= Out_0); |
| Operation result |  | |
| Principle analysis |  | |

1-4-6-2. N bits read group ReadNbit_XXX

(1) Instruction overview

Read multiple bits in the bit train.

| N bits read group [ReadNbit_XXX] | | | |
|---|---|---|---|
| Execution condition | Edge triggering | Suitable model | XS, XSA |
| Firmware | - | Software | XS StudioV2.2.0 and above |
| Library | XJ_DataUtils | | |

| Instruction | Name | Graph | ST |
|---|---|---|---|
| ReadNbit_XXX | N bits read group |  | ReadNbit_BYTE(In:= ,<br>Pos:= ,<br>Size:= ,<br>Out=> ); |

(2) Related variables

| VAR_INPUT | Name | Data type | Effective range | Initial value | Description |
|---|---|---|---|---|---|
| In | Read source | BYTE/WORD/ DWORD/LWORD | As the data type | - | This bit train is used to read the value to be written to "InOut" |
| Pos | Read position | USINT | As the data type | 0 | To read the position of bit |
| Size | Read size | USINT | As the data type | 0 | The number of bits to be read |
| VAR_OUTPUT | Name | Data type | Effective range | Initial value | Description |
| Out | Read result | Same as "In" type | As the data type | - | Read result |

(3) Function

- Substitute the "Size" bits starting from the reading position "Pos" of the source bit column "In" into the reading result "Out".
- The instruction name varies depending on the data type of "In" and "Out". For example, when "In" and "Out" are BYTE types, the instruction name is ReadNbit_BYTE.

(4) Note

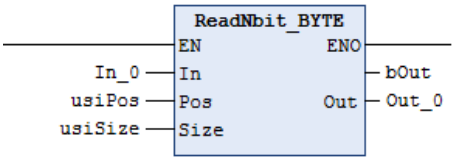
- It is recommended to unify the data types for "In" and "Out".
- When the value of "Size" is 0, the value of "Out" is 16#0.
- In the following situations, an exception will occur, with the function returning a value of FALSE and the value of "Out" being 16#0.
  - When the value of "Pos" exceeds the valid range.
  - When the value of "Size" exceeds the valid range.
  - When the specified values of "Pos" and "Size" exceed the number of digits of "In".

(5) Program example

<table>
<tr><th>DEMO</th><th>LD</th><th>ST</th></tr>
<tr><td>Define variables</td><td colspan="2">

```
VAR
    In_0   :BYTE:=16#76;
    usiPos :USINT:=1;
    usiSize:USINT:=4;
    Out_0  :BYTE;
    bOut   :BOOL;
END_VAR
```

</td></tr>
<tr><td>Sample program</td><td>



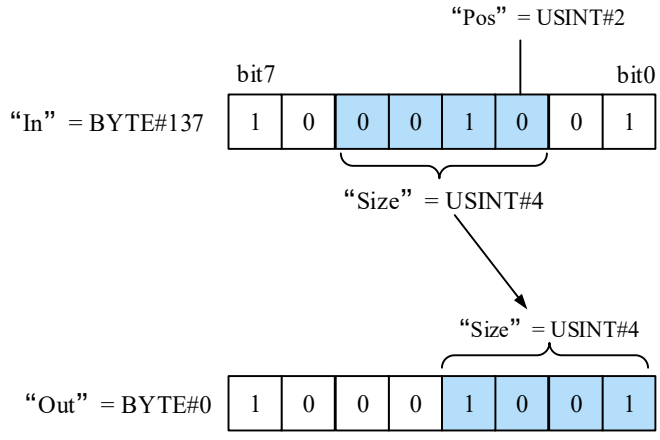

</td><td>

```
1  ReadNbit_BYTE(In:= In_0,
2      Pos:= usiPos,
3      Size:= usiSize,
4      Out=> Out_0);
```

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

| 表达式 | 类型 | 值 |
|---|---|---|
| In_0 | BYTE | 16#76 |
| usiPos | USINT | 16#01 |
| usiSize | USINT | 16#04 |
| Out_0 | BYTE | 16#0B |
| bOut | BOOL | TRUE |

</td></tr>
<tr><td>Principle analysis</td><td colspan="2">




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

## 1-4-7. Data conversion instructions

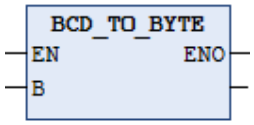
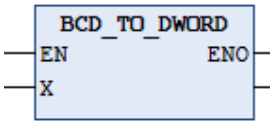
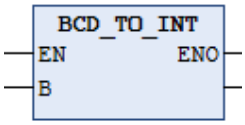
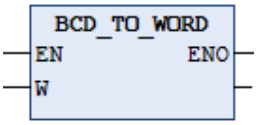
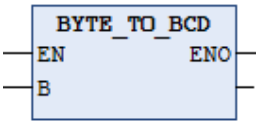
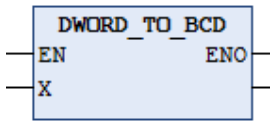
- It is strictly prohibited to implicitly convert "larger" data types to "smaller" data types for use, as there is a risk of losing information when transitioning from larger data types to smaller data types.
- If the converted value exceeds the storage range of the target data type, the high byte of this number will be ignored. Example: Convert INT type to BYTE type or DINT type to WORD type.

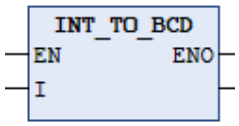
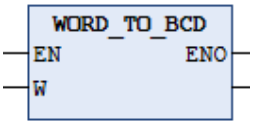
### 1-4-7-1. BCD code and integer data conversion

(1) Instruction overview

BCD code is converted to BYTE, DWORD, INT, and WORD data.

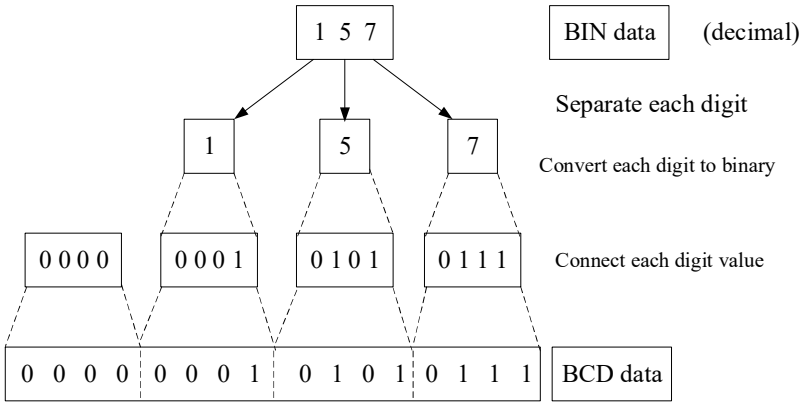
| BCD code and integer data conversion [BCD_TO_XXX/XXX_TO_BCD] | | | |
|---|---|---|---|
| Execution condition | Edge triggering | Suitable model | XS, XSA |
| Firmware | - | Software | - |
| Library | Util | | |

| Instruction | Name | Graph | ST |
|---|---|---|---|
| BCD_TO_BYTE | BCD convert to BYTE |  | BCD_TO_BYTE(B:= ); |
| BCD_TO_DWORD | BCD convert to DWORD |  | BCD_TO_DWORD(X:= ); |
| BCD_TO_INT | BCD convert to INT |  | BCD_TO_INT(B:= ); |
| BCD_TO_WORD | BCD convert to WORD |  | BCD_TO_WORD(W:= ); |
| BYTE_TO_BCD | BYTE convert to BCD |  | BYTE_TO_BCD(B:= ); |
| DWORD_TO_BCD | DWORD convert to BCD |  | DWORD_TO_BCD(X:= ); |

| Instruction | Name | Graph | ST |
|---|---|---|---|
| INT_TO_BCD | INT convert to BCD |  | INT_TO_BCD(I:= ); |
| WORD_TO_BCD | WORD convert to BCD |  | WORD_TO_BCD(W:= ); |

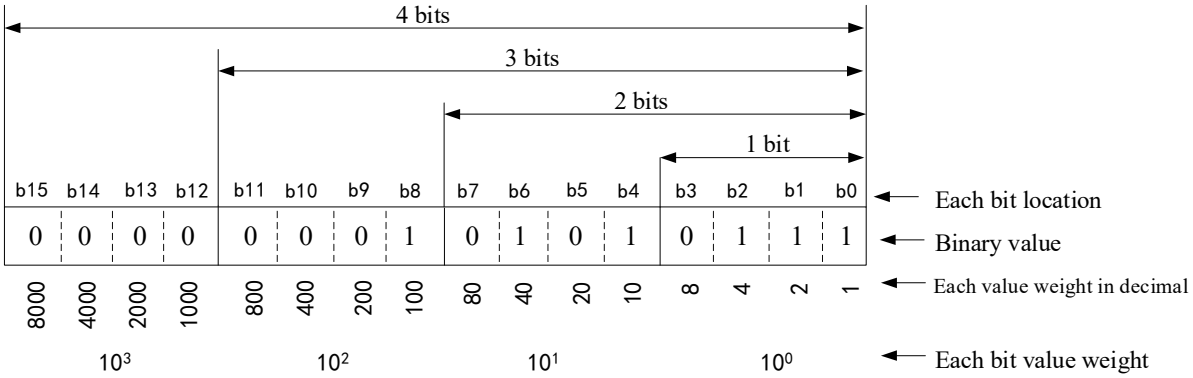
(2) Function

BCD (Binary Coded Decimal... BCD) refers to the use of 4-bit binary numbers to represent the values of each digit in a decimal number in parallel. For example, in BIN data, BCD data 0000 0001 0101 0111 (343) is used to represent the decimal number "157" in the following way.




BCD example

When BCD data is stored in 16 bits memory, it can handle values ranging from 0 to 9999 (the maximum value of 4 bits). The weights of each bit are shown in the following figure:




Weights of each value represented by BCD in decimal system

(3) Note

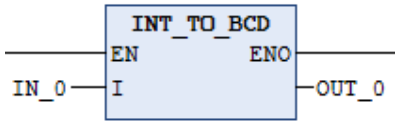
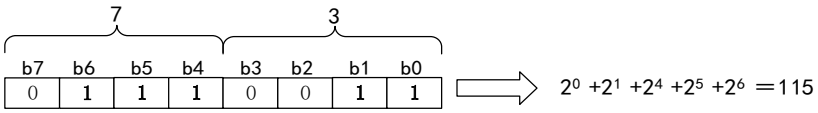
BCD code and integer conversion instruction data types

| Command | Input type | Return value type |
|---|---|---|
| BCD_TO_BYTE | BYTE | BYTE |
| BCD_TO_DWORD | DWORD | DWORD |

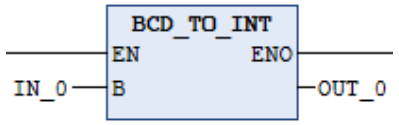
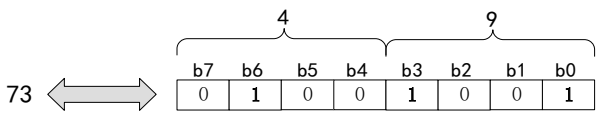
| Command | Input type | Return value type |
|---|---|---|
| BCD_TO_INT | BYTE | INT |
| BCD_TO_WORD | WORD | WORD |
| BYTE_TO_BCD | BYTE | BYTE |
| DWORD_TO_BCD | DWORD | DWORD |
| INT_TO_BCD | INT | BYTE |
| WORD_TO_BCD | WORD | WORD |

(4) Program example

① Convert integer data 73 to BCD code

| DEMO | LD | ST |
|---|---|---|
| Define variables | | VAR<br>IN_0 :INT:=73;<br>OUT_0:BYTE;<br>END_VAR |
| Sample program |  | 1 OUT_0:=INT_TO_BCD(I:= IN_0); |
| Operation result | | |
| Principle analysis |  | |

② Convert BCD code 73 to integer data

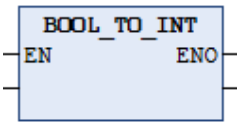
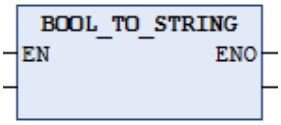
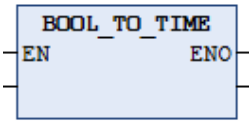
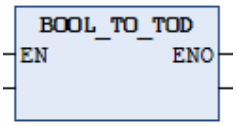
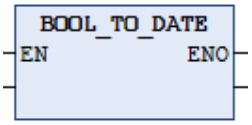
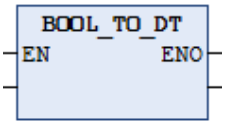
| DEMO | LD | ST |
|---|---|---|
| Define variables | | VAR<br>IN_0 :BYTE:=73;<br>OUT_0:INT;<br>END_VAR |
| Sample program |  | 1 OUT_0:=BCD_TO_INT(B:= IN_0); |
| Operation result | | |
| Principle analysis |  | |

1-4-7-2. BOOL_TO_<TYPE> boolean type conversion data

(1) Instuction overview

Convert Boolean data types to other data types.

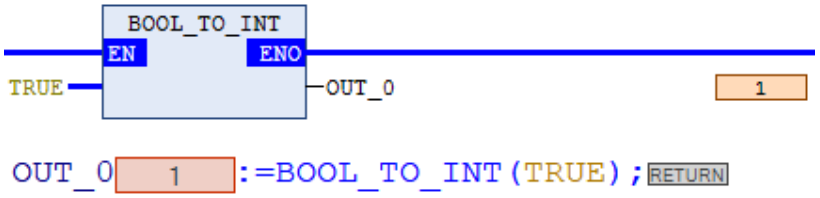
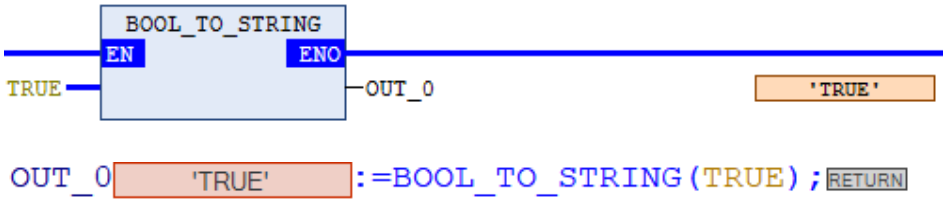
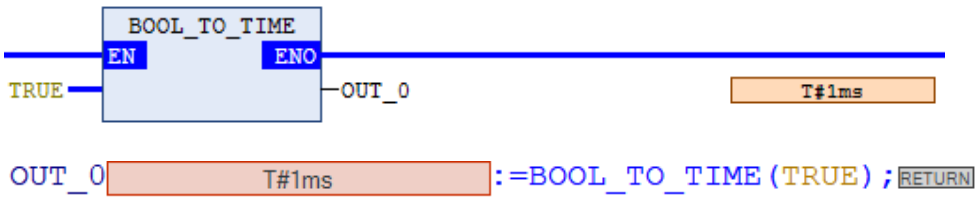
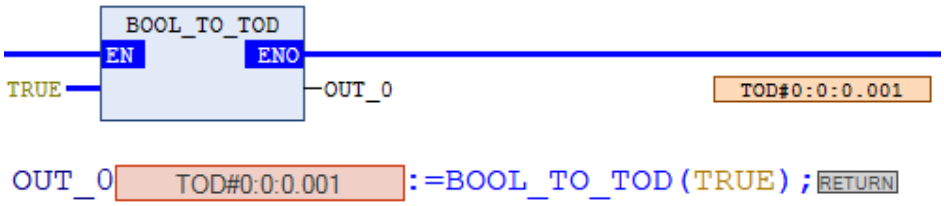
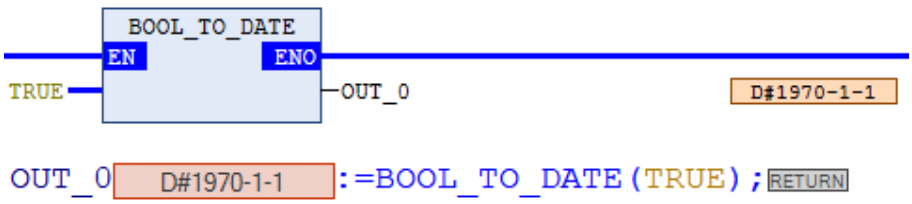
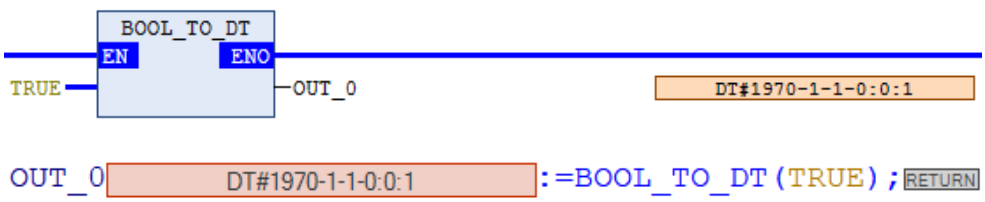
| boolean type conversion data [BOOL_TO_XXX] | | | |
|---|---|---|---|
| Execution condition | Edge triggering | Suitable model | XS, XSA |
| Firmware | - | Software | - |
| Library | - | | |

| Instruction | Name | Graph | ST |
|---|---|---|---|
| BOOL_TO_INT | BOOL convert to INT |  | BOOL_TO_INT |
| BOOL_TO_STRING | BOOL convert to STRING |  | BOOL_TO_STRING |
| BOOL_TO_TIME | BOOL convert to TIME |  | BOOL_TO_TIME |
| BOOL_TO_TOD | BOOL convert to TOD |  | BOOL_TO_TOD |
| BOOL_TO_DATE | BOOL convert to DATE |  | BOOL_TO_DATE |
| BOOL_TO_DT | BOOL convert to DT |  | BOOL_TO_DT |

(2) Function

Supported data types: BYTE, WORD, DWORD, SINT, USINT, INT, UINT, DINT, UDINT, REAL, TIME, DATE, TOD, DT and STRING.

(3) Program example

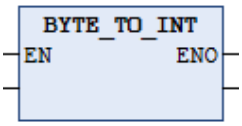
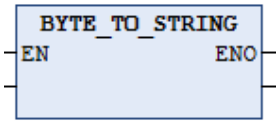
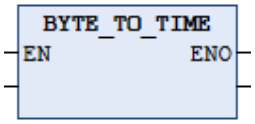
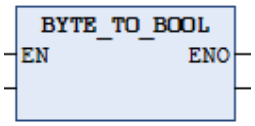
| DEMO | LD/ST |
| --- | --- |
| Program example |  |
| |  |
| |  |
| |  |
| |  |
| |  |

1-4-7-3. BYTE_TO_<TYPE> byte type conversion data

(1) Instruction overview

Convert byte types to other data types.

| byte type conversion data [BYTE_TO_XXX] | | | |
|---|---|---|---|
| Execution condition | Edge triggering | Suitable model | XS, XSA |
| Firmware | - | Software | - |
| Library | - | | |

| Instruction | Name | Graph | ST |
|---|---|---|---|
| BYTE_TO_INT | BYTE convert to INT |  | BYTE_TO_INT |
| BYTE_TO_STRING | BYTE convert to STRING |  | BYTE_TO_STRING |
| BYTE_TO_TIME | BYTE convert to TIME |  | BYTE_TO_TIME |
| BYTE_TO_BOOL | BYTE convert to BOOL |  | BYTE_TO_BOOL |
| ..... | | | |

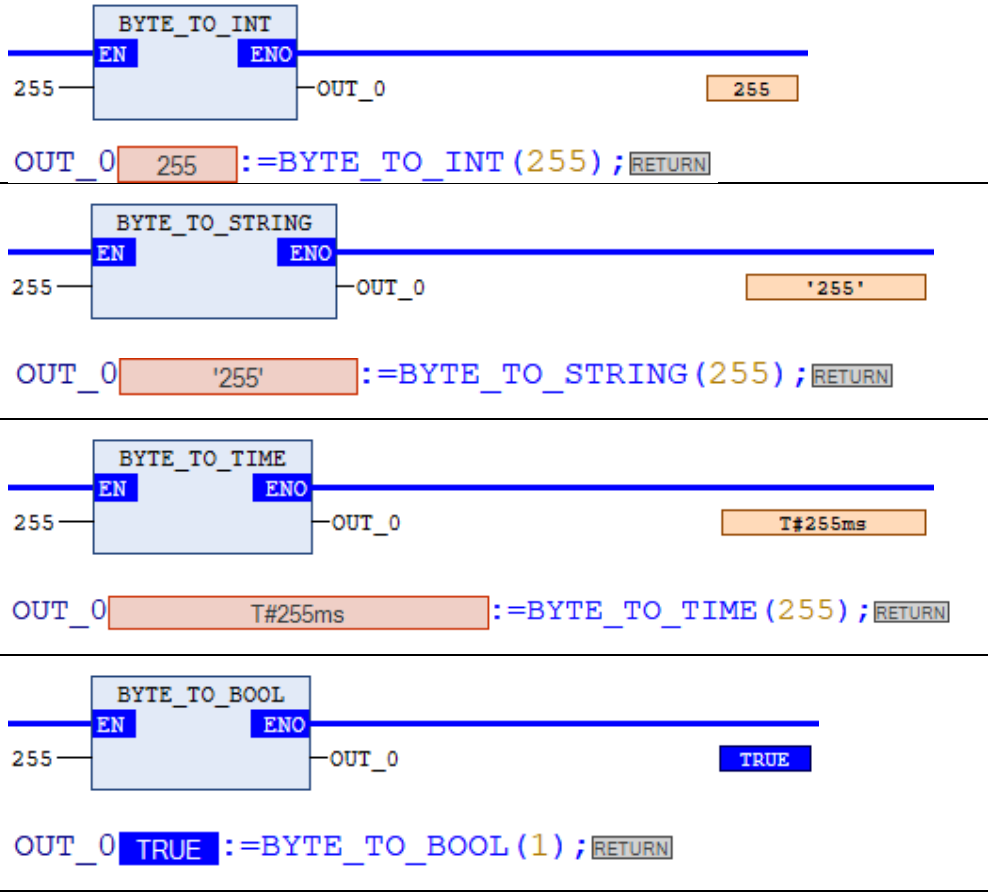
(2) Function

- Support data types: BOOL, WORD, DWORD, SINT, USINT, INT, UINT, DINT, UDINT, REAL, TIME, DATE, TOD, DT, and STRING.
- When the output is BOOL: When the input is not equal to 0, the output is True. When the input is equal to 0, the output is FALSE.
- When the output is TIME or TOD: the input will be converted in milliseconds.
- When outputting as DATE or DT: The input will be converted in seconds.

(3) Note

- More byte type conversion instructions can be found in the software's online help.

(4) Program example

| DEMO | LD/ST |
|---|---|
| Program example |  |

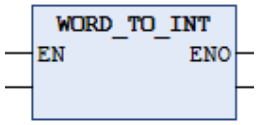
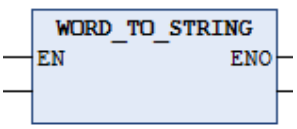


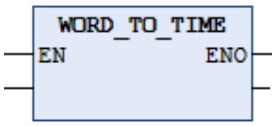
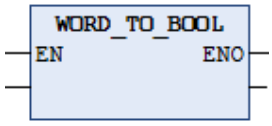
1-4-7-4. < Integer data > _TO_<TYPE> integer type conversion instruction

(1) Instruction overview

Convert integer data to other data types.

| integer type conversion instruction [< Integer data >_TO_XXX] | | | |
|---|---|---|---|
| Execution condition | Edge triggering | Suitable model | XS, XSA |
| Firmware | - | Software | - |
| Library | - | | |

| Instruction | Name | Graph | ST |
|---|---|---|---|
| WORD_TO_INT | WORD convert to INT |  | WORD_TO_INT |
| WORD_TO_STRING | WORD convert to STRING |  | WORD_TO_STRING |

| Instruction | Name | Graph | ST |
|---|---|---|---|
| WORD_TO_TIME | WORD convert to TIME |  | WORD_TO_TIME |
| WORD_TO_BOOL | WORD convert to BOOL |  | WORD_TO_BOOL |
| ..... | | | |

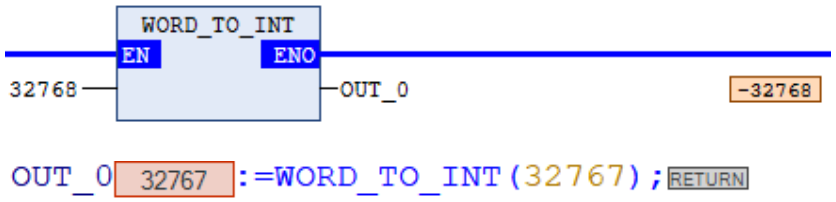
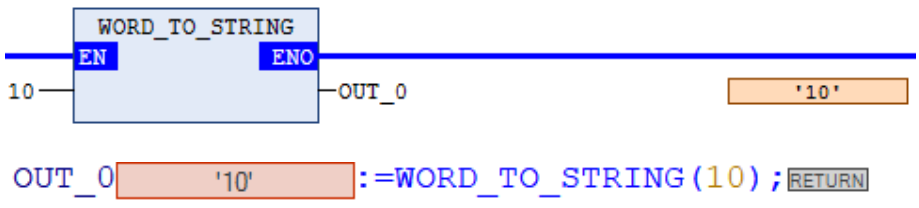
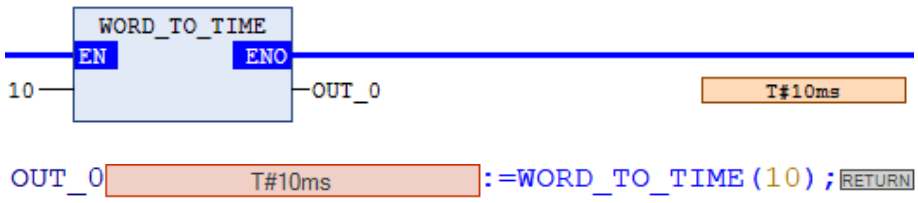
(2) Function

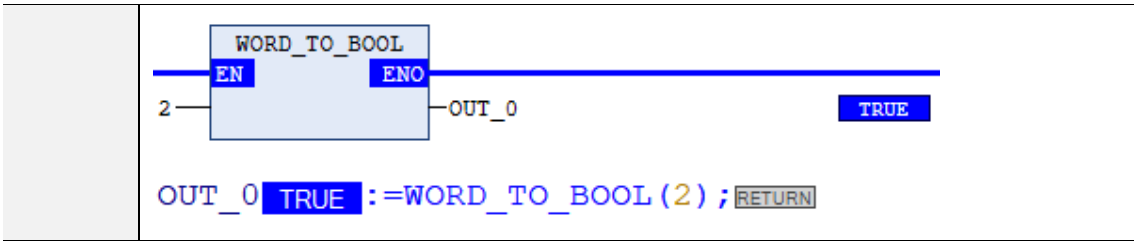
- Supports data types BOOL, BYTE, SINT, WORD, DWORD, USINT, INT, UINT, force, UDINT, REAL, TIME, DATE, TOD, DT, and strings.
- When the output is BOOL: if the input is not equal to 0, the output is TRUE and the input is equal to 0, the output is FALSE.
- When output time or TOD, the input is converted in milliseconds.
- When the output is a date or DT, the input is converted in seconds.

(3) Note

See the software's online help for more integer conversion instructions.

(4) Program example

| DEMO | LD/ST |
|---|---|
| Program example |  <br>  <br>  |

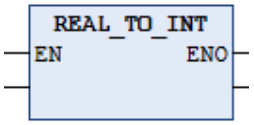
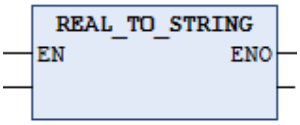
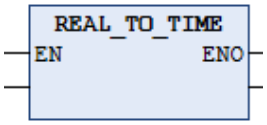
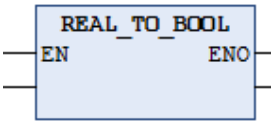



1-4-7-5. REAL_TO_<TYPE> Real type conversion instruction

(1) Instruction overview

Convert floating-point numbers to other types of data. When converting a floating-point number to another type of data, round the value to an integer value (except when converting to STRING) and then convert to the new quantity type.

| Real type conversion instruction [REAL_TO_XXX] | | | |
|---|---|---|---|
| Execution condition | Edge triggering | Suitable model | XS, XSA |
| Firmware | - | Software | - |
| Library | - | | |

| Instruction | Name | Graph | ST |
|---|---|---|---|
| REAL_TO_INT | REAL convert to INT |  | REAL_TO_INT |
| REAL_TO_STRING | REAL convert to STRING |  | REAL_TO_STRING |
| REAL_TO_TIME | REAL convert to TIME |  | REAL_TO_TIME |
| REAL_TO_BOOL | REAL convert to BOOL |  | REAL_TO_BOOL |
| ..... | | | |






(2) Function

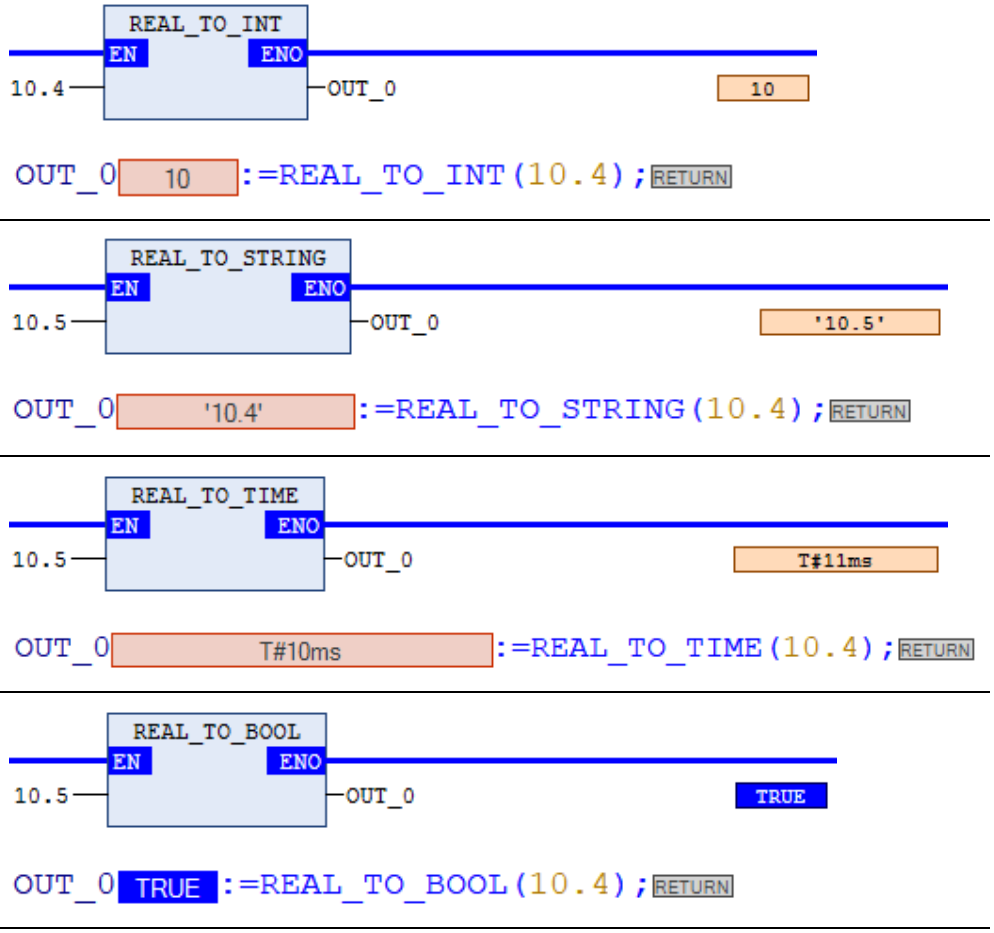
- Support data types BOOL, BYTE, WORD, DWORD, SINT, USINT, INT, UINT, DINT, UDINT, REAL, TIME, DATE, TOD, DT, and STRING.

- When the output is BOOL: The output is TRUE if the input is not equal to 0. When the input is 0, the output is FALSE.
- When the output is TIME or TOD, the input is converted in milliseconds.
- When the output is DATE or DT, the input is converted in seconds.

(3) Note

See the software's online help for more floating-point number type conversion instructions.

(4) Program example

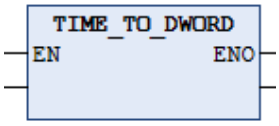
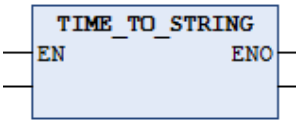
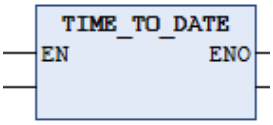
| DEMO | LD/ST |
| --- | --- |
| Program example |  |

1-4-7-6. TIME_TO_<TYPE> time type conversion instruction

(1) Instruction overview

Converts time-based data to other types, and the time is stored internally as a DWORD type in milliseconds (starting at 00:00 am for the TIME_OF_DAY variable).

| time type conversion instruction [TIME_TO_XXX] | | | |
|---|---|---|---|
| Execution condition | Edge triggering | Suitable model | XS, XSA |
| Firmware | - | Software | - |
| Library | - | | |

| Instruction | Name | Graph | ST |
|---|---|---|---|
| TIME_TO_DWORD | TIME convert to DWORD |  | TIME_TO_DWORD |
| TIME_TO_STRING | TIME convert to STRING |  | TIME_TO_STRING |
| TIME_TO_DATE | TIME convert to DATE |  | TIME_TO_DATE |
| ..... | | | |

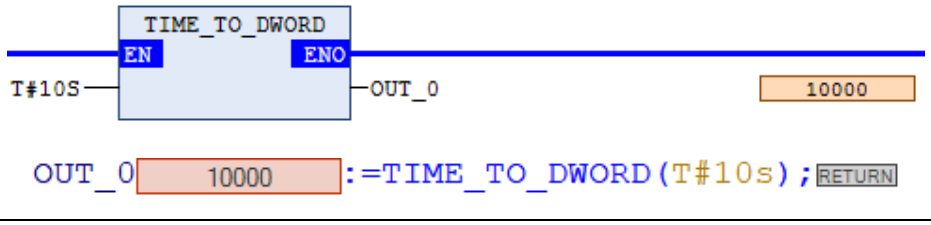
(2) Function

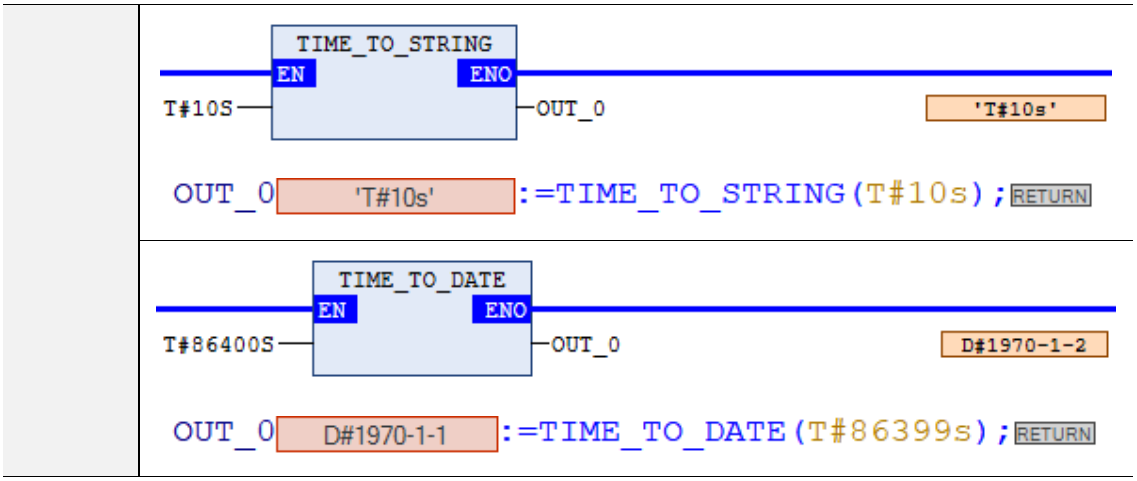
- Supports data types BOOL, BYTE, WORD, DWORD, SINT, USINT, INT, UINT, DINT, UDINT, REAL, TIME, DATE, TOD, DT, and STRING.

(3) Note

- See the software's online help for more time type conversion instructions.

(4) Program example

| DEMO | LD/ST |
|---|---|
| Program example |  |




1-4-7-7. DATE_TO_<TYPE> Date type conversion instruction

(1) Instruction overview

Converts date-type data to other types of data, and dates are stored internally in seconds, starting on January 1, 1970.

| Date type conversion instruction [DATE_TO_XXX] | | | |
|---|---|---|---|
| Execution condition | Edge triggering | Suitable model | XS, XSA |
| Firmware | - | Software | - |
| Library | - | | |

| Instruction | Name | Graph | ST |
|---|---|---|---|
| DATE_TO_INT | DATE convert to INT | DATE_TO_INT EN ENO | DATE_TO_INT |
| DATE_TO_STRING | DATE convert to STRING | DATE_TO_STRING EN ENO | DATE_TO_STRING |
| DATE_TO_BOOL | DATE convert to BOOL | DATE_TO_BOOL EN ENO | DATE_TO_BOOL |
| ..... | | | |

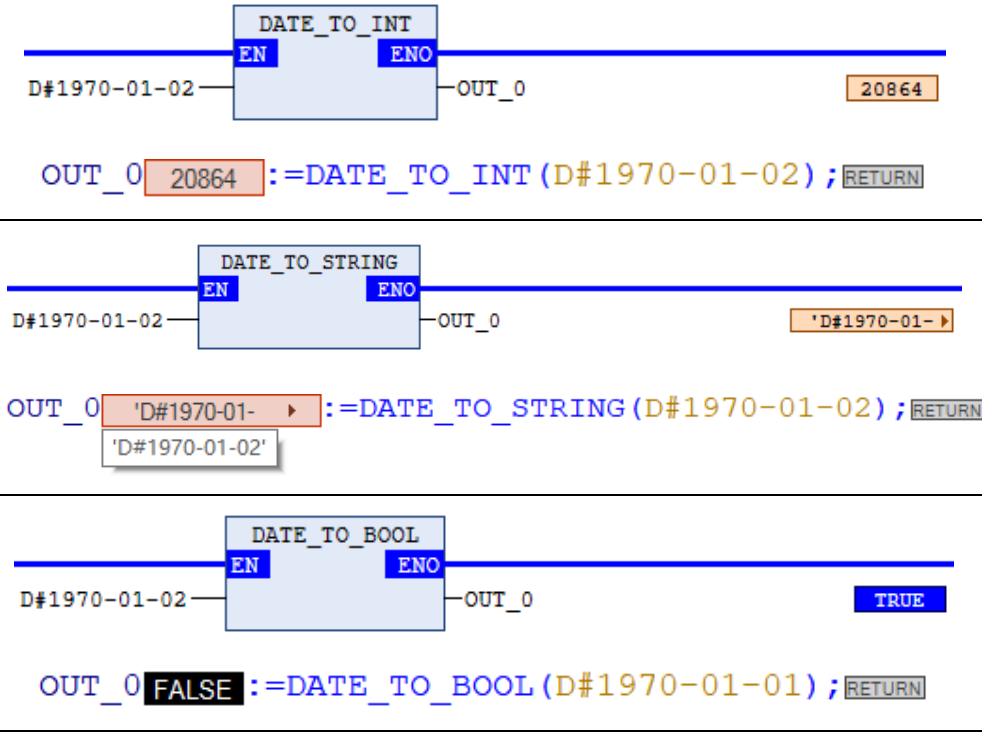
(2) Function

- Supports data types BOOL, BYTE, WORD, DWORD, SINT, USINT, INT, UINT, DINT, UDINT, REAL, TIME, DATE, TOD, DT, and STRING.

(3) Note

- See the software's online help for more date type conversion instructions.

(4) Program example

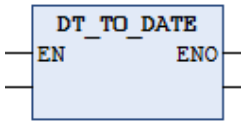
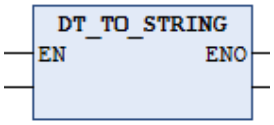
| DEMO | LD/ST |
|---|---|
| Program example |  |

1-4-7-8. DT_TO_<TYPE> Date time type conversion instruction

(1) Instruction overview

Converts date-time data to other types of data, with dates stored internally in seconds, starting on January 1, 1970.

| Date time type conversion instruction [DT_TO_XXX] | | | |
|---|---|---|---|
| Execution condition | Edge triggering | Suitable model | XS, XSA |
| Firmware | - | Software | - |
| Library | - | | |

| Instruction | Name | Graph | ST |
|---|---|---|---|
| DT_TO_DATE | DT convert to DATE |  | DT_TO_DATE |
| DT_TO_STRING | DT convert to STRING |  | DT_TO_STRING |
| ..... | | | |

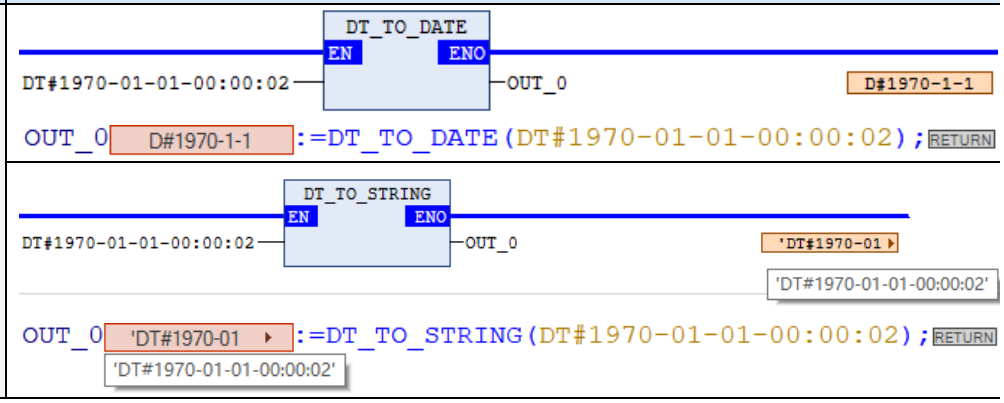
(2) Function

- Supports data types BOOL, BYTE, WORD, DWORD, SINT, USINT, INT, UINT, DINT, UDINT, REAL, TIME, DATE, TOD, DT, and STRING.

(3) Note

- See the software's online help for more date-time type conversion instructions.

(4) Program example

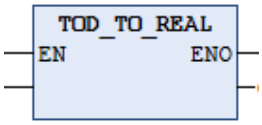
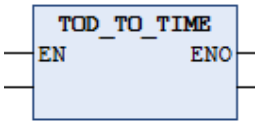
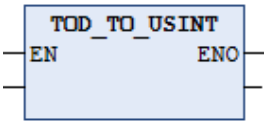
| DEMO | LD/ST |
|---|---|
| Program example |  |

1-4-7-9. TOD_TO_<TYPE> Time type conversion instruction

(1) Instruction overview

Converts time-based data to other types of data, with dates internally converted in milliseconds.

| Time type conversion instruction [TOD_TO_XXX] | | | |
|---|---|---|---|
| Execution condition | Edge triggering | Suitable model | XS, XSA |
| Firmware | - | Software | - |
| Library | - | | |

| Instruction | Name | Graph | ST |
|---|---|---|---|
| TOD_TO_REAL | TOD convert to REAL |  | TOD_TO_REAL |
| TOD_TO_TIME | TOD convert to TIME |  | TOD_TO_TIME |
| TOD_TO_USINT | TOD convert to USINT |  | TOD_TO_USINT |
| ..... | | | |

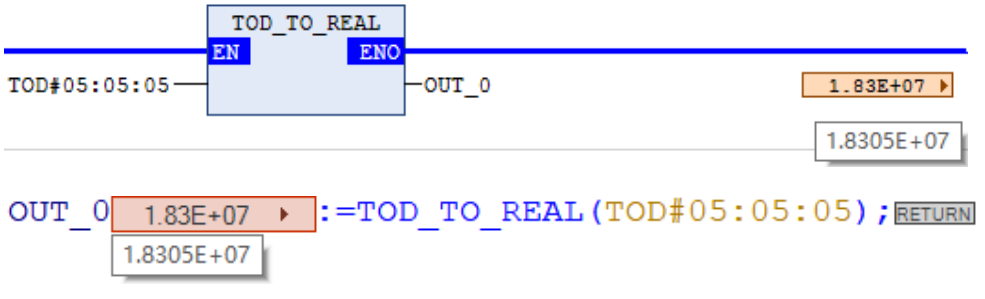
(2) Function

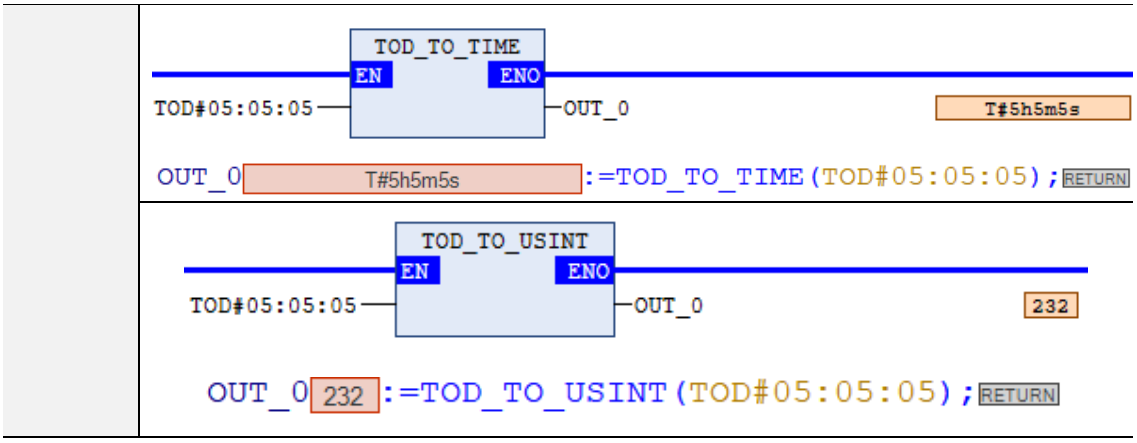
- Supports data types BOOL, BYTE, WORD, DWORD, SINT, USINT, INT, UINT, DINT, UDINT, REAL, TIME, DATE, TOD, DT, and STRING.

(3) Note

More time type data conversion instructions can be found in the software's online help.

(4) Program example

| DEMO | LD/ST |
|---|---|
| Program example |  |




1-4-7-10. STRING_TO_<TYPE> Character type conversion instruction

(1) Instruction overview

To convert a string to another type of data, the string variable must contain a valid target variable value, otherwise the conversion result will be 0.

| Character type conversion instruction [STRING_TO_XXX] | | | |
|---|---|---|---|
| Execution condition | Edge triggering | Suitable model | XS, XSA |
| Firmware | - | Software | - |
| Library | - | | |

| Instruction | Name | Graph | ST |
|---|---|---|---|
| STRING_TO_WORD | STRING convert to WORD | STRING_TO_WORD<br>EN ENO | STRING_TO_WORD |
| STRING_TO_TIME | STRING convert to TIME | STRING_TO_TIME<br>EN ENO | STRING_TO_TIME |
| ..... | | | |

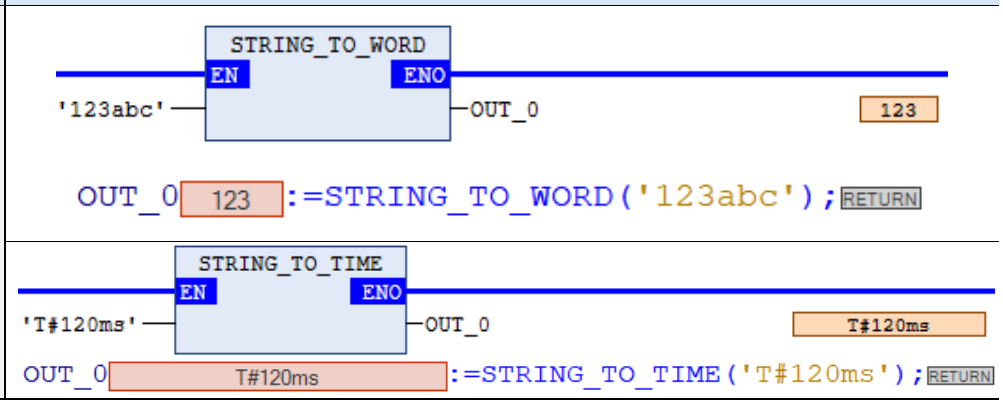
(2) Function

- Supports data types BOOL, BYTE, WORD, DWORD, SINT, USINT, INT, UINT, DINT, UDINT, REAL, TIME, DATE, TOD, DT, and STRING.

(3) Note

- See the software's online help for more string conversion instructions.

(4) Program example

| DEMO | LD/ST |
|---|---|
| Program example |  |

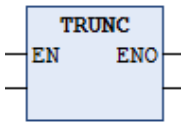


1-4-7-11. Rounding TRUNC

(1) Instruction overview

Truncate the decimal part of the data and only retain the integer part.

| Rounding [TRUNC] | | | |
|---|---|---|---|
| Execution condition | Edge triggering | Suitable model | XS, XSA |
| Firmware | - | Software | - |
| Library | - | | |

| Instruction | Name | Graph | ST |
|---|---|---|---|
| TRUNC | Rounding |  | TRUNC |



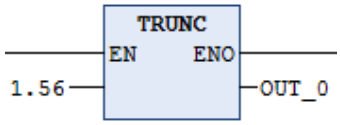
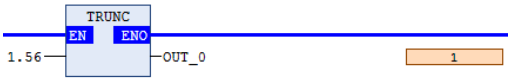
(2) Related variables

| VAR_INPUT | Name | Data type | Valid range | Initial value | Description |
|---|---|---|---|---|---|
| IN | Input data | REAL | As the data type | 0 | Input data |
| VAR_OUTPUT | Name | Data type | Valid range | Initial value | Description |
| OUT | Data after rounding | DINT | As the data type | 0 | Data after rounding |

(3) Note

- See the software's online help for more string conversion instructions.
- When it change from a larger data type to a smaller data type, it may lose information.
- This directive only cuts an integer. If you want to round it up, use the REAL_TO_INT directive.

(4) Program example

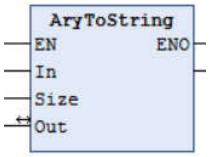
| DEMO | LD | ST |
|---|---|---|
| Define variable | VAR<br>OUT_0:DINT;<br>END_VAR | |
| Program example |  | 1 OUT_0:=TRUNC(-3.56); |
| Operation result |  | 1 OUT_0 -3 :=TRUNC(-3.56) |

1-4-7-12. Array conver to string AryToString

(1) Instruction overview

Converts a BYTE array to a string.

| Array conver to string [AryToString] | | | |
|---|---|---|---|
| Execution condition | Edge triggering | Suitable model | XS, XSA |
| Firmware | - | Software | XS StudioV2.2.0 and above |
| Library | XJ_DataUtils | | |

| Instruction | Name | Graph | ST |
|---|---|---|---|
| AryToString | Array conver to string |  | XDONE:=AryToString(In:= ,<br>Size:= ,<br>Out:= ); |

(2) Related variables

| VAR_INPUT | Name | Data type | Valid range | Initial value | Description |
|---|---|---|---|---|---|
| In | Converted array source | BYTE | As the data type | - | BYTE array, the maximum number of elements is 1985 |
| Size | Convert size | UINT | As the data type | 1 | The number of elements of the array In[] to be converted |
| VAR_IN_OUT | Name | Data type | Valid range | Initial value | Description |
| Out | Convert result | STRING | 1985 bytes | - | String |

(3) Function

- The element values after the BYTE array "In[X]" (inclusive) are treated as character encoding and saved to the string "Out". Add a NULL character at the end of "Out".
- Specify the number of elements of "In[]" to be converted in "Size".
- If there are NULL characters between In[X] and In[X+“Size-1”], only the content before the NULL character is saved to Out.

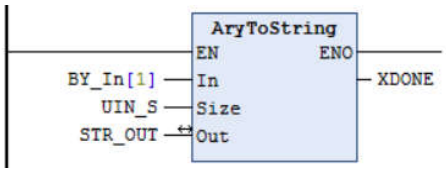
(4) Note

- If the value of Size exceeds the array area of In[], avoid a PLC exception.
- In[] must be a BYTE array; otherwise, the function returns FALSE and Out remains unchanged.
- If the value of "Size" is 0, "Out" is a null-only string, and the function returns True.
- If Size exceeds 1985 bytes, truncated it, the function returns True.

(5) Program example

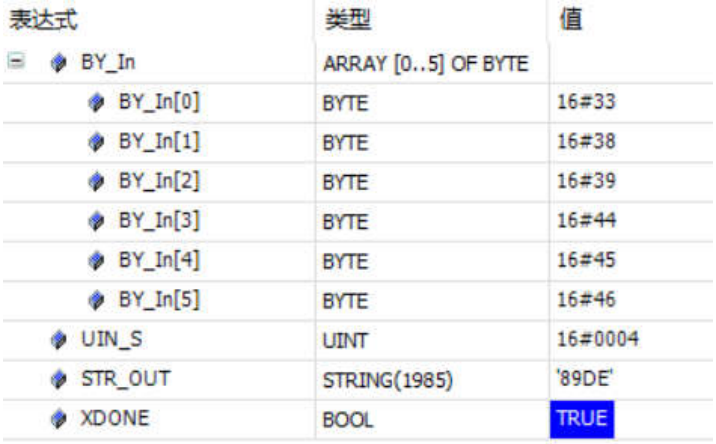
<table>
<tr><th>DEMO</th><th>LD</th><th>ST</th></tr>
<tr><td>Define variable</td><td colspan="2">

```
VAR
   BY_In: ARRAY[0..5] OF BYTE:= [51,56,57,68,69,70];
   UIN_S: UINT:= 4;
   STR_OUT: STRING(1985);
   XDONE: BOOL;
END_VAR
```

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


</td><td>

```
XDONE:=AryToString(In:= BY_In[1],
    Size:= UIN_S,
    Out:= STR_OUT);
```

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


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

1-4-7-13. Date convert to string DateToString

(1) Instruction overview

Convert the date to a string.

| Date convert to string [DateToString] | | | |
|---|---|---|---|
| Execution condition | Edge triggering | Suitable model | XS, XSA |
| Firmware | - | Software | XS StudioV2.2.0 and above |
| Library | XJ_DataUtils | | |

| Instruction | Name | Graph | ST |
|---|---|---|---|
| DateToString | Date convert to string | | xDone:=DateToString(In:= , Out=> ); |

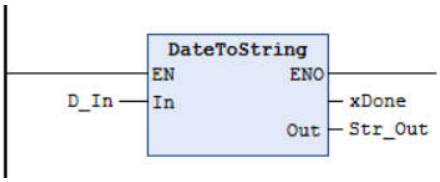
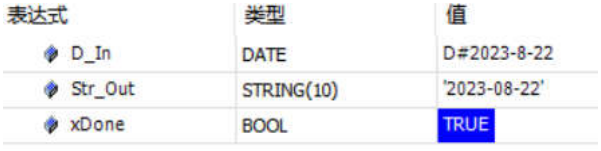
(2) Related variable

| VAR_INPUT | Name | Data type | Valid range | Initial value | Description |
|---|---|---|---|---|---|
| In | Convert source | DATE | As the data type | D#1970-1-1 | Date |
| VAR_OUTPUT | Name | Data type | Valid range | Initial value | Description |
| Out | Convert result | STRING[10] | 10 bytes | - | Convert result string |

(3) Function

- Convert the date "In" to a string.
- Add NULL character at the end of the string "Out".

(4) Program example

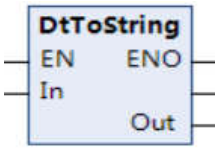
| DEMO | LD | ST |
|---|---|---|
| Define variable | VAR<br>D_In: DATE := D#2023-8-22;<br>Str_Out: STRING(10);<br>xDone: BOOL;<br>END_VAR | |
| Program example |  | xDone := DateToString(<br>In:= D_In,<br>Out=> Str_Out); |
| Operation result |  | |

1-4-7-14. Date time conver to string DtToString

(1) Instruction overview

Convert date and time to a string.

| Date time convert to string [DtToString] | | | |
|---|---|---|---|
| Execution condition | Edge triggering | Suitable model | XS, XSA |
| Firmware | - | Software | XS StudioV2.2.0 and above |
| Library | XJ_DataUtils | | |

| Instruction | Name | Graph | ST |
|---|---|---|---|
| DtToString | Date time conver to string |  | xDone:=DtToString(In:= , Out=> ); |

(2) Related variables

| VAR_INPUT | Name | Data type | Valid range | Initial value | Description |
|---|---|---|---|---|---|
| In | Convert source | DT | As the data type | D#1970-1-1-0:0:0 | Date time |
| VAR_OUTPUT | Name | Data type | Valid range | Initial value | Description |
| Out | Convert result | STRING[19] | 19 bytes | - | string |

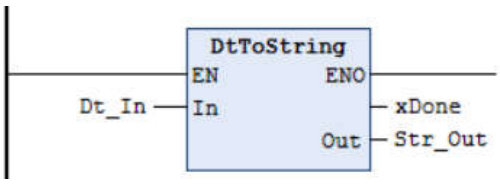
(3) Function

- Convert the date time "In" to a string.
- Add NULL character at the end of the string "Out".

(4) Program example

| DEMO | LD | ST |
|---|---|---|
| Define variable | | |
| Program example | | |
| Operation result | | |

```
VAR
    Dt_In: DATE_AND_TIME:= DT#2023-8-22-13:15:50;
    Str_Out: STRING(19);
    xDone: BOOL;
END_VAR
```



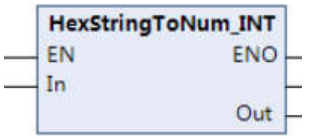

```
xDone := DtToString(
    In:= Dt_In,
    Out=> Str_Out);
```

1-4-7-15. Convert hexadecimal strings to integers HexStringToNum_XXX

(1) Instruction overview

Convert hexadecimal string format to integers.

| Convert hexadecimal strings to integers [HexStringToNum_XXX] | | | |
|---|---|---|---|
| Execution condition | Edge triggering | Suitable model | XS, XSA |
| Firmware | - | Software | XS StudioV2.2.0 and above |
| Library | XJ_DataUtils | | |

| Instruction | Name | Graph | ST |
|---|---|---|---|
| HexStringToNum_XXX | hexadecimal strings →integer |  | xDone:=HexStringToNum_INT(In:= , Out=> ); |

(2) Related variables

| VAR_INPUT | Name | Data type | Valid range | Initial value | Description |
|---|---|---|---|---|---|
| In | Convert source | STRING | As the data type | - | hexadecimal strings |
| VAR_OUTPUT | Name | Data type | Valid range | Initial value | Description |
| Out | Convert result | USINT/UINT/UDINT/ULINT<br>SINT/INT/DINT/LINT | As the data type | - | Integer type |

(3) Function

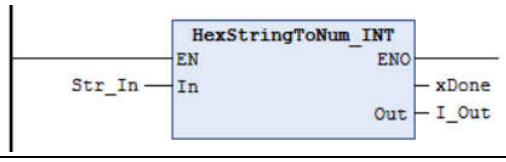
- Convert the string "In" in hexadecimal format to an integer. Regardless of the high half space (16#20) and '0' (16#30). Simultaneously ignore the underline in the string (16#5F).
- The instruction name varies depending on the data type of the "Out" output variable. For example, when "Out" is of type INT, the instruction name is HexStringToNum_INT.

(4) Note

- Even if the conversion result exceeds the valid range of "Out", no exception will occur, but illegal values are saved in "Out".
- When the content of "In" does not end with a NULL character, it may cause conversion exceptions.
- When the content of "In" is a character that cannot be converted to a numerical value, the program encounters an exception and the function returns a value of FALSE, while "Out" remains unchanged.

(5) Program example

| DEMO | LD | ST |
|---|---|---|
| Define variable | | VAR<br>Str_In: STRING(1985);<br>I_Out: INT;<br>xDone: BOOL;<br>END_VAR |

<table>
<tr>
<td>Program example</td>
<td> 

<pre>xDone := HexStringToNum_INT(
    In:= Str_In,
    Out=> I_Out);</pre></td>
</tr>
<tr>
<td>Operation result</td>
<td>
<table>
<tr><th>表达式</th><th>类型</th><th>值</th></tr>
<tr><td>Str_In</td><td>STRING(1985)</td><td>'200'</td></tr>
<tr><td>I_Out</td><td>INT</td><td>512</td></tr>
<tr><td>xDone</td><td>BOOL</td><td>TRUE</td></tr>
</table>
<table>
<tr><th>表达式</th><th>类型</th><th>值</th></tr>
<tr><td>Str_In</td><td>STRING(1985)</td><td>'-F'</td></tr>
<tr><td>I_Out</td><td>INT</td><td>-15</td></tr>
<tr><td>xDone</td><td>BOOL</td><td>TRUE</td></tr>
</table>
</td>
</tr>
</table>

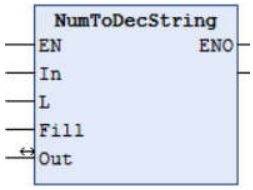
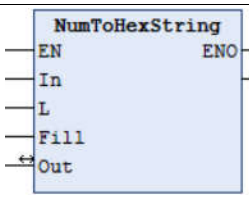
1-4-7-16. Convert integers to fixed length decimal/hexadecimal strings NumToDecString/NumToHexString

(1) Instruction overview

NumToDecString: Converts an integer to a fixed length decimal string format.

NumToHexString: Converts an integer to a fixed length hexadecimal string format.

| Convert integers to fixed length decimal/hexadecimal strings [NumToDecString/NumToHexString] | | | |
|---|---|---|---|
| Execution condition | Edge triggering | Suitable model | XS, XSA |
| Firmware | - | Software | XS StudioV2.2.0 and above |
| Library | XJ_DataUtils | | |

| Instruction | Name | Graph | ST |
|---|---|---|---|
| NumToDecString | Fixed length decimal string conversion |  | xDone:=NumToDecString(In:= ,<br>L:= ,<br>Fill:= ,<br>Out:= ); |
| NumToHexString | Fixed length hexadecimal string conversion |  | xDone:=NumToHexString(In:= ,<br>L:= ,<br>Fill:= ,<br>Out:= ); |

(2) Related variables

| VAR_INPUT | Name | Data type | Valid range | Initial value | Description |
|---|---|---|---|---|---|
| In | Convert source | USINT/UINT/UDINT/ULINT<br>SINT/INT/DINT/LINT | As the data type | - | Integer |
| L | Characters | UINT | 0-1985 | 1 | Out characters |
| Fill | Add characters | _eFiLL_CHR | 1 (_BLANK)<br>2 (_ZERO) | 1 (_BLANK) | Add characters |
| VAR_IN_OUT | Name | Data type | Valid range | Initial value | Description |
| Out | Convert result | STRING | As the data type | - | String |

(3) Function

- NumToDecString: Converts the integer "In" to a decimal format string of UTF-8 half width English numerals. When "In" is a negative number, add a "-" (minus sign) at the beginning.
- NumToHexString: Converts the integer "In" to a hexadecimal format string of UTF-8 half width English numerals. When "In" is a negative number, it is expressed in the form of a complement of 2 (taking the inverse then add 1).
- The above instructions set the number of characters in the string "Out" to "L". When the number of characters is insufficient, add the character specified by "Fill" to the high position.
- When the number of characters in the conversion result is more than "L", only the "L" character is taken into "Out" from the lowest position of the conversion result.
- The data type of "Fill" is the enumerator_eFilL_CHR. The meaning of the enumeration body is shown in the following figure:

| Enumerating elements | Meaning |
|---|---|
| 1 (_BLANK) | ‘’ (Half width blank character) |
| 2 (_ZERO) | ‘0’ |

(4) Note

- When the value of "L" is 0, the value of "Out" remains unchanged.
- When the number of characters in the conversion result exceeds "L", save the "L" bits in "Out" from the low position of the conversion result, as shown in the following example:

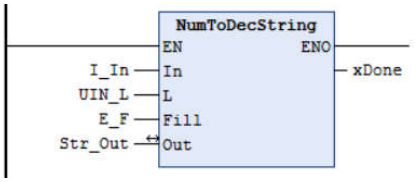
| Instruction | “In” value | “L” value | “Out” value |
|---|---|---|---|
| NumToDecString | 128 | 2 | 28 |
| NumToHexString | | | 80 |

- When the value of "L" exceeds the valid range, the program encounters an exception, and the function returns a value of FALSE with no change in "Out".
- Compilation exception occurs when the value of "Fill" exceeds the valid range.
- When the conversion result exceeds the range of "Out", the output is an error value.

(5) Program example

| NumToDecString | LD | ST |
|---|---|---|
| Define variable | | |
| Program example | | |
| Operation result | | |

```
VAR
    I_In: INT;
    UIN_L: UINT;
    E_F: _eFILL_CHR;
    Str_Out: STRING(1985);
    xDone: BOOL;
END_VAR
```



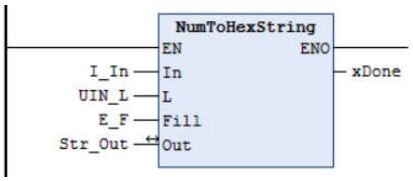

```
xDone:=NumToDecString(
    In:= I_In,
    L:= UIN_L,
    Fill:= E_F,
    Out:= Str_Out);
```

| 表达式 | 类型 | 值 |
|---|---|---|
| I_In | INT | 512 |
| UIN_L | UINT | 8 |
| E_F | _EFILL_CHR | _BLANK |
| Str_Out | STRING(1985) | ' 512' |
| xDone | BOOL | TRUE |

| 表达式 | 类型 | 值 |
|---|---|---|
| I_In | INT | -512 |
| UIN_L | UINT | 8 |
| E_F | _EFILL_CHR | _ZERO |
| Str_Out | STRING(1985) | '-0000512' |
| xDone | BOOL | TRUE |

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

```
VAR
    I_In: INT;
    UIN_L: UINT;
    E_F: _eFILL_CHR;
    Str_Out: STRING(1985);
    xDone: BOOL;
END_VAR
```

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

```
xDone:=NumToHexString(
    In:= I_In,
    L:= UIN_L,
    Fill:= E_F,
    Out:= Str_Out);
```

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

| 表达式 | 类型 | 值 |
|---|---|---|
| I_In | INT | 128 |
| UIN_L | UINT | 8 |
| E_F | _EFILL_CHR | _BLANK |
| Str_Out | STRING(1985) | ' 80' |
| xDone | BOOL | TRUE |

| 表达式 | 类型 | 值 |
|---|---|---|
| I_In | INT | -128 |
| UIN_L | UINT | 8 |
| E_F | _EFILL_CHR | _BLANK |
| Str_Out | STRING(1985) | 'FFFFFF80' |
| xDone | BOOL | TRUE |

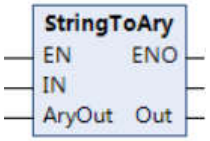
</td></tr>
</table>

1-4-7-17. String conver to array StringToAry

(1) Instruction overview

Convert the string to a BYTE array.

| String conver to array [StringToAry] | | | |
|---|---|---|---|
| Execution condition | Edge triggering | Suitable model | XS, XSA |
| Firmware | - | Software | XS StudioV2.2.0 and above |
| Library | XJ_DataUtils | | |

| Instruction | Name | Graph | ST |
|---|---|---|---|
| StringToAry | String conver to array |  | XDONE:=StringToAry(In:= , AryOut:= , Out=> ); |



(2) Related variables

| VAR_INPUT | Name | Data type | Valid range | Initial value | Description |
|---|---|---|---|---|---|
| In | Convert source | STRING | Maximum 1985 characters | - | String |
| VAR_IN_OUT | Name | Data type | Valid range | Initial value | Description |
| AryOut | Array after converting | BYTE | As the data type | - | BYTE type array |
| VAR_OUTPUT | Name | Data type | Valid range | Initial value | Description |
| Out | Bytes after converting | UINT | 0-1985 | - | Bytes after converting |

(3) Function

- Treat the character encoding of the string "n" as a numerical value and save each character one by one in the BYTE arrangement "AryOut[]". Save the number of converted bytes in "Out".

(4) Note

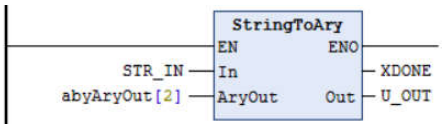
- The NULL character at the end of "In" will not be saved in "AryOut[]".
- When "In" is a string with only NULL characters, the value of "Out" is 0, and "AryOut[]" remains unchanged.
- When the number of bytes in "In" is greater than the number of elements in "AryOut[]", avoid PLC exceptions.
- When the number of bytes in "In" exceeds the valid range (1985), characters outside the valid range will be truncated, and the function will convert 1985 characters normally.

(5) Program example

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

```
VAR
   STR_IN: STRING(1985):= 'XYZ';
   abyAryOut: ARRAY[0..5] OF BYTE;
   U_OUT: UINT;
   XDONE: BOOL;
END_VAR
```

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



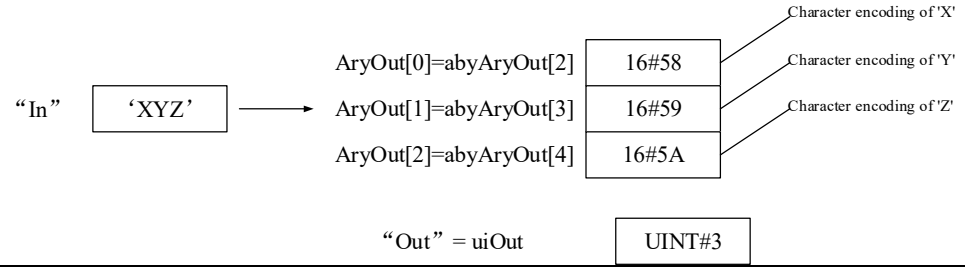

</td><td>

```
XDONE:=StringToAry(In:= STR_IN,
    AryOut:= abyAryOut[2],
    Out=> U_OUT);
```

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

| 表达式 | 类型 | 值 |
|---|---|---|
| STR_IN | STRING(1985) | '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 |
| U_OUT | UINT | 16#0003 |
| XDONE | BOOL | TRUE |

</td></tr>
<tr><td>Principle analysis</td><td colspan="2">




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

1-4-7-18. Time convert to string TodToString

(1) Instruction overview

Time convert to string.

| Time convert to string [TodToString] | | | |
|---|---|---|---|
| Execution condition | Edge triggering | Suitable model | XS, XSA |
| Firmware | - | Software | XS StudioV2.2.0 and above |
| Library | XJ_DataUtils | | |

| Instruction | Name | Graph | ST |
|---|---|---|---|
| TodToString | Time convert to string | | xDone:=TodToString(In:= , Out=> ); |

(2) Related variables

| VAR_INPUT | Name | Data type | Valid range | Initial value | Description |
|---|---|---|---|---|---|
| In | Convert source | TOD | As the data type | TOD#0:0:0 | Hour minute second |
| VAR_OUTPUT | Name | Data type | Valid range | Initial value | Description |
| Out | Convert result | STRING[13] | 13 bytes | - | Conversion result string |

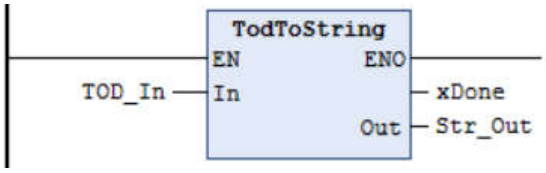
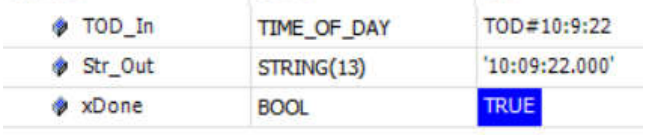
(3) Function

- Convert the time "In" to a string. Add NULL character at the end of "Out".

(4) Note

- "Out" is expressed in milliseconds.

(5) Program example

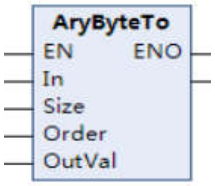
| DEMO | LD | ST |
|---|---|---|
| Define variable | VAR<br>TOD_In: TIME_OF_DAY;<br>Str_Out: STRING(13);<br>xDone: BOOL;<br>END_VAR | |
| Program example |  | xDone:=TodToString(<br>In:= TOD_In,<br>Out=> Str_Out); |
| Operation result |  | |

1-4-7-19. Convert from byte array AryByteTo

(1) Instruction overview

Combine BYTE type array elements and save them to variables.

| Convert from byte array [AryByteTo] | | | |
|---|---|---|---|
| Execution condition | Edge triggering | Suitable model | XS, XSA |
| Firmware | - | Software | XS StudioV2.2.0 and above |
| Library | XJ_DataUtils | | |

| Instruction | Name | Graph | ST |
|---|---|---|---|
| AryByteTo | Convert from byte array |  | XDONE:=AryByteTo(In:= ,<br>Size:= ,<br>Order:= ,<br>OutVal:= ); |

(2) Related variables

| VAR_INPUT | Name | Data type | Valid range | Initial value | Description |
|---|---|---|---|---|---|
| In | Convert object array | BYTE | As the data type | - | Convert object array |
| Size | Convert object element sizes | UINT | As the data type | - | The number of In[] elements to convert |
| Order | Convert order | _eBYTE_ORDER | As the data type | _LOW_HIGH | The position to be converted |
| OutVal | Convert result | BOOL/BYTE/WORD/DWORD/ LWORD/USINT/UINT/UDINT/ULINT/ SINT/INT/DINT/LINT/REAL/LREAL/ TIME/DATE/TOD/DT/STRING | As the data type | - | Convert result |

(3) Function

- Combine the starting "Size" elements of the conversion object array "In[X]" according to the data type size of the conversion result "OutVal", and then save them to "OutVal".
- When combining the "In[]" element, specify the order in the conversion order "Order"; The data type of "Order" is enumeration_eBYTE_ORDER; The meaning of enumeration values is as follows.

| Enum | Meaning |
|---|---|
| _LOW_HIGH | First low byte, then high byte |
| _HIGH_LOW | First high byte, then low byte |

- The following two situations introduce the functions of AryByteTo.

**(1) The data type of "OutVal" is 1 byte**

When the data type of "OutVal" is 1 byte, save "In[]" in "OutVal" in units of 1 byte.

There are several types of 1-byte data:

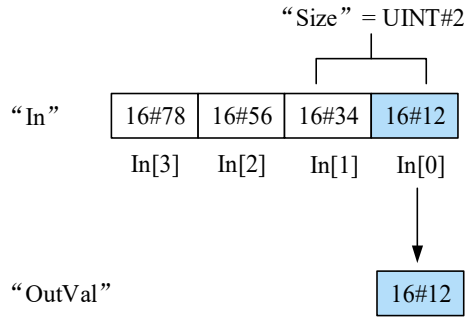
| Type | Data type |
|---|---|
| Bit string | BYTE |

| | |
|---|---|
| Integer | USINT, SINT |
| Real | None |
| Time, duration, date, string | None |

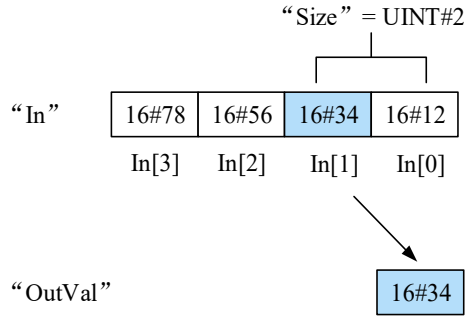
The saving method is as follows:

| "Order" value | "OutVal" data type | Method for saving to "OutVal" |
|---|---|---|
| LOW_HIGH | BOOL | Save the value of the 0th bit of In[0] to "OutVal" |
| | Not BOOL | Save the value of In[0] to "OutVal" |
| HIGH_LOW | BOOL | Save the value of the 0th bit of In[1] to "OutVal" |
| | Not BOOL | Save the value of In[1] to "OutVal" |

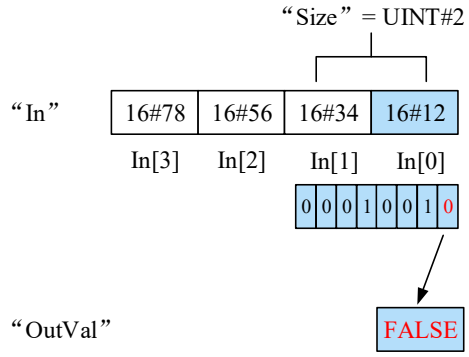
The example of "OutVal" being a SINT type variable with "Size"=UINT#2 and "Order"=_LOW_HIGH is as follows:



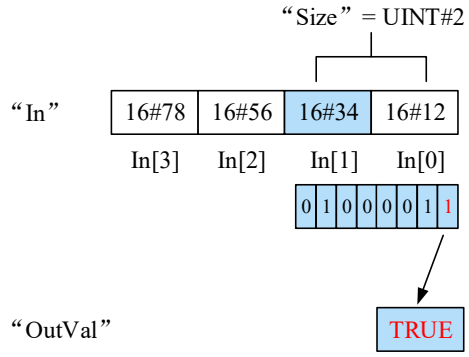

The example of "OutVal" being a SINT type variable with "Size"=UINT#2 and "Order"= _HIGH_LOW is as follows:



The example of "OutVal" being a BOOL type variable with "Size"=UINT#2 and "Order"= _LOW_HIGH is as follows:




The example of "OutVal" being a BOOL type variable with "Size"=UINT#2 and "Order"= _HIGH_LOW is as follows:




**(2) The data type of “OutVal” is 2 bytes or more**

When the data type of "OutVal" is 2 bytes or more, combine "In[]" in units of the data type size of "OutVal" and save it to "OutVal".

There are several types of data with 2 bytes or more:

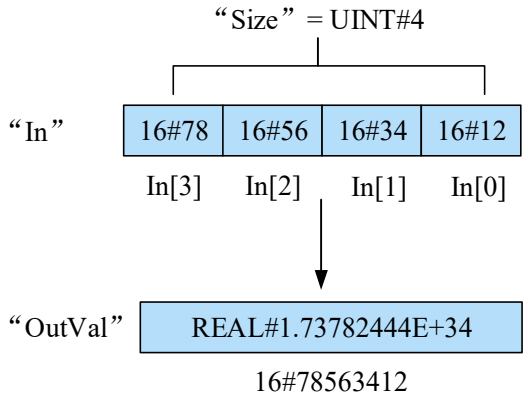
| Type | Data type |
|---|---|
| Bit string | WORD, DWORD, LWORD |
| Integer | UINT, UDINT, ULINT, INT, DINT, LINT |
| Real | REAL, LREAL |
| Time, duration, date, string | TIME, DATE, TOD, TD, STRING of 2 bytes or more |

The saving method is as follows:

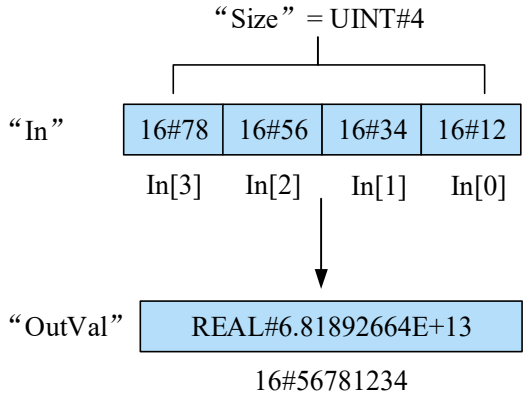
- Create 1 word (2 bytes) of data based on the value of "Order" combined with "In[0]" and "In[1]". If the value of "Order" is _LOW_HIGH, then the high byte is "In[1]" and the low byte is "In[0]"; If the value of "Order" is _HIGH_LOW, then the high byte is "In[0]" and the low byte is "In[1]". Combine the values after "In[2]" and "In[3]" using the same method to create multiple 1-word data.
- Based on the data type size of "OutVal", combine the multiple 1-word data. For example, when "OutVal" is a DWORD type, combine four 1-word data. If "OutVal" is the entire array, save "In[i]" (where i is even) to "OutVal[i+1]". Save "In[i]" (where i is odd) to "OutVal[i-1]". If the value of "Size" is odd, it will be saved until "OutVal ["Size"]", and 16#00 will be saved in "OutVal[" Size

"-1]". Save the obtained data to "OutVal".

The example of "OutVal" being a REAL type variable with "Size"=UINT#4 and "Order"=_LOW_HIGH is as follows:




The example of "OutVal" being a REAL type variable, with "Size"=UINT#4 and "Order"=_HIGH_LOW is as follows:

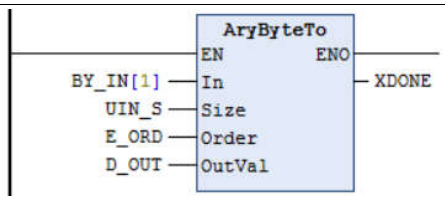



(4) Note

- When the value of "Size" exceeds the array area of "In[]", it is necessary to avoid PLC exceptions.
- "OutVal" cannot be the entire BOOL type array, otherwise the program running result will be incorrect.
- When the value of "Size" is 0, the value of "Out" is true, and "OutVal" remains unchanged.

(5) Program example

| DEMO | LD | ST |
|---|---|---|
| Define variable | VAR<br>BY_IN: ARRAY[0..5] OF BYTE:= [16#21,16#52,16#62,16#81,16#66,16#55];<br>UIN_S: UINT:= 3;<br>E_ORD: _eBYTE_ORDER:= _eBYTE_ORDER._LOW_HIGH;<br>D_OUT: DWORD;<br>XDONE: BOOL;<br>END_VAR | |

| | | |
|---|---|---|
| Program example |  | XDONE:=AryByteTo(<br>In:= BY_IN[1],<br>Size:= UIN_S,<br>Order:= E_ORD,<br>OutVal:= D_OUT); |
| Operation result | | |

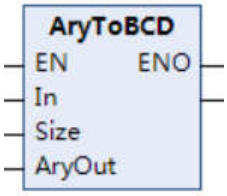
| 表达式 | 类型 | 值 |
|---|---|---|
| BY_IN | ARRAY [0..5] OF BYTE | |
| BY_IN[0] | BYTE | 16#21 |
| BY_IN[1] | BYTE | 16#52 |
| BY_IN[2] | BYTE | 16#62 |
| BY_IN[3] | BYTE | 16#81 |
| BY_IN[4] | BYTE | 16#66 |
| BY_IN[5] | BYTE | 16#55 |
| UIN_S | UINT | 16#0003 |
| E_ORD | _EBYTE_ORDER | _LOW_HIGH |
| D_OUT | DWORD | 16#00816252 |
| XDONE | BOOL | TRUE |

1-4-7-20. Batch conversion of arrays to BCD AryToBCD

(1) Instruction overview

Convert the elements of an unsigned integer array to BCD.

| Batch conversion of arrays to BCD [AryToBCD] | | | |
|---|---|---|---|
| Execution condition | Edge triggering | Suitable model | XS, XSA |
| Firmware | - | Software | XS StudioV2.2.0 and above |
| Library | XJ_DataUtils | | |

| Instruction | Name | Graph | ST |
|---|---|---|---|
| AryToBCD | Batch conversion of arrays to BCD |  | AryToBCD(In:=,<br>Size:= ,<br>AryOut:= ); |

(2) Related variables

| VAR_INPUT | Name | Data type | Valid range | Initial value | Description |
|---|---|---|---|---|---|
| In | Convert array source | USINT/UINT/ UDINT/ULINT | As the data type | - | Unsigned integer array |
| Size | Convert size | UINT | As the data type | 1 | The number of In[] elements to convert |
| VAR_IN_OUT | Name | Data type | Valid range | Initial value | Description |
| AryOut | Result array | Consistent with In | As the data type | - | BCD array |

(3) Function

- Convert the "Size" elements starting from In[0] of the unsigned integer array In[] to BCD and output them to the BCD array AryOut[].
- The effective range of In[] and AryOut[] varies depending on the combination of element data types.

| In[] element data type | In[]effective range | AryOut[] effective range |
|---|---|---|
| USINT | 0-99 | 16#00-16#99(BCD) |
| UINT | 0-9999 | 16#0000-16#9999(BCD) |
| UDINT | 0-99999999 | 16#0000_0000-16#9999_9999(BCD) |
| ULINT | 0-9999999999999999 | 16#0000_0000_0000_0000-16#9999_9999_9999_9999(BCD) |

(4) Note

- Unable to convert signed BCD. Therefore, please set the data type of In[] to any one of the unsigned integers (USINT, UINT, UDINT, ULINT).
- When the value of "Size" is 0, the function returns a value of True, and the value of AryOut[] remains unchanged.
- When the value of In[] exceeds the valid range, the function returns a value of FALSE, and the value of AryOut[] remains unchanged.

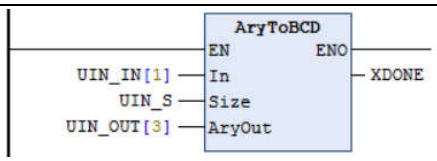
- When the value of "Size" exceeds the array area of In[] or AryOut[], it is necessary to avoid PLC exceptions.
- The "In[] array" needs to use an integer type. If it is determined that it does not belong to this type, a compilation error will be reported.
- The "In[] array" type is not USINT/UINT/UDINT/ULINT, the runtime function returns a value of FALSE, and the value of AryOut[] remains unchanged.

(5) Program example

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

```
VAR
   UIN_IN: ARRAY[0..5] OF UINT:= [5,16,26,168,12,512];
   UIN_S: UINT:= 3;
   UIN_OUT: ARRAY[0..5] OF UINT;
   XDONE: BOOL;
END_VAR
```

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




</td><td>

```
XDONE:=AryToBCD(
    In:= UIN_IN[1],
    Size:= UIN_S,
    AryOut:= UIN_OUT[3]);
```

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

| 表达式 | 类型 | 值 |
|---|---|---|
| UIN_IN | ARRAY [0..5] OF UINT | |
| UIN_IN[0] | UINT | 16#0005 |
| UIN_IN[1] | UINT | 16#0010 |
| UIN_IN[2] | UINT | 16#001A |
| UIN_IN[3] | UINT | 16#00A8 |
| UIN_IN[4] | UINT | 16#000C |
| UIN_IN[5] | UINT | 16#0200 |
| UIN_S | UINT | 16#0003 |
| UIN_OUT | ARRAY [0..5] OF UINT | |
| UIN_OUT[0] | UINT | 16#0000 |
| UIN_OUT[1] | UINT | 16#0000 |
| UIN_OUT[2] | UINT | 16#0000 |
| UIN_OUT[3] | UINT | 16#0016 |
| UIN_OUT[4] | UINT | 16#0026 |
| UIN_OUT[5] | UINT | 16#0168 |
| XDONE | BOOL | TRUE |

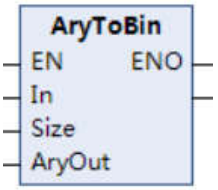
</td></tr>
</table>

1-4-7-21. Batch convert arrays to bit strings AryToBin

(1) Instruction overview

Convert the elements of the BCD array into bit strings.

| Batch convert arrays to bit strings [AryToBin] | | | |
|---|---|---|---|
| Execution condition | Edge triggering | Suitable model | XS, XSA |
| Firmware | - | Software | XS StudioV2.2.0 and above |
| Library | XJ_DataUtils | | |

| Instruction | Name | Graph | ST |
|---|---|---|---|
| AryToBin | Batch convert arrays to bit strings |  | XDONE:=AryToBin(In:= ,<br>Size:= ,<br>AryOut:= ); |

(2) Related variables

| VAR_INPUT | Name | Data type | Valid range | Initial value | Description |
|---|---|---|---|---|---|
| In | Convert array source | BYTE/WORD/ DWORD/LWORD | As the data type | - | BCD array |
| Size | Number of Conversion Elements | UINT | As the data type | 1 | The number of In[] elements to convert |
| VAR_IN_OUT | Name | Data type | Valid range | Initial value | Description |
| AryOut | Result array | Consistent with In | As the data type | - | Converted bit string array |

(3) Function

- Convert the "Size" elements starting from In[0] of the BCD array In[] to bit strings, and output the bit string array AryOut[].
- The effective range of In[] and AryOut[] varies depending on the combination of element data types.

| The data type of the In[] element | The effective range of In[] | The effective range of AryOut[] |
|---|---|---|
| BYTE | 16#00-16#99(BCD) | 0-99 |
| WORD | 16#0000-16#9999(BCD) | 0-9999 |
| DWORD | 16#0000_0000-16#9999_9999(BCD) | 0-99999999 |
| LWORD | 16#0000_0000_0000_0000-16#9999_9999_9999_9999(BCD) | 0-9999999999999999 |

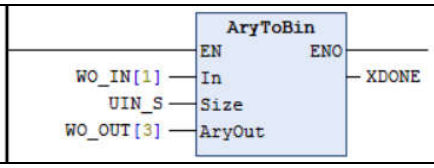
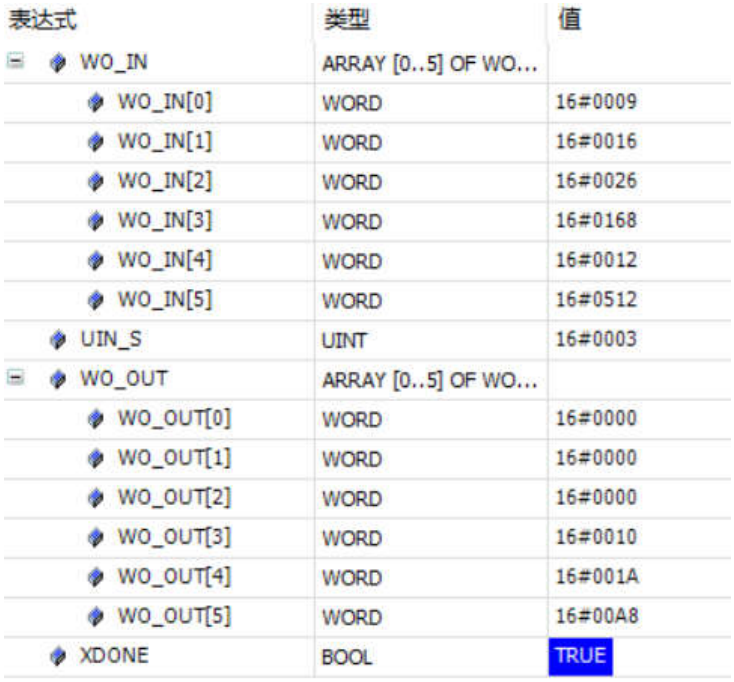
(4) Note

- Unable to convert signed BCD. Therefore, please set the data type of AryOut[] to any one of the bit strings (BYTE, WORD, DWORD, LWORD).
- When the value of "Size" is 0, the function returns a value of True, and the value of AryOut[] remains unchanged.
- When the value of In[] exceeds the valid range, the function returns a value of FALSE, and the value of AryOut[]

remains unchanged.

- When the value of "Size" exceeds the array area of In[] or AryOut[], it is necessary to avoid PLC exceptions.
- The data types of In[] and AryOut[] should be consistent, otherwise the return value will return FALSE, while the value of AryOut[] remains unchanged.

(5) Program example

| DEMO | LD | ST |
|---|---|---|
| Define variable | VAR<br>WO_IN: ARRAY[0..5] OF WORD:= [16#9,16#16,16#26,16#168,16#12,16#512];<br>UIN_S: UINT:= 3;<br>WO_OUT: ARRAY[0..5] OF WORD;<br>XDONE: BOOL;<br>END_VAR | |
| Program example |  | XDONE:=AryToBin(<br>In:= WO_IN[1],<br>Size:= UIN_S,<br>AryOut:= WO_OUT[3]); |
| Operation result |  | |

| 表达式 | 类型 | 值 |
|---|---|---|
| WO_IN | ARRAY [0..5] OF WO... | |
| WO_IN[0] | WORD | 16#0009 |
| WO_IN[1] | WORD | 16#0016 |
| WO_IN[2] | WORD | 16#0026 |
| WO_IN[3] | WORD | 16#0168 |
| WO_IN[4] | WORD | 16#0012 |
| WO_IN[5] | WORD | 16#0512 |
| UIN_S | UINT | 16#0003 |
| WO_OUT | ARRAY [0..5] OF WO... | |
| WO_OUT[0] | WORD | 16#0000 |
| WO_OUT[1] | WORD | 16#0000 |
| WO_OUT[2] | WORD | 16#0000 |
| WO_OUT[3] | WORD | 16#0010 |
| WO_OUT[4] | WORD | 16#001A |
| WO_OUT[5] | WORD | 16#00A8 |
| XDONE | BOOL | TRUE |

1-4-7-22. Bit string conver to Gray code BinToGray_XXX

(1) Instruction overview

Bit string conver to Gray code.

| Bit string conver to Gray code [BinToGray_XXX] | | | |
|---|---|---|---|
| Execution condition | Edge triggering | Suitable model | XS, XSA |
| Firmware | - | Software | XS StudioV2.2.0 and above |
| Library | XJ_DataUtils | | |

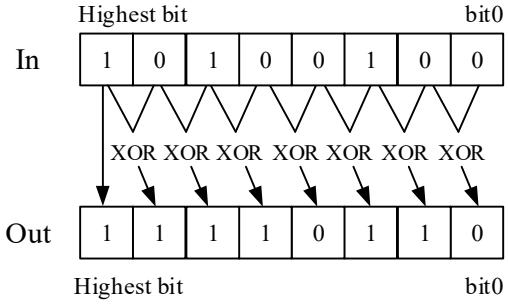
| Instruction | Name | Graph | ST |
|---|---|---|---|
| BinToGray_XXX | Bit string conver to Gray code | | xDone:=BinToGray_BYTE(In:= , Out=> ); |

(2) Related variables

| VAR_INPUT | Name | Data type | Valid range | Initial value | Description |
|---|---|---|---|---|---|
| In | Convert object | BYTE/WORD/DWORD/LWORD | As the data type | 0 | Bit string |
| VAR_OUTPUT | Name | Data type | Valid range | Initial value | Description |
| Out | Convert result | Consistent with In | As the data type | - | Gray code |

(3) Function

- Convert the bit string object "In" to Gray code.
- The instruction name varies depending on the data type of "In" and "Out". For example, when "In" and "Out" are WORD types, the instruction name is BinToGray_WORD.
  - Taking "In" and "Out" as BYTE types as an example, explain the conversion steps below:
    (1) The highest bit (7th bit) of "In" is directly used as the highest bit (7th bit) of "Out".
    (2) The XOR of the 7th value of "In" and the 6th value of "In" is the 6th value of "Out".
    (3) Using the same method, determine the value to the lowest (0th) position of "Out".




(4) Note

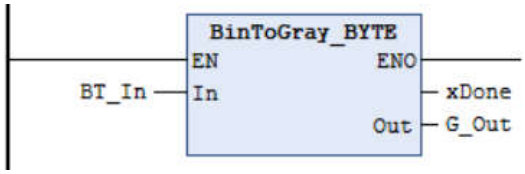
- It is recommended to unify the data types for "In" and "Out".

(5) Program example

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

```
VAR
   BT_In: BYTE;
   G_Out: BYTE;
   xDone: BOOL;
END_VAR
```

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




</td><td>

```
xDone:=BinToGray_BYTE(
    In:= BT_In,
    Out=> G_Out);
```

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

| 表达式 | 类型 | 值 |
|---|---|---|
| BT_In | BYTE | 2#10010110 |
| G_Out | BYTE | 2#11011101 |
| xDone | BOOL | TRUE |

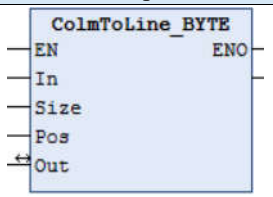
</td></tr>
</table>

1-4-7-23. Bit train/bit line conver to array ColmToLine_XXX

(1) Instruction overview

Extract the value of the bits at the specified position of each array element and output it as a bit string.

| Bit train/bit line conver to array [ColmToLine_XXX] | | | |
|---|---|---|---|
| Execution condition | Edge triggering | Suitable model | XS, XSA |
| Firmware | - | Software | XS StudioV2.2.0 and above |
| Library | XJ_DataUtils | | |

| Instruction | Name | Graph | ST |
|---|---|---|---|
| ColmToLine_XXX | Bit train/bit line conver to array |  | xDone:=ColmToLine_BYTE(In:= ,<br>Size:= ,<br>Pos:= ,<br>Out:= ); |

(2) Related variables

| VAR_INPUT | Name | Data type | Valid range | Initial value | Description |
|---|---|---|---|---|---|
| In | Convert object array | BYTE/WORD/ DWORD/LWORD | As the data type | - | Convert object array |
| Size | Number of elements for converting objects | USINT | As the data type | 1 | The number of elements in In[] that serve as conversion objects |
| Pos | The position to be converted | USINT | As the data type | 0 | The position to be converted |
| VAR_IN_OUT | Name | Data type | Valid range | Initial value | Description |
| Out | Convert result | BYTE/WORD/ DWORD/LWORD | As the data type | - | Convert result |

(3) Function

- Extract the values of the bits at the specified positions of each element in the array, arrange them, and output them as bit strings.
- Firstly, extract the "Size" elements of the transformation object array In[] starting from the input array element In[]. Then, only extract the value of the "Pos" position of the above elements. Arrange it into a bit string of "Size" bits and save it in the lowest bit position of the conversion result "Out". Save FALSE in the remaining bits of "Out".
- The instruction name varies depending on the data type of "Out". For example, when "Out" is of BYTE type, the instruction name is ColmToLine_BYTE.

(4) Note

- When the value of "Size" is 0, the function returns a value of True, and all bits of "Out" are FALSE.
- When the value of "Size" exceeds the valid range, the function returns a value of FALSE and "Out" remains unchanged.
- When the value of "Pos" exceeds the valid range, the function returns a value of FALSE and "Out" remains

unchanged.

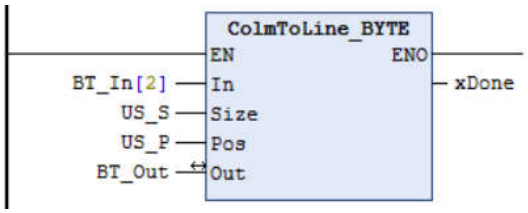
- When the value of "Size" exceeds the array area of In[], it is necessary to avoid PLC exceptions.
- When the data type passed in by "In[]" does not belong to the bit string range, an error occurs during compilation.

(5) Program example

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

```
VAR
   BT_In: ARRAY[0..5] OF BYTE:= [82,176,13,47,125,34];
   US_S: USINT:= 4;
   US_P: USINT:= 1;
   BT_Out: BYTE;
   xDone: BOOL;
END_VAR
```

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




</td><td>

```
xDone:=ColmToLine_BYTE(
    In:= BT_In[2],
    Size:= US_S,
    Pos:= US_P,
    Out:= BT_Out);
```

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

| 表达式 | 类型 | 值 |
|---|---|---|
| BT_In | ARRAY [0..5] OF BYTE | |
| BT_In[0] | BYTE | 2#01010010 |
| BT_In[1] | BYTE | 2#10110000 |
| BT_In[2] | BYTE | 2#00001101 |
| BT_In[3] | BYTE | 2#00101111 |
| BT_In[4] | BYTE | 2#01111101 |
| BT_In[5] | BYTE | 2#00100010 |
| US_S | USINT | 2#00000100 |
| US_P | USINT | 2#00000001 |
| BT_Out | BYTE | 2#00001010 |
| xDone | BOOL | TRUE |

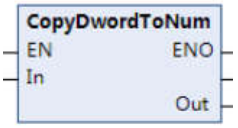
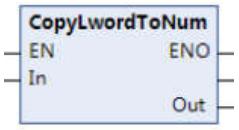
</td></tr>
</table>

1-4-7-24. Copy bit mode CopyByteToNum/CopyWordToNum/CopyDwordToNum/ CopyLwordToNum

(1) Instruction overview

Copy the content of the bit string directly into a signed integer.

| Copy bit mode [CopyByteToNum/CopyWordToNum/CopyDwordToNum/ CopyLwordToNum] | | | |
|---|---|---|---|
| Execution condition | Edge triggering | Suitable model | XS, XSA |
| Firmware | - | Software | XS StudioV2.2.0 and above |
| Library | XJ_DataUtils | | |

| Instruction | Name | Graph | ST |
|---|---|---|---|
| CopyByteToNum | Copy bit mode | | XDONE:=CopyByteToNum(In:= , Out=> ); |
| CopyWordToNum | Copy bit mode | | XDONE:=CopyWordToNum(In:= , Out=> ); |
| CopyDwordToNum | Copy bit mode |  | XDONE:=CopyDwordToNum(In:= , Out=> ); |
| CopyLwordToNum | Copy bit mode |  | XDONE:=CopyLwordToNum(In:= , Out=> ); |

(2) Related variables

| VAR_INPUT | Name | Data type | Valid range | Initial value | Description |
|---|---|---|---|---|---|
| In | Copy source | BYTE/WORD/DWORD/LWORD | As the data type | 0 | Copy source |
| VAR_OUTPUT | Name | Data type | Valid range | Initial value | Description |
| Out | Paste target | SINT/INT/DINT/LINT | As the data type | （*） | Paste target |

(3) Function

- Copy the content of the source without any processing, and directly copy it to the paste target "Out".
- There are four types of instruction names due to different combinations of data types for "In" and "Out".

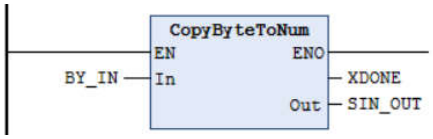
| “In” | “Out” | Command name |
|---|---|---|
| BYTE | SINT | CopyByteToNum |
| WORD | INT | CopyWordToNum |
| DWORD | DINT | CopyDwordToNum |
| LWORD | LINT | CopyLwordToNum |

(4) Program example

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

```
VAR
   BY_IN: BYTE:= 20;
   XDONE: BOOL;
   SIN_OUT: SINT;
END_VAR
```

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




</td><td>

```
XDONE:=CopyByteToNum(
    In:= BY_IN,
    Out=> SIN_OUT);
```

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

| 表达式 | 类型 | 值 |
|---|---|---|
| BY_IN | BYTE | 2#00010100 |
| XDONE | BOOL | TRUE |
| SIN_OUT | SINT | 2#00010100 |

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

1-4-7-25. Copy bit mode CopyDwordToReal/CopyLwordToLreal

(1) Instruction overview

Copy the content of the bit string directly into the real number.

| Copy bit mode [CopyDwordToReal/CopyLwordToLreal] | | | |
|---|---|---|---|
| Execution condition | Edge triggering | Suitable model | XS, XSA |
| Firmware | - | Software | XS StudioV2.2.0 and above |
| Library | XJ_DataUtils | | |

| Instruction | Name | Graph | ST |
|---|---|---|---|
| CopyDwordToReal | Copy bit mode | | XDONE:=CopyDwordToReal(In:= , Out=> ); |
| CopyLwordToLreal | Copy bit mode | | XDONE:=CopyLwordToLreal(In:= , Out=> ); |

(2) Related variables

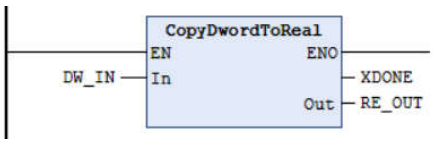
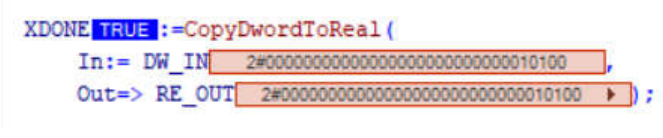
| VAR_INPUT | Name | Data type | Valid range | Initial value | Description |
|---|---|---|---|---|---|
| In | Copy source | DWORD/LWORD | As the data type | 0 | Copy source |
| VAR_OUTPUT | Name | Data type | Valid range | Initial value | Description |
| Out | Paste target | REAL/LREAL | As the data type | (*) | Paste target |

(3) Function

- Copy the content of the source without any processing, and directly copy it to the paste target "Out".
- There are two types of instruction names due to different combinations of data types for "In" and "Out".

| “In” | “Out” | Command name |
|---|---|---|
| DWORD | REAL | CopyDwordToReal |
| LWORD | LREAL | CopyLwordToLreal |

(4) Program example

| DEMO | LD | ST |
|---|---|---|
| Define variable | | VAR<br>DW_IN: DWORD:= 20;<br>XDONE: BOOL;<br>RE_OUT: REAL;<br>END_VAR |
| Program example |  | XDONE:=CopyDwordToReal(<br>In:= DW_IN,<br>Out=> RE_OUT); |
| Operation result |  | |

1-4-7-26. Copy bit mode CopyRealToDWord/CopyLrealToLword

(1) Instruction overview

Copy the content of real numbers directly into the bit string.

| Copy bit mode [CopyRealToDWord/CopyLrealToLword] | | | |
|---|---|---|---|
| Execution condition | Edge triggering | Suitable model | XS, XSA |
| Firmware | - | Software | XS StudioV2.2.0 and above |
| Library | XJ_DataUtils | | |

| Instruction | Name | Graph | ST |
|---|---|---|---|
| CopyRealToDWord | Copy bit mode | | XDONE:=CopyRealToDWord(In:= , Out=> ); |
| CopyLrealToLword | Copy bit mode | | XDONE:=CopyLrealToLword(In:= , Out=> ); |

(2) Related variables

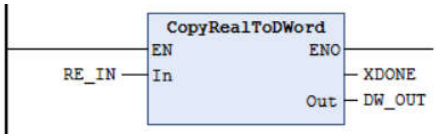
| VAR_INPUT | Name | Data type | Valid range | Initial value | Description |
|---|---|---|---|---|---|
| In | Copy source | REAL/LREAL | As the data type | 0 | Copy source |
| VAR_OUTPUT | Name | Data type | Valid range | Initial value | Description |
| Out | Paste target | DWORD/LWORD | As the data type | （*） | Paste target |

(3) Function

- Copy the content of the source without any processing, and directly copy it to the paste target "Out".
- There are two types of instruction names due to different combinations of data types for "In" and "Out".

| “In” | “Out” | Command name |
|---|---|---|
| REAL | DWORD | CopyRealToDWord |
| LREAL | LWORD | CopyLrealToLword |

(4) Program example

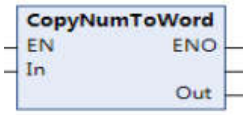
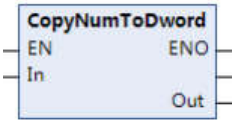
| DEMO | LD | ST |
|---|---|---|
| Define variable | | VAR<br>RE_IN: REAL:= 8.9;<br>DW_OUT: DWORD;<br>XDONE: BOOL;<br>END_VAR |
| Program example |  | XDONE:=CopyRealToDWord(<br>In:= RE_IN,<br>Out=> DW_OUT); |
| Operation result | XDONE TRUE :=CopyRealToDWord(<br>In:= RE_IN 2#01000001000011100110011001100110 ,<br>Out=> DW_OUT 2#01000001000011100110011001100110 ); | |

1-4-7-27. Copy bit mode CopyNumToXXX

(1) Instruction overview

Copy the contents of signed integers directly into real numbers.

| Copy bit mode [CopyNumToXXX] | | | |
|---|---|---|---|
| Execution condition | Edge triggering | Suitable model | XS, XSA |
| Firmware | - | Software | XS StudioV2.2.0 and above |
| Library | XJ_DataUtils | | |

| Instruction | Name | Graph | ST |
|---|---|---|---|
| CopyNumToByte | Copy bit mode | | XDONE:=CopyNumToByte(In:= , Out=> ); |
| CopyNumToWord | Copy bit mode |  | XDONE:=CopyNumToWord(In:= , Out=> ); |
| CopyNumToDword | Copy bit mode |  | XDONE:=CopyNumToDword(In:= , Out=> ); |
| CopyNumToLword | Copy bit mode | | XDONE:=CopyNumToLword(In:= , Out=> ); |

(2) Related variables

| VAR_INPUT | Name | Data type | Valid range | Initial value | Description |
|---|---|---|---|---|---|
| In | Copy source | SINT/INT/DINT/LINT | As the data type | 0 | Copy source |
| VAR_OUTPUT | Name | Data type | Valid range | Initial value | Description |
| Out | Paste target | BYTE/WORD/DWORD/LWORD | As the data type | （*） | Paste target |

(3) Function

- Copy the content of the source without any processing, and directly copy it to the paste target "Out".
- There are four types of instruction names due to different combinations of data types for "In" and "Out".

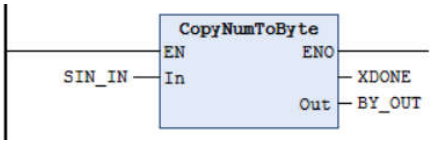
| “In” | “Out” | Command name |
|---|---|---|
| SINT | BYTE | CopyNumToByte |
| INT | WORD | CopyNumToWord |
| DINT | DWORD | CopyNumToDword |
| LINT | LWORD | CopyNumToLword |

(4) Program example

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

```
VAR
   SIN_IN: SINT:= 20;
   BY_OUT: BYTE;
   XDONE: BOOL;
END_VAR
```

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




</td><td>

```
XDONE:=CopyNumToByte(
    In:= SIN_IN,
    Out=> BY_OUT);
```

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

| 表达式 | 类型 | 值 |
|---|---|---|
| SIN_IN | SINT | 2#00010100 |
| BY_OUT | BYTE | 2#00010100 |
| XDONE | BOOL | TRUE |

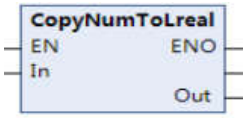
</td></tr>
</table>

1-4-7-28. Copy bit mode CopyNumToReal/CopyNumToLreal

(1) Instruction overview

Copy the contents of signed integers directly into real numbers.

| Copy bit mode [CopyNumToReal/CopyNumToLreal] | | | |
|---|---|---|---|
| Execution condition | Edge triggering | Suitable model | XS, XSA |
| Firmware | - | Software | XS StudioV2.2.0 and above |
| Library | XJ_DataUtils | | |

| Instruction | Name | Graph | ST |
|---|---|---|---|
| CopyNumToReal | Copy bit mode | | XDONE:=CopyNumToReal(In:= , Out=> ); |
| CopyNumToLreal | Copy bit mode |  | XDONE:=CopyNumToLreal(In:= , Out=> ); |

(2) Related variables

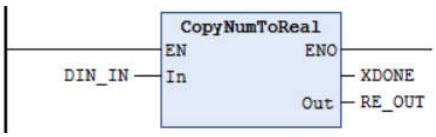
| VAR_INPUT | Name | Data type | Valid range | Initial value | Description |
|---|---|---|---|---|---|
| In | Copy source | DINT/LINT | As the data type | 0 | Copy source |
| VAR_OUTPUT | Name | Data type | Valid range | Initial value | Description |
| Out | Paste target | REAL/LREAL | As the data type | （*） | Paste target |

(3) Function

- Copy the content of the source without any processing, and directly copy it to the paste target "Out".
- There are two types of instruction names due to different combinations of data types for "In" and "Out".

| “In” | “Out” | Command name |
|---|---|---|
| DINT | REAL | CopyNumToReal |
| LINT | LREAL | CopyNumToLreal |

(4) Program example

| DEMO | LD | ST |
|---|---|---|
| Define variable | | VAR<br>DIN_IN: DINT:= 20;<br>RE_OUT: REAL;<br>XDONE: BOOL;<br>END_VAR |
| Program example |  | XDONE:=CopyNumToReal(<br>In:= DIN_IN,<br>Out=> RE_OUT); |
| Operation result | XDONE TRUE :=CopyNumToReal(<br>In:= DIN_IN 2#00000000000000000000000000010100 ,<br>Out=> RE_OUT 2#00000000000000000000000000010100 ); | |

1-4-7-29. Copy bit mode CopyRealToNum/CopyLrealToNum

(1) Instruction overview

Copy the contents of real numbers directly into integers.

| Copy bit mode [CopyRealToNum/CopyLrealToNum] | | | |
|---|---|---|---|
| Execution condition | Edge triggering | Suitable model | XS, XSA |
| Firmware | - | Software | XS StudioV2.2.0 and above |
| Library | XJ_DataUtils | | |

| Instruction | Name | Graph | ST |
|---|---|---|---|
| CopyRealToNum | Copy bit mode | | XDONE:=CopyRealToNum(In:= , Out=> ); |
| CopyLrealToNum | Copy bit mode | | XDONE:=CopyLrealToNum(In:= , Out=> ); |

(2) Related variables

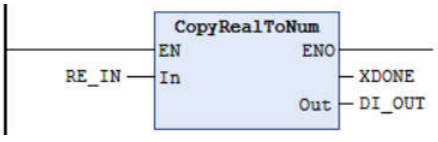
| VAR_INPUT | Name | Data type | Valid range | Initial value | Description |
|---|---|---|---|---|---|
| In | Copy source | REAL/LREAL | As the data type | 0 | Copy source |
| VAR_OUTPUT | Name | Data type | Valid range | Initial value | Description |
| Out | Paste target | DINT/LINT | As the data type | （*） | Paste target |

(3) Function

- Copy the content of the source without any processing, and directly copy it to the paste target "Out".
- There are two types of instruction names due to different combinations of data types for "In" and "Out".

| “In” | “Out” | Command name |
|---|---|---|
| REAL | DINT | CopyRealToNum |
| LREAL | LINT | CopyLrealToNum |

(4) Program example

| DEMO | LD | ST |
|---|---|---|
| Define variable | | VAR<br>RE_IN: REAL:= 8.8;<br>DI_OUT: DINT;<br>XDONE: BOOL;<br>END_VAR |
| Program example |  | XDONE:=CopyRealToNum(<br>In:= RE_IN,<br>Out=> DI_OUT); |
| Operation result | XDONE TRUE :=CopyRealToNum(<br>In:= RE_IN 2#01000001000011001100110011001101 ,<br>Out=> DI_OUT 2#01000001000011001100110011001101 ); | |

1-4-7-30. Gray code convert to bit string GrayToBin_XXX

(1) Instruction overview

Convert Gray code to bit string.

| Gray code convert to bit string [GrayToBin_XXX] | | | |
|---|---|---|---|
| Execution condition | Edge triggering | Suitable model | XS, XSA |
| Firmware | - | Software | XS StudioV2.2.0 and above |
| Library | XJ_DataUtils | | |

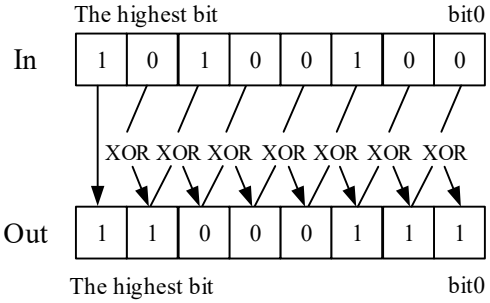
| Instruction | Name | Graph | ST |
|---|---|---|---|
| GrayToBin_XXX | Gray code convert to bit string | | XDONE:=GrayToBin_BYTE(In:= , Out=> ); |

(2) Related variables

| VAR_INPUT | Name | Data type | Valid range | Initial value | Description |
|---|---|---|---|---|---|
| In | Convert object | BYTE/WORD/DWORD/LWORD | As the data type | 0 | Gray code |
| VAR_OUTPUT | Name | Data type | Valid range | Initial value | Description |
| Out | Convert result | Consistent with In | As the data type | - | Bit string |

(3) Function

- Convert the Gray code conversion object "In" to a bit string.
- The instruction name varies depending on the data type of "In" and "Out". For example, when "In" and "Out" are WORD types, the instruction name is GrayToBin_WORD.
- Taking "In" and "Out" as BYTE types as an example, explain the conversion steps below:
  - ① The highest bit (7th bit) of "In" is directly used as the highest bit (7th bit) of "Out";
  - ② The XOR of the 6th value of "In" and the 7th value of "Out" is the 6th value of "Out";
  - ③ Determine the value to the lowest (0th) position of "Out" using the same method.




(4) Note

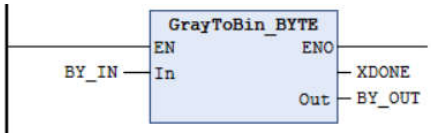
- It is recommended to unify the data types for "In" and "Out".

(5) Program example

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

```
VAR
   BY_IN: BYTE:= 2#10011101;
   BY_OUT: BYTE;
   XDONE: BOOL;
END_VAR
```

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




</td><td>

```
XDONE:=GrayToBin_BYTE(
    In:= BY_IN,
    Out=> BY_OUT);
```

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

| 表达式 | 类型 | 值 |
|---|---|---|
| BY_IN | BYTE | 2#10011101 |
| BY_OUT | BYTE | 2#11101001 |
| XDONE | BOOL | TRUE |

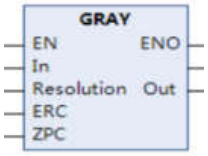
</td></tr>
</table>

1-4-7-31. Gray code convert to angle GRAY

(1) Instruction overview

Convert Gray code to angle.

| Gray code convert to angle [Gray] | | | |
|---|---|---|---|
| Execution condition | Edge triggering | Suitable model | XS, XSA |
| Firmware | - | Software | XS StudioV2.2.0 and above |
| Library | XJ_DataUtils | | |

| Instruction | Name | Graph | ST |
|---|---|---|---|
| GRAY | Gray code convert to angle |  | XDONE:=GRAY(In:= ,<br>Resolution:= ,<br>ERC:= ,<br>ZPC:= ,<br>Out=> ); |

(2) Related variables

| VAR_INPUT | Name | Data type | Valid range | Initial value | Description |
|---|---|---|---|---|---|
| In | Convert object | WORD | As the data type | 0 | Gray code |
| Resolution | Resolution | _eGRY_RESOLUTION | As the data type | _R256 | Resolution |
| ERC | Encoder residual correction value | UINT | As the data type | 0 | Encoder residual correction value |
| ZPC | Origin correction value | UINT | As the data type | 0 | Origin correction value |
| VAR_OUTPUT | Name | Data type | Valid range | Initial value | Description |
| Out | Convert result | LREAL | As the data type | - | Convert result |

(3) Function

- Convert the output value "In" of the rotary encoder represented by Gray code to an angle value. The unit of conversion result "Out" is "degrees".
- The data type of "Resolution" is enumeration type _eGRY_RESOLUTION. The meaning of enumeration values is as follows:

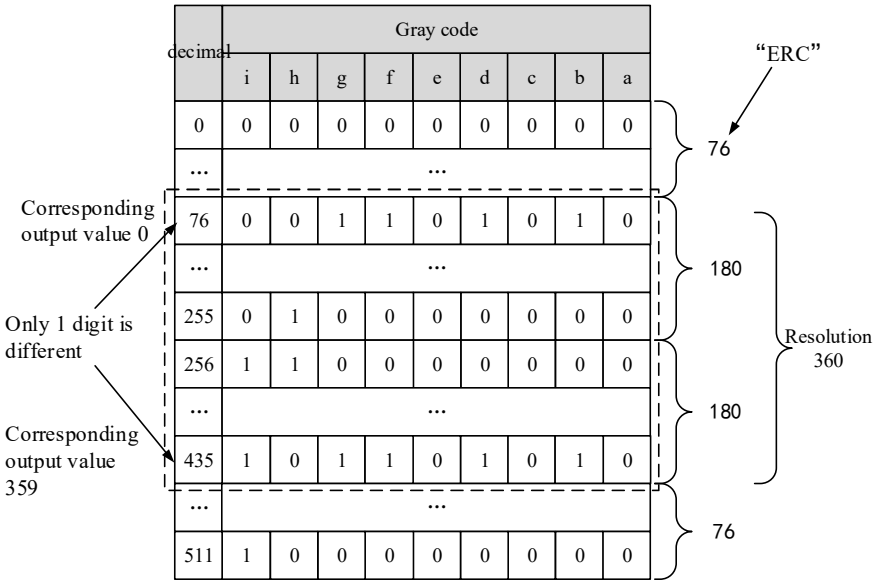
| Enum | Meaning | Enum | Meaning |
|---|---|---|---|
| _R256 | 256 | _R10B | 10 bits (1024) |
| _R1B | 1 bit (2) | _R11B | 11 bits (2048) |
| _R2B | 2 bits (4) | _R12B | 12 bits (4096) |
| _R3B | 3 bits (8) | _R13B | 13 bits (8192) |
| _R4B | 4 bits (16) | _R14B | 14 bits (16384) |
| _R5B | 5 bits (32) | _R15B | 15 bits (32768) |
| _R6B | 6 bits (64) | _R360 | 360 |
| _R7B | 7 bits (128) | _R720 | 720 |

| | | | |
|---|---|---|---|
| _R8B | 8 bits (256) | _R1024 | 1024 |
| _R9B | 9 bits (512) | | |

- Gray code refers to a type of alternating binary code. Its characteristic is that any two adjacent codes have only one binary digit different.
- Gray code is commonly used for the output of absolute encoders, etc.
- The following are 4-bit binary codes and Gray codes.

| 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 |
| 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 |

- If Gray code is used, when the output value of the encoder increases or decreases by 1, only one bit of the value changes, thus preventing the output of incorrect values. The encoder residual correction value "ERC" is the correction value specified within the Gray code range to ensure that the maximum and minimum output values of the encoder are only one bit different in the Gray code when the encoder resolution is not a power of 2.
- For example, using an absolute encoder with a resolution of 360. The number of digits in the Gray code is 9. The range of values that can be represented by 9 bits is 0-511. At this point, within a range of 180 degrees from the center of 0-511, i.e. 76-435, use the Gray code. Therefore, the Gray code representing output value 0 is 001101010 (decimal number 76), and the Gray code representing output value 359 is 101101010 (decimal number 435), with only one difference between the two.
- At this point, the value of the encoder residual correction value "ERC" is 76.




- The origin correction value "ZPC" is set when it is necessary to offset the origin angle of the rotary encoder. For example, when the origin of a rotary encoder with a resolution of 256 is offset by 90°, the value of "ZPC" is 256 x (90/360)=64.
- When "In"=WORD#16#256, "Resolution"=_R10B, "ERC"=UINT#0, "ZPC"=UINT#2, the examples are as the following:

  Firstly, due to the resolution of 10 bits, one unit of the Gray code is 360/1024=0.3515625°.

  The value of the decimal number corresponding to Gray code 16#256 is 923. Therefore, the angle value before correction is 0.3515625°×923=324.492188°

  The value of "ERC" is 0, and the value of "ZPC" is 2. Therefore, the corrected angle value is 324.492188°-(0+2) ×0.3515625°=323.7890625°

  The value of "Out" is LREAL #323.7890625.

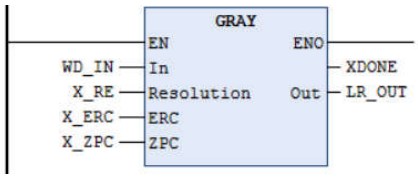
(4) Note

An exception will occur in the following situations. The return value of the function becomes FALSE, and "Out" is initialized to 0.

- When the value of "ERC" is greater than the resolution specified in "Resolution".
- When the value of "ZPC" is greater than the resolution specified in "Resolution".
- When the value after converting "In" to a rank is less than the value of "ERC".
- When the value corrected by "ERC" is greater than the resolution specified in "Resolution".

(5) Program example

| DEMO | LD | ST |
|---|---|---|
| Define variable | VAR<br>WD_IN: WORD:= 16#256;<br>X_RE: _eGRY_RESOLUTION:= _eGRY_RESOLUTION._R10B;<br>X_ERC: UINT:= 0;<br>X_ZPC: UINT:= 2;<br>LR_OUT: LREAL;<br>XDONE: BOOL;<br>END_VAR | |

| | | |
|---|---|---|
| Program example |  | XDONE:=GRAY(<br>In:= WD_IN,<br>Resolution:= X_RE,<br>ERC:= X_ERC,<br>ZPC:= X_ZPC,<br>Out=> LR_OUT); |
| Operation result | 表达式 / 类型 / 值<br>WD_IN / WORD / 598<br>X_RE / _EGRY_RESOLUTION / _R10B<br>X_ERC / UINT / 0<br>X_ZPC / UINT / 2<br>LR_OUT / LREAL / 323.7890625<br>XDONE / BOOL / TRUE | |

1-4-7-32. Convert fixed decimal point number to decimal strings FixNumToString

(1) Instruction overview

Convert signed fixed decimal points to decimal string format.

| Convert fixed decimal point number to decimal strings [FixNumToString] | | | |
|---|---|---|---|
| Execution condition | Edge triggering | Suitable model | XS, XSA |
| Firmware | XS: V2.2.0 and up<br>XSA: - | Software | XS StudioV2.2.0 and above |
| Library | XJ_DataUtils | | |

| Instruction | Name | Graph | ST |
|---|---|---|---|
| FixNumToString | Convert fixed decimal point number to decimal strings | | XDONE:=FixNumToString(In:= ,<br>Zero:= ,<br>Out:= ); |

(2) Related variables

| VAR_INPUT | Name | Data type | Valid range | Initial value | Description |
|---|---|---|---|---|---|
| In | Convert object | DWORD | As the data type | 0 | Signed fixed decimal point number |
| Zero | Zero display | BOOL | TRUE/FALSE | FALSE | Display when the number of decimal places is less than 3<br>True: Fill in "0"<br>FALSE: Do not add "0" |
| VAR_IN_OUT | Name | Data type | Valid range | Initial value | Description |
| Out | Convert result | STRING | As the data type | - | 10 base string |

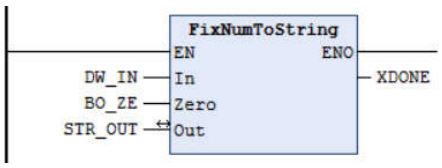
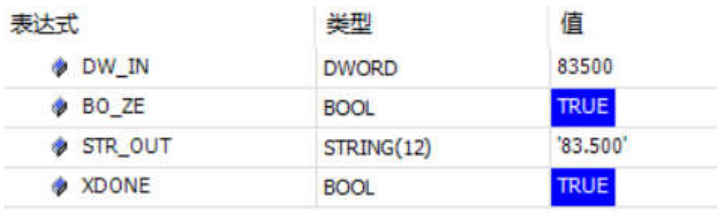
(3) Function

- Convert the signed fixed decimal point number "In" to a decimal string. The conversion steps are as follows:
  ① Convert "In" in hexadecimal format to decimal format.
  ② Divide the value by 1000.
- When the number of decimal places of "In" is less than 3, zero is displayed. "Zero" specifies whether to fill in "0" before the 3rd decimal place of "Out". If the value of "Zero" is true, then fill in "0"; If it is FALSE, do not fill in "0".
- "Out" ends with a NULL character.

(4) Note

- When the conversion result exceeds the valid range of "Out", the function returns a value of FALSE and "Out" remains unchanged.

(5) Program example

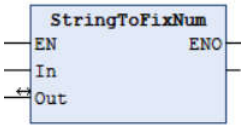
| DEMO | LD | ST |
|---|---|---|
| Define variable | | VAR<br>DW_IN: DWORD:= 16#1462C;<br>BO_ZE: BOOL:= TRUE;<br>STR_OUT: STRING(12);<br>XDONE: BOOL;<br>END_VAR |
| Program example |  | XDONE:=FixNumToString(<br>In:= DW_IN,<br>Zero:= BO_ZE,<br>Out:= STR_OUT); |
| Operation result |  | |

1-4-7-33. String convert to fixed decimal point number StringToFixNum

(1) Instruction overview

Convert a decimal string to a signed fixed decimal point format.

| String convert to fixed decimal point number [StringToFixNum] | | | |
|---|---|---|---|
| Execution condition | Edge triggering | Suitable model | XS, XSA |
| Firmware | - | Software | - |
| Library | XJ_DataUtils | | |

| Instruction | Name | Graph | ST |
|---|---|---|---|
| StringToFixNum | String convert to fixed decimal point number |  | XDONE:=StringToFixNum(In:= ,<br>Out:= ); |

(2) Related variables

| VAR_INPUT | Name | Data type | Valid range | Initial value | Description |
|---|---|---|---|---|---|
| In | Convert object | STRING | As the data type | 0 | 10 base string |
| VAR_IN_OUT | Name | Data type | Valid range | Initial value | Description |
| Out | Convert result | DWORD | As the data type | - | Fixed decimal places number |

(3) Function

- Convert the decimal string "In" to a fixed number of decimal points. The conversion steps are as follows:
  ① Multiply the value represented by "In" by 1000.

② Round off the decimal point of this value.

③ Convert the value to 32-bit hexadecimal format (DWORD type).

| Name | Format |
|---|---|
| Symbol part | ◆ Ignore a single '+' or '-' that appears after the consecutive ' ' (spaces 16#20) that appear at the beginning of the string.<br>◆ Compliance can be omitted.<br>◆ Simultaneously ignore consecutive spaces after symbols. |
| Integer part | ◆ Take the consecutive numerical values ('0' to '9') from the symbol to the decimal point as the integer part. Sometimes symbols are also omitted. Sometimes there may be consecutive spaces between the symbol and integer parts.<br>◆ When there is no decimal point or decimal part, the exponential part is preceded by the integer part.<br>◆ When there is no decimal point, decimal and exponential parts, the integer part is until the end of the string.<br>◆ The integer part cannot be omitted.<br>◆ The maximum number of digits in the integer part is the maximum string length of 1986 subtracting the symbol, decimal point, decimal part, exponential part, and total bytes of space before and after the symbol. |
| Decimal point | ◆ The single '.' after the integer part (dot) is a decimal point.<br>◆ When there is no decimal part, please omit the decimal point. |
| Decimal part | ◆ Take the consecutive numerical values ('0' to '9') from the decimal point to the exponential part as the decimal part.<br>◆ When there is no exponential part, it is the decimal part until the end of the string.<br>◆ Decimal parts can be omitted. When there is no decimal point, there is also no decimal part.<br>◆ The maximum number of decimal places is 15. |
| Exponent part | ◆ The single 'e' or 'E' after the decimal part, and the consecutive numerical values ('0' to '9') from the single '+' or ' - ' to the end of the string are the exponential part.<br>◆ When there is no decimal part, the string after the decimal point is the exponential part.<br>◆ When there is no decimal point or decimal part, the above string after the integer is the exponential part.<br>◆ The exponential part can be omitted.<br>◆ The maximum number of digits in the exponential part is 3. |

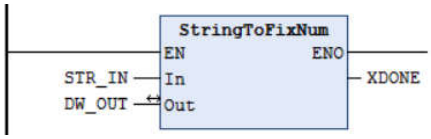
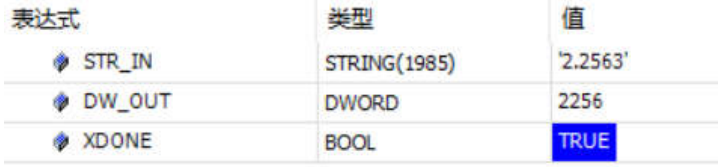
(4) Note

- Round off after 4 decimal places for "In".
- Ignoring the underline (16#5F) in the "In" string.
- When "In" does not end with a NULL character, the function returns a value of FALSE while "Out" remains unchanged.
- When the content of "In" is a character that cannot be converted to a numerical value, the function returns a value of FALSE while "Out" remains unchanged.
- When the content of "In" has a decimal point but no decimal part, the function returns a value of FALSE while "Out" remains unchanged.
- When the conversion result exceeds the valid range of "Out", the function returns a value of FALSE and "Out"

remains unchanged.

- In the symbol section, the "+" sign can be omitted, and negative numbers are used for overflow processing. When two or more symbols appear, the runtime function returns FALSE.
- When there are spaces in the middle and at the end of the input data, the function returns a value of FALSE and "Out" remains unchanged.

(5) Progarm example

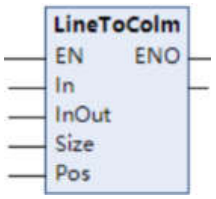
| DEMO | LD | ST |
|---|---|---|
| Define variable | VAR<br>STR_IN: STRING(1985);<br>DW_OUT: DWORD;<br>XDONE: BOOL;<br>END_VAR | |
| Program example |  | XDONE:=StringToFixNum(<br>In:= STR_IN,<br>Out:= DW_OUT); |
| Operation result |  | |

1-4-7-34. Row/column conversion LineToColm

(1) Instruction overview

Decompose the bit string and output it to the specified bit position of the array element.

| Row/column conversion [LineToColm] | | | |
|---|---|---|---|
| Execution condition | Edge triggering | Suitable model | XS, XSA |
| Firmware | - | Software | XS StudioV2.2.0 and above |
| Library | XJ_DataUtils | | |

| Instruction | Name | Graph | ST |
|---|---|---|---|
| LineToColm | Row/column conversion |  | XDONE:=LineToColm(In:= ,<br>InOut:= ,<br>Size:= ,<br>Pos:= ); |

(2) Related variables

| VAR_INPUT | Name | Data type | Valid range | Initial value | Description |
|---|---|---|---|---|---|
| In | Convert object | BYTE/WORD DWORD/LWORD | As the data type | - | Convert object |
| Size | The number of elements in the conversion result | UINT | As the data type | 1 | The number of elements in the conversion result |
| Pos | Conversion bit position | UINT | As the data type | 0 | The position to be converted |
| VAR_IN_OUT | Name | Data type | Valid range | Initial value | Description |
| InOut | Conversion result array | BYTE/WORD DWORD/LWORD | As the data type | - | Conversion result array |

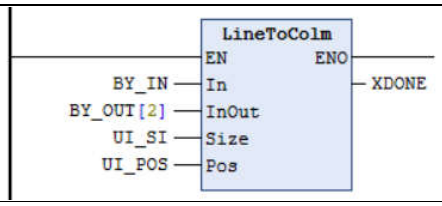
(3) Function

- Decompose the bit string and output it to the specified bit position of the array element.
- Firstly, extract the "Size" bit from the lowest bit of the transformation object "In" and decompose it into each bit.
- Then, save it in the "Pos" position of each element starting from InOut[0] in the conversion result array InOut[]. There are "Size" saved array elements.
- Unsaved value bits are used to maintain the value.

(4) Note

- When the value of "Size" is 0, the value of InOut[] remains unchanged, and the function returns a value of True.
- When the value of "Size" exceeds the valid range, the value of InOut[] remains unchanged, and the function returns a value of FALSE.
- When the value of "Pos" exceeds the valid range, the value of InOut[] remains unchanged, and the function returns a value of FALSE.
- When the value of "Size" exceeds the array area of InOut[], it is necessary to avoid PLC exceptions.

(5) Program example

| DEMO | LD | ST |
|---|---|---|
| Define variable | | VAR<br>BY_IN: BYTE:= 173;<br>BY_OUT: ARRAY[0..5] OF BYTE;<br>UI_SI: UINT:= 3;<br>UI_POS: UINT:= 1;<br>XDONE: BOOL;<br>END_VAR |
| Program example |  | XDONE:=LineToColm(<br>In:= BY_IN,<br>InOut:= BY_OUT[2],<br>Size:= UI_SI,<br>Pos:= UI_POS); |
| Operation result | | |

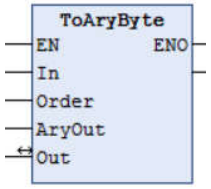
| 表达式 | 类型 | 值 |
|---|---|---|
| BY_IN | BYTE | 2#10101101 |
| BY_OUT | ARRAY [0..5] OF BYTE | |
| BY_OUT[0] | BYTE | 2#00000000 |
| BY_OUT[1] | BYTE | 2#00000000 |
| BY_OUT[2] | BYTE | 2#00000010 |
| BY_OUT[3] | BYTE | 2#00000000 |
| BY_OUT[4] | BYTE | 2#00000010 |
| BY_OUT[5] | BYTE | 2#00000000 |
| UI_SI | UINT | 2#0000000000000011 |
| UI_POS | UINT | 2#0000000000000001 |
| XDONE | BOOL | TRUE |

1-4-7-35. Convert to byte array ToAryByte

(1) Instruction overview

Divide variables by 1 byte and save them to a BYTE type array.

| Convert to byte array [ToAryByte] | | | |
|---|---|---|---|
| Execution condition | Edge triggering | Suitable model | XS, XSA |
| Firmware | - | Software | XS StudioV2.2.0 and above |
| Library | XJ_DataUtils | | |

| Instruction | Name | Graph | ST |
|---|---|---|---|
| ToAryByte | Convert to byte array |  | XDONE:=ToAryByte(In:= ,<br>Order:= ,<br>AryOut:= ,<br>Out:= ); |

(2) Related variables

| VAR_INPUT | Name | Data type | Valid range | Initial value | Description |
|---|---|---|---|---|---|
| In | Convert object | BOOL/ BYTE/WORD/DWORD/LWORD/ USINT/UINT/UDINT/ULINT/ SINT/INT/DINT/LINT/ REAL/LREAL/ TIME/DATE/TOD/DT | As the data type | - | Convert object |
| Order | Conversion order | _eBYTE_ORDER | As the data type | _LOW_HIGH | The position to be converted |
| VAR_IN_OUT | Name | Data type | Valid range | Initial value | Description |
| AryOut | Conversion result array | BYTE | As the data type | - | Conversion result array |
| Out | Conversion result element number | UINT | As the data type | - | Conversion result element number |

(3) Function

- Split the conversion object "In" in units of 1 byte and save it sequentially from the AryOut[] array of conversion results, starting from AryOut [0]. Save the number of elements already saved in AryOut[] to the number of elements in the conversion result "Out".
- The order in which the value of "In" is converted in units of 1 byte is specified by the conversion order "Order". The data type of "Order" is the enumerator_eBYTEE_ORDER; The meaning of enumeration elements is as follows:

| Enum | Meaning |
|---|---|
| _LOW_HIGH | Low byte before high byte |
| _HIGH_LOW | High byte before low byte |

- When the data type of "In" is more than 2 bytes in size; Divide "In" by 1 byte and save it in AryOut[]. The data types with a size of 2 bytes or more are as follows:

| Type | Data type |
|---|---|
| Bit string | WORD, DWORD, LWORD |
| Integer | UINT, UDINT, ULINT, INT, DINT, LINT |
| Real number | REAL, LREAL |
| Time, duration, date, string | TIME, DATE, TOD, TD |

- The processing steps are as follows:
  ① Firstly, divide the value of "In" in units of 1 word (2 bytes);
  ② Divide the word with the lowest bit by 1 byte (2 bytes);
  ③ If the value of "Order" is _LOW_HIGH, save the low byte in AryOut[0] and the high byte in AryOut[1]. If the value of "Order" is _HIGH_LOW, save the high byte in AryOut[0] and the low byte in AryOut[1].
  ④ Similarly, divide the next word by 1 byte and save it in AryOut[2] and AryOut[3]. Perform the same processing until the end of the In value.

- When the data type of "In" is 1 byte, it is still saved in AryOut[] in units of 1 byte. The data types with a size of 1 byte are as follows:

| Type | Data type |
|---|---|
| Bit string | BYTE |
| Integer | USINT, SINT |
| Real number | None |
| Time, duration, date, string | None |

- The storage method is as follows:

| “Order” value | Method to save to "AryOut[]" |
|---|---|
| LOW_HIGH | Save the value of "In" in AryOut[0] |
| HIGH_LOW | Save the value of "In" in AryOut[1]; Save 16#00 in AryOut[0]. |

- When the data type of "In" is BOOL (1 bit), the method to save to AryOut[] is as follows:

| “Order” value | Method to save to "AryOut[]" |
|---|---|
| LOW_HIGH | Save the value of "In" with the logical sum of 16#00 in AryOut[0] |
| HIGH_LOW | Connect the value of In[0] and save it in AryOut[0]; Connect the value of In[1], save it in AryOut[1], and then use the same method to save it. Be sure to set the value of "Out" to an even number. |

(4) Note

- Please make sure to set the input parameters transmitted to "In" as variables. If a constant is transmitted, an exception will occur during compilation.
- When the value of "Order" is _HIGH_LOW and the total size of "In" is odd bytes, add 16#00 at the end of "In", set it to even bytes, and then start the conversion.
- When the value of "Order" exceeds the valid range, the function returns FALSE, while "Out" and "AryOut[]

remain unchanged.

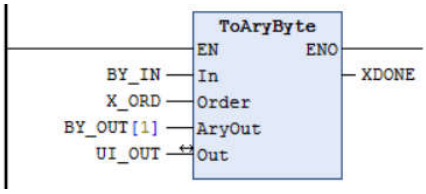
- When the conversion result exceeds the array area of AryOut[], it is necessary to avoid PLC exceptions.

(5) Program example

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

```
VAR
   BY_IN: DWORD:= 16#5121C168;
   X_ORD: _eBYTE_ORDER:=_eBYTE_ORDER._LOW_HIGH;
   BY_OUT: ARRAY[0..5] OF BYTE;
   UI_OUT: UINT;
   XDONE: BOOL;
END_VAR
```

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




</td><td>

```
XDONE:=ToAryByte(
    In:= BY_IN,
    Order:= X_ORD,
    AryOut:= BY_OUT[1],
    Out:= UI_OUT);
```

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

| 表达式 | 类型 | 值 |
|---|---|---|
| BY_IN | DWORD | 16#5121C168 |
| X_ORD | _EBYTE_ORDER | _LOW_HIGH |
| BY_OUT | ARRAY [0..5] OF BYTE | |
| BY_OUT[0] | BYTE | 16#00 |
| BY_OUT[1] | BYTE | 16#68 |
| BY_OUT[2] | BYTE | 16#C1 |
| BY_OUT[3] | BYTE | 16#21 |
| BY_OUT[4] | BYTE | 16#51 |
| BY_OUT[5] | BYTE | 16#00 |
| UI_OUT | UINT | 16#0004 |
| XDONE | BOOL | TRUE |

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

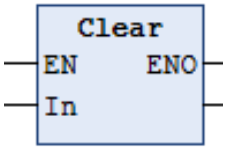
## 1-4-8. Initialization instruction

### 1-4-8-1. Initialization Clear

(1) Instruction overview

Initialize variables.

| Initialization [Clear] | | | |
|---|---|---|---|
| Execution condition | Edge triggering | Suitable model | XS, XSA |
| Firmware | - | Software | XS StudioV2.2.0 and above |
| Library | XJ_DataUtils | | |

| Instruction | Name | Graph | ST |
|---|---|---|---|
| Clear | Initialization |  | `Clear(In:= );` |

(2) Related variables

| VAR_IN_OUT | Name | Data type | Valid range | Initial value | Description |
|---|---|---|---|---|---|
| In | Initialize Object | BOOL/BYTE/WORD/DWORD/ LWORD/USINT/UINT/UDINT/ ULINT/SINT/INT/DINT/LINT/ REAL/LREAL/TIME/DATE/ TOD/DT/STRING | As the data type | - | Initialize Object |

(3) Function

- Initialize the value of the initialization object "In" to the default initial value of each data type.
- The default initial values for each data type are shown in the table below:

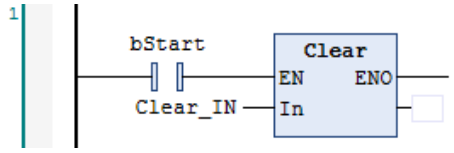
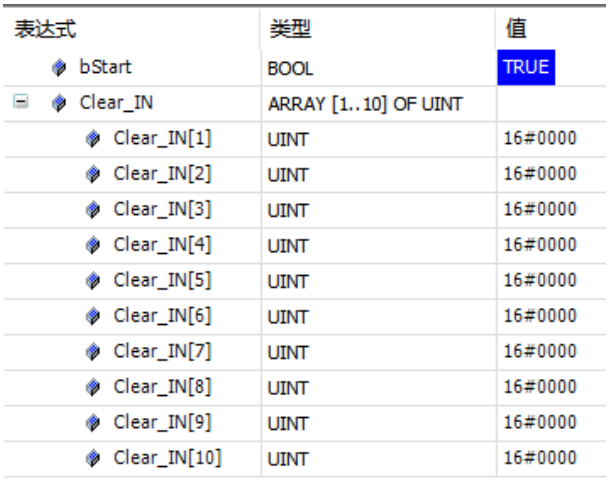
| Data type | Initial value |
|---|---|
| BOOL | FALSE |
| BYTE, WORD, DWORD, LWORD | 16#0 |
| USINT, UINT, UDINT, ULINT, SINT, INT, DINT, LINT, REAL, LREAL | 0 |
| TIME | T#0ms |
| DATE | D#1970-1-1 |
| TOD | TOD#0:0:0 |
| DT | DT#1970-1-1-0:0:0 |
| STRING | ‘’ |

- When "In" represents the entire array, with one element and one variable in the array, the following table shows the processing:

| “In” | Processing content |
|---|---|
| The entire array | Initialize all elements of an array |
| An element of an array | Only initialize this element |
| The entire structure | Initialize all structural elements of the structure |

| | |
|---|---|
| One structural element of a structure | Only initialize this structural element |

(4) Program example

| DEMO | LD | ST |
|---|---|---|
| Define variable | VAR<br>bStart:BOOL;<br>Clear_IN:ARRAY[1..10] OF UINT:=[66,77,52,3,5,789,56,366,32,75];<br>END_VAR | |
| Program example |  | 1 Clear(In:= Clear_IN); |
| Operation result |  | |

| 表达式 | 类型 | 值 |
|---|---|---|
| bStart | BOOL | TRUE |
| Clear_IN | ARRAY [1..10] OF UINT | |
| Clear_IN[1] | UINT | 16#0000 |
| Clear_IN[2] | UINT | 16#0000 |
| Clear_IN[3] | UINT | 16#0000 |
| Clear_IN[4] | UINT | 16#0000 |
| Clear_IN[5] | UINT | 16#0000 |
| Clear_IN[6] | UINT | 16#0000 |
| Clear_IN[7] | UINT | 16#0000 |
| Clear_IN[8] | UINT | 16#0000 |
| Clear_IN[9] | UINT | 16#0000 |
| Clear_IN[10] | UINT | 16#0000 |

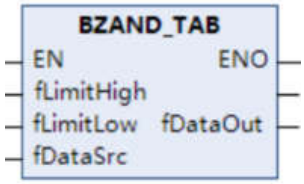
## 1-4-9. Regional control instructions

### 1-4-9-1. Dead zone control command BZAND_TAB

(1) Instruction overview

Determine whether the input value is within the specified upper and lower limits of the dead zone, in order to control the output value.

| Dead zone control command [BZAND_TAB] | | | |
|---|---|---|---|
| Execution condition | Edge triggering | Suitable model | XS, XSA |
| Firmware | - | Software | XS StudioV2.2.0 and above |
| Library | XJ_DataUtils | | |

| Instruction | Name | Graph | ST |
|---|---|---|---|
| BZAND_TAB | Dead zone control command |  | XDONE:=BZAND_TAB(fLimitHigh:= ,<br>fLimitLow:= ,<br>fDataSrc:= ,<br>fDataOut=> ); |

(2) Related variables

| VAR_INPUT | Name | Data type | Valid range | Initial value | Description |
|---|---|---|---|---|---|
| fLimitHigh | Dead zone upper limit value | REAL | As the data type | 0 | Dead zone upper limit value |
| fLimitLow | Dead zone lower limit value | REAL | As the data type | 0 | Dead zone lower limit value |
| fDataSrc | Source data | REAL | As the data type | 0 | Input value through dead zone |
| VAR_OUTPUT | Name | Data type | Valid range | Initial value | Description |
| fDataOut | Output data | REAL | As the data type | - | Output value through dead zone |

(3) Function

- Control the output value by determining whether the input value is within the specified upper and lower limits of the dead zone; Set the dead zone range in fLimitLow, fLimitHigh to output the input value fDatasrc outside the dead zone range to fDataOut.
- When fLimitLow>fDatasrc, fDataOut = fDataSrc - fLimitLow;
- When fLimitHigh<fDatasrc, fDataOut = fDataSrc - fLimitHigh;
- When fLimitLow ≤ fDataSrc ≤ fLimitHigh, fDataOut = 0.

(4) Note

- When fDataSrc input is a non numeric value, FDataOut output is 0.
- When fLimitLow is a non numerical value and fDataSrc>fLimitHigh, When fDataOut=fDataSrc-fLimitHigh; fDataSrc<fLimitHigh, FDataOut is 0.

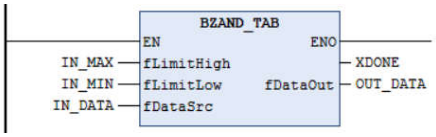
- When fLimitHigh is a non numerical value and fDataSrc<fLimitLow, When fDataOut=fDataSrc fLimitLow; fDataSrc>fLimitLow, FDataOut is 0.
- When fLimitHigh, fLimitLow are all non numeric, FDataOut is 0.

(5) Program example

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

```
VAR
    IN_MAX: REAL:= 10;
    IN_MIN: REAL:= 1;
    IN_DATA: REAL:= -2;
    OUT_DATA: REAL;
    XDONE: BOOL;
END_VAR
```

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




</td><td>

```
XDONE:= BZAND_TAB(
    fLimitHigh:= IN_MAX,
    fLimitLow:= IN_MIN,
    fDataSrc:= IN_DATA,
    fDataOut=> OUT_DATA);
```

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

| 表达式 | 类型 | 值 |
|---|---|---|
| IN_MAX | REAL | 10 |
| IN_MIN | REAL | 1 |
| IN_DATA | REAL | -2 |
| OUT_DATA | REAL | -3 |
| XDONE | BOOL | TRUE |

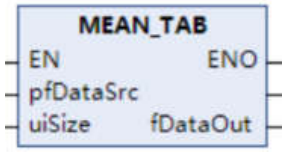
</td></tr>
</table>

1-4-9-2. Calculation of average data value MEAN_TAB

(1) Instruction overview

Calculate the average value of the specified data.

| Calculation of average data value [MEAN_TAB] | | | |
|---|---|---|---|
| Execution condition | Edge triggering | Suitable model | XS, XSA |
| Firmware | - | Software | XS StudioV2.2.0 and above |
| Library | XJ_DataUtils | | |

| Instruction | Name | Graph | ST |
|---|---|---|---|
| MEAN_TAB | Calculation of average data value |  | XDONE:=MEAN_TAB(pfDataSrc:= ,<br>uiSize:= ,<br>fDataOut=> ); |

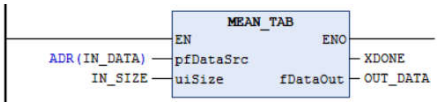
(2) Related variables

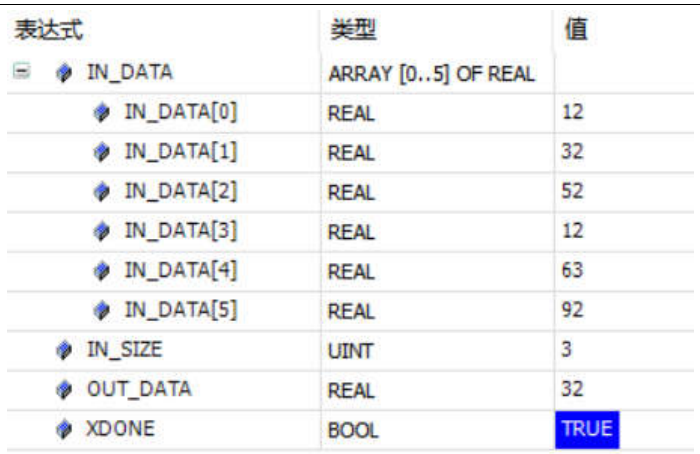
| VAR_INPUT | Name | Data type | Valid range | Initial value | Description |
|---|---|---|---|---|---|
| pfDataSrc | Source data | POINTER TO REAL | - | - | Input data address |
| uiSize | Data length | UINT | 1-1024 | 0 | Data length |
| VAR_OUTPUT | Name | Data type | Valid range | Initial value | Description |
| fDataOut | Output data | REAL | As the data type | - | Output average |

(3) Function

- Perform the average operation on the number of uiSize starting from pfDataSrc, and store the result in fDataOut.

(4) Program example

| DEMO | LD | ST |
|---|---|---|
| Define variable | VAR<br>IN_DATA: ARRAY[0..5] OF REAL:= [12,32,52,12,63,92];<br>IN_SIZE: UINT:= 3;<br>OUT_DATA: REAL;<br>XDONE: BOOL;<br>END_VAR | |
| Program example |  | XDONE:= MEAN_TAB(<br>pfDataSrc:= ADR(IN_DATA),<br>uiSize:= IN_SIZE,<br>fDataOut=> OUT_DATA); |

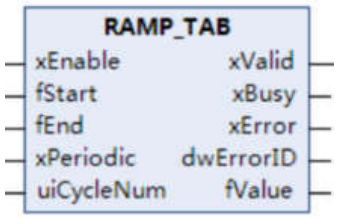
| | |
|---|---|
| Operation result |  |

1-4-9-3. Slope command RAMP_TAB

(1) Instruction overview

Slope command, with a specified interval, the output value changes from the lowest value to the highest value, or from the highest value to the lowest value, performing linear interpolation between high and low values.

| Slope command [RAMP_TAB] | | | |
|---|---|---|---|
| Execution condition | Edge triggering | Suitable model | XS, XSA |
| Firmware | - | Software | XS StudioV2.2.0 and above |
| Library | XJ_FunctionUtils | | |

| Instruction | Name | Graph | ST |
|---|---|---|---|
| RAMP_TAB | Slope command |  | RAMP_TAB_0(<br>xEnable:= ,<br>fStart:= ,<br>fEnd:= ,<br>xPeriodic:= ,<br>uiCycleNum:=<br>xValid=> ,<br>xBusy=> ,<br>xError=> ,<br>dwErrorID=> ,<br>fValue=> ); |

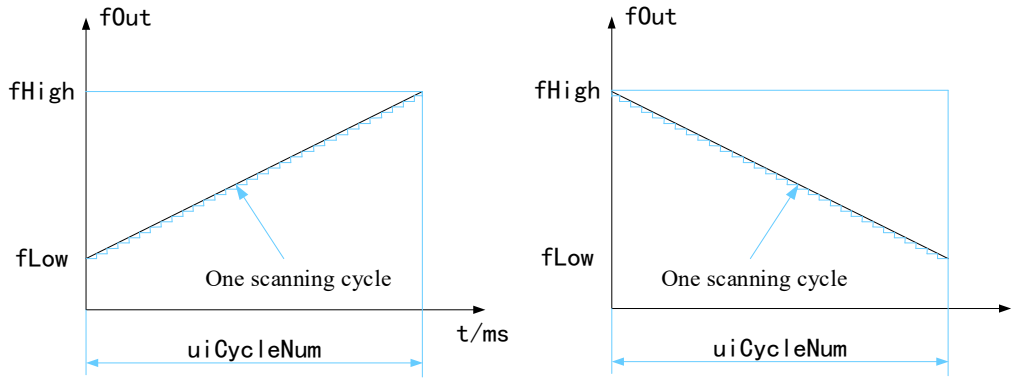
(2) Related variables

| VAR_INPUT | Name | Data type | Valid range | Initial value | Description |
|---|---|---|---|---|---|
| xEnable | Execute | BOOL | FALSE/TRUE | FALSE | Input high-level start command |
| fStart | Start value | REAL | As the data type | 0 | Start value |
| fEnd | End value | REAL | As the data type | 0 | End value |
| xPeriodic | Cyclic output | BOOL | FALSE/TRUE | FALSE | FALSE: Execute once<br>True: Loop execution |
| uiCycleNum | Number of completed cycles | UINT | As the data type | 0 | Number of completed cycles |
| VAR_OUTPUT | Name | Data type | Valid range | Initial value | Description |
| xValid | Completed | BOOL | FALSE/TRUE | FALSE | True: Instruction execution completed<br>FALSE: Instruction not completed |
| xBusy | Executing | BOOL | FALSE/TRUE | FALSE | True: Instruction execution in progress<br>FALSE: Instruction not in progress |
| xError | Error | BOOL | FALSE/TRUE | FALSE | True: Instruction execution error<br>FALSE: No error occurred during |

| | | | | | |
|---|---|---|---|---|---|
| | | | | | instruction execution |
| dwErrorID | Error code | DWORD | - | 0 | Error code |
| fValue | Current value | REAL | - | 0 | Current output value |

(3) Function

- The function of the instruction is to perform linear interpolation between two given data points within a specified time interval, and output process values sequentially according to the scanning execution time until the endpoint value at the end of the interval.



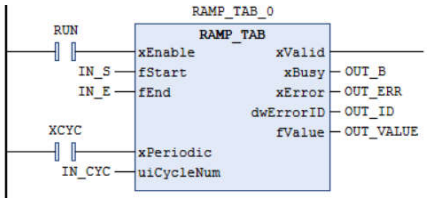

(4) Note

- xEnable changes from FALSE to True, The fValue value starts to change again, xBusy is True, after the run ends, xValid, xBusy is True.
- After xEnable changes from True to False, fValue value is maintained.
- Only xEnable is False, fStart, fEnd, xPeriodic, uiCycleNum can be written and take effect when xEnable is true.
- When xEnable is False, the value of fValue can be modified, but it does not take effect.
- When xEnable is True, the values of all input and output variables can be modified, but it cannot take effect.
- When uiCycleNum is set to 0, When xEnable is True, XError is True, dwErrorID is 1, xValid, xBusy is False.

(5) Program example

| DEMO | LD | ST |
|---|---|---|
| Define variable | | VAR<br>RAMP_TAB_0: RAMP_TAB;<br>RUN: BOOL;<br>IN_S: REAL:= 5;<br>IN_E: REAL:=100;<br>XCYC: BOOL;<br>IN_CYC: UINT:=95;<br>OUT_B: BOOL;<br>OUT_ERR: BOOL;<br>OUT_ID: DWORD;<br>OUT_VALUE: REAL;<br>XDONE: BOOL;<br>END_VAR |

Program example




```
RAMP_TAB_0(
   xEnable:= RUN,
   fStart:= IN_S,
   fEnd:= IN_E,
   xPeriodic:= XCYC,
   uiCycleNum:= IN_CYC,
   xValid=> XDONE,
   xBusy=> OUT_B,
   xError=> OUT_ERR,
   dwErrorID=> OUT_ID,
   fValue=> OUT_VALUE);
```

Operation result

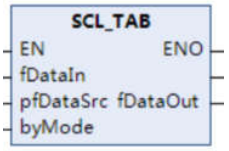
| 表达式 | 类型 | 值 |
|---|---|---|
| RAMP_TAB_0 | RAMP_TAB | |
| RUN | BOOL | TRUE |
| IN_S | REAL | 5 |
| IN_E | REAL | 100 |
| XCYC | BOOL | FALSE |
| IN_CYC | UINT | 95 |
| OUT_B | BOOL | TRUE |
| OUT_ERR | BOOL | FALSE |
| OUT_ID | DWORD | 0 |
| OUT_VALUE | REAL | 100 |
| XDONE | BOOL | TRUE |

1-4-9-4. Table coordinate acquisition instruction SCL_TAB

(1) Instruction overview

Pre store table data, input values, and then obtain corresponding output values based on the table data.

| Table coordinate acquisition instruction [SCL_TAB] | | | |
|---|---|---|---|
| Execution condition | Edge triggering | Suitable model | XS, XSA |
| Firmware | - | Software | XS StudioV2.2.0 and above |
| Library | XJ_FunctionUtils | | |

| Instruction | Name | Graph | ST |
|---|---|---|---|
| SCL_TAB | Table coordinate acquisition instruction |  | `XDONE:=SCL_TAB(fDataIn:= , pfDataSrc:= , byMode:= , fDataOut=> );` |



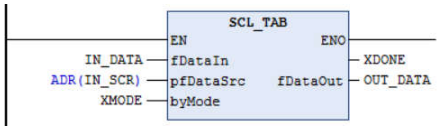
(2) Related variables

| VAR_INPUT | Name | Data type | Valid range | Initial value | Description |
|---|---|---|---|---|---|
| fDataIn | Input value | REAL | As the data type | 0 | Input data |
| pfDataSrc | Source data | POINTER TO REAL | - | - | Table data |
| byMode | Mode selection | _modeType | 1/2 | _xymode | 1：x1,y1;x2,y2;....xn,yn<br>2：x1,x2,...xn;y1,y2...yn |
| VAR_OUTPUT | Name | Data type | Valid range | Initial value | Description |
| fDataOut | Output value | REAL | - | - | Output value obtained by referring to table data |

(3) Function

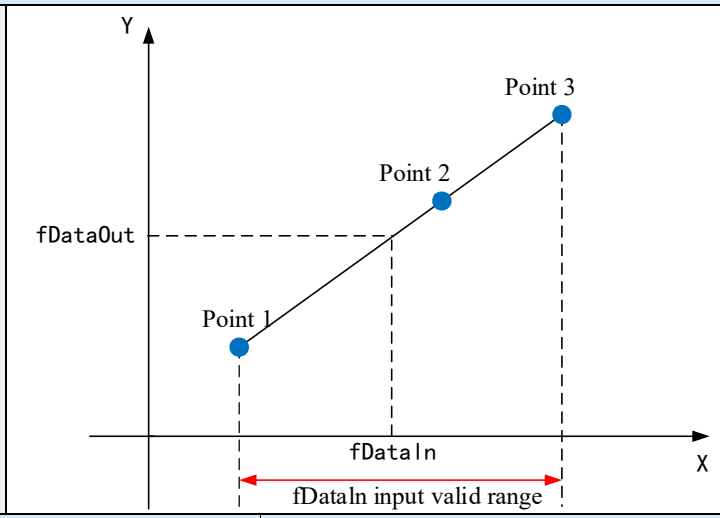
- Based on the graph determined by the table in pfDatasrc, find the output value fDataOut corresponding to the input value fDataIn.

(4) Program example

| DEMO | LD | ST |
|---|---|---|
| Define variable | | `VAR`<br>`  IN_DATA: REAL;`<br>`  IN_SCR: ARRAY[0..6] OF REAL:= [10,12,12,24,24,31,31];`<br>`  XMODE: _modeType:= 1;`<br>`  OUT_DATA: REAL;`<br>`  XDONE: BOOL;`<br>`END_VAR` |
| Program example |  | `XDONE:= SCL_TAB(`<br>`  fDataIn:= IN_DATA,`<br>`  pfDataSrc:=ADR(IN_SCR),`<br>`  byMode:= XMODE,`<br>`  fDataOut=> OUT_DATA);` |

| | |
|---|---|
| Operation result | 表达式 / 类型 / 值<br>IN_DATA / REAL / 18<br>IN_SCR / ARRAY [0..6] OF REAL<br>IN_SCR[0] / REAL / 10<br>IN_SCR[1] / REAL / 12<br>IN_SCR[2] / REAL / 12<br>IN_SCR[3] / REAL / 24<br>IN_SCR[4] / REAL / 24<br>IN_SCR[5] / REAL / 31<br>IN_SCR[6] / REAL / 31<br>XMODE / _MODETYPE / xymode<br>OUT_DATA / REAL / 18<br>XDONE / BOOL / TRUE |

The principle analysis of the example program is as follows:

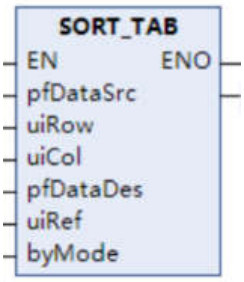
| Principle analysis | | |
|---|---|---|
| Table Data Diagram |  | |
| Source data (pfDataSrc) | | Table data address |
| Coordinate point data value | | (pfDataSrc)^ |
| Point 1 | X1 coordinate | (pfDataSrc+1)^ |
| | Y1 coordinate | (pfDataSrc+2)^ |
| Point 2 | X2 coordinate | (pfDataSrc+3)^ |
| | Y2 coordinate | (pfDataSrc+4)^ |
| Point 3 | X3 coordinate | (pfDataSrc+5)^ |
| | Y3 coordinate | (pfDataSrc+6)^ |

1-4-9-5. Table data sorting instruction SORT_TAB

(1) Instruction overview

Sort the array of uiRow rows × uiCol columns by the uiRef column or row parameter, and store the sorting results in the variable area starting with the pfDataDes unit.

| Table data sorting instruction [SORT_TAB] | | | |
|---|---|---|---|
| Execution condition | Edge triggering | Suitable model | XS, XSA |
| Firmware | - | Software | XS StudioV2.2.0 and above |
| Library | XJ_FunctionUtils | | |

| Instruction | Name | Graph | ST |
|---|---|---|---|
| SORT_TAB | Table data sorting instruction |  | XDONE:=SORT_TAB(pfDataSrc:= ,<br>uiRow:= ,<br>uiCol:= ,<br>pfDataDes:= ,<br>uiRef:= ,<br>byMode:= ); |

(2) Related variables

| VAR_INPUT | Name | Data type | Valid range | Initial value | Description |
|---|---|---|---|---|---|
| pfDataSrc | Source data | POINTER TO REAL | - | - | Arrange data objects |
| uiRow | Row | UINT | 1-1024 | 0 | Data row |
| uiCol | column | UINT | 1-1024 | 0 | Data column |
| pfDataDes | Output data | POINTER TO REAL | - | - | Data output |
| uiRef | Row/column | UINT | 1-1024 | 0 | Based on rows or columns |
| byMode | Arrangement method | BYTE | 1-4 | 0 | Arrangement method<br>1: Column reference ascending order<br>2: Column reference descending order<br>3: Row reference ascending order<br>4: Row Reference Descending |

(3) Function

- pfDatasrc is the starting unit of the first variable in the first row (or the first record).
- The number of rows in the uiRow array, also known as the number of records.
- The number of columns in the uiCol array, also known as the number of columns per record.
- The starting unit stored after sorting in pfDataDes occupies the same number of subsequent variable units as the number of array variables before sorting.
- uiRef is a row/column number based on sorting.

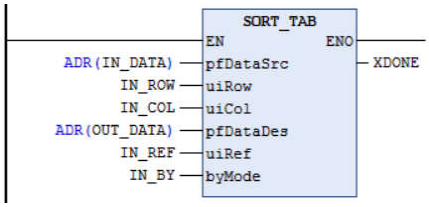
- byMode sorting mode 1: column reference ascending, 2: column reference descending, 3: row reference ascending, 4: row reference descending.

(4) Program example

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

```
VAR
   IN_DATA: ARRAY[0..8] OF REAL:= [1,2,3,4,5,6,7,8,9];
   IN_ROW: UINT:= 3;
   IN_COL: UINT:= 3;
   OUT_DATA: ARRAY[0..8] OF REAL;
   IN_REF: UINT:=2;
   IN_BY: BYTE:= 2;
   XDONE: BOOL;
END_VAR
```

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




</td><td>

```
XDONE:= SORT_TAB(
    pfDataSrc:= ADR(IN_DATA),
    uiRow:= IN_ROW,
    uiCol:= IN_COL,
    pfDataDes:= ADR(OUT_DATA),
    uiRef:= IN_REF,
    byMode:= IN_BY);
```

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

| 表达式 | 类型 | 值 |
|---|---|---|
| IN_DATA | ARRAY [0..8] OF REAL | |
| IN_DATA[0] | REAL | 1 |
| IN_DATA[1] | REAL | 2 |
| IN_DATA[2] | REAL | 3 |
| IN_DATA[3] | REAL | 4 |
| IN_DATA[4] | REAL | 5 |
| IN_DATA[5] | REAL | 6 |
| IN_DATA[6] | REAL | 7 |
| IN_DATA[7] | REAL | 8 |
| IN_DATA[8] | REAL | 9 |
| IN_ROW | UINT | 3 |
| IN_COL | UINT | 3 |
| OUT_DATA | ARRAY [0..8] OF REAL | |
| OUT_DATA[0] | REAL | 3 |
| OUT_DATA[1] | REAL | 2 |
| OUT_DATA[2] | REAL | 1 |
| OUT_DATA[3] | REAL | 6 |
| OUT_DATA[4] | REAL | 5 |
| OUT_DATA[5] | REAL | 4 |
| OUT_DATA[6] | REAL | 9 |
| OUT_DATA[7] | REAL | 8 |
| OUT_DATA[8] | REAL | 7 |
| IN_REF | UINT | 2 |
| IN_BY | BYTE | 2 |
| XDONE | BOOL | TRUE |

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

- The principle analysis of the example program is as follows: perform descending processing on the second column.

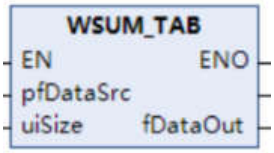
| Data before execution | | | |
|---|---|---|---|
| - | Column 1 | Column 2 | Column 3 |
| Row 1 | pfDataSrc[0]=1 | pfDataSrc[3]=4 | pfDataSrc[6]=7 |
| Row 2 | pfDataSrc[1]=2 | pfDataSrc[4]=5 | pfDataSrc[7]=8 |
| Row 3 | pfDataSrc[2]=3 | pfDataSrc[5]=6 | pfDataSrc[8]=9 |
| Data after execution | | | |
| - | Column 1 | Column 2 | Column 3 |
| Row 1 | pfDataSrc[0]=3 | pfDataSrc[3]=6 | pfDataSrc[6]=9 |
| Row 2 | pfDataSrc[1]=2 | pfDataSrc[4]=5 | pfDataSrc[7]=8 |
| Row 3 | pfDataSrc[2]=1 | pfDataSrc[5]=4 | pfDataSrc[8]=7 |

1-4-9-6. Data calculation total instruction WSUM_TAB

(1) Instruction overview

Calculate the total value of a specified amount of table data.

| Data calculation total instruction [WSUM_TAB] | | | |
|---|---|---|---|
| Execution condition | Edge triggering | Suitable model | XS, XSA |
| Firmware | - | Software | XS StudioV2.2.0 and above |
| Library | XJ_FunctionUtils | | |

| Instruction | Name | Graph | ST |
|---|---|---|---|
| WSUM_TAB | Data calculation total instruction |   | XDONE:=WSUM_TAB(pfDataSrc:= ,<br>uiSize:= ,<br>fDataOut=> ); |

(2) Related variables

| VAR_INPUT | Name | Data type | Valid range | Initial value | Description |
|---|---|---|---|---|---|
| pfDataSrc | Source data | POINTER TO REAL | - | - | Input data address |
| uiSize | Data length | UINT | 1-1024 | 0 | Data length |
| VAR_OUTPUT | Name | Data type | Valid range | Initial value | Description |
| fDataOut | Output data | REAL | As the data type | - | Output total value |

(3) Function

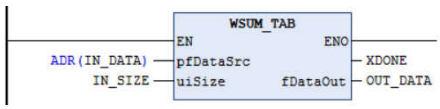
- Calculate the total length of uiSize data in the input data pfDataSrc and output it to fDataOut.

(4) Program example

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

```
VAR
   IN_DATA: ARRAY[0..5] OF REAL:= [12,16,22,10,17,31];
   IN_SIZE: UINT:= 3;
   OUT_DATA: REAL;
   XDONE: BOOL;
END_VAR
```

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




</td><td>

```
XDONE:= WSUM_TAB(
    pfDataSrc:= ADR(IN_DATA),
    uiSize:= IN_SIZE,
    fDataOut=> OUT_DATA);
```

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

| 表达式 | 类型 | 值 |
|---|---|---|
| IN_DATA | ARRAY [0..5] OF REAL | |
| IN_DATA[0] | REAL | 12 |
| IN_DATA[1] | REAL | 16 |
| IN_DATA[2] | REAL | 22 |
| IN_DATA[3] | REAL | 10 |
| IN_DATA[4] | REAL | 17 |
| IN_DATA[5] | REAL | 31 |
| IN_SIZE | UINT | 3 |
| OUT_DATA | REAL | 50 |
| XDONE | BOOL | TRUE |

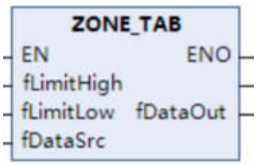
</td></tr>
</table>

1-4-9-7. Zone control instruction ZONE_TAB

(1) Instruction overview

Control the output value with the specified deviation value based on whether the input value is positive or negative (add the deviation value to the input value).

| Zone control instruction [ZONE_TAB] | | | |
|---|---|---|---|
| Execution condition | Edge triggering | Suitable model | XS, XSA |
| Firmware | - | Software | XS StudioV2.2.0 and above |
| Library | XJ_FunctionUtils | | |

| Instruction | Name | Graph | ST |
|---|---|---|---|
| ZONE_TAB | Zone control instruction |  | XDONE:=ZONE_TAB(fLimitHigh:= ,<br>fLimitLow:= ,<br>fDataSrc:= ,<br>fDataOut=> ); |

(2) Related variables

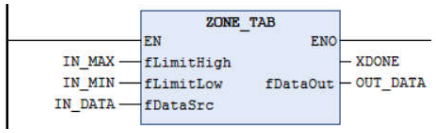
| VAR_INPUT | Name | Data type | Valid range | Initial value | Description |
|---|---|---|---|---|---|
| fLimitHigh | Positive deviation | REAL | As the data type | 0 | Positive deviation, the deviation value added to the input value |
| fLimitLow | Negative deviation | REAL | As the data type | 0 | Negative deviation, the deviation value added to the input value |
| fDataSrc | Input value | REAL | As the data type | 0 | Source data |
| VAR_OUTPUT | Name | Data type | Valid range | Initial value | Description |
| fDataOut | Output value | REAL | As the data type | - | Output value |

(3) Function

- Determine the sign of the input value fDataSrc, add fLimitHigh or fLimitLow, and save the execution result to fDataOut.
- When fDataSrc<0, fDataOut = fDataSrc + fLimitLow;
- When fDataSrc>0, fDataOut = fDataSrc + fLimitHigh;
- When fDataSrc=0, fDataOut = 0.

(4) Program example

| DEMO | LD | ST |
|---|---|---|
| Define variable | | VAR<br>IN_MAX: REAL:= 6;<br>IN_MIN: REAL:= 2;<br>IN_DATA: REAL;<br>OUT_DATA: REAL;<br>XDONE: BOOL;<br>END_VAR |

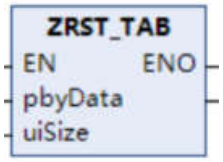
| | | |
|---|---|---|
| Program example |  | XDONE:= ZONE_TAB(<br>fLimitHigh:= IN_MAX,<br>fLimitLow:= IN_MIN,<br>fDataSrc:= IN_DATA,<br>fDataOut=> OUT_DATA); |
| Operation result | 表达式 / 类型 / 值<br>IN_MAX / REAL / 6<br>IN_MIN / REAL / 2<br>IN_DATA / REAL / -5<br>OUT_DATA / REAL / -3<br>XDONE / BOOL / TRUE | |

1-4-9-8. Batch data reset instruction ZRST_TAB

(1) Instruction overview

Batch reset the data in the table.

| Batch data reset instruction [ZRST_TAB] | | | |
|---|---|---|---|
| Execution condition | Edge triggering | Suitable model | XS, XSA |
| Firmware | - | Software | XS StudioV2.2.0 and above |
| Library | XJ_FunctionUtils | | |

| Instruction | Name | Graph | ST |
|---|---|---|---|
| ZRST_TAB | Batch data reset instruction |  | XDONE:=ZRST_TAB(pbyData:= , uiSize:= ); |

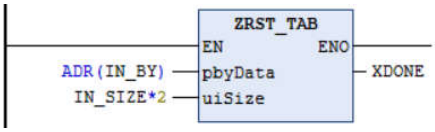
(2) Related variables

| VAR_INPUT | Name | Data type | Valid range | Initial value | Description |
|---|---|---|---|---|---|
| pbyData | Source data | POINTER TO BYTE | - | - | Data header address |
| uiSize | Data length | UINT | 1-1024 | 0 | Data length |

(3) Function

- Batch reset input source data of specified length.

(4) Program example

| DEMO | LD | ST |
|---|---|---|
| Define variable | VAR<br>IN_BY: ARRAY[0..5] OF WORD:= [9,5,3,2,4,6];<br>IN_SIZE: UINT:= 5;<br>XDONE: BOOL;<br>END_VAR | |
| Program example |  | XDONE:= ZRST_TAB(<br>pbyData:= ADR(IN_BY),<br>uiSize:= IN_SIZE*2); |

| | |
|---|---|
| Operation result | |

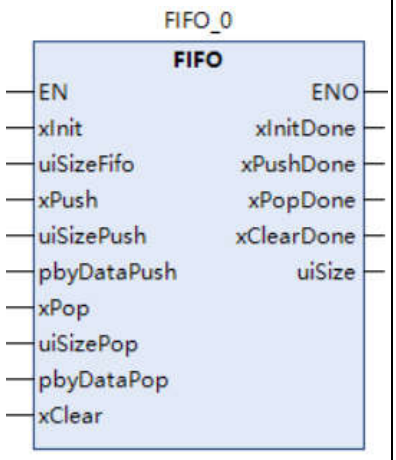
## 1-4-10. Stack and Table

### 1-4-10-1. First in, first out queue function block FIFO

(1) Instruction overview

Including queue initialization, queue entry, queue exit, and queue clearing functions.

| First in, first out queue function block [FIFO] | | | |
|---|---|---|---|
| Execution condition | Edge triggering | Suitable model | XS, XSA |
| Firmware | - | Software | XS StudioV2.3.1 and above |
| Library | XJ_FunctionUtils | | |

| Instruction | Name | Graph | ST |
|---|---|---|---|
| FIFO | First in, first out queue function block |  | FIFO_0(<br>xInit:= ,<br>uiSizeFifo:= ,<br>xPush:= ,<br>uiSizePush:= ,<br>pbyDataPush:= ,<br>xPop:= ,<br>uiSizePop:= ,<br>pbyDataPop:= ,<br>xClear:= ,<br>xInitDone=> ,<br>xPushDone=> ,<br>xPopDone=> ,<br>xClearDone=> ,<br>uiSize=> ); |

(2) Related variables

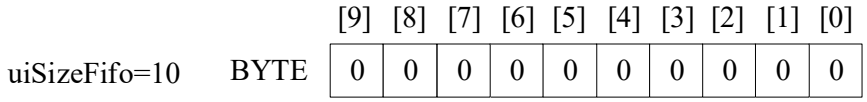
| VAR_INPUT | Name | Data type | Effective range | Initial value | Description |
|---|---|---|---|---|---|
| xInit | Initialization | BOOL | TRUE/FALSE | FALSE | Rising edge triggering, initialize to open memory, initialize once after power on |
| uiSizeFifo | Initialize and allocate memory space | UINT | 0-1024 | 0 | Initialize queue size |
| xPush | Data write in the queue | BOOL | TRUE/FALSE | FALSE | Rising edge triggered, data enter queue |
| uiSizePush | Write in queue data size | UINT | 0-1024 | 0 | The number of data entering the queue, in BYTE units, should not exceed the initialization value |
| pbyDataPush | Write in queue data starting address | POINTER_TO_BYTE | - | - | Head address of the enqueued data pointer |

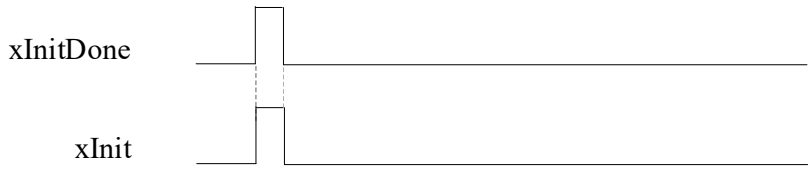
| VAR_INPUT | Name | Data type | Effective range | Initial value | Description |
|---|---|---|---|---|---|
| | | | | | |
| xPop | Data dequeue | BOOL | TRUE/FALSE | FALSE | On the rising edge, dequeue the data |
| uiSizePop | Read the queue data size | UINT | 0-1024 | 0 | The number of dequeue data, in BYTE units, should not exceed the initialization value |
| pbyDataPop | Read queue data head address | POINTER_TO_BYTE | - | - | Head address of the dequeued data pointer |
| xClear | Empty queue | BOOL | TRUE/FALSE | FALSE | Rising edge triggered, queue data cleared |
| VAR_OUTPUT | Name | Data type | Effective range | Initial value | Description |
| xInitDone | Initialization completion flag | BOOL | TRUE/FALSE | FALSE | Initialization completion flag |
| xPushDone | Enqueue Completion Flag | BOOL | TRUE/FALSE | FALSE | Enqueue Completion Flag |
| xPopDone | Dequeue Completion Flag | BOOL | TRUE/FALSE | FALSE | Dequeue Completion Flag |
| xClearDone | Clear queue completion flag | BOOL | TRUE/FALSE | FALSE | Clear queue completion flag |
| uiSize | Remaining bytes in the queue | UINT | 0-1024 | 0 | Remaining bytes in the queue |

(3) Steps for using FIFO functional block

Step 1: Initialize and apply for queue cache

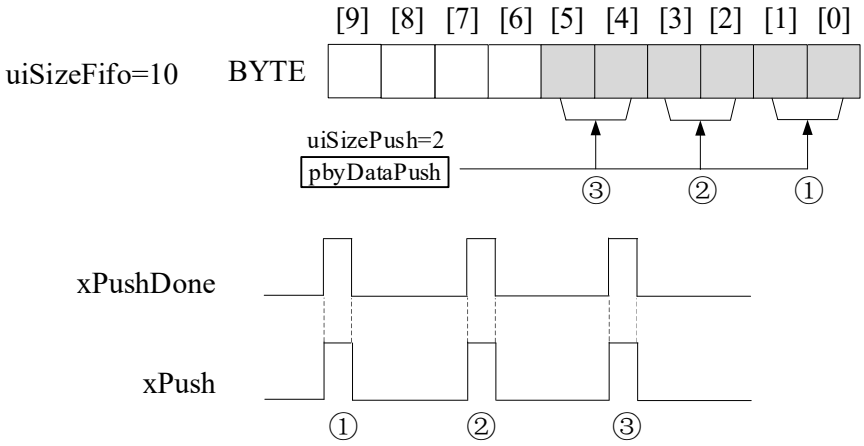
Trigger xInit to initialize the FIFO (initialization once is sufficient), at which point the FIFO creates cache space. If successful, xInitDone will be set to TRUE within one cycle. This example successfully created a 10 byte cache space in the PLC memory.




Step 2: Enqueue

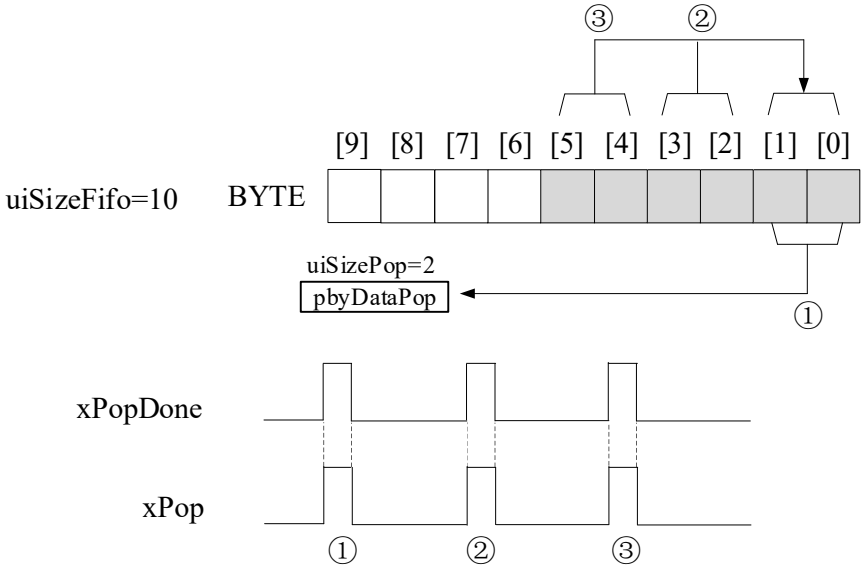
Trigger xPush to push the length uiSizePush byte data pointed to by pbyDataPush into the queue. If successful, set xPushDone to TRUE within one cycle; UiSize increase uiSizePush.




uiSizePush=2. Each time xPush is triggered to write two bytes of data to the queue, uiSize increases by 2.

Step 3: Dequeue

Trigger xPop, dequeues data of length uiSizePop (in bytes) from the FIFO buffer queue and stores it into pbyDataPop. Upon success, xPopDone is set to TRUE within one cycle, and uiSize is decremented by uiSizePop.




uiSizePop=2， Each time xPop is triggered, it reads 2 bytes of content from the header byte of the data queue to the pbyDataPop address, and at the same time, the data in the queue moves 2 bytes from left to right, reducing uiSize by 2.

Step 4: clear the queue

Whether to clear the queue as needed. Trigger xClean, if successful, set xClearDone to TRUE within one cycle.

(4) Note

- xInit initializes the queue. This operation only needs to be triggered once. Repeated initialization may cause memory fragmentation, which could affect subsequent memory allocation. The requested memory size should preferably be less than 1024 bytes

(5) Program example

| DEMO | LD | ST |
|---|---|---|
| Define the variables | | |

```
VAR
    FIFO_0: FIFO;
    xInit0:BOOL;//上升沿触发，初始化，创建缓存空间
    uiSizeFifo0:UINT;//设置的初始化分配内存空间大小，单位是字节

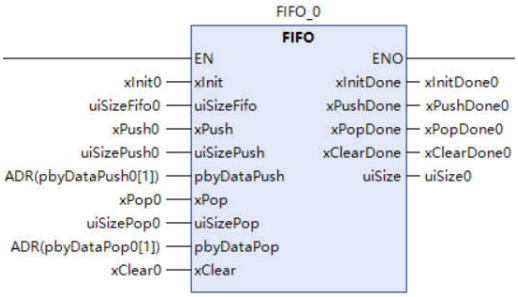
    xPush0:BOOL;//上升沿触发，数据写入队列
    uiSizePush0:UINT:=2;//写入队列数据数量，单位是字节，数量不要大于初始化值，这边设为2个字节
    pbyDataPush0:ARRAY[1..3]OF BYTE:=[1,2,3];//写入队列的数据

    xPop0:BOOL;//上升沿触发，数据出队列
    uiSizePop0:UINT:=2;//出队列数据数量，单位是字节，数量不要大于初始化值，这边设为2个字节
    pbyDataPop0:ARRAY[1..3]OF BYTE;//出队列的数据

    xClear0:BOOL;//上升沿触发，队列数据清空

    xInitDone0:BOOL;//初始化完成标志
    xPushDone0:BOOL;//写入（进）队列完成标志
    xPopDone0:BOOL;//出队列完成标志
    xClearDone0:BOOL;//清空队列完成标志
    uiSize0:UINT;//队列剩余字节数
END_VAR
```

Program example




```
FIFO_0(
    xInit:= xInit0,
    uiSizeFifo:= uiSizeFifo0,
    xPush:= xPush0,
    uiSizePush:= uiSizePush0,
    pbyDataPush:= ADR(pbyDataPush0[1]),
    xPop:= xPop0,
    uiSizePop:= uiSizePop0,
    pbyDataPop:= ADR(pbyDataPop0[1]),
    xClear:= xClear0,
    xInitDone=> xInitDone0,
    xPushDone=> xPushDone0,
    xPopDone=> xPopDone0,
    xClearDone=> xClearDone0,
    uiSize=> uiSize0);
```

Operation result

Example of running results using ST programming language

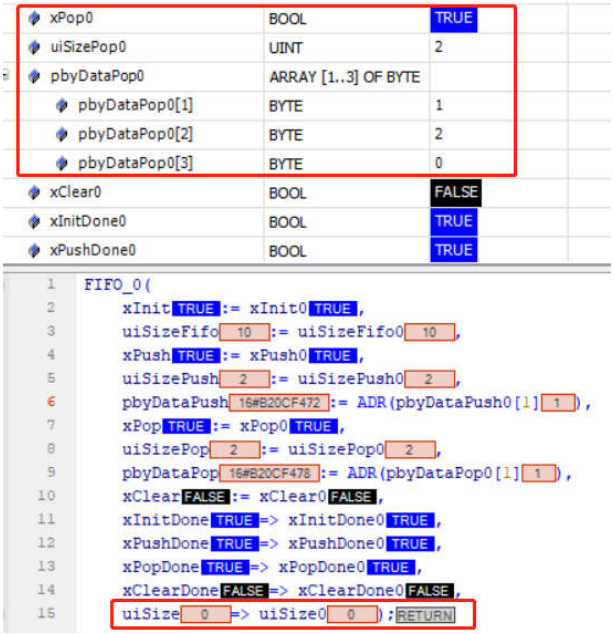
1. Set uiSizeFifo to 10, then trigger xInit for initialization. Display "Initialization Complete" when done.

```
FIFO_0(
    xInit TRUE := xInit0 TRUE,
    uiSizeFifo 10 := uiSizeFifo0 10,
    xPush FALSE := xPush0 FALSE,
    uiSizePush 2 := uiSizePush0 2,
    pbyDataPush 16#B20CF472 := ADR(pbyDataPush0[1] 1),
    xPop FALSE := xPop0 FALSE,
    uiSizePop 2 := uiSizePop0 2,
    pbyDataPop 16#B20CF478 := ADR(pbyDataPop0[1] 0),
    xClear FALSE := xClear0 FALSE,
    xInitDone TRUE => xInitDone0 TRUE,
    xPushDone FALSE => xPushDone0 FALSE,
    xPopDone FALSE => xPopDone0 FALSE,
    xClearDone FALSE => xClearDone0 FALSE,
    uiSize 0 => uiSize0 0 );RETURN
```

2. Set uiSizePush to 2 to configure the queue write data, then trigger xPush. uiSize will display 2.

| DEMO | LD | ST |
|---|---|---|

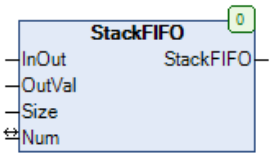
3. Set uiSizePop to 2, then trigger xPop. Reading queue data shows values, and uiSize displays 0



1-4-10-2. First in, first out instruction StackFIFO

(1) Instruction overview

| First in, first out instruction [StackFIFO] | | | |
|---|---|---|---|
| Execution condition | Edge triggering | Suitable model | XS, XSA |
| Firmware | - | Software | XS StudioV2.3.1 and above |
| Library | XJ_FunctionUtils | | |

| Instruction | Name | Graph | ST |
|---|---|---|---|
| StackFIFO | First in, first out instruction |  | StackFIFO(InOut:= , OutVal:= , Size:= , Num:= ); |

(2) Related variables

| VAR_INPUT | Name | Data type | Effective range | Initial value | Description |
|---|---|---|---|---|---|
| OutVal | Output value | - | As the data type | - | The value output from the stack |
| uiSize | Number of stacked elements | UINT | As the data type | 1 | Number of stacked elements |
| VAR_IN_OUT | Name | Data type | Effective range | Initial value | Description |
| InOut[] array | Stacked array | - | As the data type | - | Array that forms a stack |
| Num | The number of elements saved in the stack | UINT | As the data type | - | The number of elements saved in the stack |

(3) Function

- Consider the array InOut[] that currently holds the required number "Num" elements as a stacked array. Extract the value of the lowest bit InOut[0] from the stack and substitute it into the output value "OutVal".
- Then, move the "Num" -1 elements after InOut[1] towards the lower positions of the stacked array, reducing "Num" by 1.
- The stacking element number "Size" is the number of elements to be used as stacking in InOut[].

(4) Note

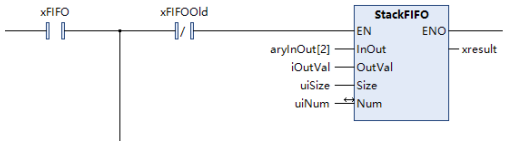
- When the value of 'Size' exceeds the array range of InOut[], the function returns a value of False, while the values of InOut[], 'Num', and 'OutVal' remain unchanged.
- OutVal "and InOut[] are of STRING type. When the byte count of InOut[] exceeds the size of "OutVal", the function returns a value of False, while the values of InOut[]," Num ", and" OutVal "remain unchanged.

- The data types of "OutVal" and "InOut[]" elements should be consistent, otherwise the program cannot run, the function returns a value of False, and the values of "InOut[]", "Num", and "OutVal" remain unchanged.
- When the value of "Size" is 0, the function returns a value of TRUE, and the values of InOut[], "Num", and "OutVal" remain unchanged.
- When the value of "Size" or "Num" is outside of 0 and "Num">"Size", the return value of the function is False, and the values of InOut[], "Num", and "OutVal" remain unchanged.
- If an element in the middle of an array is passed to InOut[], the elements after that element are the processing objects.

(5) Program example

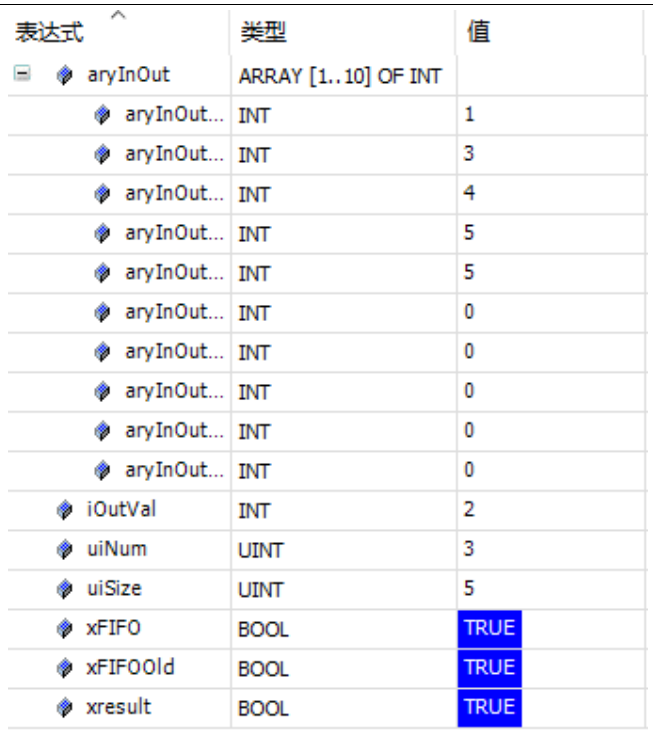
<table>
<tr><th>DEMO</th><th>LD</th><th>ST</th></tr>
<tr><td>Define the variables</td><td colspan="2">

```
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
```

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

```
IF xFIFO AND NOT xFIFOOld THEN
    xresult:=StackFIFO(InOut:=aryInOut[2] ,
        OutVal:=iOutVal ,
        Size:=uiSize ,
        Num:=uiNum
        );
END_IF
xFIFOOld:=xFIFO;
```

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

| | |
|---|---|
| Operation result |  |

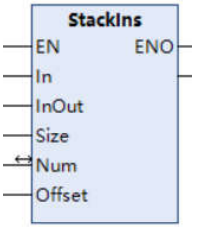
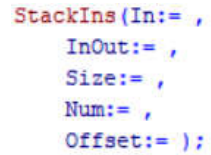


1-4-10-3. Insert stacked data StackIns

(1) Instruction overview

Insert the value into any position on the stack.

| Insert stacked data [StackIns] | | | |
|---|---|---|---|
| Execution condition | Edge triggering | Suitable model | XS, XSA |
| Firmware | - | Software | XS StudioV2.3.1 and above |
| Library | XJ_FunctionUtils | | |

| Instruction | Name | Graph | ST |
|---|---|---|---|
| StackIns | Insert stacked data |  |  |



```
StackIns(In:= ,
    InOut:= ,
    Size:= ,
    Num:= ,
    Offset:= );
```

(2) Related variables

| VAR_INPUT | Name | Data type | Effective range | Initial value | Description |
|---|---|---|---|---|---|
| In | Input value | - | As the data type | - | Input the value into the stack |

| | | | | | |
|---|---|---|---|---|---|
| InOut[] array | Stack array | - | As the data type | - | Array that forms a stack |
| Size | Number of stacked elements | UINT | As the data type | 1 | Number of stacked elements |
| Offset | Offset position | UINT | As the data type | 0 | The offset position of the stack inserted with 'In' |

| VAR_IN_OUT | Name | Data type | Effective range | Initial value | Description |
|---|---|---|---|---|---|
| Num | The number of elements saved in the stack | UINT | As the data type | - | The number of elements saved in the stack |

(3) Function

- Consider the array InOut[] that currently holds the required number "Num" elements as a stacked array. Insert the input value "In" into the position InOut["Offset"] in the stack determined by the offset position "Offset".
- The higher position elements, namely InOut["Offset"] to InOut["Num" -1], move towards the higher position direction of the stacked array. Then, "Num" increases by 1.
- The stacking element number "Size" is the number of elements to be used as stacking in InOut[].

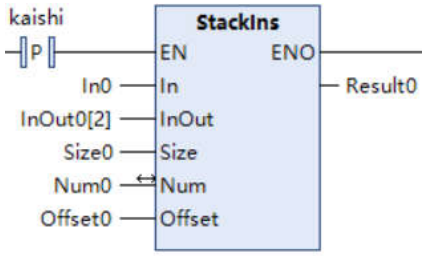
(4) Note

- When the value of 'Size' exceeds the array range of InOut[], the function returns a value of False, while the values of InOut[] and 'Num' remain unchanged.
- When "In" and "InOut" are of STRING type and the number of bytes in "In" exceeds the size of "InOut[]", the return value of the function is False, while the values of "InOut[]" and "Num" remain unchanged.
- When the data types of "In" and InOut[] are different, the function returns a value of False.
- When the value of "Size" is 0, the function returns a value of TRUE, while the values of InOut[] and "Num" remain unchanged.
- When the value of "Size" is outside of 0 and "Size">"Num"≥"Offset" is false, the program cannot run, the function returns a value of False, and the values of InOut[] and "Num" remain unchanged.
- If an element in the middle of an array is passed to InOut[], the elements after that element are the processing objects.
- The input parameter passed to 'In' must be a variable. If a constant is passed, an exception will occur during linking.

(5) Program example

| DEMO | LD | ST |
| --- | --- | --- |
| Define the variables | | |
| Program example | | |
| Operation result | | |

```
VAR
    R_TRIG0:R_TRIG;
    kaishi : BOOL;
    Result0: BOOL;
    In0    : INT  :=80;
    InOut0 :ARRAY[1..10]OF INT:=[1,2,3,4,5,6,7,8,9,10];
    Size0  : UINT :=6;
    Num0   : UINT :=4;
    Offset0: UINT :=1;
END_VAR
```




```
R_TRIG0(CLK:= kaishi, Q=> );
IF R_TRIG0.Q THEN
    Result0:=StackIns(In:= In0,
                      InOut:= InOut0[2],
                      Size:= Size0,
                      Num:= Num0,
                      Offset:= Offset0);
END_IF
```

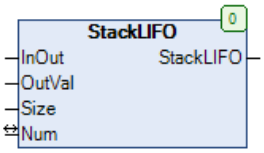
| 表达式 | 类型 | 值 |
| --- | --- | --- |
| In0 | INT | 80 |
| InOut0 | ARRAY [1..10] OF INT | |
| InOut0[1] | INT | 1 |
| InOut0[2] | INT | 2 |
| InOut0[3] | INT | 80 |
| InOut0[4] | INT | 3 |
| InOut0[5] | INT | 4 |
| InOut0[6] | INT | 5 |
| InOut0[7] | INT | 7 |
| InOut0[8] | INT | 8 |
| InOut0[9] | INT | 9 |
| InOut0[10] | INT | 10 |

```
1  R_TRIG0(CLK TRUE := kaishi TRUE, Q=> );
2  IF R_TRIG0.Q FALSE THEN
3      Result0 TRUE :=StackIns(In:= In0 80,
4                              InOut:= InOut0[2] 2,
5                              Size:= Size0 6,
6                              Num:= Num0 5,
7                              Offset:= Offset0 1 );
8  END_IF
9  RETURN
```

1-4-10-4. Last in, first out instruction StackLIFO

(1) Instruction overview

| Last in, first out instruction [StackLIFO] | | | |
|---|---|---|---|
| Execution condition | Edge triggering | Suitable model | XS, XSA |
| Firmware | - | Software | XS StudioV2.3.1 and above |
| Library | XJ_FunctionUtils | | |

| Instruction | Name | Graph | ST |
|---|---|---|---|
| StackLIFO | Last in, first out instruction |  | StackLIFO(InOut:= , OutVal:= , Size:= , Num:= ); |

(2) Related variables

| VAR_INPUT | Name | Data type | Effective range | Initial value | Description |
|---|---|---|---|---|---|
| OutVal | Output value | - | As the data type | - | The value output from the stack |
| uiSize | Number of stacked elements | UINT | As the data type | 1 | Number of stacked elements |
| VAR_IN_OUT | Name | Data type | Effective range | Initial value | Description |
| InOut[] array | Stacked array | - | As the data type | - | Array that forms a stack |
| Num | The number of elements saved in the stack | UINT | As the data type | - | The number of elements saved in the stack |

(3) Function

- Consider the array InOut[] that currently holds the required number "Num" elements as a stacked array. Extract the value of the highest bit InOut["- Num" -1] from the stack and substitute it into the output value "OutVal", Num "decreases by 1.
- The stacking element number "Size" is the number of elements to be used as stacking in InOut[].

(4) Note

- When the value of 'Size' exceeds the array range of InOut[], the function returns a value of False, while the values of InOut[], 'Num', and 'OutVal' remain unchanged.
- OutVal and InOut[] are of STRING type. When the byte count of InOut[] exceeds the size of "OutVal", the function returns a value of False, while the values of InOut[]," Num ", and" OutVal "remain unchanged.
- The data types of "OutVal" and "InOut[]" elements should be consistent, otherwise the program cannot

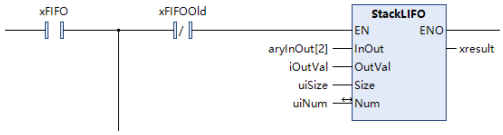
run, the function returns a value of False, and the values of "InOut[]", "Num", and "OutVal" remain unchanged.

- When the value of "Size" is 0, the function returns a value of TRUE, and the values of InOut[], "Num", and "OutVal" remain unchanged.
- When the value of "Size" or "Num" is outside of 0 and "Num">"Size", the return value of the function is False, and the values of InOut[], "Num", and "OutVal" remain unchanged.
- If an element in the middle of an array is passed to InOut[], the elements after that element are the processing objects.

(5) Program example

<table>
<tr><th>DEMO</th><th>LD</th><th>ST</th></tr>
<tr><td>Define the variables</td><td colspan="2">

```
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
```

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

```
IF xFIFO AND NOT xFIFOOld THEN
    xresult:=StackLIFO(InOut:=aryInOut[2] ,
        OutVal:=iOutVal ,
        Size:=uiSize ,
        Num:=uiNum
        );
END_IF
xFIFOOld:=xFIFO;
```

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

| 表达式 | 类型 | 值 |
|---|---|---|
| aryInOut | ARRAY [1..10] OF INT | |
| aryInOut... | INT | 1 |
| aryInOut... | INT | 2 |
| aryInOut... | INT | 3 |
| aryInOut... | INT | 4 |
| aryInOut... | INT | 5 |
| aryInOut... | INT | 0 |
| aryInOut... | INT | 0 |
| aryInOut... | INT | 0 |
| aryInOut... | INT | 0 |
| aryInOut... | INT | 0 |
| iOutVal | INT | 5 |
| uiSize | UINT | 5 |
| uiNum | UINT | 3 |
| xFIFO | BOOL | TRUE |
| xFIFOOld | BOOL | TRUE |
| xresult | BOOL | TRUE |

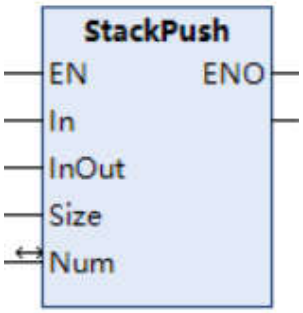
</td></tr>
</table>

1-4-10-5. Save stacked data StackPush

(1) Instruction overview

Save the value to the stack.

| Save stacked data [StackPush] | | | |
|---|---|---|---|
| Execution condition | Edge triggering | Suitable model | XS, XSA |
| Firmware | - | Software | XS StudioV2.3.1 and above |
| Library | XJ_FunctionUtils | | |

| Instruction | Name | Graph | ST |
|---|---|---|---|
| StackPush | Save stacked data |  | StackPush(In:= ,<br>InOut:= ,<br>Size:= ,<br>Num:= ); |

(2) Related variables

| VAR_INPUT | Name | Data type | Effective range | Initial value | Description |
|---|---|---|---|---|---|
| In | Input value | - | As the data type | - | Input the value into the stack |
| InOut[] array | Stacked array | - | As the data type | - | Array that forms a stack |
| Size | Number of stacked elements | UINT | As the data type | 1 | Number of stacked elements |
| VAR_OUTPUT | Name | Data type | Effective range | Initial value | Description |
| | | | As the data type | | |
| VAR_IN_OUT | Name | Data type | Effective range | Initial value | Description |
| Num | The number of elements saved in the stack | UINT | As the data type | 0 | The number of elements saved in the stack |

(3) Function

- Consider the array InOut[] that currently holds the required number "Num" elements as a stacked array.

The input value "In" will overwrite the next element InOut["Num"]. Then, "Num" increases by 1.

- The stacking element number "Size" is the number of elements to be used as stacking in InOut[].

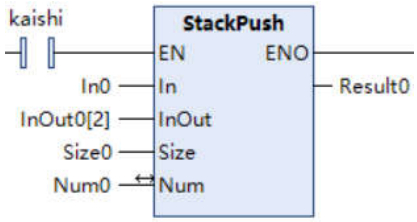
(4) Note

- When the value of 'Size' exceeds the array area of InOut[], the function returns a value of False, and InOut[] remains unchanged.
- When "In" and "InOut" are of STRING type and the number of bytes in "In" exceeds the size of "InOut[]", the return value of the function is False, and "InOut[]" remains unchanged.
- When the data types of "In" and InOut[] are different, the return value of the function is False, and the InOut value remains unchanged.
- When the value of "Size" is 0, the values of InOut[] and "Num" remain unchanged, and the function returns a value of TRUE.
- When the value of "Size" is out of 0 and "Num" ≧"Size", the program cannot run and the function returns a value of False.
- If an element in the middle of an array is passed to InOut[], the elements after that element are the processing objects.
- The input parameter passed to 'In' must be a variable. If a constant is passed, an exception will occur during linking.

(5) Program example

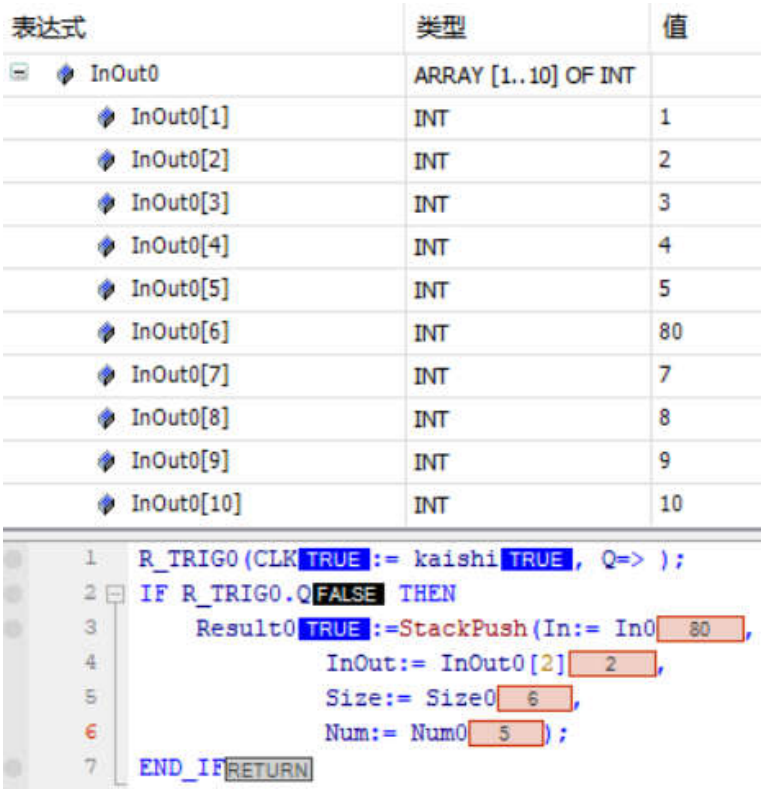
<table>
<tr><th>DEMO</th><th>LD</th><th>ST</th></tr>
<tr><td>Define the variables</td><td colspan="2">

```
VAR
    R_TRIG0:R_TRIG;
    kaishi : BOOL;
    Result0: BOOL;
    In0    : INT  :=80;
    InOut0 :ARRAY[1..10]OF INT:=[1,2,3,4,5,6,7,8,9,10];
    Size0  : UINT :=6;
    Num0   : UINT :=4;
END_VAR
```

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

```
R_TRIG0(CLK:= kaishi, Q=> );
IF R_TRIG0.Q THEN
    Result0:=StackPush(In:= In0,
                       InOut:= InOut0[2],
                       Size:= Size0,
                       Num:= Num0);
END_IF
```

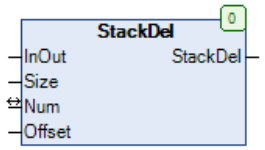
</td></tr>
</table>

| | |
|---|---|
| Operation result |  |

1-4-10-6. Delete stacked data instruction StackDel

(1) Instruction overview

| Delete stacked data instruction [StackDel] | | | |
|---|---|---|---|
| Execution condition | Edge triggering | Suitable model | XS, XSA |
| Firmware | - | Software | XS StudioV2.3.1 and above |
| Library | XJ_FunctionUtils | | |

| Instruction | Name | Graph | ST |
|---|---|---|---|
| StackDel | Delete stacked data instruction |  | StackDel(InOut:= , Size:= , Num:= , Offset:= ); |

(2) Related variables

| VAR_INPUT | Name | Data type | Effective range | Initial value | Description |
|---|---|---|---|---|---|
| Size | Number of stacked elements | UINT | As the data type | 1 | Number of stacked elements |
| Offset | Offset position | UINT | As the data type | 0 | The offset position of the element to be deleted in the stack |
| VAR_IN_OUT | Name | Data type | Effective range | Initial value | Description |
| InOut[] array | Stacked array | - | As the data type | - | Array that forms a stack |
| Num | The number of elements saved in the stack | UINT | As the data type | - | The number of elements saved in the stack |

(3) Function

- Consider the array InOut[] that currently holds the required number "Num" elements as a stacked array. Remove the value of InOut["Offset"] in the stack determined by the offset position from it.
- The higher position elements, namely InOut["Offset"+1] to InOut["Num" -1], move towards the lower position direction of the stacked array. Then, "Num" decreases by 1.
- The stacking element number "Size" is the number of elements to be used as stacking in InOut[].

(4) Note

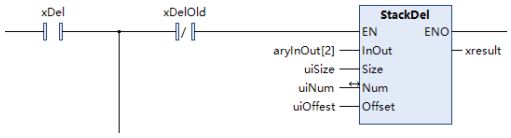
- When the value of 'Size' exceeds the array range of InOut[], the function returns a value of False, while the values of InOut[], 'Num', and 'OutVal' remain unchanged.
- When the value of "Size" is 0, the function returns a value of TRUE, while the values of InOut[] and "Num" remain unchanged.

- When the value of "Size" or "Num" is outside of 0 and "Size"≥"Num">"Offset" is false, the program cannot run, the function returns a value of False, and the values of InOut[] and "Num" remain unchanged.

(5) Program example

<table>
<tr><th>DEMO</th><th>LD</th><th>ST</th></tr>
<tr><td>Define the variables</td><td colspan="2">

```
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
```

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

```
IF xDel AND NOT xDelOld THEN
    xResult:=StackDel(InOut:=aryInOut[2] ,
        Size:=uiSize ,
        Num:=uiNum ,
        Offset:=uiOffest );
    ;
END_IF
xDelOld:=xDel;
```

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

| 表达式 | 类型 | 值 |
|---|---|---|
| aryInOut | ARRAY [1..10] OF INT | |
| aryInOut... | INT | 1 |
| aryInOut... | INT | 2 |
| aryInOut... | INT | 3 |
| aryInOut... | INT | 5 |
| aryInOut... | INT | 5 |
| aryInOut... | INT | 0 |
| aryInOut... | INT | 0 |
| aryInOut... | INT | 0 |
| aryInOut... | INT | 0 |
| aryInOut... | INT | 0 |
| uiNum | UINT | 3 |
| uiOffest | UINT | 2 |
| uiSize | UINT | 5 |
| xDel | BOOL | TRUE |
| xDelOld | BOOL | TRUE |
| xResult | BOOL | TRUE |

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

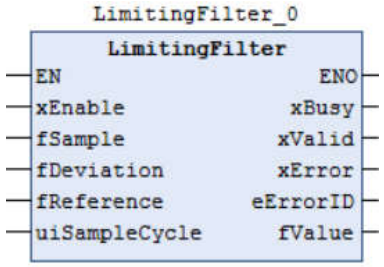
## 1-4-11. Filter command

### 1-4-11-1. Limiting filter LimitingFilter

(1) Instruction overview

When the xEnable level is enabled, set the number of scan cycles uiSampleCycle, the sampling reference value fReference and the maximum allowable deviation value fDeviation for two samplings , and then the sampling value fSample is inputted. Judge every time a new value is sampled:
If the difference between this value and the previous value is less than or equal to fDeviation, then this value is valid and output as a filtered value;
If the difference between the current value and the previous value is greater than fDeviation, the current value is invalid, the current value is discarded, and the previous value is used instead as the filtered value output.

| Limiting filter [LimitingFilter] | | | |
|---|---|---|---|
| Execution condition | Normally open/closed | Suitable model | XS, XSA |
| Firmware | - | Software | XS StudioV2.2.0 and above |
| Library | XJ_FunctionUtils | | |

| Instruction | Name | Graph | ST |
|---|---|---|---|
| LimitingFilter | Limiting filter |  | LimitingFilter_0(<br>xEnable:= ,<br>fSample:= ,<br>fDeviation:= ,<br>fReference:= ,<br>uiSampleCycle:= ,<br>xBusy=> ,<br>xValid=> ,<br>xError=> ,<br>eErrorID=> ,<br>fValue=> ); |

(2) Related variables

| VAR_INPUT | Name | Data type | Effective range | Initial value | Description |
|---|---|---|---|---|---|
| xEnable | Filter enable | BOOL | TRUE/FALSE | FALSE | TRUE: Execute function block<br>FALSE: Do not execute function block |
| fSample | Input sampling value | REAL | - | 0 | Input sampling value |
| fDeviation | Input the maximum allowable deviation value for two | REAL | - | 0 | Input the maximum allowable deviation value for two samplings |

| | | | | | |
|---|---|---|---|---|---|
| | samplings | | | | |
| fReference | Input reference value | REAL | - | 0 | Input reference value |
| uiSampleCycle | Input sampling period | UINT | 1-1000 | 0 | Input sampling period |
| VAR_OUTPUT | Name | Data type | Effective range | Initial value | Description |
| xBusy | The instruction is being executed | BOOL | TRUE/FALSE | FALSE | TRUE: function block execution in progress |
| xValid | Output valid | BOOL | TRUE/FALSE | FALSE | TRUE: instruction execution is effective |
| xError | Error | BOOL | TRUE/FALSE | FALSE | TRUE: abnormal occurrence |
| eErrorID | Error ID | UTIL_ERROR | - | 0 | Error ID |
| fValue | Filter output effective value | REAL | - | 0 | Filter output effective value |

(3) Function

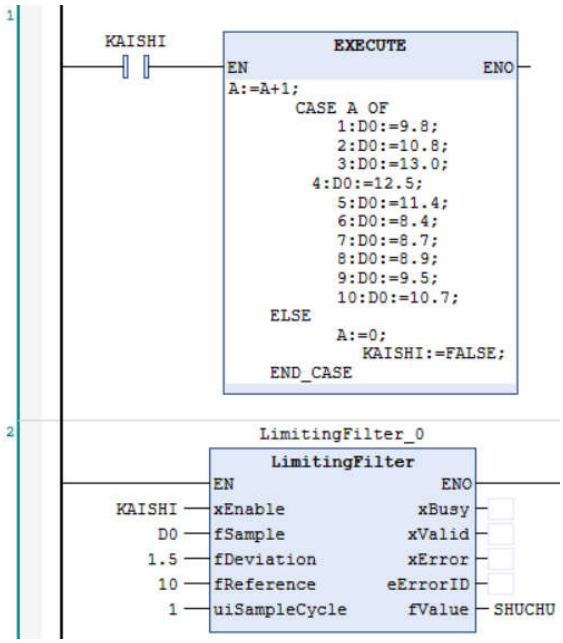
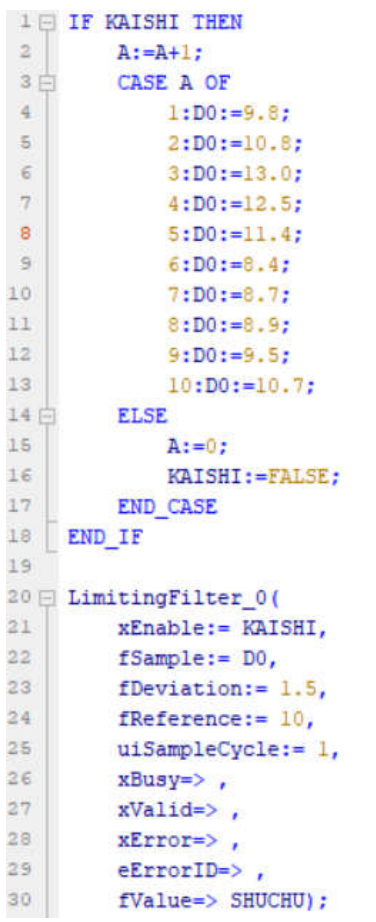
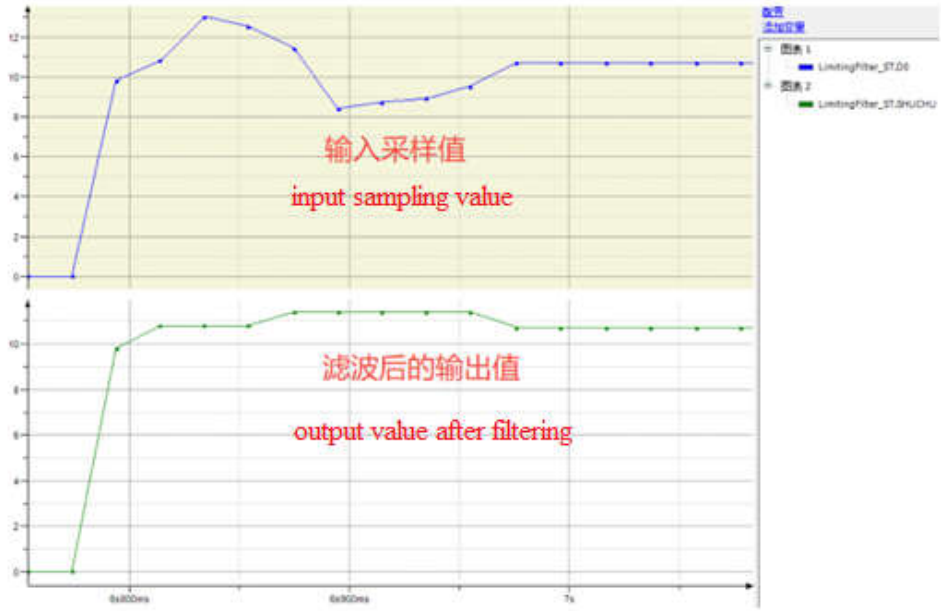
- Advantages: It can effectively overcome pulse interference caused by accidental factors.
- Disadvantages: Unable to suppress periodic interference and poor smoothness.

(4) Note

- The input parameter uiSampleCycle in the filtering function block is the sampling period. If uiSampleCycle=1, it means that sampling operations are performed for each period; If uiSampleCycle=3, it means sampling is done every 3 cycles. If the flag of the function block LimitingFilter.xValid outputs TRUE, it indicates that the filtering is valid. The filtered output value fValue will update the output when the filtered output is valid.
- If the sampled data is stored in an array, the data in the array needs to be input into the filtering function block one by one, and the user needs to process the data themselves.

(5) Program example

| DEMO | LD | ST |
|---|---|---|
| Define the variables | VAR<br>A:INT;<br>D0,SHUCHU:REAL;<br>KAISHI:BOOL;<br>LimitingFilter_0:LimitingFilter;<br>END_VAR | |

| | | |
|---|---|---|
| Program example |  |  |
| Operation result |  | |
| Result analysis | For 10 consecutive cycles, the sampling values are set to (9.8, 10.8, 13.0, 12.5, 11.4, 8.4, 8.7, 8.9, 9.5, 10.7).<br>The amplitude reference value is 10.0 (given reference value), the maximum allowable deviation value for amplitude is 1.5 (deviation range), and the filtering calculation cycle is 1 (given the number of operating cycles, perform sampling filtering once). When xEnable is set to TRUE, the limiting filtering function block starts sampling filtering and | |

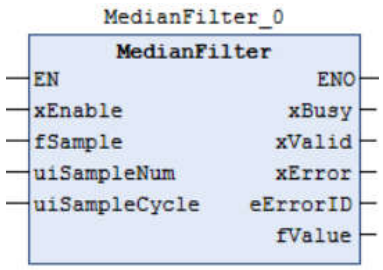
| | |
|---|---|
| | compares the sampled value with the last valid value. If the absolute value of the difference between the two is greater than the set maximum deviation value of 1.5, the last valid value is the current valid value; If it is less than or equal to the maximum deviation value, the current value is the valid value, and the valid value is output as the filtering value. The filtering execution valid flag xValid is set to TRUE.<br>As can be seen, after filtering, the results are (9.8, 10.8, 10.8, 10.8, 11.4, 11.4, 11.4, 11.4, 10.7). |

1-4-11-2. Median filter MedianFilter

(1) Instruction overview

When the xEnable level is enabled, set the number of scan cycles uiSampleCycle, the sampling numbers uiSampleNum, input sampling variable fValue. Continuously sample uiSampleNum times, arrange the uiSampleNum values by size, and take the middle value as the current filtering value.

| Median filter [MedianFilter] | | | |
|---|---|---|---|
| Execution condition | Normally open/closed | Suitable model | XS, XSA |
| Firmware | - | Software | XS StudioV2.2.0 and above |
| Library | XJ_FunctionUtils | | |

| Instruction | Name | Graph | ST |
|---|---|---|---|
| MedianFilter | Median filter |  | MedianFilter_0(<br>xEnable:= ,<br>fSample:= ,<br>uiSampleNum:= ,<br>uiSampleCycle:= ,<br>xBusy=> ,<br>xValid=> ,<br>xError=> ,<br>eErrorID=> ,<br>fValue=> ); |

(2) Related variables

| VAR_INPUT | Name | Data type | Effective range | Initial value | Description |
|---|---|---|---|---|---|
| xEnable | Filter enable | BOOL | TRUE/FALSE | FALSE | TRUE: Execute function block<br>FALSE: Do not execute function block |
| fSample | Input sampling value | REAL | - | 0 | Input sampling value |
| uiSampleNum | Input sampling number N | UINT | 1-1000 | 0 | Input sampling number N |
| uiSampleCycle | Input sampling period | UINT | 1-1000 | 0 | Input sampling period |
| VAR_OUTPUT | Name | Data type | Effective range | Initial value | Description |
| xBusy | The instruction is being executed | BOOL | TRUE/FALSE | FALSE | TRUE: Function block execution in progress |
| xValid | Output is valid | BOOL | TRUE/FALSE | FALSE | TRUE: Instruction execution is effective |
| xError | Error | BOOL | TRUE/FALSE | FALSE | TRUE: Abnormal occurrence |
| eErrorID | Error ID | UTIL_ERROR | - | 0 | Error ID |

| | | | | | |
|---|---|---|---|---|---|
| fValue | Filter output effective value | REAL | - | 0 | Filter output effective value |

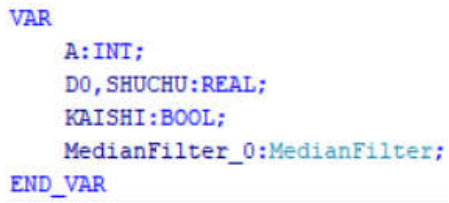
(3) Function

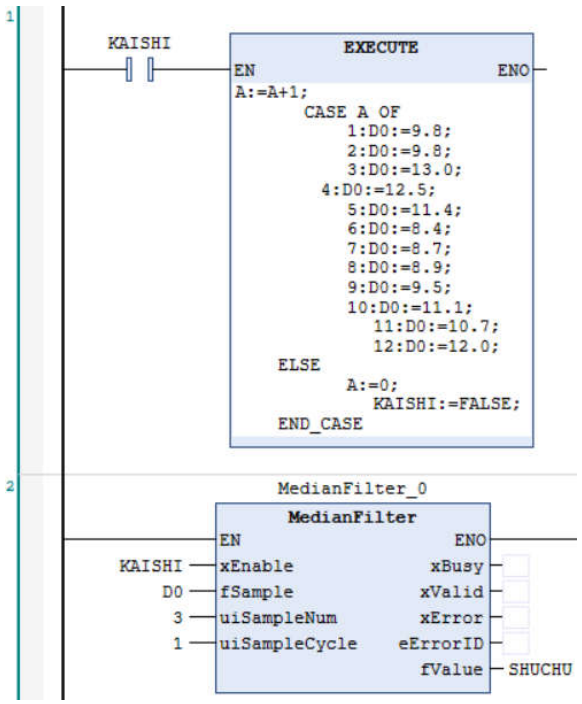
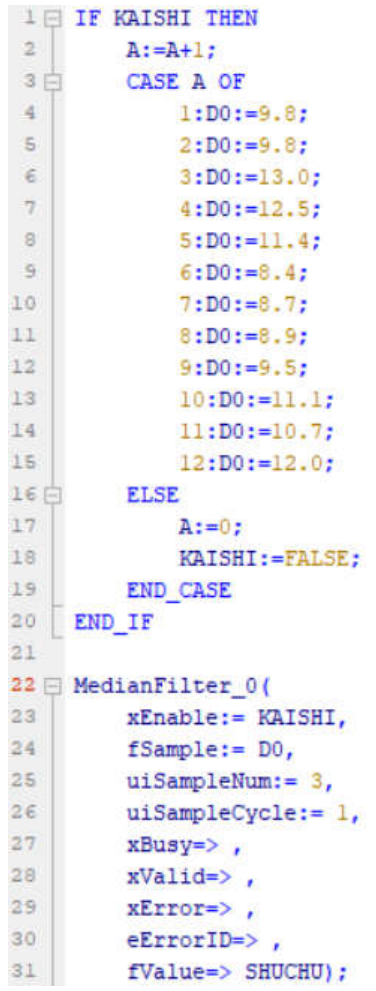
- Advantages: It can effectively overcome the interference of fluctuating pulses caused by accidental factors; Has good filtering effect on measured parameters with slow changes in temperature and liquid level.
- Disadvantages: Not suitable for rapidly changing parameters such as flow and speed.

(4) Note

- The input parameter uiSampleCycle in the filtering function block is the sampling period. If uiSampleCycle=1, it means that sampling operations are performed for each period; If uiSampleCycle=3, it means sampling is done every 3 cycles. If MedianFilter.xValid outputs TRUE, it indicates that the filtering is valid. The filtered output value fValue will update the output when the filtered output is valid.
- If the sampled data is stored in an array, the data in the array needs to be input into the filtering function block one by one, and the user needs to process the data themselves.

(5) Program example

| DEMO | LD | ST |
|---|---|---|
| Define the variables |  | |

| | | |
|---|---|---|
| Program example |  |  |
| Operation result | | |
| Result analysis | For 12 consecutive cycles, the sampling values are set to (9.8, 9.8, 13.0, 12.5, 11.4, 8.4, 8.7, 8.9, 9.5, 11.1, 10.7, 12.0)<br>The number of samples for sampling calculation is 3, and the sampling period is 1 (given the number of running cycles, perform sampling filtering once). When the input xEnable | |

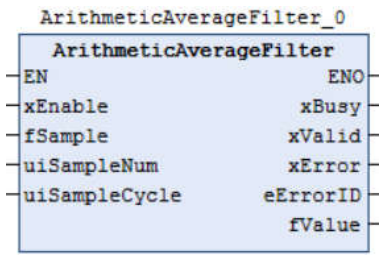
| | |
|---|---|
| | is set to TRUE, sort the 3 sampled values by size, and then take the middle value as the filter value output. The filter execution effective flag xValid is set to TRUE.<br>As can be seen, after filtering, the results are (0, 0, 9.8, 9.8, 9.8, 11.4, 11.4, 11.4, 8.9, 8.9, 8.9, 11.1). |

1-4-11-3. Arithmetic average filter ArithmeticAverageFilter

(1) Instruction overview

When the xEnable is enabled, set the number of scan cycles uiSampleCycle, the number of sampling times uiSampleNum, input sampling variable fValue. Continuously sample uiSampleNum times, perform arithmetic mean operation, and output the filtered value.

| Arithmetic average filter [ArithmeticAverageFilter] | | | |
|---|---|---|---|
| Execution condition | Normally open/closed | Suitable model | XS, XSA |
| Firmware | - | Software | XS StudioV2.2.0 and above |
| Library | XJ_FunctionUtils | | |

| Instruction | Name | Graph | ST |
|---|---|---|---|
| ArithmeticAverageFilter | Arithmetic average filter |  | ArithmeticAverageFilter_0(<br>xEnable:= ,<br>fSample:= ,<br>uiSampleNum:= ,<br>uiSampleCycle:= ,<br>xBusy=> ,<br>xValid=> ,<br>xError=> ,<br>eErrorID=> ,<br>fValue=> ); |

(2) Related variables

| VAR_INPUT | Name | Data type | Effective range | Initial value | Description |
|---|---|---|---|---|---|
| xEnable | Filter enable | BOOL | TRUE/FALSE | FALSE | TRUE: Execute function block<br>FALSE: Do not execute function block |
| fSample | Input sampling value | REAL | - | 0 | Input sampling value |
| uiSampleNum | Input sampling number N | UINT | 1-1000 | 0 | Input sampling number N |
| uiSampleCycle | Input sampling period | UINT | 1-1000 | 0 | Input sampling period |
| VAR_OUTPUT | Name | Data type | Effective range | Initial value | Description |
| xBusy | The instruction is being executed | BOOL | TRUE/FALSE | FALSE | TRUE: Function block execution in progress |
| xValid | Output is valid | BOOL | TRUE/FALSE | FALSE | TRUE: Instruction execution is effective |
| xError | Error | BOOL | TRUE/FALSE | FALSE | TRUE: Abnormal occurrence |

| | | | | | |
|---|---|---|---|---|---|
| eErrorID | Error ID | UTIL_ERROR | - | 0 | Error ID |
| fValue | Filter output effective value | REAL | - | 0 | Filter output effective value |

(3) Function

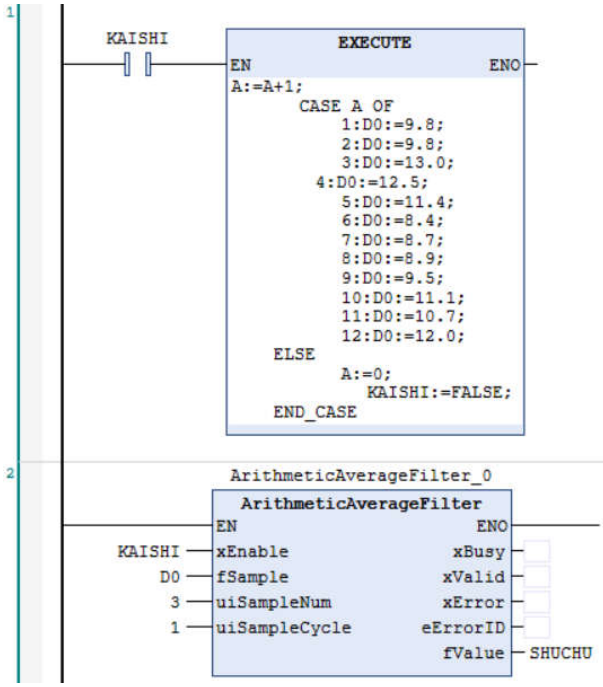
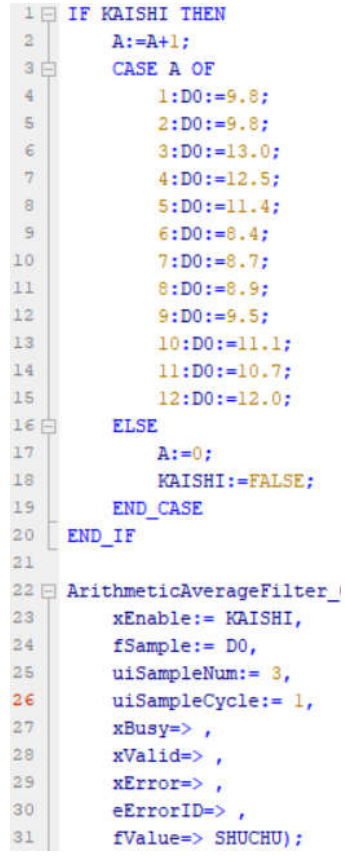
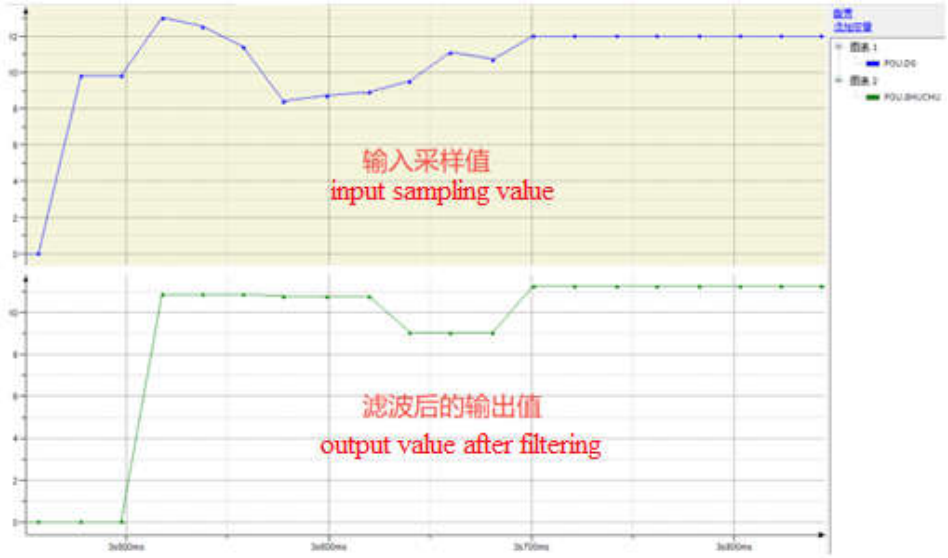
- When the input xEnable is set to TRUE, the function block can effectively start sampling and filtering. The number of samples for sampling calculation is set to N (i.e. filtering calculation is performed once every N sampling values). The arithmetic mean of N sampling values is taken as the filtering value output, and the effective flag xValid is set to TRUE for filtering execution.
- Advantages: Suitable for filtering general random interference signals, which have an average value and fluctuate up and down around a certain range of values.
- Disadvantages: Not suitable for real-time control with slow measurement speed or fast data calculation speed requirements, which wastes RAM.
- The larger the uiSampleNum value, the smoother the signal, but the lower the sensitivity.
- The smaller the uiSampleNum value, the lower the signal smoothness, but the higher the sensitivity.
- Selection of uiSampleNum value: general flow, uiSampleNum=12; Pressure: uiSampleNum=4.

(4) Note

- The input parameter uiSampleCycle in the filtering function block is the sampling period. If uiSampleCycle=1, it means that sampling operations are performed for each period; If uiSampleCycle=3, it means sampling is done every 3 cycles. The flag of the function block is ArithmeticAverageFilter. xValid is TRUE indicates that the filtering is effective. The filtered output value fValue will update the output when the filtered output is valid.
- If the sampled data is stored in an array, the data in the array needs to be input into the filtering function block one by one, and the user needs to process the data themselves.
- The input sampling number uiSampleNum parameter is not within the valid range, xError is set to TRUE, and the fault code eErrorID of the function block prompts the error message SAMPLENUM_PARAMETER_SET_ERROR. The parameter needs to be reset to be within the valid range and needs to be triggered again to reset the fault and xError signals of the function block.

(5) Program example

| DEMO | LD | ST |
|---|---|---|
| Define the variables | | |

| DEMO | LD | ST |
|---|---|---|
| Program example |  |  |
| Operation result |  | |
| Result analysis | For 12 consecutive cycles, the sampling values are set to (9.8, 9.8, 13.0, 12.5, 11.4, 8.4, 8.7, 8.9, 9.5, 11.1, 10.7, 12.0).<br>The number of samples for sampling calculation is set to 3, and the sampling period is 1 (1 data sampling is performed once per cycle). When the input xEnable is set to TRUE, the function block can effectively start sampling and filtering. The arithmetic mean of three sampling values is taken as the filtering value output, and the effective flag xValid is set to TRUE for filtering execution.<br>As can be seen, after filtering, the results are (0, 0, 10.87, 10.87, 10.77, 10.77, 10.77, 9.03, | |

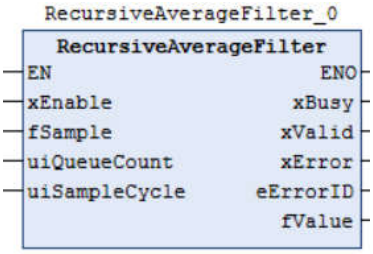
| DEMO | LD | ST |
|---|---|---|
| | 9.03, 11.27). | |

1-4-11-4. Recursive average filter RecursiveAverageFilter

(1) Instruction overview

When the xEnable is enabled, set the number of scan cycles uiSampleCycle, the queue length uiQueueCount, input sampling variable fSample. Consider consecutive uiQueueCount samples as a queue, with a fixed length of uiQueueCount. Each time a new data is sampled and placed at the end of the queue, the original data at the beginning of the queue is discarded (first in, first out principle), and then the N data in the queue are arithmetic averaged to obtain a new filtering result.

| Recursive average filter [RecursiveAverageFilter] | | | |
|---|---|---|---|
| Execution condition | Normally open/closed | Suitable model | XS, XSA |
| Firmware | - | Software | XS StudioV2.2.0 and above |
| Library | XJ_FunctionUtils | | |

| Instruction | Name | Graph | ST |
|---|---|---|---|
| RecursiveAverageFilter | Recursive average filter |  | RecursiveAverageFilter_0(<br>xEnable:= ,<br>fSample:= ,<br>uiQueueCount:= ,<br>uiSampleCycle:= ,<br>xBusy=> ,<br>xValid=> ,<br>xError=> ,<br>eErrorID=> ,<br>fValue=> ); |

(2) Related variables

| VAR_INPUT | Name | Data type | Effective range | Initial value | Description |
|---|---|---|---|---|---|
| xEnable | Filter enable | BOOL | TRUE/FALSE | FALSE | TRUE: Execute function block<br>FALSE: Do not execute function block |
| fSample | Input sampling value | REAL | - | 0 | Input sampling value |
| uiQueueCount | Input queue length N | UINT | 1-3000 | 0 | Input queue length N |
| uiSampleCycle | Input sampling period | UINT | 1-1000 | 0 | Input sampling period |
| VAR_OUTPUT | Name | Data type | Effective range | Initial value | Description |
| xBusy | The instruction is | BOOL | TRUE/FALSE | FALSE | TRUE: Function block |

| | | | | | |
|---|---|---|---|---|---|
| | being executed | | | | execution in progress |
| xValid | Output is valid | BOOL | TRUE/FALSE | FALSE | TRUE: Instruction execution is effective |
| xError | Error | BOOL | TRUE/FALSE | FALSE | TRUE: Abnormal occurrence |
| eErrorID | Error ID | UTIL_ERROR | - | 0 | Error ID |
| fValue | Filter output effective value | REAL | - | 0 | Filter output effective value |

(3) Function

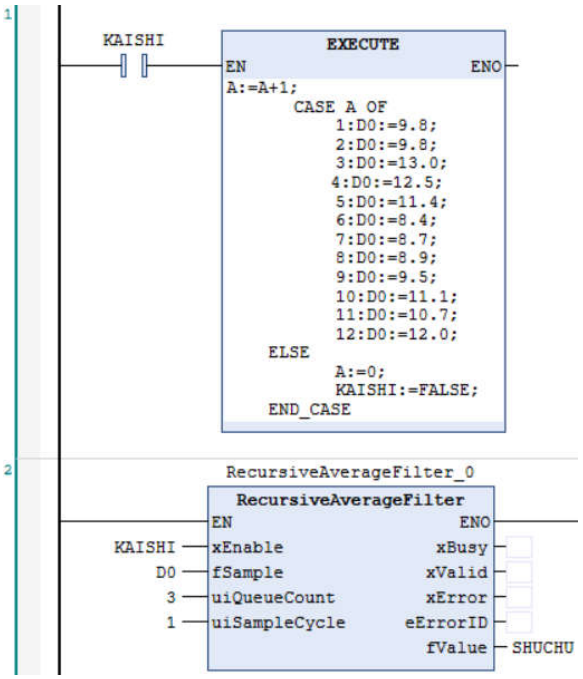
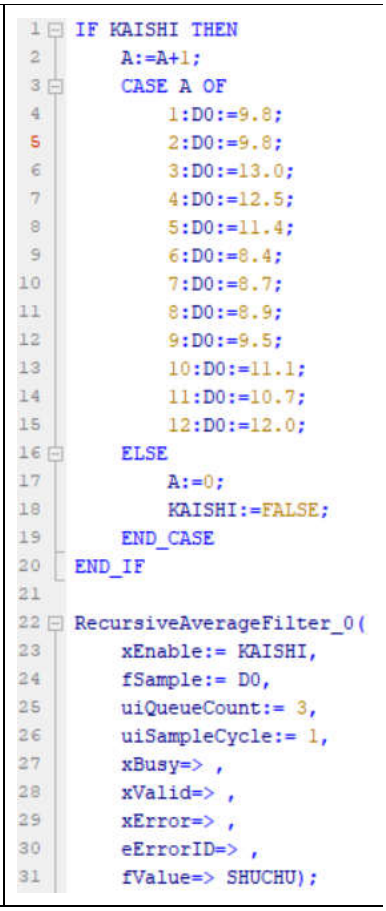
- The larger the uiQueueCount value, the smoother the signal, but the lower the sensitivity;
- The smaller the uiQueueCount value, the lower the signal smoothness, but the higher the sensitivity.
- Selection of uiQueueCount value: General flow: uiQueueCount=12; Pressure: uiQueueCount=4; liquid surface: uiQueueC-ount=4-12; Temperature: uiQueueCount=4.
- Advantages: It has a good inhibitory effect on periodic interference and high smoothness; Suitable for systems with high-frequency oscillations.
- Disadvantages: Low sensitivity and poor suppression of occasional pulse interference; Difficult to eliminate sampling value deviation caused by pulse interference; Not suitable for situations with severe pulse interference; Wasting RAM.

(4) Note

- The input parameter uiSampleCycle in the filtering function block is the sampling period. If uiSampleCycle=1, it means that sampling is performed every cycle. If uiSampleCycle=3, it means that sampling is performed every 3 cycles. RecursiveAverageFilter.xValid outputs TRUE to indicate that the filter is valid. The filtered output value fValue will update the output when the filtered output is valid.
- If the sampled data is stored in an array, the data in the array needs to be input into the filtering function block one by one, and the user needs to process the data themselves.

(5) Program example

| DEMO | LD | ST |
|---|---|---|
| Define the variables | VAR<br>A:INT;<br>D0,SHUCHU:REAL;<br>KAISHI:BOOL;<br>RecursiveAverageFilter_0:RecursiveAverageFilter;<br>END_VAR | |

| | | |
|---|---|---|
| Program example |  |  |
| Operation result | | |
| Result analysis | For 12 consecutive cycles, the sampling values are set to (9.8, 9.8, 13.0, 12.5, 11.4, 8.4, 8.7, 8.9, 9.5, 11.1, 10.7, 12.0).<br>The number of samples for sampling calculation is set to 3, and the sampling period is 1 (1 data sampling is performed once per cycle). When the input xEnable is set to TRUE, the function block can effectively start sampling and filtering. Each time a new data is sampled and placed at the end of the queue, and the original data at the beginning of the queue is discarded (first in first out principle). The arithmetic mean of the three sampled values in | |

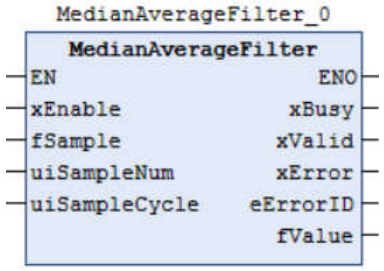
| | |
|---|---|
| | the queue is output as the filtering value, and the effective flag xValid is set to TRUE.<br>As can be seen, after filtering, the results are (0, 0, 10.87, 11.77, 12.30, 10.77, 9.50, 8.67, 9.03, 9.83, 10.43, 11.27). |

### 1-4-11-5. Median average filter MedianAverageFilter

(1) Instruction overview

When the xEnable is enabled, set the number of scan cycles uiSampleCycle, the number of samples uiSampleNum, input sampling variable fSample. Continuously sample uiSampleNum times, sort the group of data, remove the maximum and minimum values, and take the average, which is equivalent to the "median filtering method"+"arithmetic mean filtering method". The uiSampleNum value is usually selected from 3 to 14.

| Median average filter [MedianAverageFilter] | | | |
|---|---|---|---|
| Execution condition | Normally open/closed | Suitable model | XS, XSA |
| Firmware | - | Software | XS StudioV2.2.0 and above |
| Library | XJ_FunctionUtils | | |

| Instruction | Name | Graph | ST |
|---|---|---|---|
| MedianAverageFilter | Median average filter |  | MedianAverageFilter_0(<br>xEnable:= ,<br>fSample:= ,<br>uiSampleNum:= ,<br>uiSampleCycle:= ,<br>xBusy=> ,<br>xValid=> ,<br>xError=> ,<br>eErrorID=> ,<br>fValue=> ); |

(2) Related variables

| VAR_INPUT | Name | Data type | Effective range | Initial value | Description |
|---|---|---|---|---|---|
| xEnable | Filter enable | BOOL | TRUE/FALSE | FALSE | TRUE: Execute function block<br>FALSE: Do not execute function block |
| fSample | Input sampling value | REAL | - | 0 | Input sampling value |
| uiSampleNum | Input sampling times | UINT | 3-3000 | 0 | Input sampling times |
| uiSampleCycle | Input sampling | UINT | 1-1000 | 0 | Input sampling period |

| | period | | | | |
|---|---|---|---|---|---|
| VAR_OUTPUT | Name | Data type | Effective range | Initial value | Description |
| xBusy | The instruction is being executed | BOOL | TRUE/FALSE | FALSE | TRUE: Function block execution in progress |
| xValid | Output is valid | BOOL | TRUE/FALSE | FALSE | TRUE: Instruction execution is effective |
| xError | Error | BOOL | TRUE/FALSE | FALSE | TRUE: Abnormal occurrence |
| eErrorID | Error ID | UTIL_ERROR | - | 0 | Error ID |
| fValue | Filter output effective value | REAL | - | 0 | Filter output effective value |

(3) Function

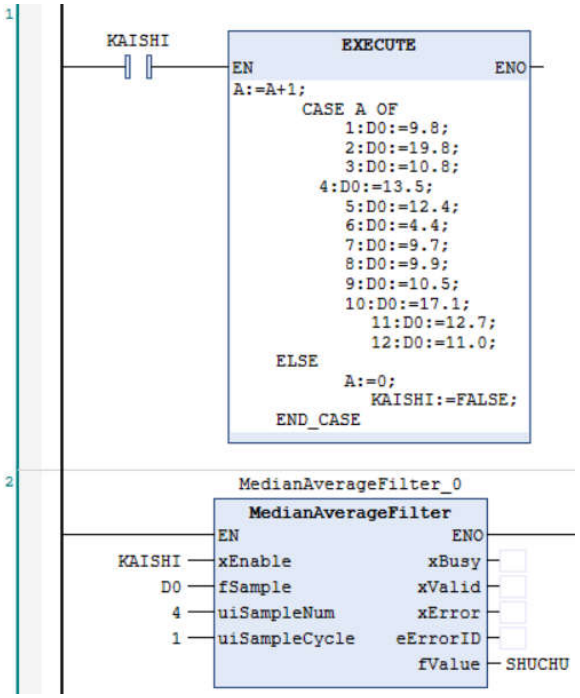
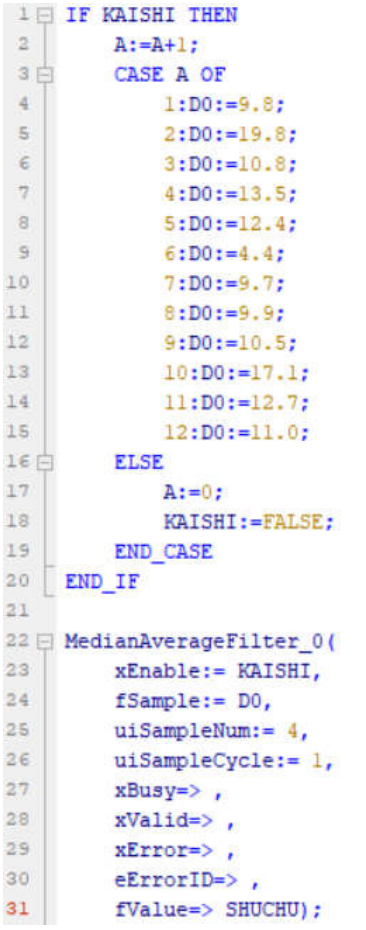
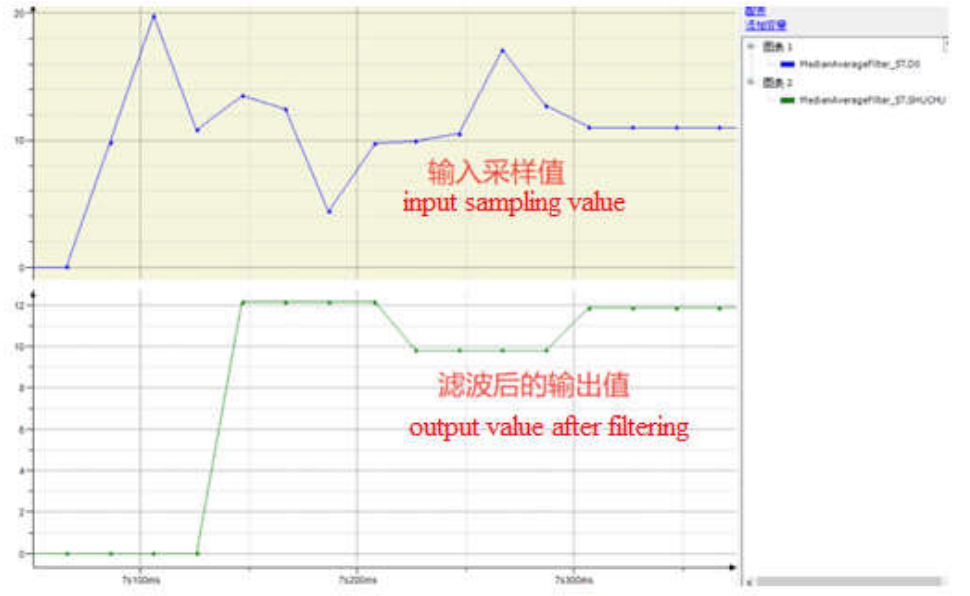
- Advantages: It combines the advantages of two filtering methods: median filtering and arithmetic mean filtering; For occasional pulse interference, the sampling value deviation caused by it can be eliminated; Has a good inhibitory effect on periodic interference; High smoothness, suitable for systems with high-frequency oscillation.
- Disadvantages: Slow calculation speed, similar to arithmetic mean filtering method; Wasting RAM.

(4) Note

- The input parameter uiSampleCycle in the filtering function block is the sampling period. If uiSampleCycle=1, it means that sampling is performed every cycle. If uiSampleCycle=3, it means that sampling is performed every 3 cycles. If MedianAverageFilter.xValid outputs TRUE, it indicates that the filter is valid. The filtered output value fValue will update the output when the filtered output is valid.
- If the sampled data is stored in an array, the data in the array needs to be input into the filtering function block one by one, and the user needs to process the data themselves.

(5) Program example

| DEMO | LD | ST |
|---|---|---|
| Define the variables | VAR<br>A:INT;<br>D0,SHUCHU:REAL;<br>KAISHI:BOOL;<br>MedianAverageFilter_0:MedianAverageFilter;<br>END_VAR | |

| | | |
|---|---|---|
| Program example |  |  |
| Operation result |  | |
| Result analysis | For 12 consecutive cycles, the sampling values are set to (9.8, 19.8, 10.8, 13.5, 12.4, 4.4, 9.7, 9.9, 10.5, 17.1, 12.7, 11.0).<br>The number of samples for sampling calculation is set to 4, and the sampling period is 1 (1 data sampling is performed once per cycle). When xEnable is set to TRUE, the function block can effectively start sampling filtering, sort the 4 sampled values by size, and then remove the maximum and minimum values to calculate the arithmetic mean as the filtering | |

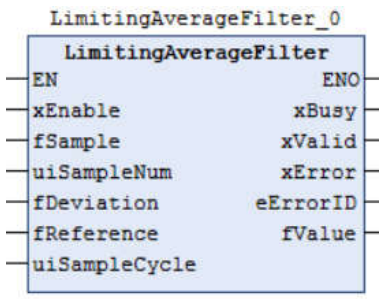
| | |
|---|---|
| | value output. The filtering execution flag xValid is set to TRUE.<br>As can be seen, after filtering, the results are (0, 0, 0, 12.15, 12.15, 12.15, 9.80, 9.80, 9.80, 9.80, 11.85). |

### 1-4-11-6. Limiting average filter LimitingAverageFilter

(1) Instruction overview

When the xEnable is enabled, set the number of scan cycles uiSampleCycle, the number of samples uiSampleNum, the sampling reference effective value fReference, and the maximum allowable deviation value fDeviation for two samplings, input sampling variable fSample. The filtering method is equivalent to "amplitude limiting filtering method"+"recursive average filtering method". Each time new data is sampled, it is first subjected to amplitude limiting processing and then sent to the queue for recursive average filtering processing.

| Limiting average filter [LimitingAverageFilter] | | | |
|---|---|---|---|
| Execution condition | Normally open/closed | Suitable model | XS, XSA |
| Firmware | - | Software | XS StudioV2.2.0 and above |
| Library | XJ_FunctionUtils | | |

| Instruction | Name | Graph | ST |
|---|---|---|---|
| LimitingAverageFilter | Limiting average filter |  | MedianAverageFilter_0(<br>xEnable:= ,<br>fSample:= ,<br>uiSampleNum:= ,<br>uiSampleCycle:= ,<br>xBusy=> ,<br>xValid=> ,<br>xError=> ,<br>eErrorID=> ,<br>fValue=> ); |

(2) Related variables

| VAR_INPUT | Name | Data type | Effective range | Initial value | Description |
|---|---|---|---|---|---|
| xEnable | Filter enable | BOOL | TRUE/FALSE | FALSE | TRUE: Execute function block<br>FALSE: Do not execute function block |
| fSample | Input sampling value | REAL | - | 0 | Input sampling value |
| uiSampleNum | Input sampling times | UINT | 1-1000 | 0 | Input sampling times |
| fDeviation | Input the maximum allowable | REAL | - | 0 | Input the maximum allowable deviation value for two samplings |

| | | | | | |
|---|---|---|---|---|---|
| | deviation value for two samplings | | | | |
| fReference | Input valid value | REAL | - | 0 | Input valid value |
| uiSampleCycle | Input sampling period | UINT | 1-1000 | 0 | Input sampling period |
| VAR_OUTPUT | Name | Data type | Effective range | Initial value | Description |
| xBusy | The instruction is being executed | BOOL | TRUE/FALSE | FALSE | TRUE: Function block execution in progress |
| xValid | Output is valid | BOOL | TRUE/FALSE | FALSE | TRUE: Instruction execution is effective |
| xError | Error | BOOL | TRUE/FALSE | FALSE | TRUE: Abnormal occurrence |
| eErrorID | Error ID | UTIL_ERROR | - | 0 | Error ID |
| fValue | Filter output effective value | REAL | - | 0 | Filter output effective value |

(3) Function

- Advantages: It combines the advantages of "amplitude limiting filtering method" and "recursive average filtering method" in filtering; For occasional pulse interference, the sampling value deviation caused by it can be eliminated.
- Disadvantages: Wasting RAM.

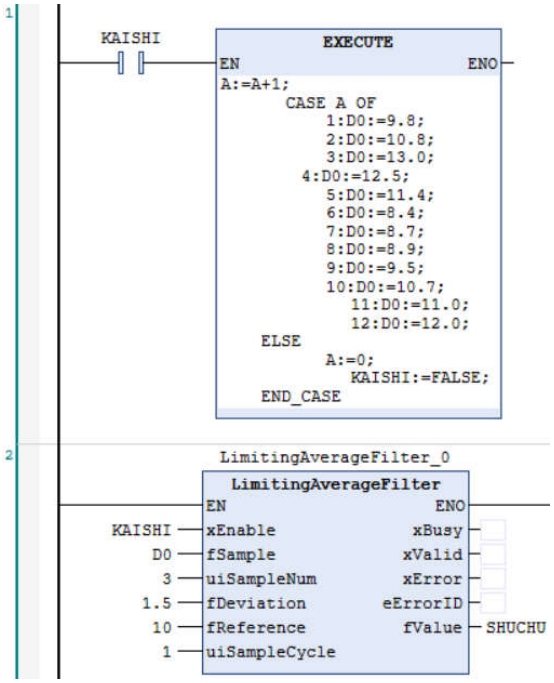
(4) Note

- The input parameter uiSampleCycle in the filtering function block is the sampling period. If uiSampleCycle=1, it means that sampling is performed every cycle. If uiSampleCycle=3, it means that sampling is performed every 3 cycles. If the flag of the function block LimitingAverageFilter.xValid outputs TRUE, it indicates that the filtering is valid. The filtered output value fValue will update the output when the filtered output is valid.
- If the sampled data is stored in an array, the data in the array needs to be input into the filtering function block one by one, and the user needs to process the data themselves.

(5) Program example

| DEMO | LD | ST |
|---|---|---|
| Define the variables | VAR<br>A:INT;<br>D0,SHUCHU:REAL;<br>KAISHI:BOOL;<br>LimitingAverageFilter_0:LimitingAverageFilter;<br>END_VAR | |

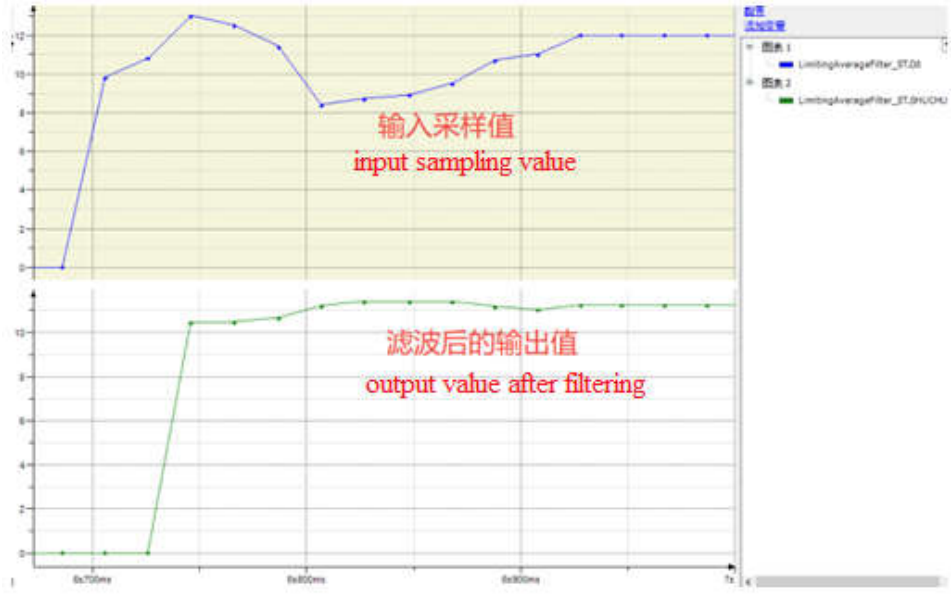
Program example




```
IF KAISHI THEN
    A:=A+1;
    CASE A OF
        1:D0:=9.8;
        2:D0:=19.8;
        3:D0:=10.8;
        4:D0:=13.5;
        5:D0:=12.4;
        6:D0:=4.4;
        7:D0:=9.7;
        8:D0:=9.9;
        9:D0:=10.5;
        10:D0:=17.1;
        11:D0:=12.7;
        12:D0:=11.0;
    ELSE
        A:=0;
        KAISHI:=FALSE;
    END_CASE
END_IF

MedianAverageFilter_0(
    xEnable:= KAISHI,
    fSample:= D0,
    uiSampleNum:= 4,
    uiSampleCycle:= 1,
    xBusy=> ,
    xValid=> ,
    xError=> ,
    eErrorID=> ,
    fValue=> SHUCHU);
```

Operation result




Result analysis

For 12 consecutive cycles, the sampling values are set to (9.8, 10.8, 13.0, 12.5, 11.4, 8.4, 8.7, 8.9, 9.5, 10.7, 11.0, 12.0).

The maximum allowable deviation value for amplitude is set to 1.5, the amplitude reference value is set to 10, the number of samples for sampling calculation is 3, and the sampling period is 1 (1 data sampling is performed once per cycle). When xEnable is set to TRUE,

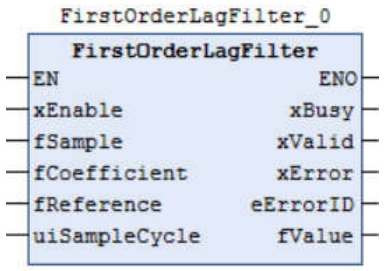
| | |
|---|---|
| | the function block can effectively start sampling and filtering. The initial reference value is set to 10.0 as the effective value, and the sampled value is compared with the last effective value. If the absolute value of the difference between the two is greater than the set maximum deviation value of 1.5, then the last valid value is the current valid value; If it is less than or equal to the maximum deviation value, the current value is considered valid. Finally, the effective value is output as the filtering value, and the filtering execution effective flag xValid is set to TRUE.<br>As can be seen, after filtering, the results are (0, 0, 10.47, 10.8, 11.0, 11.2, 11.4, 11.4, 11.4, 11.17, 11.03, 11.23). |

1-4-11-7. First-order lag filter FirstOrderLagFilter

(1) Instruction overview

When the xEnable is enabled, set the number of scan cycles uiSampleCycle, the sampling reference value fReference, the first-order low-pass filter coefficient fCoefficient (taken from 0 to 1), and the input sampling variable fSample. Calculate every time a new value is sampled:
The current filtering result = fCoefficient * the current sampling value + (1-fCoefficient) * the previous filtering result.

| First-order lag filter [FirstOrderLagFilter] | | | |
|---|---|---|---|
| Execution condition | Normally open/closed | Suitable model | XS, XSA |
| Firmware | - | Software | XS StudioV2.2.0 and above |
| Library | XJ_FunctionUtils | | |

| Instruction | Name | Graph | ST |
|---|---|---|---|
| FirstOrderLag Filter | First-order lag filter |  | FirstOrderLagFilter_0(<br>xEnable:= ,<br>fSample:= ,<br>fCoefficient:= ,<br>fReference:= ,<br>uiSampleCycle:= ,<br>xBusy=> ,<br>xValid=> ,<br>xError=> ,<br>eErrorID=> ,<br>fValue=> ); |

(2) Related variables

| VAR_INPUT | Name | Data type | Effective range | Initial value | Description |
|---|---|---|---|---|---|
| xEnable | Filter enable | BOOL | TRUE/FALSE | FALSE | TRUE: Execute function block<br>FALSE: Do not execute function block |
| fSample | Input sampling value | REAL | - | 0 | Input the value to be filtered |
| fCoefficient | Input first-order low-pass filter coefficient a=0~1 | REAL | 0-1 | 0 | Input first-order low-pass filter coefficient |
| fReference | Input valid value | REAL | - | 0 | Input valid value |
| uiSampleCycle | Input sampling cycle | UINT | 1-1000 | 0 | Input sampling cycle |
| VAR_OUTPUT | Name | Data type | Effective range | Initial value | Description |
| xBusy | The instruction is being executed | BOOL | TRUE/FALSE | FALSE | TRUE: Function block execution in progress |

| | | | | | |
|---|---|---|---|---|---|
| xValid | Output is valid | BOOL | TRUE/FALSE | FALSE | TRUE: Instruction execution is effective |
| xError | Error | BOOL | TRUE/FALSE | FALSE | TRUE: Abnormal occurrence |
| eErrorID | Error ID | UTIL_ERROR | - | 0 | Error ID |
| fValue | Filter output effective value | REAL | - | 0 | Filter output effective value |

(3) Function

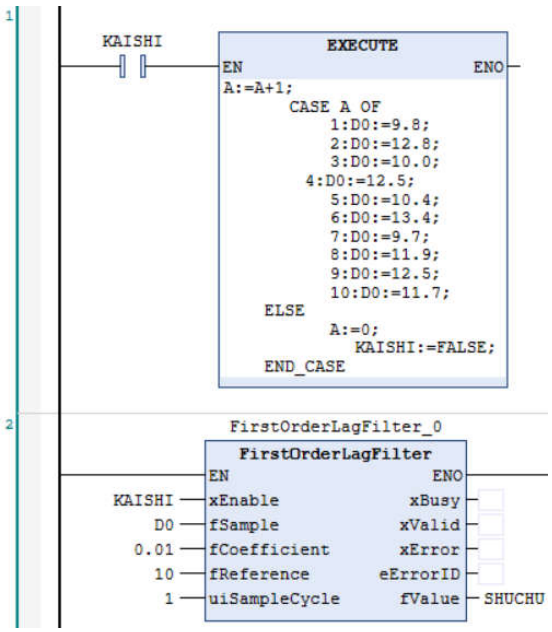
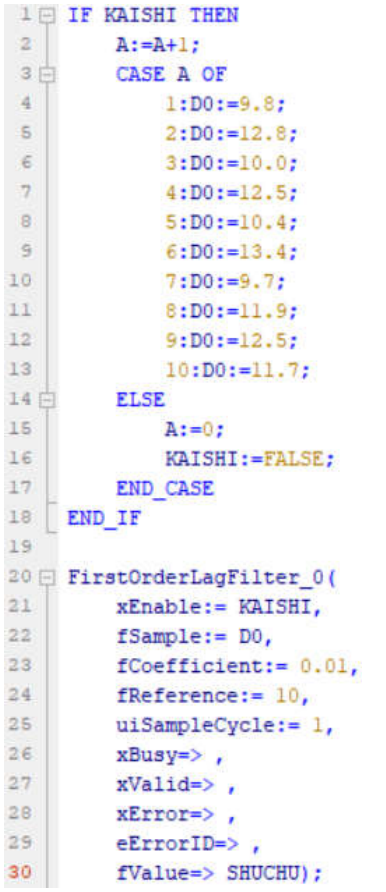
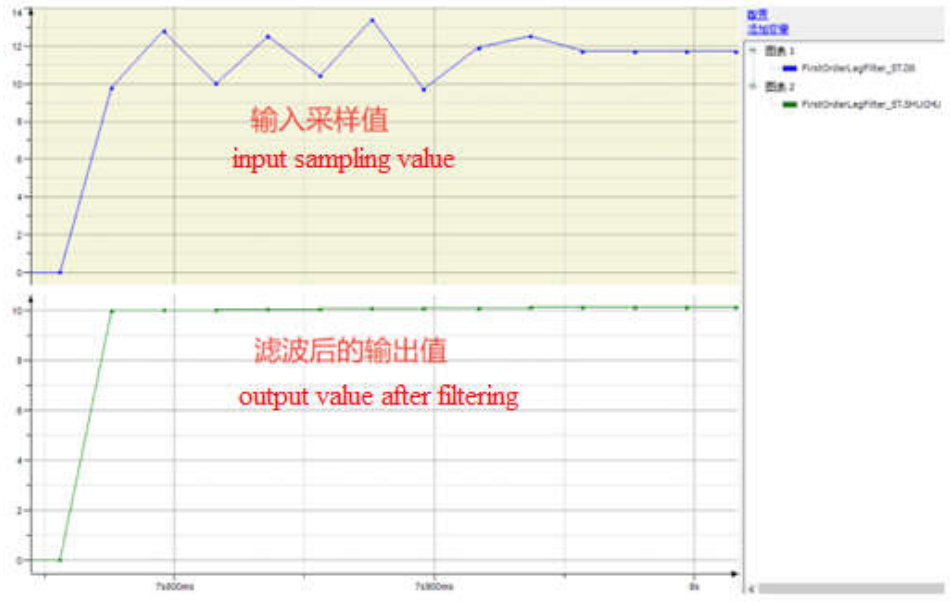
- Advantages: It has a good inhibitory effect on periodic interference; Suitable for situations with high fluctuation frequency.
- Disadvantages: Phase lag, low sensitivity; The lag program depends on the size of the fCoefficient value; Cannot eliminate interference signals with a filtering frequency higher than half of the sampling frequency.

(4) Note

- The input parameter uiSampleCycle in the filtering function block is the sampling period. If uiSampleCycle=1, it means that sampling operations are performed for each period; If uiSampleCycle=3, it means sampling is done every 3 cycles. If the flag of the function block FirstOrderLagFilter.xValid outputs TRUE, it indicates that the filtering is valid. The filtered output value fValue will update the output when the filtered output is valid.
- If the sampled data is stored in an array, the data in the array needs to be input into the filtering function block one by one, and the user needs to process the data themselves.

(5) Program example

| DEMO | LD | ST |
|---|---|---|
| Define the variables | VAR<br>A:INT;<br>D0,SHUCHU:REAL;<br>KAISHI:BOOL;<br>FirstOrderLagFilter_0:FirstOrderLagFilter;<br>END_VAR | |

| | | |
|---|---|---|
| Program example |  |  |
| Operation result |  | |
| Result analysis | For 10 consecutive cycles, the sampling values are set to (9.8, 12.8, 10.0, 12.5, 10.4, 13.4, 9.7, 11.9, 12.5, 11.7).<br>The first-order low-pass filtering coefficient is set to 0.01, the filtering reference value is set to 10, and the sampling period is 1 (data sampling is performed once per cycle). When the input xEnable is set to TRUE, the function block start sampling and filtering. The filtered value = fCoefficient * current sampling value + (1-fCoefficient) * last filtering result, and the filtered value is output. The filtering execution effective flag xValid is set to | |

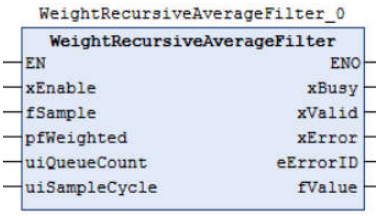
| | |
|---|---|
| | TRUE.<br>As can be seen, after filtering, the results are (9.998, 10.03, 10.03, 10.05, 10.05, 10.09, 10.08, 10.10, 10.13, 10.14). |

1-4-11-8. Weighted recursive average filter WeightRecursiveAverageFilter

(1) Instruction overview

When the xEnable is enabled, set the number of scan cycles uiSampleCycle, the queue length uiQueueCount, input sampling variable fSample and the address pfWeighted of the corresponding weighting coefficient array. This function is an improvement on the recursive average filtering method, which assigns different weights to data at different times. Usually, the closer the data is to the current time, the greater the weight. The greater the weight coefficient given to the new sampling value, the higher the sensitivity, but the lower the signal smoothness.

| Weighted recursive average filter [WeightRecursiveAverageFilter] | | | |
|---|---|---|---|
| Execution condition | Normally open/closed | Suitable model | XS, XSA |
| Firmware | - | Software | XS StudioV2.2.0 and above |
| Library | XJ_FunctionUtils | | |

| Instruction | Name | Graph | ST |
|---|---|---|---|
| WeightRecursiveAverageFilter | Weighted recursive average filter |  | WeightRecursiveAverageFilter_0(<br>xEnable:= ,<br>fSample:= ,<br>pfWeighted:= ,<br>uiQueueCount:= ,<br>uiSampleCycle:= ,<br>xBusy=> ,<br>xValid=> ,<br>xError=> ,<br>eErrorID=> ,<br>fValue=> ); |

(2) Related variables

| VAR_INPUT | Name | Data type | Effective range | Initial value | Description |
|---|---|---|---|---|---|
| xEnable | Filter enable | BOOL | TRUE/FALSE | FALSE | TRUE: Execute function block<br>FALSE: Do not execute function block |
| fSample | Input sampling value | REAL | - | 0 | Input the value to be filtered |
| pfWeighted | Input weighting coefficient and store in an array | POINTER TO REAL | - | 0 | Input weighting coefficient and store in an array, not all can be equal to 0 |
| uiQueueCount | Input queue length N | UINT | 1-3000 | | Input queue length N |

| | | | | | |
|---|---|---|---|---|---|
| uiSampleCycle | Input sampling cycle | UINT | 1-1000 | 0 | Input sampling cycle |
| VAR_OUTPUT | Name | Data type | Effective range | Initial value | Description |
| xBusy | The instruction is being executed | BOOL | TRUE/FALSE | FALSE | TRUE: Function block execution in progress |
| xValid | Output is valid | BOOL | TRUE/FALSE | FALSE | TRUE: Instruction execution is effective |
| xError | Error | BOOL | TRUE/FALSE | FALSE | TRUE: Abnormal occurrence |
| eErrorID | Error ID | UTIL_ERROR | - | 0 | Error ID |
| fValue | Filter output effective value | REAL | - | 0 | Filter output effective value |

(3) Function

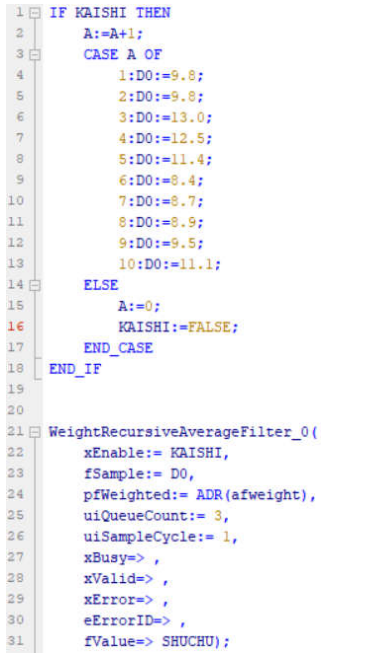
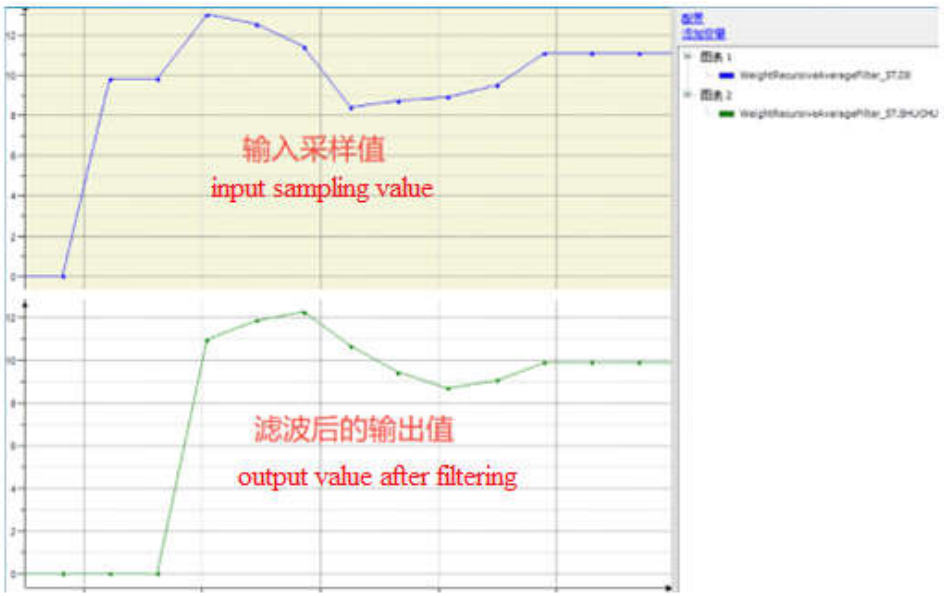
- Filter value = (weight 1 * sampling 1+weight 2 * sampling 2+...)/(weight 1+weight 2+...).
- Advantages: Suitable for objects with large pure lag time constants and systems with short sampling periods.
- Disadvantages: For signals with small pure lag time constants, long sampling periods, and slow changes; The severe program that cannot quickly reflect the current interference on the system has poor filtering effect.

(4) Note

- The input parameter uiSampleCycle in the filtering function block is the sampling period. If uiSampleCycle=1, it means that sampling operations are performed for each period; If uiSampleCycle=3, it means sampling is done every 3 cycles. WeightRecursiveAverageFilter.xValid outputs TRUE to indicate that the filtering is effective. The filtered output value fValue will update the output when the filtered output is valid.
- If the sampled data is stored in an array, the data in the array needs to be input into the filtering function block one by one, and the user needs to process the data themselves.
- The weighting coefficients cannot all be 0.
- The input queue length uiQueueCount cannot be greater than the length of the weighting coefficient.

(5) Program example

| DEMO | LD | ST |
|---|---|---|
| Define the variables | VAR<br>A:INT;<br>D0,SHUCHU:REAL;<br>KAISHI:BOOL;<br>afweight:ARRAY[0..2]OF REAL:=[1.0,1.1,1.2];<br>WeightRecursiveAverageFilter_0:WeightRecursiveAverageFilter;<br>END_VAR | |

| | | |
|---|---|---|
| Program example | |  |
| Operation result |  | |
| Result analysis | The weighting coefficients are (1.0, 1.1, 1.2), and the sampling values are set to (9.8, 9.8, 13.0, 12.5, 11.4, 8.4, 8.7, 8.9, 9.5, 11.1) for 10 consecutive cycles.<br>The sample queue for sampling calculation is set to 3, and the sampling period is 1 (data sampling is performed once per cycle). When the input xEnable is set to TRUE, the function block can effectively start sampling filtering. Each time a new data is sampled and placed at the end of the queue, and the original data at the beginning of the queue is discarded (first in, first out principle). The three sampled values in the queue are multiplied by their corresponding weighting coefficients, and then the sum is taken and divided by the sum of the weighting coefficients to obtain the sampled filtering value. The filtering value is output, and the effective flag xValid is set to TRUE.<br>As can be seen, after filtering, the results are (0, 0, 10.96, 11.85, 12.25, 10.64, 9.42, 8.68, 9.06, 9.9). | |

1-4-11-9. Debounce filter DebounceFilter

(1) Instruction overview

When the xEnable is enabled, set the number of scan cycles uiSampleCycle, the sampling reference value fReference, and a filtering upper limit value uiUpLimit, input sampling variable fSample. Compare each sampled value with the current valid value.

If the sampled value is equal to the current valid value, then the current count value is reset to zero. If the sampled value is less than or greater than the current valid value, the count value will be increased by 1, and it will be determined whether the count value has overflowed (count value>upper limit uiUpLimit). If the count value overflows, the current sampled value will be replaced with the current filtered value output, and the count value will be reset to zero.

| Debounce filter [DebounceFilter] | | | |
|---|---|---|---|
| Execution condition | Normally open/closed | Suitable model | XS, XSA |
| Firmware | - | Software | XS StudioV2.2.0 and above |
| Library | XJ_FunctionUtils | | |

| Instruction | Name | Graph | ST |
|---|---|---|---|
| DebounceFilter | Debounce filter | DebounceFilter_0<br>DebounceFilter<br>EN / ENO<br>xEnable / xBusy<br>fSample / xValid<br>uiUpLimit / xError<br>fReference / eErrorID<br>uiSampleCycle / fValue | DebounceFilter_0(<br>xEnable:= ,<br>fSample:= ,<br>uiUpLimit:= ,<br>fReference:= ,<br>uiSampleCycle:= ,<br>xBusy=> ,<br>xValid=> ,<br>xError=> ,<br>eErrorID=> ,<br>fValue=> ); |

(2) Related variables

| VAR_INPUT | Name | Data type | Effective range | Initial value | Description |
|---|---|---|---|---|---|
| xEnable | Filter enable | BOOL | TRUE/FALSE | FALSE | TRUE: Execute function block<br>FALSE: Do not execute function block |
| fSample | Input sampling value | REAL | - | 0 | Input the value to be filtered |
| uiUpLimit | Set the upper limit of the filter counter | UINT | 1-65535 | 0 | Set the upper limit of the filter counter |
| fReference | Input reference value | REAL | - | 0 | Input reference value |
| uiSampleCycle | Input sampling cycle | UINT | 1-1000 | 0 | Input sampling cycle |

| VAR_OUTPUT | Name | Data type | Effective range | Initial value | Description |
|---|---|---|---|---|---|
| xBusy | The instruction is being executed | BOOL | TRUE/FALSE | FALSE | TRUE: Function block execution in progress |
| xValid | Output is valid | BOOL | TRUE/FALSE | FALSE | TRUE: Instruction execution is effective |
| xError | Error | BOOL | TRUE/FALSE | FALSE | TRUE: Abnormal occurrence |
| eErrorID | Error ID | UTIL_ERROR | - | 0 | Error ID |
| fValue | Filter output effective value | REAL | - | 0 | Filter output effective value |

(3) Function

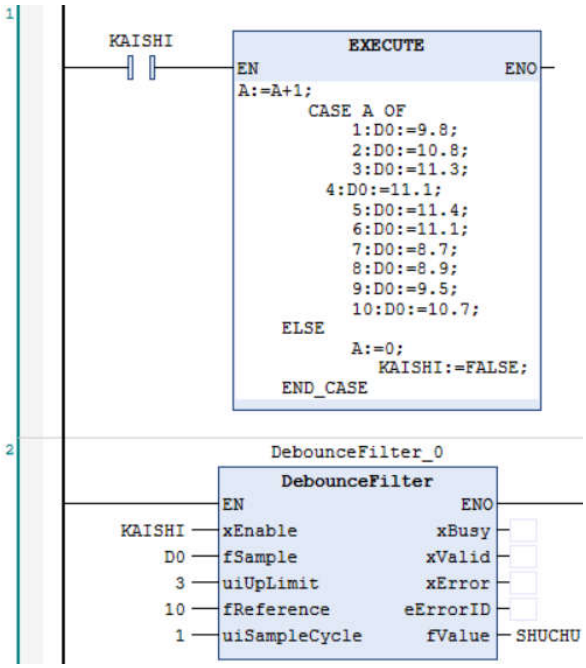
- If the sampled value is equal to the current valid value, then the current count value is reset to zero.
  If the sampled value is less than or greater than the current valid value, the count value will be increased by 1, and it will be determined whether the count value has overflowed (count value>upper limit uiUpLimit). If the count value overflows, the current sampled value will be replaced with the current filtered value output, and the count value will be reset to zero.
- Advantages: It has good filtering effect on slowly changing measured parameters; It can avoid repeated on/off jumping of the controller or numerical jitter on the display near the critical value.
- Disadvantages: Not suitable for rapidly changing parameters; If the value sampled during the counter overflow happens to be an interference value, the interference value will be imported into the system as a valid value.

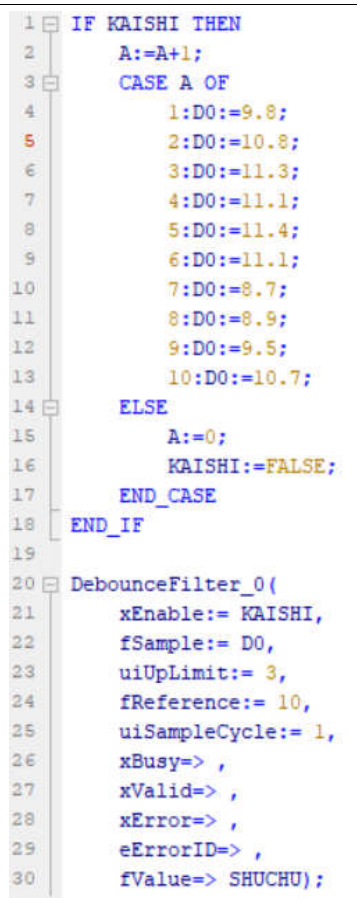
(4) Note

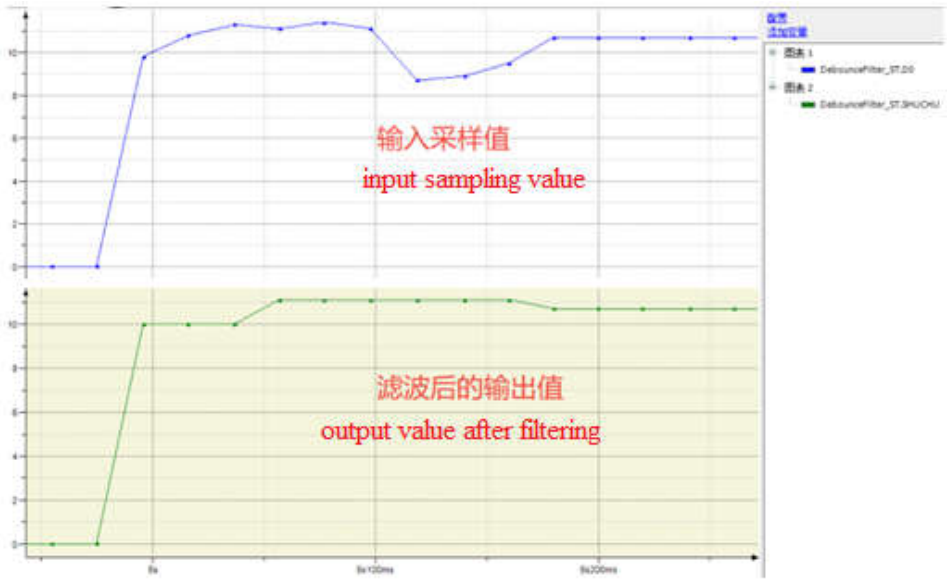
- The input parameter uiSampleCycle in the filtering function block is the sampling period. If uiSampleCycle=1, it means that sampling operations are performed for each period; If uiSampleCycle=3, it means sampling is done every 3 cycles. If the flag of the function block DebounceFilter.xValid outputs TRUE, it indicates that the filtering is valid. The filtered output value fValue will update the output when the filtered output is valid.
- If the sampled data is stored in an array, the data in the array needs to be input into the filtering function block one by one, and the user needs to process the data themselves.

(5) Program example

| DEMO | LD | ST |
|---|---|---|
| Define the variables | VAR<br>A:INT;<br>D0,SHUCHU:REAL;<br>KAISHI:BOOL;<br>DebounceFilter_0:DebounceFilter;<br>END_VAR | |

<table>
<tr>
<td>Program example</td>
<td>




</td>
</tr>
<tr>
<td>Operation result</td>
<td>


</td>
</tr>
<tr>
<td>Result analysis</td>
<td>For 10 consecutive cycles, the sampled values are (9.8, 10.8, 11.3, 11.1, 11.4, 11.1, 8.7, 8.9, 9.5, 10.7).<br>The reference value is set to 10, the maximum allowable counter limit for sample value jitter is 3 (the number of jitter values allowed), the sampling period is set to 1 (1 sampling per cycle), and when the input xEnable is set to TRUE, the function block can effectively start sampling filtering by comparing the sampled value D0 with the current effective value of 10. If they are equal, the sample value jitter counter is reset to zero; If the sampled value</td>
</tr>
</table>

| | |
|---|---|
| | is not equal to the current valid value, the counter is incremented by 1. If the counter value is greater than the maximum counter upper limit of 3, the current sampled value is replaced with the current valid value as the filtered valid value, and the filtered value is output. The filtering execution valid flag xValid is set to TRUE.<br>After filtering, the results are (10.0, 10.0, 10.0, 11.1, 11.1, 11.1, 11.1, 11.1, 11.1, 11.1, 10.7). |

### 1-4-11-10. Limiting and debounce filter LimitingDebounceFilter

(1) Instruction overview

When the xEnable is enabled, set the number of scan cycles uiSampleCycle, set the upper limit of the filter counter uiUpLimit, the sampling reference value fReference, and the maximum allowable deviation value fDeviation for two samplings, and input the sampling variable fSample. The filtering method is equivalent to "amplitude limiting filtering method"+"debounce filtering method". Limit the amplitude first, then reduce the shaking.

If the difference between this value and the previous value is less than or equal to fDeviation, then this value is valid as the sampling value for the debounce filter;

If the difference between the current value and the previous value is greater than fDeviation, the current value is invalid, the current value is discarded, and the previous value is used instead as the sampling value for the debounce filter.

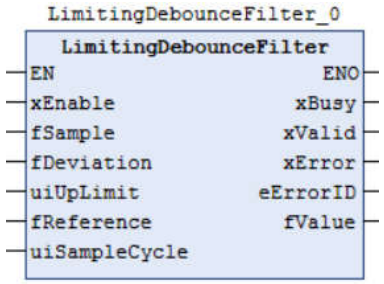
Compare the output of the amplitude limiting filter as the sampled value of the debounce filter with the current effective value:

If the sampled value is equal to the current valid value, then the current count value is reset to zero.

If the sampled value is less than or greater than the current valid value, the count value is increased by 1, and it is determined whether the count value has overflowed (count value>upper limit uiUpLimit).

If the count value overflows, replace the current filtered value with the current sampled value and reset the count value.

| Limiting and debounce filter [LimitingDebounceFilter] | | | |
|---|---|---|---|
| Execution condition | Normally open/closed | Suitable model | XS, XSA |
| Firmware | - | Software | XS StudioV2.2.0 and above |
| Library | XJ_FunctionUtils | | |

| Instruction | Name | Graph | ST |
|---|---|---|---|
| LimitingDebounceFilter | Limiting and debounce filter |  | LimitingDebounceFilter_0(<br>xEnable:= ,<br>fSample:= ,<br>fDeviation:= ,<br>uiUpLimit:= ,<br>fReference:= ,<br>uiSampleCycle:= ,<br>xBusy=> ,<br>xValid=> ,<br>xError=> ,<br>eErrorID=> ,<br>fValue=> ); |

(2) Related variables

| VAR_INPUT | Name | Data type | Effective range | Initial value | Description |
|---|---|---|---|---|---|
| xEnable | Filter enable | BOOL | TRUE/FALSE | FALSE | TRUE: Execute function block<br>FALSE: Do not execute function block |
| fSample | Input sampling value | REAL | - | 0 | Input the value to be filtered |
| fDeviation | Input the maximum allowable deviation value for two samplings | REAL | - | 0 | Input the maximum allowable deviation value for two samplings |
| uiUpLimit | Set the upper limit of the filter counter | UINT | 1-65535 | 0 | Set the upper limit of the filter counter |
| fReference | Input reference value | REAL | - | 0 | Input reference value |
| uiSampleCycle | Input sampling cycle | UINT | 1-1000 | 0 | Input sampling cycle |
| VAR_OUTPUT | Name | Data type | Effective range | Initial value | Description |
| xBusy | The instruction is being executed | BOOL | TRUE/FALSE | FALSE | TRUE: Function block execution in progress |
| xValid | Output is valid | BOOL | TRUE/FALSE | FALSE | TRUE: Instruction execution is effective |
| xError | Error | BOOL | TRUE/FALSE | FALSE | TRUE: Abnormal occurrence |

| | | | | | |
|---|---|---|---|---|---|
| eErrorID | Error ID | UTIL_ERROR | - | 0 | Error ID |
| fValue | Filter output effective value | REAL | - | 0 | Filter output effective value |

(3) Function

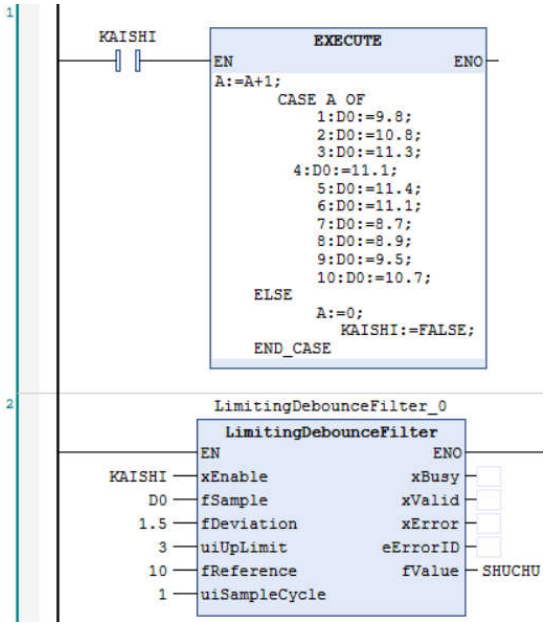
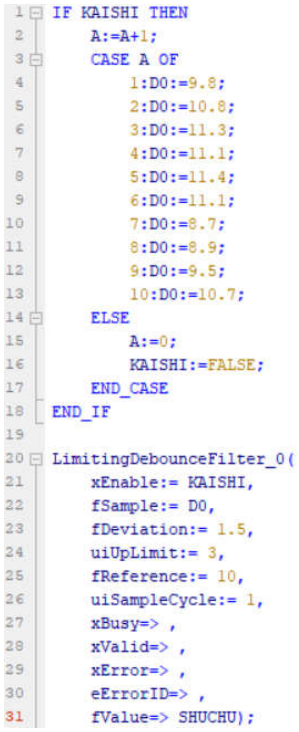
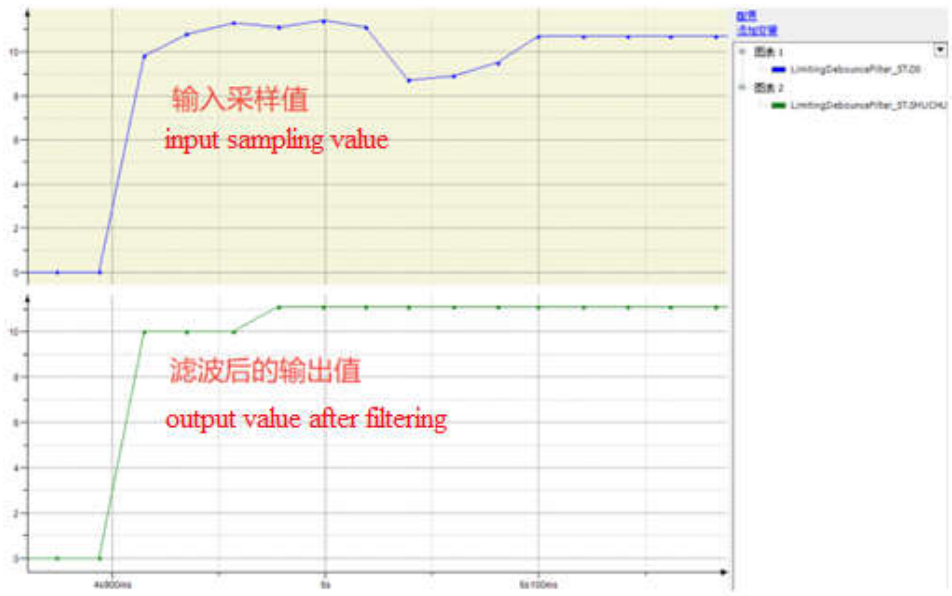
- Advantages: Inheriting the advantages of both "amplitude limiting filtering method" and "debounce filtering method" in filtering; Improved some defects in the "debounce filtering method" to avoid importing interference values into the system.
- Disadvantages: Not suitable for rapidly changing parameters.

(4) Note

- The input parameter uiSampleCycle in the filtering function block is the sampling period. If uiSampleCycle=1, it means that sampling operations are performed for each period; If uiSampleCycle=3, it means sampling is done every 3 cycles. The flag of the function block LimitingDebonceFilter.xValid outputs TRUE, it indicates that the filtering is effective. The filtered output value fValue will update the output when the filtered output is valid.
- If the sampled data is stored in an array, the data in the array needs to be input into the filtering function block one by one, and the user needs to process the data themselves.

(5) Program example

| DEMO | LD | ST |
|---|---|---|
| Define the variables | VAR<br>A:INT;<br>D0,SHUCHU:REAL;<br>KAISHI:BOOL;<br>LimitingDebounceFilter_0:LimitingDebounceFilter;<br>END_VAR | |

| | | |
|---|---|---|
| Program example |  |  |
| Operation result |  | |
| Result analysis | For 10 consecutive cycles, the sampling values are set to (9.8, 10.8, 11.3, 11.1, 11.4, 11.1, 8.7, 8.9, 9.5, 10.7).<br>The reference effective value of the sample is 10.0 (given reference value), the maximum allowable counter upper limit for sample value jitter is 3, and the sampling period is 1 (given the number of running cycles, perform sampling filtering once). When xEnable is set to TRUE, the function block starts sampling and filtering, comparing the sampled value with the current effective value. If they are equal, the jitter count value of the sample value is reset to zero; If the sampled value is not equal to the current effective value, the count value is incremented by 1. If the counter value is greater than the maximum count value limit, the current sampled value is replaced with the current effective value as the filtered effective value, and the filtered value is output. The filtering execution effective flag xValid | |

| | |
|---|---|
| | is set to TRUE.<br>As can be seen, after filtering, the results are (10.0, 10.0, 10.0, 11.1, 11.1, 11.1, 11.1, 11.1, 11.1). |

1-4-11-11. UTIL_ERROR data introduction

| Name | Return value | Description |
|---|---|---|
| NO_ERROR | 0 | No error |
| DEVIATION_PARAMETER_SET_ERROR | 1 | The absolute value of the maximum deviation value for two samplings is set incorrectly and cannot be less than 0 |
| UPLIMIT_PARAMETER_SET_ERROR | 2 | The upper limit of the jitter count value is set incorrectly |
| MAF_SAMPLENUM_PARAMETER_SET_ERROR | 3 | The setting of the number of consecutive samples for median average filtering is incorrect, ranging from 3-3000 |
| SAMPLENUM_PARAMETER_SET_ERROR | 4 | Error in setting the number of consecutive sampling, 1-1000 |
| QUEUECOUNT_PARAMETER_SET_ERROR | 5 | The queue length for storing sampling values is set incorrectly |
| COEFFICIENT_PARAMETER_SET_ERROR | 6 | First order low-pass filter coefficient setting error |
| WEIGHTED_PARAMETER_SET_ERROR | 7 | The weighting coefficient is set incorrectly and cannot all be 0 |
| SAMPLECYCLE_PARAMETER_SET_ERROR | 8 | Sampling cycle setting error, 1-1000 |

# 1-5. Operation instructions

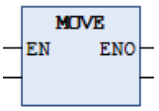
## 1-5-1. Assignment instruction

### 1-5-1-1. Assignment instruction MOVE

(1) Instruction overview

Function: Assign the value of a constant or variable to another variable.

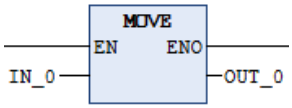
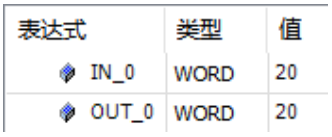
| Assignment instruction [MOVE] | | | |
|---|---|---|---|
| Execution condition | Edge triggering | Suitable model | XS, XSA |
| Firmware | - | Software | - |
| Library | - | | |

| Instruction | Name | Graph | ST |
|---|---|---|---|
| MOVE | Assignment instruction |  | MOVE |

(2) Related variables

| VAR_INPUT | Name | Data type | Valid range | Initial value | Description |
|---|---|---|---|---|---|
| IN | Send data | BOOL/BYTE/WORD/DWORD/LWORD/USINT/UINT/UDINT/ULINT/SINT/INT/DINT/LINT/REAL/LREAL/TIME/DATE/TOD/DT/STRING | As the data type | - | Variables for transmitting data |
| VAR_IN_OUT | Name | Data type | Valid range | Initial value | Description |
| OUT | Receive data | BOOL/BYTE/WORD/DWORD/LWORD/USINT/UINT/UDINT/ULINT/SINT/INT/DINT/LINT/REAL/LREAL/TIME/DATE/TOD/DT/STRING | As the data type | - | Variables receiving data |

(3) Program example

| DEMO | LD | ST |
|---|---|---|
| Define variable | | VAR<br>IN_0 :WORD:=20;<br>OUT_0:WORD;<br>END_VAR |
| Program example |  | 1 OUT_0:=MOVE(IN_0); |
| Operation result |  | |

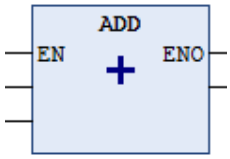
## 1-5-2. Arithmetic operation instructions

### 1-5-2-1. Addition operation ADD

(1) Instruction overview

Function: Addition operation instruction, adding two (or more) variables or constants. Two time variables can also be added, resulting in another time variable.

| Addition operation [ADD] | | | |
|---|---|---|---|
| Execution condition | Edge triggering | Suitable model | XS, XSA |
| Firmware | - | Software | - |
| Library | - | | |

| Instruction | Name | Graph | ST |
|---|---|---|---|
| ADD | Addition operation |  | + |

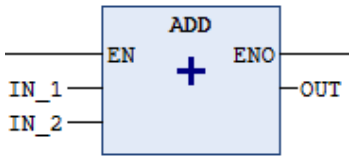
(2) Related variables

| VAR_INPUT | Name | Data type | Valid range | Initial value | Description |
|---|---|---|---|---|---|
| IN1 | Addend 1 | BYTE/WORD/DWORD/LWORD/USINT/UINT/UDINT/ULINT/SINT/INT/DINT/LINT/REAL/LREAL/TIME | As the data type | - | Addend 1 |
| IN2 | Addend 2 | BYTE/WORD/DWORD/LWORD/USINT/UINT/UDINT/ULINT/SINT/INT/DINT/LINT/REAL/LREAL/TIME | As the data type | - | Addend 2 |
| VAR_IN_OUT | Name | Data type | Valid range | Initial value | Description |
| OUT | Sum | BYTE/WORD/DWORD/LWORD/USINT/UINT/UDINT/ULINT/SINT/INT/DINT/LINT/REAL/LREAL/TIME | As the data type | - | Sum |

(3) Program example

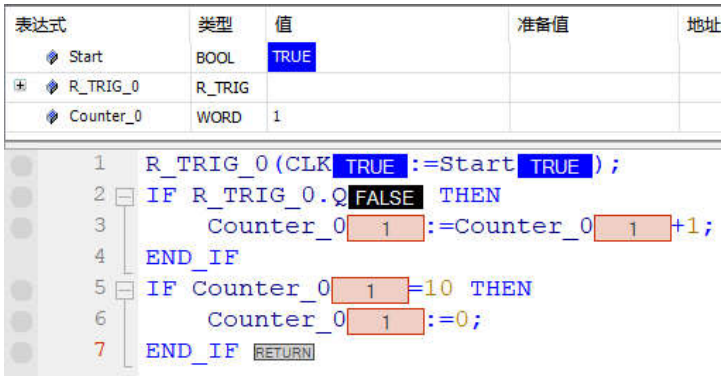
① Add the values of two INT variables:

| DEMO | LD | ST |
|---|---|---|
| Define variable | | VAR<br>IN_1:INT:=10;<br>IN_2:INT:=20;<br>OUT :INT;<br>END_VAR |

| | | |
|---|---|---|
| Program example |  | 1 OUT:=IN_1+IN_2; |
| Operation result | | |

| 表达式 | 类型 | 值 |
|---|---|---|
| IN_1 | INT | 10 |
| IN_2 | INT | 20 |
| OUT | INT | 30 |

② Using the ST programming language, the rising edge trigger function block accumulates the addend Counter_0. When the number accumulates to 10, the cumulative variable is reset to zero.

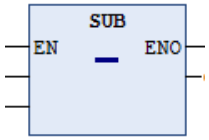
| DEMO | ST |
|---|---|
| Define variable | VAR<br>Start : BOOL;<br>R_TRIG_0 :R_TRIG;<br>Counter_0:WORD;<br>END_VAR |
| Program example | 1 R_TRIG_0(CLK:=Start);<br>2 IF R_TRIG_0.Q THEN<br>3 Counter_0:=Counter_0+1;<br>4 END_IF<br>5 IF Counter_0=10 THEN<br>6 Counter_0:=0;<br>7 END_IF |
| Operation result |  |

1-5-2-2. Subtraction operation SUB

(1) Instruction overview

Function: Subtraction operation instruction, subtracting two variables or constants.

| Subtraction operation [SUB] | | | |
|---|---|---|---|
| Execution condition | Edge triggering | Suitable model | XS, XSA |
| Firmware | - | Software | - |
| Library | - | | |

| Instruction | Name | Graph | ST |
|---|---|---|---|
| SUB | Subtraction operation |  | - |

(2) Related variables

| VAR_INPUT | Name | Data type | Valid range | Initial value | Description |
|---|---|---|---|---|---|
| IN1 | Minuend | BYTE/WORD/DWORD/LWORD/USINT/UINT/UDINT/ULINT/SINT/INT/DINT/LINT/REAL/LREAL/TIME/TOD | As the data type | - | Minuend |
| IN2 | Subtrahend | BYTE/WORD/DWORD/LWORD/USINT/UINT/UDINT/ULINT/SINT/INT/DINT/LINT/REAL/LREAL/TIME/TOD | As the data type | - | Subtrahend |
| VAR_IN_OUT | Name | Data type | Valid range | Initial value | Description |
| OUT | Difference | BYTE/WORD/DWORD/LWORD/USINT/UINT/UDINT/ULINT/SINT/INT/DINT/LINT/REAL/LREAL/TIME | As the data type | - | Difference |

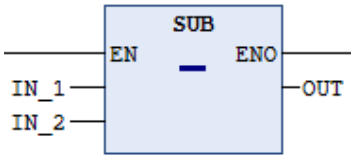
(3) Note

TOD type variables can also use subtraction function, where two TOD types are subtracted to obtain a new TIME type data.

Example: TOD#20:10:0- TOD#20:9:59=T#1s, but the time result cannot have a negative value.

(4) Program example

| DEMO | LD | ST |
|---|---|---|
| Define variable | | VAR<br>IN_1:INT:=20;<br>IN_2:INT:=3;<br>OUT :INT;<br>END_VAR |

| | | |
|---|---|---|
| Program example |  | 1 OUT:=IN_1-IN_2; |
| Operation result | | |

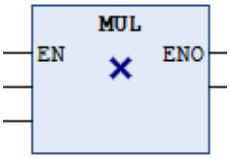
| 表达式 | 类型 | 值 |
|---|---|---|
| IN_1 | INT | 20 |
| IN_2 | INT | 3 |
| OUT | INT | 17 |

1-5-2-3. Multiplication operation MUL

(1) Instruction overview

Function: Multiplication operation instruction, multiplying two (or more) variables or constants.

| Multiplication operation [MUL] | | | |
|---|---|---|---|
| Execution condition | Edge triggering | Suitable model | XS, XSA |
| Firmware | - | Software | - |
| Library | - | | |

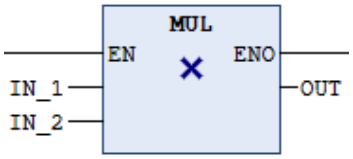
| Instruction | Name | Graph | ST |
|---|---|---|---|
| MUL | Multiplication operation |  | * |

(2) Related variables

| VAR_INPUT | Name | Data type | Valid range | Initial value | Description |
|---|---|---|---|---|---|
| IN1 | Multiplicator 1 | BYTE/WORD/DWORD/LWORD/USINT/UINT/UDINT/ULINT/SINT/INT/DINT/LINT/REAL/LREAL/TIME/TOD | As the data type | - | Multiplicator 1 |
| IN2 | Multiplicator 2 | BYTE/WORD/DWORD/LWORD/USINT/UINT/UDINT/ULINT/SINT/INT/DINT/LINT/REAL/LREAL/TIME/TOD | As the data type | - | Multiplicator 2 |
| VAR_IN_OUT | Name | Data type | Valid range | Initial value | Description |
| OUT | Product | BYTE/WORD/DWORD/LWOR | As the data | - | Product |

| | | | | | |
|---|---|---|---|---|---|
| | | D/USINT/UINT/UDINT/ULINT/ SINT/INT/DINT/LINT/REAL/L REAL/TIME/TOD | type | | |

(3) Program example

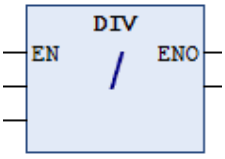
| DEMO | LD | ST |
|---|---|---|
| Define variable | VAR<br>IN_1:INT:=20;<br>IN_2:INT:=3;<br>OUT :INT;<br>END_VAR | |
| Program example |  | 1 OUT:=IN_1*IN_2; |
| Operation result | | |

1-5-2-4. Division operation DIV

(1) Instruction overview

Function: Division operation instruction, dividing two variables or constants.

| Division operation [DIV] | | | |
|---|---|---|---|
| Execution condition | Edge triggering | Suitable model | XS, XSA |
| Firmware | - | Software | - |
| Library | - | | |

| Instruction | Name | Graph | ST |
|---|---|---|---|
| DIV | Division operation |  | / |

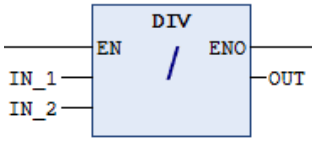
(2) Related variables

| VAR_INPUT | Name | Data type | Valid range | Initial value | Description |
|---|---|---|---|---|---|
| IN1 | Dividend | BYTE/WORD/DWORD/LWORD/ USINT/UINT/UDINT/ULINT/SIN T/INT/DINT/LINT/REAL/LREAL | As the data type | - | Dividend |
| IN2 | Divisor | BYTE/WORD/DWORD/LWORD/ USINT/UINT/UDINT/ULINT/SIN T/INT/DINT/LINT/REAL/LREAL | As the data type | - | Divisor |
| VAR_IN_OUT | Name | Data type | Valid range | Initial value | Description |
| OUT | Quotient | BYTE/WORD/DWORD/LWORD/ USINT/UINT/UDINT/ULINT/SIN T/INT/DINT/LINT/REAL/LREAL | As the data type | - | Quotient |

(3) Note

When using the DIV command in engineering, CheckDivByte, CheckDivWord, CheckDivDWord, and CheckDivReal can be used to check whether the divisor is zero, avoiding the phenomenon of divisors being zero.

(4) Program example

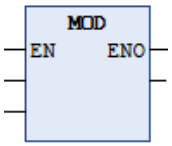
| DEMO | LD | ST |
|---|---|---|
| Define variable | VAR<br>IN_1:INT:=20;<br>IN_2:INT:=5;<br>OUT :INT;<br>END_VAR | |
| Program example |  | 1 OUT:=IN_1/IN_2; |
| Operation result | | |

1-5-2-5. Remainder operation MOD

(1) Instruction overview

Function: Divide variables or constants to obtain remainder, and the result is the remainder after dividing two numbers, which is an integer data.

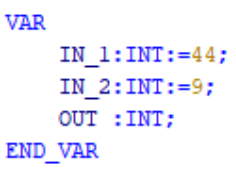
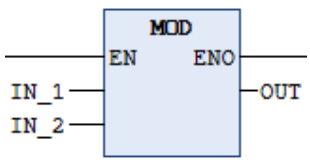
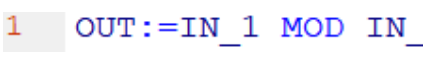
| Remainder operation [MOD] | | | |
|---|---|---|---|
| Execution condition | Edge triggering | Suitable model | XS, XSA |
| Firmware | - | Software | - |
| Library | - | | |

| Instruction | Name | Graph | ST |
|---|---|---|---|
| MOD | Remainder operation |  | MOD |

(2) Related variables

| VAR_INPUT | Name | Data type | Valid range | Initial value | Description |
|---|---|---|---|---|---|
| IN1 | Dividend | BYTE/WORD/DWORD/LWORD/USINT/UINT/UDINT/ULINT/SINT/INT/DINT/LINT/REAL/LREAL | As the data type | - | Dividend |
| IN2 | Divisor | BYTE/WORD/DWORD/LWORD/USINT/UINT/UDINT/ULINT/SINT/INT/DINT/LINT/REAL/LREAL | As the data type | - | Divisor |
| VAR_IN_OUT | Name | Data type | Valid range | Initial value | Description |
| OUT | Remainder | BYTE/WORD/DWORD/LWORD/USINT/UINT/UDINT/ULINT/SINT/INT/DINT/LINT/REAL/LREAL | As the data type | - | Remainder |

(3) Program example

| DEMO | LD | ST |
|---|---|---|
| Define variable |  | |
| Program example |  |  |

| | |
|---|---|
| Operation result | |



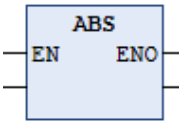
## 1-5-3. Mathematical operation instructions

### 1-5-3-1. Absolute value ABS

(1) Instruction overview

Function: This function instruction is used to calculate the absolute value of a number. It has nothing to do with the sign of positive and negative numbers.

| Absolute value [ABS] | | | |
|---|---|---|---|
| Execution condition | Edge triggering | Suitable model | XS, XSA |
| Firmware | - | Software | - |
| Library | - | | |

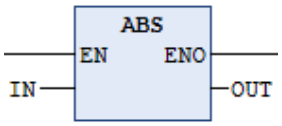
| Instruction | Name | Graph | ST |
|---|---|---|---|
| ABS | Absolute value |  | ABS |



(2) Related variables

| VAR_INPUT | Name | Data type | Valid range | Initial value | Description |
|---|---|---|---|---|---|
| IN | Input variables | BYTE/WORD/DWORD/LWORD /USINT/UINT/UDINT/ULINT/SINT/INT/DINT/LINT/REAL/LREAL | As the data type | - | Parameters to be taken as absolute values |
| VAR_IN_OUT | Name | Data type | Valid range | Initial value | Description |
| OUT | Output value | BYTE/WORD/DWORD/LWORD /USINT/UINT/UDINT/ULINT/SINT/INT/DINT/LINT/REAL/LREAL | As the data type | - | Output value |

(3) Program example

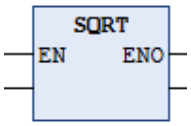
| DEMO | LD | ST |
|---|---|---|
| Define variable | VAR<br>IN :INT:=-12;<br>OUT:INT;<br>END_VAR | |
| Program example |  | 1 OUT:=ABS(IN); |
| Operation result | | |

1-5-3-2. Square root SQRT

(1) Instruction overview

Function: The square root of non negative real numbers.

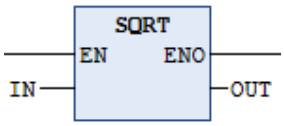
| Square root [SQRT] | | | |
|---|---|---|---|
| Execution condition | Edge triggering | Suitable model | XS, XSA |
| Firmware | - | Software | - |
| Library | - | | |

| Instruction | Name | Graph | ST |
|---|---|---|---|
| SQRT | Square root |  | SQRT |

(2) Related variables

| VAR_INPUT | Name | Data type | Valid range | Initial value | Description |
|---|---|---|---|---|---|
| IN | Input parameter | BYTE/WORD/DWORD/LWORD/USINT/UINT/UDINT/ULINT/SINT/INT/DINT/LINT/REAL/LREAL | As the data type | - | Variable to be calculated |
| VAR_IN_OUT | Name | Data type | Valid range | Initial value | Description |
| OUT | Calculation results | REAL/LREAL | As the data type | - | Result |

(3) Program example

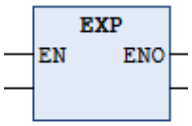
| DEMO | LD | ST |
|---|---|---|
| Define variable | VAR<br>IN :INT:=16;<br>OUT:REAL;<br>END_VAR | |
| Program example |  | 1 OUT:=SQRT(IN); |
| Operation result | | |

1-5-3-3. Exponent EXP

(1) Instruction overview

Function: return the power of e (the base of Natural logarithm). e is a constant of 2.71828.

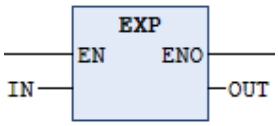
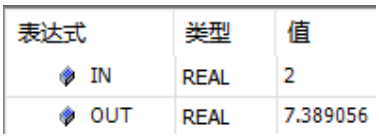
| Exponent [EXP] | | | |
|---|---|---|---|
| Execution condition | Edge triggering | Suitable model | XS, XSA |
| Firmware | - | Software | - |
| Library | - | | |

| Instruction | Name | Graph | ST |
|---|---|---|---|
| EXP | Exponent |  | EXP |

(2) Related variables

| VAR_INPUT | Name | Data type | Valid range | Initial value | Description |
|---|---|---|---|---|---|
| IN | Input parameter | BYTE/WORD/DWORD/LWORD/USINT/UINT/UDINT/ULINT/SINT/INT/DINT/LINT/REAL/LREAL | As the data type | - | Variable to be calculated |
| VAR_IN_OUT | Name | Data type | Valid range | Initial value | Description |
| OUT | Output value | REAL/LREAL | As the data type | - | Output result |

(3) Program example

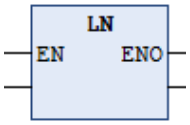
| DEMO | LD | ST |
|---|---|---|
| Define variable | VAR<br>IN :REAL:=2;<br>OUT:REAL;<br>END_VAR | |
| Program example |  | 1 OUT:=EXP(IN); |
| Operation result |  | |

1-5-3-4. Natural logarithm LN

(1) Instruction overview

Function: returns the Natural logarithm of a number. The base of Natural logarithm is the constant term e (2.71828182845904).

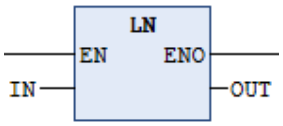
| Natural logarithm [LN] | | | |
|---|---|---|---|
| Execution condition | Edge triggering | Suitable model | XS, XSA |
| Firmware | - | Software | - |
| Library | - | | |

| Instruction | Name | Graph | ST |
|---|---|---|---|
| LN | Natural logarithm |  | LN |

(2) Related variables

| VAR_INPUT | Name | Data type | Valid range | Initial value | Description |
|---|---|---|---|---|---|
| IN | Input parameter | BYTE/WORD/DWORD/LWORD/USINT/UINT/UDINT/ULINT/SINT/INT/DINT/LINT/REAL/LREAL | As the data type | - | Variable to be calculated |
| VAR_IN_OUT | Name | Data type | Valid range | Initial value | Description |
| OUT | Output value | REAL/LREAL | As the data type | - | Output result |

(3) Program example

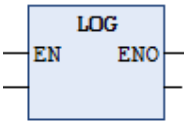
| DEMO | LD | ST |
|---|---|---|
| Define variable | VAR<br>IN :REAL:=45;<br>OUT:REAL;<br>END_VAR | |
| Program example |  | 1 OUT:=LN(IN); |
| Operation result | | |

1-5-3-5. Logarithm with a base of 10 LOG

(1) Instruction overview

Function: Returns the logarithm of a base of 10.

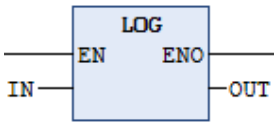
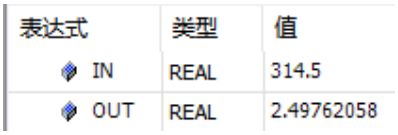
| Logarithm with a base of 10 [LOG] | | | |
|---|---|---|---|
| Execution condition | Edge triggering | Suitable model | XS, XSA |
| Firmware | - | Software | - |
| Library | - | | |

| Instruction | Name | Graph | ST |
|---|---|---|---|
| LOG | Logarithm with a base of 10 |  | LOG |

(2) Related variables

| VAR_INPUT | Name | Data type | Valid range | Initial value | Description |
|---|---|---|---|---|---|
| IN | Input parameter | BYTE/WORD/DWORD/LWORD/USINT/UINT/UDINT/ULINT/SINT/INT/DINT/LINT/REAL/LREAL | As the data type | - | Variable to be calculated |
| VAR_IN_OUT | Name | Data type | Valid range | Initial value | Description |
| OUT | Output value | REAL/LREAL | As the data type | - | Output result |

(3) Program example

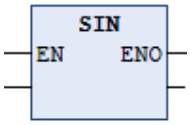
| DEMO | LD | ST |
|---|---|---|
| Define variable | VAR<br>IN :REAL:=314.5;<br>OUT:REAL;<br>END_VAR | |
| Program example |  | 1 OUT:=LOG(IN); |
| Operation result |  | |

1-5-3-6. Sine function SIN

(1) Instruction overview

Function: Sine function.

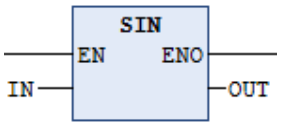
| Sine function [SIN] | | | |
|---|---|---|---|
| Execution condition | Edge triggering | Suitable model | XS, XSA |
| Firmware | - | Software | - |
| Library | - | | |

| Instruction | Name | Graph | ST |
|---|---|---|---|
| SIN | Sine function |  | SIN |

(2) Related variables

| VAR_INPUT | Name | Data type | Valid range | Initial value | Description |
|---|---|---|---|---|---|
| IN | Input parameter | BYTE/WORD/DWORD/LWORD/ USINT/UINT/UDINT/ULINT/SIN T/ INT/DINT/LINT/REAL/LREAL | As the data type | - | Variable to be calculated |
| VAR_IN_OUT | Name | Data type | Valid range | Initial value | Description |
| OUT | Output value | REAL/LREAL | As the data type | - | Output result |

(3) Program example

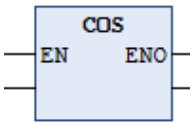
| DEMO | LD | ST |
|---|---|---|
| Define variable | VAR<br>IN :REAL:=0.5;<br>OUT:REAL;<br>END_VAR | |
| Program example |  | 1 OUT:=SIN(IN); |
| Operation result | 表达式 / 类型 / 值<br>IN / REAL / 0.5<br>OUT / REAL / 0.47942555 | |

1-5-3-7. Cosine function COS

(1) Instruction overview

Function: Cosine function.

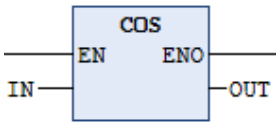
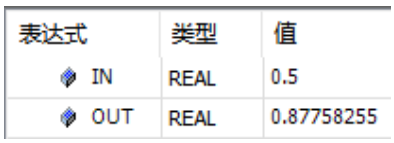
| Cosine function [COS] | | | |
|---|---|---|---|
| Execution condition | Edge triggering | Suitable model | XS, XSA |
| Firmware | - | Software | - |
| Library | - | | |

| Instruction | Name | Graph | ST |
|---|---|---|---|
| COS | Cosine function |  | COS |

(2) Related variables

| VAR_INPUT | Name | Data type | Valid range | Initial value | Description |
|---|---|---|---|---|---|
| IN | Input parameter | BYTE/WORD/DWORD/LWORD /USINT/UINT/UDINT/ULINT/SINT/INT/DINT/LINT/REAL/LREAL | As the data type | - | Variable to be calculated |
| VAR_IN_OUT | Name | Data type | Valid range | Initial value | Description |
| OUT | Output value | REAL/LREAL | As the data type | - | Output result |

(3) Program example

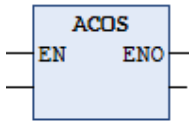
| DEMO | LD | ST |
|---|---|---|
| Define variable | VAR<br>IN :REAL:=0.5;<br>OUT:REAL;<br>END_VAR | |
| Program example |  | 1 OUT:=COS(IN); |
| Operation result |  | |

1-5-3-8. Arccosine function ACOS

(1) Instruction overview

Cosine radian (arccosine function).

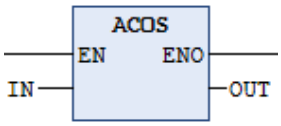
| Arccosine function [ACOS] | | | |
|---|---|---|---|
| Execution condition | Edge triggering | Suitable model | XS, XSA |
| Firmware | - | Software | - |
| Library | - | | |

| Instruction | Name | Graph | ST |
|---|---|---|---|
| ACOS | Arccosine function |  | ACOS |

(2) Related variables

| VAR_INPUT | Name | Data type | Valid range | Initial value | Description |
|---|---|---|---|---|---|
| IN | Input parameter | BYTE/WORD/DWORD/LWORD/USINT/UINT/UDINT/ULINT/SINT/INT/DINT/LINT/REAL/LREAL | As the data type | - | Variable to be calculated |
| VAR_IN_OUT | Name | Data type | Valid range | Initial value | Description |
| OUT | Output value | REAL/LREAL | As the data type | - | Output result |

(3) Program example

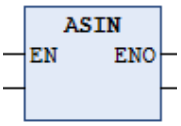
| DEMO | LD | ST |
|---|---|---|
| Define variable | VAR<br>IN :REAL:=0.5;<br>OUT:REAL;<br>END_VAR | |
| Program example |  | 1 OUT:=ACOS(IN); |
| Operation result | | |

1-5-3-9. Arcsin function ASIN

(1) Instruction overview

Function: Sine radian (Arcsine function).

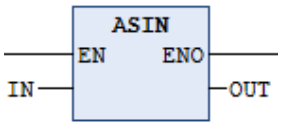
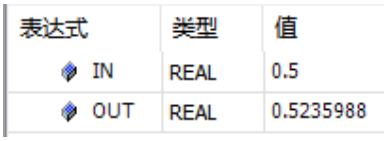
| Arcsin function [ASIN] | | | |
|---|---|---|---|
| Execution condition | Edge triggering | Suitable model | XS, XSA |
| Firmware | - | Software | - |
| Library | - | | |

| Instruction | Name | Graph | ST |
|---|---|---|---|
| ASIN | Arcsin function |  | ASIN |

(2) Related variables

| VAR_INPUT | Name | Data type | Valid range | Initial value | Description |
|---|---|---|---|---|---|
| IN | Input parameter | BYTE/WORD/DWORD/LWORD/USINT/UINT/UDINT/ULINT/SINT/INT/DINT/LINT/REAL/LREAL | As the data type | - | Variable to be calculated |
| VAR_IN_OUT | Name | Data type | Valid range | Initial value | Description |
| OUT | Output value | REAL/LREAL | As the data type | - | Output result |

(3) Program example

| DEMO | LD | ST |
|---|---|---|
| Define variable | VAR<br>IN :REAL:=0.5;<br>OUT:REAL;<br>END_VAR | |
| Program example |  | 1 OUT:=ASIN(IN); |
| Operation result |  | |

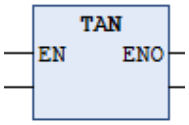
| 表达式 | 类型 | 值 |
|---|---|---|
| IN | REAL | 0.5 |
| OUT | REAL | 0.5235988 |

1-5-3-10. Tangent function TAN

(1) Instruction overview

Function: Tangent function.

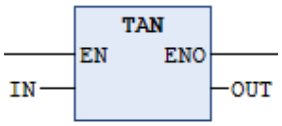
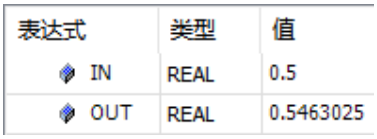
| Tangent function [TAN] | | | |
|---|---|---|---|
| Execution condition | Edge triggering | Suitable model | XS, XSA |
| Firmware | - | Software | - |
| Library | - | | |

| Instruction | Name | Graph | ST |
|---|---|---|---|
| TAN | Tangent function |  | TAN |

(2) Related variables

| VAR_INPUT | Name | Data type | Valid range | Initial value | Description |
|---|---|---|---|---|---|
| IN | Input parameter | BYTE/WORD/DWORD/LWORD/USINT/UINT/UDINT/ULINT/SINT/INT/DINT/LINT/REAL/LREAL | As the data type | - | Variable to be calculated |
| VAR_IN_OUT | Name | Data type | Valid range | Initial value | Description |
| OUT | Output value | REAL/LREAL | As the data type | - | Output result |

(3) Program example

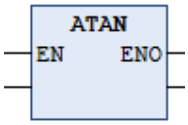
| DEMO | LD | ST |
|---|---|---|
| Define variable | VAR<br>IN :REAL:=0.5;<br>OUT:REAL;<br>END_VAR | |
| Program example |  | 1 OUT:=TAN(IN); |
| Operation result |  | |

1-5-3-11. Arctangent function ATAN

(1) Instruction overview

Function: Tangent radian (Arctangent function).

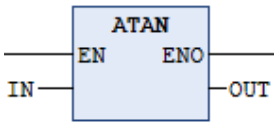
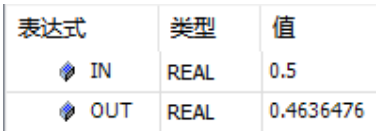
| Arctangent function [ATAN] | | | |
|---|---|---|---|
| Execution condition | Edge triggering | Suitable model | XS, XSA |
| Firmware | - | Software | - |
| Library | - | | |

| Instruction | Name | Graph | ST |
|---|---|---|---|
| ATAN | Arctangent function |  | ATAN |

(2) Related variables

| VAR_INPUT | Name | Data type | Valid range | Initial value | Description |
|---|---|---|---|---|---|
| IN | Input parameter | BYTE/WORD/DWORD/LWORD/USINT/UINT/UDINT/ULINT/SINT/INT/DINT/LINT/REAL/LREAL | As the data type | - | Variable to be calculated |
| VAR_IN_OUT | Name | Data type | Valid range | Initial value | Description |
| OUT | Output value | REAL/LREAL | As the data type | - | Output result |

(3) Program example

| DEMO | LD | ST |
|---|---|---|
| Define variable | VAR<br>IN :REAL:=0.5;<br>OUT:REAL;<br>END_VAR | |
| Program example |  | 1 OUT:=ATAN(IN); |
| Operation result |  | |

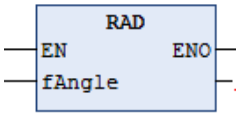
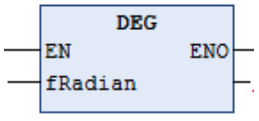
1-5-3-12. Floating point angle/radian conversion RAD/DEG

(1) Instruction overview

RAD: The operation of converting floating-point angles to radians. The calculation formula is [radian unit=angle unit ×л /180].

DEG: The operation of converting floating-point numbers from radians to angles. The calculation formula is [angle unit=radian unit ×л /180].

| Floating point angle/radian conversion [RAD/DEG] | | | |
|---|---|---|---|
| Execution condition | Edge triggering | Suitable model | XS, XSA |
| Firmware | - | Software | XS StudioV2.2.0 and above |
| Library | XJ_DataUtils | | |

| Instruction | Name | Graph | ST |
|---|---|---|---|
| RAD | Convert angles to radians |  | RAD(fAngle:= ); |
| DEG | Convert radians to angles |  | DEG(fRadian:= ); |

(2) Related variables

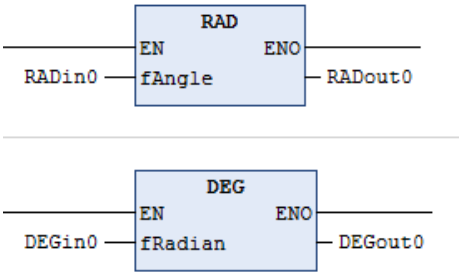
| VAR_INPUT | Name | Data type | Valid range | Initial value | Description |
|---|---|---|---|---|---|
| fAngle | Input angle | LREAL | As the data type | 0 | Input angle |
| | Output radian | LREAL | As the data type | 0 | Output radian |

| | | | | | |
|---|---|---|---|---|---|
| fRadian | Input radian | LREAL | As the data type | 0 | Input radian |
| | Output angle | LREAL | As the data type | 0 | Output angle |

(3) Function

- RAD: The operation of converting floating-point angles to radians. The calculation formula is [radian unit=angle unit ×л /180].
- DEG: The operation of converting floating-point numbers from radians to angles. The calculation formula is [angle unit=radian unit ×л /180].

(4) Program example

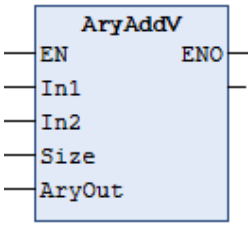
| DEMO | LD | ST |
|---|---|---|
| Define variable | | VAR<br>RADout0: LREAL;<br>RADin0: LREAL;<br>DEGin0: LREAL;<br>DEGout0: LREAL;<br>END_VAR |
| Program example |  | RADout0:=RAD(fAngle:= RADin0);<br>DEGout0:=DEG(fRadian:= DEGin0); |
| Operation result | | |

1-5-3-13. Array element addition AryAddV

(1) Instruction overview

Add the same value to each element of the arrangement.

| Array element addition [AryAddV] | | | |
|---|---|---|---|
| Execution condition | Edge triggering | Suitable model | XS, XSA |
| Firmware | - | Software | XS StudioV2.2.0 and above |
| Library | XJ_DataUtils | | |

| Instruction | Name | Graph | ST |
|---|---|---|---|
| AryAddV | Array element addition |  | AryAddV(In1:= ,<br>In2:= ,<br>Size:= ,<br>AryOut:= ); |

(2) Related variables

| VAR_INPUT | Name | Data type | Valid range | Initial value | Description |
|---|---|---|---|---|---|
| In1[] | Additive arrangement array | USINT/UINT/UDINT/ ULINT/SINT/INT/ DINT/LINT | As the data type | - | Additive arrangement array |
| In2 | Additive value | USINT/UINT/UDINT/ ULINT/SINT/INT/ DINT/LINT | As the data type | - | Additive value |
| Size | The number of In1[] element to be added | UINT | As the data type | 1 | The number of In1[] element to be added |
| VAR_IN_OUT | Name | Data type | Valid range | Initial value | Description |
| AryOut | Add result arrangement | USINT/UINT/UDINT/ ULINT/SINT/INT/ DINT/LINT | As the data type | - | Add result arrangement |

(3) Function

- Add the additive value "In2" on the "Size" elements starting from In1[0] in the additive arrangement In1[], and output it to the additive result arrangement AryOut[].

(4) Note

- The value of "Size" should not exceed the array area of In1[] or AryOut[] to avoid PLC exceptions.
- The data types for "In[]", "In2", and "AryOut" should be consistent. If different, the value returned by the runtime function is FALSE, AryOut[] remains unchanged.
- If the sum result exceeds the valid range of AryOut[], the element of AryOut[] has an incorrect value. At this point, no abnormalities will occur. In addition, it will not damage the adjacent storage area of the element.
- When the value of "Size" is 0, the value of AryOut[] remains unchanged, and the function returns a value of

True.

- In1, In2, AryOut[] supports both integer and real types, otherwise the runtime function returns FALSE.

(5) Program example

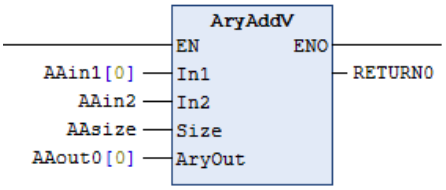
<table>
<tr><th>DEMO</th><th>LD</th><th>ST</th></tr>
<tr><td>Define variable</td><td colspan="2">

```
VAR

    AAin1:ARRAY[0..5] OF INT;
    AAsize: UINT;
    AAin2: INT;
    AAout0:ARRAY[0..5] OF INT;
    RETURN0: BOOL;

END_VAR
```

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




</td><td>

```
AryAddV(In1:=AAin1[0] ,
    In2:= AAin2,
    Size:= AAsize ,
    AryOut:=AAout0[0] );
```

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

| 表达式 | 类型 | 值 |
|---|---|---|
| AAin1 | ARRAY [0..5] OF INT | |
| AAin1[0] | INT | 11 |
| AAin1[1] | INT | 22 |
| AAin1[2] | INT | 33 |
| AAin1[3] | INT | 44 |
| AAin1[4] | INT | 55 |
| AAin1[5] | INT | 66 |
| AAsize | UINT | 4 |
| AAin2 | INT | 12 |
| AAout0 | ARRAY [0..5] OF INT | |
| AAout0[0] | INT | 23 |
| AAout0[1] | INT | 34 |
| AAout0[2] | INT | 45 |
| AAout0[3] | INT | 56 |
| AAout0[4] | INT | 0 |
| AAout0[5] | INT | 0 |
| RETURN0 | BOOL | TRUE |

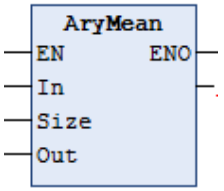
</td></tr>
</table>

1-5-3-14. Calculation of array elements average value AryMean

(1) Instruction overview

Calculate the average value of array elements.

| Calculation of array elements average value [AryMean] | | | |
|---|---|---|---|
| Execution condition | Edge triggering | Suitable model | XS, XSA |
| Firmware | - | Software | XS StudioV2.2.0 and above |
| Library | XJ_DataUtils | | |

| Instruction | Name | Graph | ST |
|---|---|---|---|
| AryMean | Calculation of array elements average value |  | AryMean(In:= ,<br>Size:= ,<br>Out:= ); |

(2) Related variables

| VAR_INPUT | Name | Data type | Valid range | Initial value | Description |
|---|---|---|---|---|---|
| In[] | Arithmetic object array | USINT/UINT/UDINT/ ULINT/SINT/INT/DINT/ LINT/REAL/LREAL/TIME | As the data type | - | Arithmetic object array |
| Size | In[] element numbers | UINT | As the data type | - | In[] element numbers |
| VAR_IN_OUT | Name | Data type | Valid range | Initial value | Description |
| Out | Operation result | USINT/UINT/UDINT/ ULINT/SINT/INT/DINT/ LINT/REAL/LREAL/TIME | As the data type | - | Operation result |

(3) Function

- Calculate the average value of array elements.

(4) Note

- The four arithmetic operations are not supported when the value of In[] is +∞/-∞ and a non number.
- When the data types of In[] and "Out" are different, the function returns a value of FALSE while "Out" remains unchanged.
- The value of "Size" should not exceed the array area of In[] to avoid PLC exceptions.
- When "In" and "Out" are integers, the number after the decimal point of the average is rounded off.
- When the value of "Size" is 0, the value of "Out" is 0, and the function returns a value of True.
- In[], "Out" needs to use integers, real numbers, and TIME types, otherwise the runtime function returns a value of FALSE while "Out" remains unchanged.

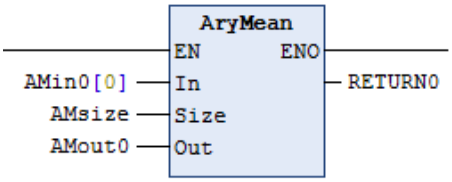
(5) Program example

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

```
VAR

    AMin0:ARRAY[0..5] OF INT;
    AMsize: UINT;
    AMout0: INT;
    RETURN0: BOOL;

END_VAR
```

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


</td><td>

```
AryMean(In:= AMin0[0],
    Size:= AMsize,
    Out:= AMout0);
```

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

| 表达式 | 类型 | 值 |
|---|---|---|
| AMin0 | ARRAY [0..5] OF INT | |
| AMin0[0] | INT | 12 |
| AMin0[1] | INT | 22 |
| AMin0[2] | INT | 32 |
| AMin0[3] | INT | 42 |
| AMin0[4] | INT | 0 |
| AMin0[5] | INT | 0 |
| AMsize | UINT | 4 |
| AMout0 | INT | 27 |
| RETURN0 | BOOL | TRUE |

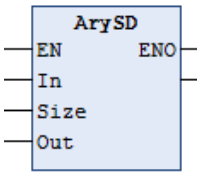
</td></tr>
</table>

1-5-3-15. Standard deviation of array elements ArySD

(1) Instruction overview

Calculate the standard deviation of the arranged elements.

| Standard deviation of array elements [ArySD] | | | |
|---|---|---|---|
| Execution condition | Edge triggering | Suitable model | XS, XSA |
| Firmware | - | Software | XS StudioV2.2.0 and above |
| Library | XJ_DataUtils | | |

| Instruction | Name | Graph | ST |
|---|---|---|---|
| ArySD | Standard deviation of array elements |  | ArySD(In:= ,<br>Size:= ,<br>Out:= ); |

(2) Related variables

| VAR_INPUT | Name | Data type | Valid range | Initial value | Description |
|---|---|---|---|---|---|
| In[] | Arithmetic object array | REAL/LREAL | As the data type | - | Arithmetic object array |
| Size | In[] element numbers | UINT | As the data type | | In[] element numbers |
| VAR_IN_OUT | Name | Data type | Valid range | Initial value | Description |
| Out | standard deviation | REAL/LREAL | As the data type | - | standard deviation |

(3) Function

- Calculate the standard deviation of the "Size" elements starting from In[0] in the operation object arrangement In[].

  i: The suffix of ln[] (0~"Size" -1);

  LnM: The average value of ln[0]~ln["Size" -1].

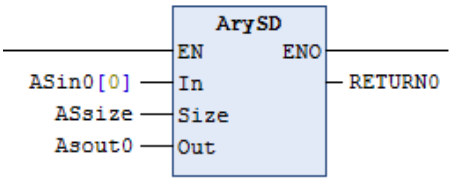
(4) Note

- When the value of "Size" is 0 or 1, the value of "Out" is 0, and the function returns a value of FALSE.
- When the intermediate value during the operation exceeds the valid range of In[], the value of "Out" is an illegal value. At this point, no abnormalities will occur.
- The value of "Size" should not exceed the array area of In[] to avoid PLC exceptions.
- In[] and Out data types should be consistent, otherwise the runtime function returns FALSE, Out remains unchanged.
- In[] supports real number types, otherwise the program compilation will report errors.

(5) Program example

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

```
VAR
    ASin0:ARRAY[0..5] OF REAL;
    ASsize: UINT;
    Asout0: REAL;
    RETURN0: BOOL;
END_VAR
```

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

```
ArySD(In:= ASin0[0],
    Size:= ASsize,
    Out:= ASout0[0] );
```

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

| 表达式 | 类型 | 值 |
|---|---|---|
| ASin0 | ARRAY [0..5] OF REAL | |
| ASin0[0] | REAL | 5 |
| ASin0[1] | REAL | 6 |
| ASin0[2] | REAL | 7 |
| ASin0[3] | REAL | 8 |
| ASin0[4] | REAL | 9 |
| ASin0[5] | REAL | 10 |
| ASsize | UINT | 6 |
| Asout0 | REAL | 1.87082875 |
| RETURN0 | BOOL | TRUE |

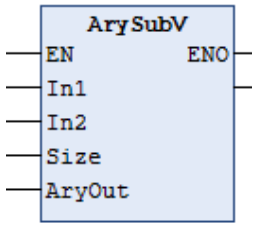
</td></tr>
</table>

1-5-3-16. Subtraction of array elements ArySubV

(1) Instruction overview

Subtract the same value from each element of the arrangement.

| Subtraction of array elements [ArySubV] | | | |
|---|---|---|---|
| Execution condition | Edge triggering | Suitable model | XS, XSA |
| Firmware | - | Software | XS StudioV2.2.0 and above |
| Library | XJ_DataUtils | | |

| Instruction | Name | Graph | ST |
|---|---|---|---|
| ArySubV | Subtraction of array elements |  | ArySubV(In1:= ,<br>In2:= ,<br>Size:= ,<br>AryOut:= ); |

(2) Related variables

| VAR_INPUT | Name | Data type | Valid range | Initial value | Description |
|---|---|---|---|---|---|
| In1[] | Subtracted arrangement | USINT/UINT/UDINT/ULINT/ SINT/INT/DINT/LINT | As the data type | - | Subtracted arrangement |
| In2 | Subtracted value | USINT/UINT/UDINT/ULINT/ SINT/INT/DINT/LINT | As the data type | - | Subtracted value |
| Size | The number of elements to subtract from In1[] | UINT | As the data type | 1 | The number of elements to subtract from In1[] |
| VAR_IN_OUT | Name | Data type | Valid range | Initial value | Description |
| AryOut | Subtraction result arrangement | USINT/UINT/UDINT/ULINT/ SINT/INT/DINT/LINT | As the data type | - | Subtraction result arrangement |

(3) Function

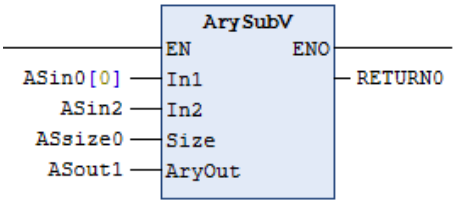
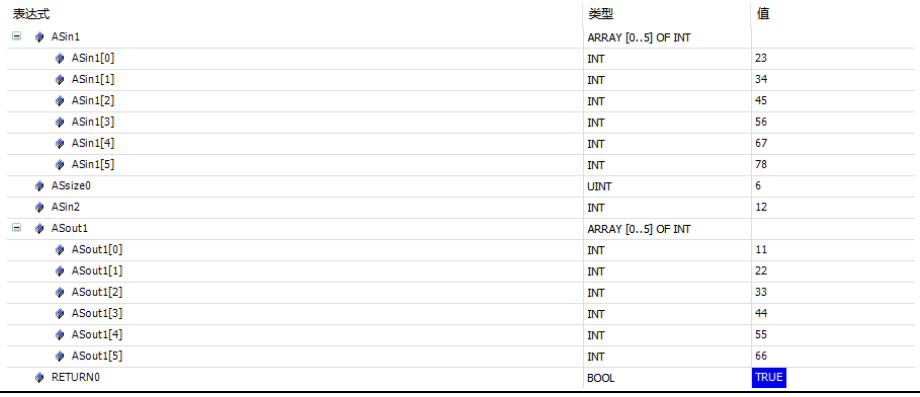
- Subtract "In2" from the "Size" elements starting from In1[0] in the subtracted group In1[] and output it to the subtraction result array AryOut[].

(4) Note

- The value of "Size" should not exceed the array area of In1[] or AryOut[] to avoid PLC exceptions.
- The data types for "In[]", "In2", and "AryOut" should be consistent. If different, the value returned by the runtime function is FALSE, AryOut[] remains unchanged.
- When the subtraction result exceeds the valid range of AryOut[], the element of AryOut[] is an unexpected value. At this point, there will be no abnormalities or damage to the adjacent storage area of the element.
- When the value of "Size" is 0, the value of AryOut[] remains unchanged, and the function returns a value of True.

- In1, In2, AryOut[] supports integer types as input, otherwise the program will compile an error.

(5) Program example

| DEMO | LD | ST |
|---|---|---|
| Define variable | VAR<br>ASin1:ARRAY[0..5] OF INT;<br>ASsize0: UINT;<br>ASin2: INT;<br>ASout1:ARRAY[0..5] OF INT;<br>RETURN0: BOOL;<br><br>END_VAR | |
| Program example |  | ArySubV(In1:= ASin1[0],<br>In2:= ASin2[0],<br>Size:=ASsize0 ,<br>AryOut:= ASout1[0]); |
| Operation result |  | |

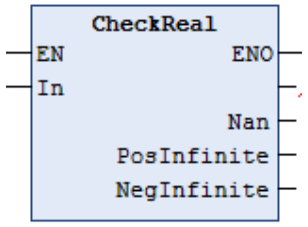
| 表达式 | 类型 | 值 |
|---|---|---|
| ASin1 | ARRAY [0..5] OF INT | |
| ASin1[0] | INT | 23 |
| ASin1[1] | INT | 34 |
| ASin1[2] | INT | 45 |
| ASin1[3] | INT | 56 |
| ASin1[4] | INT | 67 |
| ASin1[5] | INT | 78 |
| ASsize0 | UINT | 6 |
| ASin2 | INT | 12 |
| ASout1 | ARRAY [0..5] OF INT | |
| ASout1[0] | INT | 11 |
| ASout1[1] | INT | 22 |
| ASout1[2] | INT | 33 |
| ASout1[3] | INT | 44 |
| ASout1[4] | INT | 55 |
| ASout1[5] | INT | 66 |
| RETURN0 | BOOL | TRUE |

1-5-3-17. Real number check CheckReal

(1) Instruction overview

Determine if the input value is non numeric, positive infinity, negative infinity.

| Real number check [CheckReal] | | | |
|---|---|---|---|
| Execution condition | Edge triggering | Suitable model | XS, XSA |
| Firmware | - | Software | XS StudioV2.2.0 and above |
| Library | XJ_DataUtils | | |

| Instruction | Name | Graph | ST |
|---|---|---|---|
| CheckReal | Real number check |  | CheckReal(In:= ,<br>Nan=> ,<br>PosInfinite=> ,<br>NegInfinite=> ); |

(2) related variables

| VAR_INPUT | Name | Data type | Valid range | Initial value | Description |
|---|---|---|---|---|---|
| In | Real number | REAL/LREAL | As the data type | - | Real number |
| VAR_IN_OUT | Name | Data type | Valid range | Initial value | Description |
| Nan | Non numerical judgment | BOOL | [FALSE,TRUE] | FALSE | True: Non numerical value<br>FALSE: Not a non-numerical value |
| PosInfinite | Positive infinity judgment | BOOL | [FALSE,TRUE] | FALSE | True: Positive infinity<br>FALSE: Not positive infinity |
| NegInfinite | Negative infinity judgment | BOOL | [FALSE,TRUE] | FALSE | True: Negative infinity<br>FALSE: Not negative infinity |

(3) Function

- Check if the real number "In" is non numerical, positive infinity, and negative infinity, and output the results to "Nan", "PosInfinite", and "NegInfinite" respectively.

(4) Note

- When the parameter passed to "In" is of a data type other than real, an error occurs during compilation.
- When the judgment result is non numerical, positive infinity, or negative infinity, "Nan", "PosInfinite", and "NegInfinite" return True respectively, and the function returns True.
- When the judgment result is a normal value, the function returns a value of FALSE.

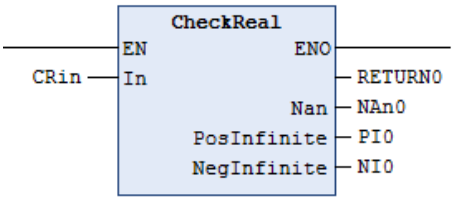
(5) Program example

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

```
VAR
    CRin: REAL;
    NAn0: BOOL;
    PI0: BOOL;
    NI0: BOOL;
    RETURN0: BOOL;

END_VAR
```

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




</td><td>

```
    CRin:=CRin/1E-38;
CheckReal(In:= CRin,
    Nan=> NAn0,
    PosInfinite=>PI0 ,
    NegInfinite=> NI0);
```

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

| 表达式 | 类型 | 值 |
|---|---|---|
| CRin | REAL | Infinity |
| NAn0 | BOOL | FALSE |
| PI0 | BOOL | TRUE |
| NI0 | BOOL | FALSE |

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

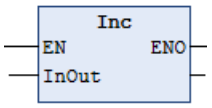
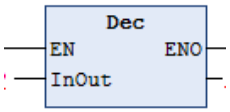
1-5-3-18. Increase/decrease Inc/Dec

(1) Instruction overview

Inc: Increase integer values.

Dec: Decrease integer value.

| Increase/decrease [Inc/Dec] | | | |
|---|---|---|---|
| Execution condition | Edge triggering | Suitable model | XS, XSA |
| Firmware | - | Software | XS StudioV2.2.0 and above |
| Library | XJ_DataUtils | | |

| Instruction | Name | Graph | ST |
|---|---|---|---|
| Inc | Increase |  | `Inc(InOut:= );` |
| Dec | Decrease |  | `Dec(InOut:= );` |

(2) Related variables

| VAR_IN_OUT | Name | Data type | Valid range | Initial value | Description |
|---|---|---|---|---|---|
| Inout | Object data | USINT/UINT/UDINT/ULINT/ SINT/INT/DINT/LINT | As the data type | - | Object data |

(3) Function

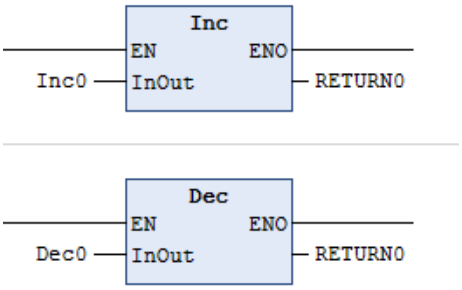
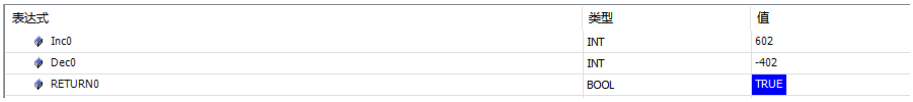
- Inc: Increment the object data "InOut". Finally, when the maximum value of "InOut" is exceeded, it returns to the minimum value.
- Dec: Decrement the object data "InOut". Finally, when the minimum value of "InOut" is exceeded, it returns to the maximum value.

(4) Note

- When using this instruction in the ST program, the return value "Out" is not used.

(5) Program sample

| DEMO | LD | ST |
|---|---|---|
| Define variable | | `VAR`<br>`  Inc0: INT:=100;`<br>`  Dec0: INT:=100;`<br>`  RETURN0: BOOL;`<br><br>`END_VAR` |

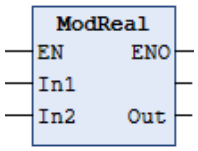
| | | |
|---|---|---|
| Program example |  | `Inc(InOut:= Inc0);`<br>`Dec(InOut:= Dec0);` |
| Operation result |  | |
| Result analysis | Command conduction, the Inc instruction will continue to increase; The Dec command will continue to decrease. | |

1-5-3-19. Real remainder ModReal

(1) Instruction overview

Calculate the remainder obtained by dividing the dividend "In1" by the divisor "In2".

| Real remainder [ModReal] | | | |
|---|---|---|---|
| Execution condition | Edge triggering | Suitable model | XS, XSA |
| Firmware | - | Software | XS StudioV2.2.0 and above |
| Library | XJ_DataUtils | | |

| Instruction | Name | Graph | ST |
|---|---|---|---|
| ModReal | Real remainder |  | `ModReal(In1:= ,`<br>`In2:= ,`<br>`Out=> );` |

(2) Related variables

| VAR_INPUT | Name | Data type | Valid range | Initial value | Description |
|---|---|---|---|---|---|
| In1 | Dividend | REAL | As the data type | (*) | Dividend |
| In2 | Divisor | REAL | As the data type | (*) | Divisor |
| VAR_OUTPUT | Name | Data type | Valid range | Initial value | Description |
| Out | Remainder | REAL | As the data type | (*) | Remainder |

(3) Function

- Calculate the remainder obtained by dividing the dividend "In1" by the divisor "In2".
- The operation of this instruction is carried out according to the following expression.

- "Out"="In1" - ("In1"/"In2") * "In2", where the division operation in parentheses is the integer value rounded down.
- Examples of values for "In1", "In2", and "Out" are as follows:

| “In1” value | “In2” value | “Out” value |
|---|---|---|
| 8.8 | 3.2 | 2.4 |
| 8.8 | -3.2 | -0.8 |
| -8.8 | 3.2 | 0.8 |
| -8.8 | -3.2 | -2.4 |

(4) Note

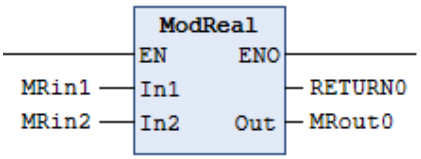
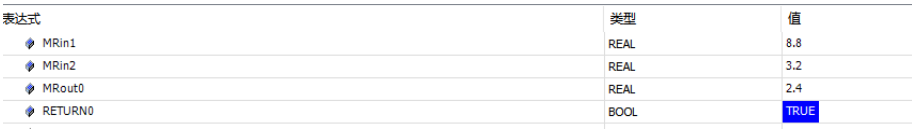
- According to the different combinations of values for "In1" and "In2", the values for "Out" are shown in the table below:

| 0 | | “In1” | | | | |
|---|---|---|---|---|---|---|
| | | Value | +∞ | -∞ | Non numerical value | |
| “In2” | 0 | Non numerical value | Non numerical value | Non numerical value | Non numerical value | Non numerical value |
| | Value | 0 | The remainder of dividing "In1" by "In2" | Non numerical value | Non numerical value | Non numerical value |
| | +∞ | 0 | “In1” value | Non numerical value | Non numerical value | Non numerical value |
| | -∞ | 0 | “In1” value | Non numerical value | Non numerical value | Non numerical value |
| | Non numerical value | Non numerical value | Non numerical value | Non numerical value | Non numerical value | Non numerical value |

- When the parameters passed to "In1" and "In2" are data types other than real numbers, an exception will occur during compilation.

(5) Program sample

| DEMO | LD | ST |
|---|---|---|
| Define variable | | VAR<br><br>MRin1: REAL;<br>MRin2: REAL;<br>MRout0: REAL;<br>RETURN0: BOOL;<br><br>END_VAR |

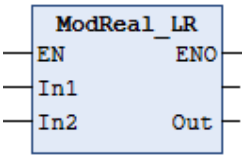
| | | |
|---|---|---|
| Program example |  | ```<br>ModReal(In1:= MRin1,<br>In2:= MRin2,<br>Out=> MRout0);<br>``` |
| Operation result |  | |

1-5-3-20. Long real remainder ModReal_LR

(1) Instruction overview

Calculate the remainder obtained by dividing the dividend "In1" by the divisor "In2".

| Long real remainder [ModReal_LR] | | | |
|---|---|---|---|
| Execution condition | Edge triggering | Suitable model | XS, XSA |
| Firmware | - | Software | XS StudioV2.2.0 and above |
| Library | XJ_DataUtils | | |

| Instruction | Name | Graph | ST |
|---|---|---|---|
| ModReal_LR | Long real remainder |  | ```<br>ModReal_LR(In1:= ,<br>In2:= ,<br>Out=> );<br>``` |

(2) Related variables

| VAR_INPUT | Name | Data type | Valid range | Initial value | Description |
|---|---|---|---|---|---|
| In1 | Dividend | LREAL | As the data type | (*) | Dividend |
| In2 | Divisor | LREAL | As the data type | (*) | Divisor |
| VAR_OUTPUT | Name | Data type | Valid range | Initial value | Description |
| Out | Remainder | LREAL | As the data type | (*) | Remainder |

(3) Function

- Calculate the remainder obtained by dividing the dividend "In1" by the divisor "In2".
- The operation of this instruction is carried out according to the following expression.
- "Out"="In1" - ("In1"/"In2") * "In2", where the division operation in parentheses is the integer value rounded down.
- Examples of values for "In1", "In2", and "Out" are as follows:

| “In1” value | “In2” value | “Out” value |
|---|---|---|
| 8.8 | 3.2 | 2.4 |

| | | |
|---|---|---|
| 8.8 | -3.2 | -0.8 |
| -8.8 | 3.2 | 0.8 |
| -8.8 | -3.2 | -2.4 |

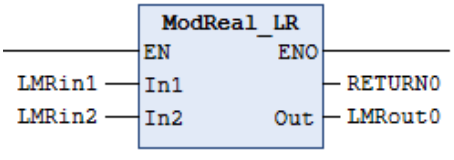
(4) Note

- According to the different combinations of values for "In1" and "In2", the values for "Out" are shown in the table below:

| 0 | | "In1" | | | | |
|---|---|---|---|---|---|---|
| | | Value | +∞ | -∞ | Non numerical value | |
| "In2" | 0 | Non numerical value | Non numerical value | Non numerical value | Non numerical value | Non numerical value |
| | Value | 0 | The remainder of dividing "In1" by "In2" | Non numerical value | Non numerical value | Non numerical value |
| | +∞ | 0 | "In1" value | Non numerical value | Non numerical value | Non numerical value |
| | -∞ | 0 | "In1" value | Non numerical value | Non numerical value | Non numerical value |
| | Non numerical value | Non numerical value | Non numerical value | Non numerical value | Non numerical value | Non numerical value |

- When the parameters passed to "In1" and "In2" are data types other than real numbers, an exception will occur during compilation.

(5) Program example

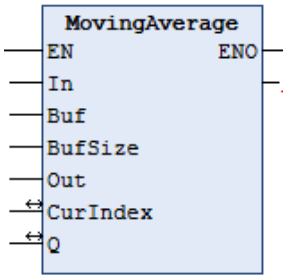
| DEMO | LD | ST |
|---|---|---|
| Define variable | | VAR<br>LMRin1: LREAL;<br>LMRin2: LREAL;<br>LMRout0: LREAL;<br>RETURN0: BOOL;<br><br>END_VAR |
| Program example |  | ModReal_LR(In1:= LMRin1,<br>In2:= LMRin2,<br>Out=> LMRout0); |
| Operation result | | |

1-5-3-21. Moving average MovingAverage

(1) Instruction overview

Calculate the moving average.

| Moving average [MovingAverage] | | | |
|---|---|---|---|
| Execution condition | Edge triggering | Suitable model | XS, XSA |
| Firmware | - | Software | XS StudioV2.2.0 and above |
| Library | XJ_DataUtils | | |

| Instruction | Name | Graph | ST |
|---|---|---|---|
| MovingAverage | moving average |  | MovingAverage(In:= ,<br>Buf:= ,<br>BufSize:= ,<br>Out:= ,<br>CurIndex:= ,<br>Q:= ); |

(2) Related variables

| VAR_INPUT | Name | Data type | Valid range | Initial value | Description |
|---|---|---|---|---|---|
| In | Used to calculate the average value data | USINT/UINT/UDINT/ ULINT/SINT/INT/ DINT/LINT/REAL/ LREAL | As the data type | - | Used to calculate the average value data |
| Buf | Used to save the arrangement of "In" | USINT/UINT/UDINT/ ULINT/SINT/INT/ DINT/LINT/REAL/ LREAL | As the data type | 1 | Used to save the arrangement of "In" |
| BufSize | Used to calculate the max element number of average value | UINT | As the data type | - | Used to calculate the max element number of average value |
| VAR_IN_OUT | Name | Data type | Valid range | Initial value | Description |
| CurIndex | Buf[] location for saving "In" | UINT | As the data type | 0 | Buf[] location for saving "In" |
| Q | True: The number of values saved to Buf[] is greater than "BufSize"; FALSE: The number of values saved to Buf[] is less than "BufSize" | BOOL | [FALSE,TRUE] | FALSE | True: The number of values saved to Buf[] is greater than "BufSize"; FALSE: The number of values saved to Buf[] is less than "BufSize" |

| VAR_OUTPUT | Name | Data type | Valid range | Initial value | Description |
|---|---|---|---|---|---|
| Out | Operation result | USINT/UINT/UDINT/ ULINT/SINT/INT/ DINT/LINT/REAL/ LREAL | As the data type | 0 | Operation result |

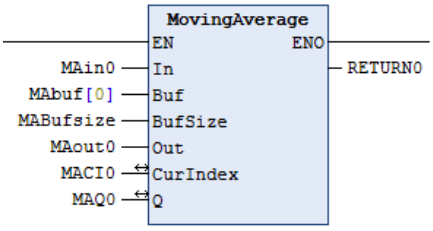
(3) Function

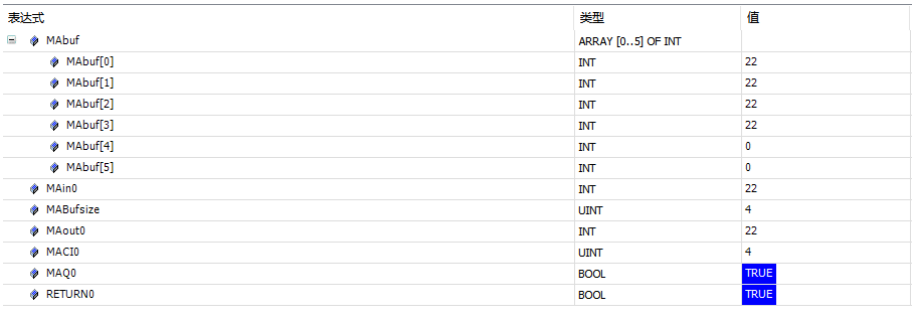
- Every time this command is executed, the input value "In" will be saved to the input value save arrangement Buf[]. The average value of the saved values is saved in the operation result "Out". The maximum number of element used to calculate the average is specified in "BufSize".

(4) Note

- The data types of "In", "Out", and Buf[] elements should be consistent, otherwise the function return value will be FALSE, Out[] remains unchanged.
- Even if the operation result exceeds the valid range of "Out", no exception will occur, and illegal values will be saved in "Out".
- When the value of "BufSize" is 0, the values of "Out" and "CurIndex" are 0, and the value of "Q" becomes True.
- After changing the value of "BufSize", "CurIndex" will maintain its current value.
- The value of "BufSize" should not exceed the size of Buf[] to avoid PLC anomalies.

(5) Program example

| DEMO | LD | ST |
|---|---|---|
| Define variable | VAR<br>MAbuf:ARRAY[0..5] OF INT;<br>MAin0: INT;<br>MABufsize: UINT;<br>MAout0: INT;<br>MACI0: UINT;<br>MAQ0: BOOL;<br>RETURN0: BOOL;<br>END_VAR | |
| Program example |  | MovingAverage(In:= MAin0,<br>Buf:= MAbuf[0],<br>BufSize:= MABufsize,<br>Out:= MAout0,<br>CurIndex:= MACI0,<br>Q:= MAQ0); |

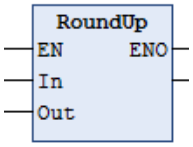
| | |
|---|---|
| Operation result |  |
| Result analysis | Every time the instruction is executed, the data of MAin0 will be saved to the starting array of MAbuf[0]; Execute MABufsize a total of times and save MABufsize data in MAbuf. |

## 1-5-3-22. Real carry RoundUp

(1) Instruction overview

Carry the first place of decimal part of a real number and take the whole number.

| Real carry [RoundUp] | | | |
|---|---|---|---|
| Execution condition | Edge triggering | Suitable model | XS, XSA |
| Firmware | - | Software | XS StudioV2.2.0 and above |
| Library | XJ_DataUtils | | |

| Instruction | Name | Graph | ST |
|---|---|---|---|
| RoundUp | Real carry |  | RoundUp(In:= , Out:= ); |

(2) Related variables

| VAR_INPUT | Name | Data type | Valid range | Initial value | Description |
|---|---|---|---|---|---|
| In | Convert object | REAL/LREAL | As the data type | - | Convert object |
| VAR_IN_OUT | Name | Data type | Valid range | Initial value | Description |
| Out | Convert result | DINT/LINT | As the data type | - | Convert result |

(3) Function

- Process the first decimal place of the real number "In" to make it an integer.

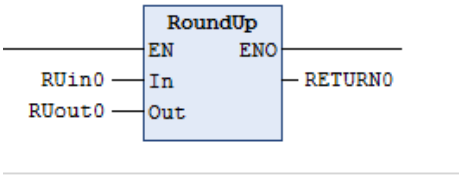
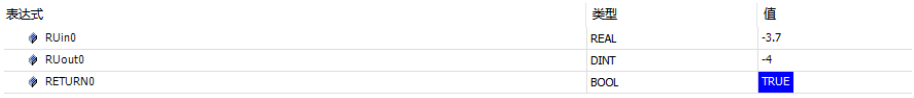
| Input value | Output value | Input value | Output value |
|---|---|---|---|
| REAL#1.6 | DINT#2 | REAL#-1.4 | DINT#-2 |
| REAL#1.5 | DINT#2 | REAL#-2.5 | DINT#-3 |
| REAL#1.4 | DINT#2 | REAL#3.2 | DINT#4 |

| | | | |
|---|---|---|---|
| REAL#2.5 | DINT#3 | REAL#3.7 | DINT#4 |
| REAL#-1.6 | DINT#-2 | REAL#-3.2 | DINT#-4 |
| REAL#-1.5 | DINT#-2 | REAL#-3.7 | DINT#-4 |

(4) Note

- When the conversion result exceeds the valid range of "Out", the value of "Out" is an incorrect value.
- The data types of "In" and "Out" should match, for example, REAL corresponds to DINT data, otherwise the function returns a value of FALSE and "Out" remains unchanged.
- "Out" requires the use of Dint and Lint types, otherwise the runtime function returns FALSE and "Out" remains unchanged.

(5) Program example

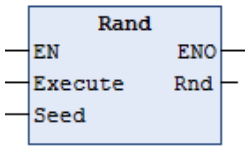
| DEMO | LD | ST |
|---|---|---|
| Define variable | | VAR<br>RUin0: REAL;<br>RUout0: DINT;<br>END_VAR |
| Program example |  | RoundUp(In:= RUin0,<br>Out:=RUout0); |
| Operation result |  | |

1-5-3-23. Generate random number RAND

(1) Instruction overview

Generate pseudo-random numbers.

| Generate random number [RAND] | | | |
|---|---|---|---|
| Execution condition | Edge triggering | Suitable model | XS, XSA |
| Firmware | - | Software | XS StudioV2.2.0 and above |
| Library | XJ_DataUtils | | |

| Instruction | Name | Graph | ST |
|---|---|---|---|
| RAND | Generate random number |  | RAND_0(Execute:= ,<br>Seed:= ,<br>Rnd=> ); |

(2) Related variables

| VAR_INPUT | Name | Data type | Valid range | Initial value | Description |
|---|---|---|---|---|---|
| Execute | Trigger | BOOL | [FALSE,TRUE] | FALSE | Trigger generation of random numbers |
| Seed | Random number mode | UINT | As the data type | 0 | Specify a sequence of random numbers 0: Not specified |
| VAR_OUTPUT | Name | Data type | Valid range | Initial value | Description |
| Rnd | Random number | LREAL | 0.00000000000000e+0～1.00000000000000e+0 | - | Random number |

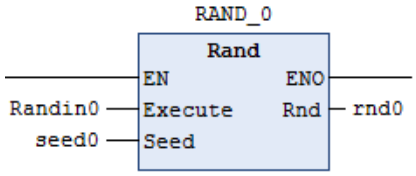
(3) Function

- Generate a random number "Rnd". The value of "Rnd" changes after each instruction execution.
- The random number mode "Seed" is used to specify a sequence of random numbers. If the value of "Seed" is the same, the same random number sequence will be generated every time the power is turned on. Therefore, reproducible random numbers can be generated.
- When the value of "Seed" is 0, a non repetitive random number will be generated. If you do not want to generate the same random number sequence every time the power is turned on, please set the value of "Seed" to 0.

(4) Note

- The value of "Rnd" is a real number between 0 and 1.

(5) Program example

| DEMO | LD | ST |
|---|---|---|
| Define variable | VAR<br>RAND_0: Rand;<br>Randin0: BOOL;<br>seed0: UINT;<br>rnd0: LREAL;<br>END_VAR | |
| Program example |  | RAND_0(Execute:= Randin0,<br>Seed:= seed0,<br>Rnd=> rnd0); |
| Operation result | | |

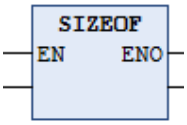
## 1-5-4. Address operation instruction

### 1-5-4-1. Data type size SIZEOF

(1) Instruction overview

Function: Execute this function to determine the number of bytes required for the given data type. Simply put, its function is to return the number of memory bytes occupied by an object or type.

| Data type size [SIZEOF] | | | |
|---|---|---|---|
| Execution condition | Edge triggering | Suitable model | XS, XSA |
| Firmware | - | Software | - |
| Library | - | | |

| Instruction | Name | Graph | ST |
|---|---|---|---|
| SIZEOF | Data type size |  | SIZEOF |

(2) Related variables

| VAR_INPUT | Name | Data type | Valid range | Initial value | Description |
|---|---|---|---|---|---|
| IN | Target data | BOOL/BYTE/WORD/DWORD/ LWORD/USINT/UINT/UDINT/ ULINT/SINT/INT/DINT/LINT/ REAL/LREAL/TIME/DATE/TOD/ DT/STRING | As the data type | - | Target data |
| VAR_IN_OUT | Name | Data type | Valid range | Initial value | Description |
| OUT | Data length | USINT/UINT/UDINT/ULINT | As the data type | - | Data length |

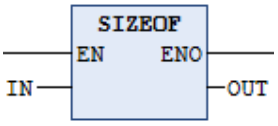
(3) Function

The return value of SIZEOF is an unsigned value, and the return value of type will be used to find the size of variable IN, the output value of OUT is in bytes, IN can be of any data type.

Return data type of SIZEOF

| SIZEOF return value | Implicit data type |
|---|---|
| 0 <= size of x < 256 | USINT |
| 256 <= size of x < 65536 | UINT |
| 65536 <= size of x < 4294967296 | UDINT |
| 4294967296 <= size of x | ULINT |

(4) Program example

| DEMO | LD | ST |
|---|---|---|
| Define variable | VAR<br>IN :ARRAY[0..4] OF INT;<br>OUT:UINT;<br>END_VAR | |
| Program example |  | 1 OUT:=SIZEOF(IN); |
| Operation result | | |
| Result analysis | The program assigns the result to OUT, and the final OUT equals 10, because the IN array consists of 5 INT integer elements, the result unit of SIZEOF is BYTE, so the final program running result is a total of 10 BYTEs, indicating that IN occupies 10 bytes of memory. | |

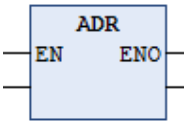
1-5-4-2. Address operator ADR

(1) Instruction overview

ADR: Obtain the memory address of the input variable, assign the result to the output variable, operator, which is an extension of the IEC 61131-3 standard.

^: Obtain the address content of the input variable and assign the result to the output variable.

| Address operator [ADR] | | | |
|---|---|---|---|
| Execution condition | Edge triggering | Suitable model | XS, XSA |
| Firmware | - | Software | - |
| Library | - | | |

| Instruction | Name | Graph | ST |
|---|---|---|---|
| ADR | Obtain address instruction |  | ADR |
| ^ | Obtain address content instruction | - | ^ |

(2) Related variables

ADR

| VAR_INOUT | Name | Data type | Valid range | Initial value | Description |
|---|---|---|---|---|---|
| IN | Target data | BOOL/BYTE/WORD/DWORD / LWORD/USINT/UINT/UDINT/ ULINT/SINT/INT/DINT/LINT/ REAL/LREAL/TIME/DATE/ TOD/DT/STRING | As the data type | - | Target data |
| OUT | Target data address | POINTER TO | - | - | Target data address |

^

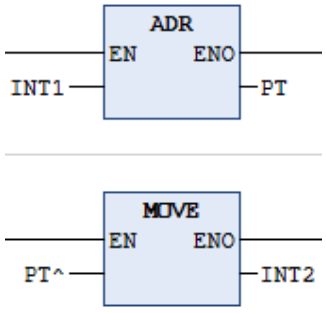
| VAR_OUTPUT | Name | Data type | Valid range | Initial value | Description |
|---|---|---|---|---|---|
| OUT | Point to address content | BOOL/BYTE/WORD/DWORD/LWORD/USINT/UINT/UDINT/ULINT/SINT/INT/DINT/LINT/REAL/ LREAL/TIME/DATE/TOD/DT/STRING | As the data type | - | Point to address content |

(3) Function

The return value of ADR is only the memory address of the variable. The length of data that can be stored in this memory address is 1 BYTE. The content in the corresponding address can be extracted using the content operator "^". To obtain the memory address of var_int1, assign its address to a pointer variable, and then extract the specific content in its corresponding address using the "^" operator and assign it to var_int2. The implementation program is as follows:

```
pt := ADR(var_int1);
var_int2:= pt^;
```

(4) Program example

| DEMO | LD | ST |
|---|---|---|
| Define variable | VAR<br>INT1:INT:=10;<br>PT :POINTER TO INT;<br>INT2:INT;<br>END_VAR | |
| Program example |  | 1 PT:=ADR(INT1);<br>2 INT2:=PT^; |

| | |
|---|---|
| Operation result | |

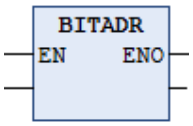


1-5-4-3. Bit address operator BITADR

(1) Instruction overview

Function: Returns the bit address information offset of the allocation variable.

| Bit address operator [BITADR] | | | |
|---|---|---|---|
| Execution condition | Edge triggering | Suitable model | XS, XSA |
| Firmware | - | Software | - |
| Library | - | | |

| Instruction | Name | Graph | ST |
|---|---|---|---|
| BITADR | Bit address operator |  | BITADR |



(2) Related variables

| VAR_INPUT | Name | Data type | Valid range | Initial value | Description |
|---|---|---|---|---|---|
| IN | Target data | BOOL | FALSE,TRUE | - | Target data |
| VAR_IN_OUT | Name | Data type | Valid range | Initial value | Description |
| OUT | Point to the target address | DWORD | As the data type | - | Point to the target address |

(3) Function

BITADR returns a bit offset numerical address as a DWORD variable type. Note that the offset value depends on whether the option type address can be obtained from the target system. The maximum value of DWORD defines the memory area as shown in the table below:

BITADR offset addresses for each address area

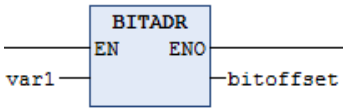
| Address area | Start address | Explanation |
|---|---|---|
| Memory | 16x40000000 | %M |
| Input | 16x80000000 | %I |
| Output | 16xC0000000 | %Q |

(4) Program example

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

```
VAR
    var1 AT %IX2.3:BOOL;
    bitoffset     : DWORD;
END_VAR
```

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




</td><td>

```
1  bitoffset:=BITADR(var1)
```

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

| 表达式 | 类型 | 值 |
|---|---|---|
| var1 | BIT | FALSE |
| bitoffset | DWORD | 16#80000013 |

</td></tr>
<tr><td>Result analysis</td><td colspan="2">

The running result is 80000013 in hexadecimal, where "2" in %IX2.3 is 2 bytes and ". 3" is the 4th bit, so its address is equal to 2*8+4=20. Convert decimal 20 to hexadecimal 14. Also, because the first address corresponding to Zone I is stored starting from 80000000, it is not difficult to understand that the actual address corresponding to hexadecimal 14 is 16#80000013. The schematic diagram is shown in the table below:

BITADR example

| Offset value | Data content |
|---|---|
| 16#80000000 | |
| 16#80000001 | |
| 16#80000002 | |
| ... | |
| ... | |
| 16#80000011 | |
| 16#80000012 | |
| 16#80000013 | bitoffset |

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

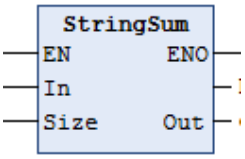
# 1-6. String command

## 1-6-1. Sum calculation StringSum

(1) Instruction overview

Calculate the SUM value of a string.

| Sum calculation [StringSum] | | | |
|---|---|---|---|
| Execution condition | Edge triggering | Suitable model | XS, XSA |
| Firmware | - | Software | XS StudioV2.2.0 and above |
| Library | XJ_DataUtils | | |

| Instruction | Name | Graph | ST |
|---|---|---|---|
| StringSum | Sum calculation |  | StringSum(In:= ,<br>Size:= ,<br>Out=> ); |

(2) Related variables

| VAR_INPUT | Name | Data type | Valid range | Initial value | Description |
|---|---|---|---|---|---|
| In | Object string | STRING | As the data type | - | Object string |
| Size | Byte size | USINT | 1,2 | 1 | Byte size of Sum value |
| VAR_IN_OUT | Name | Data type | Valid range | Initial value | Description |
| Out | Sum value | STRING | The number of bytes represented in 'size' | - | The string after Sum value conversion |

(3) Function

- Calculate the SUM value of the object string "In" (the sum of character encodings for each character). The SUM value "Out" is the number of bytes specified in the byte size "Size". "Out" is represented as a string of hexadecimal numbers, with a NULL character saved at the end.

(4) Note

- When the total character encoding of "In" exceeds the number of bits that "Size" can represent, the high bits will be rounded off, and the function will return a value of FALSE.
- When the value of "Size" exceeds the valid range, the program encounters an exception and the function returns a value of FALSE.
- When the byte count of "In" is 0 (NULL), the program encounters an exception and the function returns a value of FALSE.
- When "In" does not end with a NULL, it may result in abnormal results.

(5) Program example

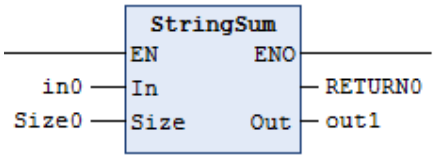
<table>
<tr><th>DEMO</th><th>LD</th><th>ST</th></tr>
<tr><td>Define variable</td><td colspan="2">

```
VAR

    Size0: USINT;
    in0:STRING;
    out1: STRING(2);
    RETURN0: BOOL;


END_VAR
```

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




</td><td>

```
StringSum(In:=in0 ,
    Size:=size0,
    Out=>out1);
```

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

| 表达式 | 类型 | 值 |
|---|---|---|
| Size0 | USINT | 2 |
| in0 | STRING | '1234' |
| out1 | STRING(2) | 'CA' |
| RETURN0 | BOOL | TRUE |

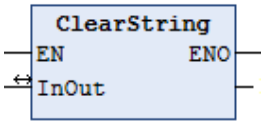
</td></tr>
<tr><td>Result analysis</td><td colspan="2">The character data in In is converted to hexadecimal, and the low bits are added, the high bits are added, to obtain the calculated value. If the low bit sum exceeds 16#F, it adds the high bit by 1. If the high bit sum exceeds 16#F, the data is not displayed; for example '1234'=31 32 33 34 =3+3+3+3 1+2+3+4=CA.</td></tr>
</table>

## 1-6-2. String clear Clearstring

(1) Instruction overview

Clear the string.

| String clear [Clearstring] | | | |
|---|---|---|---|
| Execution condition | Edge triggering | Suitable model | XS, XSA |
| Firmware | - | Software | XS StudioV2.2.0 and above |
| Library | XJ_DataUtils | | |

| Instruction | Name | Graph | ST |
|---|---|---|---|
| Clearsteing | String clear |  | `ClearString(InOut:= );` |

(2) Related variables

| VAR_IN_OUT | Name | Data type | Valid range | Initial value | Description |
|---|---|---|---|---|---|
| InOut | Clear the string | STRING | As the data type | - | The string needs to be cleared |

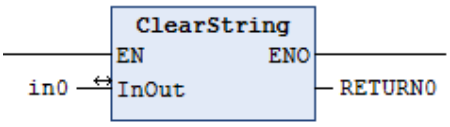
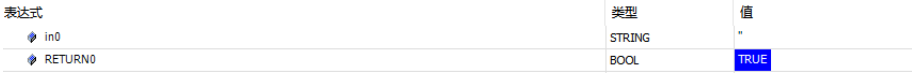
(3) Function

- Clear the string "InOut". Save NULL characters in all regions of "InOut".

(4) Note

- The return value remains true.

(5) Program example

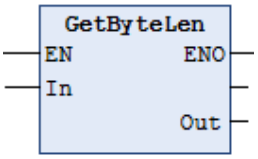
| DEMO | LD | ST |
|---|---|---|
| Define variable | | `VAR`<br>`in0:STRING;`<br>`RETURN0: BOOL;`<br>`END_VAR` |
| Program example |  | `RETURN0:=ClearString(`<br>`InOut:=string0);` |
| Operation result |  | |
| Result analysis | The value of In0 is '1234', and the string data is cleared after writing. | |

## 1-6-3. Get the number of bytes in the string GetByteLen

(1) Instruction overview

Calculate the number of bytes in a string.

| Get the number of bytes in the string [GetByteLen] | | | |
|---|---|---|---|
| Execution condition | Edge triggering | Suitable model | XS, XSA |
| Firmware | - | Software | XS StudioV2.2.0 and above |
| Library | XJ_DataUtils | | |

| Instruction | Name | Graph | ST |
|---|---|---|---|
| GetByteLen | Get the number of bytes in the string |  | GetByteLen(In:= , Out=> ); |

(2) Related variables

| VAR_INPUT | Name | Data type | Valid range | Initial value | Description |
|---|---|---|---|---|---|
| In | Object string | STRING | As the data type | - | The object string to calculate the number of bytes |
| VAR_OUTPUT | Name | Data type | Valid range | Initial value | Description |
| Out | String byte numbers | UINT | 0~1985 | - | Byte numbers |

(3) Function

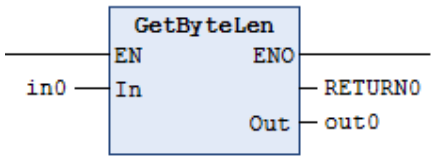
- Calculate the number of bytes from the beginning to the end of the object string "In". The NULL character at the end of the string is not within the number of bytes.

(4) Note

- The length of the input string is limited by the data type STRING.

(5) Program example

| DEMO | LD | ST |
|---|---|---|
| Define variable | | VAR<br><br>in0:STRING;<br>out0: UINT;<br><br>END_VAR |

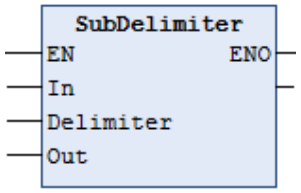
| | | |
|---|---|---|
| Program example |  | GetByteLen(In:=in0,<br>Out=>out0); |
| Operation result | | |

## 1-6-4. Real string read delimiter delete SubDelimiter

(1) Instruction overview

Read data separated by delimiters from a string and save it as a real number array.

| Real string read delimiter delete [SubDelimiter] | | | |
|---|---|---|---|
| Execution condition | Edge triggering | Suitable model | XS, XSA |
| Firmware | - | Software | XS StudioV2.2.0 and above |
| Library | XJ_DataUtils | | |

| Instruction | Name | Graph | ST |
|---|---|---|---|
| SubDelimiter | Real string read delimiter delete |  | SubDelimiter(In:= ,<br>Delimiter:= ,<br>Out:= ); |

(2) Related variables

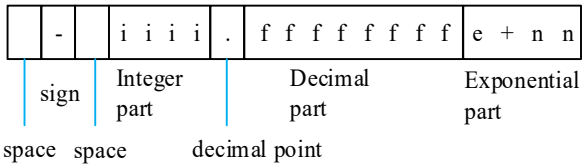
| VAR_INPUT | Name | Data type | Valid range | Initial value | Description |
|---|---|---|---|---|---|
| In | Input string | STRING | Maximum 1985 bytes | - | String to be read separated by delimiters |
| Delimiter | Delimiter | _eDELIMITER | As the data type | _eDELIMITER_COMMA | Delimiter |
| VAR_IN_OUT | Name | Data type | Valid range | Initial value | Description |
| Out | Save position real number array | REAL | As the data type | - | Save long real arrays of converted data |

(3) Function

- Convert the string separated by the delimiter "Delimiter" in the input string "In" to a storage position real number array "Out", and save it sequentially as the values of each array element.
- The data type of the delimiter "Delimiter" is the enumeration type _eDELIMITER, and the meaning of the enumeration value is as follows:

| Enum value | Meaning |
|---|---|
| _COMMA | ‘,’ (comma) |
| _TAB | ‘$T’ (Tab) |
| _SEMICOLON | ‘;’ (semicolon) |
| _SPACE | ‘ ’ (space) |

  - When the number of data separated by delimiters in "In" is greater than the number of members in "Out", excess data will be ignored.
  - When the number of data separated by delimiters in "In" is less than the number of members in "Out", the value of members with insufficient data remains unchanged.
  - The data separated by delimiters in "In" is of string type. The string format for representing real type data is as follows:




| Name | Format |
|---|---|
| Sign | After ignoring the consecutive ' ' (spaces) at the beginning of the string, consider the individual occurrences of '+' or '-' as symbols.<br>The '+' symbol can be omitted.<br>The consecutive spaces after the symbol will be ignored. |
| Integer part | The integer part is from the end of the symbol to the front of the decimal point. However, the consecutive spaces after the symbol are not within the integer part. In addition, symbols may be omitted.<br>When there is no decimal point or decimal part, consider the front of the exponential part as the integer part.<br>When there is no decimal point, decimal part, or exponential part, treat the entire string as an integer part.<br>The integer part is composed of numbers ranging from 0 to 9.<br>The integer part cannot be omitted.<br>The maximum number of digits in the integer part refers to the maximum string length of 1985 minus the number of digits in the symbol, decimal point, decimal part, exponential part, and spaces before and after the symbol. |
| Decimal point | A separate '.' (period) followed by the integer part is a decimal point.<br>When there is no decimal part, please omit the decimal point. |
| Decimal part | The decimal part starts from after the decimal point and ends before the exponential part.<br>When there is no exponential part, the entire string after the decimal point is the decimal part.<br>The decimal part is composed of numbers ranging from 0 to 9.<br>The decimal part can be omitted.<br>The maximum number of decimal places is 15.<br>When there is no decimal point, the decimal part also does not exist. |
| Exponential part | Starting from the individual 'e' followed by the decimal part and the individual '+' or '-' after it, until the end of the string, it is considered as the exponential part.<br>When there is no decimal part, consider the above string after the decimal point as the exponential part.<br>When there is no decimal point or decimal part, consider the above string after the integer part as the exponential part.<br>The numerical part of the exponential consists of numbers ranging from 0 to 9.<br>The exponential part can be omitted.<br>The maximum number of digits in the numerical part of the exponential is 3. |

An example of conversion rules is as follows:

| Format | STRING | REAL |
|---|---|---|
| Signed, with decimal point, with decimal part, without exponential part | ‘-123.4567’ | -123.4567 |

| | | |
|---|---|---|
| Signed, with decimal point, with decimal part, with exponential part | '+123.4567e+02' | 12345.67 |
| Without sign, with decimal point, with decimal part, with exponential part | '123.4567e-02' | 1.234567 |
| Without sign, without decimal point, without decimal part, without exponential part | '1' | 1 |

Conversion precautions:

- When the numerical value represented by the input string exceeds the valid range of REAL data (1.0E-44~3.402823E+38), if it is a positive number, the conversion result is Infinity; If it is a negative number, the conversion result is -Infinity.
- The input string cannot contain any other characters that are not real strings, otherwise the conversion will encounter an exception and the function will return a value of FALSE.
- When there are two or more consecutive "_" (underline) at any middle position of the input string, the conversion is normal.
- When there is an '_' (underline) between the '-' (negative sign) or '+' (positive sign) at the beginning of the input string and a number, the conversion is normal.
- When there is an '_' (underline) at the beginning or end of the input string, the conversion exception occurs (the function returns FALSE).
- When there is a separate '_' (underline) in any middle position of the input string, it will be ignored.
- When the content of the input string exceeds the precision that can be represented by the REAL data type, it will be rounded.
- When the content of the input string is closer to 0 than the value that can be represented by the data type of REAL, the conversion result is 0.
- When the decimal part of the input string exceeds the maximum of 15 digits, the program will not execute and cannot be converted. The function returns FALSE.
- When the exponential part of the input string exceeds the maximum number of 3 digits, the program will not execute and cannot be converted. The function returns FALSE.
- When the input string contains integers, decimals, and exponents with numbers other than 0~... 9, the program does not execute and cannot be converted. The function returns FALSE.
- The input string has no decimal part, but the decimal point is not omitted. The program does not execute and cannot be converted. The function returns FALSE.
- The exponential part of the input string has no numerical part, but there are symbols "e+"/"e -". The program does not execute and cannot be converted. The function returns FALSE.
- If other input strings are any non regular numbers, the program will not execute and cannot be converted. The function returns FALSE.
- If there is a space character before and after the exponential part of the input string, the program will not execute and cannot be converted. The function returns FALSE.
- The continuous space characters at the beginning and end of the input string (including before and after the delimiter) will be ignored, and the instruction will execute normally.
- When the exponential part is a decimal, the program does not execute and cannot be converted. The function returns FALSE.

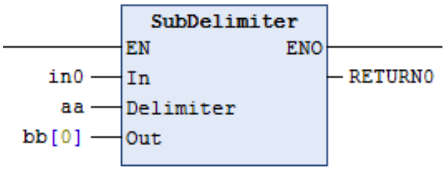
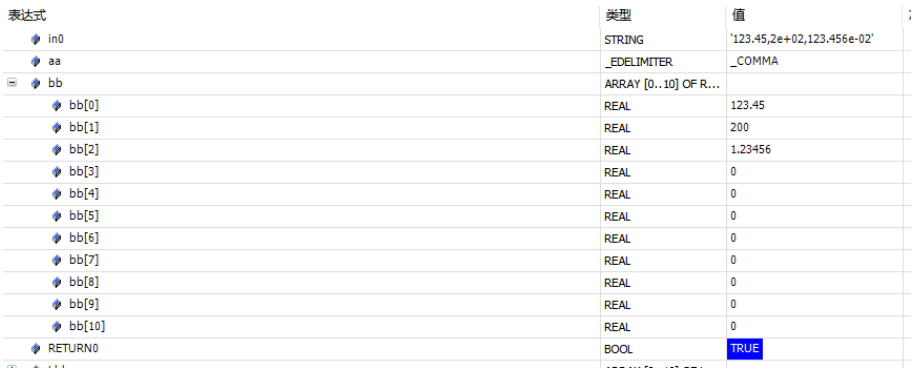
(4) Note

- When connecting with delimiters in "In", there is no data separated by delimiters. If the data does not exist, the value of the corresponding member in Out[] is REAL#0.
- When the string in "In" contains data that does not conform to the real data format, illegal data and subsequent

data are not converted. The function returns a value of FALSE, and the successfully converted values remain unchanged.

- Do not use separators for purposes other than separation in "In". Even if used for purposes other than separation, it will still be recognized as a delimiter in this instruction.
- When the conversion result exceeds the array area of Out[], it is necessary to avoid PLC exceptions, and data that does not exceed the array area of Out[] will be saved.
- When the input string is empty, the function returns True and Out remains unchanged.

(5) Program example

| DEMO | LD | ST |
|---|---|---|
| Define variable | VAR<br>in0:STRING;<br>aa: XJ_DataUtils._eDELIMITER;<br>bb:ARRAY[0..10] OF REAL;<br>RETURN0: BOOL;<br>END_VAR | |
| Program example |  | SubDelimiter(In:=in0,<br>Delimiter:=aa,<br>Out:=bb[0]); |
| Operation result |  | |

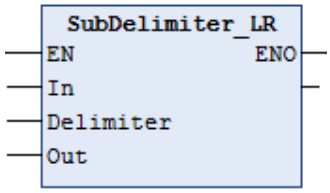
| 表达式 | 类型 | 值 |
|---|---|---|
| in0 | STRING | '123.45,2e+02,123.456e-02' |
| aa | _EDELIMITER | _COMMA |
| bb | ARRAY [0..10] OF R... | |
| bb[0] | REAL | 123.45 |
| bb[1] | REAL | 200 |
| bb[2] | REAL | 1.23456 |
| bb[3] | REAL | 0 |
| bb[4] | REAL | 0 |
| bb[5] | REAL | 0 |
| bb[6] | REAL | 0 |
| bb[7] | REAL | 0 |
| bb[8] | REAL | 0 |
| bb[9] | REAL | 0 |
| bb[10] | REAL | 0 |
| RETURN0 | BOOL | TRUE |

## 1-6-5. Long real string read delimiter delete SubDelimiter_LR

(1) Instruction overview

Read data separated by delimiters from a string and save it as a long real number array.

| Long real string read delimiter delete [SubDelimiter_LR] | | | |
|---|---|---|---|
| Execution condition | Edge triggering | Suitable model | XS, XSA |
| Firmware | - | Software | XS StudioV2.2.0 and above |
| Library | XJ_DataUtils | | |

| Instruction | Name | Graph | ST |
|---|---|---|---|
| SubDelimiter_LR | Long real string read delimiter delete |  | SubDelimiter_LR(In:= ,<br>Delimiter:= ,<br>Out:= ); |

(2) Related variables

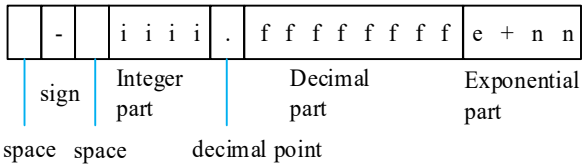
| VAR_INPUT | Name | Data type | Valid range | Initial value | Description |
|---|---|---|---|---|---|
| In | Input string | STRING | Maximum 1985 bytes | - | String to be read separated by delimiters |
| Delimiter | Delimiter | _eDELIMITER | As the data type | _eDELIMITER._COMMA | Delimiter |
| VAR_IN_OUT | Name | Data type | Valid range | Initial value | Description |
| Out | Save position real number array | LREAL | As the data type | - | Save long real arrays of converted data |

(3) Function

- Convert the string separated by the delimiter "Delimiter" in the input string "In" to a storage position real number array "Out", and save it sequentially as the values of each array element.
- The data type of the delimiter "Delimiter" is the enumeration type _eDELIMITER, and the meaning of the enumeration value is as follows:

| Enum value | Meaning |
|---|---|
| _COMMA | ‘,’ (comma) |
| _TAB | ‘$T’ (Tab) |
| _SEMICOLON | ‘;’ (semicolon) |
| _SPACE | ‘ ’ (space) |

  - ◆ When the number of data separated by delimiters in "In" is greater than the number of members in "Out", excess data will be ignored.
  - ◆ When the number of data separated by delimiters in "In" is less than the number of members in "Out", the value of members with insufficient data remains unchanged.
  - ◆ The data separated by delimiters in "In" is of string type. The string format for representing real type data is as follows:




| Name | Format |
|---|---|
| Sign | After ignoring the consecutive ' ' (spaces) at the beginning of the string, consider the individual occurrences of '+' or '-' as symbols.<br>The '+' symbol can be omitted.<br>The consecutive spaces after the symbol will be ignored. |
| Integer part | The integer part is from the end of the symbol to the front of the decimal point. However, the consecutive spaces after the symbol are not within the integer part. In addition, symbols may be omitted.<br>When there is no decimal point or decimal part, consider the front of the exponential part as the integer part.<br>When there is no decimal point, decimal part, or exponential part, treat the entire string as an integer part.<br>The integer part is composed of numbers ranging from 0 to 9.<br>The integer part cannot be omitted.<br>The maximum number of digits in the integer part refers to the maximum string length of 1985 minus the number of digits in the symbol, decimal point, decimal part, exponential part, and spaces before and after the symbol. |
| Decimal point | A separate '.' (period) followed by the integer part is a decimal point.<br>When there is no decimal part, please omit the decimal point. |
| Decimal part | The decimal part starts from after the decimal point and ends before the exponential part.<br>When there is no exponential part, the entire string after the decimal point is the decimal part.<br>The decimal part is composed of numbers ranging from 0 to 9.<br>The decimal part can be omitted.<br>The maximum number of decimal places is 15.<br>When there is no decimal point, the decimal part also does not exist. |
| Exponential part | Starting from the individual 'e' followed by the decimal part and the individual '+' or '-' after it, until the end of the string, it is considered as the exponential part.<br>When there is no decimal part, consider the above string after the decimal point as the exponential part.<br>When there is no decimal point or decimal part, consider the above string after the integer part as the exponential part.<br>The numerical part of the exponential consists of numbers ranging from 0 to 9.<br>The exponential part can be omitted.<br>The maximum number of digits in the numerical part of the exponential is 3. |

An example of conversion rules is as follows:

| Format | STRING | REAL |
|---|---|---|
| Signed, with decimal point, with decimal part, without exponential part | ‘-123.4567’ | -123.4567 |

| | | |
|---|---|---|
| Signed, with decimal point, with decimal part, with exponential part | ‘+123.4567e+02’ | 12345.67 |
| Without sign, with decimal point, with decimal part, with exponential part | ‘123.4567e-02’ | 1.234567 |
| Without sign, without decimal point, without decimal part, without exponential part | ‘1’ | 1 |

Conversion precautions:

- When the numerical value represented by the input string exceeds the valid range of LREAL data (4.94065645841247E-324~1.7976931348623157E+308), if it is a positive number, the conversion result is Infinity; If it is a negative number, the conversion result is -Infinity.
- When the input string does not represent a long real number value, the conversion result is 0, and the function return value is FALSE.
- When there are two or more consecutive "_" (underline) at any middle position of the input string, the conversion is normal.
- When there is an '_' (underline) between the '-' (negative sign) or '+' (positive sign) at the beginning of the input string and a number, the conversion is normal.
- When there is an '_' (underline) at the beginning or end of the input string, the conversion exception occurs (the function returns FALSE).
- When there is a separate '_' (underline) in any middle position of the input string, it will be ignored.
- When the content of the input string exceeds the precision that can be represented by the LREAL data type, it will be rounded.
- When the content of the input string is closer to 0 than the value that can be represented by the LREAL data type, the conversion result is 0.
- When the decimal part of the input string exceeds the maximum of 15 digits, the program will not execute and cannot be converted. The function returns FALSE.
- When the exponential part of the input string exceeds the maximum number of 3 digits, the program will not execute and cannot be converted. The function returns FALSE.
- When the input string contains integers, decimals, and exponents that are not between 0 and 9, the program does not execute and cannot be converted. The function returns FALSE.
- The input string has no decimal part, but the decimal point is not omitted. The program does not execute and cannot be converted. The function returns FALSE.
- The exponential part of the input string has no numerical part, but there are symbols 'e+'/'e-'. The program does not execute and cannot be converted. The function returns FALSE.
- If other input strings are any non regular numbers, the program will not execute and cannot be converted. The function returns FALSE.
- If there is a space character before and after the ‘+/-’ of exponential part of the input string, the program will not execute and cannot be converted. The function returns FALSE.
- The continuous space characters at the beginning and end of the input string (including before and after the delimiter) will be ignored, and the instruction will execute normally.
- When the exponential part is a decimal, the program does not execute and cannot be converted. The function returns FALSE.

(4) Note

- "In" contains data separated by delimiters. If the data does not exist, the value of the corresponding member in Out[] is LREAL#0.
- When the string in "In" contains data that does not conform to the real data format, illegal data and subsequent

data are not converted. The function returns a value of FALSE, and the successfully converted values remain unchanged.

- Do not use separators for purposes other than separation in "In". Even if used for purposes other than separation, it will still be recognized as a delimiter in this instruction.
- When the conversion result exceeds the array area of Out[], it is necessary to avoid PLC exceptions, and data that does not exceed the array area of Out [] will be saved.
- When the input string is empty, the function returns a value of True, Out has no change.

(5) Program example

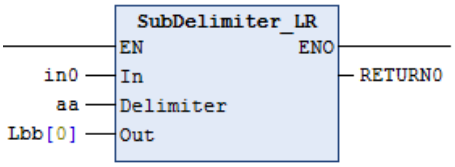
<table>
<tr><th>DEMO</th><th>LD</th><th>ST</th></tr>
<tr><td>Define variable</td><td colspan="2">

```
VAR

    in0:STRING;
    aa: XJ_DataUtils._eDELIMITER;
    Lbb:ARRAY[0..10] OF LREAL;
    RETURN0: BOOL;

END_VAR
```

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




</td><td>

```
SubDelimiter_LR(In:=in0,
    Delimiter:=aa ,
    Out:=lbb[0] );
```

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

| 表达式 | 类型 | 值 |
|---|---|---|
| in0 | STRING | '123.45,2e+02,123.456e-02' |
| aa | _EDELIMITER | _COMMA |
| Lbb | ARRAY [0..10] OF L... | |
| Lbb[0] | LREAL | 123.45 |
| Lbb[1] | LREAL | 200 |
| Lbb[2] | LREAL | 1.23456 |
| Lbb[3] | LREAL | 0 |
| Lbb[4] | LREAL | 0 |
| Lbb[5] | LREAL | 0 |
| Lbb[6] | LREAL | 0 |
| Lbb[7] | LREAL | 0 |
| Lbb[8] | LREAL | 0 |
| Lbb[9] | LREAL | 0 |
| Lbb[10] | LREAL | 0 |
| RETURN0 | BOOL | TRUE |

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

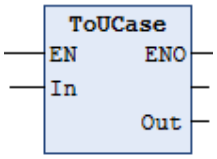
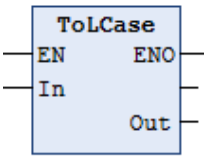
## 1-6-6. String letter conversion ToUCase/ToLCase

(1) Instruction overview

ToUCase: Converts all half width letters in a string to uppercase letters.

ToLCase: Convert all half width letters in a string to lowercase letters.

| String letter conversion [ToUCase/ToLCase] | | | |
|---|---|---|---|
| Execution condition | Edge triggering | Suitable model | XS, XSA |
| Firmware | - | Software | XS StudioV2.2.0 and above |
| Library | XJ_DataUtils | | |

| Instruction | Name | Graph | ST |
|---|---|---|---|
| ToUCase | String uppercase conversion |  | ToUCase(In:= , Out=> ); |
| ToLCase | String lowercase conversion |  | ToLCase(In:= , Out=> ); |

(2) Related variables

| VAR_INPUT | Name | Data type | Valid range | Initial value | Description |
|---|---|---|---|---|---|
| In | Conversion object | STRING | Maximum 1985 bytes | - | Converted object string |
| VAR_OUTPUT | Name | Data type | Valid range | Initial value | Description |
| Out | Conversion result | STRING | Maximum 1985 bytes | - | Converted string |

(3) Function

- ToUCase: Convert the half width letters of the converted object string "In" to uppercase letters.
- ToLCase: Convert the half width letters of the conversion object string "In" to lowercase letters.
- All instructions contain a NULL character at the end of the string when outputting. Characters with non half width letters are not affected.

(4) Note

- Full width letters do not belong to the conversion object.
- When the text code of "In" is abnormal, an exception may occur, and the "Out" value is inaccurate.
- When "In" does not end with a NULL character, it may cause abnormal conversion results.
- When the conversion result exceeds the size of "Out", only the conversion value of Out length will be displayed.

(5) Program example

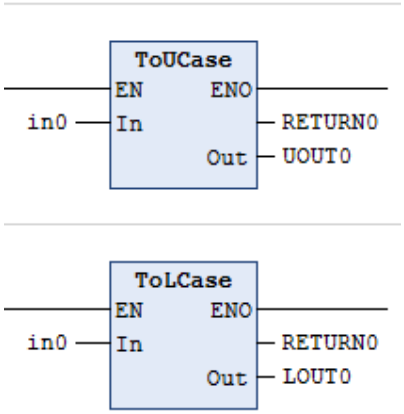
<table>
<tr><th>DEMO</th><th>LD</th><th>ST</th></tr>
<tr><td>Define variable</td><td colspan="2">

```
VAR

    in0:STRING;
    UOUT0: STRING(1985);
    LOUT0: STRING(1985);
    RETURN0: BOOL;

END_VAR
```

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

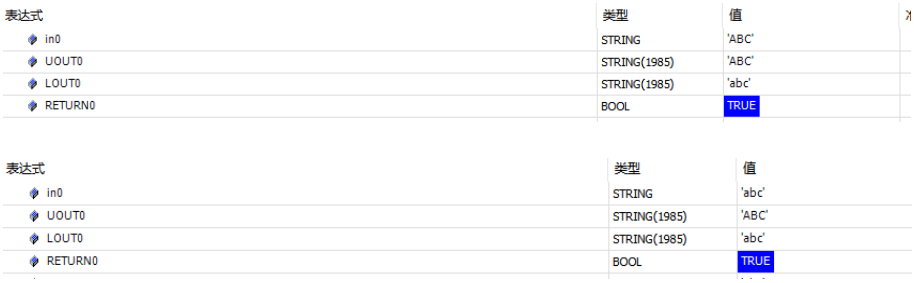



</td><td>

```
ToLCase(In:=in0 ,
    Out=>Lout0);

ToUCase(In:=in0 ,
    Out=>Uout0);
```

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



| 表达式 | 类型 | 值 |
|---|---|---|
| in0 | STRING | 'ABC' |
| UOUT0 | STRING(1985) | 'ABC' |
| LOUT0 | STRING(1985) | 'abc' |
| RETURN0 | BOOL | TRUE |

| 表达式 | 类型 | 值 |
|---|---|---|
| in0 | STRING | 'abc' |
| UOUT0 | STRING(1985) | 'ABC' |
| LOUT0 | STRING(1985) | 'abc' |
| RETURN0 | BOOL | TRUE |

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

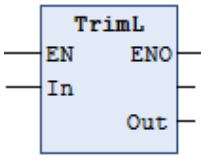
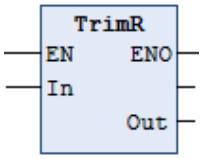
## 1-6-7. String left/right adjustment TrimL/TrimR

(1) Instruction overview

TrimL: Remove spaces at the beginning of the string.

TrimR: Remove spaces at the end of the string.

| String left/right adjustment [TrimL/TrimR] | | | |
|---|---|---|---|
| Execution condition | Edge triggering | Suitable model | XS, XSA |
| Firmware | - | Software | XS StudioV2.2.0 and above |
| Library | XJ_DataUtils | | |

| Instruction | Name | Graph | ST |
|---|---|---|---|
| TrimL | Left adjustment of string |  | `TrimL(In:= , Out=> );` |
| TrimR | Right adjustment of string |  | `TrimR(In:= , Out=> );` |

(2) Related variables

| VAR_INPUT | Name | Data type | Valid range | Initial value | Description |
|---|---|---|---|---|---|
| In | Delete object | STRING | Maximum 1985 bytes | - | Object string deleted |
| VAR_OUTPUT | Name | Data type | Valid range | Initial value | Description |
| Out | Delete result | STRING | Maximum 1985 bytes | - | String after deletion |

(3) Function

- TrimL: Remove spaces from the beginning of the object string "In" to the first character that appears. When there are no spaces at the beginning, no processing is done.
- TrimR: Delete spaces from the last character of the object string "In" to the end. When there are no spaces at the end, no processing is done.
- Both instructions will output results including the NULL character at the end of the string.

(4) Note

- Both instructions will output results including the NULL character at the end of the string.

(5) Program example

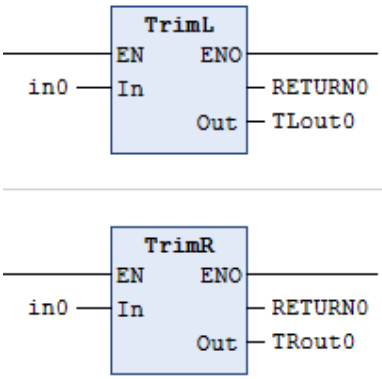
<table>
<tr><th>DEMO</th><th>LD</th><th>ST</th></tr>
<tr><td>Define variable</td><td colspan="2">

```
VAR

    in0:STRING;
    TLout0: STRING(1985);
    TRout0: STRING(1985);

END_VAR
```

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




</td><td>

```
TrimL(In:=string0 ,
    Out=>outl[2] );
TrimR(In:=string0 ,
    Out=>outl[3] );
```

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

| 表达式 | 类型 | 值 |
|---|---|---|
| in0 | STRING | ' ABC ' |
| TLout0 | STRING(1985) | 'ABC ' |
| TRout0 | STRING(1985) | ' ABC' |

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

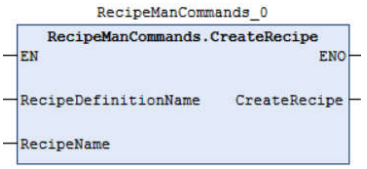
## 1-7. Recipe instructions

### 1-7-1. Create new recipe CreateRecipe

(1) Instruction overview

Create a new standard recipe file in the recipe definition.

| Create new recipe [CreateRecipe] | | | |
|---|---|---|---|
| Execution condition | Edge triggering | Suitable model | XS, XSA |
| Firmware | - | Software | - |
| Library | Recipe Management | | |

| Instruction | Name | Graph | ST |
|---|---|---|---|
| CreateRecipe | Create new recipe |  | RecipeManCommands_0.CreateRecipe(<br>RecipeDefinitionName:= ,<br>RecipeName:= ); |

(2) Related variables

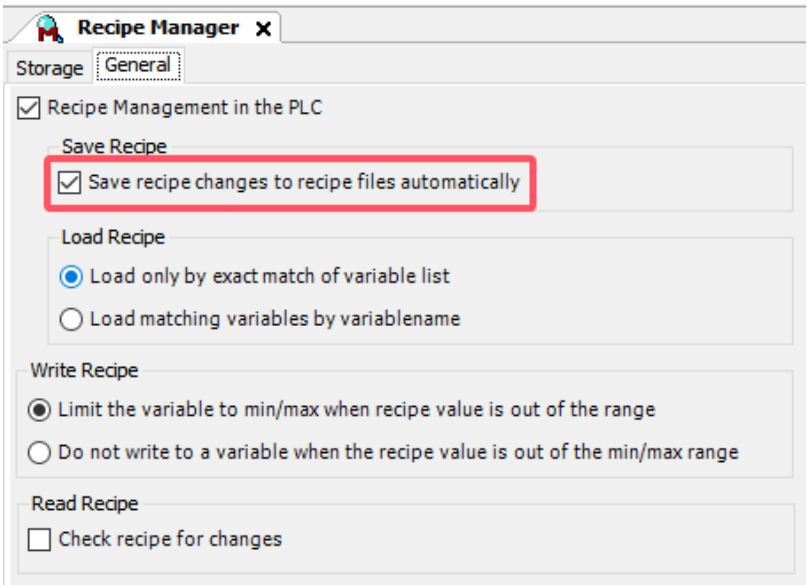
| VAR_INPUT | Name | Data type | Valid range | Initial value | Description |
|---|---|---|---|---|---|
| RecipeDefinitionName | Name of formula definition | STRING | As the data type | - | Name of formula definition |
| RecipeName | Recipe name | STRING | As the data type | - | Recipe name |
| VAR_OUTPUT | Name | Data type | Valid range | Initial value | Description |
| CreateRecipe | Return value | DWORD | As the data type | - | Return value |

(3) Function

- Create a new standard recipe file and write the current values of variables in the recipe definition to the recipe file and save it to the default folder. The name of the recipe file is specified by the "RecipeName" pin.

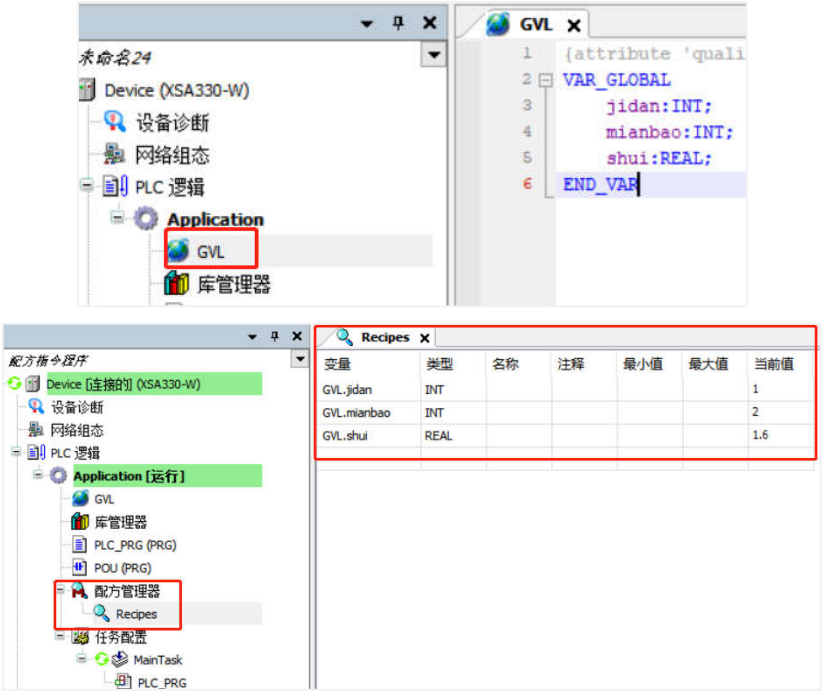
(4) Note

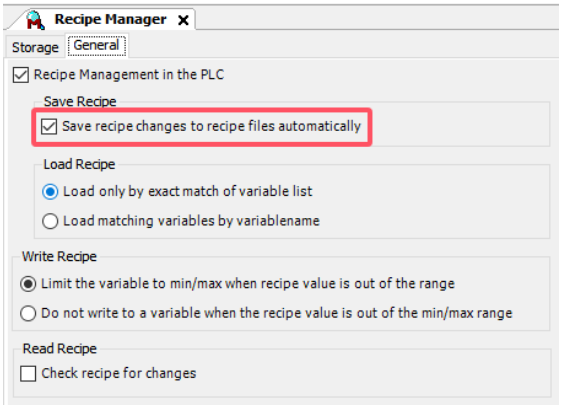
- You need to check "Save recipe changes to recipe files automatically" in the "Recipe Manager" before executing Create Recipe.




- After executing CreateRecipe, a new recipe file will be generated in the "PlcLogic" folder of the PLC root directory.

(5) Program example

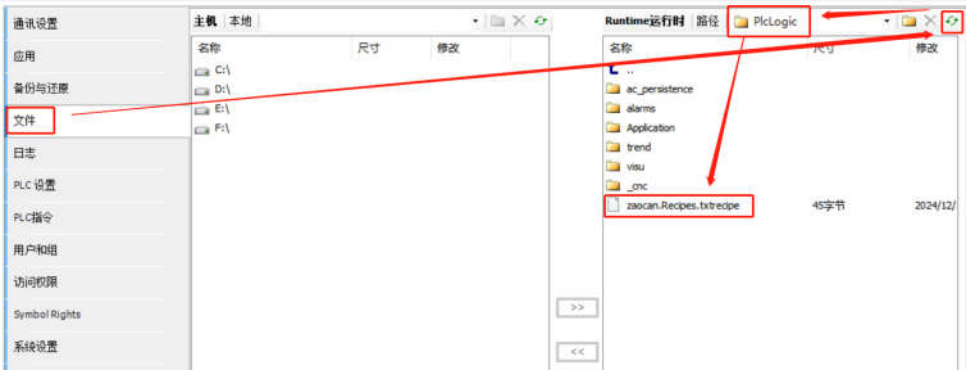
| DEMO | LD | ST |
|---|---|---|
| Add recipe manager, recipe definition, and define variables |  | |

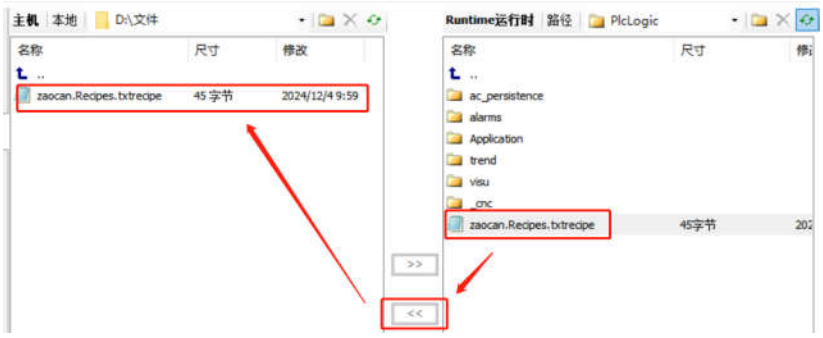
<table>
<tr><th>DEMO</th><th>LD</th><th>ST</th></tr>
<tr><td></td><td colspan="2"> 

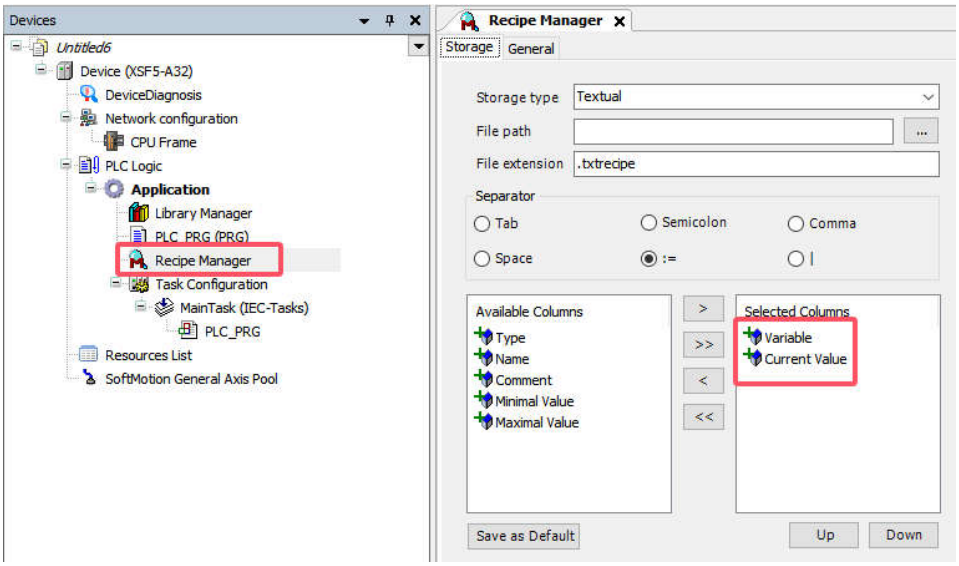

```
VAR
    RTRIG               :ARRAY[0..20]OF Standard.R_TRIG;
    chufa               :ARRAY[0..20]OF BOOL;
    RecipeManCommands_0: RecipeManCommands;
    fanhui              :ARRAY[0..20]OF DWORD;
    ifanhui             :INT;
    name                :ARRAY[0..20]OF STRING;
END_VAR
```</td></tr>
<tr><td>Program example</td><td>
</td><td>

```
RTRIG[0](CLK:= chufa[0], Q=> );
IF RTRIG[0].Q THEN
    fanhui[0]:=RecipeManCommands_0.CreateRecipe(
        RecipeDefinitionName:= 'Recipes',
        RecipeName:= name[0]);
END_IF
```</td></tr>
<tr><td>Operation result</td><td colspan="2">

```
RTRIG[0](CLK TRUE := chufa[0] TRUE, Q=> );
IF RTRIG[0].Q FALSE THEN
    fanhui[0]  0  :=RecipeManCommands_0.CreateRecipe(
        RecipeDefinitionName:= 'Recipes',
        RecipeName:= name[0]  'zaocan'  );
END_IF
```





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

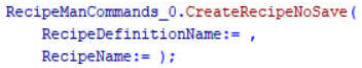
| DEMO | LD | ST |
| --- | --- | --- |
| | | |
| Result analysis | Create a new recipe file in the "PlcLogic" folder under the PLC root directory. You can transfer the file to a computer disk and open it using Notepad or other methods to view recipe variables and other parameters. If you want to modify display content and other configurations, you can configure them in the "Recipe Manager" interface.<br>  | |

## 1-7-2. Create new recipe CreateRecipeNoSave

(1) Instruction overivew

Create a new standard recipe file in the recipe definition, which needs to be used in conjunction with the SaveRecipe command.

| Create new recipe [CreateRecipeNoSave] | | | |
|---|---|---|---|
| Execution condition | Edge triggering | Suitable model | XS, XSA |
| Firmware | - | Software | - |
| Library | Recipe Management | | |

| Instruction | Name | Graph | ST |
|---|---|---|---|
| CreateRecipeNoSave | Create new recipe | |  |



```
RecipeManCommands_0.CreateRecipeNoSave(
    RecipeDefinitionName:= ,
    RecipeName:= );
```

(2) Related variables

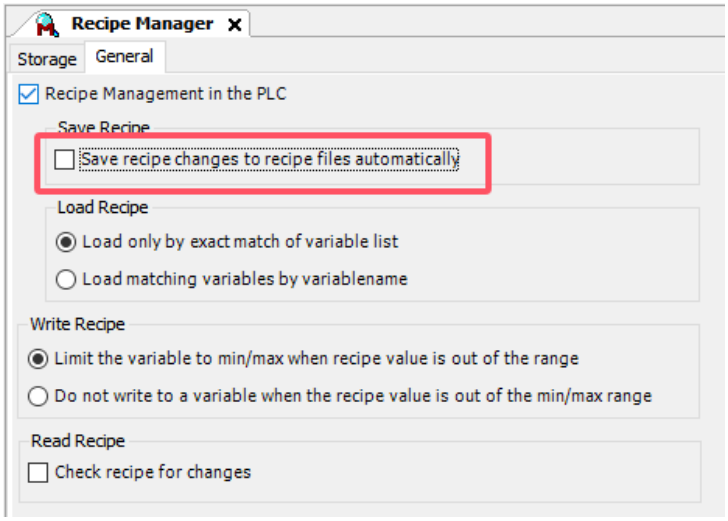
| VAR_INPUT | Name | Data type | Valid range | Initial value | Description |
|---|---|---|---|---|---|
| RecipeDefinitionName | Name of recipe definition | STRING | As the data type | - | Name of recipe definition |
| RecipeName | Recipe name | STRING | As the data type | - | Name of the recipe created |
| VAR_OUTPUT | Name | Data type | Valid range | Initial value | Description |
| CreateRecipeNoSave | Return value | DWORD | As the data type | - | Return value |

(3) Function

- The function is to create a new standard recipe file and write the variable values in the recipe definition into it. By default, a new recipe file will not be generated in the default folder of the PLC. It needs to be used in conjunction with SaveRecipe to generate a new recipe file. The name of the recipe file is specified by the "RecipeName" pin.

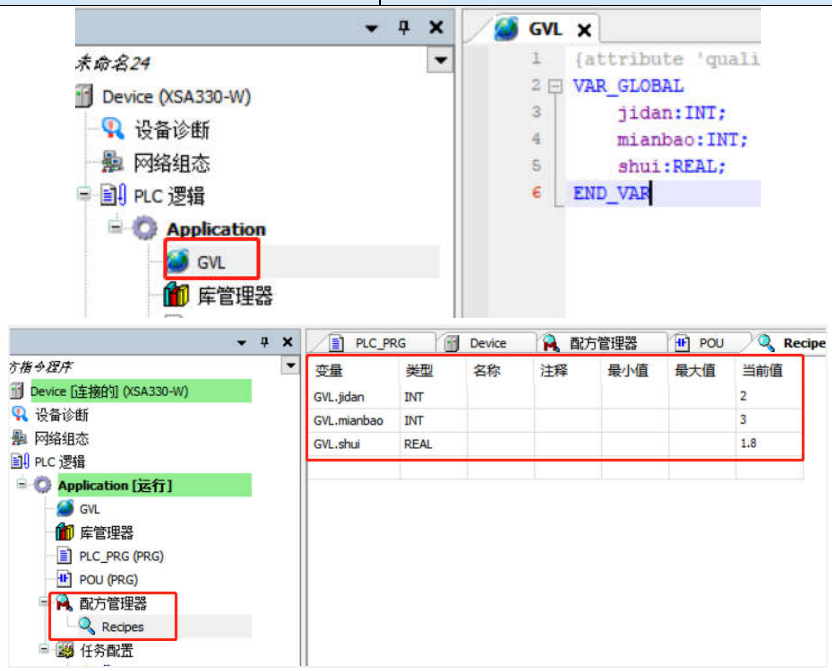
(4) Note

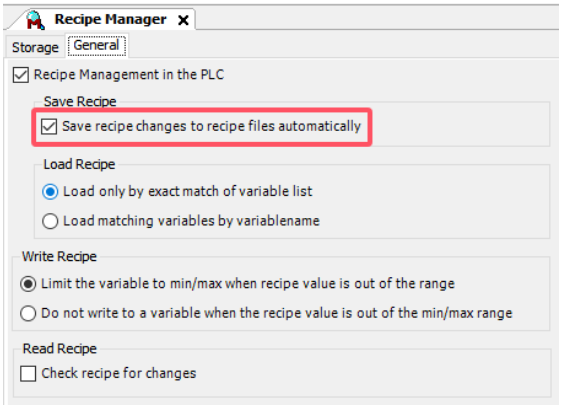
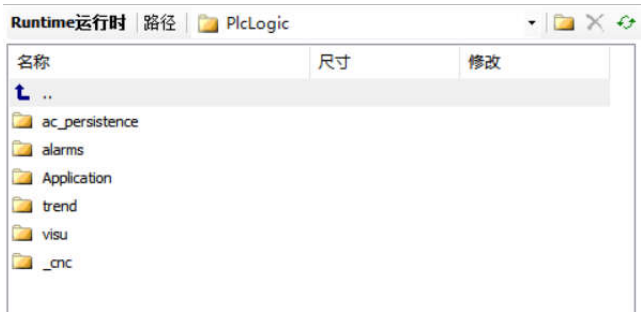
- In order to generate a new recipe file in the "PlcLogic" folder of the PLC root directory. You need to uncheck "save recipe changes to recipe files atomatically" in the "Recipe Manager" in advance, and then execute "SaveRecipe" after executing "CreateRecipeNoSave".




- If " save recipe changes to recipe files atomatically " is selected in the "Recipe Manager", executing the "CreateRecipeNoSave" command and then executing the "SaveRecipe" command will return the value 16387 (16 # 4003). The recipe does not exist.

(5) Program example

| DEMO | LD | ST |
|---|---|---|
| Add recipe manager, recipe definition, and define variables |  | |

| DEMO | LD | ST |
| --- | --- | --- |
| |  <br>  <br>VAR<br>RTRIG :ARRAY[0..20]OF Standard.R_TRIG;<br>chufa :ARRAY[0..20]OF BOOL;<br>RecipeManCommands_0: RecipeManCommands;<br>fanhui :ARRAY[0..20]OF DWORD;<br>ifanhui :INT;<br>name :ARRAY[0..20]OF STRING;<br>END_VAR |
| Program example | | RTRIG[1](CLK:= chufa[1], Q=> );<br>IF RTRIG[1].Q THEN<br>fanhui[1]:=RecipeManCommands_0.CreateRecipeNoSave(<br>RecipeDefinitionName:= 'Recipes',<br>RecipeName:= name[1]);<br>END_IF<br><br>RTRIG[11](CLK:= chufa[11], Q=> );<br>IF RTRIG[11].Q THEN<br>fanhui[11]:=RecipeManCommands_0.SaveRecipe(<br>RecipeDefinitionName:= 'Recipes',<br>RecipeName:= name[1]);<br>END_IF |
| Operation result | RTRIG[1](CLK TRUE := chufa[1] TRUE , Q=> );<br>IF RTRIG[1].Q FALSE THEN<br>fanhui[1] 0 :=RecipeManCommands_0.CreateRecipeNoSave(<br>RecipeDefinitionName:= 'Recipes',<br>RecipeName:= name[1] 'zancan111' );<br>END_IF<br><br>RTRIG[11](CLK TRUE := chufa[11] TRUE , Q=> );<br>IF RTRIG[11].Q FALSE THEN<br>fanhui[11] 0 :=RecipeManCommands_0.SaveRecipe(<br>RecipeDefinitionName:= 'Recipes',<br>RecipeName:= name[1] 'zancan111' );<br>END_IF RETURN | |

| DEMO | LD | ST |
| --- | --- | --- |
| | | |

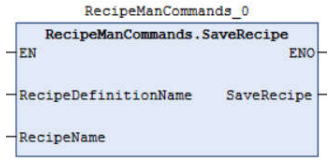
Runtime运行时
路径
PlcLogic
名称
尺寸
..
ac_persistence
alarms
Application
trend
visu
_cnc
zancan111.Recipes.txtrecipe
45字节

## 1-7-3. Save recipe SaveRecipe

(1) Instruction overview

Store the recipe in the recipe file.

| Save recipe [SaveRecipe] | | | |
|---|---|---|---|
| Execution condition | Edge triggering | Suitable model | XS, XSA |
| Firmware | - | Software | - |
| Library | Recipe Management | | |

| Instruction | Name | Graph | ST |
|---|---|---|---|
| SaveRecipe | Save recipe |  | RecipeManCommands_0.SaveRecipe(<br>RecipeDefinitionName:= ,<br>RecipeName:= ); |

(2) Related variables

| VAR_INPUT | Name | Data type | Valid range | Initial value | Description |
|---|---|---|---|---|---|
| RecipeDefinitionName | Name of recipe definition | STRING | As the data type | - | Name of recipe definition |
| RecipeName | Recipe name | STRING | As the data type | - | Name of the recipe created |
| VAR_OUTPUT | Name | Data type | Valid range | Initial value | Description |
| SaveRecipe | Return value | DWORD | As the data type | - | Return value |

(3) Function

- Store the recipe in the recipe file.

## 1-7-4. Delete recipe DeleteRecipe

(1) Instruction overview

Delete the recipe file.

| Delete recipe [DeleteRecipe] | | | |
|---|---|---|---|
| Execution condition | Edge triggering | Suitable model | XS, XSA |
| Firmware | - | Software | - |
| Library | Recipe Management | | |

| Instruction | Name | Graph | ST |
|---|---|---|---|
| DeleteRecipe | Delete recipe | | RecipeManCommands_0.DeleteRecipe(<br>RecipeDefinitionName:= ,<br>RecipeName:= ); |

(2) Related variables

| VAR_INPUT | Name | Data type | Valid range | Initial value | Description |
|---|---|---|---|---|---|
| RecipeDefinitionName | Name of recipe definition | STRING | As the data type | - | Name of recipe definition |
| RecipeName | Recipe name | STRING | As the data type | - | Name of the recipe created |
| VAR_OUTPUT | Name | Data type | Valid range | Initial value | Description |
| DeleteRecipe | Return value | DWORD | As the data type | - | Return value |

(3) Function

- Check " save recipe changes to recipe files atomatically " in the "Recipe Manager", and then execute "Delete Recipe" to delete the recipe files in the "PlcLogic" folder under the PLC root directory. And executing DeleteRecipeFile does not require checking "save recipe changes to recipe files atomatically".

## 1-7-5. Delete recipe file DeleteRecipeFile

(1) Instruction overview

Delete recipe file.

| Delete recipe file [DeleteRecipeFile] | | | |
|---|---|---|---|
| Execution condition | Edge triggering | Suitable model | XS, XSA |
| Firmware | - | Software | - |
| Library | Recipe Management | | |

| Instruction | Name | Graph | ST |
|---|---|---|---|
| DeleteRecipeFile | Delete recipe file | | RecipeManCommands_0.DeleteRecipeFile(<br>RecipeDefinitionName:= ,<br>RecipeName:= ); |

(2) Related variables

| VAR_INPUT | Name | Data type | Valid range | Initial value | Description |
|---|---|---|---|---|---|
| RecipeDefinitionName | Name of recipe definition | STRING | As the data type | - | Name of recipe definition |
| RecipeName | Recipe name | STRING | As the data type | - | Name of the recipe created |
| VAR_OUTPUT | Name | Data type | Valid range | Initial value | Description |
| DeleteRecipeFile | Return value | DWORD | As the data type | - | Return value |

(3) Function

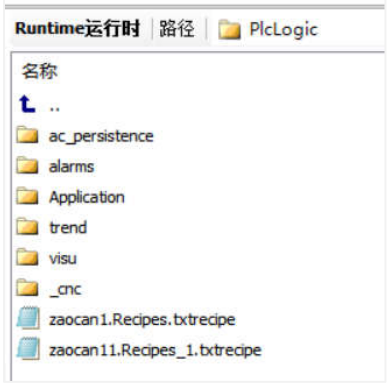
- Delete the recipe file with the name "RecipeName" in the "PlcLogic" folder under the PLC root directory.

(4) Note

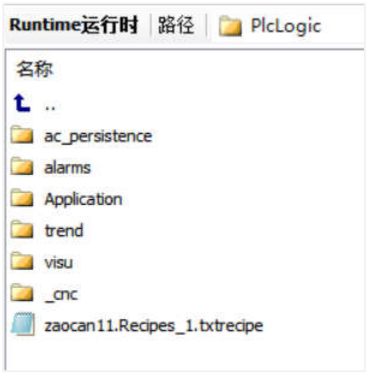
- The 'RecipeName' pin must be filled with the name of an existing recipe file, otherwise it returns 16#4000, indicating that the recipe file was not found.

(5) Program example

| DEMO | LD | ST |
|---|---|---|
| Add recipe manager, recipe definition, | | |

<table>
<tr><td>and define variables</td><td> 

<pre>VAR
    RTRIG              :ARRAY[0..20]OF Standard.R_TRIG;
    chufa              :ARRAY[0..20]OF BOOL;
    RecipeManCommands_0: RecipeManCommands;
    fanhui             :ARRAY[0..20]OF DWORD;
    ifanhui            :INT;
    name               :ARRAY[0..20]OF STRING;
END_VAR</pre></td></tr>
<tr><td>Program example</td><td>

<pre>RTRIG[3](CLK:= chufa[3], Q=> );
IF RTRIG[3].Q THEN
    fanhui[3]:=RecipeManCommands_0.DeleteRecipeFile(
        RecipeDefinitionName:= 'Recipes',
        RecipeName:= name[3]);
END_IF</pre></td></tr>
<tr><td>Operation result</td><td><pre>RTRIG[3](CLK TRUE := chufa[3] TRUE , Q=> );
IF RTRIG[3].Q FALSE THEN
    fanhui[3]  0  :=RecipeManCommands_0.DeleteRecipeFile(
        RecipeDefinitionName:= 'Recipes',
        RecipeName:= name[3]  'zaocan1'  );
END_IF</pre>

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

## 1-7-6. Obtain the quantity of recipe GetRecipeCount

(1) Instruction overview

Return the number of recipe defined for the corresponding recipe.

| Obtain the quantity of recipe [GetRecipeCount] | | | |
|---|---|---|---|
| Execution condition | Edge triggering | Suitable model | XS, XSA |
| Firmware | - | Software | - |
| Library | Recipe Management | | |

| Instruction | Name | Graph | ST |
|---|---|---|---|
| GetRecipe Count | Obtain the quantity of recipe | | ifanhui:=RecipeManCommands_0.GetRecipeCount(<br>RecipeDefinitionName:= ); |

(2) Related variables

| VAR_INPUT | Name | Data type | Valid range | Initial value | Description |
|---|---|---|---|---|---|
| RecipeDefinitionName | Name of recipe definition | STRING | As the data type | - | Name of recipe definition |
| VAR_OUTPUT | Name | Data type | Valid range | Initial value | Description |
| GetRecipeCount | Return value | INT | As the data type | - | Return value |

(3) Function

- Obtain the number of recipe defined for the corresponding recipe.

(4) Program example

| DEMO | LD | ST |
|---|---|---|
| Add recipe manager, recipe definition, and define variables | | |

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


```
VAR
    RTRIG               :ARRAY[0..20]OF Standard.R_TRIG;
    chufa               :ARRAY[0..20]OF BOOL;
    RecipeManCommands_0: RecipeManCommands;
    fanhui              :ARRAY[0..20]OF DWORD;
    ifanhui             :INT;
    name                :ARRAY[0..20]OF STRING;
END_VAR
```
</td></tr>
<tr><td>Program example</td><td>
</td><td>

```
RTRIG[4](CLK:= chufa[4], Q=> );
IF RTRIG[4].Q THEN
    ifanhui:=RecipeManCommands_0.GetRecipeCount(
        RecipeDefinitionName:= 'Recipes');
END_IF
```
</td></tr>
<tr><td>Operation result</td><td colspan="2">

```
RTRIG[4](CLK TRUE := chufa[4] TRUE, Q=> );
IF RTRIG[4].Q FALSE THEN
    ifanhui 2 :=RecipeManCommands_0.GetRecipeCount(
        RecipeDefinitionName:= 'Recipes');
END_IF
```

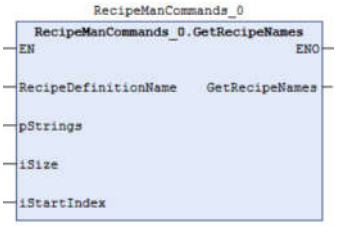
</td></tr>
</table>

## 1-7-7. Get recipe name GetRecipeNames

(1) Instruction overview

Return the recipe name of the corresponding recipe definition.

| Get recipe name [GetRecipeNames] | | | |
|---|---|---|---|
| Execution condition | Edge triggering | Suitable model | XS, XSA |
| Firmware | - | Software | - |
| Library | Recipe Management | | |

| Instruction | Name | Graph | ST |
|---|---|---|---|
| GetRecipe Names | Get recipe name |  | RecipeManCommands_0.GetRecipeNames(<br>RecipeDefinitionName:= ,<br>pStrings:= ,<br>iSize:= ,<br>iStartIndex:= ); |

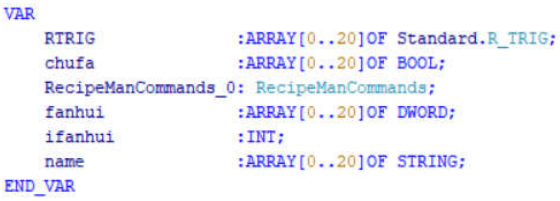
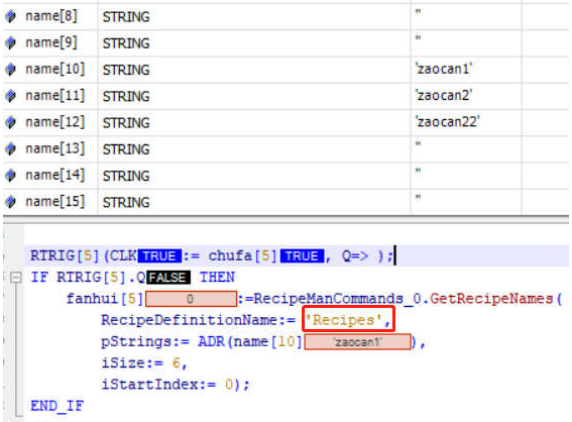
(2) Related variables

| VAR_INPUT | Name | Data type | Valid range | Initial value | Description |
|---|---|---|---|---|---|
| RecipeDefinition Name | Name of recipe definition | STRING | As the data type | - | Name of recipe definition |
| pStrings | String for storing recipe names | POINTER_TO_ARRAY[0..0]OF STRING | As the data type | - | String for storing recipe names |
| iSize | Size of string array | INT | As the data type | 0 | Size of string array |
| iStartIndex | Start index | INT | As the data type | 0 | Specify which recipe to obtain its name from |
| VAR_OUTPUT | Name | Data type | Valid range | Initial value | Description |
| GetRecipeNames | Return value | DWORD | As the data type | - | Return value |

(3) Function

- Return the recipe name of the corresponding recipe definition.

(4) Program example

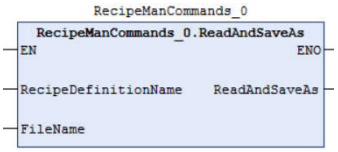
| DEMO | LD | ST |
| --- | --- | --- |
| Add recipe manager, recipe definition, and define variables | |  |
| Program example | | |
| Operation result |  | |

## 1-7-8. Read values and save as recipe file ReadAndSaveAs

(1) Instruction overview

Read the current PLC value and create a new recipe file to store the value in the recipe file without changing any existing recipes.

| Read values and save as recipe file [ReadAndSaveAs] | | | |
|---|---|---|---|
| Execution condition | Edge triggering | Suitable model | XS, XSA |
| Firmware | - | Software | - |
| Library | Recipe Management | | |

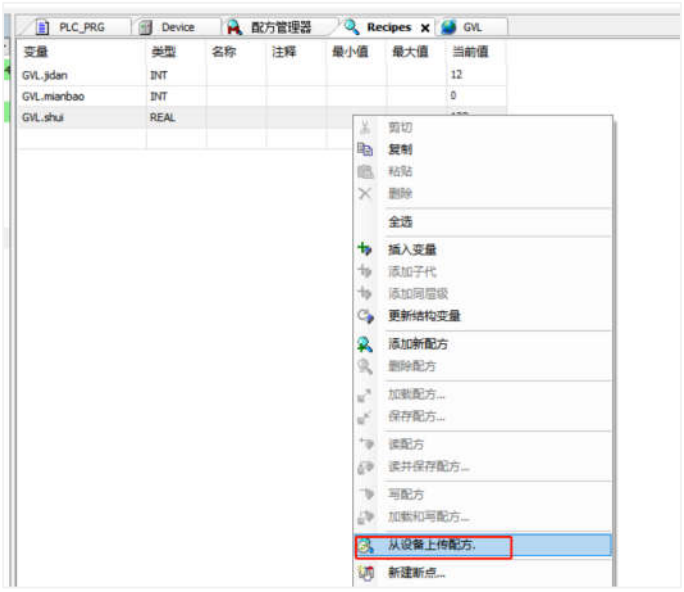
| Instruction | Name | Graph | ST |
|---|---|---|---|
| ReadAndSaveAs | Read values and save as recipe file |  | RecipeManCommands_0.ReadAndSaveAs(<br>RecipeDefinitionName:= ,<br>FileName:= ); |

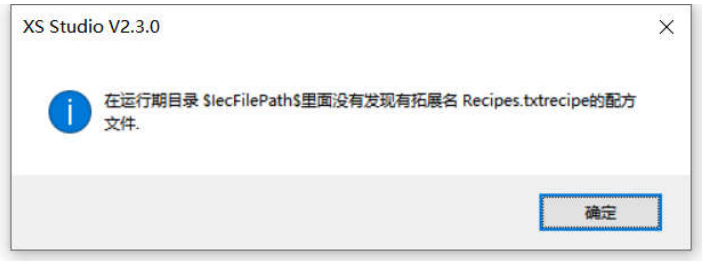
(2) Related variables

| VAR_INPUT | Name | Data type | Valid range | Initial value | Description |
|---|---|---|---|---|---|
| RecipeDefinitionName | Name of recipe definition | STRING | As the data type | - | Name of recipe definition |
| FileName | File name | STRING(255) | As the data type | - | File name |
| VAR_OUTPUT | Name | Data type | Valid range | Initial value | Description |
| ReadAndSaveAs | Return value | DWORD | As the data type | - | Return value |

(3) Function

- After executing ReadAndSaveAs, the current PLC value will be read and a new recipe file will be created to store the value in the recipe file. The recipe file is placed in the "PlcLogic" folder under the PLC root directory in the "Device" - "File" window. The name of the recipe file does not include the name of the recipe definition, so it is not possible to execute "Upload Recipe from Device" on the recipe definition interface.



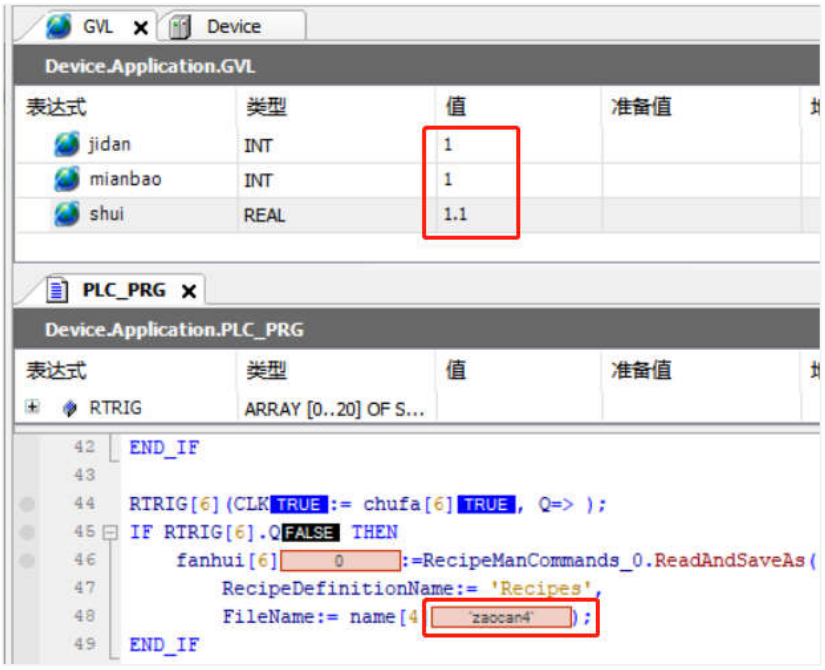




(4) Note

- If the FileName matches the existing recipe file name in the "PlcLogic" folder under the PLC root directory in the "Device" - "File" window, although the instruction execution does not report an error, the variables and other information in the file will not be updated.
- The difference between the ReadAndSaveAs instruction and the CreateRecipe instruction is that the name of the recipe file created by the ReadAndSaveAs instruction does not contain the name of the recipe definition, while the name of the recipe file created by the CreateRecipe instruction contains the name of the recipe definition.

(5) Program example

| DEMO | LD | ST |
|---|---|---|
| Add recipe manager, recipe definition, and define variables | | |

| DEMO | LD | ST |
| --- | --- | --- |
| | <br>VAR<br>RTRIG :ARRAY[0..20]OF Standard.R_TRIG;<br>chufa :ARRAY[0..20]OF BOOL;<br>RecipeManCommands_0: RecipeManCommands;<br>fanhui :ARRAY[0..20]OF DWORD;<br>ifanhui :INT;<br>name :ARRAY[0..20]OF STRING;<br>END_VAR | |
| Program example | | RTRIG[6](CLK:= chufa[6], Q=> );<br>IF RTRIG[6].Q THEN<br>fanhui[6]:=RecipeManCommands_0.ReadAndSaveAs(<br>RecipeDefinitionName:= 'Recipes',<br>FileName:= name[4]);<br>END_IF |
| Operation result |  | |

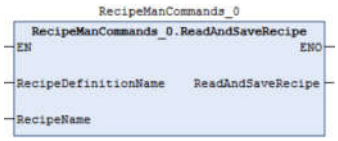
| DEMO | LD | ST |
| --- | --- | --- |
| | Runtime运行时<br>路径<br>PlcLogic<br>名称<br>尺寸<br>修改<br>..<br>ac_persistence<br>alarms<br>Application<br>trend<br>visu<br>_cnc<br>zaocan4.txtrecipe<br>45字节<br>2024/12/8 2<br>zaocan4 - 记事本<br>文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)<br>GVL.jidan:=1<br>GVL.mianbao:=1<br>GVL.shui:=1.1 | |

## 1-7-9. Read values and save in the recipe file ReadAndSaveRecipe

(1) Instruction overview

Read the current PLC value into the recipe and store the recipe in an existing standard recipe file.

| Read values and save in the recipe file [ReadAndSaveRecipe] | | | |
|---|---|---|---|
| Execution condition | Edge triggering | Suitable model | XS, XSA |
| Firmware | - | Software | - |
| Library | Recipe Management | | |

| Instruction | Name | Graph | ST |
|---|---|---|---|
| ReadAndSaveRecipe Count | Read values and save in the recipe file |  | RecipeManCommands_0.ReadAndSaveRecipe(<br>RecipeDefinitionName:= ,<br>RecipeName:= ); |

(2) Related variables

| VAR_INPUT | Name | Data type | Valid range | Initial value | Description |
|---|---|---|---|---|---|
| RecipeDefinitionName | Name of recipe definition | STRING | As the data type | - | Name of recipe definition |
| RecipeName | Recipe name | STRING(255) | As the data type | - | Recipe name |
| VAR_OUTPUT | Name | Data type | Valid range | Initial value | Description |
| ReadAndSaveRecipe | Return value | DWORD | As the data type | - | Return value |

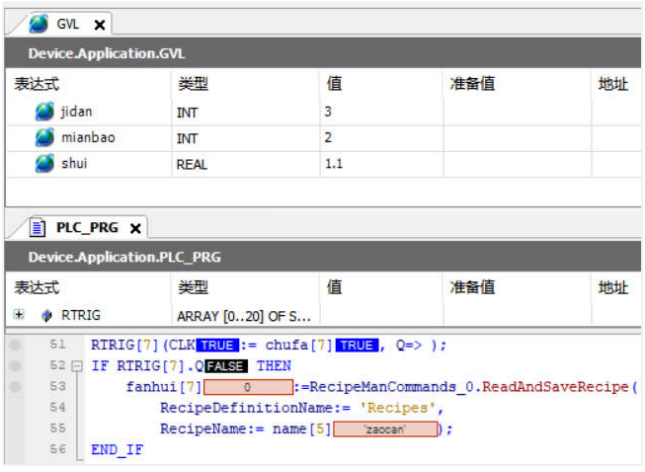
(3) Function

- You need to check " save recipe changes to recipe files atomatically " in the "Recipe Manager" in advance, and then execute ReadAndSaveRecipe to read the current PLC value and update it to the recipe file in the "PlcLogic" folder in the specified root directory.

(4) Note

- Only support updating to standard recipe files with recipe definition names included in the name.

(5) Program example

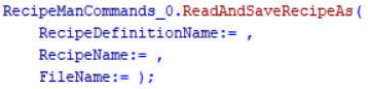
| DEMO | LD | ST |
| --- | --- | --- |
| Add recipe manager, recipe definition, and define variables | <br> <br> | |
| Program example | | |
| Operation result |  | |

## 1-7-10. Read values and save ReadAndSaveRecipeAs

(1) Instruction overview

Read the current PLC value and support creating a new recipe file, or store the value in an existing standard recipe file.

| Read values and save [ReadAndSaveRecipeAs] | | | |
|---|---|---|---|
| Execution condition | Edge triggering | Suitable model | XS, XSA |
| Firmware | - | Software | - |
| Library | Recipe Management | | |

| Instruction | Name | Graph | ST |
|---|---|---|---|
| ReadAndSaveRecipeAs | Read values and save | |  |





(2) Related variables

| VAR_INPUT | Name | Data type | Valid range | Initial value | Description |
|---|---|---|---|---|---|
| RecipeDefinitionName | Name of recipe definition | STRING | As the data type | - | Name of recipe definition |
| RecipeName | Recipe name | STRING | As the data type | - | Name of the recipe created |
| FileName | File name | STRING(255) | | | File name |
| VAR_OUTPUT | Name | Data type | Valid range | Initial | Description |

| | | | | value | |
|---|---|---|---|---|---|
| ReadAndSaveRecipeAs | Return value | DWORD | As the data type | - | Return value |

(3) Note

- RecipeName and FileName pins can be filled in only one or both, corresponding to the functions of ReadAndSaveRecipe and ReadAndSaveAs, respectively. Please refer to the introduction of these two instructions in the manual for details.

(4) Program example

| DEMO | LD | ST |
|---|---|---|
| Add recipe manager, recipe definition, and define variables | | VAR<br>RTRIG :ARRAY[0..20]OF Standard.R_TRIG;<br>chufa :ARRAY[0..20]OF BOOL;<br>RecipeManCommands_0: RecipeManCommands;<br>fanhui :ARRAY[0..20]OF DWORD;<br>ifanhui :INT;<br>name :ARRAY[0..20]OF STRING;<br>END_VAR |
| Program example | | RTRIG[8](CLK:= chufa[8], Q=> );<br>IF RTRIG[8].Q THEN<br>fanhui[8]:=RecipeManCommands_0.ReadAndSaveRecipeAs(<br>RecipeDefinitionName:= 'Recipes',<br>RecipeName:= name[6],<br>FileName:= name[7]);<br>END_IF |
| Operation result | RTRIG[8](CLK TRUE := chufa[8] TRUE, Q=> );<br>IF RTRIG[8].Q FALSE THEN<br>fanhui[8] 0 :=RecipeManCommands_0.ReadAndSaveRecipeAs(<br>RecipeDefinitionName:= 'Recipes',<br>RecipeName:= name[6] 'zaocan20',<br>FileName:= name[7] 'zaocan21');<br>END_IF<br> | |
| Result analysis | RecipeName and FileName pins can be filled in only one, corresponding to the functions of ReadAndSaveRecipe and ReadAndSaveAs, respectively. If both are filled in, the | |

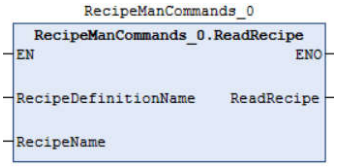
| | |
|---|---|
| | functions of ReadAndSaveRecipe and ReadAndSaveAs will be executed. |

## 1-7-11. Read recipe ReadRecipe

(1) Instruction overview

Read the PLC values into the recipe.

| Read recipe [ReadRecipe] | | | |
|---|---|---|---|
| Execution condition | Edge triggering | Suitable model | XS, XSA |
| Firmware | - | Software | - |
| Library | Recipe Management | | |

| Instruction | Name | Graph | ST |
|---|---|---|---|
| ReadRecipe | Read recipe |  | RecipeManCommands_0.ReadRecipe(<br>RecipeDefinitionName:= ,<br>RecipeName:= ); |

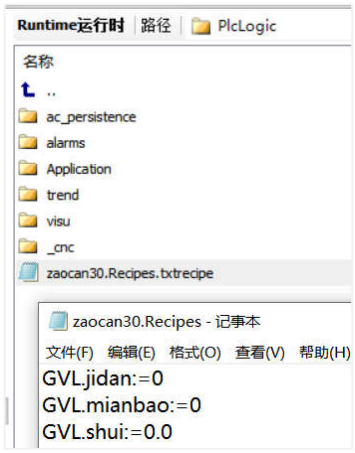
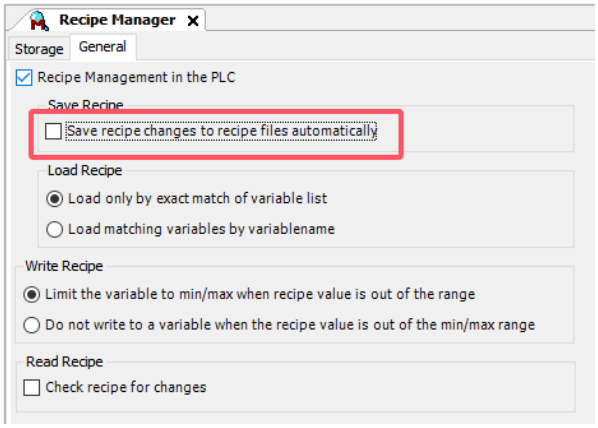
(2) Related variables

| VAR_INPUT | Name | Data type | Valid range | Initial value | Description |
|---|---|---|---|---|---|
| RecipeDefinitionName | Name of recipe definition | STRING | As the data type | - | Name of recipe definition |
| RecipeName | Recipe name | STRING | As the data type | - | Name of the recipe created |
| VAR_OUTPUT | Name | Data type | Valid range | Initial value | Description |
| ReadRecipe | Return value | DWORD | As the data type | - | Return value |

(3) Function

- Read the values of the PLC into the recipe. If " save recipe changes to recipe files atomatically " is checked in the "Recipe Manager", and then ReadRecipe is executed, the values in the recipe file in the "PlcLogic" folder in the PLC root directory will also be updated.

(4) Program example

| DEMO | LD | ST |
| --- | --- | --- |
| Add recipe manager, recipe definition, and define variables | <br>  <br>  <br>VAR<br>RTRIG :ARRAY[0..20]OF Standard.R_TRIG;<br>chufa :ARRAY[0..20]OF BOOL;<br>RecipeManCommands_0: RecipeManCommands;<br>fanhui :ARRAY[0..20]OF DWORD;<br>ifanhui :INT;<br>name :ARRAY[0..20]OF STRING;<br>END_VAR |
| Program example | | RTRIG[9](CLK:= chufa[9], Q=> );<br>IF RTRIG[9].Q THEN<br>fanhui[9]:=RecipeManCommands_0.ReadRecipe(<br>RecipeDefinitionName:= 'Recipes',<br>RecipeName:= name[8]);<br>END_IF |
| Operation result | RTRIG[9](CLK TRUE := chufa[9] TRUE, Q=> );<br>IF RTRIG[9].Q FALSE THEN<br>fanhui[9] 0 :=RecipeManCommands_0.ReadRecipe(<br>RecipeDefinitionName:= 'Recipes',<br>RecipeName:= name[8] 'zaocan30' );<br>END_IF | |

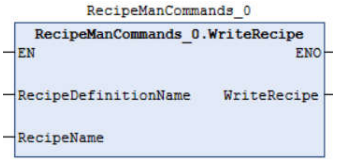
| DEMO | LD | ST |
| --- | --- | --- |
| | | |
| Result analysis | If " save recipe changes to recipe files atomatically " is checked in the "Recipe Manager", and then ReadRecipe is executed, the values in the recipe file in the "PlcLogic" folder in the PLC root directory will also be updated. | |

## 1-7-12. Write recipe WriteRecipe

(1) Instruction overview

Write the recipe value to the PLC variable.

| Write recipe [WriteRecipe] | | | |
|---|---|---|---|
| Execution condition | Edge triggering | Suitable model | XS, XSA |
| Firmware | - | Software | - |
| Library | Recipe Management | | |

| Instruction | Name | Graph | ST |
|---|---|---|---|
| WriteRecipe | Write recipe |  | RecipeManCommands_0.WriteRecipe(<br>RecipeDefinitionName:= ,<br>RecipeName:= ); |

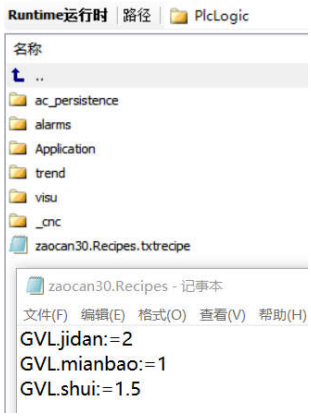
(2) Related variables

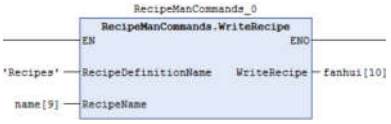
| VAR_INPUT | Name | Data type | Valid range | Initial value | Description |
|---|---|---|---|---|---|
| RecipeDefinitionName | Name of recipe definition | STRING | As the data type | - | Name of recipe definition |
| RecipeName | Recipe name | STRING | As the data type | - | Name of the recipe created |
| VAR_OUTPUT | Name | Data type | Valid range | Initial value | Description |
| WriteRecipe | Return value | DWORD | As the data type | - | Return value |

(3) Function

- Transfer the value of the recipe to the PLC variable.

(4) Program example

<table>
<tr><th>DEMO</th><th>LD</th><th>ST</th></tr>
<tr><td>Add recipe manager, recipe definition, and define variables</td><td colspan="2">


```
VAR
    RTRIG               :ARRAY[0..20]OF Standard.R_TRIG;
    chufa               :ARRAY[0..20]OF BOOL;
    RecipeManCommands_0: RecipeManCommands;
    fanhui              :ARRAY[0..20]OF DWORD;
    ifanhui             :INT;
    name                :ARRAY[0..20]OF STRING;
END_VAR
```
</td></tr>
<tr><td>Program example</td><td>


</td><td>

```
RTRIG[10](CLK:= chufa[10], Q=> );
IF RTRIG[10].Q THEN
    fanhui[10]:=RecipeManCommands_0.WriteRecipe(
        RecipeDefinitionName:= 'Recipes',
        RecipeName:= name[9]);
END_IF
```
</td></tr>
<tr><td>Operation result</td><td colspan="2">

```
RTRIG[10](CLK TRUE := chufa[10] TRUE , Q=> );
IF RTRIG[10].Q FALSE THEN
    fanhui[10]  0  :=RecipeManCommands_0.WriteRecipe(
        RecipeDefinitionName:= 'Recipes',
        RecipeName:= name[9]  'zaocan30'  );
END_IF
```

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

## 1-7-13. Recipe instruction return values

| Name | Return value | Description |
|---|---|---|
| ERR_OK | 16#0 | Operation successful |
| ERR_FAILED | 16#1 | Operation failed |
| ERR_PARAMETER | 16#2 | Parameter error |
| ERR_NOTINITIALIZED | 16#3 | The data server object has not been initialized. If recipe management is combined with CODESYS HMI, a data server is necessary. |
| ERR_NOTIMPLEMENTED | 16#C | If recipe management is used in conjunction with CODESYS HMI, then the IDataServer4 interface is necessary. The data server does not implement the IDataServer4 interface. |
| ERR_NO_OBJECT | 16#10 | Not all recipe definition variables can be written through the data server. Only valid recipe definition variables will be written. |
| ERR_NOMEMORY | 16#11 | The data server did not obtain sufficient memory |
| ERR_RECIPE_FILE_NOT_FOUND | 16#4000 | Recipe file not found |
| ERR_RECIPE_MISMATCH | 16#4001 | The content of the recipe file does not match the current recipe. This error only occurs when the storage type is text type (please refer to "Recipe Manager" and "Storage Type" in your program) and the variable names in the file do not match the variable names in the recipe definition. When this error occurs, the recipe file will not be loaded. Possible reason: The variables defined by the recipe in the program have been deleted. |
| ERR_RECIPE_SAVE_ERR | 16#4002 | The save operation failed.<br>Possible reasons: 1. Due to the disk being full, the file cannot be created or opened; 2. The configured file path does not exist (please refer to "Recipe Manager" and "File Path" in the program); 3. The use of configured file extensions is not allowed during runtime (please refer to "Recipe Manager" and "File Extensions" in the program). |
| ERR_RECIPE_NOT_FOUND | 16#4003 | Recipe does not exist |
| ERR_RECIPE_DEFINITION_NOT_FOUND | 16#4004 | Recipe definition does not exist |
| ERR_RECIPE_ALREADY_EXIST | 16#4005 | The recipe already exists in the formula definition. Create a new recipe with another name. |
| ERR_NO_RECIPE_MANAGER_SET | 16#4006 | The global recipe manager has not been set up. Possible reason: This situation may occur when the option recipe management in the PLC is not set in |

| Name | Return value | Description |
| --- | --- | --- |
| | | the recipe manager of the current application. |
| ERR_RECIPE_NOT_ALL_VARIABLES_WERE_LOADED | 16#4007 | The recipe definition contains more variables than the recipe file. In this case, the variable values in the recipe file will be written no matter what. This is just information, not an error. |
| ERR_RECIPE_NOMEMORY | 16#4008 | The recipe definition does not have available memory to create a new recipe.<br>Possible reasons: 1. This situation may occur when the " save recipe changes to recipe files atomatically " option is not set in the recipe manager of the current application. In this case, each recipe definition can only have 50 recipes. 2. If the " save recipe changes to recipe files atomatically " option is set, no errors will occur. When the disk is full, an error ERR_RECIPE_SAVE_ERR will be created. |
| ERR_RECIPE_MANAGER_LOCKED_DURING_ONLINE_CHANGE | 16#4009 | The recipe manager is locked during online changes.<br>Possible reason: Some recipe management commands should be executed when online changes occur. These commands were not executed during the online change period. |
| ERR_SOURCE_EXHAUSTED | 16#40A0 | Used for UTF8 assistant |
| ERR_TARGET_EXHAUSTED | 16#40A1 | Used for UTF8 assistant |
| ERR_SOURCE_ILLEGAL | 16#40A2 | Used for UTF8 assistant |

# 2. Special instructions

## 2-1. Model and library correspondence table

| Model | Firmware | High speed count library |
|---|---|---|
| XS3-26T4 | / | XinjeCnt |
| XSDH-60A32 | Below 2.4.0 | XinjeCnt |
| | 2.4.0 and above | XJ_HSIO |
| XSLH-24A8, XSLH-24A16 | / | XJ_HSIO |
| XSLH-30A32 | Below 2.4.0 | XinjeCnt |
| | 2.4.0 and above | XJ_HSIO |
| XSF5 series | / | Not support |
| XSA330 | / | XJ_HSIO |
| XSA520, XSA530, XSA550 | / | XJ_HSIO |

## 2-2. XJ_HSIO high speed count library

The reason why the graphical representation and ST representation of some instructions are marked with red wavy lines is that VAR_IN_OUT Counter must be assigned a value when called in order to compile and download the program.

### 2-2-1. Enable the high speed counter【XJ_Counter_Enable】

| High speed counter enable [XJ_Counter_Enable] | | | |
|---|---|---|---|
| Execution condition | Normally ON/OFF triggering | Suitable model | XS series (except XSF) |
| Firmware | XSLH-24, XSA no requirements, other XS models require 2.4.0 or above | Software | XS StudioV2.2.0 and above |
| Library | XJ_HSIO | | |

| Command | Name | Graph | ST language |
|---|---|---|---|
| XJ_Counter_Enable | Enable the high speed counter | XJ_Counter_Enable_0<br>XJ_Counter_Enable<br>EN<br>Counter<br>xEnable<br>eMode<br>xDirection<br>ENO<br>diValue<br>udiFrequency<br>udiRPM<br>xBusy<br>xError<br>eErrorID | XJ_Counter_Enable_0(<br>Counter:= ,<br>xEnable:= ,<br>eMode:= ,<br>xDirection:= ,<br>diValue=> ,<br>udiFrequency=> ,<br>udiRPM=> ,<br>xBusy=> ,<br>xError=> ,<br>eErrorID=> ); |

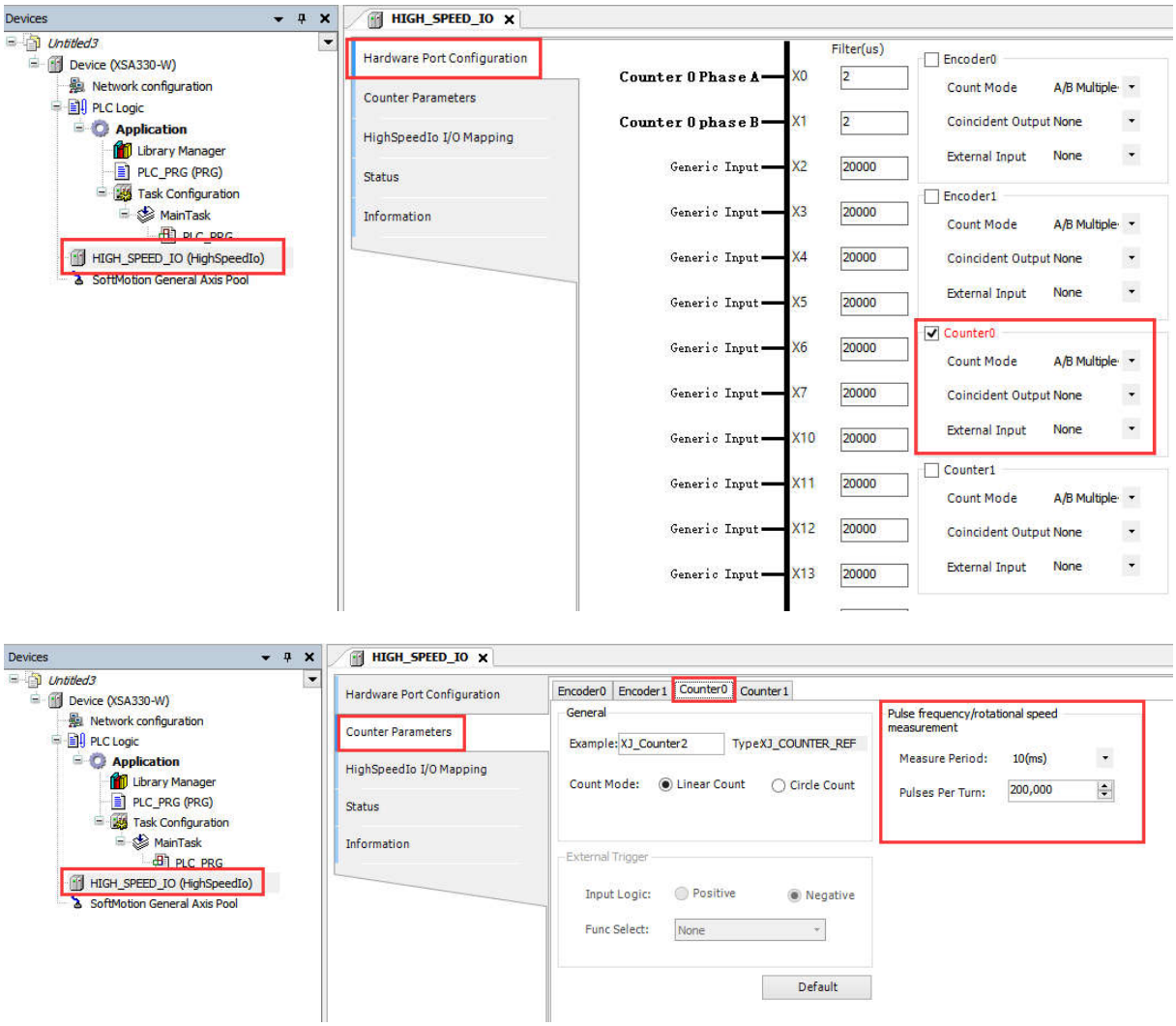
(1) Input variables

| Input variable | Name | Data type | Effective range | Initial value | Description |
|---|---|---|---|---|---|
| Counter | High speed counting input port | XJ_COUNTER_REF | - | - | After configuring the high-speed counter input in the high-speed IO interface, it will be automatically instantiated. Select the corresponding instantiation name of the high-speed counter |
| xEnable | Enable | BOOL | TRUE, FALSE | FALSE | Normally ON the enable to count |
| eMode | Counting mode | HSC_EDGE_MODE | PosEdge, NegEdge, BothEdge | PosEdge | PosEdge: rising edge counting<br>NegEdge: falling edge counting<br>BothEdge: double edge counting |
| xDirection | Direction | BOOL | TRUE, FALSE | FALSE | FALSE: up counting<br>TRUE: down counting |

(2) Output variables

| Output variables | Name | Data type | Effective range | Initial value | Description |
|---|---|---|---|---|---|
| diValue | High speed counting value | DINT | Data range | 0 | High speed counting value |
| udiFrequency | Pulse frequency measurement value | UDINT | | 0 | Unit: Hz, if it is a low frequency, it can be used in conjunction with the measurement period through the interface |
| udiRPM | Rotation speed per minute | UDINT | | 0 | Unit: r/min, used in conjunction with the number of pulses per turn in the interface configuration |
| xBUSY | Busy | BOOL | TRUE,FALSE | FALSE | |
| xError | Error flag | BOOL | TRUE,FALSE | FALSE | |
| eErrorID | Error type | HSIO_ERROR | - | 0 | |

To use high-speed counting, it is necessary to check and configure the corresponding high-speed counter in the hardware parameters, set the counting method and pulse frequency/rotation speed measurement related parameters for the counter parameters.







**Note:**

(1) Please assigning a value to VAR_IN_OUT'Counter' when calling XJ_Counter_Enable, then can the program be compiled and downloaded.

(2) Notes when setting xDirection:
- When single-phase or AB phase counting, counting up/down based on the xDirection terminal status.
- When in P+D mode, the direction terminal FALSE and the xDirection terminal FALSE are counting up. Direction terminal FALSE, xDirection terminal TRUE, represents the subtraction count. The direction terminal is TRUE, while the xDirection terminal is FALSE, indicating a subtractive count. The direction terminal is true, and the xDirection terminal is true, indicating the addition count.

(3) The meaning of measurement cycle is to collect the number of pulses during this period for calculation:
- The measurement cycle is 1000ms, and the minimum measurable frequency is 1Hz. The measurement cycle is 1ms, and the minimum measurable frequency is 1kHz.
- The measurement cycle is 10ms, and the minimum measurable frequency is 100Hz. The measurement cycle is 100ms, and the minimum measurable frequency is 10Hz. The maximum measurement frequency is the frequency that can be received by high-speed counting.

(4) The unit of rotation speed per minute is r/min, and the measured values may fluctuate back and forth due to hardware influence.

(5) The difference between linear counting and circular counting:
- Linear counting: (DownLimitValue, UpLimitValue)
- Circular counting: [iRingDownValue, iRingUpValue]
- Counter 0-3: The linear counting range is (-2147483648, 2147483647), not include -2147483648 and 2147483647.
- Ring counting range is [-2147483648, 2147483647], include -2147483648 and 2147483647.

(6) When Counter0/Counter1 selects AB phase counting, only A-phase frequency is displayed. Set in single phase, AB phase interface.

(7) If the count value reaches near the boundary value, the counter overflow may not report an error due to the influence of the scanning cycle.

(8) When downloading the program, the counter value is not cleared to zero.

(9) Please enable the XJ_Counter_Enable before using XJ_Counter_Sample function block.

## 2-2-2. Compare consistent output【XJ_Counter_Compare】

Set consistent output. When high-speed counting interrupt function is required, XJ_EnableInterrupt needs to be called to turn on the interrupt.

Multi segment comparison and single segment comparison are executed in the triggering order, and the later triggered will not interrupt the first triggered. The execution triggered later will take effect after the execution triggered earlier is completed.

| Compare consistent output [XJ_Counter_Compare] | | | |
|---|---|---|---|
| Execution condition | Edge triggering | Suitable model | XS series (except XSF) |
| Firmware | XSLH-24, XSA no requirements, other XS models require 2.4.0 or above | Software | XS StudioV2.2.0 and above |
| Library | XJ_HSIO | | |

| Command | Name | Graph | ST language |
|---|---|---|---|
| XJ_Counter_Compare | Compare consistent output | XJ_Counter_Compare_0<br>XJ_Counter_Compare<br>EN / ENO<br>Counter / xDone<br>xExcute / xBusy<br>diCompareValue / xError<br>uiImRefreshCycle / eErrorID | XJ_Counter_Compare_0(<br>Counter:= ,<br>xExcute:= ,<br>diCompareValue:= ,<br>uiImRefreshCycle:= ,<br>xDone=> ,<br>xBusy=> ,<br>xError=> ,<br>eErrorID=> ); |

(1) Input variable

| Input variables | Name | Data type | Effective range | Initial value | Description |
|---|---|---|---|---|---|
| Counter | High speed counter input port | XJ_COUNTER_REF | - | - | High speed counter, it is necessary to define the high-speed counting input terminal (see functional description) |
| xExecute | Trigger | BOOL | TRUE,FALSE | FALSE | Trigger |
| diCompareValue | appoint the compare value | DINT | | 0 | |
| uiImRefreshCycle | Hardware direct output | UINT | | 0 | Unit: 100us, maximum output time is 3000ms. |

| | | | | | |
|---|---|---|---|---|---|
| | time | | | | |

(2) Output variables

| Output variables | Name | Data type | Effective range | Initial value | Description |
|---|---|---|---|---|---|
| xDone | Completed flag | BOOL | TRUE,FALSE | FALSE | After the instruction execution is completed, the flag is TRUE |
| xBusy | Running | BOOL | TRUE,FALSE | FALSE | |
| xError | Error flag | BOOL | TRUE,FALSE | FALSE | |
| eErr1orID | Error type | HSIO_ERROR | - | 0 | |

**Note:**

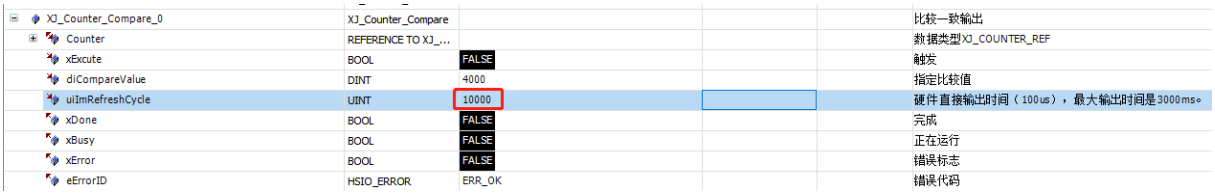
(1) The control of Y0-Y7 can be configured in the background, and each counter can freely select Y0-Y7, with a certain delay in output. The output time is 100us-3000ms. If ImRefreshCycle is 0, call XJ_Reset CmpOutput to lower the output.

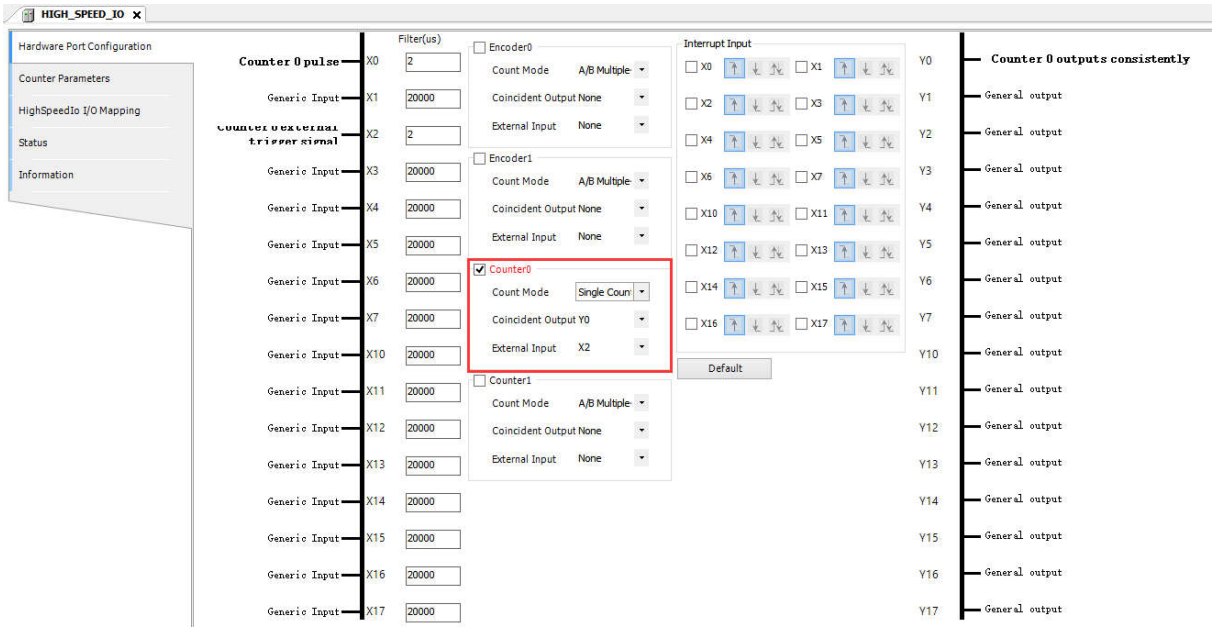
(2) ImRefresh=1: Hardware immediately outputs, Y0-Y7 cannot be freely selected, there is no delay, and the output time is set through ImRefreshCycle. The output time is 0-3000ms.

(3) XJ_EnableInterrupt needs to be called in advance to open compare consistent interrupt.

(4) After hot reset and cold reset, maintain the previous comparison value.

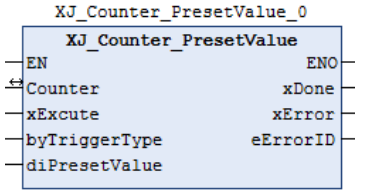
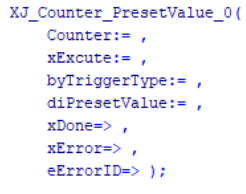
Set compare consistent output and call XJ_EnableInterrupt to open the interrupt.
If hardware output is not set, it is necessary to turn on compare consistent interrupt. If hardware output is set, compare consistent interrupt may not be turned on.
For example, if the output time is set to 1 second, the hardware output of Y will be 1 second after comparing the values, and the software status of Y will remain unchanged, it will not display any output. Execute interrupt program after external trigger input.



## 2-2-3. Preset value write in【XJ_Counter_PresetValue】

| Preset value write in [XJ_Counter_PresetValue] | | | |
|---|---|---|---|
| Execution condition | Edge triggering | Suitable model | XS series (except XSF) |
| Firmware | XSLH-24, XSA no requirements, other XS models require 2.4.0 or above | Software | XS StudioV2.2.0 and above |
| Library | XJ_HSIO | | |

| Command | Name | Graph | ST language |
|---|---|---|---|
| XJ_Counter_PresetValue | Preset value write in |  |  |





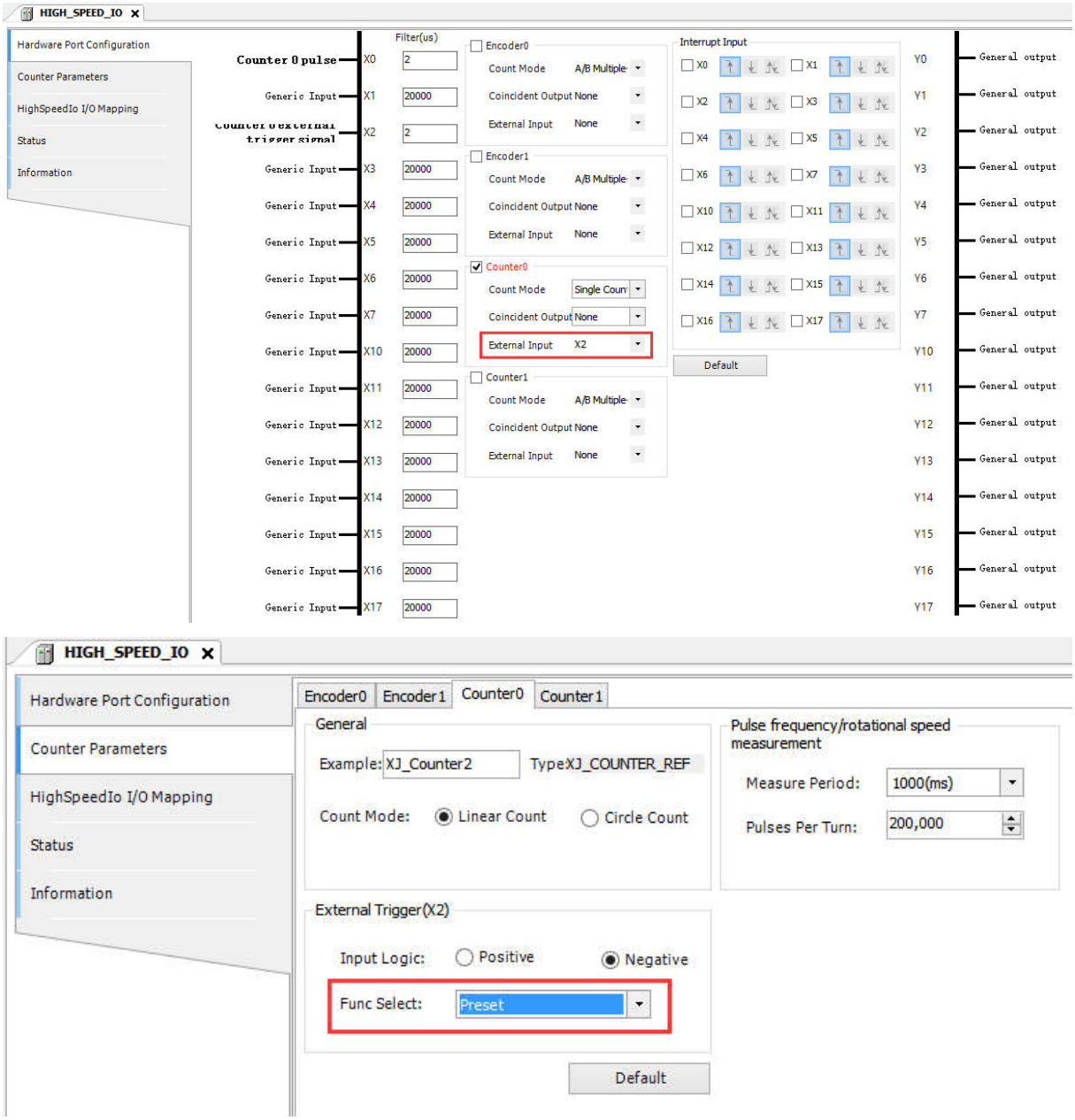
(1) Input variables

| Input variables | Name | Data type | Effective range | Initial value | Description |
|---|---|---|---|---|---|
| Counter | High speed counter input port | XJ_COUNTER_REF | - | - | High speed counter, it is necessary to define the high-speed counting input terminal (see functional description) |
| xExecute | Trigger | BOOL | TRUE,FALSE | FALSE | Trigger |
| byTrigger Type | Trigger type | HSC_PSV_TIGGLE | | 0 | 0: Rising edge triggered write;<br>1: External input triggering;<br>2: When comparing consistent outputs, it is preset and triggered at the rising edge. |
| diPresetVa lue | Preset value | DINT | Data range | 0 | Write high-speed count preset value |

(2) Output variables

| Output variables | Name | Data type | Effective range | Initial value | Description |
|---|---|---|---|---|---|
| xDone | Completed flag | BOOL | TRUE,FALSE | FALSE | After completing the write, the flag is TRUE |
| xError | Error flag | BOOL | TRUE,FALSE | FALSE | |
| eErrorID | Error type | HSIO_ERROR | - | 0 | |

Note: If the value displayed for ErrorID is 2, it is because the CounterID range is not between 0 and 3.
0: Rising edge triggered writing. After the instruction is triggered, the preset value can be written into the high-speed count.
1: External input triggering. After the command is triggered, wait for the set external input terminal signal, and if there is a signal, the preset value can be written in.







2: When comparing consistent outputs, it is preset and triggered at the rising edge.

## 2-2-4. Probe【XJ_TouchProbe】

| Probe [XJ_TouchProbe] | | | |
|---|---|---|---|
| Execution condition | Edge triggering | Suitable model | XS series (except XSF) |
| Firmware | XSLH-24, XSA no requirements, other XS models require 2.4.0 or above | Software | XS StudioV2.2.0 and above |
| Library | XJ_HSIO | | |

| Command | Name | Graph | ST language |
|---|---|---|---|
| XJ_TouchProbe | Probe | XJ_TouchProbe_0<br>XJ_TouchProbe<br>EN, Counter, xExcute<br>ENO, xDone, xBusy, diTouchValue, xError, eErrorID | XJ_TouchProbe_0(<br>Counter:= ,<br>xExcute:= ,<br>xDone=> ,<br>xBusy=> ,<br>diTouchValue=> ,<br>xError=> ,<br>eErrorID=> ); |

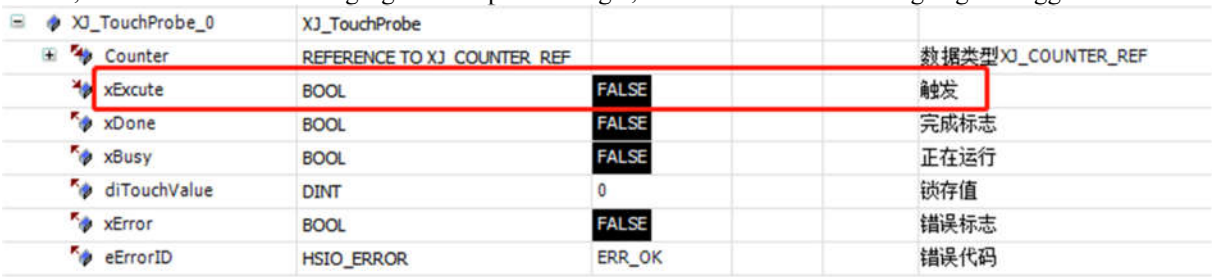
(1) Input variables

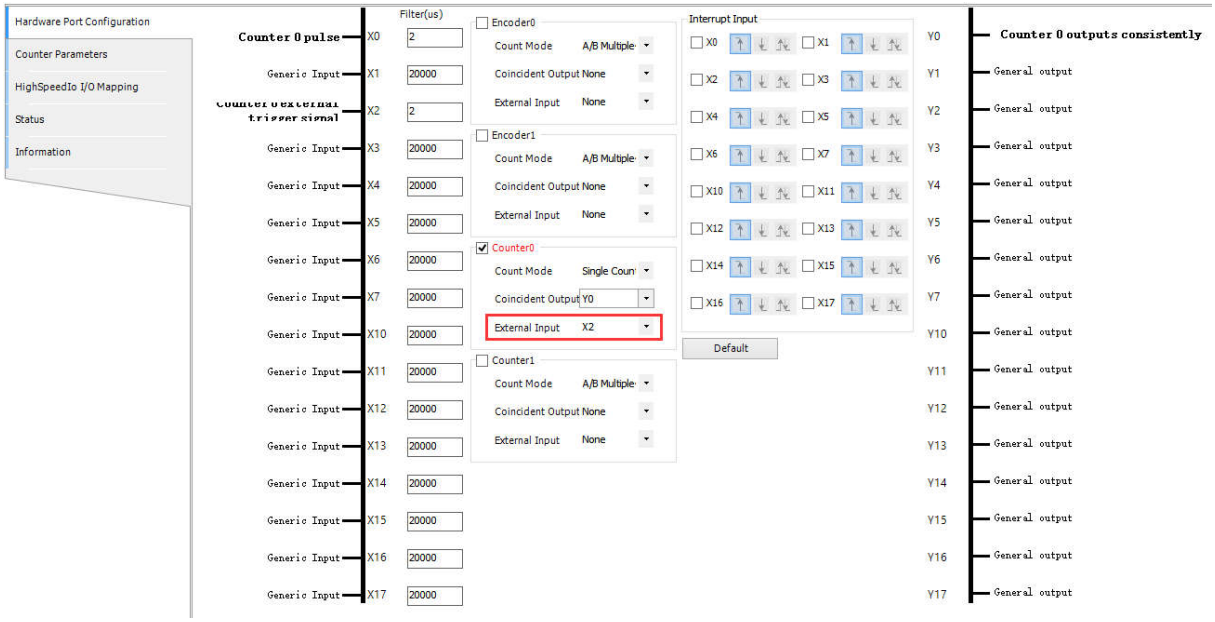
| Input variables | Name | Data type | Effective range | Initial value | Description |
|---|---|---|---|---|---|
| Counter | High speed counter input port | XJ_COUNTER_REF | - | - | High speed counter, it is necessary to define the high-speed counting input terminal (see functional description) |
| xExcute | Trigger | BOOL | TRUE,FALSE | FALSE | Trigger |

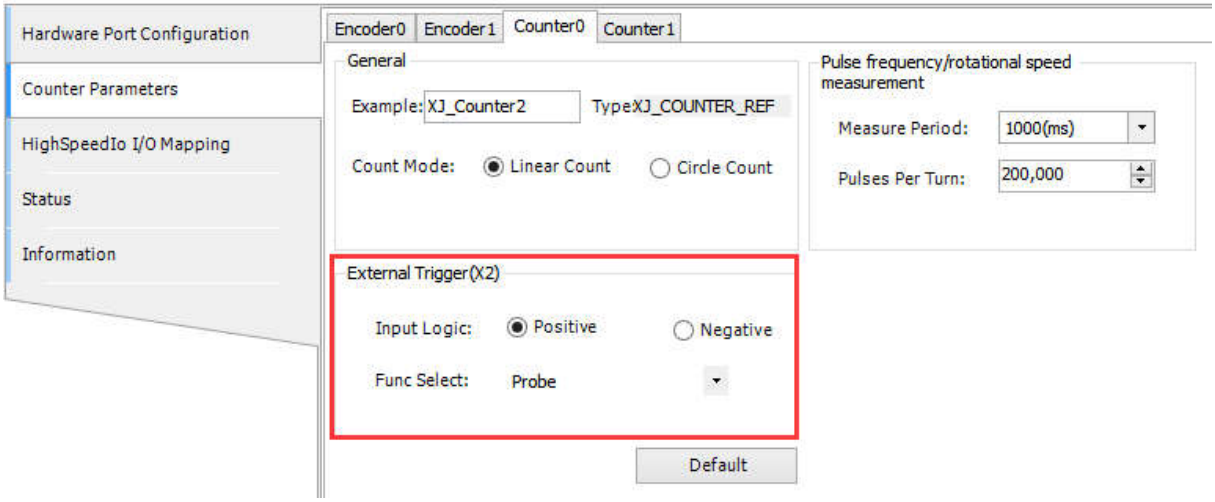
(2) Output variables

| Output variables | Name | Data type | Effective range | Initial value | Description |
|---|---|---|---|---|---|
| xDone | Completed flag | BOOL | TRUE,FALSE | FALSE | After completing the write, the flag is TRUE |
| xBusy | Running | BOOL | TRUE,FALSE | FALSE | |
| diTouchVaule | Latch value | DINT | | 0 | |
| xError | Error flag | BOOL | TRUE,FALSE | FALSE | |
| eErrorID | Error type | HSIO_ERROR | - | 0 | |

Set the terminal for the probe function. After triggering this terminal, the current high-speed count value can be locked, as shown in the following figure. The positive logic, takes effect after the rising edge is triggered.




Hardware Port Configuration
Counter Parameters
HighSpeedIo I/O Mapping
Status
Information
Filter(us)
Counter 0 pulse
Generic Input
Counter 0 external trigger signal
Encoder0
Encoder1
Counter0
Counter1
Count Mode
Coincident Output
External Input
None
Interrupt Input
Default
Counter 0 outputs consistently
General output


Hardware Port Configuration
Counter Parameters
HighSpeedIo I/O Mapping
Status
Information
Encoder0
Encoder1
Counter0
Counter1
General
Example: XJ_Counter2
TypeXJ_COUNTER_REF
Count Mode:
Linear Count
Circle Count
Pulse frequency/rotational speed measurement
Measure Period:
1000(ms)
Pulses Per Turn:
200,000
External Trigger(X2)
Input Logic:
Positive
Negative
Func Select:
Probe
Default

## 2-2-5. Read the pulse width measurement value of the counter【XJ_MeasurePulseWidth】

| Read the pulse width measurement value of the counter [XJ_MeasurePulseWidth] | | | |
|---|---|---|---|
| Execution condition | Normally ON/OFF triggering | Suitable model | XS series (except XSF) |
| Firmware | XSLH-24, XSA no requirements, other XS models require 2.4.0 or above | Software | XS StudioV2.2.0 and above |
| Library | XJ_HSIO | | |

| Command | Name | Graph | ST language |
|---|---|---|---|
| XJ_MeasurePulseWidth | Read the pulse width measurement value of the counter | XJ_MeasurePulseWidth_0<br>XJ_MeasurePulseWidth<br>EN / ENO<br>Counter / udiValue<br>xEnable / xBusy<br>eMode / xDone<br>xError<br>eErrorID | XJ_MeasurePulseWidth_0(<br>Counter:= ,<br>xEnable:= ,<br>eMode:= ,<br>udiValue=> ,<br>xBusy=> ,<br>xDone=> ,<br>xError=> ,<br>eErrorID=> ); |

(1) Input variables

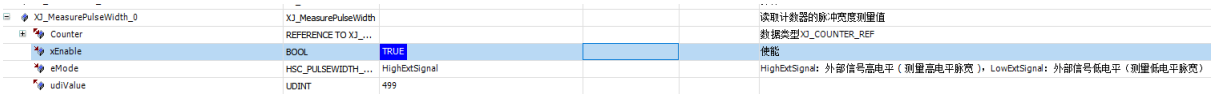
| Input variables | Name | Data type | Effective range | Initial value | Description |
|---|---|---|---|---|---|
| Counter | High speed counter input port | XJ_COUNTER_REF | - | - | High speed counter, it is necessary to define the high-speed counting input terminal (see functional description) |
| xExecute | Trigger | BOOL | TRUE,FALSE | FALSE | Trigger |
| eMode | Measure high/low level pulse width | HSC_PULSEWIDTH_TYPE | 0,1 | 0 | 0: External signal high level (measuring high level pulse width)<br>1: External signal low-level (measuring low-level pulse width) |

(2) Output variables

| Output variables | Name | Data type | Effective range | Initial value | Description |
|---|---|---|---|---|---|
| udiValue | Measuring value | UDINT | | 0 | |
| xDone | Completed flag | BOOL | TRUE,FALSE | FALSE | After completing the write, the flag is TRUE |
| xBusy | Running | BOOL | TRUE,FALSE | FALSE | |
| xError | Error flag | BOOL | TRUE,FALSE | FALSE | |
| eErrorID | Error type | HSIO_ERROR | - | 0 | |

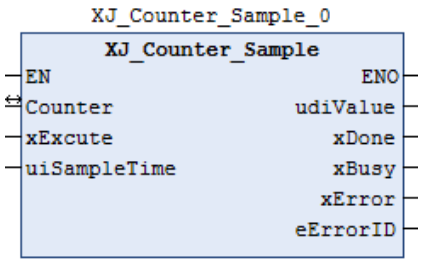
Note: Measure the duration of high and low levels in us microseconds.

Example: If the frequency is 1kHz, it is equivalent to sending a pulse every 1ms, and if there is a high or low level every 0.5ms, the measured high or low level is about 500us.



## 2-2-6. Counter sample【XJ_Counter_Sample】

| Counter sample [XJ_Counter_Sample] | | | |
|---|---|---|---|
| Execution condition | Edge triggering | Suitable model | XS series (except XSF) |
| Firmware | XSLH-24, XSA no requirements, other XS models require 2.4.0 or above | Software | XS StudioV2.2.0 and above |
| Library | XJ_HSIO | | |

| Command | Name | Graph | ST language |
|---|---|---|---|
| XJ_Counter_Sample | Counter sample |  | XJ_Counter_Sample_0(<br>Counter:= ,<br>xExcute:= ,<br>uiSampleTime:= ,<br>udiValue=> ,<br>xDone=> ,<br>xBusy=> ,<br>xError=> ,<br>eErrorID=> ); |

(1) Input variables

| Input variables | Name | Data type | Effective range | Initial value | Description |
|---|---|---|---|---|---|
| Counter | High speed counter input port | XJ_COUNTER_REF | - | - | High speed counter, it is necessary to define the high-speed counting input terminal (see functional description) |
| eExecute | Trigger | BOOL | TRUE,FALSE | FALSE | Trigger |
| uiSampleTime | Sampl time | UINT | | 0 | Sample time (10ms~65535ms) |

(2) Output variables

| Output variables | Name | Data type | Effective range | Initial value | Description |
|---|---|---|---|---|---|
| udiValue | Sample value | UDINT | | 0 | |
| xDone | Completed flag | BOOL | TRUE,FALSE | FALSE | After completing the write, the flag is TRUE |
| xBusy | Running | BOOL | TRUE,FALSE | FALSE | |
| xError | Error flag | BOOL | TRUE,FALSE | FALSE | |
| eErrorID | Error type | HSIO_ERROR | - | 0 | |

**Note:**

(1) XJ_Counter_Sample can run after XJ_Counter_Enable is enabled.

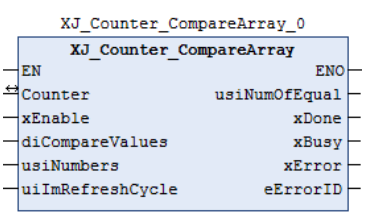
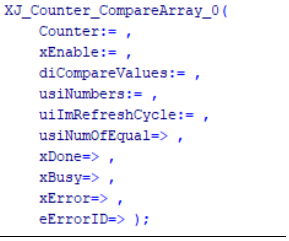
(2) After the command is triggered, the sampled signal count value will not change in real time, but will be displayed after the command is completed, displaying the high-speed count value collected within the set time.

## 2-2-7. Multiple segments compare【XJ_Counter_CompareArray】

Compare Consistent Output: Up to 100 comparison values can be set, and the Done signal is the output.

Multi segment comparison and single segment comparison are executed in the triggering order, and the later triggered will not interrupt the first triggered. The execution triggered later will take effect after the execution triggered earlier is completed.

| Multiple segments compare [XJ_Counter_CompareArray] | | | |
|---|---|---|---|
| Execution condition | Normally ON/OFF triggering | Suitable model | XS series (except XSF) |
| Firmware | XSLH-24, XSA no requirements, other XS models require 2.4.0 or above | Software | XS StudioV2.2.0 and above |
| Library | XJ_HSIO | | |

| Command | Name | Graph | ST language |
|---|---|---|---|
| XJ_Counter_CompareArray | Multiple segments compare |  |  |





(1) Input variables

| Input variables | Name | Data type | Effective range | Initial value | Description |
|---|---|---|---|---|---|
| Counter | High speed counter input port | XJ_COUNTER_REF | - | - | High speed counter, it is necessary to define the high-speed counting input terminal (see functional description) |
| xEnable | Enable | BOOL | TRUE,FALSE | FALSE | Enable |
| diCompareValues | Comparison value one-dimensional array | ARRAY | | 0 | A one-dimensional array of comparison values for the set counters, supporting a maximum of 100 |
| usiNumbers | set actual value | USINT | | 0 | The actual value set, allowed to be set to 1, with a maximum of 100 |
| uiImRefreshcycle | hardware direct output time | UINT | | 0 | The unit is 100us, and the maximum output time is 3000ms. For example, setting 10000 means 1000ms |

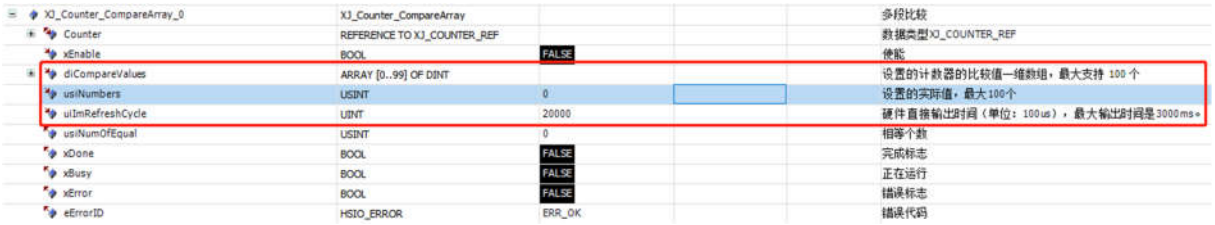
(2) Output variables

| Output variables | Name | Data type | Effective range | Initial value | Description |
|---|---|---|---|---|---|
| usiNumOfEqual | Equal numbers | USINT | | 0 | |
| xDone | Completed flag | BOOL | TRUE,FALSE | FALSE | After completing the write, the flag is TRUE |
| xBusy | Running | BOOL | TRUE,FALSE | FALSE | |
| xError | Error flag | BOOL | TRUE,FALSE | FALSE | |
| eErrorID | Error type | HSIO_ERROR | - | 0 | |

- Multiple comparison values can be set, and when the comparison value is reached, it will enter the task execution instruction of external events, with hardware output.
- Difference from single segment comparison: It is necessary to open a compare consistent interrupt,

otherwise the instruction will report an error.

- As shown in the figure below, set the corresponding parameters as needed:



| | | | | |
|---|---|---|---|---|
| XJ_Counter_CompareArray_0 | XJ_Counter_CompareArray | | | 多段比较 |
| Counter | REFERENCE TO XJ_COUNTER_REF | | | 数据类型XJ_COUNTER_REF |
| xEnable | BOOL | FALSE | | 使能 |
| diCompareValues | ARRAY [0..99] OF DINT | | | 设置的计数器的比较值一维数组，最大支持 100 个 |
| usiNumbers | USINT | 0 | | 设置的实际值，最大100个 |
| uiImRefreshCycle | UINT | 20000 | | 硬件直接输出时间（单位：100us），最大输出时间是3000ms。 |
| usiNumOfEqual | USINT | 0 | | 相等个数 |
| xDone | BOOL | FALSE | | 完成标志 |
| xBusy | BOOL | FALSE | | 正在运行 |
| xError | BOOL | FALSE | | 错误标志 |
| eErrorID | HSIO_ERROR | ERR_OK | | 错误代码 |

## 2-2-8. Ring counting【XJ_Counter_SetRing】

| Ring counting [XJ_Counter_SetRing] | | | |
|---|---|---|---|
| Execution condition | Edge triggering | Suitable model | XS series (except XSF) |
| Firmware | XSLH-24, XSA no requirements, other XS models require 2.4.0 or above | Software | XS StudioV2.2.0 and above |
| Library | XJ_HSIO | | |

| Command | Name | Graph | ST language |
|---|---|---|---|
| XJ_Counter_SetRing | Set ring counter | XJ_Counter_SetRing_0<br>XJ_Counter_SetRing<br>EN / ENO<br>Counter / xDone<br>xExcute / xBusy<br>diMaxValue / xError<br>diMinValue / eErrorID | XJ_Counter_SetRing_0(<br>Counter:= ,<br>xExcute:= ,<br>diMaxValue:= ,<br>diMinValue:= ,<br>xDone=> ,<br>xBusy=> ,<br>xError=> ,<br>eErrorID=> ); |

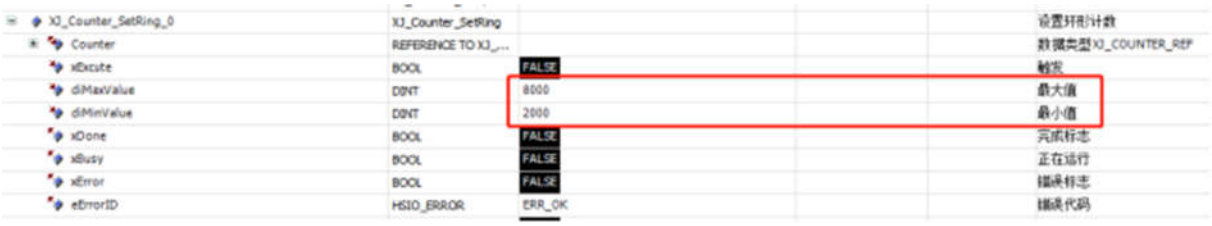
(1) Input variables

| Input variables | Name | Data type | Effective range | Initial value | Description |
|---|---|---|---|---|---|
| Counter | High speed counter input port | XJ_COUNTER_REF | - | - | High speed counter, it is necessary to define the high-speed counting input terminal (see functional description) |
| xExcute | Trigger | BOOL | TRUE,FALSE | FALSE | Trigger |
| diMaxValue | Maximum value | DINT | | 0 | Maximum value |
| diMinValue | Minimum value | DINT | | 0 | Minimum value |

(2) Output variables

| Output variables | Name | Data type | Effective range | Initial value | Description |
|---|---|---|---|---|---|
| xDone | Completed flag | BOOL | TRUE,FALSE | FALSE | After completing the write, the flag is TRUE |
| xBusy | Running | BOOL | TRUE,FALSE | FALSE | |
| xError | Error flag | BOOL | TRUE,FALSE | FALSE | |
| eErrorID | Error type | HSIO_ERROR | - | 0 | |

- The current count value needs to be within the set value range, otherwise the command will report an error. After the instruction is executed, no instruction can switch back to linear counting. If switching back to linear technology is required, the counting range needs to be changed to a linear counting range.

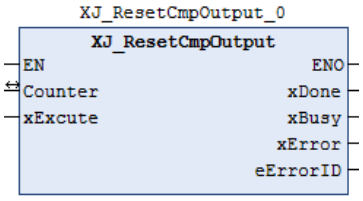



- The minimum value for linear counting is -2147483647, and the maximum value is 2147483646. For example, adding one signal input after increasing the count to 2147483646 will change to -2147483647, and then continue counting upwards. It will not reach the minimum value of double word signed numbers -2147483648 and the maximum value of 2147483647.

- There are two configuration methods for ring counting: firstly, when not logged in, in the "High speed IO" interface - "Counter parameter settings", you can click "Ring counting" and set the range. The second is to use the XJ_Counter_SetRing instruction setting, which can directly switch linear counting to circular counting during login.

## 2-2-9. Reset port of compare consistent output【XJ_ResetCmpOutput】

| Reset port of compare consistent output [XJ_ResetComOutput] | | | |
|---|---|---|---|
| Execution condition | Edge triggering | Suitable model | XS series (except XSF) |
| Firmware | XSLH-24, XSA no requirements, other XS models require 2.4.0 or above | Software | XS StudioV2.2.0 and above |
| Library | XJ_HSIO | | |

| Command | Name | Graph | ST language |
|---|---|---|---|
| XJ_ResetCmpOutput | Reset the compare consistent output port |  | XJ_ResetCmpOutput_0(<br>Counter:= ,<br>xExcute:= ,<br>xDone=> ,<br>xBusy=> ,<br>xError=> ,<br>eErrorID=> ); |

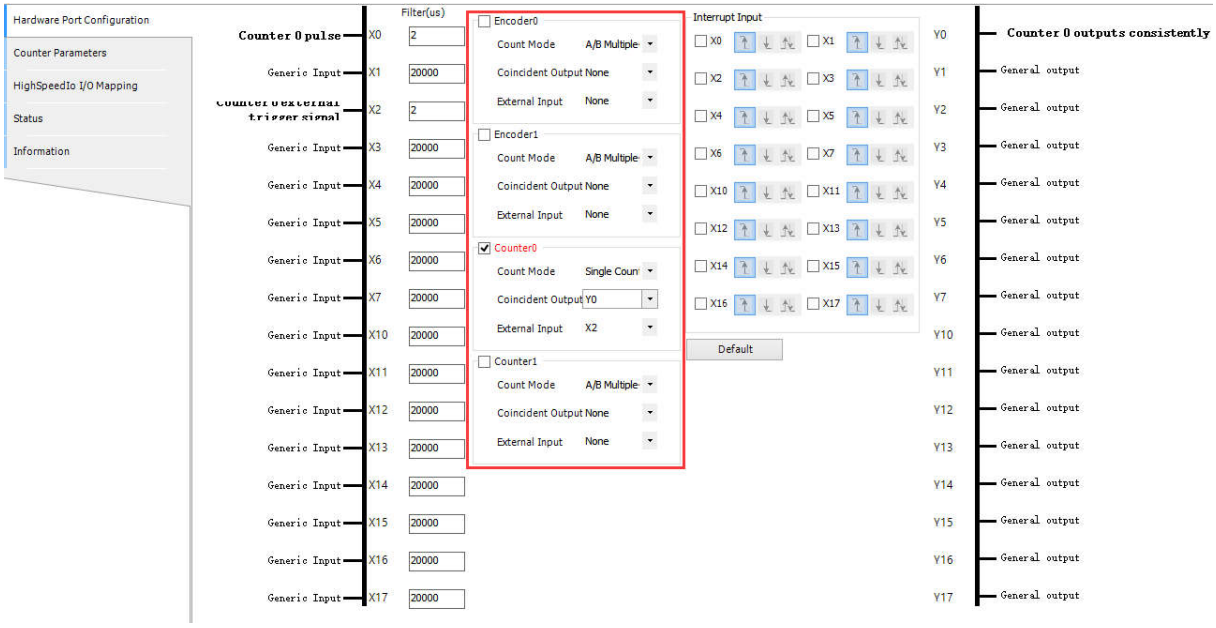
(1) Input variables

| Input variables | Name | Data type | Effective range | Initial value | Description |
|---|---|---|---|---|---|
| Counter | High speed counter input port | XJ_COUNTER_REF | - | - | High speed counter, it is necessary to define the high-speed counting input terminal (see functional description) |
| xExcute | Trigger | BOOL | TRUE,FALSE | FALSE | Trigger |

(2) Output variables

| Output variables | Name | Data type | Effective range | Initial value | Description |
|---|---|---|---|---|---|
| xDone | Completed flag | BOOL | TRUE,FALSE | FALSE | After completing the setting, the flag is TRUE |
| xBusy | Running | BOOL | TRUE,FALSE | FALSE | |
| xError | Error flag | BOOL | TRUE,FALSE | FALSE | |
| eErrorID | Error type | HSIO_ERROR | - | 0 | |

Release all compare consistent output ports. At this time, Y, which was originally set as a compare consistent output, can be used as a normal Y. As for whether the Y-point is used as a compare consistent output or a regular output, it depends on which port instruction is triggered first for the compare consistent output command and reset the compare consistent output command.




## 2-2-10. Clear the error【XJ_Counter_Reset】

Clear errors related to high-speed counting instructions.

| Clear the error [XJ_Counter_Reset] | | | |
|---|---|---|---|
| Execution condition | Edge triggering | Suitable model | XS series (except XSF) |
| Firmware | XSLH-24, XSA no requirements, other XS models require 2.4.0 or above | Software | XS StudioV2.2.0 and above |
| Library | XJ_HSIO | | |

| Command | Name | Graph | ST language |
|---|---|---|---|
| XJ_Counter_Reset | Clear the error | XJ_Counter_Reset_0<br>XJ_Counter_Reset<br>EN, Counter, xExcute<br>ENO, xDone, xBusy, xError, eErrorID | XJ_Counter_Reset_0(<br>Counter:= ,<br>xExcute:= ,<br>xDone=> ,<br>xBusy=> ,<br>xError=> ,<br>eErrorID=> ); |

(1) Input variables

| Input variables | Name | Data type | Effective range | Initial value | Description |
|---|---|---|---|---|---|
| Counter | High speed counter input port | XJ_COUNTER_REF | - | - | High speed counter, it is necessary to define the high-speed counting input terminal (see functional description) |
| xExcute | Trigger | BOOL | TRUE,FALSE | FALSE | Trigger |

(2) Output variables

| Output variables | Name | Data type | Effective range | Initial value | Description |
|---|---|---|---|---|---|

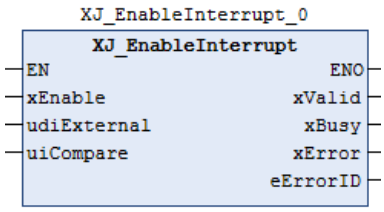
| | | | | | |
|---|---|---|---|---|---|
| xDone | Completed flag | BOOL | TRUE,FALSE | FALSE | After completing the setting, the flag is TRUE |
| xBusy | Running | BOOL | TRUE,FALSE | FALSE | |
| xError | Error flag | BOOL | TRUE,FALSE | FALSE | |
| eErrorID | Error type | HSIO_ERROR | - | 0 | |

## 2-2-11. Enable external interrupt and compare consistent interrupt XJ_EnableInterrupt

### (1) Overview

Interrupts are used in Codesys through external event forms in task types. For example, X2R_TRIG represents the rising edge interrupt of X2, and X2F_TRIG represents the falling edge interrupt. The number and types of interrupts supported by each model can be found in the "External event" option.

| Enable external interrupt and compare consistent interrupt [XJ_EnableInterrupt] | | | |
|---|---|---|---|
| Execution condition | Edge triggering | Suitable model | XS series (except XSF) |
| Firmware | XSLH-24, XSA no requirements, other XS models require 2.4.0 or above | Software | XS StudioV2.2.0 and above |
| Library | XJ_HSIO | | |

| Command | Name | Graph | ST language |
|---|---|---|---|
| XJ_EnableInterrupt | Enable external interrupt and compare consistent interrupt |  | XJ_EnableInterrupt(<br>xEnable:= ,<br>udiExternal:= ,<br>uiCompare:= ,<br>xValid=> ,<br>xBusy=> ,<br>xError=> ,<br>eErrorID=> ); |

### (2) Input variables

| Input variables | Name | Data type | Effective range | Initial value | Description |
|---|---|---|---|---|---|
| xEnable | Enable | BOOL | TRUE,FALSE | FALSE | Enable |
| udiExternal | Enable external input interrupt | UINT | | 0 | |
| uiCompare | Enable consistent interrupt | UINT | | 0 | For example, 4 means CounterID:=2 |

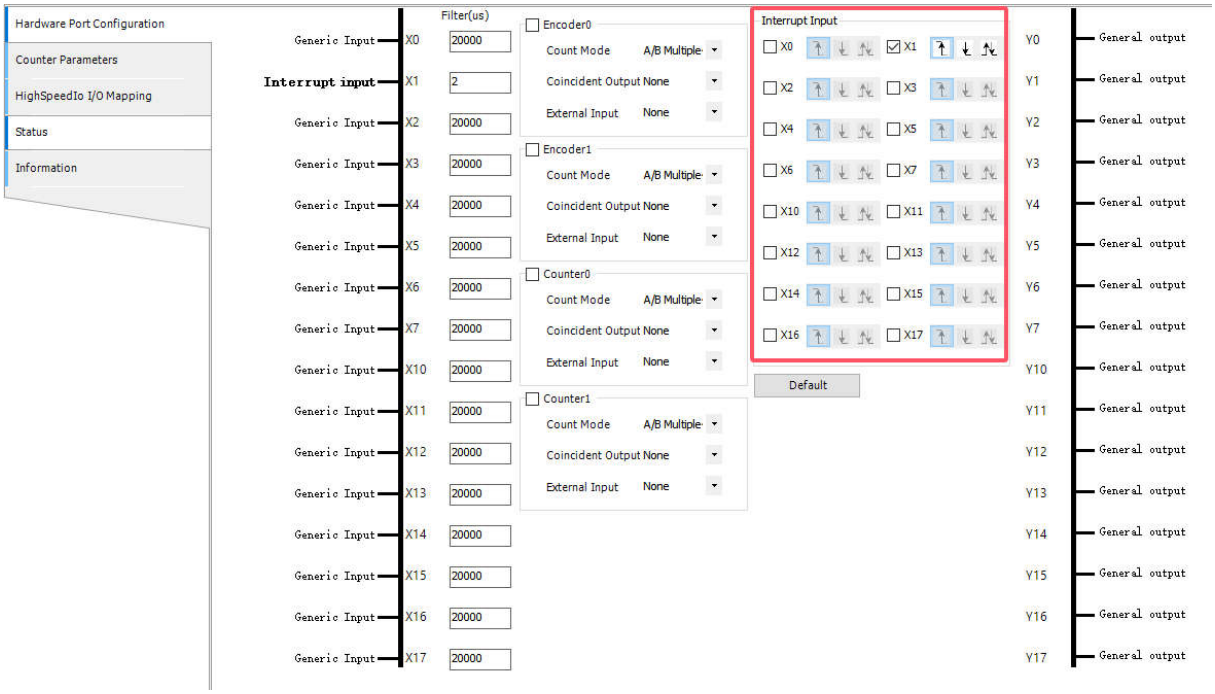
### (3) Output variables

| Output variables | Name | Data type | Effective range | Initial value | Description |
|---|---|---|---|---|---|
| xValid | Interruption effective | BOOL | TRUE,FALSE | FALSE | After the interrupt takes effect, the flag is set to TRUE |
| xBusy | Running | BOOL | TRUE,FALSE | FALSE | |
| xError | Error flag | BOOL | TRUE,FALSE | FALSE | |
| eErrorID | Error type | HSIO_ERROR | - | 0 | |

(4) Function

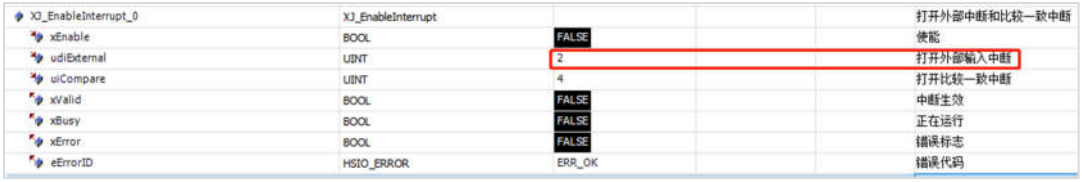
- **Enable external input interrupt**

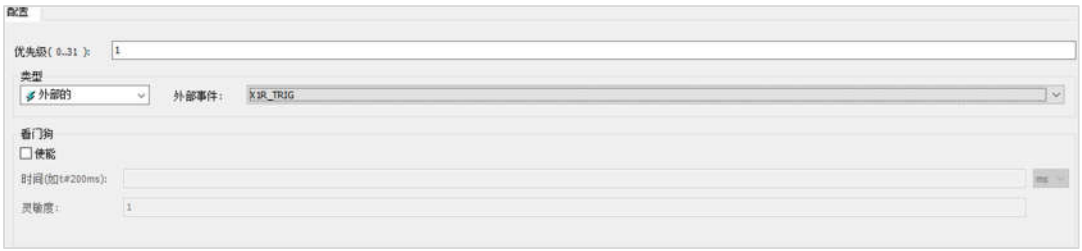
① select the external interrupt input terminal, such as X1.




② Enable XJ_EnableInterrupt to add interrupt tasks, where 1 represents X0, 2 represents X1, 3 represents X0, X1, and so on. As shown in the figure below, execute an external interrupt task when the rising edge of X1 is valid.

That is, the udiExternal enables the external interrupt to display the binary address and then gives the value. For example, if binary 2#11 or decimal 10#3 is filled in, then X0 and X1 are enabled. In the "Hardware Parameter Configuration", the required X0 and X1 should also be checked, and can be selected as either edge or double edge. When there is a signal from X0 or X1, the task configured as external "X0" or "X1" will execute the program.

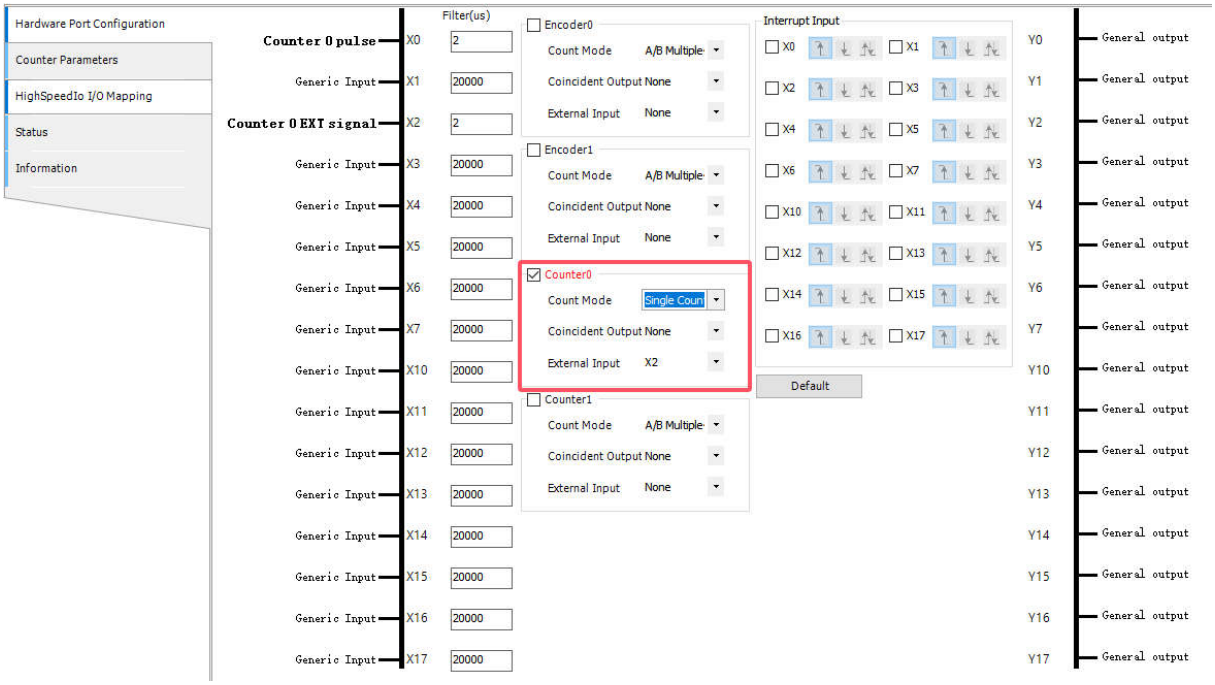


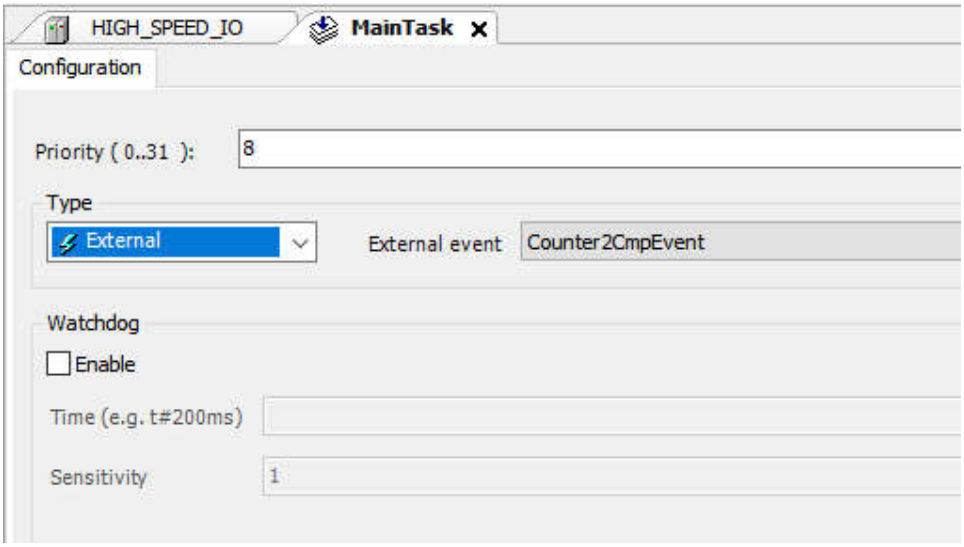

Ports need to be configured on the interface and in the instructions to take effect.

- **Enable compare consistent interruption**

Set the counter ID that needs to be enabled, such as 3 indicating CounterID:=0, CounterID:=1. In the case, M210 and CounterID:=2 are used, so enable compare consistent interrupt writing to 4. After reaching the comparison value, it will enter the external event task to execute instruction.

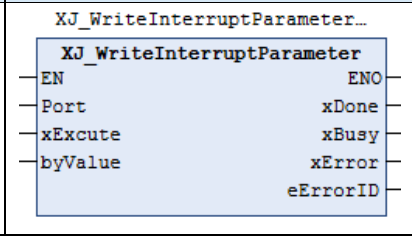



| | | | | | |
|---|---|---|---|---|---|
| XJ_EnableInterrupt_0 | XJ_EnableInterrupt | | | | 打开外部中断和比较一致中断 |
| xEnable | BOOL | FALSE | | | 使能 |
| udiExternal | UINT | 0 | | | 打开外部输入中断 |
| uiCompare | UINT | 4 | | | 打开比较一致中断 |
| xValid | BOOL | FALSE | | | 中断生效 |
| xBusy | BOOL | FALSE | | | 正在运行 |
| xError | BOOL | FALSE | | | 错误标志 |
| eErrorID | HSIO_ERROR | ERR_OK | | | 错误代码 |



## 2-2-12. Interrupt parameter write XJ_WriteInterruptParameter

| Interrupt parameter write [XJ_WriteInterruptParameter] | | | |
|---|---|---|---|
| Execution condition | Edge triggering | Suitable model | XS series (except XSF) |
| Firmware | XSLH-24, XSA no requirements, other XS models require 2.4.0 or above | Software | XS StudioV2.2.0 and above |
| Library | XJ_HSIO | | |

| Command | Name | Graph | ST language |
|---|---|---|---|
| XJ_WriteInterruptParameter | Interrupt parameter write |  | XJ_WriteInterruptParameter(<br>Port:= ,<br>xExcute:= ,<br>byValue:= ,<br>xDone=> ,<br>xBusy=> ,<br>xError=> ,<br>eErrorID=> ); |

(1) Input variables

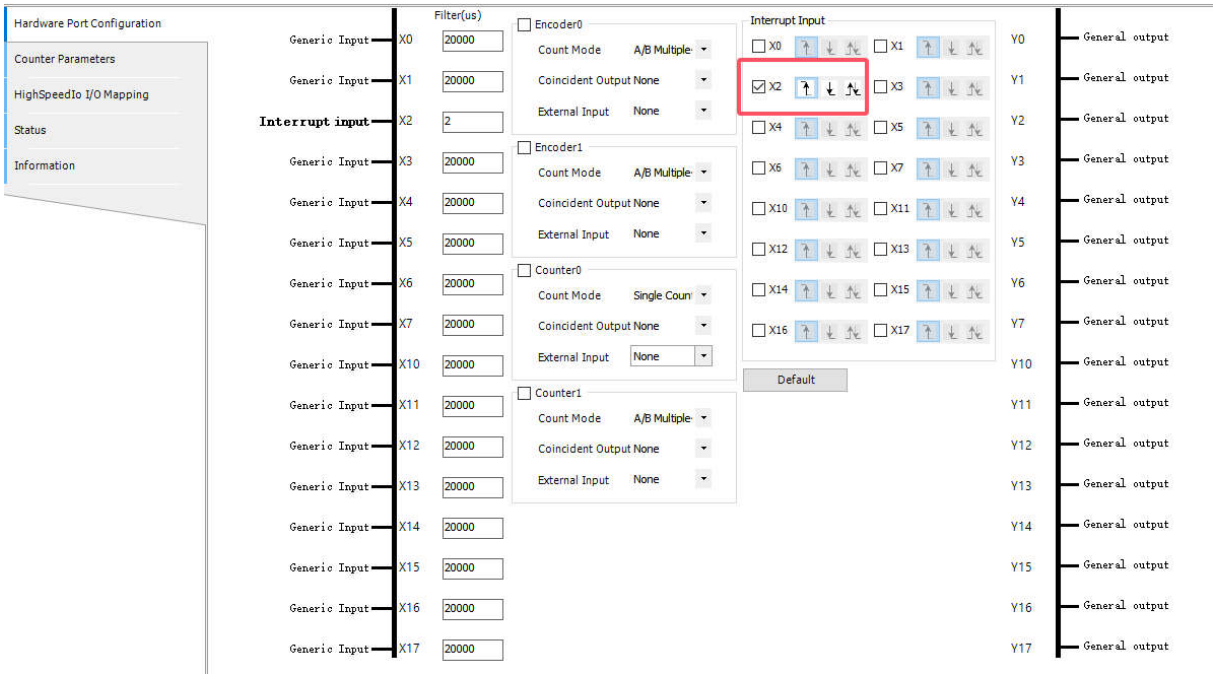
| Input variables | Name | Data type | Effective range | Initial value | Description |
|---|---|---|---|---|---|
| Port | Port no. | UINT | | 0 | Binary such as 2#11 means X0, X1 |
| xExcute | Trigger | BOOL | TRUE,FALSE | FALSE | |
| byValue | Value | BYTE | | 0 | |

(2) Output variables

| Output variables | Name | Data type | Effective range | Initial value | Description |
|---|---|---|---|---|---|
| xDone | Completion flag | BOOL | TRUE,FALSE | FALSE | After completing the setting, the flag is set to TRUE |
| xBusy | Running | BOOL | TRUE,FALSE | FALSE | |
| xError | Error flag | BOOL | TRUE,FALSE | FALSE | |
| eErrorID | Error type | HSIO_ERROR | - | 0 | |

(3) Function

- External interrupt parameters can be configured on the interface or modified using this command.



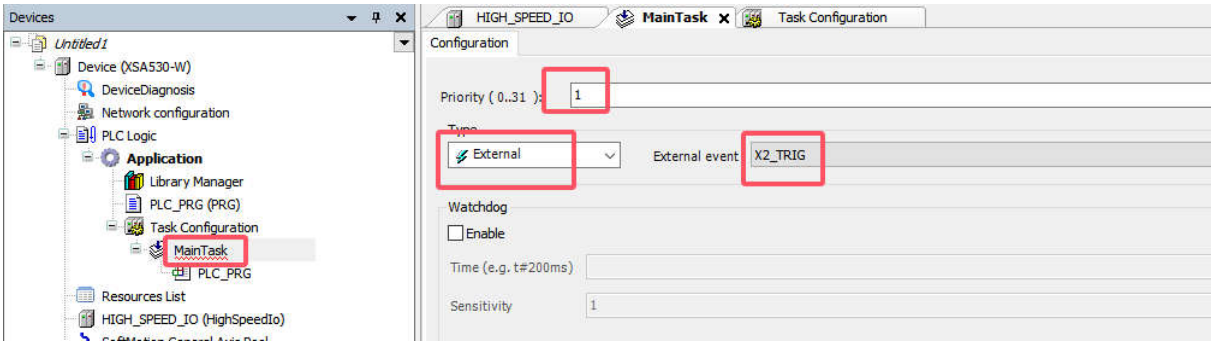

| XJ_WriteInterruptParameter_0 | XJ_HSIO.XJ_WriteInterruptParameter | | | | 中断参数写入 |
|---|---|---|---|---|---|
| Port | UINT | 4 | | | 端口号二进制如：2#11代表X0、X1 |
| xExcute | BOOL | FALSE | | | 触发 |
| byValue | BYTE | 0 | | | 值(0:上升沿，1为下降沿，2为上升沿+下降沿) |
| xDone | BOOL | FALSE | | | 完成标志 |
| xBusy | BOOL | FALSE | | | 正在运行 |
| xError | BOOL | FALSE | | | 错误代码 |
| eErrorID | HSIO_ERROR | ERR_OK | | | 错误标志 |

- If XJ_WriteInterruptParameter is executed earlier than XJ_EnableIterrupt, then the XJ_WriteInterruptParameter is valid.
- XJ_EnableInterrupt is executed earlier than XJ_WriteInterruptParameter, the backend interrupt parameter is valid.
- XJ_WriteInterruptParameter is executed again then XJ_WritelnterruptParameter is valid.
- When executing this instruction, it is also necessary to enable an external input interrupt.

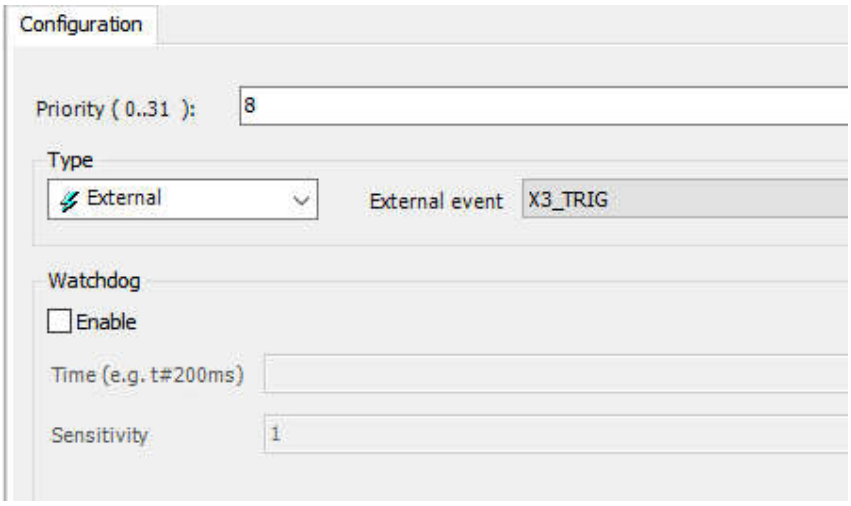
| XJ_EnableInterrupt_0 | XJ_EnableInterrupt | | | | 打开外部中断和比较一致中断 |
|---|---|---|---|---|---|
| xEnable | BOOL | FALSE | | | 使能 |
| udiExternal | UINT | 4 | | | 打开外部输入中断 |
| uiCompare | UINT | 4 | | | 打开比较一致中断 |
| xValid | BOOL | FALSE | | | 中断生效 |
| xBusy | BOOL | FALSE | | | 正在运行 |
| xError | BOOL | FALSE | | | 错误标志 |
| eErrorID | HSIO_ERROR | ERR_OK | | | 错误代码 |

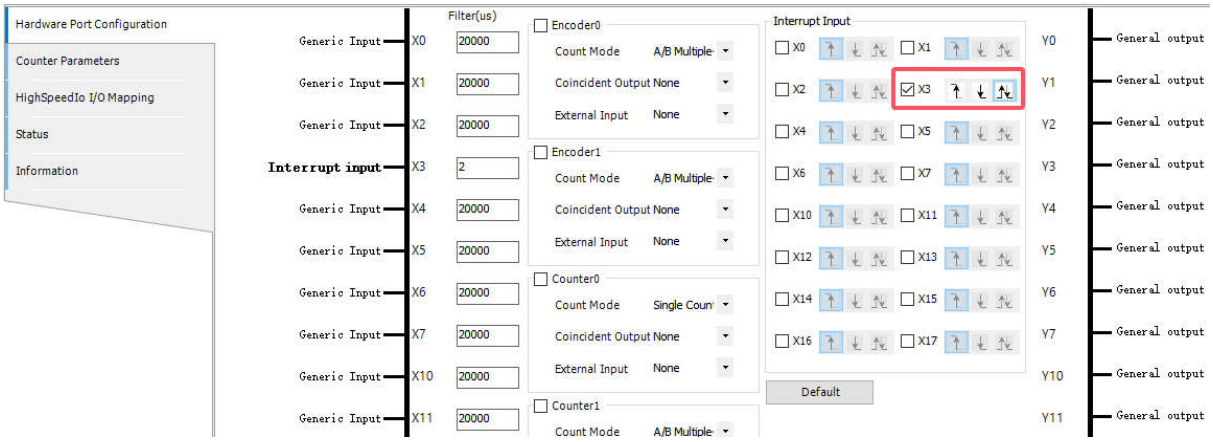
## 2-2-13. Interrupt application

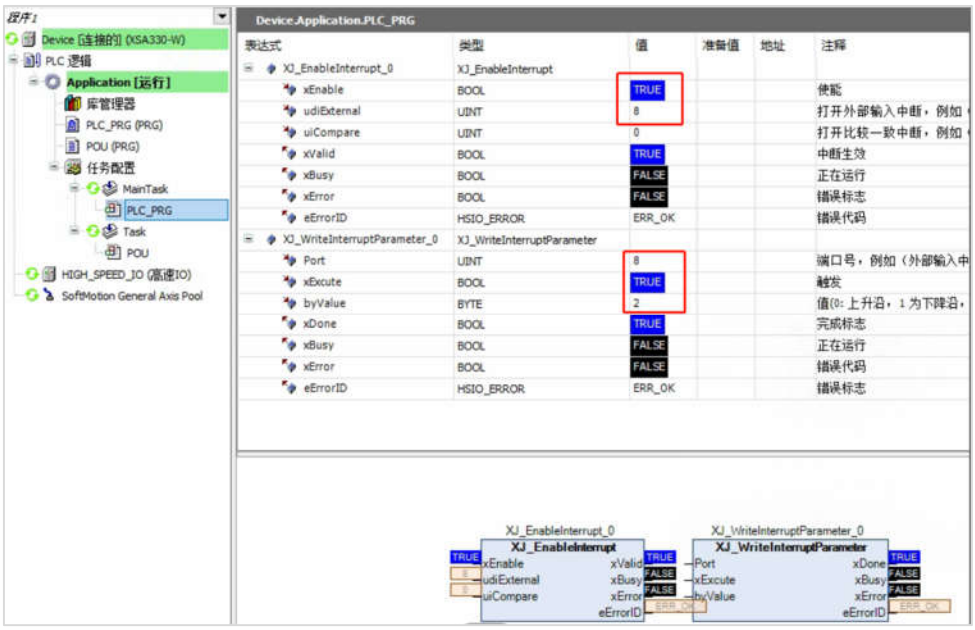
Double click on "Task" and set it as an external event "External" in the pop-up interface - external interrupt uses terminal X, or you can set the priority of external interrupt events.




Example: Use the [XJ_EnnableInterrupt] and [XJ_WriteInterruptParameter] instructions. Set X3 as an external interrupt input and take its dual edge signals. This can be configured in the hardware parameter configuration interface or using the XJ_WriteInterruptParameter instruction. Once an edge signal is given to X3, the self increment instruction in the POU program under another task (configured as external, X3_TRIG) will be executed. The parameter configuration and instructions are shown in the following figure.




程序1
Device [连接的] (XSA330-W)
PLC 逻辑
Application [运行]
库管理器
PLC_PRG (PRG)
POU (PRG)
任务配置
MainTask
PLC_PRG
Task
POU
HIGH_SPEED_IO (高速IO)
SoftMotion General Axis Pool
Device.Application.PLC_PRG
表达式 类型 值 准备值 地址 注释
XJ_EnableInterrupt_0 XJ_EnableInterrupt
xEnable BOOL TRUE 使能
udiExternal UINT 8 打开外部输入中断，例如
uiCompare UINT 0 打开比较一致中断，例如
xValid BOOL TRUE 中断生效
xBusy BOOL FALSE 正在运行
xError BOOL FALSE 错误标志
eErrorID HSIO_ERROR ERR_OK 错误代码
XJ_WriteInterruptParameter_0 XJ_WriteInterruptParameter
Port UINT 8 端口号，例如（外部输入中
xExcute BOOL TRUE 触发
byValue BYTE 2 值(0: 上升沿，1 为下降沿，
xDone BOOL TRUE 完成标志
xBusy BOOL FALSE 正在运行
xError BOOL FALSE 错误代码
eErrorID HSIO_ERROR ERR_OK 错误标志
XJ_EnableInterrupt_0
XJ_EnableInterrupt
xEnable
udiExternal
uiCompare
xValid
xBusy
xError
eErrorID
XJ_WriteInterruptParameter_0
XJ_WriteInterruptParameter
Port
xExcute
byValue
xDone
xBusy
xError
eErrorID

Device.Application.POU
表达式 类型 值
A INT 1
1 A 1 :=A 1 +1; RETURN

## 2-3. XinjeCnt high speed count library

### 2-3-1. High speed counter【XJ_Counter】

| High speed counter [XJ_Counter] | | | |
|---|---|---|---|
| Execution condition | Normally ON/OFF | Suitable model | XS3, XSDH, XSLH-30A32 |
| Firmware | Below v2.4.0 | Software | - |
| Library | XinjeCnt | | |

| Instruction | Name | Graph | ST language |
|---|---|---|---|
| XJ_Counter | High speed count | XJ_Counter_0<br>XJ_Counter<br>EN ENO<br>Counter CounterValue<br>Enable Error<br>Mode ErrorID | XJ_Counter(<br>Counter:= ,<br>Enable:= ,<br>Mode:= ,<br>CounterValue=> ,<br>Error=> ,<br>ErrorID=> ); |

(1) Input variables

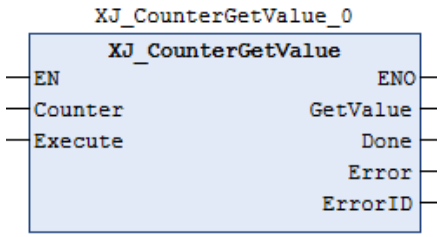
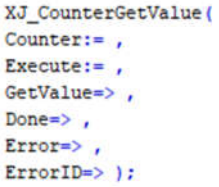
| Input variables | Name | Data type | Effective range | Initial value | Description |
|---|---|---|---|---|---|
| Counter | Counter | COUNTER_REF | - | - | High speed counter, which specifies the input terminal and initial value of high-speed counting |
| Enable | Enable | BOOL | TRUE,FALSE | FALSE | Normally open the enable to start counting |
| Mode | Count mode | Mode | AB_Mode，Single_Mode | FALSE | High speed count mode: MODE=XJ.AB_Mode is AB phase high speed count. MODE=XJ.Single_Mode is single phase high speed count. |

(2) Output variables

| Output variables | Name | Data type | Effective range | Initial value | Description |
|---|---|---|---|---|---|
| CounterValue | Count value | DINT | Data range | 0 | High speed count value |
| Error | Error flag | BOOL | TRUE,FALSE | FALSE | |
| ErrorID | Error type | UINT | - | 0 | |

## 2-3-2. Read high speed counter【XJ_CounterGetValue】

| Read high speed counter [XJ_CounterGetValue] | | | |
|---|---|---|---|
| Execution condition | Normally ON/OFF | Suitable model | XS3, XSDH, XSLH-30A32 |
| Firmware | Below v2.4.0 | Software | - |
| Library | XinjeCnt | | |

| Instruction | Name | Graph | ST language |
|---|---|---|---|
| XJ_CounterGetValue | Read high speed counter |  |  |





(1) Input variables

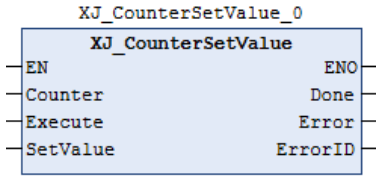
| Input variables | Name | Data type | Effective range | Initial value | Description |
|---|---|---|---|---|---|
| Counter | Counter | COUNTER_REF | - | - | High speed counter, which specifies the input terminal and initial value of high-speed counting |
| Execute | Enable | BOOL | TRUE,FALSE | FALSE | Triggered by the rising edge, reading the current high-speed count value |

(2) Output variables

| Output variables | Name | Data type | Effective range | Initial value | Description |
|---|---|---|---|---|---|
| GetValue | Read value | DINT | Data range | 0 | Present count value |
| Done | Completion | BOOL | TRUE,FALSE | FALSE | After completing the read, the flag is TRUE |
| Error | Error flag | BOOL | TRUE,FALSE | FALSE | |
| ErrorID | Error type | UINT | - | 0 | |

## 2-3-3. Write high speed counter【XJ_CounterSetValue】

| Write high speed counter [XJ_CounterSetValue] | | | |
|---|---|---|---|
| Execution condition | Normally ON/OFF | Suitable model | XS3, XSDH, XSLH-30A32 |
| Firmware | Below v2.4.0 | Software | - |
| Library | XinjeCnt | | |

| Instruction | Name | Graph | ST language |
|---|---|---|---|
| XJ_CounterSetValue | Write high speed counter |  | XJ_CounterSetValue(<br>Counter:= ,<br>Execute:= ,<br>SetValue:= ,<br>Done=> ,<br>Error=> ,<br>ErrorID=> ); |

(1) Input variables

| Input variables | Name | Data type | Effective range | Initial value | Description |
|---|---|---|---|---|---|
| Counter | Counter | COUNTER_REF | - | - | High speed counter, which specifies the input terminal and initial value of high-speed counting |
| Execute | Enable | BOOL | TRUE,FALSE | FALSE | Trigger by rising edge, write high-speed count value, write SetValue in CounterValue |
| SetValue | Write in value | DINT | Data range | 0 | Write high-speed counting set value |

(2) Output variables

| Output variables | Name | Data type | Effective range | Initial value | Description |
|---|---|---|---|---|---|
| Done | Completion | BOOL | TRUE,FALSE | FALSE | After completing the write, the flag bit is TRUE |
| Error | Error flag | BOOL | TRUE,FALSE | FALSE | |
| ErrorID | Error type | UINT | - | 0 | |

Note: If the value displayed for ErrorID is 2, it is because the CounterID range is not between 0 and 3.

(3) Function description

- The high-speed counting function has three functional blocks, namely the high-speed counting function block, the read high-speed counting function block, and the write high-speed counting function block. XS3 series high-speed input can only receive differential signal (DIFF), but cannot receive Open collector signal (OC). Please be sure to select differential signal encoder. XSDH series high-speed input is to receive Open collector signal (OC).
- Counter is COUNTER_REF data type:
  COUNTER_REF specific description is as follows:

| Member | Name | Data type | Effective range | Initial value | Description |
|---|---|---|---|---|---|
| CounterID | Counter terminal | INT | 0,1,2,3 | 0 | Select high-speed counter input port |
| CounterValue | Counter initial value | DINT | data range | 0 | Setting the initial value of the counter |

- The XS3 and XSDH series high-speed counting functions have two modes, namely single phase incremental mode and AB phase mode.

(1) Mode= Single_Mode

In this mode, the input pulse signal is counted, and the count value increases with the rising edge of each pulse signal.

(2) Mode=AB_Mode

In this mode, the high-speed counting value is incremented or decremented based on the pulse signal (A phase and B phase) with a phase difference of 90°. The default counting mode is 4 times frequency.

- XS series high-speed counting input port allocation

| XS3-26T4 | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | Single phase mode | | | | AB phase mode | | | |
| CounterID | 0 | 1 | 2 | 3 | 0 | 1 | 2 | 3 |
| Max frequency | 200k | 200k | 200k | 200k | 200k | 200k | 200k | 200k |
| X0+ | U+ | | | | A+ | | | |
| X0- | U- | | | | A- | | | |
| X1+ | | | | | B+ | | | |
| X1- | | | | | B- | | | |
| X2 | | | | | | | | |
| X3+ | | U+ | | | | A+ | | |
| X1- | | U- | | | | A- | | |
| X4+ | | | | | | B+ | | |
| X2- | | | | | | B- | | |
| X5 | | | | | | | | |
| X6+ | | | U+ | | | | A+ | |
| X6- | | | U- | | | | A- | |
| X7+ | | | | | | | B+ | |
| X7- | | | | | | | B- | |
| X10 | | | | | | | | |
| X11+ | | | | U+ | | | | A+ |
| X11- | | | | U- | | | | A- |
| X12+ | | | | | | | | B+ |
| X12- | | | | | | | | B- |
| X13 | | | | | | | | |

| XSDH-60A32-E | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | Single phase mode | | | | AB phase mode | | | |
| CounterID | 0 | 1 | 2 | 3 | 0 | 1 | 2 | 3 |
| Max frequency | 200k | 200k | 200k | 200k | 100k | 100k | 100k | 100k |
| X0 | U | | | | A | | | |
| X1 | | | | | B | | | |
| X2 | | | | | | | | |
| X3 | | U | | | | A | | |
| X4 | | | | | | B | | |
| X5 | | | | | | | | |
| X6 | | | U | | | | A | |
| X7 | | | | | | | B | |
| X10 | | | | | | | | |
| X11 | | | | U | | | | A |
| X12 | | | | | | | | B |
| X13 | | | | | | | | |

| XSLH-30A32 | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | Single phase mode | | | | AB phase mode | | | |
| CounterID | 0 | 1 | 2 | 3 | 0 | 1 | 2 | 3 |
| Max frequency | 200k | 200k | 80k | 80k | 100k | 100k | 50k | 50k |
| X0+ | U+ | | | | A+ | | | |
| X0- | U- | | | | A- | | | |
| X1+ | | | | | B+ | | | |
| X1- | | | | | B- | | | |
| X2 | | | | | | | | |
| X3+ | | U+ | | | | A+ | | |
| X1- | | U- | | | | A- | | |
| X4+ | | | | | | B+ | | |
| X2- | | | | | | B- | | |
| X5 | | | | | | | | |
| X6 | | | U | | | | A | |
| X7 | | | | | | | B | |
| X10 | | | | | | | | |
| X11 | | | | U | | | | A |
| X12 | | | | | | | | B |
| X13 | | | | | | | | |
| X14 | | | | | | | | |
| X15 | | | | | | | | |

## 2-3-4. Application example

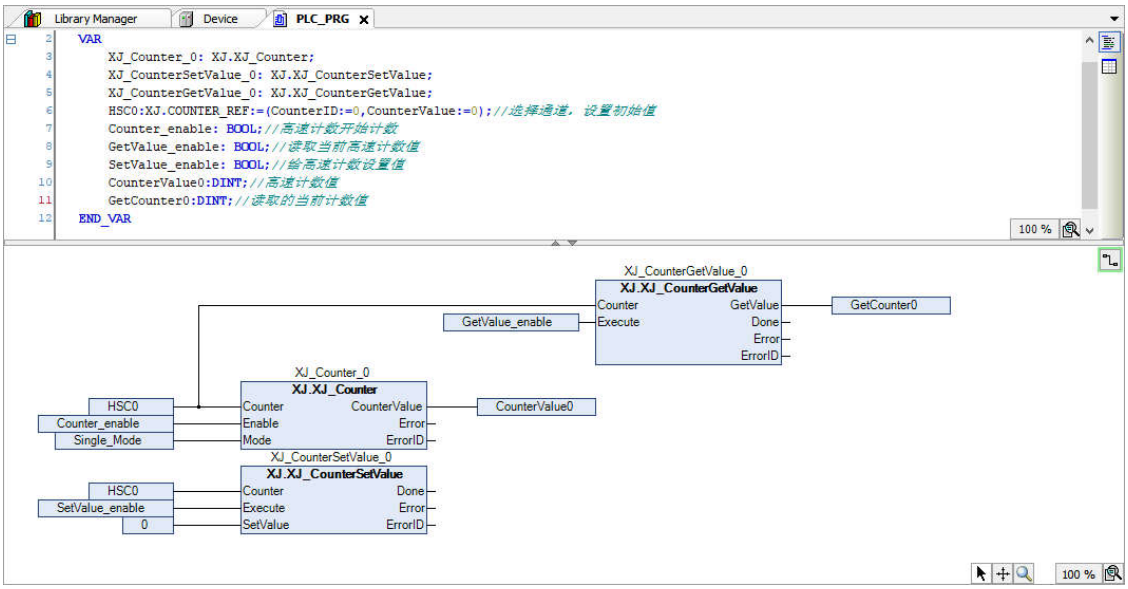
Example 1: Use the first channel of high-speed count and read the current count value in the count to modify the current high-speed count value.

Program operation:

(1) Install the required libraries according to the steps in sections 2-1-3.

(2) Write a high-speed counting program.

Programming: Use the function blocks "XJ.XJ_Counter", "XJ.XJ_CounterGetValue", and "XJ.XJ_CounterSetValue" to set the high-speed counting port, high-speed counting mode, and high-speed counting value used in the program.

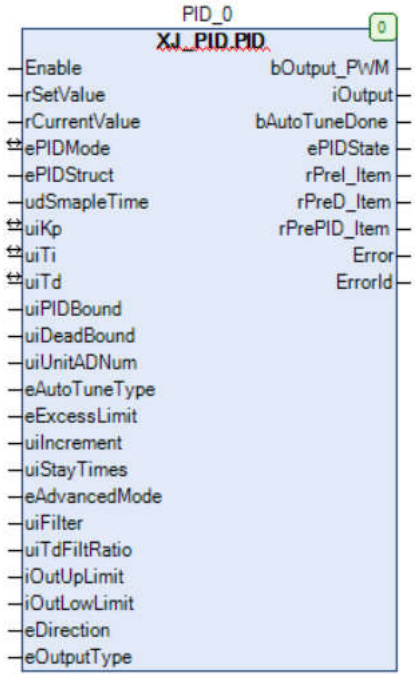


# 3. PID instructions

The PID function block is copyrighted and can only be used on Xinje's PLC! Simulation not available!

## 3-1. Command format

| PID function [PID] | | | |
|---|---|---|---|
| Execution condition | Normally ON/OFF triggering | Suitable model | XS, XSA model |
| Firmware | - | Software | XS StudioV2.2.0 and above |
| Library | XJ_PID | | |

| Command | Name | Graph | ST language |
|---|---|---|---|
| XJ_PID.PID | PID instruction |  | PID_0(<br>Enable:= ,<br>rSetValue:= ,<br>rCurrentValue:= ,<br>ePIDMode:= ,<br>ePIDStruct:= ,<br>udSmapleTime:= ,<br>uiKp:= ,<br>uiTi:= ,<br>uiTd:= ,<br>uiPIDBound:= ,<br>uiDeadBound:= ,<br>uiUnitADNum:= ,<br>eAutoTuneType:= ,<br>eExcessLimit:= ,<br>uiIncrement:= ,<br>uiStayTimes:= ,<br>eAdvancedMode:= ,<br>uiFilter:= ,<br>uiTdFiltRatio:= ,<br>iOutUpLimit:= ,<br>iOutLowLimit:= ,<br>eDirection:= ,<br>eOutputType:= ,<br>bOutput_PWM=> ,<br>iOutput=> ,<br>bAutoTuneDone=> ,<br>ePIDState=> ,<br>rPreI_Item=> ,<br>rPreD_Item=> ,<br>rPrePID_Item=> ,<br>Error=> ,<br>ErrorId=> ); |

3-2. Related variables

| VAR_INPUT | Name | Data type | Effective range | Initial value | Description |
|---|---|---|---|---|---|
| Enable | Enable | BOOL | TRUE/FALSE | FALSE | Must be set to TRUE to activate the processing of the function block |
| rSetValue | Set value | REAL | Data range | 0 | *Target setting value |
| rCurrentValue | Current feedback | REAL | Data range | 0 | *Current feedback value |
| ePIDStruct | PD control structure | PIDCONTRLMODE | - | - | PD control structure |
| udSmapleTime | Sampling time | UDINT | Data range | 1000 | *Sampling time (ms) Suggest setting it between 1000-2000 |
| uiPIDBound | Calculation range | UINT | 0~32767 | 0 | PID calculation range |
| uiDeadbaund | Control dead zone | UINT | 0~32767 | 0 | PID control dead zone |
| uiUnitADNum | Self tuning unit temperature corresponding digital value | UINT | 0~65535 | 10 | Self tuning unit temperature corresponding digital value |
| eAutoTuneType | Self tuning method | AUTOTUNETYPE | - | - | Self tuning method<br>0: Step response method<br>1: Critical oscillation method |
| eExcessLimit | Overshoot setting, only effective in self-tuning mode - step response method | PIDEXCESSLIMIT | - | - | Overshoot setting, only effective in self-tuning mode - step response method |
| uiIncrement | Step response method tuning completion transition section percentage of target value increment for each adjustment | UINT | - | 0 | Step response method tuning completion transition section percentage of target value increment for each adjustment 1~100%. Only the self-tuning mode step response method is effective |
| uiStayTimes | The current target value staying times, only the self-tuning mode step response method is effective | UINT | 0~32767 | 0 | The current target value staying times, only the self-tuning mode step response method is effective |
| eAdvancedMode | Advanced mode | PidMode | | | Advanced mode |
| uiFilter | Input filtering constant | UINT | 0~99% | - | Input filtering constant |
| uiTdFiltRatio | Delay coefficient of differential action | UINT | 0~100% | 50 | Delay coefficient of differential action |
| iOutUpLimit | Output upper | INT | -32767~32767 | 4095 | Output upper limit setting |

| | | | | | |
|---|---|---|---|---|---|
| | limit setting value | | | | value |
| iOutLowLimit | Output lower limit setting value | INT | -32767~32767 | 0 | Output lower limit setting value |
| eDirection | Direction<br>0: reverse action<br>1: positive action | DIRECTIONPARAMETER | - | - | Direction<br>0: reverse action<br>1: positive action |
| eOutputType | Output mode | OUTPUTTYPE | - | - | Output<br>0: PWM output<br>1: Digital output |
| VAR_IN_OUT | Name | Data type | Effective range | Initial value | Description |
| ePIDMode | Control mode | PIDTUNEMODE | - | - | PID mode<br>0: Manual PID mode (default)<br>1: Self tuning mode |
| uiKp | Proportional gain | UINT | 0~32767 | 0 | Proportional gain |
| VAR_IN_OUT | Name | Data type | Effective range | Initial value | Description |
| uiTi | Integration time | UINT | 0~32767 | 0 | integration time<br>0~32767*100ms |
| uiTd | Differential time | UINT | 0~32767 | 0 | Differential time<br>0~32767*10ms |
| VAR_OUTPUT | Name | Data type | Effective range | Initial value | Description |
| bOutput_PWM | PWM output | BOOL | TRUE/FALSE | FALSE | PWM output |
| iOutput | Digital output | INT | - | - | Digital output |
| bAutoTuneDone | Self tuning completion flag | BOOL | TRUE/FALSE | FALSE | Self tuning completion TRUE |
| ePIDState | Current PID status | PIDSTATE | - | - | Current PID status |
| rPreI_Item | Last time integral item | REAL | - | - | Last time integral item |
| rPreD_Item | Last time differential item | REAL | - | - | Last time differential item |
| rPrePID_Item | Last time PID total output | REAL | - | - | Last time PID total output |
| Error | Error flag | BOOL | | | Error flag |
| ErrorId | Error code | ERRID_PID | | | Error code |

## 3-3. Function description

- Direction of action

Positive action: The action where the output value MV of the operation increases with the increase of the measured value PV, usually used for cooling control.
Reverse action: An action in which the output value MV decreases as the measured value PV increases, usually used for heating control.

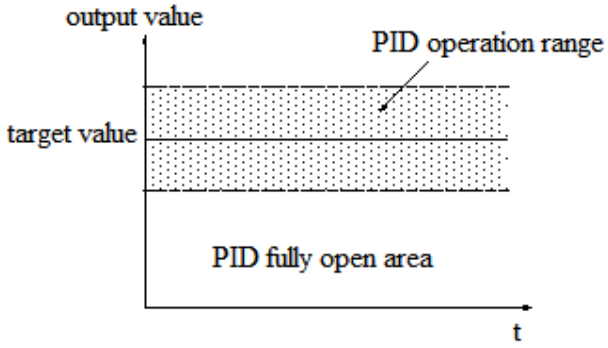
- Advanced mode setting: enable advanced mode, and users can set parameters such as filtering time, differential gain, and output upper and lower limits.

- Sample time

The system samples the current value at a certain time interval and compares it with the output value. This time interval is the sampling time T. When DA is output, T has no limit. When the port outputs, T must be greater than 1 PLC program scan cycle. The value of T should be within the range of 100-1000 PLC scanning cycles.

- PID operation range

When the system is running, it is initially in the PID fully open stage, which approaches the target value at the fastest speed (default is 4095). When it reaches the operating range of the PID, the parameters Kp, TI, and TD begin to take control. As shown in the following figure:




When the user is not clear about the specific set value of PID parameters, the Self-tuning mode can be selected to enable the system to automatically find the best control parameters (proportional gain Kp, integration time Ti, differential time TD).

- Control object applicable to Self-tuning mode: temperature and pressure. Unsuitable control objects: liquid level and flow rate, etc.

- Self tuning is the process of extracting PID parameters. Sometimes, self-tuning once may not find the optimal parameters, and multiple rounds of self-tuning are required. It is normal for oscillations to occur during the process. After self-tuning is completed and the optimal parameters are found, the PID should be manually adjusted. If the controlled object is unstable during the manual PID process and cannot be controlled at a constant target value, it may be due to poor parameter adjustment, and PID parameters need to be adjusted again to achieve stable control.

- To enter self-tuning mode, please set "ePIDMode" to AutoTuneMode and enable PID operation conditions. After observing the "bAutoTuneDone" (successful self-tuning flag), it indicates successful self-tuning.

- Two self-tuning methods

Step response method: Before self-tuning, it is necessary to ensure that the starting temperature is consistent with the ambient temperature.

  - The step response method allows users to set the PID control cycle (sampling time) to 0 in advance at the beginning of self-tuning, and can also be manually modified according to actual needs after tuning is completed.

  - For the step response method, the system must be in a steady state under uncontrolled conditions before self-tuning. For temperature control objects, it means that the current measured temperature is consistent with the ambient temperature.

Critical oscillation method: The self-tuning starting temperature can be arbitrary.

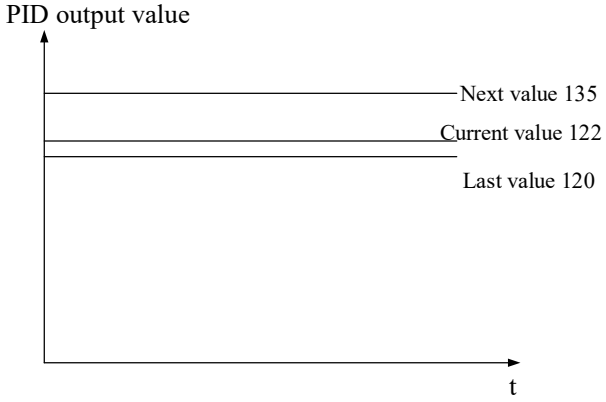
- At the beginning of self-tuning in the critical oscillation method, users need to set the PID control cycle (sampling time) in advance. Reference value: For systems with slow response, it can be set to 1000ms, while for systems with fast response, it can be set to 10ms-100ms.
- The critical oscillation method is used for self-tuning, and the system can start from any state. For temperature control objects, it means that the current measured temperature does not need to be consistent with the ambient temperature. It can be below or above the target temperature.

- Switching of output values

  When the output value required by the user is a digital value, set the eOutputType value to 1; When the output value required by the user is a digital value, set the eOutputType value to 0.

- Control dead zone

  When the PID output value fluctuates within a small range, it will not have much impact on the controlled system. Therefore, it is not necessary to control the active power output of the system very accurately in practice. To achieve this, a dead zone can be set in the PID controller that controls active power. In addition, if the output value is constantly adjusted, the relevant mechanical components will wear out quickly. By setting a control dead zone, this situation can be avoided.




Assuming that we set the control dead zone value to 10 at this time, in the above figure, the PID output current value (122) has only a change of 2 compared to the previous value (120). After PID operation, the result (122) is kept in the internal register, and the final output value remains the previous output value (still 120). The next value 135 has a change of 13 compared to 122, which is greater than the control dead zone value. The actual output value of PID becomes 135.

- PID self-tuning cycle variation value

When self-tuning, set this value in "uiUnitADNum". This setting determines the self-tuning performance, usually setting an AD value corresponding to a standard measurement unit. The default value is 10, and it is recommended to set the range: full-scale AD value × 0.3~1%. Generally, no changes are required, but if the system is greatly disturbed by external factors, it is necessary to increase this value moderately to avoid errors in determining positive/negative actions. If the value is too large, the PID control cycle (sampling time) set may be too long, so it is necessary to avoid setting larger values.

- PID self-tuning overshoot allowed setting

  - When set to NoLimit, overshoot is allowed, and the system can always select the optimal PID parameters. However, during the tuning process, the measured value may be lower or higher than the target value, and safety factors should be considered.
  - When set to LittleExcess, overshoot is not allowed.

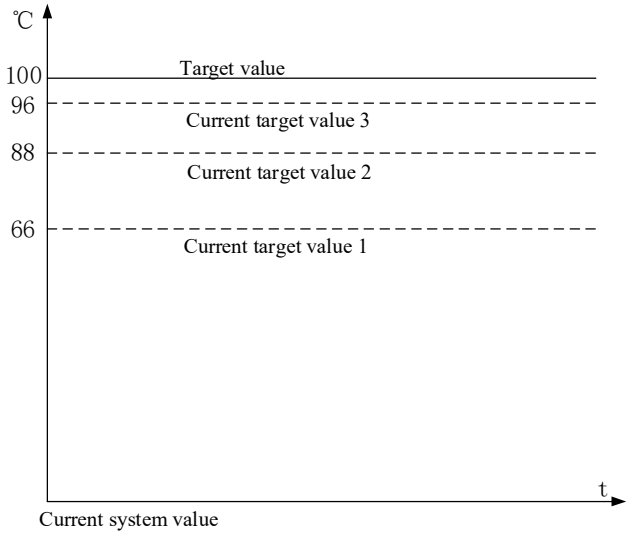
For control objects with strict safety requirements, such as pressure vessels, in order to avoid situations where the measured values significantly exceed the target values during the self-tuning process. Set to 'LittleExcess' to avoid overshoot.

During this process, if 'bAutoTuneDone' changes from 0 to 1, it indicates successful self-tuning and obtaining the optimal parameters; If "bAutoTuneDone" remains at 0 until "ePIDMode" changes from AutoTuneMode to ManualPIDMode, it indicates that the self-tuning is complete and the obtained parameters are not the optimal parameters, which may require some manual adjustments.

- The percentage of each adjustment of the current target value during the transition phase after self-tuning is completed (uiIncrement)

This parameter is only valid when 'eExcessLimit' is NoLimit. If entering normal PID control directly after self-tuning, it is easy to produce small overshoot. Appropriately reducing the parameter value is beneficial for suppressing overshoot, but if the value is too small, it can easily cause response lag. The default value is 100%, which means that this parameter is not working. Suggest adjusting the range to 50-80%.

For example, if the current target value is adjusted by a ratio of 2/3 each time (i.e. "uiIncrement" is 67%), the initial temperature of the system is 0 degrees, and the target temperature is 100 degrees, the current target temperature adjustment situation is as follows:




The next current target value=current target value+(final target value - current target value) * 2/3; The current target value of the system changes in the order of 66℃, 88℃, 96℃, 98℃, 99℃, and 100℃.

- Limit the number of times exceeding the target value in the self-tuning transition phase during overshoot (uiStayTimes)

  This parameter is only valid when 'eExcessLimit' is NoLimit. If entering normal PID control directly after self-tuning, it is easy to produce small overshoot. Appropriately increasing the parameter value is beneficial for suppressing overshoot, but if the value is too large, it can easily cause response lag. The default value is 15 times, and it is recommended to adjust the range to 5-20.

- Advanced mode

  In order to improve the PID control effect, users can set relevant parameters in advanced mode.

  - Input filtering constant: It has the effect of smoothing the changes in sampling values. The default value is 0%, indicating no filtering.
  - Differential action delay coefficient: a low-pass filtering element that has the effect of mitigating sharp changes in output values. The default value is 50%, increasing this value will make the mitigating effect more pronounced, and most users do not need to make any changes.

- Output upper and lower limit set values

  Users can select the output range of analog signals by setting this value.

  Default value: Output lower limit=0, output upper limit=4095

- Output methods

PWM output: The output point is used to output digital signals;

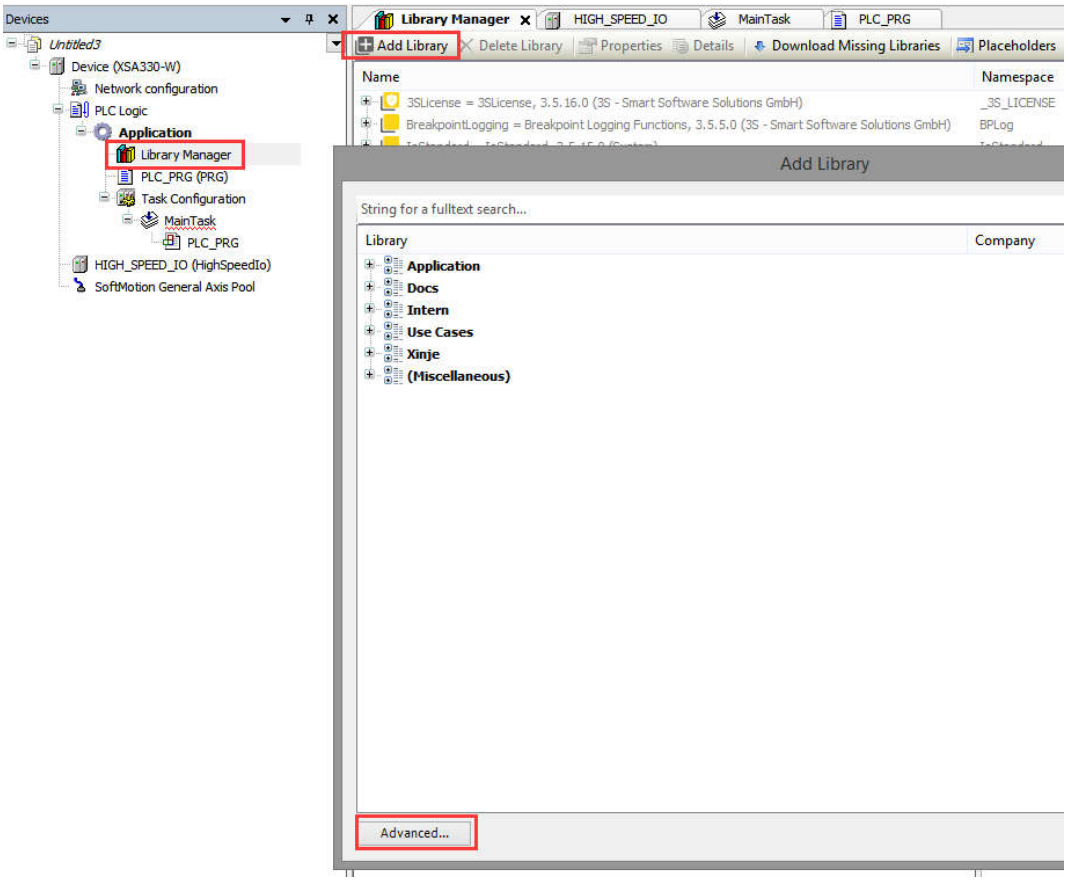
Digital output: The calculation result is stored in the ioutput output value of the PID block.

- Control structure

It takes effect during self tuning and does not take effect in manual mode. The PID structure is set, and after tuning, three parameters kp, Ti, and Td will be given; Set the PI structure, and after tuning, kp and Ti parameters will be provided; Set the P structure and provide kp parameters after tuning is complete.

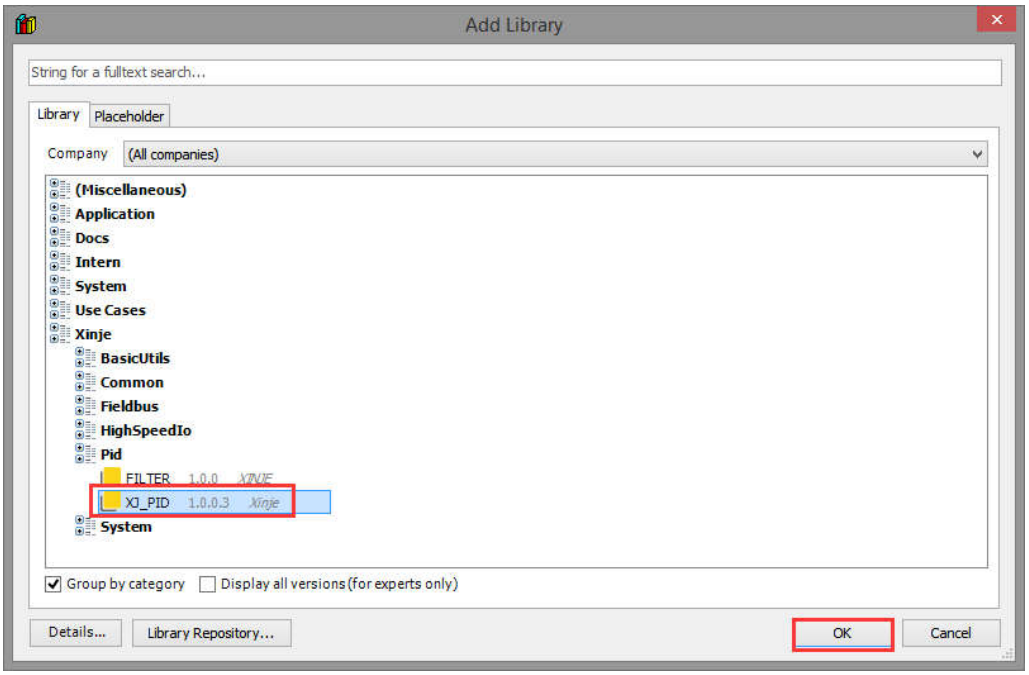
## 3-4. Application example

Double click Library manager → add library →advanced.



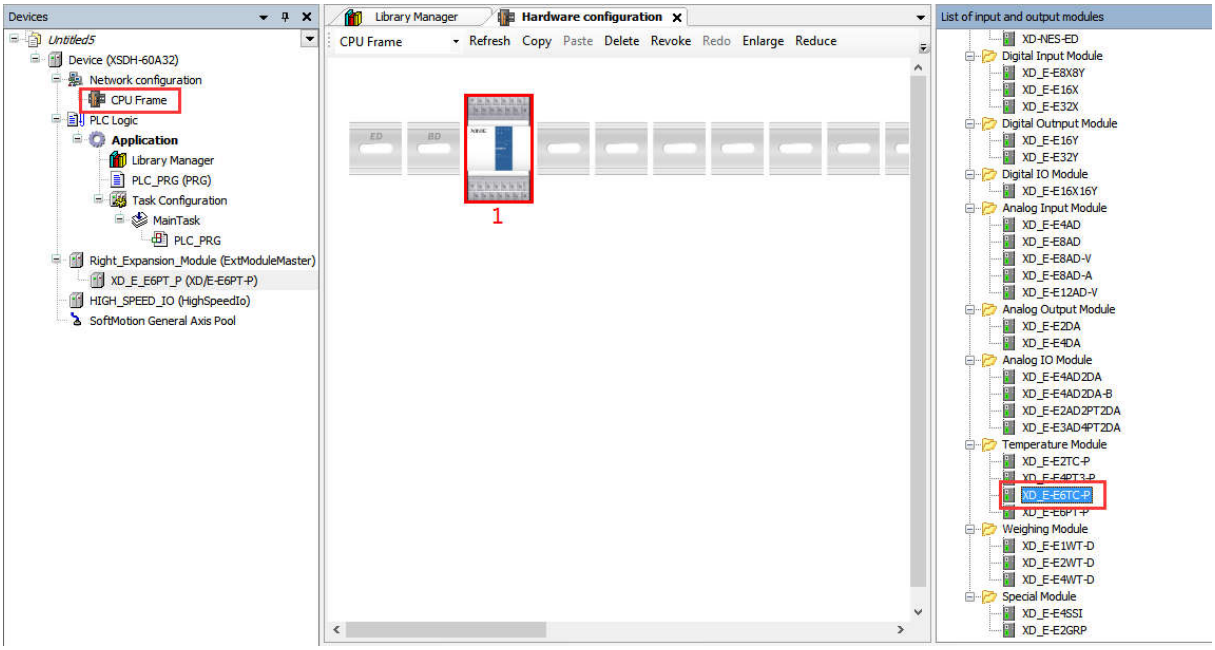

The PID function block is protected by copyright and can only be used on Xinje's PLC!
Simulation not available!

Add XJ_PID.

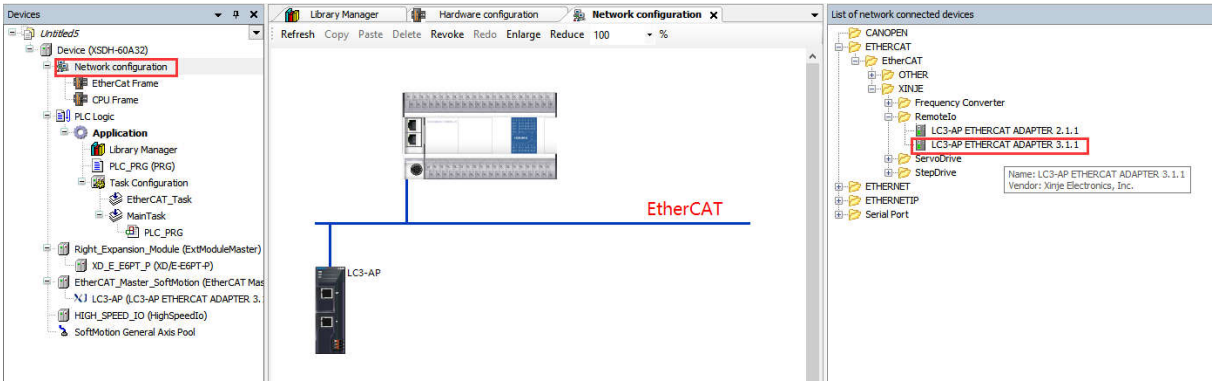



Add temperature expansion module.

- Double click CPU Frame, select position 1, then double click XD-E6TC-P.

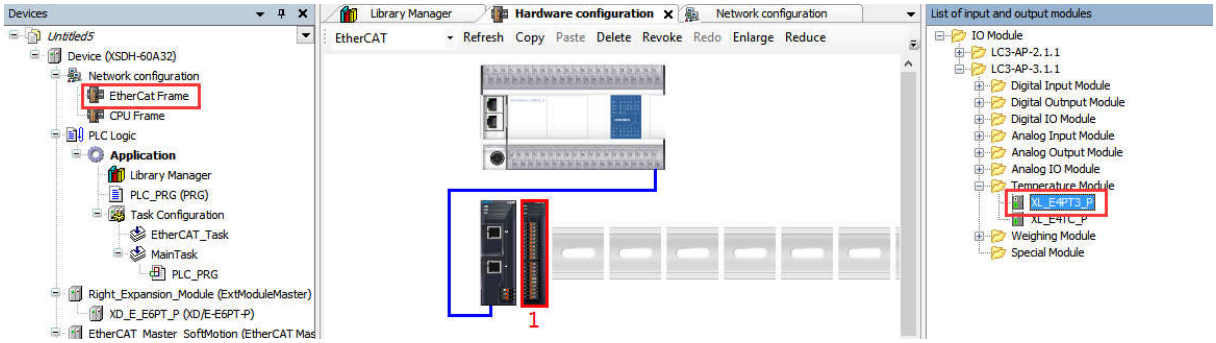



- Remote IO module: double click Network configuration, double click LC3-AP Ethercat adapter 3.1.1.

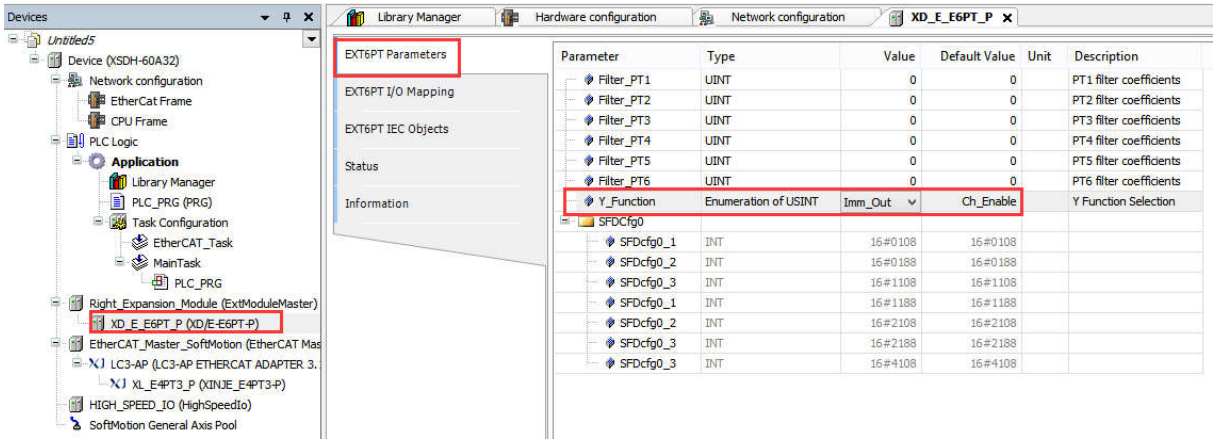



Note: Currently, only LC3-AP V3.1 and above support EC_From TO instruction.

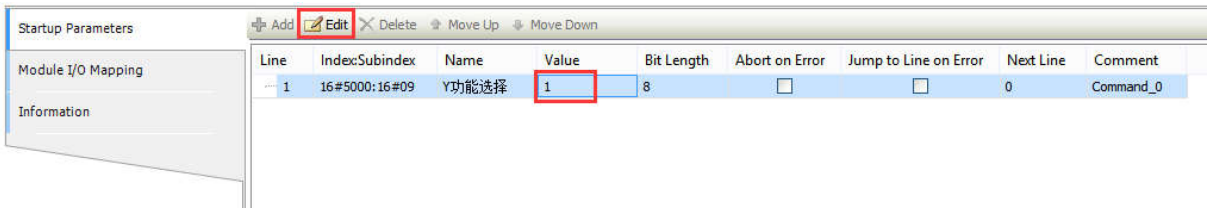
Double click EtherCat Frame, select position 1, double click XL-E4PT3-P.




If a switch quantity control relay is needed, the module on the main body needs to set the Y_Function to 'Imm Out'.




The module on remote IO needs to add the corresponding startup parameter to the default value of 1.




After modifying the module, the parameters need to be powered on again to take effect.

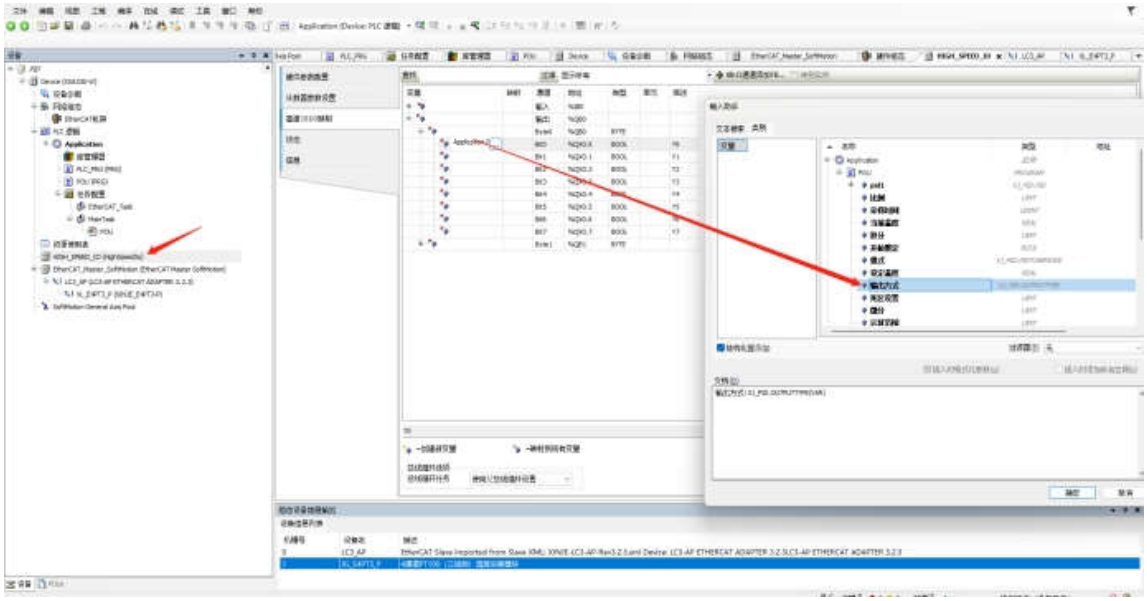
Define the required Global variable.

```
PROGRAM POU
VAR
    pid1:XJ_PID.PID;
    开始整定:BOOL;
    设定温度:REAL;
    当前温度:REAL;
    模式: XJ_PID.PIDTUNEMODE;
    采样时间: UDINT;
    比例: UINT;
    积分: UINT;
    微分: UINT;
    运算范围: UINT;
    死区设置: UINT;
    自整定方式: XJ_PID.AUTOTUNETYPE;
END_VAR
```

Write the program.

```
pid1(
    Enable:= 开始整定, //使能信号
    rSetValue:=设定温度 , //温度设定值
    rCurrentValue:=当前温度 , //当前温度值
    ePIDMode:= 模式, //PID模式：0为手动PID模式，1为自整定模式
    ePIDStruct:= , //PID控制结构
    udSmapleTime:= 采样时间, //采样时间（ms）
    uiKp:= 比例, //比例增益（0-327670%）
    uiTi:=积分, //积分时间（0-32767*100ms）
    uiTd:= 微分, //微分时间（0-32767*10ms）
    uiPIDBound:=运算范围, //PID运算范围（0-32767）
    uiDeadBound:= 死区设置, //PID控制死区（0-32767）
    uiUnitADNum:= , //自整定单位温度对应数字量值（0-65535）
    eAutoTuneType:=自整定方式, //自整定方法(0：阶跃响应法，1:临界震荡法)
    eExcessLimit:= , //超调设置，(仅自整定模式-阶跃响应法生效)
    uiIncrement:= , //阶跃响应法整定完成过渡段每次调整目标值增量的百分比，1~100%,仅自整定模式-阶跃响应法生效)
    uiStayTimes:= ,//当前目标值停留次数,0~32767,仅自整定模式-阶跃响应法生效
    eAdvancedMode:= , //高级模式
    uiFilter:= , //输入滤波系数（0-99%）
    uiTdFiltRatio:= , //微分作用延迟系数（0-100%）
    iOutUpLimit:= , //输出上限设定值（-32767~32767）
    iOutLowLimit:= , //输出下限设定值（-32767~32767）
    eDirection:= , //方向设置,0:逆动作，1: 正动作
    eOutputType:= , //输出方式,0:PWM 输出1:数字量输出
    bOutput_PWM=> , // PWM量输出
    iOutput=> , //数字量输出
    bAutoTuneDone=> , //自整定完成状态
    ePIDState=> , //当前PID状态
    rPreI_Item=> , //上次积分项输出值
    rPreD_Item=> , //上次微分项输出值
    rPrePID_Item=> , //上次PID 输出总量
    Error=> , //错误标志
    ErrorId=> );//错误代码
```

By binding the output state of the PID to the Y terminal, it is clear whether the PID instruction is being executed.



PID output.

The PID tuning modes are divided into self-tuning mode and manual PID mode. Before triggering the Enable signal of the command, necessary parameters need to be set as follows: in self-tuning mode, the step response mode only needs to provide the target temperature and current temperature, and other parameters can be set by default; The critical oscillation mode requires the target temperature, current temperature, and sampling period, while other parameters can be set as default; In manual PID mode, it is necessary to set the current temperature, target temperature, sampling period, Kp, Ti, and Td, and the remaining parameters can be set by default.

- Self tuning PID mode example

AutoPid example:

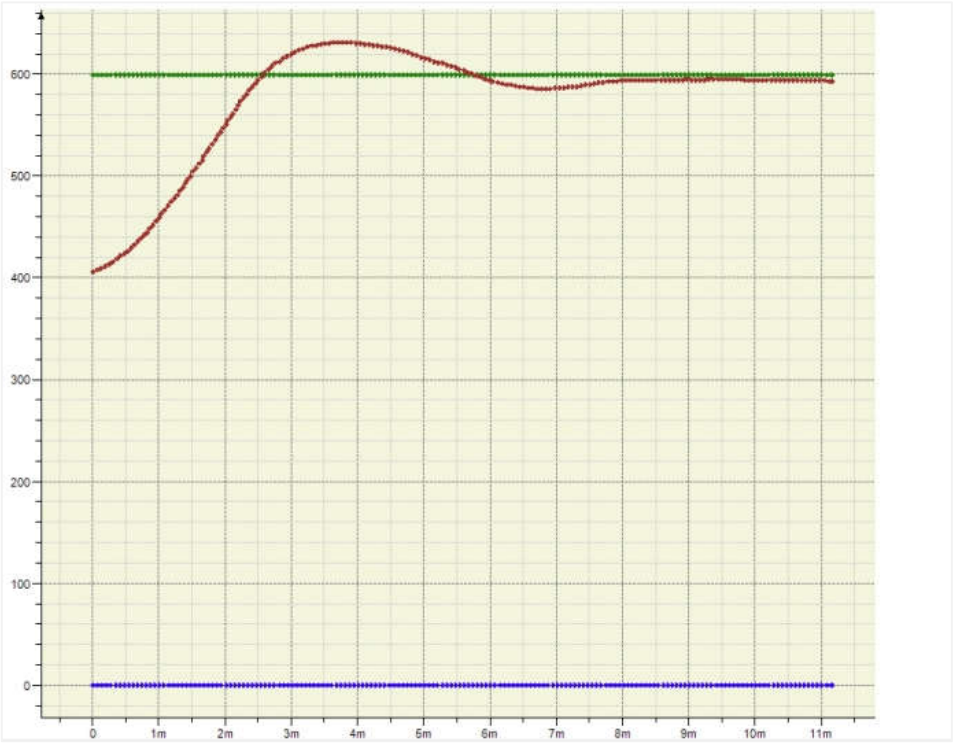
```
PID_1(
    Enable TRUE := bEnable TRUE ,
    rSetValue 300 := SetValue 300 ,
    rCurrentValue 300 := CurrentValue 300 ,
    ePIDMode 1 := Mode AutoTuneMo ,
    ePIDStruct:= ,
    udSmapleTime:= ,
    uiKp 1500 := iKp 1500 ,
    uiTi 1000 := iTi 1000 ,
    uiTd 50 := iTD 50 ,
    uiPIDBound:= ,
    uiDeadBound:= ,
    uiUnitADNum:= ,
    eAutoTuneType:= ,
    eExcessLimit:= ,
    uiIncrement:= ,
    uiStayTimes:= ,
    eAdvancedMode:= ,
    uiFilter:= ,
    uiTdFiltRatio:= ,
    iOutUpLimit:= ,
    iOutLowLimit:= ,
    eDirection:= ,
    eOutputType BoolType := OutputType BoolType ,
    bOutput_PWM=> ,
    iOutput=> ,
    bAutoTuneDone=> ,
    ePIDState=> ,
    rPreI_Item=> ,
    rPreD_Item=> ,
    rPrePID_Item=> ,
    Error=> ,
    ErrorId=> );
```

Successfully tuned waveform diagram:

- Manual PID mode example

```
PID_1(
    Enable TRUE := bEnable TRUE ,
    rSetValue 400 := SetValue 400 ,
    rCurrentValue 410 := CurrentValue 400 ,
    ePIDMode 0 := Mode ManualPIDM ,
    ePIDStruct:= ,
    udSmapleTime:= ,
    uiKp 6523 := iKp 6523 ,
    uiTi 68 := iTi 68 ,
    uiTd 170 := iTD 170 ,
    uiPIDBound:= ,
    uiDeadBound:= ,
    uiUnitADNum:= ,
    eAutoTuneType:= ,
    eExcessLimit:= ,
    uiIncrement:= ,
    uiStayTimes:= ,
    eAdvancedMode:= ,
    uiFilter:= ,
    uiTdFiltRatio:= ,
    iOutUpLimit:= ,
    iOutLowLimit:= ,
    eDirection:= ,
    eOutputType:= ,
    bOutput_PWM=> ,
    iOutput=> ,
    bAutoTuneDone=> ,
    ePIDState=> ,
    rPreI_Item=> ,
    rPreD_Item=> ,
    rPrePID_Item=> ,
    Error=> ,
    ErrorId=> );
```

Successfully tuned waveform diagram:



- Make sure to turn off the PID enable when modifying PID parameters!
- In the case of continuous output, a system whose ability to function gradually weakens as the feedback value continues to increase can be self-tuning, such as temperature or pressure. For flow or liquid level objects, it may not be suitable for self-tuning.
- Under the condition of allowing overshoot, the PID parameters obtained by self-tuning are the optimal parameters of the system.
- Under the premise of not allowing overshoot, the PID parameters obtained from self-tuning depend on the target value, that is, different set target values may result in different PID parameters, and these parameters may not be the optimal parameters of the system, but they can be used as a reference.
- If users are unable to perform self-tuning, they can also rely on certain engineering experience values to manually adjust. However, in actual debugging, appropriate modifications need to be made based on the adjustment effect. Below are several common control system experience values for users' reference:
  - Temperature system: P (%) 2000~6000, I (minutes) 3~10, D (minutes) 0.5~3
  - Traffic system: P (%) 4000~10000, I (minutes) 0.1~1

# 4. System library

## 4-1. Function overview

Users can read or write system parameters of the controller through instructions.

## 4-2. Function block introduction

### 4-2-1. CPU dominant frequency【XJ_GetCPUFrequency】

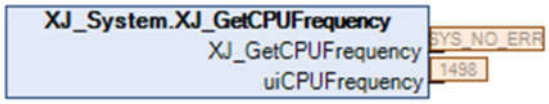
| CPU main frequency [XJ_GetCPUFrequency] | | | |
|---|---|---|---|
| Execution condition | - | Suitable model | XS, XSA |
| Firmware | V1.1.0 and up | Software | XS StudioV2.2.0 and above |
| Library | XJ_System | | |

| Command | Name | Graph representation | ST language |
|---|---|---|---|
| XJ_GetCPUFrequency | CPU dominant frequency | | `XJ_GetCPUFrequency(uiCPUFrequency=> );` |

(1) Output variables

| Output variables | Name | Data type | Effective range | Initial value | Description |
|---|---|---|---|---|---|
| GetCPUFrequency | Error code | SYS_XJ_ERR | - | 0 | Error code |
| CPUFrequency | CPU dominant frequency | UINT | - | 0 | Unit is M |

The execution results are as follows:




### 4-2-2. CPU temperature【XJ_GetCPUTemperature】

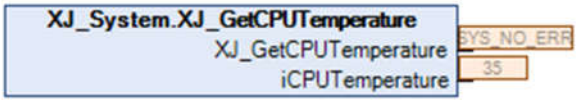
| CPU temperature [XJ_GetCPUTemperature] | | | |
|---|---|---|---|
| Execution condition | - | Suitable model | XS, XSA |
| Firmware | V1.1.0 and up | Software | XS StudioV2.2.0 and above |
| Library | XJ_System | | |

| Command | Name | Graph representation | ST language |
|---|---|---|---|
| XJ_GetCPUTemperature | CPU temperature | | `XJ_GetCPUTemperature(iCPUTemperature=> )` |

(1) Output variables

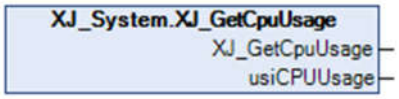
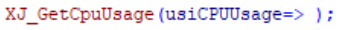
| Output variables | Name | Data type | Effective range | Initial value | Description |
|---|---|---|---|---|---|
| GetCPUTemperature | Error code | SYS_XJ_ERR | - | 0 | Error code |
| CPUTemperature | CPU temperature | INT | - | 0 | CPU temperature |

The execution results are as follows:




## 4-2-3. CPU occupancy【XJ_GetCPUUsage】

| CPU usage [XJ_GetCPUUsage] | | | |
|---|---|---|---|
| Execution condition | - | Suitable model | XS, XSA |
| Firmware | V1.1.0 and up | Software | XS StudioV2.2.0 and above |
| Library | XJ_System | | |

| Command | Name | Graph representation | ST language |
|---|---|---|---|
| XJ_GetCPUUsage | CPU occupancy |  |  |





(1) Output variables

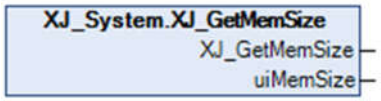
| Output variables | Name | Data type | Effective range | Initial value | Description |
|---|---|---|---|---|---|
| GetCPUUsage | Error code | SYS_XJ_ERR | - | 0 | Error code |
| CPUUsage | CPU occupancy | USINT | - | 0 | Return percentage, 0~100 |

The execution results are as follows:

## 4-2-4. Memory size【XJ_GetMemSize】

| Memory size [XJ_GetMemSize] |
|---|

| | | | |
|---|---|---|---|
| Execution condition | - | Suitable model | XS, XSA |
| Firmware | V1.1.0 and up | Software | XS StudioV2.2.0 and above |
| Library | XJ_System | | |

| Command | Name | Graph representation | ST language |
|---|---|---|---|
| XJ_GetMemSize | Memory size |  | `XJ_GetMemSize(uiMemSize=> );` |

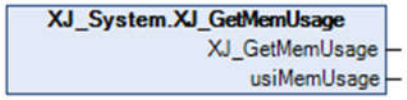
(1) Output variables

| Output variables | Name | Data type | Effective range | Initial value | Description |
|---|---|---|---|---|---|
| GetMemSize | Error code | SYS_XJ_ERR | - | 0 | Error code |
| MemSize | Memory size | UINT | - | 0 | Unit is MB |

The execution results are as follows:

## 4-2-5. Memory occupancy【XJ_GetMemUsage】

| Memory occupancy [XJ_GetMemUsage | | | |
|---|---|---|---|
| Execution condition | - | Suitable model | XS, XSA |
| Firmware | V1.1.0 and up | Software | XS StudioV2.2.0 and above |
| Library | XJ_System | | |

| Command | Name | Graph representation | ST language |
|---|---|---|---|
| XJ_GetMemUsage | Memory occupancy |  | `XJ_GetMemUsage(usiMemUsage=> );` |

(1) Output variables

| Output variables | Name | Data type | Effective range | Initial value | Description |
|---|---|---|---|---|---|
| GetMemUsage | Error code | SYS_XJ_ERR | - | 0 | Error code |
| MemUsage | Memory occupancy | USINT | - | 0 | Return percentage, 0~100 |

The execution results are as follows:

## 4-2-6. On time【XJ_GetBootTime】

| On time [XJ_GetBootTime] | | | |
|---|---|---|---|
| Execution condition | - | Suitable model | XS, XSA |
| Firmware | V1.1.0 and up | Software | XS StudioV2.2.0 and above |

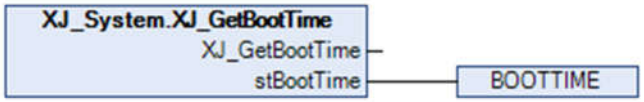
| Library | XJ_System | |
|---|---|---|

| Command | Name | Graph representation | ST language |
|---|---|---|---|
| XJ_GetBootTime | On time | XJ_System.XJ_GetBootTime<br>XJ_GetBootTime<br>stBootTime | XJ_GetBootTime(stBootTime=> ); |

(1) Output variables

| Output variables | Name | Data type | Effective range | Initial value | Description |
|---|---|---|---|---|---|
| GetBootTime | Error code | SYS_XJ_ERR | - | 0 | Error code |
| BootTime | On time | SYS_XJ_TIME | - | 0 | structural morphology |

The execution results are as follows:




```
PROGRAM POU
VAR
    // 获取网口信息
    XJ_GetNetInfo_0: XJ_System.XJ_GetNetInfo;
    PLCID: DWORD;
    BOOTTIME: XJ_SYSTEM.SYS_XJ_TIME;
    AA: BOOL;
    NETID: BYTE;
    XJ_SetNetInfo_0: XJ_System.XJ_SetNetInfo;
    runtimeversion: DWORD;
    STTIME: XJ_SYSTEM.SYS_XJ_TIME;
    timezone: INT;
    SetTIME: XJ_System.SYS_XJ_TIME;
    itimezone: INT;
END_VAR
```

| | | | | | |
|---|---|---|---|---|---|
| BOOTTIME | XJ_SYSTEM.SYS_XJ_TIME | | | | |
| Year | UINT | 2022 | | | 年 |
| Month | UINT | 11 | | | 月 |
| Day | UINT | 19 | | | 日 |
| Hour | UINT | 1 | | | 时 |
| Minute | UINT | 55 | | | 分 |
| Second | UINT | 43 | | | 秒 |
| Milliseconds | UINT | 0 | | | 微妙 |
| DayOfWeek | UINT | 6 | | | 周 |

## 4-2-7. PLC name【XJ_GetPLCName】

| PLC name [XJ_GetPLCName] | | | |
|---|---|---|---|
| Execution condition | - | Suitable model | XS, XSA |
| Firmware | V1.1.0 and up | Software | XS StudioV2.2.0 and above |

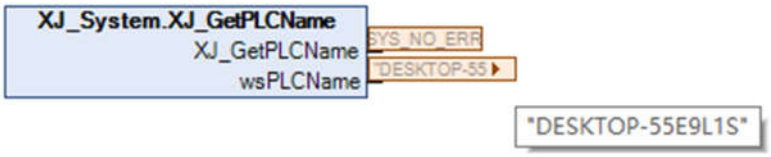
| | | |
|---|---|---|
| Library | XJ_System | |

| Command | Name | Graph representation | ST language |
|---|---|---|---|
| XJ_GetPLCName | PLC name | XJ_System.XJ_GetPLCName<br>XJ_GetPLCName<br>wsPLCName | XJ_GetPLCName(wsPLCName=> ); |

(1) Output variables

| Output variables | Name | Data type | Effective range | Initial value | Description |
|---|---|---|---|---|---|
| GetPLCName | Error code | SYS_XJ_ERR | - | 0 | Error code |
| PLCName | PLC name | WSTRING | - | 0 | PLC name |

The execution results are as follows:




## 4-2-8. PLCID【XJ_GetPLCID】

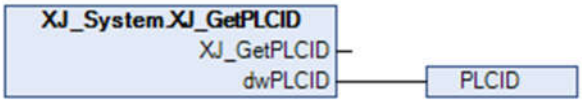
| PLCID [XJ_GetPLCID] | | | |
|---|---|---|---|
| Execution condition | - | Suitable model | XS, XSA |
| Firmware | V1.1.0 and up | Software | XS StudioV2.2.0 and above |
| Library | XJ_System | | |

| Command | Name | Graph representation | ST language |
|---|---|---|---|
| XJ_GetPLCID | PLCID | XJ_System.XJ_GetPLCID<br>XJ_GetPLCID<br>dwPLCID | XJ_GetPLCID(dwPLCID=> ); |

(1) Output variables

| Output variables | Name | Data type | Effective range | Initial value | Description |
|---|---|---|---|---|---|
| GetPLCID | Error code | SYS_XJ_ERR | - | 0 | Error code |
| PLCID | PLCID | DWORD | - | 0 | PLCID information, hexadecimal display |

The execution result is as follows, and the PLCID information is displayed in hexadecimal:



```
PROGRAM POU
VAR
    // 获取网口信息
    XJ_GetNetInfo_0: XJ_System.XJ_GetNetInfo;
    PLCID: DWORD;
    BOOTTIME: XJ_SYSTEM.SYS_XJ_TIME;
    AA: BOOL;
    NETID: BYTE;
    XJ_SetNetInfo_0: XJ_System.XJ_SetNetInfo;
    runtimeversion: DWORD;
    STTIME: XJ_SYSTEM.SYS_XJ_TIME;
    timezone: INT;
    SetTIME: XJ_System.SYS_XJ_TIME;
    itimezone: INT;
END_VAR
```

| 表达式 | 类型 | 值 | 准 |
|---|---|---|---|
| XJ_GetNetInfo_0 | XJ_System.XJ_GetNetInfo | | |
| PLCID | DWORD | 16#17070200 | |
| BOOTTIME | XJ_SYSTEM.SYS_XJ_TIME | | |
| AA | BOOL | FALSE | |
| NETID | BYTE | 16#00 | |
| XJ_SetNetInfo_0 | XJ_System.XJ_SetNetInfo | | |
| runtimeversion | DWORD | 16#03050F28 | |
| STTIME | XJ_SYSTEM.SYS_XJ_TIME | | |
| timezone | INT | 16#0000 | |
| SetTIME | XJ_System.SYS_XJ_TIME | | |
| itimezone | INT | 16#0000 | |

## 4-2-9. Firmware version【XJ_GetPLCVersion】

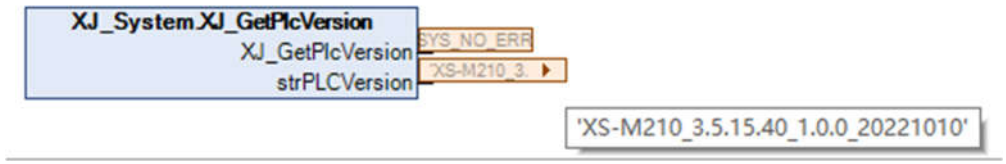
| Firmware version [XJ_GetPLCVersion] | | | |
|---|---|---|---|
| Execution condition | - | Suitable model | XS, XSA |
| Firmware | V1.1.0 and up | Software | XS StudioV2.2.0 and above |
| Library | XJ_System | | |

| Command | Name | Graph representation | ST language |
|---|---|---|---|
| XJ_GetPLCVersion | PLC firmware version | XJ_System.XJ_GetPlcVersion<br>XJ_GetPlcVersion<br>strPLCVersion | XJ_GetPlcVersion(strPLCVersion=> ); |

(1) Output variables

| Output variables | Name | Data type | Effective range | Initial value | Description |
|---|---|---|---|---|---|
| GetPLCVersion | Error code | SYS_XJ_ERR | - | 0 | Error code |
| PLCVersion | PLC firmware version | STRING | - | 0 | PLC firmware version |

The execution result is as follows:




## 4-2-10. Obtain network port information【XJ_GetNetInfo】

| Obtain network port information [XJ_GetNetInfo] | | | |
|---|---|---|---|
| Execution condition | - | Suitable model | XS, XSA |
| Firmware | V1.1.0 and up | Software | XS StudioV2.2.0 and above |
| Library | XJ_System | | |

| Command | Name | Graph representation | ST language |
|---|---|---|---|
| XJ_GetNetInfo | Obtain network port information | XJ_GetNetInfo_0<br>XJ_System.XJ_GetNetInfo<br>xEnable, byEtherID<br>xValid, xBusy, xError, eErrorID, strIPAddress, strMAC, strNetmask, strGateway, xDHCP | XJ_GetNetInfo_0(<br>xEnable:= ,<br>byEtherID:= ,<br>xValid=> ,<br>xBusy=> ,<br>xError=> ,<br>eErrorID=> ,<br>strIPAddress=> ,<br>strMAC=> ,<br>strNetmask=> ,<br>strGateway=> ,<br>xDHCP=> ); |

(1) Input variables

| Input variables | Name | Data type | Effective range | Initial value | Description |
|---|---|---|---|---|---|
| Enable | Enable | BOOL | TRUE,FALSE | FALSE | Enable |
| EtherID | Network port ID | BYTE | - | 0 | Corresponding network port no. on the display screen |

(2) Output variables

| Output variables | Name | Data type | Effective range | Initial value | Description |
|---|---|---|---|---|---|
| Valid | Output valid | BOOL | TRUE,FALSE | FALSE | Output valid |
| Busy | Executing | BOOL | TRUE,FALSE | FALSE | Executing |
| Error | Error flag | BOOL | TRUE,FALSE | FALSE | Error flag |
| ErrorID | Error code | SYS_XJ_ERR | - | 0 | Error code |
| IPAddress | IP address | STRING | - | 0 | IP address |
| MAC | MAC address | STRING | - | 0 | MAC address |
| Netmask | Subnet address | STRING | - | 0 | Subnet address |
| Gateway | Gateway Information | STRING | - | 0 | Gateway Information |
| DHCP | DHCP properties | BOOL | TRUE，FALSE | FALSE | Is it automatic property acquisition |

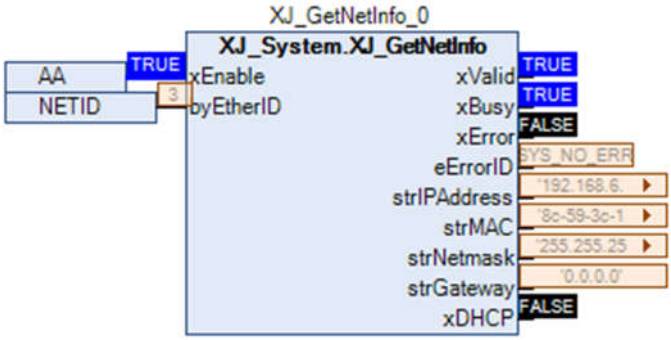
The network port IDs of the industrial computer are arranged in 0-3 order from left to right in the network adapter,

as shown in the figure Ethernet1, and the IDs are written in 2.

The definition of XSA-L, also known as the XSA industrial computer network port ID for Linux systems, is that the network port ID near the DP port is 0, increasing in order of 0, 1, 2, 3, 4, and 5.

```
PROGRAM POU
VAR
    // 获取网口信息
    XJ_GetNetInfo_0: XJ_System.XJ_GetNetInfo;
    PLCID: DWORD;
    BOOTTIME: XJ_SYSTEM.SYS_XJ_TIME;
    AA: BOOL;
    NETID: BYTE;
END_VAR
```

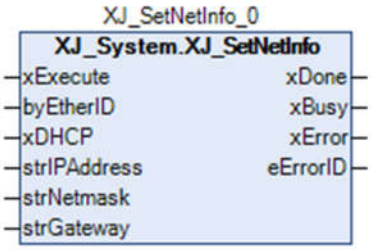
The execution result is as follows：




| | | | | |
|---|---|---|---|---|
| XJ_GetNetInfo_0 | XJ_System.XJ_GetNetInfo | | | 获取网口信息 |
| xEnable | BOOL | TRUE | | 功能块使能 |
| byEtherID | BYTE | 3 | | 网口 Id |
| xValid | BOOL | TRUE | | 输出有效 |
| xBusy | BOOL | TRUE | | 正在执行 |
| xError | BOOL | FALSE | | 错误 |
| eErrorID | SYS_XJ_ERR | SYS_NO_ERR | | 错误 ID |
| strIPAddress | STRING(17) | '192.168.6.6' | | IP地址 |
| strMAC | STRING(17) | '8c-59-3c-10-e9-... | | MAC地址 |
| strNetmask | STRING(17) | '255.255.255.0' | | 子网地址 |
| strGateway | STRING(17) | '0.0.0.0' | | 网关信息 |
| xDHCP | BOOL | FALSE | | 是否为自动获取属性 |

## 4-2-11. Set network port information【XJ_SetNetInfo】

| Set network port information [XJ_SetNetInfo] | | | |
|---|---|---|---|
| Execution condition | - | Suitable model | XS, XSA |
| Firmware | V1.1.0 and up | Software | XS StudioV2.2.0 and above |
| Library | XJ_System | | |

| Command | Name | Graph representation | ST language |
|---|---|---|---|
| XJ_SetNetInfo | Set network port information |  | XJ_SetNetInfo_0(<br>xExecute:= ,<br>byEtherID:= ,<br>xDHCP:= ,<br>strIPAddress:= ,<br>strNetmask:= ,<br>strGateway:= ,<br>xDone=> ,<br>xBusy=> ,<br>xError=> ,<br>eErrorID=> ); |

(1) Input variables

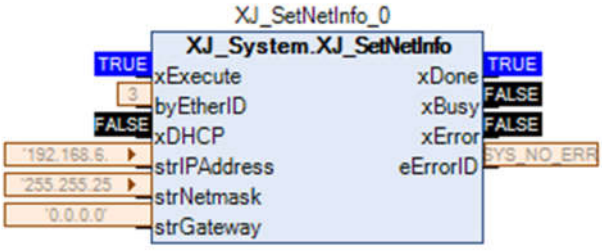
| Input variables | Name | Data type | Effective range | Initial value | Description |
|---|---|---|---|---|---|
| Excute | Excute | BOOL | TRUE,FALSE | FALSE | Execute the function block at the rising edge |
| EtherID | Network port ID | BYTE | - | 0 | Corresponding network port on the display screen |
| DPCH | DPCH property | BOOL | TRUE,FALSE | FALSE | Is it automatic property acquisition |
| IPAddress | IP address | STRING | - | 0 | IP address |
| Netmask | Subnet address | STRING | - | 0 | Subnet address |
| Gateway | Gateway information | STRING | - | 0 | Gateway information |

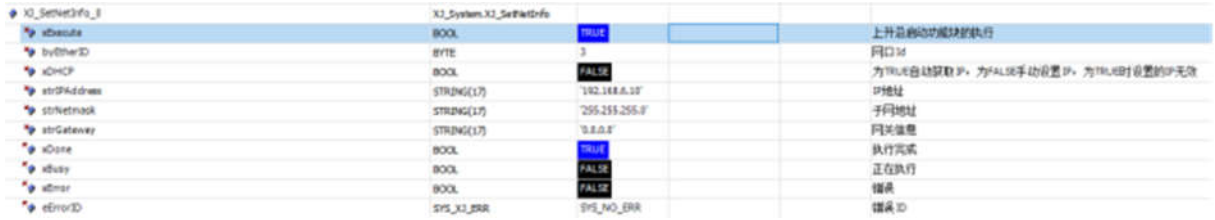
(2) Output variables

| Output variables | Name | Data type | Effective range | Initial value | Description |
|---|---|---|---|---|---|
| Done | Execution completed | BOOL | TRUE,FALSE | FALSE | Execution completed |
| Busy | Executing | BOOL | TRUE,FALSE | FALSE | Executing |
| Error | Error flag | BOOL | TRUE,FALSE | FALSE | Error flag |
| ErrorID | Error code | SYS_XJ_ERR | - | 0 | Error code |

The modification will only take effect after powering off and restarting the PLC or RTE.

The definition of XSA-L, also known as the XSA industrial computer network port ID for Linux systems, is that the network port ID near the DP port is 0, increasing in order of 0, 1, 2, 3, 4, and 5.





## 4-2-12. Runtime version【XJ_GetRuntimeVersion】

| Runtime version [XJ_GetRuntimeVersion] | | | |
|---|---|---|---|
| Execution condition | - | Suitable model | XS, XSA |
| Firmware | V1.1.0 and up | Software | XS StudioV2.2.0 and above |
| Library | XJ_System | | |

| Command | Name | Graph representation | ST language |
|---|---|---|---|
| XJ_GetRuntimeVersion | Runtime version | XJ_System.XJ_GetRuntimeVersion<br>XJ_GetRuntimeVersion<br>dwRuntimeVersion | XJ_GetRuntimeVersion(dwRuntimeVersion=> ); |

(1) Output variables

| Output variables | Name | Data type | Effective range | Initial value | Description |
|---|---|---|---|---|---|
| GetRuntimeVersion | Error code | SYS_XJ_ERR | - | 0 | Error code |
| RuntimeVersion | Runtim version | DWORD | - | 0 | Runtim version |

The runtime version is viewed in hexadecimal.

## 4-2-13. BOOT version【XJ_GetBootVersion】

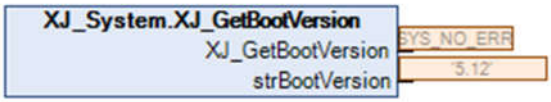
| BOOT version [XJ_GetBootVersion] | | | |
|---|---|---|---|
| Execution condition | - | Suitable model | XS, XSA |
| Firmware | V1.1.0 and up | Software | XS StudioV2.2.0 and above |
| Library | XJ_System | | |

| Command | Name | Graph representation | ST language |
|---|---|---|---|
| XJ_GetBootVersion | BOOT version | | `XJ_GetBootVersion(strBootVersion=> );` |

(1) Output variables

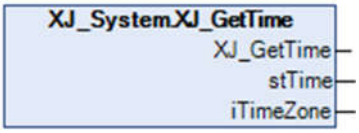
| Output variables | Name | Data type | Effective range | Initial value | Description |
|---|---|---|---|---|---|
| GetBootVersion | Error code | SYS_XJ_ERR | - | 0 | Error code |
| BootVersion | BOOT version | STRNG | - | 0 | BOOT version |

The execution results are as follows:




## 4-2-14. Obtain date and time【XJ_GetTime】

| Obtain date and time [XJ_GetTime] | | | |
|---|---|---|---|
| Execution condition | Edge triggering | Suitable model | XS, XSA |
| Firmware | V1.1.0 and up | Software | XS StudioV2.2.0 and above |
| Library | XJ_System | | |

| Command | Name | Graph representation | ST language |
|---|---|---|---|
| XJ_GetTime | Obtain date and time |  | `XJ_GetTime(stTime=> , iTimeZone=> )` |

(1) Output variables

| Output variables | Name | Data type | Effective range | Initial value | Description |
|---|---|---|---|---|---|
| GetTime | Error code | SYS_XJ_ERR | - | 0 | Error code |
| Time | Time | SYS_XJ_TIME | - | 0 | Structural morphology |
| TimeZone | Get time zone | INT | - | - | Unit is minute |

The execution results are as follows:

```
PROGRAM POU
VAR
    // 获取网口信息
    XJ_GetNetInfo_0: XJ_System.XJ_GetNetInfo;
    PLCID: DWORD;
    BOOTTIME: XJ_SYSTEM.SYS_XJ_TIME;
    AA: BOOL;
    NETID: BYTE;
    XJ_SetNetInfo_0: XJ_System.XJ_SetNetInfo;
    runtimeversion: DWORD;
    STTIME: XJ_SYSTEM.SYS_XJ_TIME;
    timezone: INT;
    SetTIME: XJ_System.SYS_XJ_TIME;
    itimezone: INT;
END_VAR
```

| | | | | | |
|---|---|---|---|---|---|
| STTIME | XJ_SYSTEM.SYS_XJ_TIME | | | | |
| Year | UINT | 2022 | | | 年 |
| Month | UINT | 11 | | | 月 |
| Day | UINT | 18 | | | 日 |
| Hour | UINT | 13 | | | 时 |
| Minute | UINT | 9 | | | 分 |
| Second | UINT | 8 | | | 秒 |
| Milliseconds | UINT | 909 | | | 微秒 |
| DayOfWeek | UINT | 5 | | | 周 |

## 4-2-15. Set date and time【XJ_SetTime】

| Set date and time [XJ_SetTime] | | | |
|---|---|---|---|
| Execution condition | Edge triggering | Suitable model | XS, XSA |
| Firmware | V1.1.0 and up | Software | XS StudioV2.2.0 and above |
| Library | XJ_System | | |

| Command | Name | Graph representation | ST language |
|---|---|---|---|
| XJ_SetTime | Set date and time | XJ_SystemXJ_SetTime<br>stTime<br>iTimeZone<br>XJ_SetTime | XJ_SetTime(stTime:= , iTimeZone:= ); |

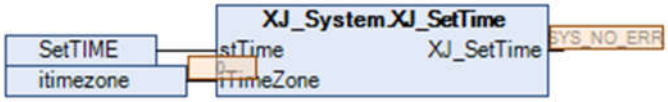
(1) Input variables

| Input variables | Name | Data type | Effective range | Initial value | Description |
|---|---|---|---|---|---|
| Time | Set time | SYS_XJ_TIME | - | 0 | Set time |
| TimeZone | Set time zone | INT | - | -480 | Unit is minute |

(2) Output variables

| Output variables | Name | Data type | Effective range | Initial value | Description |
|---|---|---|---|---|---|
| SetTime | Error code | SYS_XJ_ERR | - | 0 | Error code |

The change in time takes effect immediately after writing. Due to the lack of conduction conditions, the time remains unchanged after writing the time. The execution result is as follows:




```
PROGRAM POU
VAR
    // 获取网口信息
    XJ_GetNetInfo_0: XJ_System.XJ_GetNetInfo;
    PLCID: DWORD;
    BOOTTIME: XJ_SYSTEM.SYS_XJ_TIME;
    AA: BOOL;
    NETID: BYTE;
    XJ_SetNetInfo_0: XJ_System.XJ_SetNetInfo;
    STTIME: XJ_SYSTEM.SYS_XJ_TIME;
    runtimeversion: DWORD;
    timezone: INT;
    SetTIME: XJ_System.SYS_XJ_TIME;
    itimezone: INT;
END_VAR
```

| | | | | | |
|---|---|---|---|---|---|
| SetTIME | XJ_System.SYS_XJ_TIME | | | | |
| Year | UINT | 2022 | | | 年 |
| Month | UINT | 11 | | | 月 |
| Day | UINT | 19 | | | 日 |
| Hour | UINT | 13 | | | 时 |
| Minute | UINT | 30 | | | 分 |
| Second | UINT | 10 | | | 秒 |
| Milliseconds | UINT | 855 | | | 微秒 |
| DayOfWeek | UINT | 5 | | | 周 |

The obtained time is consistent with the set time.

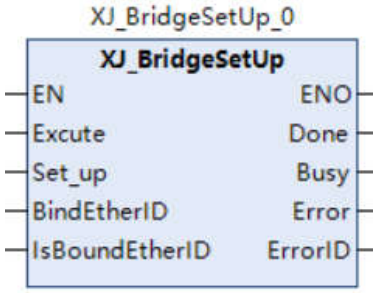
| | | | | | |
|---|---|---|---|---|---|
| STTIME | XJ_SYSTEM.SYS_XJ_TIME | | | | |
| Year | UINT | 2022 | | | 年 |
| Month | UINT | 11 | | | 月 |
| Day | UINT | 19 | | | 日 |
| Hour | UINT | 13 | | | 时 |
| Minute | UINT | 30 | | | 分 |
| Second | UINT | 11 | | | 秒 |
| Milliseconds | UINT | 938 | | | 微秒 |
| DayOfWeek | UINT | 6 | | | 周 |

## 4-2-16. Network port bridging instruction

(1) Instruction overview

XSF5's network port 1 "ENET1 | | ECAT1" supports bridging function, and shares network port 0 "ENET0" information in bridging mode. At this time, "ENET0"&"ENET1 | | ECAT1" are switch functions (similar to XD5E's two network ports).

| Bridge instruction [XJ_BridgeSetUp] | | | |
|---|---|---|---|
| Execution condition | Edge triggering | Suitable model | XSF |
| Firmware | V2.4.1 and above | Software | XS StudioV2.3.1 and above |
| Library | XJ_System | | |

| Command | Name | Graph representation | ST language |
|---|---|---|---|
| XJ_BridgeSetUp | Network port bridging |  | XJ_BridgeSetUp_0(<br>Excute:= ,<br>Set_up:= ,<br>BindEtherID:= ,<br>IsBoundEtherID:= ,<br>Done=> ,<br>Busy=> ,<br>Error=> ,<br>ErrorID=> ); |

(2) Related variables

| VAR_INPUT | Name | Data type | Effective range | Initial value | Description |
|---|---|---|---|---|---|
| Excute | Execute | BOOL | FALSE/TRUE | FALSE | Start the function block at the rising edge |
| Set_up | Set | BYTE | - | 0 | 0: not bind<br>1: bind |
| BindEtherID | Bind network port ID | BYTE | - | 0 | Bind network port ID |
| IsBoundEtherID | To be Bound network port ID | BYTE | 1~30 | xinje | To be Bound network port ID |

| VAR_OUTPUT | Name | Data type | Effective range | Initial value | Description |
|---|---|---|---|---|---|
| Done | Execution completed | BOOL | FALSE/TRUE | FALSE | Execution completed |
| Busy | Executing | BOOL | FALSE/TRUE | FALSE | Executing |
| Error | Error flag | BOOL | FALSE/TRUE | FALSE | Error flag |
| ErrorID | Error code | SYS_XJ_ERR | - | - | Error code |

(3) Execution result

The instruction parameter settings and execution results are as follows, which has the same effect as directly

changing network port 1 to bridge in the "Device" - "System Settings" interface.

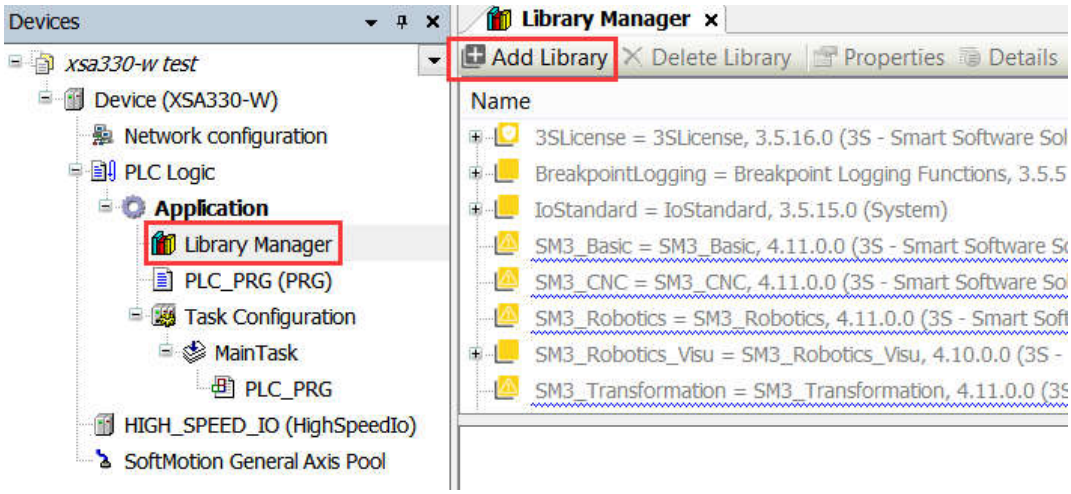
| 表达式 | 类型 | 值 | 准备值 | 地址 | 注释 |
|---|---|---|---|---|---|
| XJ_BridgeSetUp_0 | XJ_BridgeSetUp | | | | |
| Excute | BOOL | TRUE | | | 上升沿启动功能块执行 |
| Set_up | BYTE | 1 | | | 0:不绑定 1:绑定 |
| BindEtherID | BYTE | 0 | | | 绑定网口号 |
| IsBoundEtherID | BYTE | 1 | | | 被绑定的网口号 |
| Done | BOOL | TRUE | | | 执行完成 |
| Busy | BOOL | FALSE | | | 正在执行 |
| Error | BOOL | FALSE | | | 错误标志 |
| ErrorID | SYS_XJ_ERR | SYS_NO_ERR | | | 错误代码 |
| Edge | BOOL | TRUE | | | |

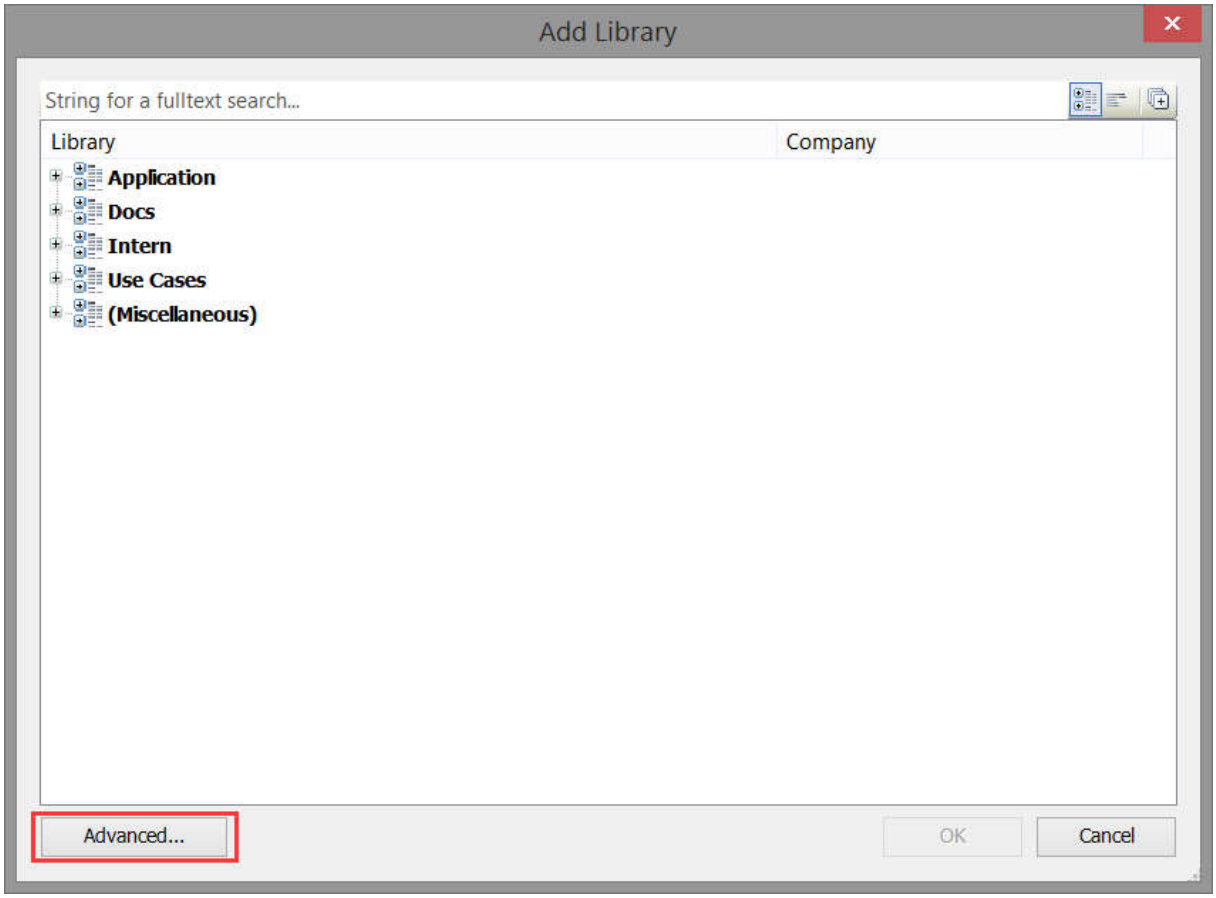
Before execution:

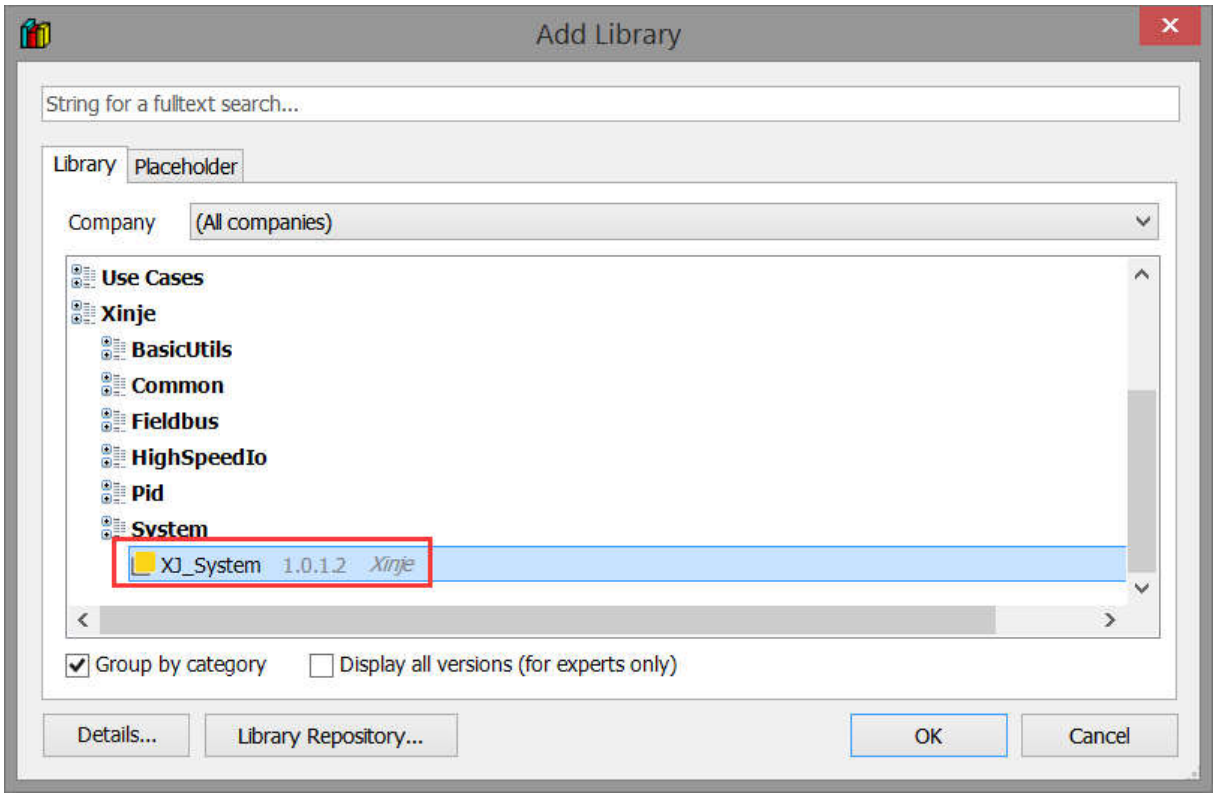
After execution:

## 4-3. Parameter configuration

Add Library File: Library Manager - Add Library - Advanced - Add XJ_System.




Add Library
String for a fulltext search...
Library
Placeholder
Company
(All companies)
Use Cases
Xinje
BasicUtils
Common
Fieldbus
HighSpeedIo
Pid
System
XJ_System 1.0.1.2 Xinje
Group by category
Display all versions (for experts only)
Details...
Library Repository...
OK
Cancel

# 5. Remote IO module read and write

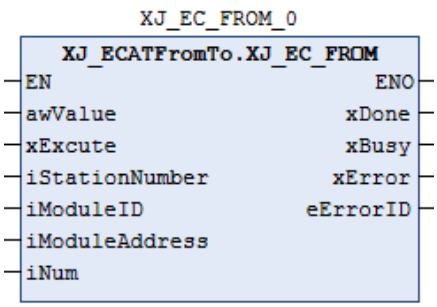
## 5-1. Function overview

Used for reading and writing module parameters on remote IO, for temperature modules and weighing modules.

## 5-2. Function block introduction

### 5-2-1. Remote IO read【XJ_ECATFromTo.XJ_EC_FROM】

| Remote IO read [XJ_ECATFromTo.XJ_EC_FROM] | | | |
|---|---|---|---|
| Execution condition | Rising edge triggering | Suitable model | XS, XSA |
| Firmware | - | Software | XS StudioV2.2.0 and above |
| Library | XJ_ECATFromTo | | |

| Command | Name | Graphical representation | ST language |
|---|---|---|---|
| XJ_ECATFromTo.XJ_EC_FROM | Remote IO read |  | XJ_ECATFromTo.XJ_EC_FROM_0(<br>awValue:= ,<br>xExcute:= ,<br>iStationNumber:= ,<br>iModuleID:= ,<br>iModuleAddress:= ,<br>iNum:= ,<br>xDone=> ,<br>xBusy=> ,<br>xError=> ,<br>eErrorID=> ); |

(1) Input variables

| VAR_INPUT | Name | Data type | Effective range | Initial value | Description |
|---|---|---|---|---|---|
| awValue | Read value | POINTER TO WORD | - | - | The longest array length [0..100] |
| xExcute | Rising edge trigger | BOOL | - | - | |
| iStationNumber | Station no. | UINT | | - | ECAT station no., such as 1001 |
| iModuleID | Module no. | INT | 0-15 | - | One remote IO with a maximum of 16 modules |
| iModuleAddress | Module address | DWORD | - | - | Refer to the module address in the expansion module manual |
| iNum | Register numbers | INT | - | - | Number of consecutive Word reads |

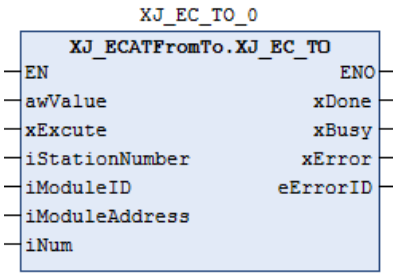
(2) Output variables

| VAR_OUTPUT | Name | Data type | Effective range | Initial value | Description |
|---|---|---|---|---|---|
| xDone | Boolean output | BOOL | TRUE,FALSE | FALSE | Read completed is TRUE |
| xBusy | Boolean output | BOOL | TRUE,FALSE | FALSE | Reading is TRUE |
| xError | Boolean output | BOOL | TRUE,FALSE | FALSE | True when an exception occurs |
| eErrorID | Present mode | EC_FT_ERRID | - | - | When normal, the value is 0. When an exception |

| VAR_OUTPUT | Name | Data type | Effective range | Initial value | Description |
|---|---|---|---|---|---|
| | | | | | occurs, an error code is output |

## 5-2-2. Remote IO write【XJ_ECATFromTo.XJ_EC_TO】

| Remote IO write [XJ_ECATFromTo.XJ_EC_TO] | | | |
|---|---|---|---|
| Execution condition | Rising edge triggering | Suitable model | XS, XSA |
| Firmware | - | Software | XS StudioV2.2.0 and above |
| Library | XJ_ECATFromTo | | |

| Command | Name | Graphical representation | ST language |
|---|---|---|---|
| XJ_ECATFromTo.XJ_EC_TO | Remote IO write |  | XJ_ECATFromTo.XJ_EC_TO_0(<br>awValue:= ,<br>xExcute:= ,<br>iStationNumber:= ,<br>iModuleID:= ,<br>iModuleAddress:= ,<br>iNum:= ,<br>xDone=> ,<br>xBusy=> ,<br>xError=> ,<br>eErrorID=> ); |

(1) Input variables

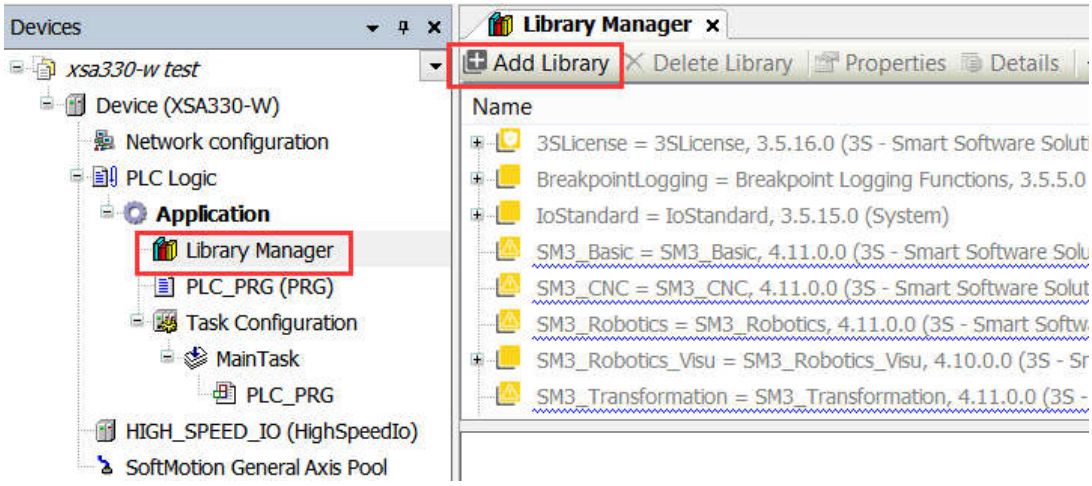
| VAR_INPUT | Name | Data type | Effective range | Initial value | Description |
|---|---|---|---|---|---|
| awValue | write in value | POINTER TO WORD | - | - | The longest array length [0..100] |
| xExcute | Rising edge trigger | BOOL | - | - | |
| iStationNumber | Station no. | UINT | | - | ECAT station no., such as 1001 |
| iModuleID | Module no. | INT | 0-15 | - | One remote IO with a maximum of 16 modules |
| iModuleAddress | Module address | DWORD | - | - | Refer to the module address in the expansion module manual |
| iNum | Register numbers | INT | - | - | Number of consecutive Word writes |

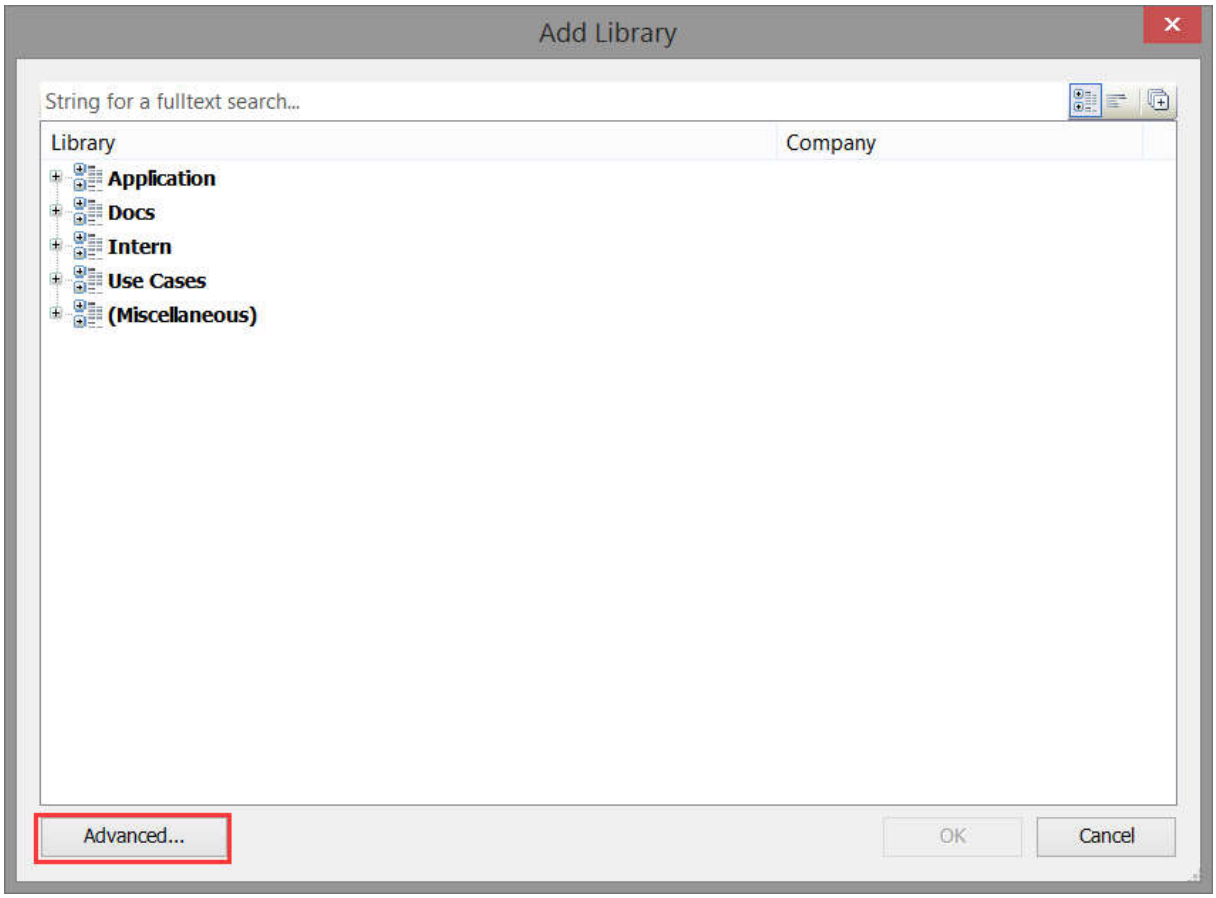
(2) Output variables

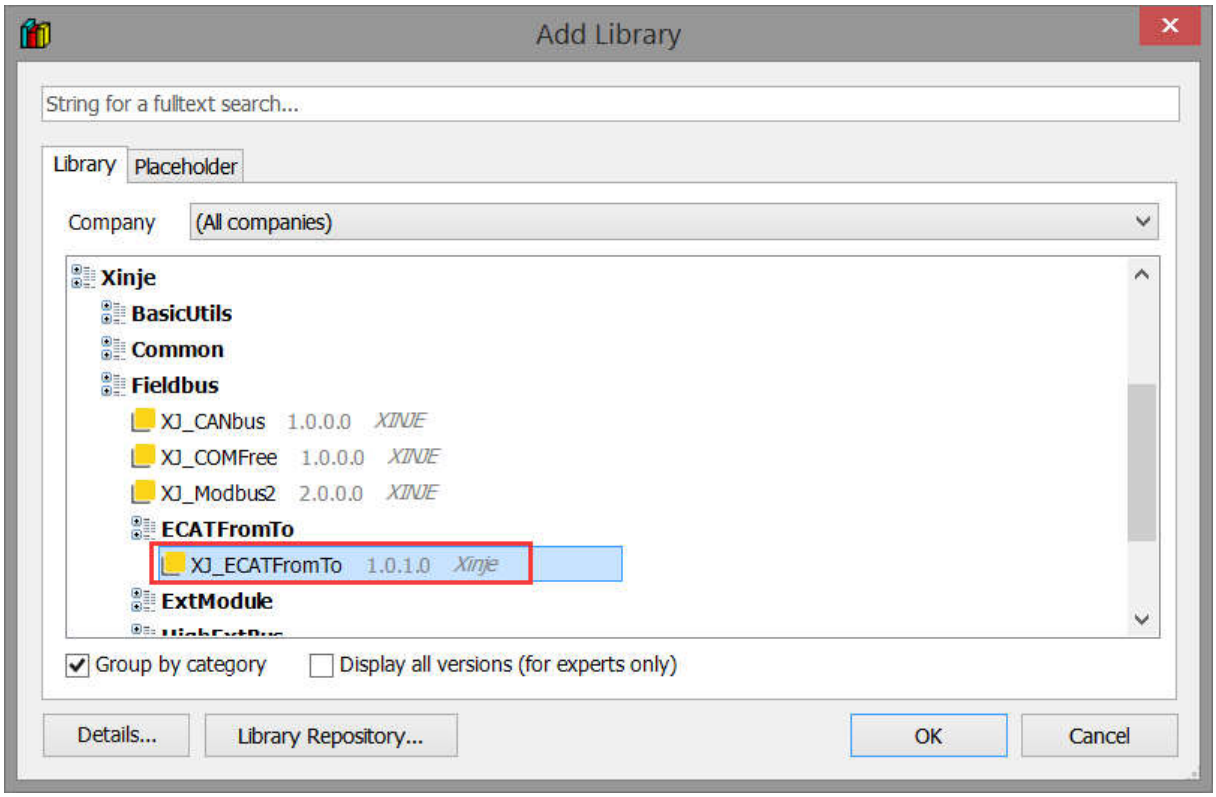
| VAR_OUTPUT | Name | Data type | Effective range | Initial value | Description |
|---|---|---|---|---|---|
| xDone | Boolean output | BOOL | TRUE,FALSE | FALSE | Write completed is TRUE |
| xBusy | Boolean output | BOOL | TRUE,FALSE | FALSE | Writing is TRUE |
| xError | Boolean output | BOOL | TRUE,FALSE | FALSE | True when an exception occurs |

| | | | | | |
|---|---|---|---|---|---|
| eErrorID | Present mode | EC_FT_ERRID | - | - | When normal, the value is 0. When an exception occurs, an error code is output |

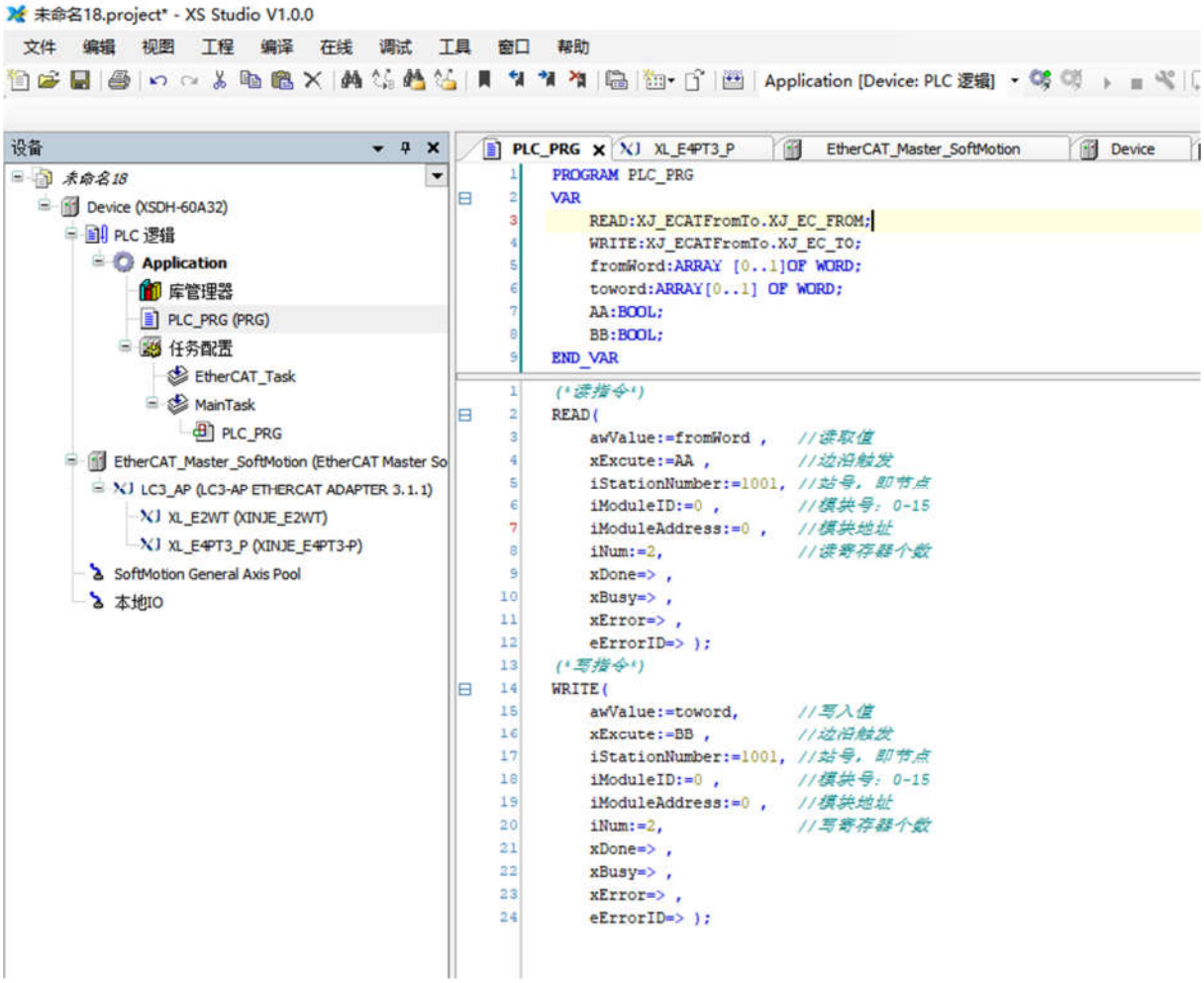
5-3. Parameter configuration




Add Library
String for a fulltext search...
Library
Placeholder
Company
(All companies)
Xinje
BasicUtils
Common
Fieldbus
XJ_CANbus 1.0.0.0 XINJE
XJ_COMFree 1.0.0.0 XINJE
XJ_Modbus2 2.0.0.0 XINJE
ECATFromTo
XJ_ECATFromTo 1.0.1.0 Xinje
ExtModule
Group by category
Display all versions (for experts only)
Details...
Library Repository...
OK
Cancel

5-4. Application



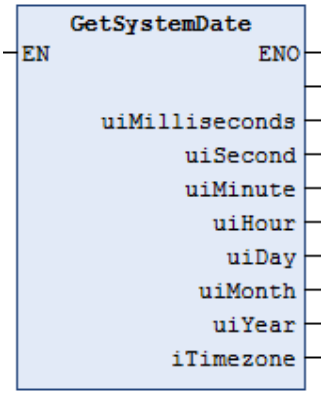
# 6. Date time

## 6-1. Obtain the current system clock GetSystemDate

(1) Instruction overview

Get the current system's time zone, date, and time.

| Obtain the current system clock [GetSystemDate] | | | |
|---|---|---|---|
| Execution condition | - | Suitable model | XS, XSA models |
| Firmware | XS: V2.2.0 and up<br>XSA: - | Software | XS StudioV2.2.0 and above |
| Library | XJ_FunctionUtils | | |

| Command | Name | Graphical representation | ST language |
|---|---|---|---|
| GetSystemDate | Obtain the current system clock |  | GetSystemDate(<br>uiMilliseconds=> ,<br>uiSecond=> ,<br>uiMinute=> ,<br>uiHour=> ,<br>uiDay=> ,<br>uiMonth=> ,<br>uiYear=> ,<br>iTimezone=> ); |

(2) Related variables

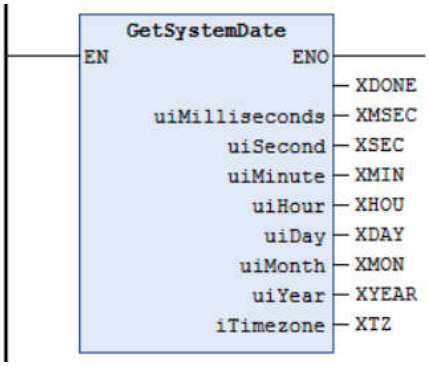
| VAR_OUTPUT | Name | Data type | Effective range | Initial value | Description |
|---|---|---|---|---|---|
| uiMilliseconds | Millisecond | UINT | - | - | Millisecond |
| uiSecond | Second | UINT | - | - | Second |
| uiMinute | Minute | UINT | - | - | Minute |
| uiHour | Hour | UINT | - | - | Hour |
| uiDay | Day | UINT | - | - | Day |
| uiMonth | Month | UINT | - | - | Month |
| uiYear | Year | UINT | - | - | Year |
| iTimezone | Time zone | INT | - | - | Time zone |

(3) Program example

<table>
<tr><th>DEMO</th><th>LD</th><th>ST</th></tr>
<tr><td>Define variables</td><td colspan="2">

```
VAR
   XDONE: BOOL;
   XMSEC: UINT;
   XSEC: UINT;
   XMIN: UINT;
   XHOU: UINT;
   XDAY: UINT;
   XMON: UINT;
   XYEAR: UINT;
   XTZ: INT;
END_VAR
```

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




</td><td>

```
XDONE:= GetSystemDate(
   uiMilliseconds=> XMSEC,
   uiSecond=> XSEC,
   uiMinute=> XMIN,
   uiHour=> XHOU,
   uiDay=> XDAY,
   uiMonth=> XMON,
   uiYear=> XYEAR,
   iTimezone=> XTZ);
```

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

| 表达式 | 类型 | 值 |
|---|---|---|
| XDONE | BOOL | TRUE |
| XMSEC | UINT | 358 |
| XSEC | UINT | 39 |
| XMIN | UINT | 11 |
| XHOU | UINT | 14 |
| XDAY | UINT | 7 |
| XMON | UINT | 9 |
| XYEAR | UINT | 2023 |
| XTZ | INT | -480 |

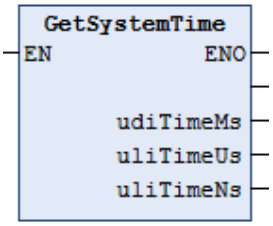
</td></tr>
</table>

## 6-2. Obtain system runtime GetSystemTime

(1) Instruction overview

Obtain the running time after startup.

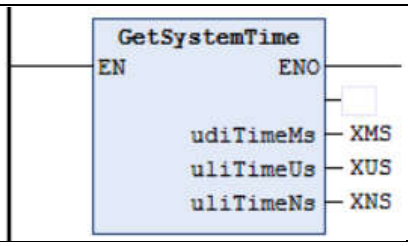
| Obtain system runtime [GetSystemTime] | | | |
|---|---|---|---|
| Execution condition | - | Suitable model | XS, XSA models |
| Firmware | - | Software | XS StudioV2.2.0 and above |
| Library | XJ_FunctionUtils | | |

| Command | Name | Graphical representation | ST language |
|---|---|---|---|
| GetSystemTime | Obtain system runtime |  | GetSystemTime(<br>udiTimeMs=> ,<br>uliTimeUs=> ,<br>uliTimeNs=> ); |

(2) Related variables

| VAR_OUTPUT | Name | Data type | Effective range | Initial value | Description |
|---|---|---|---|---|---|
| udiTimeMs | Runtime ms | UDINT | - | - | Runtime ms |
| uliTimeUs | Runtime us | ULINT | - | - | Runtime us |
| uliTimeNs | Runtime ns | ULINT | - | - | Runtime ns |

(3) Program example

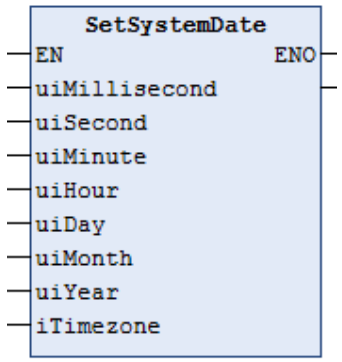
| DEMO | LD | ST |
|---|---|---|
| Define variables | | VAR<br>XMS: UDINT;<br>XUS: ULINT;<br>XNS: ULINT;<br>END_VAR |
| Program example |  | GetSystemTime(<br>udiTimeMs=> XMS,<br>uliTimeUs=> XUS,<br>uliTimeNs=> XNS); |
| Operation result | | |

## 6-3. Set the current system clock SetSystemDate

(1) Instruction overview

Set the current system's time zone, date, and time.

| Set the current system clock [SetSystemDate] | | | |
|---|---|---|---|
| Execution condition | Edge triggering | Suitable model | XS, XSA models |
| Firmware | XS: V2.2.0 and up<br>XSA: - | Software | XS StudioV2.2.0 and above |
| Library | XJ_FunctionUtils | | |

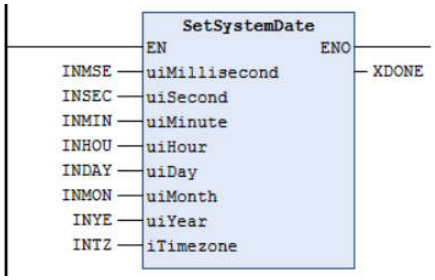
| Command | Name | Graphical representation | ST language |
|---|---|---|---|
| SetSystemDate | Set the current system clock |  | SetSystemDate(<br>uiMillisecond:= ,<br>uiSecond:= ,<br>uiMinute:= ,<br>uiHour:= ,<br>uiDay:= ,<br>uiMonth:= ,<br>uiYear:= ,<br>iTimezone:= ); |

(2) Related variables

| VAR_INPUT | Name | Data type | Effective range | Initial value | Description |
|---|---|---|---|---|---|
| uiMilliseconds | Millisecond | UINT | - | - | Millisecond |
| uiSecond | Second | UINT | - | - | Second |
| uiMinute | Minute | UINT | - | - | Minute |
| uiHour | Hour | UINT | - | - | Hour |
| uiDay | Day | UINT | - | - | Day |
| uiMonth | Month | UINT | - | - | Month |
| uiYear | Year | UINT | - | - | Year |
| iTimezone | Time zone | INT | - | - | Time zone |

(3) Program example

| DEMO | LD | ST |
|---|---|---|
| Define variables | | VAR<br>XDONE: BOOL;<br>INMSE: UINT;<br>INSEC: UINT:= 3;<br>INMIN: UINT:= 31;<br>INHOU: UINT:= 8;<br>INDAY: UINT:= 9;<br>INMON: UINT:= 9;<br>INYE: UINT:= 2023;<br>INTZ: INT;<br>END_VAR |

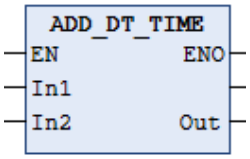
| | | |
|---|---|---|
| Program example |  | XDONE:= SetSystemDate(<br>uiMillisecond:= INMSE,<br>uiSecond:= INSEC,<br>uiMinute:= INMIN,<br>uiHour:= INHOU,<br>uiDay:= INDAY,<br>uiMonth:= INMON,<br>uiYear:= INYE,<br>iTimezone:= INTZ); |
| Operation result | 表达式 / 类型 / 值<br>XDONE / BOOL / TRUE<br>INMSE / UINT / 0<br>INSEC / UINT / 3<br>INMIN / UINT / 31<br>INHOU / UINT / 8<br>INDAY / UINT / 9<br>INMON / UINT / 9<br>INYE / UINT / 2023<br>INTZ / INT / 0 | |

## 6-4. Add date and time ADD_DT_TIME

(1) Instruction overview

Add the date and time.

| Add date and time [ADD_DT_TIME] | | | |
|---|---|---|---|
| Execution condition | Edge triggering | Suitable model | XS, XSA models |
| Firmware | - | Software | XS StudioV2.2.0 and above |
| Library | XJ_FunctionUtils | | |

| Command | Name | Graphical representation | ST language |
|---|---|---|---|
| ADD_DT_TIME | Add date and time |  | ADD_DT_TIME(<br>In1:= ,<br>In2:= ,<br>Out=> ); |

(2) Related variables

| VAR_INPUT | Name | Data type | Effective range | Initial value | Description |
|---|---|---|---|---|---|
| In1 | Date to be added | DT | As the data type | - | Date to be added |
| In2 | Time to be added | TIME | As the data type | - | Time to be added |
| VAR_OUTPUT | Name | Data type | Effective range | Initial value | Description |
| Out | The result | DT | As the data type | - | The result |

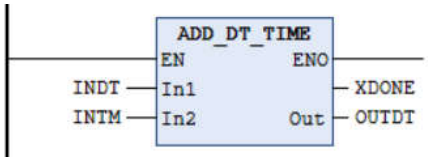
(3) Function

- The date time "In1" and time "In2" are added together, and the sum result "Out" is the date time.

(4) Note

- When the sum result exceeds the valid range of "Out", the program will not have any exceptions. Execute the following actions:
  DT#2106-2-7-6:28:15 + T#1s → DT#1970-1-1-0:0:0

(5) Program example

| DEMO | LD | ST |
|---|---|---|
| Define variables | VAR<br>INDT: DT:= DT#2023-9-9-9:11:15;<br>INTM: TIME:= T#10S;<br>OUTDT: DT;<br>XDONE: BOOL;<br>END_VAR | |
| Program example |  | XDONE:= ADD_DT_TIME(<br>In1:= INDT,<br>In2:= INTM,<br>Out=> OUTDT); |

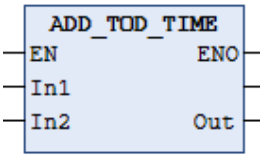
| Operation result | 表达式 | 类型 | 值 |
|---|---|---|---|
| | INDT | DATE_AND_TIME | DT#2023-9-9-9:11:15 |
| | INTM | TIME | T#10s |
| | OUTDT | DATE_AND_TIME | DT#2023-9-9-9:11:25 |
| | XDONE | BOOL | TRUE |

## 6-5. Time addition ADD_TOD_TIM

(1) Instruction overview

Time addition.

| Time addition [ADD_TOD_TIME] | | | |
|---|---|---|---|
| Execution condition | Edge triggering | Suitable model | XS, XSA models |
| Firmware | - | Software | XS StudioV2.2.0 and above |
| Library | XJ_FunctionUtils | | |

| Command | Name | Graphical representation | ST language |
|---|---|---|---|
| ADD_TOD_TIME | Time addition |  | ADD_TOD_TIME(<br>In1:= ,<br>In2:= ,<br>Out=> ); |

(2) Related variables

| VAR_INPUT | Name | Data type | Effective range | Initial value | Description |
|---|---|---|---|---|---|
| In1 | Added time | TOD | As the data type | - | Added time |
| In2 | Additive time | TIME | As the data type | - | Additive time |
| VAR_OUTPUT | Name | Data type | Effective range | Initial value | Description |
| Out | Sum result time | TOD | As the data type | - | Sum result time |

(3) Function

- Add time "In1" and time "In2", and the sum result "Out" is the time.

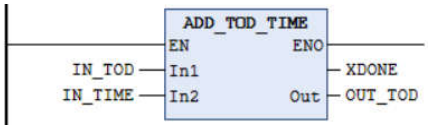
(4) Note

- When the sum result exceeds the valid range of "Out", the program will not have any exceptions. Execute the following actions:
  TOD#23:59:59.999 + T#1s1ms → TOD#0:0:1

(5) Program example

<table>
<tr><th>DEMO</th><th>LD</th><th>ST</th></tr>
<tr><td>Define variables</td><td colspan="2">

```
VAR
   IN_TOD: TOD:= TOD#10:11:16.003;
   IN_TIME: TIME:= T#2M16S;
   OUT_TOD: TOD;
   XDONE: BOOL;
END_VAR
```

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

```
XDONE:= ADD_TOD_TIME(
    In1:= IN_TOD,
    In2:= IN_TIME,
    Out=> OUT_TOD);
```

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

| 表达式 | 类型 | 值 |
|---|---|---|
| IN_TOD | TIME_OF_DAY | TOD#10:11:16.003 |
| IN_TIME | TIME | T#2m16s |
| OUT_TOD | TIME_OF_DAY | TOD#10:13:32.003 |
| XDONE | BOOL | TRUE |

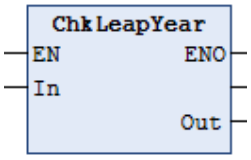
</td></tr>
</table>

## 6-6. Leap year judgment ChkLeapYear

(1) Instruction overview

Determine whether the specified year is a leap year.

| Leap year judgment [ChkLeapYear] | | | |
|---|---|---|---|
| Execution condition | Edge triggering | Suitable model | XS, XSA models |
| Firmware | - | Software | XS StudioV2.2.0 and above |
| Library | XJ_FunctionUtils | | |

| Command | Name | Graphical representation | ST language |
|---|---|---|---|
| ChkLeapYear | Leap year judgment |  | `ChkLeapYear(In:= , Out=> );` |

(2) Related variables

| VAR_INPUT | Name | Data type | Effective range | Initial value | Description |
|---|---|---|---|---|---|
| In | Year | UINT | 1970-2106 | - | Year |
| VAR_OUTPUT | Name | Data type | Effective range | Initial value | Description |
| Out | Judgment result | BOOL | FALSE/TRUE | - | True: leap year<br>FALSE: Non leap year |

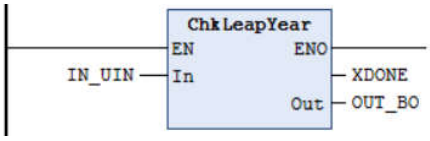
(3) Function

- Determine whether the year "In" is a leap year. If it is a leap year, the value of the judgment result "Out" is true. If not, it is FALSE.

(4) Note

- When the value of "In" exceeds the valid range, the program will encounter an exception. The output value of "Out" is FALSE, and the function return value is FALSE.

(5) Program example

| DEMO | LD | ST |
|---|---|---|
| Define variables | | `VAR`<br>`  IN_UIN: UINT:= 2020;`<br>`  OUT_BO: BOOL;`<br>`  XDONE: BOOL;`<br>`END_VAR` |
| Program example |  | `XDONE:= ChkLeapYear(`<br>`  In:= IN_UIN,`<br>`  Out=> OUT_BO);` |

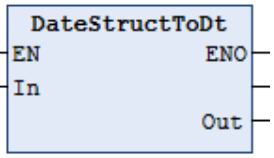
| | |
|---|---|
| Operation result | |

## 6-7. Date and time combination DateStructToDt

(1) Instruction overview

Combine dates and moments that are decomposed into years, months, days, hours, minutes, and seconds.

| Date and time combination [DateStructToDt] | | | |
|---|---|---|---|
| Execution condition | Edge triggering | Suitable model | XS, XSA models |
| Firmware | - | Software | XS StudioV2.2.0 and above |
| Library | XJ_FunctionUtils | | |

| Command | Name | Graphical representation | ST language |
|---|---|---|---|
| DateStructToDt | Date and time combination |  | `DateStructToDt(In:= , Out=> );` |

(2) Related variables

| VAR_INPUT | Name | Data type | Effective range | Initial value | Description |
|---|---|---|---|---|---|
| In | Date time | _sDT | As the data type | - | Date time before combining |
| VAR_OUTPUT | Name | Data type | Effective range | Initial value | Description |
| Out | Date time | DT | DT#1970-1-1-0:0:0-DT#2106-2-7-6:28:15 | DT#1970-1-1-0:0:0 | Date and time after combining |

(3) Function

- Combine the date and time "In" that are decomposed into "year", "month", "day", "hour", "minute", and "second".
- The members of the _sDT structure are as follows:

| Variable | Name | Data type | Valid range | Initial value | Description |
|---|---|---|---|---|---|
| uiYear | Year | UINT | 1970-2106 | 1970 | Year |
| uiMonth | Month | UINT | 1-12 | 1 | Month |
| uiDay | Day | UINT | 1-31 | 1 | Day |
| uiHour | Hour | UINT | 0-23 | 0 | Hour |
| uiMinute | Minute | UINT | 0-59 | 0 | Minute |
| uiSecond | Second | UINT | 0-59 | 0 | Second |
| uiMillisecond | Millisecond | UINT | 0-999 | 0 | Millisecond |
| uiDayOfWeek | Week | UINT | 1-7 | 1 | Week |

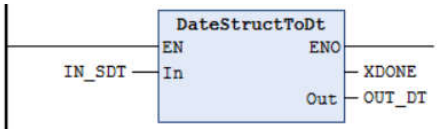
(4) Note

- The valid range for outputting the date and time "Out" is DT#1970-1-1-0:0:0 - DT#2106-2-7-6:28:15.
- If the output "Out" exceeds the valid range, the function returns a value of FALSE and the calculated result is DT#1970-1-1-0:0:0.
- If the input "In" exceeds the valid range of each element in the "_sDT" structure, the function returns a value of FALSE and the calculated result is DT#1970-1-1-0:0:0.
- The initial values of each element in the "_sDT" structure are the decomposition values of DT#1970-1-1-0:0:0.

(5) Program example

<table>
<tr><th>DEMO</th><th>LD</th><th>ST</th></tr>
<tr><td>Define variables</td><td colspan="2">

```
VAR
    IN_SDT: _sDT;
    OUT_DT: DT;
    XDONE: BOOL;
END_VAR
```

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

```
XDONE:= DateStructToDt(
    In:= IN_SDT,
    Out=> OUT_DT);
```

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

| 表达式 | 类型 | 值 |
|---|---|---|
| IN_SDT | _sDT | |
| uiYear | UINT | 2023 |
| uiMonth | UINT | 9 |
| uiDay | UINT | 9 |
| uiHour | UINT | 10 |
| uiMinute | UINT | 31 |
| uiSecond | UINT | 15 |
| uiMillisecond | UINT | 256 |
| uiDayOfWeek | UINT | 6 |
| OUT_DT | DATE_AND_TIME | DT#2023-9-9-10:31:15 |
| XDONE | BOOL | TRUE |

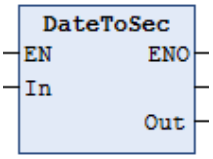
</td></tr>
</table>

## 6-8. Date second conversion DateToSec

(1) Instruction overview

Convert the input date to a number of seconds starting from January 1, 1970, 00:00:00.

| Date second conversion [DateToSec] | | | |
|---|---|---|---|
| Execution condition | Edge triggering | Suitable model | XS, XSA models |
| Firmware | - | Software | XS StudioV2.2.0 and above |
| Library | XJ_FunctionUtils | | |

| Command | Name | Graphical representation | ST language |
|---|---|---|---|
| DateToSec | Date second conversion |  | `DateToSec(In:= , Out=> );` |

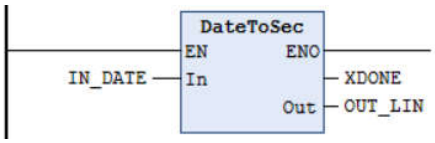
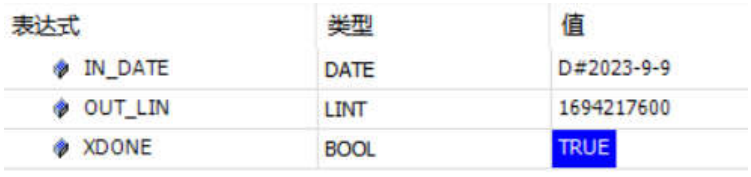
(2) Related variables

| VAR_INPUT | Name | Data type | Effective range | Initial value | Description |
|---|---|---|---|---|---|
| In | Date | DATE | As the data type | D#1970-1-1 | Date |
| VAR_OUTPUT | Name | Data type | Effective range | Initial value | Description |
| Out | Conversion result | LINT | 0-4294967295 | - | Conversion result |

(3) Function

- Convert the 0:00:00s of the date "In" to the number of seconds starting from 0:00:00s on January 1, 1970. The converted value is measured in seconds.

(4) Program example

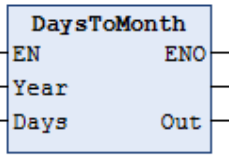
| DEMO | LD | ST |
|---|---|---|
| Define variables | | `VAR`<br>`  IN_DATE: DATE:= D#2023-9-9;`<br>`  OUT_LIN: LINT;`<br>`  XDONE: BOOL;`<br>`END_VAR` |
| Program example |  | `XDONE:= DateToSec(`<br>`  In:= IN_DATE,`<br>`  Out=> OUT_LIN);` |
| Operation result |  | |

## 6-9. Days and months conversion DaysToMonth

(1) Instruction overview

Calculate the month of the date according to the days starting from January 1st.

| Days and months conversion [DaysToMonth] | | | |
|---|---|---|---|
| Execution condition | Edge triggering | Suitable model | XS, XSA models |
| Firmware | - | Software | XS StudioV2.2.0 and above |
| Library | XJ_FunctionUtils | | |

| Command | Name | Graphical representation | ST language |
|---|---|---|---|
| DaysToMonth | Days and months conversion |  | DaysToMonth(<br>Year:= ,<br>Days:= ,<br>Out=> ); |

(2) Related variables

| VAR_INPUT | Name | Data type | Effective range | Initial value | Description |
|---|---|---|---|---|---|
| Year | Year | UINT | 1970-2106 | 1970 | Year |
| Days | Days | UINT | 1-365<br>(When 'Year' is a leap year, it is 1-366) | 1 | Days from January 1st |
| VAR_OUTPUT | Name | Data type | Effective range | Initial value | Description |
| Out | Month | USINT | 1-12 | - | Month |

(3) Function

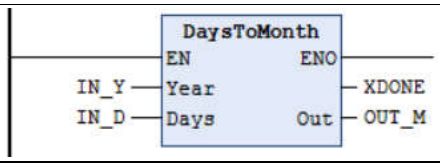
- Calculate the month of the day based on the number of days starting from January 1st of the year.

(4) Note

- When the value of "Year" exceeds the valid range, no exception will occur, and the value of "Out" is an illegal value.
- When the value of "Days" exceeds the valid range, the function returns a value of FALSE and "Out" becomes 0.

(5) Program example

| DEMO | LD | ST |
|---|---|---|
| Define variables | | VAR<br>IN_Y: UINT:= 2023;<br>IN_D: UINT:= 156;<br>OUT_M: USINT;<br>XDONE: BOOL;<br>END_VAR |

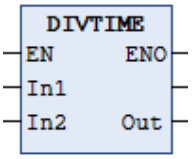
| | | |
|---|---|---|
| Program example |  | XDONE:= DaysToMonth(<br>Year:= IN_Y,<br>Days:= IN_D,<br>Out=> OUT_M); |
| Operation result | | |

## 6-10. Time division DIVTIME

(1) Instruction overview

Divide time by the specified divisor.

| Time division [DIVTIME] | | | |
|---|---|---|---|
| Execution condition | Edge triggering | Suitable model | XS, XSA models |
| Firmware | - | Software | XS StudioV2.2.0 and above |
| Library | XJ_FunctionUtils | | |

| Command | Name | Graphical representation | ST language |
|---|---|---|---|
| DIVTIME | Time division |  | DIVTIME(<br>In1:= ,<br>In2:= ,<br>Out=> ); |

(2) Related variables

| VAR_INPUT | Name | Data type | Effective range | Initial value | Description |
|---|---|---|---|---|---|
| In1 | Dividend time | TIME | As the data type | - | Dividend time |
| In2 | Divisor | USINT/UINT/UDINT/ULINT/<br>SINT/INT/DINT/LINT<br>REAL/LREAL | As the data type | - | Divisor |
| VAR_OUTPUT | Name | Data type | Effective range | Initial value | Description |
| Out | Division result | TIME | As the data type | - | Division result time |

(3) Function

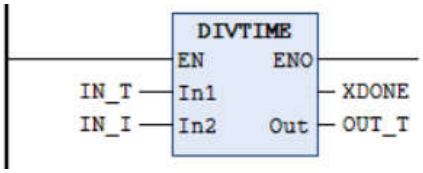
- The time "In1" is divided by the divisor "In2", and the division result "Out" is the time.

(4) Note

- When "In2" is negative, the program will encounter an exception. The output value of "Out" is T#0ms, and the function returns a value of FALSE.
- When the division result exceeds the valid range of "Out", the output value of "Out" is T#0ms, and the return value of the function is FALSE.
- When the input value of "In2" causes the output value of "Out" to exceed the upper limit of the TIME type, the output value of "Out" is T#0ms, and the function returns a value of FALSE.
- When "In2" is of integer type and has a value of 0, the function returns a value of FALSE and "Out" is 0.
- When "In2" is a real number, those less than 1ms in the division result will be rounded off.
- When "In2" is a real number, the division result with more significant digits may result in an error of several milliseconds.
- When "In2" is a real number and the value of "In2" is 0, +∞, -∞, and is not a numerical value, the function returns a value of True, and the value of "Out" is as follows:

| “In2” input value | “Out” output value |
|---|---|
| 0 | T#49d17h2m47s295ms |
| +∞ | T#0ms |
| -∞ | T#0ms |
| numerical value | T#0ms |

(5) Program example

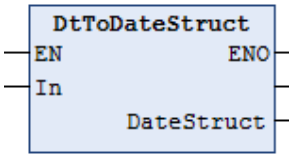
| DEMO | LD | ST |
|---|---|---|
| Define variables | | VAR<br>IN_T: TIME:= T#2D16H8M6S200MS;<br>IN_I: INT:= 2;<br>OUT_T: TIME;<br>XDONE: BOOL;<br>END_VAR |
| Program example |  | XDONE:= DIVTIME(<br>In1:= IN_T,<br>In2:= IN_I,<br>Out=> OUT_T); |
| Operation result | | |

## 6-11. Date time decomposition DtToDateStruct

(1) Instruction overview

Decompose the date and time into years, months, days, hours, minutes, and seconds.

| Date time decomposition [DtToDateStruct] | | | |
|---|---|---|---|
| Execution condition | Edge triggering | Suitable model | XS, XSA models |
| Firmware | - | Software | XS StudioV2.2.0 and above |
| Library | XJ_FunctionUtils | | |

| Command | Name | Graphical representation | ST language |
|---|---|---|---|
| DtToDateStruct | Date time decomposition |  | DtToDateStruct(In:= , DateStruct=> ); |

(2) Related variables

| VAR_INPUT | Name | Data type | Effective range | Initial value | Description |
|---|---|---|---|---|---|
| In | Date time | DT | DT#1970-1-1-0:0:0-DT#2106-2-7-6:28:15 | DT#1970-1-1-0:0:0 | Date and time before decomposition |
| VAR_OUTPUT | Name | Data type | Effective range | Initial value | Description |
| Out | Date time | _sDT | As the data type | - | Decomposed date and time |

(3) Function

- Decompose the date and time "In" into "year", "month", "day", "hour", "minute", and "second", and output them in the format of the _sDT structure.
- The members of the _sDT structure are as follows:

| Variable | Name | Data type | Effective range | Initial value | Description |
|---|---|---|---|---|---|
| uiYear | Year | UINT | 1970-2106 | 1970 | Year |
| uiMonth | Month | UINT | 1-12 | 1 | Month |
| uiDay | Day | UINT | 1-31 | 1 | Day |
| uiHour | Hour | UINT | 0-23 | 0 | Hour |
| uiMinute | Minute | UINT | 0-59 | 0 | Minute |
| uiSecond | Second | UINT | 0-59 | 0 | Second |
| uiMillisecond | Millisecond | UINT | 0-999 | 0 | Millisecond |
| uiDayOfWeek | Week | UINT | 1-7 | 1 | Week |

(4) Note

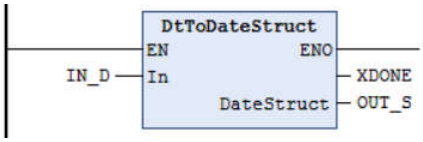
- The valid range for entering the date and time In is DT#1970-1-1-0:0:0 - DT#2106-2-7-6:28:15.
- If the input In exceeds the valid range, the function returns a value of FALSE and calculates the decomposition result of DT#1970-1-1-0:0:0.
- The initial values of each element in the structure of the _sDT are the decomposition values of DT#1970-1-1-0:0:0.

(5) Program example

<table>
<tr><th>DEMO</th><th>LD</th><th>ST</th></tr>
<tr><td>Define variables</td><td colspan="2">

```
VAR
    IN_D: DT:= DT#2023-9-9-11:31:21;
    OUT_S: _sDT;
    XDONE: BOOL;
END_VAR
```

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




</td><td>

```
XDONE:= DtToDateStruct(
    In:= IN_D,
    DateStruct=> OUT_S);
```

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

| 表达式 | 类型 | 值 |
|---|---|---|
| IN_D | DATE_AND_TIME | DT#2023-9-9-11:31:21 |
| OUT_S | _sDT | |
| uiYear | UINT | 2023 |
| uiMonth | UINT | 9 |
| uiDay | UINT | 9 |
| uiHour | UINT | 11 |
| uiMinute | UINT | 31 |
| uiSecond | UINT | 21 |
| uiMillisecond | UINT | 0 |
| uiDayOfWeek | UINT | 6 |
| XDONE | BOOL | TRUE |

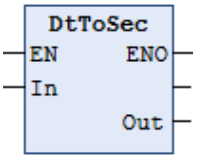
</td></tr>
</table>

## 6-12. Date, time, second conversion DtToSec

(1) Instruction overview

Convert the date and time to seconds starting from January 1, 1970, 00:00:00.

| Date, time, second conversion [DtToSec] | | | |
|---|---|---|---|
| Execution condition | Edge triggering | Suitable model | XS, XSA models |
| Firmware | - | Software | XS StudioV2.2.0 and above |
| Library | XJ_FunctionUtils | | |

| Command | Name | Graphical representation | ST language |
|---|---|---|---|
| DtToSec | Date, time, second conversion |  | `DtToSec(In:= , Out=> );` |

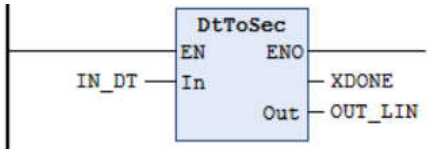
(2) Related variables

| VAR_INPUT | Name | Data type | Effective range | Initial value | Description |
|---|---|---|---|---|---|
| In | Date time | DT | As the data type | DT#1970-1-1-0:0:0 | Date time |
| VAR_OUTPUT | Name | Data type | Effective range | Initial value | Description |
| Out | Conversion result | LINT | 0-4294967295 | - | Conversion result |

(3) Function

- Convert the date and time "In" to the number of seconds starting from January 1, 1970, 00:00:00. The converted value is measured in seconds. Values less than 1 second will be rounded off.

(4) Program example

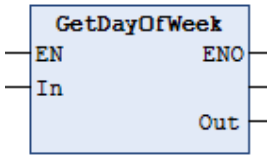
| DEMO | LD | ST |
|---|---|---|
| Define variables | | `VAR`<br>`IN_DT: DT:= DT#2023-9-9-11:36:22;`<br>`OUT_LIN: LINT;`<br>`XDONE: BOOL;`<br>`END_VAR` |
| Program example |  | `XDONE:= DtToSec(`<br>`In:= IN_DT,`<br>`Out=> OUT_LIN);` |
| Operation result | | |

## 6-13. Get week information GetDayOfWeek

(1) Instruction overview

Get the week information for the specified month, month, and day.

| Get week information [GetDayOfWeek] | | | |
|---|---|---|---|
| Execution condition | Edge triggering | Suitable model | XS, XSA models |
| Firmware | - | Software | XS StudioV2.2.0 and above |
| Library | XJ_FunctionUtils | | |

| Command | Name | Graphical representation | ST language |
|---|---|---|---|
| GetDayOfWeek | Get week information |  | GetDayOfWeek(In:= , Out=> ); |



(2) Related variables

| VAR_INPUT | Name | Data type | Effective range | Initial value | Description |
|---|---|---|---|---|---|
| In | Date | DATE/DT | DATE#1970-1-1-DATE#2106-2-7 DT#1970-1-1-0:0:0-DT#2106-2-7-6:28:15 | - | Year month day |
| VAR_OUTPUT | Name | Data type | Effective range | Initial value | Description |
| Out | Week | _eDAYOFWEEK | As the data type | _MON | Week information |

(3) Function

- Obtain the week information of the year, month, and day shown in "In".
- The data type of "Out" is the enumeration type _eDAYOFWEEK, and the meaning of the enumeration value is as follows:

| _eDAYOFWEEK enum value | Meaning |
|---|---|
| _MON | Monday |
| _TUE | Tuesday |
| _WED | Wednesday |
| _THU | Thursday |
| _FRI | Friday |
| _SAT | Saturday |
| _SUN | Sunday |

(4) Note

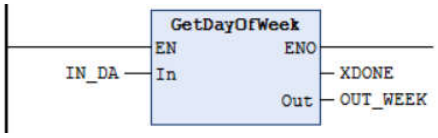
- When the value of "In" exceeds the valid range, the program will encounter an exception. The output value of "Out" is _MON, and the function returns a value of FALSE.
- The input parameter passed to "In" must be a variable. If a constant is passed, a compilation error occurs.
- "In" requires the use of DATA and DT types. For data types TIME and TOD, compile without error, and the function returns FALSE after running; For other types of data, compile an error.

(5) Program example

<table>
<tr><th>DEMO</th><th>LD</th><th>ST</th></tr>
<tr><td>Define variables</td><td colspan="2">

```
VAR
   IN_DA: DATE:= DATE#2023-9-9;
   OUT_WEEK: _eDAYOFWEEK;
   XDONE: BOOL;
END_VAR
```

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




</td><td>

```
XDONE:= GetDayOfWeek(
    In:= IN_DA,
    Out=> OUT_WEEK);
```

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

| 表达式 | 类型 | 值 |
|---|---|---|
| IN_DA | DATE | D#2023-9-9 |
| OUT_WEEK | _EDAYOFWEEK | _SAT |
| XDONE | BOOL | TRUE |

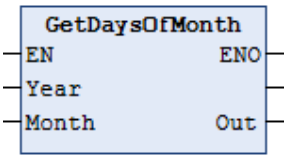
</td></tr>
</table>

## 6-14. Obtain the number of days in the target month GetDaysOfMonth

(1) Instruction overview

Get the number of days in the specified month.

| Obtain the number of days in the target month [GetDaysOfMonth] | | | |
|---|---|---|---|
| Execution condition | Edge triggering | Suitable model | XS, XSA models |
| Firmware | - | Software | XS StudioV2.2.0 and above |
| Library | XJ_FunctionUtils | | |

| Command | Name | Graphical representation | ST language |
|---|---|---|---|
| GetDaysOfMonth | Obtain the number of days in the target month |  | GetDaysOfMonth(<br>Year:= ,<br>Month:= ,<br>Out=> ); |

(2) Related variables

| VAR_INPUT | Name | Data type | Effective range | Initial value | Description |
|---|---|---|---|---|---|
| Year | Year | UINT | 1970-2106 | - | Year |
| Month | Month | USINT | 1-12 | - | Month |
| VAR_OUTPUT | Name | Data type | Effective range | Initial value | Description |
| Out | Days | USINT | 28-31 | - | Days |

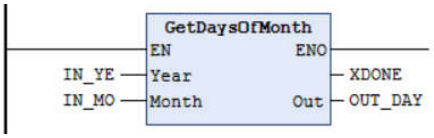
(3) Function

- Obtain the number of days for the target year "Year" and target month "Month".

(4) Note

- When the value of "Year" exceeds the valid range, the program will encounter an exception. The output value of "Out" is 0, and the function return value is FALSE.
- When the value of "Month" exceeds the valid range, the program will encounter an exception. The output value of "Out" is 0, and the function return value is FALSE.

(5) Program example

| DEMO | LD | ST |
|---|---|---|
| Define variables | | VAR<br>IN_YE: UINT:= 2023;<br>IN_MO: USINT:= 9;<br>OUT_DAY: USINT;<br>XDONE: BOOL;<br>END_VAR |
| Program example |  | XDONE:= GetDaysOfMonth(<br>Year:= IN_YE,<br>Month:= IN_MO,<br>Out=> OUT_DAY); |

| | |
|---|---|
| Operation result | |

## 6-15. Obtain the current system time structure GetSystemDate_sDt

(1) Instruction overview

Obtain the current system time structure.

| Obtain the current system time structure [GetSystemDate_sDt] | | | |
|---|---|---|---|
| Execution condition | Edge triggering | Suitable model | XS, XSA models |
| Firmware | - | Software | XS StudioV2.2.0 and above |
| Library | XJ_FunctionUtils | | |

| Command | Name | Graphical representation | ST language |
|---|---|---|---|
| GetSystemDate_sDt | Obtain the current system time structure | | GetSystemDate_sDt( stSystemDate =>); |

(2) Related variables

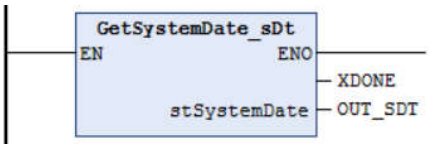
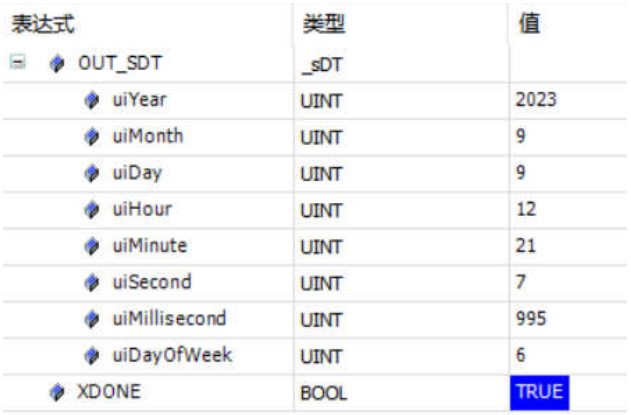
| VAR_OUTPUT | Name | Data type | Effective range | Initial value | Description |
|---|---|---|---|---|---|
| stSystemDate | System time | _sDT | - | - | System time structure |

(3) Function

- Read the current system time and output it in the format of the _sDT structure.
- The members of the _sDT structure are as follows:

| Variables | Name | Data type | Effective range | Initial value | Description |
|---|---|---|---|---|---|
| uiYear | Year | UINT | 1970-2106 | 1970 | Year |
| uiMonth | Month | UINT | 1-12 | 1 | Month |
| uiDay | Day | UINT | 1-31 | 1 | Day |
| uiHour | Hour | UINT | 0-23 | 0 | Hour |
| uiMinute | Minute | UINT | 0-59 | 0 | Minute |
| uiSecond | Second | UINT | 0-59 | 0 | Second |
| uiMillisecond | Millisecond | UINT | 0-999 | 0 | Millisecond |
| uiDayOfWeek | Week | UINT | 1-7 | 1 | Week |

(4) Program example

| DEMO | LD | ST |
|---|---|---|
| Define variables | VAR<br>OUT_SDT: _sDT;<br>XDONE: BOOL;<br>END_VAR | |
| Program example |  | XDONE:= GetSystemDate_sDt(<br>stSystemDate=> OUT_SDT); |
| Operation result |  | |

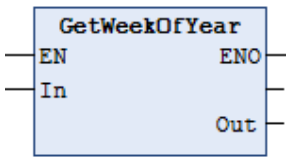
| 表达式 | 类型 | 值 |
|---|---|---|
| OUT_SDT | _sDT | |
| uiYear | UINT | 2023 |
| uiMonth | UINT | 9 |
| uiDay | UINT | 9 |
| uiHour | UINT | 12 |
| uiMinute | UINT | 21 |
| uiSecond | UINT | 7 |
| uiMillisecond | UINT | 995 |
| uiDayOfWeek | UINT | 6 |
| XDONE | BOOL | TRUE |

## 6-16. Get week information GetWeekOfYear

(1) Instruction overview

Calculate the specified year, month, and day as the week of the current year.

| Get week information [GetWeekOfYear] | | | |
|---|---|---|---|
| Execution condition | Edge triggering | Suitable model | XS, XSA models |
| Firmware | - | Software | XS StudioV2.2.0 and above |
| Library | XJ_FunctionUtils | | |

| Command | Name | Graphical representation | ST language |
|---|---|---|---|
| GetWeekOfYear | Get week information |  | GetWeekOfYear(In:= , Out=> ); |

(2) Related variables

| VAR_INPUT | Name | Data type | Effective range | Initial value | Description |
|---|---|---|---|---|---|
| In | Year | DATE/DT | DATE#1970-1-1-DATE#2106-2-7 | - | Year month day |

| | month day | | DT#1970-1-1-0:0:0-DT#2106-2-7-6:28:15 | | |
|---|---|---|---|---|---|
| VAR_OUTPUT | Name | Data type | Effective range | Initial value | Description |
| Out | Week | USINT | 1-54 | - | Weeks of the target year |

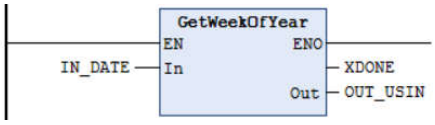
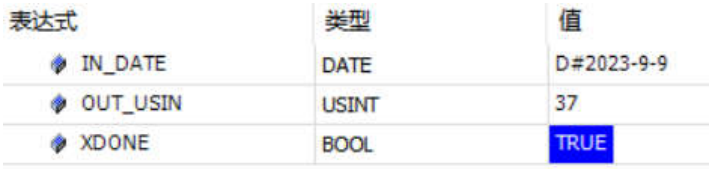
(3) Function

- Calculate the weeks of the target year, month, day “In”.
- Monday to Sunday is considered as one week, and when it changes from Sunday to Monday, it is counted as two weeks.
- January 1st must be the first week.

(4) Note

- When the value of "In" exceeds the valid range, the program will encounter an exception. The output value of "Out" is 0, and the function return value is FALSE.
- The input parameter passed to "In" must be a variable. If a constant is passed, a compilation error occurs.

(5) Program example

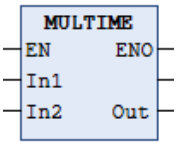
| DEMO | LD | ST |
|---|---|---|
| Define variables | | VAR<br>IN_DATE: DATE:= DATE#2023-9-9;<br>OUT_USIN: USINT;<br>XDONE: BOOL;<br>END_VAR |
| Program example |  | XDONE:= GetWeekOfYear(<br>In:= IN_DATE,<br>Out=> OUT_USIN); |
| Operation result |  | |

## 6-17. Multiplying time MULTIME

(1) Instruction overview

Multiply time by the specified multiplier.

| Multiplying time [MULTIME] | | | |
|---|---|---|---|
| Execution condition | Edge triggering | Suitable model | XS, XSA models |
| Firmware | - | Software | XS StudioV2.2.0 and above |
| Library | XJ_FunctionUtils | | |

| Command | Name | Graphical representation | ST language |
|---|---|---|---|
| MULTIME | Multiplying time |  | MULTIME(<br>In1:= ,<br>In2:= ,<br>Out=> ); |

(2) Related variables

| VAR_INPUT | Name | Data type | Effective range | Initial value | Description |
|---|---|---|---|---|---|
| In1 | Time to be multiplied | TIME | As the data type | - | Time to be multiplied |
| In2 | Multiplier | USINT/UINT/UDINT/ULINT / SINT/INT/DINT/LINT/ REAL/LREAL | As the data type | - | Multiplier |
| VAR_OUTPUT | Name | Data type | Effective range | Initial value | Description |
| Out | Multiplication result | TIME | As the data type | - | Multiplication result |

(3) Function

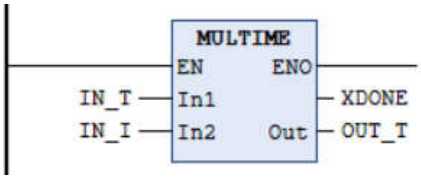
- The time "In1" is multiplied by the multiplier "In2", and the multiplication result "Out" is the time.

(4) Note

- When "In2" is negative, the program will encounter an exception. The output value of "Out" is T#0ms, and the function return value is FALSE.
- When the multiplication result exceeds the valid range of "Out", the output value of "Out" is T#0ms, and the return value of the function is FALSE.
- When the input value of "In2" causes the output value of "Out" to exceed the upper limit of the TIME type, the output value of "Out" is T#0ms, and the function returns a value of FALSE.
- When "In2" is a real number, the multiplication result with less than 1ms will be rounded off.
- When "In2" is a real number, multiplying result with more significant digits may result in an error of several milliseconds.
- The value of "In2" is +∞, -∞, and when it is not a numerical value, the value of "Out" is as follows:

| “In2” input value | “Out” output value |
|---|---|
| +∞ | T#0ms |
| -∞ | T#0ms |
| Non numerical value | T#0ms |

(5) Program example

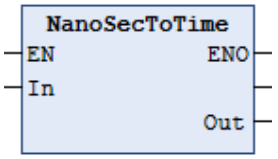
| DEMO | LD | ST |
|---|---|---|
| Define variables | VAR<br>IN_T: TIME:= T#5D3H10M16S;<br>IN_I: INT:= 2;<br>OUT_T: TIME;<br>XDONE: BOOL;<br>END_VAR | |
| Program example |  | XDONE:= MULTIME(<br>In1:= IN_T,<br>In2:= IN_I,<br>Out=> OUT_T); |
| Operation result | | |

## 6-18. Nanosecond time conversion NanoSecToTime

(1) Instruction overview

Convert nanoseconds to time.

| Nanosecond time conversion [NanoSecToTime] | | | |
|---|---|---|---|
| Execution condition | Edge triggering | Suitable model | XS, XSA models |
| Firmware | - | Software | XS StudioV2.2.0 and above |
| Library | XJ_FunctionUtils | | |

| Command | Name | Graphical representation | ST language |
|---|---|---|---|
| NanoSecToTime | Nanosecond time conversion |  | NanoSecToTime(In:= , Out=> ); |

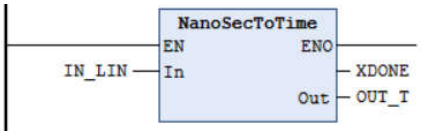
(2) Related variables

| VAR_INPUT | Name | Data type | Effective range | Initial value | Description |
|---|---|---|---|---|---|
| In | Nanoseconds | LINT | 0-4294967295000000 | 0 | Nanoseconds |
| VAR_OUTPUT | Name | Data type | Effective range | Initial value | Description |
| Out | Conversion result | TIME | As the data type | - | Time after converted |

(3) Function

- Convert nanosecond "In" to time.

(4) Program example

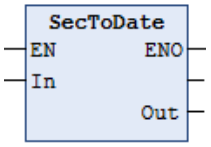
| DEMO | LD | ST |
|---|---|---|
| Define variables | VAR<br>IN_LIN: LINT:= 65535000000;<br>OUT_T: TIME;<br>XDONE: BOOL;<br>END_VAR | |
| Program example |  | XDONE:= NanoSecToTime(<br>In:= IN_LIN,<br>Out=> OUT_T); |
| Operation result | | |

## 6-19. Second date conversion SecToDate

(1) Instruction overview

Convert the number of seconds counted from January 1, 1970, 00:00:00, to a date.

| Second date conversion [SecToDate] | | | |
|---|---|---|---|
| Execution condition | Edge triggering | Suitable model | XS, XSA models |
| Firmware | - | Software | XS StudioV2.2.0 and above |
| Library | XJ_FunctionUtils | | |

| Command | Name | Graphical representation | ST language |
|---|---|---|---|
| SecToDate | Second date conversion |  | SecToDate(In:= , Out=> ); |

(2) Related variables

| VAR_INPUT | Name | Data type | Effective range | Initial value | Description |
|---|---|---|---|---|---|
| In | Seconds | LINT | 0-4294967295 | 0 | The number of seconds counted from 00:00:00 on January 1, 1970 |
| VAR_OUTPUT | Name | Data type | Effective range | Initial value | Description |

| | | | | | |
|---|---|---|---|---|---|
| Out | Conversion result | DATE | As the data type | - | Date after conversion |

(3) Function

- Convert the number of seconds "In" starting from 00:00:00 on January 1, 1970 to a date. Values less than 1 day will be rounded off.

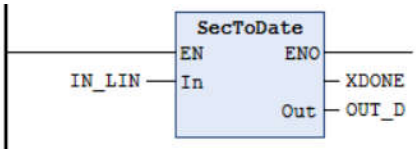
(4) Note

- When the value of "In" exceeds the valid range, the function return value becomes FALSE, and the value of "Out" becomes D#1970-1-1.

(5) Program example

| DEMO | LD | ST |
|---|---|---|
| Define variables | | |
| Program example | | |
| Operation result | | |

```
VAR
   IN_LIN: LINT:= 1694217600;
   OUT_D: DATE;
   XDONE: BOOL;
END_VAR
```




```
XDONE:= SecToDate(
    In:= IN_LIN,
    Out=> OUT_D);
```

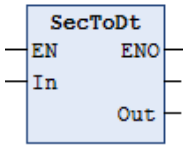
| 表达式 | 类型 | 值 |
|---|---|---|
| IN_LIN | LINT | 1694217600 |
| OUT_D | DATE | D#2023-9-9 |
| XDONE | BOOL | TRUE |

## 6-20. Second date time conversion SecToDt

(1) Instruction overview

Convert the number of seconds starting from 00:00:00 on January 1, 1970 to date time.

| Second date time conversion [SecToDt] | | | |
|---|---|---|---|
| Execution condition | Edge triggering | Suitable model | XS, XSA models |
| Firmware | - | Software | XS StudioV2.2.0 and above |
| Library | XJ_FunctionUtils | | |

| Command | Name | Graphical representation | ST language |
|---|---|---|---|
| SecToDt | Second date time conversion |  | `SecToDt(In:= , Out=> );` |

(2) Related variables

| VAR_INPUT | Name | Data type | Effective range | Initial value | Description |
|---|---|---|---|---|---|
| In | Seconds | LINT | 0-4294967295 | 0 | The number of seconds starting from 00:00:00 on January 1, 1970 |
| VAR_OUTPUT | Name | Data type | Effective range | Initial value | Description |
| Out | Conversion result | DT | As the data type | - | Date and time of conversion |

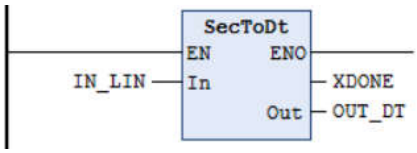
(3) Function

- Convert the number of seconds "In" starting from 00:00:00 on January 1, 1970 to a date time.

(4) Note

- When the value of "In" exceeds the valid range, the function return value becomes FALSE, and the value of "Out" becomes DT#1970-1-1-0:0:0.

(5) Program example

| DEMO | LD | ST |
|---|---|---|
| Define variables | | `VAR`<br>`  IN_LIN: LINT:= 1694263261;`<br>`  OUT_DT: DT;`<br>`  XDONE: BOOL;`<br>`END_VAR` |
| Program example |  | `XDONE:= SecToDt(`<br>`  In:= IN_LIN,`<br>`  Out=> OUT_DT);` |

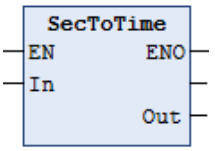
| | |
|---|---|
| Operation result | |

## 6-21. Second time conversion SecToTime

(1) Instruction overview

Convert seconds to time.

| Second time conversion [SecToTime] | | | |
|---|---|---|---|
| Execution condition | Edge triggering | Suitable model | XS, XSA models |
| Firmware | - | Software | XS StudioV2.2.0 and above |
| Library | XJ_FunctionUtils | | |

| Command | Name | Graphical representation | ST language |
|---|---|---|---|
| SecToTime | Second time conversion |  | `SecToTime(In:= , Out=> );` |

(2) Related variables

| VAR_INPUT | Name | Data type | Effective range | Initial value | Description |
|---|---|---|---|---|---|
| In | Seconds | LINT | 0-4294967295 | 0 | Seconds |
| VAR_OUTPUT | Name | Data type | Effective range | Initial value | Description |
| Out | Conversion result | TIME | As the data type | - | Time after conversion |

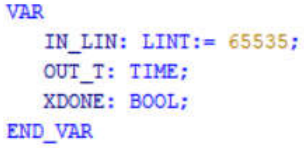
(3) Function

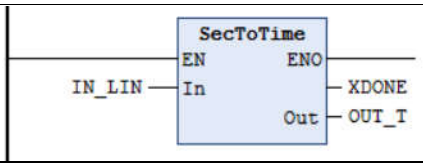
- Convert the number of seconds "In" to time.

(4) Note

- The unit of "In" is seconds, and the unit of "Out" is milliseconds.
- When the value of "In" exceeds the valid range, the function returns a value of FALSE and "Out" becomes 0.

(5) Program example

| DEMO | LD | ST |
|---|---|---|
| Define variables | |  |

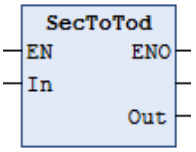
| | | |
|---|---|---|
| Program example |  | XDONE:= SecToTime(<br>In:= IN_LIN,<br>Out=> OUT_T); |
| Operation result | | |

## 6-22. Second time conversion SecToTod

(1) Instruction overview

Convert the number of seconds starting from 0:0:00s to time.

| Second time conversion [SecToTod] | | | |
|---|---|---|---|
| Execution condition | Edge triggering | Suitable model | XS, XSA models |
| Firmware | - | Software | XS StudioV2.2.0 and above |
| Library | XJ_FunctionUtils | | |

| Command | Name | Graphical representation | ST language |
|---|---|---|---|
| SecToTod | Second time conversion |  | SecToTod(In:= , Out=> ); |

(2) Related variables

| VAR_INPUT | Name | Data type | Effective range | Initial value | Description |
|---|---|---|---|---|---|
| In | Seconds | LINT | 0-4294967295 | 0 | The number of seconds counted from 0:0:00s |
| VAR_OUTPUT | Name | Data type | Effective range | Initial value | Description |
| Out | Conversion result | TOD | As the data type | - | The time after conversion |

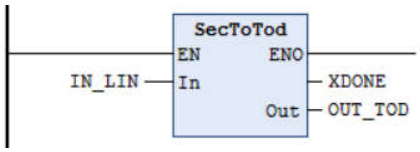
(3) Function

- Convert the number of seconds "In" starting from 0:0:00s to a time. When the value of "In" is greater than 24 hours, divide the value of "In" by 24 hours, and then convert the remainder to time.

(4) Note

- When the value of "In" exceeds the valid range, the return value of the function becomes FALSE, and the value of "Out" becomes TOD#0:0:0.

(5) Program example

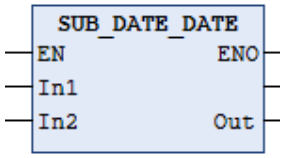
| DEMO | LD | ST |
|---|---|---|
| Define variables | | VAR<br>IN_LIN: LINT:= 65535;<br>OUT_TOD: TIME_OF_DAY;<br>XDONE: BOOL;<br>END_VAR |
| Program example |  | XDONE:= SecToTod(<br>In:= IN_LIN,<br>Out=> OUT_TOD); |
| Operation result | | |

## 6-23. Date subtraction SUB_DATE_DATE

(1) Instruction overview

Subtract the date from the date.

| Date subtraction [SUB_DATE_DATE] | | | |
|---|---|---|---|
| Execution condition | Edge triggering | Suitable model | XS, XSA models |
| Firmware | - | Software | XS StudioV2.2.0 and above |
| Library | XJ_FunctionUtils | | |

| Command | Name | Graphical representation | ST language |
|---|---|---|---|
| SUB_DATE_DATE | Date subtraction |  | SUB_DATE_DATE(<br>In1:= ,<br>In2:= ,<br>Out=> ); |

(2) Related variables

| VAR_INPUT | Name | Data type | Effective range | Initial value | Description |
|---|---|---|---|---|---|
| In1 | Date to be subtracted | DATE | As the data type | - | Date to be subtracted |
| In2 | Subtraction date | DATE | As the data type | - | Subtraction date |
| VAR_OUTPUT | Name | Data type | Effective range | Initial value | Description |
| Out | Subtraction result time | TIME | As the data type | - | Subtraction result time |

(3) Function

- Subtract the date "In1" from the date "In2", and the subtraction result "Out" is the time.

(4) Note

- When the subtraction result exceeds the valid range of "Out", the program will encounter an exception. The output value of "Out" is T#0ms, and the function return value is FALSE.
- The difference between the date "In1" and the date "In2" must be within the upper limit of the Time type (T#49d17h2m47s295ms), and the value of "In1" must be greater than "In2", otherwise the program will encounter an exception and the function will return a value of "FALSE".

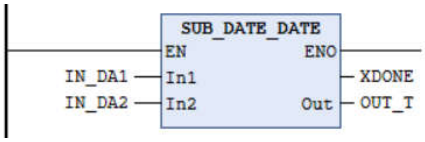
Note: "In1" (close to the lower limit value) minus "In2" (close to the upper limit value), and the difference between "In1" and "In2" is within the upper limit value of Time type (T#49d17h2m47s295ms), the program will not have any exceptions. For example, subtracting the value of the lower limit (DATE#1970-1-1) from the value of the upper limit (DATE#2106-2-7) yields (T#1d).

(5) Program example

<table>
<tr><th>DEMO</th><th>LD</th><th>ST</th></tr>
<tr><td>Define variables</td><td colspan="2">

```
VAR
   IN_DA1: DATE:= DATE#2023-9-11;
   IN_DA2: DATE:= DATE#2023-9-8;
   OUT_T: TIME;
   XDONE: BOOL;
END_VAR
```

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




</td><td>

```
XDONE:= SUB_DATE_DATE(
    In1:= IN_DA1,
    In2:= IN_DA2,
    Out=> OUT_T);
```

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

| 表达式 | 类型 | 值 |
|---|---|---|
| IN_DA1 | DATE | D#2023-9-11 |
| IN_DA2 | DATE | D#2023-9-8 |
| OUT_T | TIME | T#3d |
| XDONE | BOOL | TRUE |

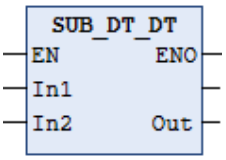
</td></tr>
</table>

## 6-24. Subtracting date and time SUB_DT_DT

(1) Instruction overview

Subtract the date time from the date time.

| Subtracting date and time [SUB_DT_DT] | | | |
|---|---|---|---|
| Execution condition | Edge triggering | Suitable model | XS, XSA models |
| Firmware | - | Software | XS StudioV2.2.0 and above |
| Library | XJ_FunctionUtils | | |

| Command | Name | Graphical representation | ST language |
|---|---|---|---|
| SUB_DT_DT | Subtracting date and time |  | SUB_DT_DT(<br>In1:= ,<br>In2:= ,<br>Out=> ); |

(2) Related variables

| VAR_INPUT | Name | Data type | Effective range | Initial value | Description |
|---|---|---|---|---|---|
| In1 | Date time to be subtracted | DT | As the data type | - | Date time to be subtracted |
| In2 | Subtraction date and time | DT | As the data type | - | Subtraction date and time |
| VAR_OUTPUT | Name | Data type | Effective range | Initial value | Description |
| Out | Subtraction result time | TIME | As the data type | - | Subtraction result time |

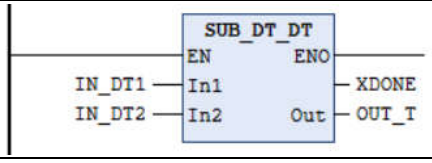
(3) Function

- Subtract the date time "In1" from the date time "In2", and the subtracted result "Out" is the time.

(4) Note

- When the subtraction result exceeds the valid range of "Out", the program will encounter an exception. The output value of "Out" is T#0ms, and the function return value is FALSE.
- If the value of the subtracted date "In1" is smaller than the value of the subtracted date "In2", the program will not have any exceptions. Execute the following action. DT#1970-1-1-0:0:0 - DT#2106-2-6-6:28:15 → T#1d1s.

(5) Program example

| DEMO | LD | ST |
|---|---|---|
| Define variables | | VAR<br>IN_DT1: DT:= DT#2023-9-11-8:15:22;<br>IN_DT2: DT:= DT#2023-9-8-9:25:18;<br>OUT_T: TIME;<br>XDONE: BOOL;<br>END_VAR |
| Program example |  | XDONE:= SUB_DT_DT(<br>In1:= IN_DT1,<br>In2:= IN_DT2,<br>Out=> OUT_T); |

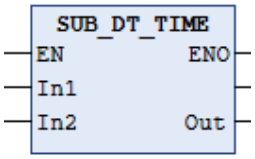
| | |
|---|---|
| Operation result | |

## 6-25. Subtracting date, time, and time SUB_DT_TIME

(1) Instruction overview

Subtract time from date and time.

| Subtracting date, time, and time [SUB_DT_TIME] | | | |
|---|---|---|---|
| Execution condition | Edge triggering | Suitable model | XS, XSA models |
| Firmware | - | Software | XS StudioV2.2.0 and above |
| Library | XJ_FunctionUtils | | |

| Command | Name | Graphical representation | ST language |
|---|---|---|---|
| SUB_DT_TIME | Subtracting date, time, and time |  | SUB_DT_TIME(<br>In1:= ,<br>In2:= ,<br>Out=> ); |

(2) Related variables

| VAR_INPUT | Name | Data type | Effective range | Initial value | Description |
|---|---|---|---|---|---|
| In1 | Date time to be subtracted | DT | As the data type | - | Date time to be subtracted |
| In2 | Subtraction time | TIME | As the data type | - | Subtraction time |
| VAR_OUTPUT | Name | Data type | Effective range | Initial value | Description |
| Out | Subtraction result date and time | DT | As the data type | - | Subtraction result date and time |

(3) Function

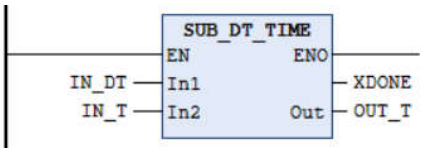
- Subtract time "In2" from date and time "In1", and the result of subtraction "Out" is the date and time.

(4) Note

- When the subtraction result exceeds the valid range of "Out", the program will not have any exceptions. Execute the following action. DT#1970-1-1-0:0:0 - T#1d → DT#2106-2-6-6:28:15.
- The minimum unit of "In2" value needs to be set to s (greater than or equal to 1000ms). When "In2" is less than 1000ms, the value of "Out" is consistent with the value of "In1".

(5) Program example

| DEMO | LD | ST |
|---|---|---|
| Define variables | | VAR<br>IN_DT: DT:= DT#2023-9-11-9:21:11;<br>IN_T: TIME:= T#1D1H1M1S;<br>OUT_T: DT;<br>XDONE: BOOL;<br>END_VAR |

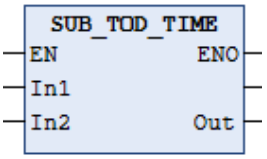
| | | |
|---|---|---|
| Program example |  | XDONE:= SUB_DT_TIME(<br>In1:= IN_DT,<br>In2:= IN_T,<br>Out=> OUT_T); |
| Operation result | | |

## 6-26. Subtracting time from time SUB_TOD_TIME

(1) Instruction overview

Subtract time from time.

| Subtracting time from time [SUB_TOD_TIME] | | | |
|---|---|---|---|
| Execution condition | Edge triggering | Suitable model | XS, XSA models |
| Firmware | - | Software | XS StudioV2.2.0 and above |
| Library | XJ_FunctionUtils | | |

| Command | Name | Graphical representation | ST language |
|---|---|---|---|
| SUB_TOD_TIME | Subtracting time from time |  | SUB_TOD_TIME(<br>In1:= ,<br>In2:= ,<br>Out=> ); |

(2) Related variables

| VAR_INPUT | Name | Data type | Effective range | Initial value | Description |
|---|---|---|---|---|---|
| In1 | Time to be subtracted | TOD | As the data type | - | Time to be subtracted |
| In2 | Subtraction time | TIME | As the data type | - | Subtraction time |
| VAR_OUTPUT | Name | Data type | Effective range | Initial value | Description |
| Out | Subtraction result time | TOD | As the data type | - | Subtraction result time |

(3) Function

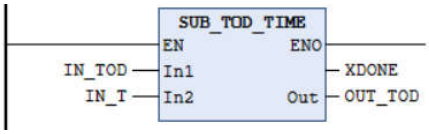
- Subtract time "In2" from time "In1", and the subtracted result "Out" is the time.

(4) Note

- TOD#0:0:0 - T#1ms → TOD#23:59:59:999。When the subtraction result exceeds the valid range of "Out", the program will not have any exceptions. Execute the following action. TOD#0:0:0 - T#1ms →

TOD#23:59:59:999.

(5) Program example

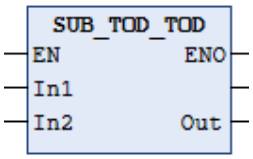
| DEMO | LD | ST |
|---|---|---|
| Define variables | | VAR<br>IN_TOD: TOD:= TOD#18:31:29;<br>IN_T: TIME:= T#3H11M2S;<br>OUT_TOD: TOD;<br>XDONE: BOOL;<br>END_VAR |
| Program example |  | XDONE:= SUB_TOD_TIME(<br>In1:= IN_TOD,<br>In2:= IN_T,<br>Out=> OUT_TOD); |
| Operation result | | |

## 6-27. Time subtraction SUB_TOD_TOD

(1) Instruction overview

Subtract time from time.

| Time subtraction [SUB_TOD_TOD] | | | |
|---|---|---|---|
| Execution condition | Edge triggering | Suitable model | XS, XSA models |
| Firmware | - | Software | XS StudioV2.2.0 and above |
| Library | XJ_FunctionUtils | | |

| Command | Name | Graphical representation | ST language |
|---|---|---|---|
| SUB_TOD_TOD | Time subtraction |  | SUB_TOD_TOD(<br>In1:= ,<br>In2:= ,<br>Out=> ); |

(2) Related variables

| VAR_INPUT | Name | Data type | Effective range | Initial value | Description |
|---|---|---|---|---|---|
| In1 | Time to be subtracted | TOD | As the data type | - | Time to be subtracted |
| In2 | Subtraction time | TOD | As the data type | - | Subtraction time |
| VAR_OUTPUT | Name | Data type | Effective range | Initial | Description |

| | | | | value | |
|---|---|---|---|---|---|
| Out | Subtraction result time | TIME | As the data type | - | Subtraction result time |

(3) Function

- Subtract time "In1" from time "In2", and the subtracted result "Out" is the time.

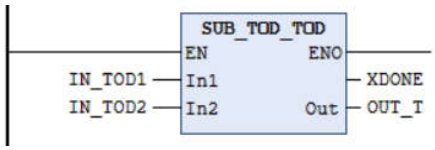
(4) Note

- When the value of the subtracted time "In1" is smaller than the value of the subtracted time "In2", the program will not have any exceptions. Execute the following actions:
  TOD#13:34:56.789 - TOD#13:35:56.789 → T#23h59m

(5) Program example

| DEMO | LD | ST |
|---|---|---|
| Define variables | | |
| Program example | | |
| Operation result | | |

```
VAR
   IN_TOD1: TOD:= TOD#19:30:21;
   IN_TOD2: TOD:= TOD#10:12:56;
   OUT_T: TIME;
   XDONE: BOOL;
END_VAR
```




```
XDONE:= SUB_TOD_TOD(
    In1:= IN_TOD1,
    In2:= IN_TOD2,
    Out=> OUT_T);
```

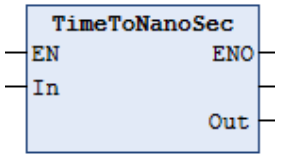
| 表达式 | 类型 | 值 |
|---|---|---|
| IN_TOD1 | TIME_OF_DAY | TOD#19:30:21 |
| IN_TOD2 | TIME_OF_DAY | TOD#10:12:56 |
| OUT_T | TIME | T#9h17m25s |
| XDONE | BOOL | TRUE |

## 6-28. Time nanosecond conversion TimeToNanoSec

(1) Instruction overview

Convert time to nanoseconds.

| Time nanosecond conversion [TimeToNanoSec] | | | |
|---|---|---|---|
| Execution condition | Edge triggering | Suitable model | XS, XSA models |
| Firmware | - | Software | XS StudioV2.2.0 and above |
| Library | XJ_FunctionUtils | | |

| Command | Name | Graphical representation | ST language |
|---|---|---|---|
| TimeToNanoSec | Time nanosecond conversion |  | `TimeToNanoSec(In:= , Out=> );` |

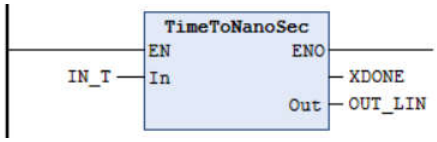
(2) Related variables

| VAR_INPUT | Name | Data type | Effective range | Initial value | Description |
|---|---|---|---|---|---|
| In | Time | TIME | As the data type | T#0s | Time |
| VAR_OUTPUT | Name | Data type | Effective range | Initial value | Description |
| Out | Conversion result | LINT | As the data type | - | Nanoseconds |

(3) Function

- Convert time "In" to nanoseconds.

(4) Program example

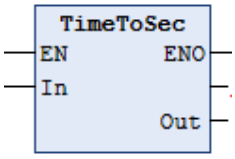
| DEMO | LD | ST |
|---|---|---|
| Define variables | | `VAR`<br>`  IN_T: TIME:= T#3H11M26S;`<br>`  OUT_LIN: LINT;`<br>`  XDONE: BOOL;`<br>`END_VAR` |
| Program example |  | `XDONE:= TimeToNanoSec(`<br>`  In:= IN_T,`<br>`  Out=> OUT_LIN);` |
| Operation result | | |

## 6-29. Time second conversion TimeToSec

(1) Instruction overview

Convert time to seconds.

| Time second conversion [TimeToSec] | | | |
|---|---|---|---|
| Execution condition | Edge triggering | Suitable model | XS, XSA models |
| Firmware | - | Software | XS StudioV2.2.0 and above |
| Library | XJ_FunctionUtils | | |

| Command | Name | Graphical representation | ST language |
|---|---|---|---|
| TimeToSec | Time second conversion |  | `TimeToSec(In:= , Out=> );` |

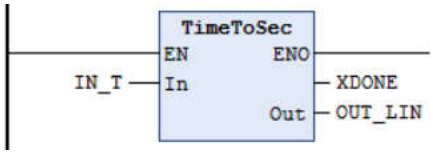
(2) Related variables

| VAR_INPUT | Name | Data type | Effective range | Initial value | Description |
|---|---|---|---|---|---|
| In | Time | TIME | As the data type | T#0s | Time |
| VAR_OUTPUT | Name | Data type | Effective range | Initial value | Description |
| Out | Conversion result | LINT | As the data type | - | Seconds |

(3) Function

- Convert the time "In" to seconds. Values less than 1 second will be rounded off.

(4) Program example

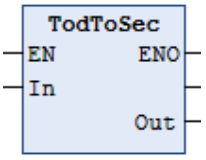
| DEMO | LD | ST |
|---|---|---|
| Define variables | | `VAR`<br>`  IN_T: TIME:= T#3M10S;`<br>`  OUT_LIN: LINT;`<br>`  XDONE: BOOL;`<br>`END_VAR` |
| Program example |  | `XDONE:= TimeToSec(`<br>`  In:= IN_T,`<br>`  Out=> OUT_LIN);` |
| Operation result | | |

## 6-30. Time second conversion TodToSec

(1) Instruction overview

Convert the time to the number of seconds starting from 0:0:00s.

| Time second conversion [TodToSec] | | | |
|---|---|---|---|
| Execution condition | Edge triggering | Suitable model | XS, XSA models |
| Firmware | - | Software | XS StudioV2.2.0 and above |
| Library | XJ_FunctionUtils | | |

| Command | Name | Graphical representation | ST language |
|---|---|---|---|
| TodToSec | Time second conversion |  | `TodToSec(In:= , Out=> );` |

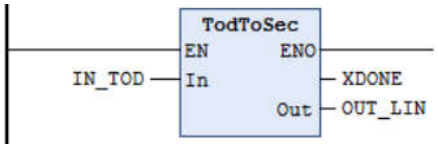
(2) Related variables

| VAR_INPUT | Name | Data type | Effective range | Initial value | Description |
|---|---|---|---|---|---|
| In | Time | TOD | As the data type | TOD#0:0:0 | Time |
| VAR_OUTPUT | Name | Data type | Effective range | Initial value | Description |
| Out | Conversion result | LINT | 0-86399 | - | Seconds |

(3) Function

- Convert the time "In" to the number of seconds starting from 0:0:0:0. The converted value is measured in seconds. Values less than 1 second will be rounded off.

(4) Program example

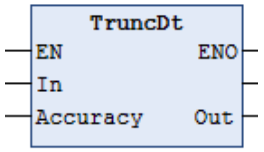
| DEMO | LD | ST |
|---|---|---|
| Define variables | | `VAR`<br>`  IN_TOD: TOD:= TOD#2:1:10;`<br>`  OUT_LIN: LINT;`<br>`  XDONE: BOOL;`<br>`END_VAR` |
| Program example |  | `XDONE:= TodToSec(`<br>`  In:= IN_TOD,`<br>`  Out=> OUT_LIN);` |
| Operation result | | |

## 6-31. Date and time rounding off TruncDt

(1) Instruction overview

Round off values smaller than the specified unit in DT type variables.

| Date and time rounding off [TruncDt] | | | |
|---|---|---|---|
| Execution condition | Edge triggering | Suitable model | XS, XSA models |
| Firmware | - | Software | XS StudioV2.2.0 and above |
| Library | XJ_FunctionUtils | | |

| Command | Name | Graphical representation | ST language |
|---|---|---|---|
| TruncDt | Date and time rounding off |  | TruncDt(<br>In:= ,<br>Accuracy:= ,<br>Out=> ); |

(2) Related variables

| VAR_INPUT | Name | Data type | Effective range | Initial value | Description |
|---|---|---|---|---|---|
| In | Target date and time | DT | As the data type | - | Abandoning objects |
| Accuracy | Rounding unit | _eSUBSEC | As the data type | _SEC | Minimum time unit reserved |
| VAR_OUTPUT | Name | Data type | Effective range | Initial value | Description |
| Out | The processed result | DT | As the data type | - | Date time after rounding off |

(3) Function

- In the object date time "In", values smaller than the rounding unit "Accuracy" will be rounded off, and the rounded value will be saved in the rounded date time "Out".
- The data type of "Accuracy" is the enumeration type _eSUBSEC, and the meaning of the enumeration value is as follows:

| Enum | Meaning |
|---|---|
| _MILLISEC | Millisecond |
| _SEC | Second |
| _MINUTE | Minute |
| _HOUR | Hour |

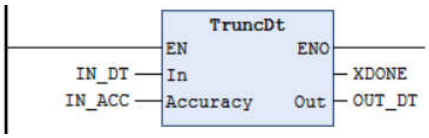
(4) Note

- The enumeration body value cannot be set to _MILLISEC, otherwise the program will encounter an exception and the function return value will be FALSE.

(5) Program example

<table>
<tr><th>DEMO</th><th>LD</th><th>ST</th></tr>
<tr><td>Define variables</td><td colspan="2">

```
VAR
   IN_DT: DT:= DT#2023-9-11-10:30:21;
   IN_ACC: _eSUBSEC:=_eSUBSEC._HOUR;
   OUT_DT: DT;
   XDONE: BOOL;
END_VAR
```

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


</td><td>

```
XDONE:= TruncDt(
    In:= IN_DT,
    Accuracy:= IN_ACC,
    Out=> OUT_DT);
```

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

| 表达式 | 类型 | 值 |
|---|---|---|
| IN_DT | DATE_AND_TIME | DT#2023-9-11-10:30:21 |
| IN_ACC | _ESUBSEC | _HOUR |
| OUT_DT | DATE_AND_TIME | DT#2023-9-11-10:0:0 |
| XDONE | BOOL | TRUE |

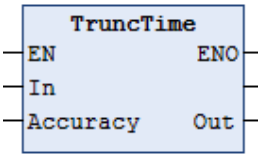
</td></tr>
</table>

## 6-32. Time rounding off TruncTime

(1) Instruction overview

Round off values smaller than the specified unit in TIME type variables.

| Time rounding off [TruncTime] | | | |
|---|---|---|---|
| Execution condition | Edge triggering | Suitable model | XS, XSA models |
| Firmware | - | Software | XS StudioV2.2.0 and above |
| Library | XJ_FunctionUtils | | |

| Command | Name | Graphical representation | ST language |
|---|---|---|---|
| TruncTime | Time rounding off |  | TruncTime(<br>In:= ,<br>Accuracy:= ,<br>Out=> ); |

(2) Related variables

| VAR_INPUT | Name | Data type | Effective range | Initial value | Description |
|---|---|---|---|---|---|
| In | Target time | TIME | As the data type | - | Rounding off target unit time |
| Accuracy | Rounding unit | _eSUBSEC | As the data type | _SEC | Minimum time unit retained |
| VAR_OUTPUT | Name | Data type | Effective range | Initial value | Description |
| Out | Result after processing | TIME | As the data type | - | Time after rounding off |

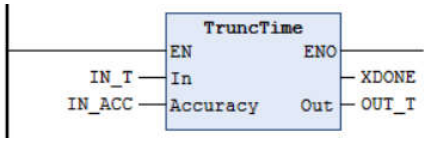
(3) Function

- In the object time "In", values smaller than the rounding unit "Accuracy" will be rounded off, and the rounded value will be saved in the rounding time "Out".
- The data type of "Accuracy" is the enumeration type _eSUBSEC, and the meaning of the enumeration value is as follows:

| Enum | Meaning |
|---|---|
| _MILLISEC | Millisecond |
| _SEC | Second |
| _MINUTE | Minute |
| _HOUR | Hour |

(4) Program example

| DEMO | LD | ST |
|---|---|---|

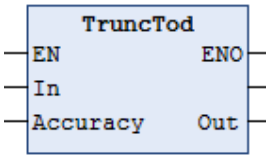
| | |
|---|---|
| Define variables | VAR<br>IN_T: TIME:= T#2D6H32M11S231MS;<br>IN_ACC: _eSUBSEC:= _eSUBSEC._HOUR;<br>OUT_T: TIME;<br>XDONE: BOOL;<br>END_VAR |
| Program example |  <br>XDONE:= TruncTime(<br>In:= IN_T,<br>Accuracy:= IN_ACC,<br>Out=> OUT_T); |
| Operation result | |

## 6-33. Rounding off time TruncTod

(1) Instruction overview

Round off values smaller than the specified unit in TOD type variables.

| Rounding off time [TruncTod] | | | |
|---|---|---|---|
| Execution condition | Edge triggering | Suitable model | XS, XSA models |
| Firmware | - | Software | - |
| Library | XJ_FunctionUtils | | |

| Command | Name | Graphical representation | ST language |
|---|---|---|---|
| TruncTod | Rounding off time |  | TruncTod(<br>In:= ,<br>Accuracy:= ,<br>Out=> ); |

(2) Related variables

| VAR_INPUT | Name | Data type | Effective range | Initial value | Description |
|---|---|---|---|---|---|
| In | Target time | TOD | As the data type | - | Rounding off target unit time |
| Accuracy | Rounding off unit | _eSUBSEC | As the data type | _SEC | Minimum time unit retained |
| VAR_OUTPUT | Name | Data type | Effective range | Initial value | Description |
| Out | Result after | TOD | As the data type | - | Time after rounding off |

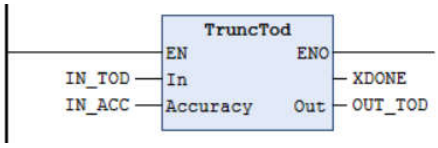
| | | | | | |
|---|---|---|---|---|---|
| | processing | | | | |

(3) Function

- In the object time "In", values smaller than the rounding unit "Accuracy" will be rounded off, and the rounded value will be saved in the time "Out" after rounding.
- The data type of "Accuracy" is the enumeration type _eSUBSEC, and the meaning of the enumeration value is as follows:

| Enum | Meaning |
|---|---|
| _MILLISEC | Millisecond |
| _SEC | Second |
| _MINUTE | Minute |
| _HOUR | Hour |

(4) Program example

| DEMO | LD | ST |
|---|---|---|
| Define variables | VAR<br>IN_TOD: TOD:= TOD#11:12:29;<br>IN_ACC: _eSUBSEC:= _eSUBSEC._HOUR;<br>OUT_TOD: TOD;<br>XDONE: BOOL;<br>END_VAR | |
| Program example |  | XDONE:= TruncTod(<br>In:= IN_TOD,<br>Accuracy:= IN_ACC,<br>Out=> OUT_TOD); |
| Operation result | | |

# 7. Communication instructions

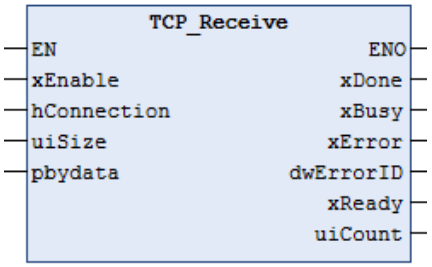
## 7-1. TCP communication

### 7-1-1. TCP communication data receive TCP_Receive

(1) Instruction overview

TCP communication data receive.

| TCP communication data receive [TCP_Receive] | | | |
|---|---|---|---|
| Execution condition | Normally ON/OFF triggering | Suitable model | XS, XSA models |
| Firmware | - | Software | - |
| Library | XJ_FunctionUtils | | |

| Command | Name | Graphical representation | ST language |
|---|---|---|---|
| TCP_Receive | TCP communication data receive |  | TCP_Receive_0(<br>xEnable:= ,<br>hConnection:= ,<br>uiSize:= ,<br>pbydata:= ,<br>xDone=> ,<br>xBusy=> ,<br>xError=> ,<br>dwErrorID=> ,<br>xReady=> ,<br>uiCount=> ); |

(2) Related variables

| VAR_INPUT | Name | Data type | Effective range | Initial value | Description |
|---|---|---|---|---|---|
| xEnable | Command enable | BOOL | FALSE/TRUE | FALSE | Input high-level start command |
| hConnection | Connection handle | DWORD | As the data type | - | Connection handle for TCP communication |
| uiSize | Receiving data size | UINT | As the data type | - | Received data area size (BYTE) |
| pbyData | Point to the receive area | POINTER TO BYTE | - | - | Pointer to the beginning byte of the receiving area |
| VAR_OUTPUT | Name | Data type | Effective range | Initial value | Description |
| xDone | End signal | BOOL | FALSE/TRUE | FALSE | Data reception termination signal |
| xBusy | Busy flag | BOOL | FALSE/TRUE | FALSE | Busy status |
| xError | Error flag | BOOL | FALSE/TRUE | FALSE | Error status |
| dwErrorID | Error code | DWORD | As the data type | - | Error code |
| xReady | Successfully established connection flag | BOOL | FALSE/TRUE | FALSE | Read data from the buffer, and if the data is not empty, set on the flag for one scan |

| | | | | | |
|---|---|---|---|---|---|
| | | | | | cycle |
| uiCount | Connection handle | _UXINT | - | - | Read the actual number of data from the receive buffer |

(3) Function

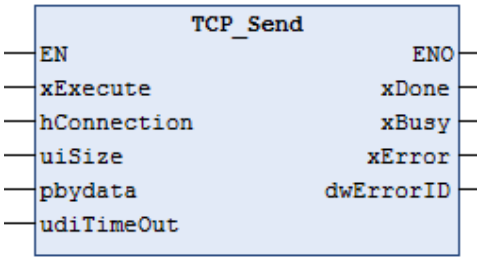
- When the TCP_Receive instruction xEnable detects a high level of power, it will attempt to read data from the TCP communication buffer and set xBusy to True.
- If the data is successfully read, the read data will be placed in an array with pbyData as the first address, and uiCount will output the actual number of data read. xReady will be set to TURE, and the two variables will maintain a scanning cycle.
- If there is an abnormality in network communication during the reading period, causing the reading to fail, xDone will be set to True, and the data reading function will fail. xEnable=True needs to be triggered again. Even trigger the xEnable=True of TCP_Connection or TCP_Client function block again.

## 7-1-2. TCP communication data send TCP_Send

(1) Instruction overview

TCP communication data send.

| TCP communication data send [TCP_Send] | | | |
|---|---|---|---|
| Execution condition | Edge triggering | Suitable model | XS, XSA models |
| Firmware | - | Software | XS StudioV2.2.0 and above |
| Library | XJ_FunctionUtils | | |

| Command | Name | Graphical representation | ST language |
|---|---|---|---|
| TCP_Send | TCP communication data send |  | TCP_Send_0(<br>xExecute:= ,<br>hConnection:= ,<br>uiSize:= ,<br>pbydata:= ,<br>udiTimeOut:= ,<br>xDone=> ,<br>xBusy=> ,<br>xError=> ,<br>dwErrorID=> ); |

(2) Related variables

| VAR_INPUT | Name | Data type | Effective range | Initial value | Description |
|---|---|---|---|---|---|
| xExecute | Command enable | BOOL | FALSE/TRUE | FALSE | Input high-level rising edge start command |
| hConnection | Connection handle | _XWORD | - | - | Connection handle for TCP communication |
| uiSize | Send data size | UINT | As the data type | - | Send data area size (BYTE) |
| pbyData | Point to send area | POINTER TO BYTE | - | - | Pointer to the starting byte of the sending area |
| UditimeOut | Timeout | UDINT | As the data type | 500000 | Send data timeout, unit μs, default is 500ms |
| VAR_OUTPUT | Name | Data type | Effective range | Initial value | Description |
| xDone | Completed signal | BOOL | FALSE/TRUE | FALSE | Data transmission completion signal |
| xBusy | Busy signal | BOOL | FALSE/TRUE | FALSE | Busy status |
| xError | Error flag | BOOL | FALSE/TRUE | FALSE | Error status |
| dwErrorID | Error code | DWORD | As the data type | - | Error code |

(3) Function

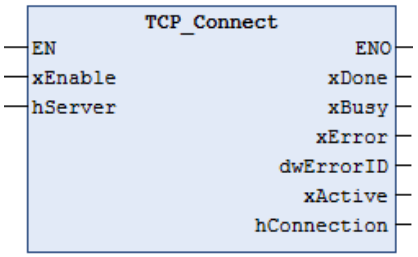
- When the TCP_Send instruction xExecute detects a rising edge, it will send data with a length of uiSize starting from the user set send buffer pbyData to the target device. If the transmission is successful within the timeout period, xDone is set to True.

## 7-1-3. Create TCP connection TCP_Connect

(1) Instruction overview

Create a TCP connection and connect to the server.

| Create TCP connection [TCP_Connect] | | | |
|---|---|---|---|
| Execution condition | Normally ON/OFF triggering | Suitable model | XS, XSA models |
| Firmware | - | Software | XS StudioV2.2.0 and above |
| Library | XJ_FunctionUtils | | |

| Command | Name | Graphical representation | ST language |
|---|---|---|---|
| TCP_Connect | Create a TCP connection and connect to the server |  | TCP_Connect_0(<br>xEnable:= ,<br>hServer:= ,<br>xDone=> ,<br>xBusy=> ,<br>xError=> ,<br>dwErrorID=> ,<br>xActive=> ,<br>hConnection=> ); |

(2) Related variables

| VAR_INPUT | Name | Data type | Effective range | Initial value | Description |
|---|---|---|---|---|---|
| xEnable | Instruction enable | BOOL | FALSE/TRUE | FALSE | Input high-level start command |
| hServer | Server-side handle | _XWORD | - | - | TCP server-side handle |
| VAR_OUTPUT | Name | Data type | Effective range | Initial value | Description |
| xDone | Termination signal | BOOL | FALSE/TRUE | FALSE | Terminate listening signal |
| xBusy | Busy flag | BOOL | FALSE/TRUE | FALSE | Busy status |
| xError | Error flag | BOOL | FALSE/TRUE | FALSE | Error status |
| dwErrorID | Error code | DWORD | As the data type | - | Error code |
| xActive | Connection flag | BOOL | FALSE/TRUE | FALSE | Server side listening to the remote client connection handle flag |
| hConnection | Connection handle | _XWORD | - | - | Connection handle for establishing communication between server and client |

(3) Function

- When the TCP_Connect command xEnable detects high power levels, the local server listens for connection requests from remote clients. After the client successfully connects to the server, a communication connection handle hConnection will be created between the server and the remote client.

(4) Note

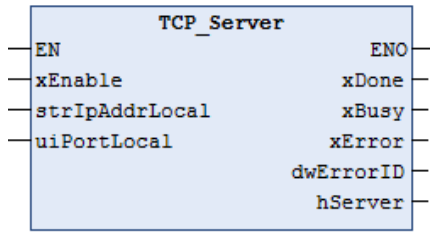
- When communication is established, xActive is set to True, and when communication is disconnected, xActive automatically resets.
- Communication monitoring requires the cooperation of the TCP_Receive function block, and can only be successfully monitored when TCP_Receive.xBusy is set to True.

## 7-1-4. Create TCP server-side communication service TCP_Server

(1) Instruction overview

Create TCP server-side communication service.

| Create TCP server-side communication service [TCP_Server] | | | |
|---|---|---|---|
| Execution condition | Normally ON/OFF triggering | Suitable model | XS, XSA models |
| Firmware | - | Software | XS StudioV2.2.0 and above |
| Library | XJ_FunctionUtils | | |

| Command | Name | Graphical representation | ST language |
|---|---|---|---|
| TCP_Server | Create TCP server-side communication service |  | TCP_Server_0(<br>xEnable:= ,<br>strIpAddrLocal:= ,<br>uiPortLocal:= ,<br>xDone=> ,<br>xBusy=> ,<br>xError=> ,<br>dwErrorID=> ,<br>hServer=> ); |

(2) Related variables

| VAR_INPUT | Name | Data type | Effective range | Initial value | Description |
|---|---|---|---|---|---|
| xEnable | Instruction enable | BOOL | FALSE/TRUE | FALSE | Input high-level start command |
| strIpAddrLocal | Local IP address | STRING | As the data type | - | The IP address of the local PLC communication network port |
| uiPortLocal | Local port | UINT | As the data type | - | Port number for local communication |
| VAR_OUTPUT | Name | Data type | Effective range | Initial value | Description |
| xDone | Termination Flag | BOOL | FALSE/TRUE | FALSE | Server end communication termination flag |
| xBusy | Busy flag | BOOL | FALSE/TRUE | FALSE | Busy status |
| xError | Error flag | BOOL | FALSE/TRUE | FALSE | Error status |
| dwErrorID | Error code | DWORD | As the data type | - | Error code |
| hServer | Server handle | _XWORD | - | - | TCP server handle |

(3) Function

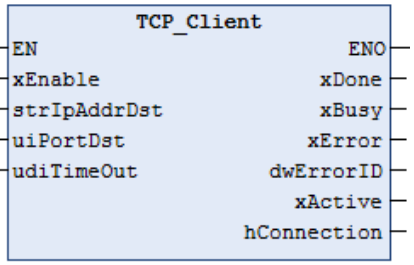
- Create a TCP server-side communication service instruction. When the TCP_Server instruction xEnable is true, a valid handle for TCP communication between the local server and the remote client will be created (the value of hServer is not equal to 0), and xBusy will remain true. If there is an error in the service related function during the communication process, xDone will be set to True, and the server-side related function blocks cannot be used, and TCP_Server needs to be triggered again.

## 7-1-5. Create TCP client communication service TCP_Client

(1) Instruction overview

Create TCP client communication service.

| Create TCP client communication service [TCP_Client] | | | |
|---|---|---|---|
| Execution condition | Normally ON/OFF triggering | Suitable model | XS, XSA models |
| Firmware | - | Software | XS StudioV2.2.0 and above |
| Library | XJ_FunctionUtils | | |

| Command | Name | Graphical representation | ST language |
|---|---|---|---|
| TCP_Client | Create TCP client communication service |  | TCP_Client_0(<br>xEnable:= ,<br>strIpAddrDst:= ,<br>uiPortDst:= ,<br>udiTimeOut:= ,<br>xDone=> ,<br>xBusy=> ,<br>xError=> ,<br>dwErrorID=> ,<br>xActive=> ,<br>hConnection=> ); |

(2) Related variables

| VAR_INPUT | Name | Data type | Effective range | Initial value | Description |
|---|---|---|---|---|---|
| xEnable | Instruction enable | BOOL | FALSE/TRUE | FALSE | Input high-level start command |
| strIpAddrDst | Remote device IP address | STRING | As the data type | - | Remote device IP address |
| uiPortDst | Remote device port number | UINT | As the data type | - | Remote device port number |
| udiTimeOut | Timeout | UDINT | As the data type | 500000 | The timeout time for requesting a connection, in μs, defaults to 500ms |
| VAR_OUTPUT | Name | Data type | Effective range | Initial value | Description |
| xDone | Termination signal | BOOL | FALSE/TRUE | FALSE | Client communication termination flag |
| xBusy | Busy flag | BOOL | FALSE/TRUE | FALSE | Busy status |

| xError | Error flag | BOOL | FALSE/TRUE | FALSE | Error status |
|---|---|---|---|---|---|
| dwErrorID | Error code | DWORD | As the data type | - | Error code |
| xActive | Successfully established connection flag | BOOL | FALSE/TRUE | FALSE | Flag bit for successful connection between client and remote server |
| hConnection | Connection handle | _XWORD | - | - | Connection handle for establishing communication between server and client |

(3) Function

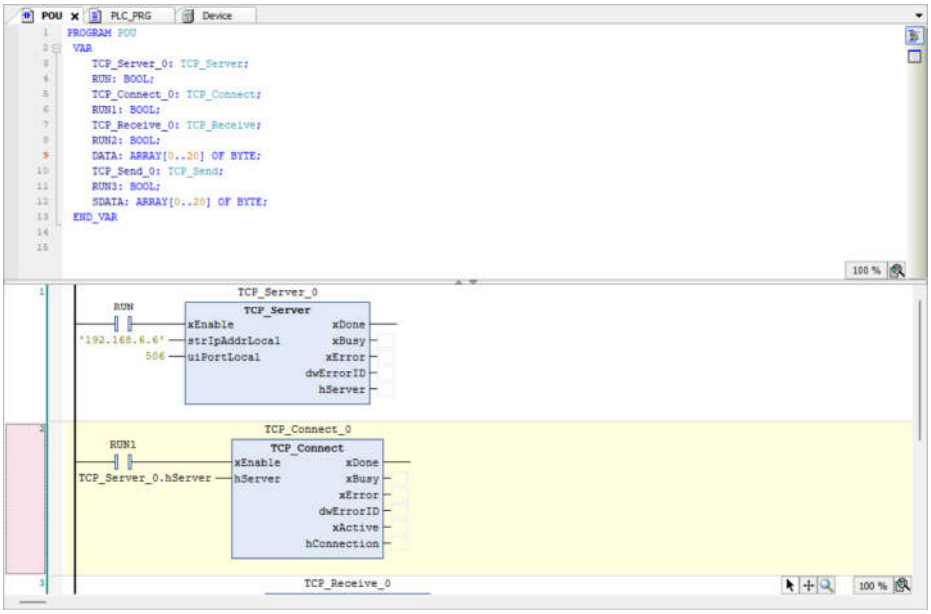
- Create a TCP client communication service instruction. When the TCP_Client instruction xEnable detects high level power, the local client requests to connect to the remote server. After the client successfully connects to the server, a communication connection handle hConnection will be created between the server and the remote client, and xActive will be set to True.

## 7-1-6. Example of TCP communication instruction program

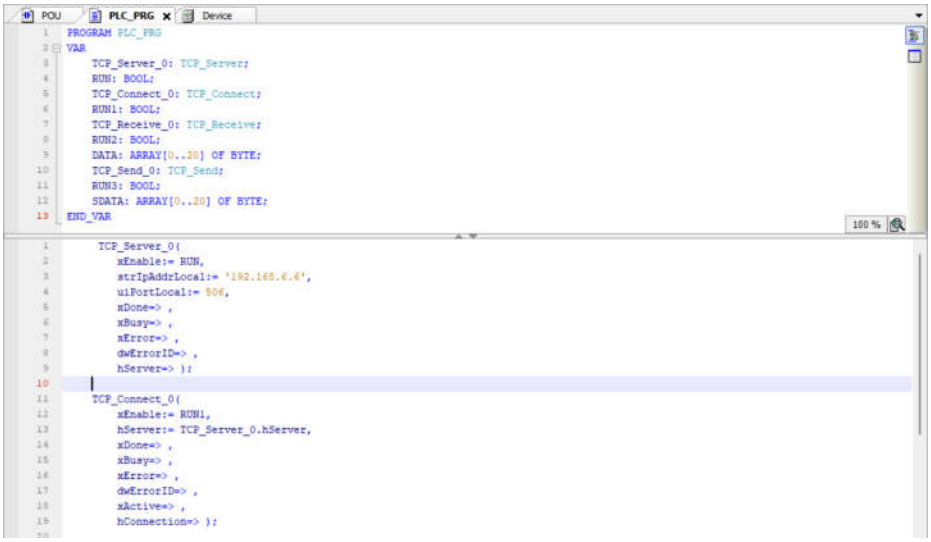
**XSDH-60A32 as server, debugging assistant as client**

The IP address of XSDH-60A32 is 192.168.6.6

1. Create TCP server-side communication services and establish connections between clients and servers; Trigger the TCP Server and TCP Connect instructions, where the input pin hServer of the TCP Connect instruction is the output pin hServer of TCP Server.
2. When triggering the debugging assistant connection, be sure to keep the IP address and port number consistent; The port number in the example is 506.




LD form

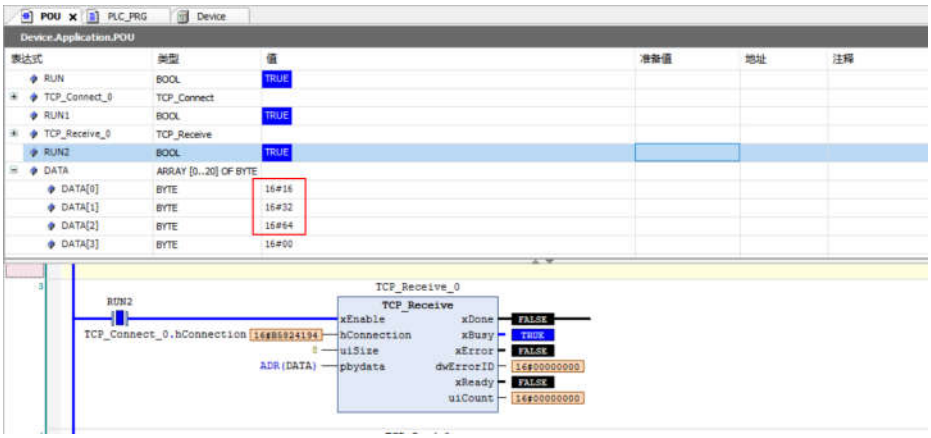



ST form

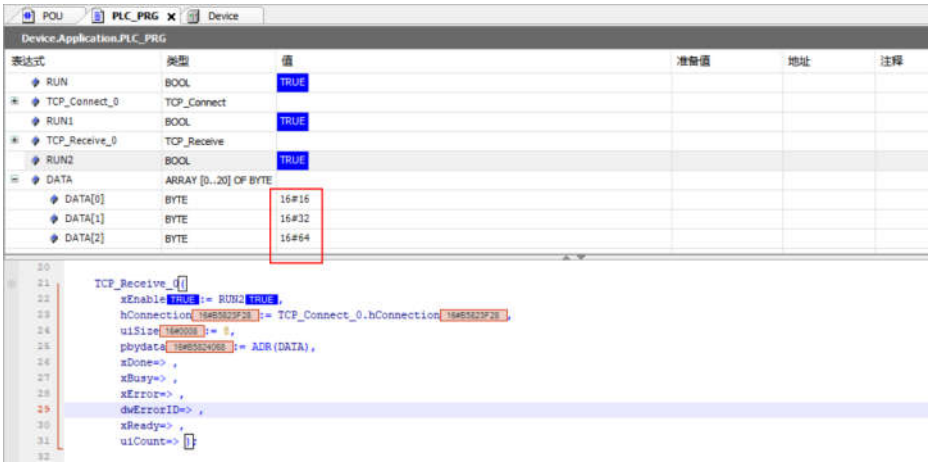
3. The server receives data sent by the client debugging assistant.

① Sending hexadecimal data 16#16 16#32 16#64 using the debugging assistant.

② Trigger the TCP_Receive instruction, and the received data is stored at the address pointed to by the pbyData pointer, as shown in the following figure. Received 16#16, 16#32, 16#64.



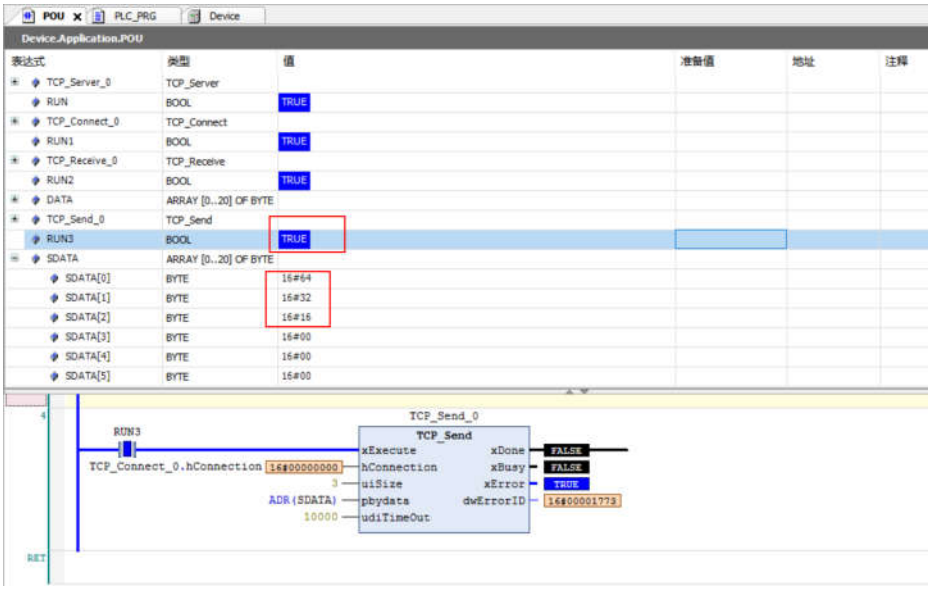
LD form



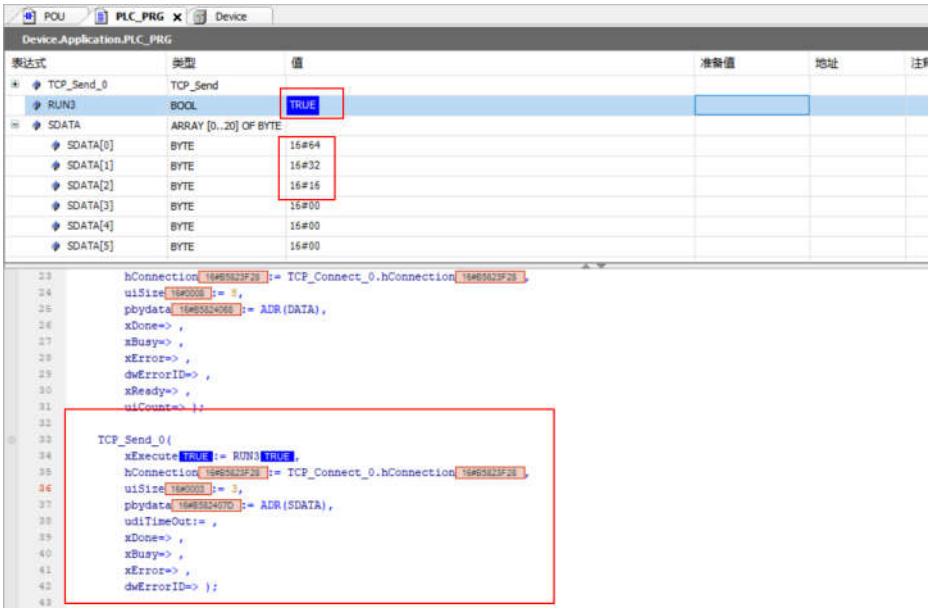

ST form

4. The server sends data to the client debugging assistant.

① Trigger the TCP_Send command to send data stored at the address pointed to by the pbyData pointer, as shown in the following figure. Send data 16#64 16#32 16#16.




LD form



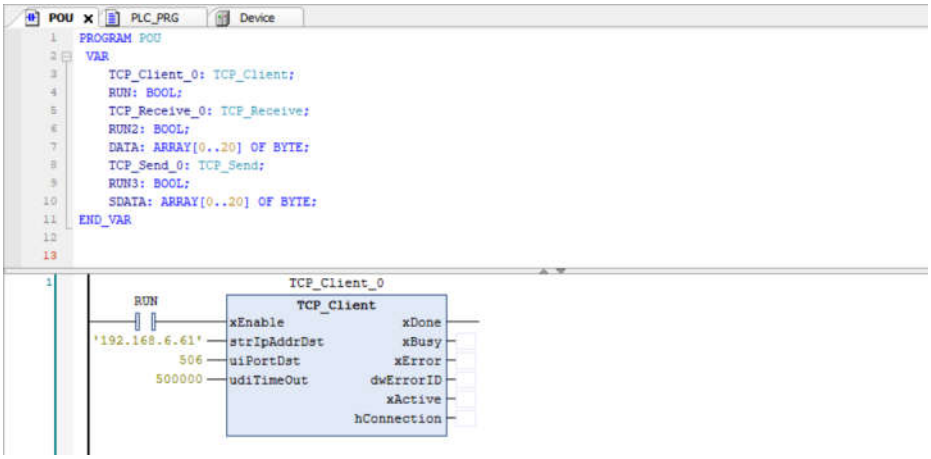

ST form

② The following figure shows the client debugging assistant receiving 16#64 16#32 16#16 data sent by the PLC server.

**XSDH-60A32 as the client, debugging assistant as the server**

1. Create a TCP client communication service and establish a connection with the debugging assistant.

① Trigger the TCP_Client command, and note that the IP address and port number are both the remote server address and port (the debugging assistant IP '192.168.6.1' is the IP address of the PC port connected to the PLC); Open the debugging assistant server before triggering the command.



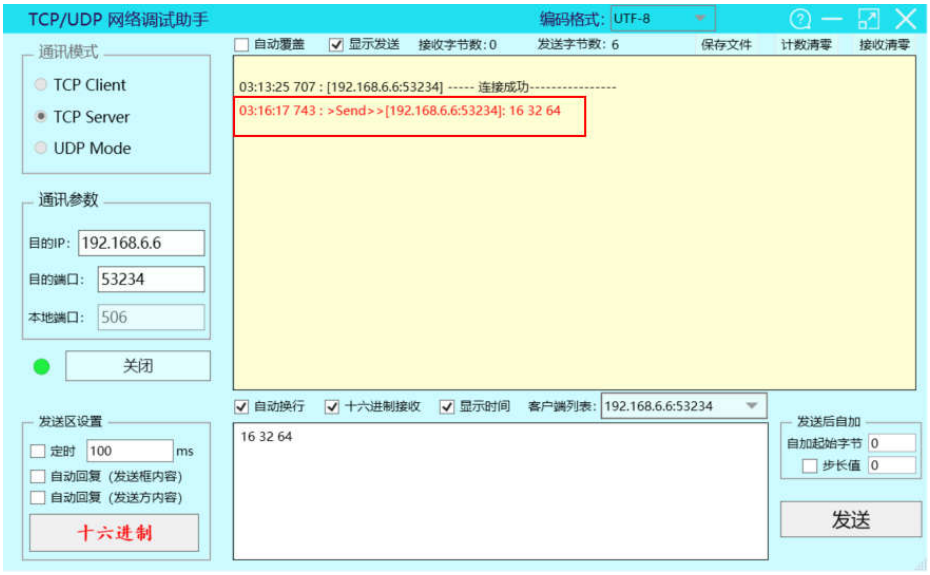

LD form

ST form

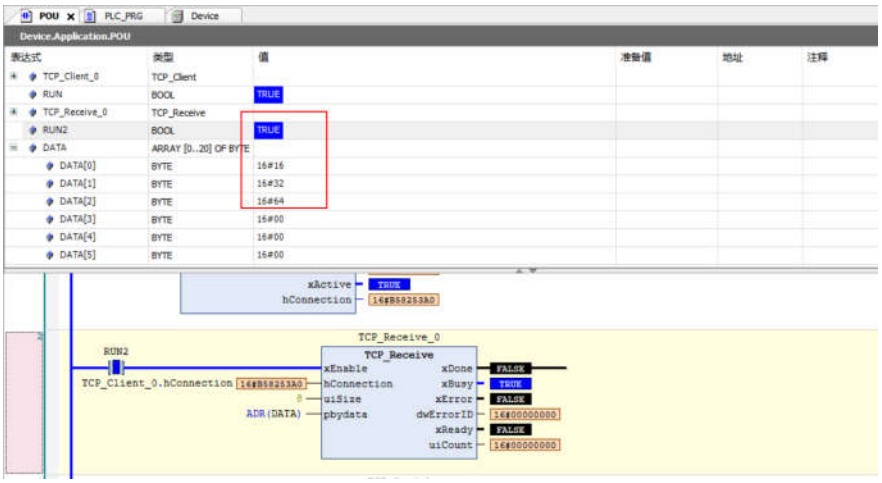
2. First, open the debugging assistant server, and then trigger the TCP_Client command mentioned above to create a client communication service. The connection has been successfully established as shown in the following figure:

3. The client receives data sent by the server-side debugging assistant.

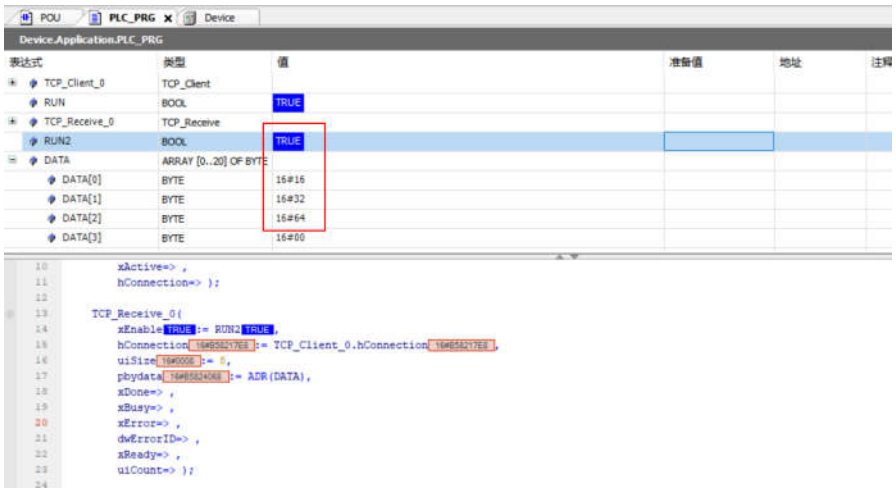
① Sending hexadecimal data 16#16 16#32 16#64 using the debugging assistant.




② Trigger the TCP_Receive instruction, and the received data is stored at the address pointed to by the pbyData pointer; the client received data 16#16 16#32 16#64 sent by the server, as shown in the following figure, and the reception was successful.



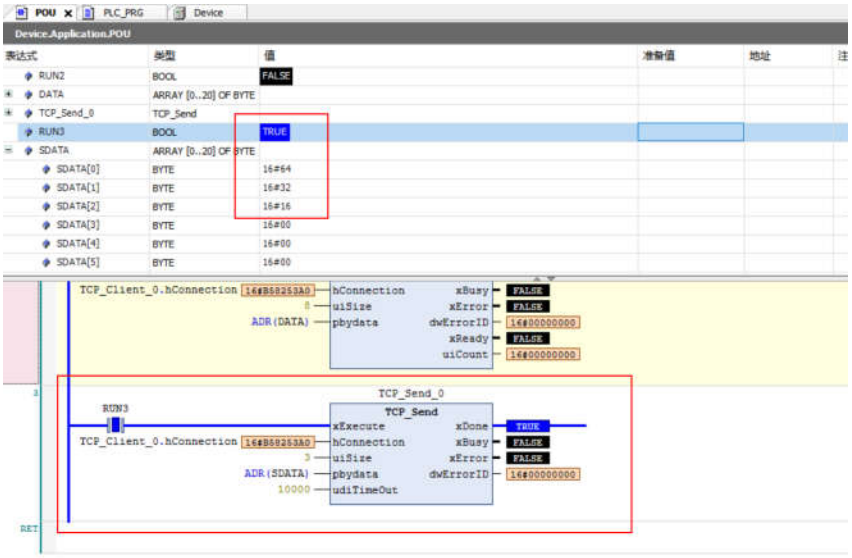

LD form



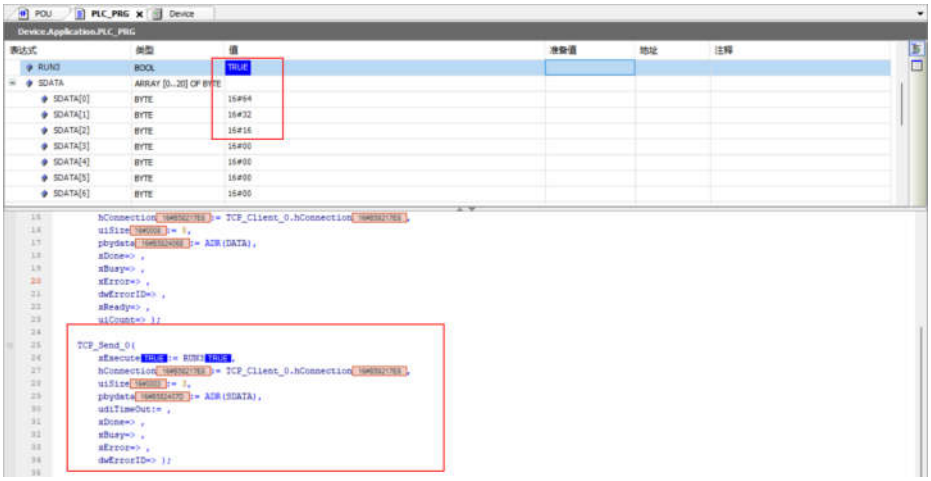

ST form

4. The client sends data to the server-side debugging assistant.

① Trigger the TCP_Send instruction to send data stored at the address pointed to by the pbyData pointer.

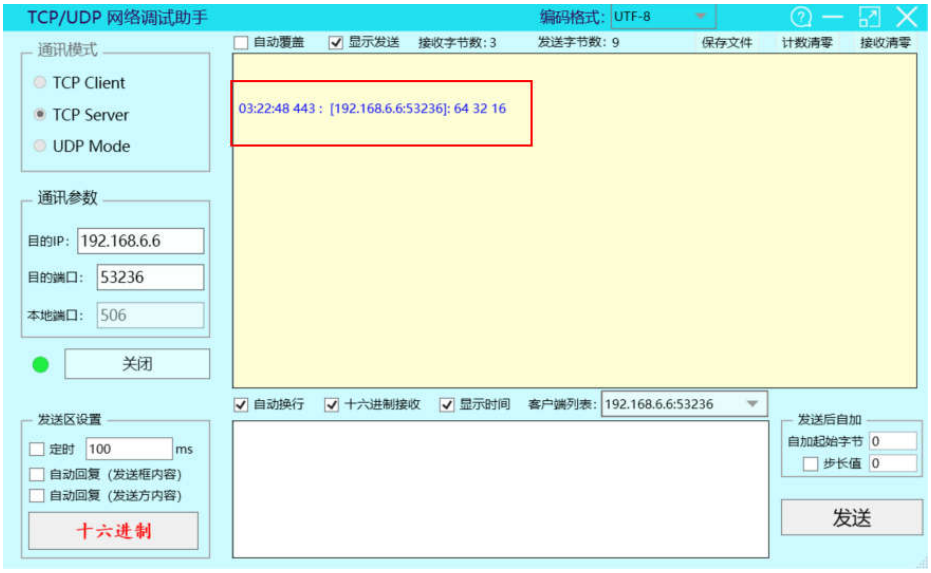



LD form




ST form

② The debugging assistant on the server side received data 16#64 16#32 16#16 sent by the PLC client, as shown in the following figure, and the transmission was successful.


TCP/UDP 网络调试助手
编码格式: UTF-8
通讯模式
TCP Client
TCP Server
UDP Mode
通讯参数
目的IP: 192.168.6.6
目的端口: 53236
本地端口: 506
关闭
自动覆盖
显示发送
接收字节数: 3
发送字节数: 9
保存文件
计数清零
接收清零
03:22:48 443 : [192.168.6.6:53236]: 64 32 16
自动换行
十六进制接收
显示时间
客户端列表: 192.168.6.6:53236
发送区设置
定时 100 ms
自动回复 (发送框内容)
自动回复 (发送方内容)
十六进制
发送后自加
自加起始字节 0
步长值 0
发送

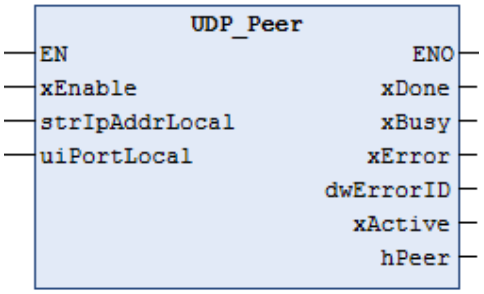
## 7-2. UDP communication

### 7-2-1. Create UDP communication connecting UDP_Peer

(1) Instruction overview

Create UDP communication connection.

| Create UDP communication connection [UDP_Peer] | | | |
|---|---|---|---|
| Execution condition | Normally open/close triggering | Suitable model | XS, XSA models |
| Firmware | - | Software | XS StudioV2.2.0 and above |
| Library | XJ_FunctionUtils | | |

| Command | Name | Graph | ST |
|---|---|---|---|
| UDP_Peer | Create UDP communication connection |  | UDP_Peer_0(<br>xEnable:= ,<br>strIpAddrLocal:= ,<br>uiPortLocal:= ,<br>xDone=> ,<br>xBusy=> ,<br>xError=> ,<br>dwErrorID=> ,<br>xActive=> ,<br>hPeer=> ); |

(2) Related variables

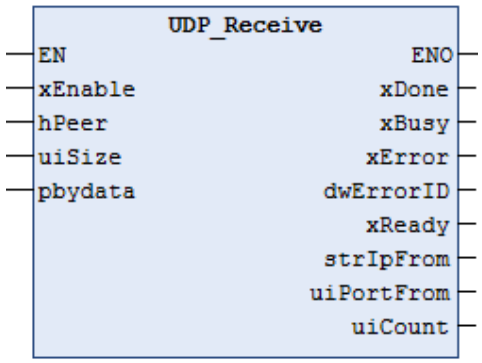
| VAR_INPUT | Name | Data type | Valid range | Initial value | Description |
|---|---|---|---|---|---|
| xEnable | Command enable | BOOL | FALSE/TRUE | FALSE | Input high-level to start command |
| strIpAddrLocal | Local IP address | STRING(80) | As the data type | - | The IP address of the local PLC communication network port |
| uiPortLocal | Local port no. | UINT | As the data type | - | Port number for local communication |
| VAR_OUTPUT | Name | Data type | Valid range | Initial value | Description |
| xDone | Termination Flag | BOOL | FALSE/TRUE | FALSE | Communication termination flag |
| xBusy | Busy flag | BOOL | FALSE/TRUE | FALSE | Busy flag |
| xError | Error flag | BOOL | FALSE/TRUE | FALSE | Error flag |
| dwErrorID | Error code | DWORD | As the data type | - | Error code |
| xActive | Communication success flag | BOOL | FALSE/TRUE | FALSE | Create UDP communication success flag |
| hpeer | Communication handle | _XWORD | - | - | UDP communication handle |

## 7-2-2. UDP communication data reception UDP_Receive

(1) Instruction overview

UDP communication data reception.

| UDP communication data reception [UDP_Receive] | | | |
|---|---|---|---|
| Execution condition | Normally open/close triggering | Suitable model | XS, XSA models |
| Firmware | - | Software | XS StudioV2.2.0 and above |
| Library | XJ_FunctionUtils | | |

| Command | Name | Graph | ST |
|---|---|---|---|
| UDP_Receive | UDP communication data reception |  | UDP_Receive_0(<br>xEnable:= ,<br>hPeer:= ,<br>uiSize:= ,<br>pbydata:= ,<br>xDone=> ,<br>xBusy=> ,<br>xError=> ,<br>dwErrorID=> ,<br>xReady=> ,<br>strIpFrom=> ,<br>uiPortFrom=> ,<br>uiCount=> ); |

(2) Related variables

| VAR_INPUT | Name | Data type | Valid range | Initial value | Description |
|---|---|---|---|---|---|
| xEnable | Command enable | BOOL | FALSE/TRUE | FALSE | Input high-level to start command |
| hPeer | Communication handle | _XWORD | - | - | UDP communication handle |
| uiSize | Receive data size | UINT | As the data type | - | Receive data area size (BYTE) |
| pbyData | Point to receiving area | POINTER TO BYTE | - | - | Pointer to the beginning byte of the receiving area |
| VAR_OUTPUT | Name | Data type | Valid range | Initial value | Description |
| xDone | Termination Flag | BOOL | FALSE/TRUE | FALSE | Data reception termination signal |
| xBusy | Busy flag | BOOL | FALSE/TRUE | FALSE | Busy status |
| xError | Error flag | BOOL | FALSE/TRUE | FALSE | Error status |
| dwErrorID | Error code | DWORD | As the data type | - | Error code |
| xReady | Received successfully flag | BOOL | FALSE/TRUE | FALSE | Read data from the buffer, and if the data is not empty, set the flag for one scan cycle |
| strIpFrom | Current packet | STRING(80) | As the data | - | Current packet source IP |

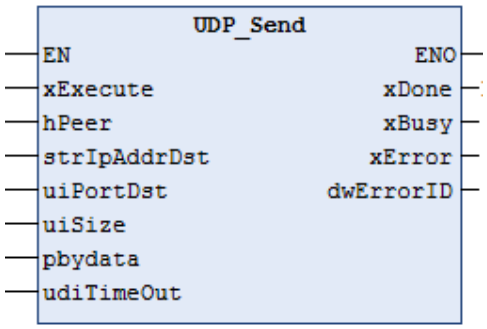
| | | | | | |
|---|---|---|---|---|---|
| | source IP | | type | | |
| uiPortFrom | Current packet source port | UINT | As the data type | - | The source port of the current packet |
| uiCount | Number of data | _UXINT | - | - | Read the actual number of data from the buffer |

## 7-2-3. UDP communication data send UDP_Send

(1) Instruction overview

UDP communication data send.

| UDP communication data send [UDP_Send] | | | |
|---|---|---|---|
| Execution condition | Edge triggering | Suitable model | XS, XSA models |
| Firmware | - | Software | XS StudioV2.2.0 and above |
| Library | XJ_FunctionUtils | | |

| Command | Name | Graph | ST |
|---|---|---|---|
| UDP_Send | UDP communication data send |  | UDP_Send_0(<br>xExecute:= ,<br>hPeer:= ,<br>strIpAddrDst:= ,<br>uiPortDst:= ,<br>uiSize:= ,<br>pbydata:= ,<br>udiTimeOut:= ,<br>xDone=> ,<br>xBusy=> ,<br>xError=> ,<br>dwErrorID=> ); |

(2) Related variables

| VAR_INPUT | Name | Data type | Valid range | Initial value | Description |
|---|---|---|---|---|---|
| xExecute | Command enable | BOOL | FALSE/TRUE | FALSE | Input high-level rising edge to start command |
| hPeer | Communication handle | _XWORD | - | - | UDP communication handle |
| strIpAddrDst | Target IP | STRING(80) | As the data type | - | Target IP for sending data |
| uiPortDst | Target port | UINT | As the data type | - | Target port for sending data |
| uiSize | Send data size | UINT | As the data type | - | Send data area size (BYTE) |
| pbyData | Point to the sending area | POINTER TO BYTE | - | - | Pointer to the starting byte of the sending area |
| UditimeOut | Timeout time | UDINT | As the data type | 500000 | Send data timeout, unit μs, default is 500ms |

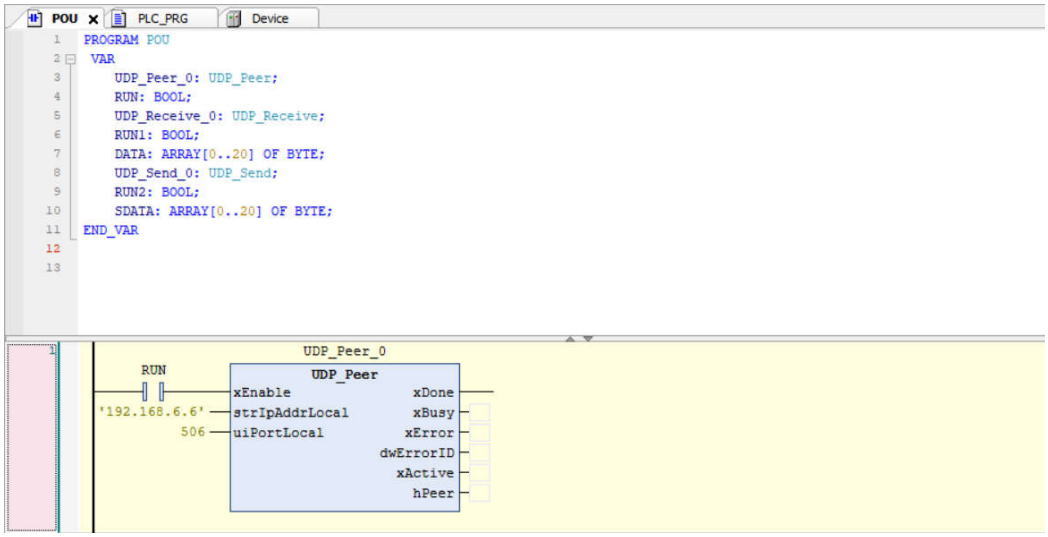
| VAR_OUTPUT | Name | Data type | Valid range | Initial value | Description |
|---|---|---|---|---|---|
| xDone | Completion signal | BOOL | FALSE/TRUE | FALSE | Data sending completion signal |
| xBusy | Busy flag | BOOL | FALSE/TRUE | FALSE | Busy status |
| xError | Error flag | BOOL | FALSE/TRUE | FALSE | Error status |
| dwErrorID | Error code | DWORD | As the data type | - | Error code |

## 7-2-4. Example of UDP communication

**XSDH-60A32 establishes UDP communication with computer debugging assistant**

Wherein: debugging assistant (PC port) IP address 192.168.6.61, port number 506
PLC end IP address 192.168.6.6, port number 506

1. Establish a UDP connection between the PLC and the debugging assistant. In the UDP communication connection instruction UDP_Peer, the strIpAddrLocal input pin is the local PLC communication network port address, and the uiPortLocal input pin is the local PLC port number.



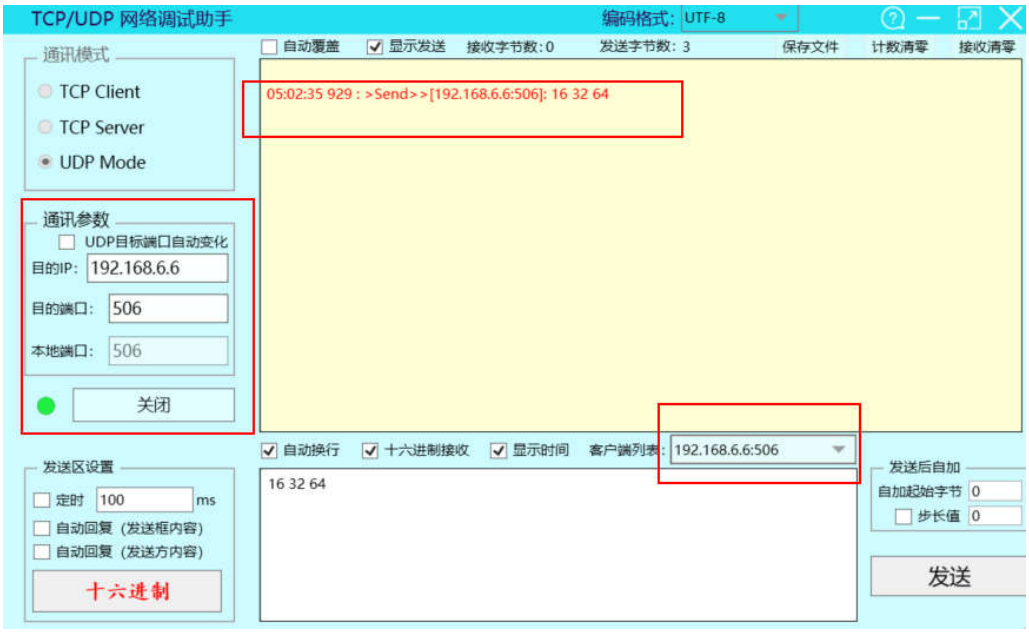

LD form

```
PROGRAM PLC_PRG
VAR
    UDP_Peer_0: UDP_Peer;
    RUN: BOOL;
    UDP_Receive_0: UDP_Receive;
    RUN1: BOOL;
    DATA: ARRAY[0..20] OF BYTE;
    UDP_Send_0: UDP_Send;
    RUN2: BOOL;
    SDATA: ARRAY[0..20] OF BYTE;
END_VAR
```

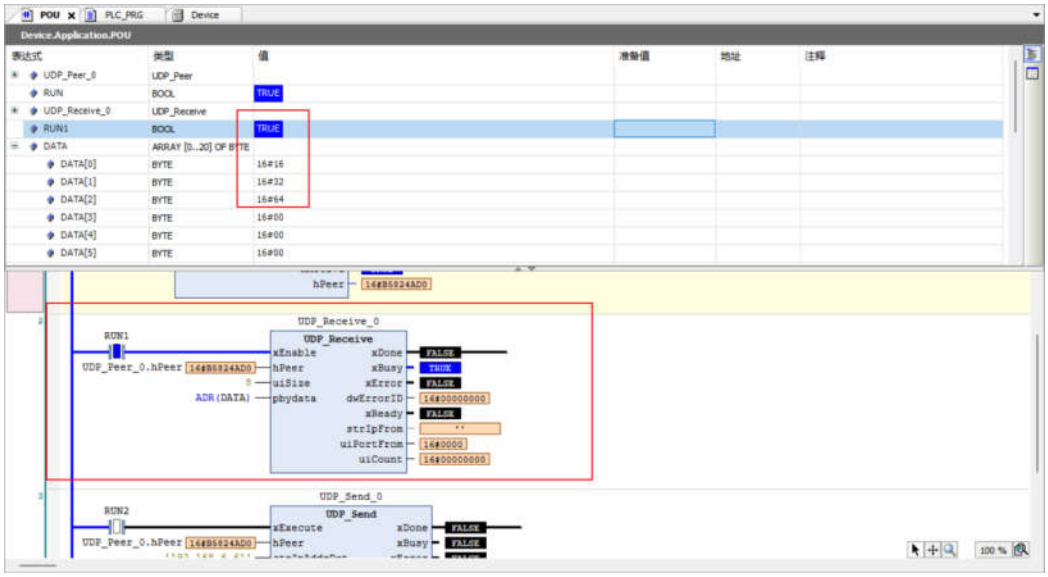
```
UDP_Peer_0(
    xEnable:= RUN,
    strIpAddrLocal:= '192.168.6.6',
    uiPortLocal:= 506,
    xDone=> ,
    xBusy=> ,
    xError=> ,
    dwErrorID=> ,
    xActive=> ,
    hPeer=> );
```

ST form

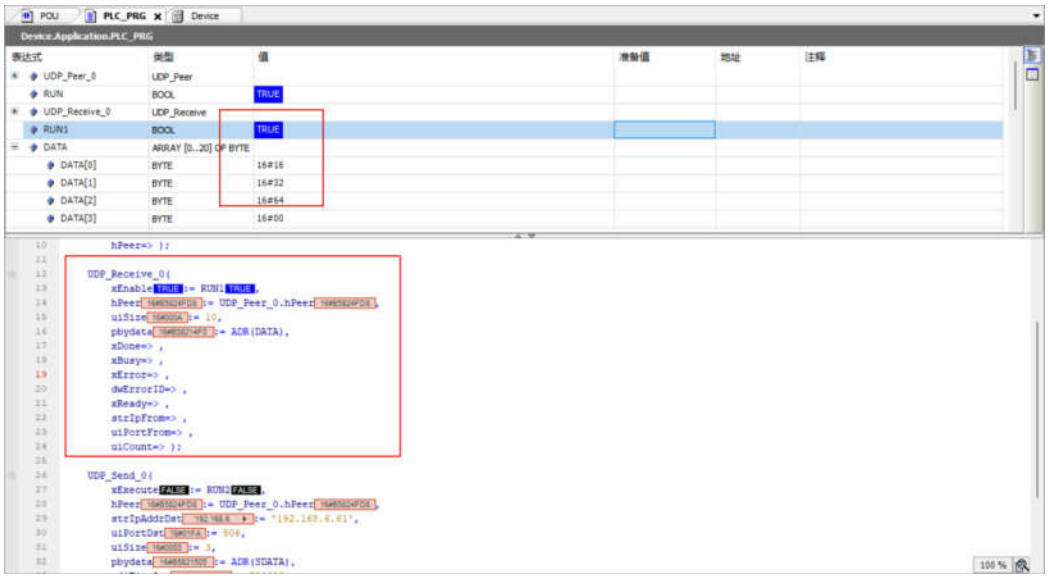
2. Receive data, the PLC receives data sent by the PC debugging assistant.




Triggering the UDP_Receive instruction, the received data is stored at the address pointed to by the pbyData pointer, as shown in the following figure. Received 16#16, 16#32, 16#64, and received successfully.



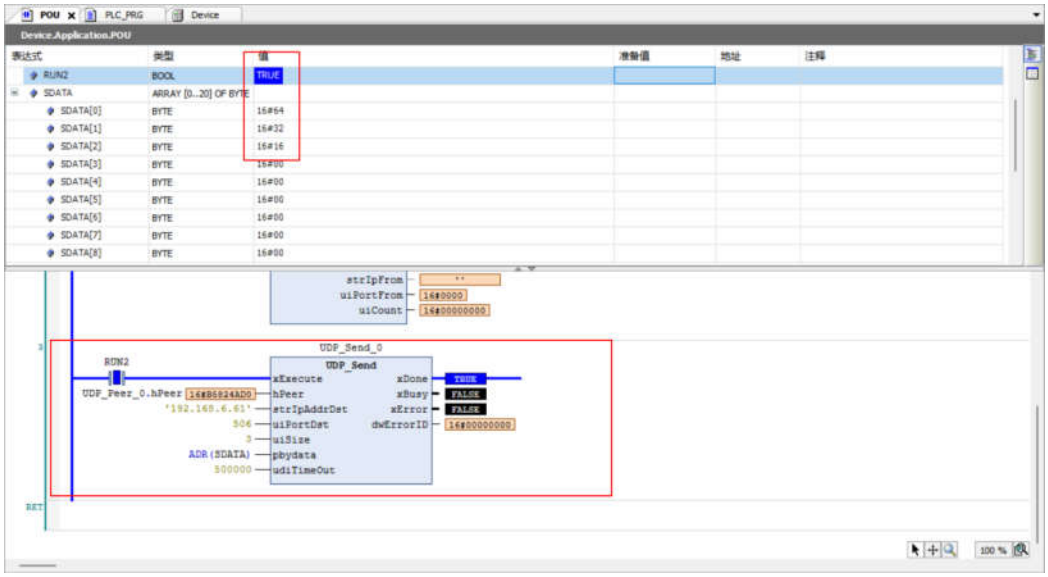

LD form



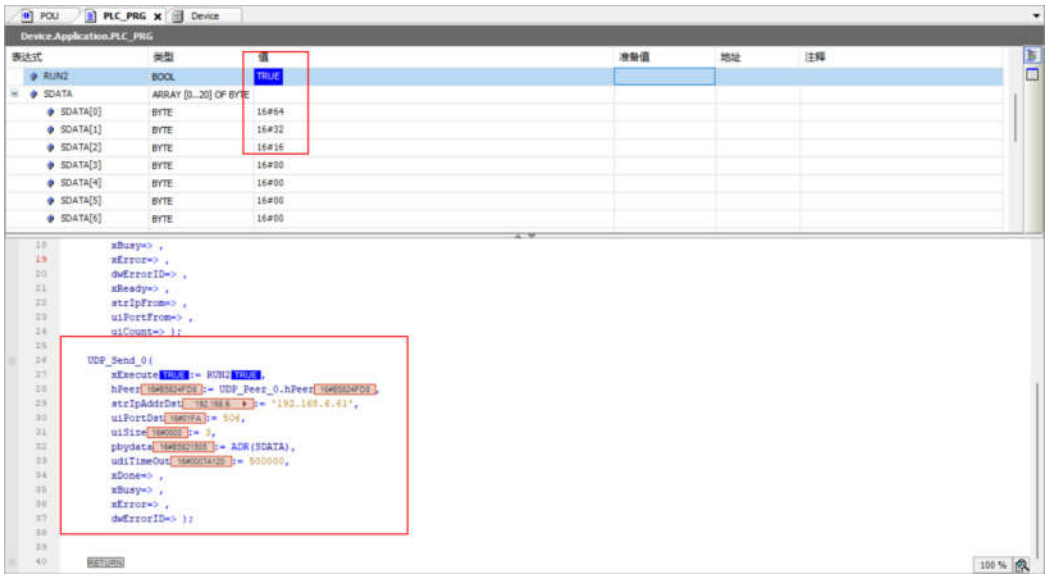

ST form

3. Send data, PLC sends data to PC debugging assistant.

Trigger the UDP_Send command to send data stored at the address pointed to by the pbyData pointer, as shown in the following figure. Send data 16#64 16#32 16#16.

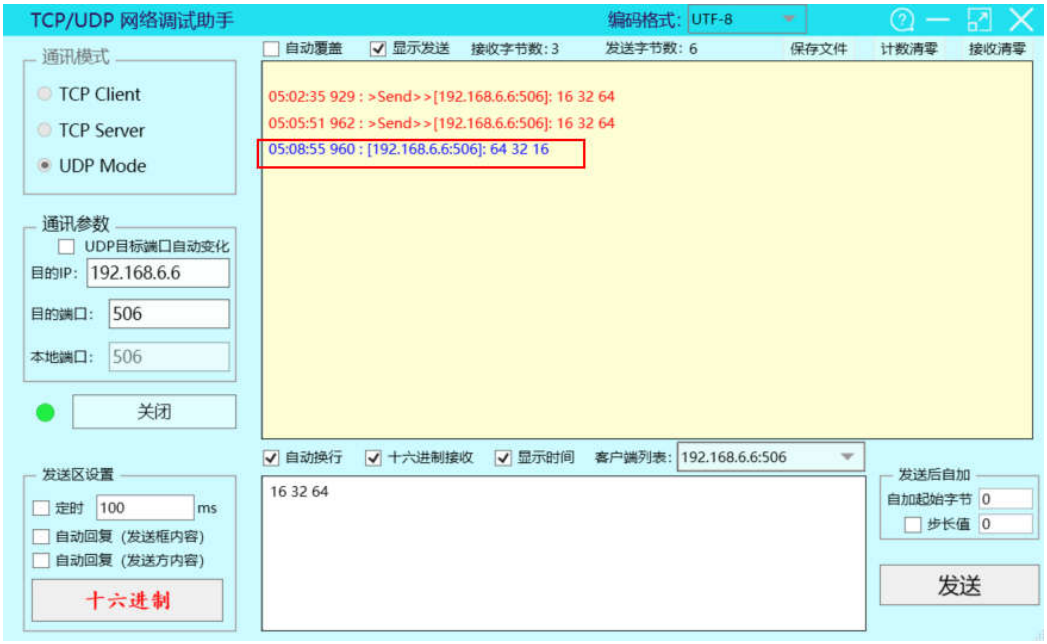



LD form




ST form

As shown in the figure below, the PC debugging assistant received data 16#64 16#32 16#16, and the PLC data was successfully sent.




7-3. TCP/UDP communication error code

| Error ID | Error code | Description |
|---|---|---|
| NO_ERROR | 0 | No error |
| FIRST_ERROR | 6000 | Reserved |
| TIME_OUT | 6001 | Timeout |
| INVALID_ADDR | 6002 | IP invalid |
| INVALID_HANDLE | 6003 | Handle invalid |
| INVALID_DATAPOINTER | 6004 | Data pointer invalid |
| INVALID_DATASIZE | 6005 | Data length invalid |
| UDP_RECEIVE_ERROR | 6006 | UDP receive error |
| UDP_SEND_ERROR | 6007 | UDP send error |
| UDP_SEND_NOT_COMPLETE | 6008 | UDP sending not complete |
| UDP_OPEN_ERROR | 6009 | UDP open error |
| UDP_CLOSE_ERROR | 6010 | UDP close error |
| TCP_SEND_ERROR | 6011 | TCP send error |
| TCP_RECEIVE_ERROR | 6012 | TCP receive error |
| TCP_OPEN_ERROR | 6013 | TCP open error |
| TCP_CONNECT_ERROR | 6014 | TCP connection error |
| TCP_CLOSE_ERROR | 6015 | TCP close error |
| TCP_SERVER_ERROR | 6016 | TCP server error |
| WRONG_PARAMETER | 6017 | Parameter error |
| ERROR_UNKNOWN | 6018 | Unknown error |
| TCP_NO_CONNECTION | 6019 | TCP no connection |
| IOCTL_ERROR | 6020 | IOCTL error |
| FIRST_MF | 6050 | Reserved |
| LAST_ERROR | 6099 | Last time error |

## 7-4. CAN free format communication

### 7-4-1. CAN free format communication specification

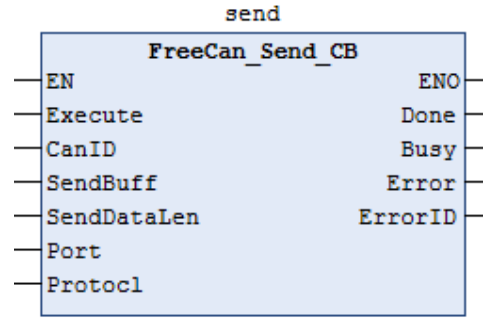
| Item | CAN interface features |
|---|---|
| Connector | 3-PIN terminal |
| Transmission cable | Two communication cables, one shielding cable, and one grounding cable |
| Supported standard protocols | CAN 2.0A (11-bit identifier)<br>CAN 2.0B (29-bit identifier) |
| Baud rate | 10,20,50,100,125,250,500,800,1000 kbit/s |
| Protocol | CANBUS 2.0A, CANBUS 2.0B |

### 7-4-2. CAN free format send FreeCan_Send_CB

(1) Instruction overview

Write the contents of the specified register in this machine to the CAN device with the specified ID number.

| CANBUS communication send data [ FreeCan_Send_CB] | | | |
|---|---|---|---|
| Execution condition | Edge triggering | Suitable model | XSLH-30A32, XSF |
| Firmware | XSLH-30A32: None<br>XSF: V2.3.0 and above | Software | - |
| Library | XJ_CANbus | | |

| Command | Name | Graph | ST |
|---|---|---|---|
| FreeCan_Send_CB | CAN free format send |  | FreeCan_Send_CB_0(<br>Execute:= ,<br>CanID:= ,<br>SendBuff:= ,<br>SendDataLen:= ,<br>Port:= ,<br>Protocl:= ,<br>Done=> ,<br>Busy=> ,<br>Error=> ,<br>ErrorID=> ); |

(2) Related variables

| VAR_INPUT | Name | Data type | Valid range | Initial value | Description |
|---|---|---|---|---|---|
| Execute | Edge triggering | BOOL | FALSE/TRUE | FALSE | Command conduction conditions |
| CanID | Specify remote communication ID number | UDINT | CAN2.0A：0-2047<br>CAN2.0B：0-536870911 | 0 | Remote communication ID number |
| SendBuff | Specify the first | POINTER | As the data type | - | The first address |

| | | | | | |
|---|---|---|---|---|---|
| | address for sending variables | TO WORD | | | for sending variables |
| SendDataLen | Specify the number of bytes to send data | BYTE | As the data type | - | Bytes of data sent |
| Port | Specified CAN port | BYTE | As the data type | - | Communication CAN port |
| Protocl | Specified CANBUS protocol | BYTE | As the data type | - | CANBUS communication protocol |
| VAR_OUTPUT | Name | Data type | Valid range | Initial value | Description |
| Done | Processing completion flag | BOOL | FALSE/TRUE | FALSE | Processing completion flag |
| Busy | Function block activation flag | BOOL | FALSE/TRUE | FALSE | Function block activation flag |
| Error | Error flag | BOOL | FALSE/TRUE | FALSE | Error flag |
| ErrorID | Error code | XJ_FreeCan_Error_Type | As the data type | - | Error code |

(3) Function

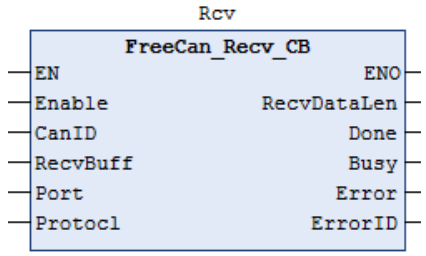
- Execute: edge triggered to send data.
- CanID: CAN2.0A: corresponds to the hexadecimal range H000- H7FF; CAN2.0B: corresponds to the hexadecimal range H0000 0000- H1FFF FFFF. It is necessary to limit the corresponding ID range based on the selected protocol type.
- SendBuf: specifies the first address of the variable for sending cached data.
- SendDataLen: The number of bytes sent is K0~K8, and the input byte count needs to be limited to a maximum of 8 bytes.
- Port: Communication CAN port range, K1: PLC first CAN channel, K2: PLC second CAN channel(reserved)
- Protocol: CAN communication protocol K0: CAN2.0A protocol, K1: CAN2.0B protocol.
- Done: instruction processing completion flag, set Done to true when sending is completed, and set FALSE when the falling edge of Execute is detected.
- Busy: Executing function block.
- Error: True: An error occurred, the function block terminated the operation, FALSE: No error.
- ErrorID: Error code.

## 7-4-3. CAN free format receive FreeCan_Recv_CB

(1) Instruction overview

The instruction to receive data from the CAN device with the specified ID number to the specified address within the local machine.

| CANBUS communication receive data [ FreeCan_Recv_CB] | | | |
|---|---|---|---|
| Execution condition | Normally open/close triggering | Suitable model | XSLH-30A32,XSF |
| Firmware | XSLH-30A32: None<br>XSF: V2.3.0 and above | Software | - |
| Library | XJ_CANbus | | |

| Command | Name | Graph | ST |
|---|---|---|---|
| FreeCan_Recv_CB | CAN free format receive |  | FreeCan_Recv_CB_0(<br>Enable:= ,<br>CanID:= ,<br>RecvBuff:= ,<br>Port:= ,<br>Protocl:= ,<br>RecvDataLen=> ,<br>Done=> ,<br>Busy=> ,<br>Error=> ,<br>ErrorID=> ); |

(2) Related variables

| VAR_INPUT | Name | Data type | Valid range | Initial value | Description |
|---|---|---|---|---|---|
| Enable | Command normally open/close coil conduction | BOOL | FALSE/TRUE | FALSE | Normally open/close coil conduction |
| CanID | Specify remote communication ID number | UDINT | As the data type | - | Remote communication ID number |
| RecvBuff | Specify the first address number for receiving data locally | POINTER TO WORD | As the data type | - | The first address number for receiving data locally |
| Port | Specify the CAN port | BYTE | As the data type | - | Communication CAN port |
| Protocl | Specify CANBUS protocol | BYTE | As the data type | - | CANBUS communication protocol |
| VAR_OUTPUT | Name | Data type | Valid range | Initial value | Description |
| RecvDataLen | The actual length of | BYTE | As the data type | - | Data reception |

| VAR_INPUT | Name | Data type | Valid range | Initial value | Description |
|---|---|---|---|---|---|
| | received data | | | | termination signal |
| Done | Instruction processing completion flag | BOOL | FALSE/TRUE | FALSE | Instruction processing completion flag |
| Busy | Function block activation flag | BOOL | FALSE/TRUE | FALSE | Function block activation flag |
| Error | Error flag | BOOL | FALSE/TRUE | FALSE | Error flag |
| ErrorID | Error code | XJ_FreeCan_Error_Type | As the data type | - | Error code |

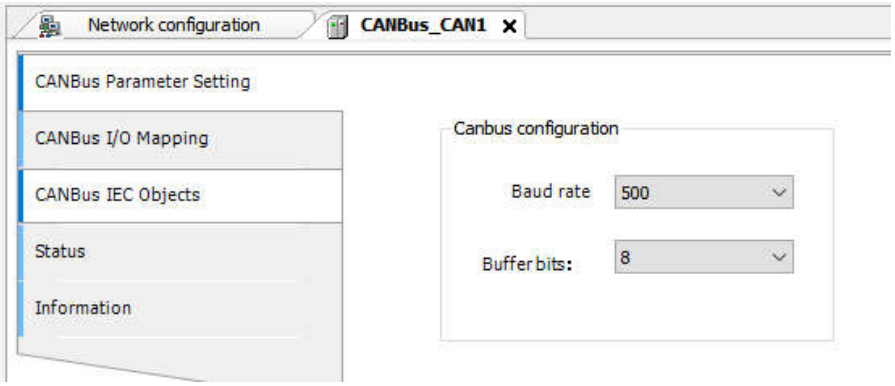
(3) Function

- Enable: Normally open/close coil waiting for reception.
- CanID: CAN2.0A: hexadecimal range H000- H7FF; CAN2.0B: Hexadecimal range H0000 0000- H1FFF FFFF. The ID number K0 serves as the identifier for enabling CAN free format broadcast reception under the CAN2.0A or CAN2.0B protocol. Other range ID numbers must comply with the standard receiving data of the sender's ID number and be limited according to the selected protocol type.
- RecvBuff: The first address of the received data byte.
- When the ID is 0, it is used as broadcast reception. CAN2.0A defaults to occupying 18 bytes after the specified receiving variable's first address. The first two bytes are used to store the broadcast reception ID number, and the last 16 bytes are used to receive data. CAN2.0B defaults to occupying the 20 bytes after the first address of the specified receiving variable. The first 4 bytes are used to store the ID number of broadcast reception, and the last 16 bytes are used to receive data.
- When receiving data with an ID other than 0, it defaults to occupying 16 bytes after the specified receiving address.
- Port: Communication CAN port range: K1, K2. K1: First CAN channel of PLC, K2: Second CAN channel of PLC (reserved)
- Protocol: CAN communication protocol K0: CAN2.0A protocol K1: CAN2.0B protocol.
- Done: Instruction processing completion flag, received data Done set to true.
- RecvDataLen: The actual length of data received.
- Busy: Function block activation.
- Error: True: An error occurred, the function block terminated the operation, FALSE: No error.
- ErrorID: Error code.

## 7-4-4. CAN free format communication example

**Establishing CAN free format communication between XSLH-30A32 and XL5N-32T**

The default parameters for the CAN port of XSLH-30A32 are a baud rate of 500Kbit/s and 8 cache bits. The parameters can be modified on the configuration interface as shown in the following figure:




1. XSLH-30A32 is used to write FreeCan_Send_CB and FreeCan_Recv_CB instructions, configure the remote communication ID number, communication address, communication byte count, communication CAN port, and protocol for the two instructions. The sample program is shown in the following figure:

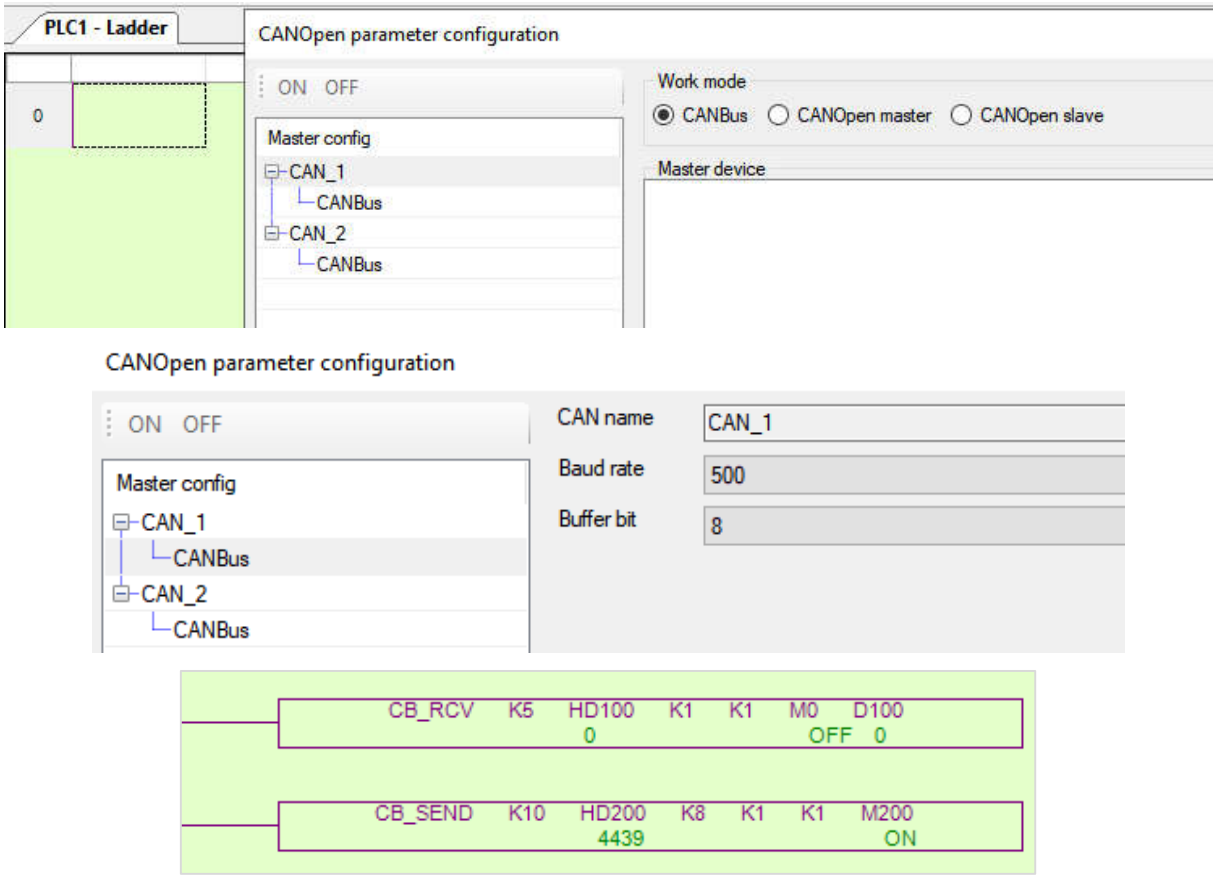
PLC_PRG

```
PROGRAM PLC_PRG
VAR
    send: FreeCan_Send_CB;
    send_start:BOOL;
    send_buffer:ARRAY[0..7] OF WORD;
    send_buffer_point:POINTER TO WORD :=ADR(send_buffer);
    TON1:TON;
    TON2:TON;
    Time_start:BOOL;
    Rcv:FreeCan_Recv_CB;
    RCV_start:BOOL;
recv_buffer:ARRAY[0..7] OF WORD;
recv_buffer_point:POINTER TO  WORD :=ADR(recv_buffer);

END_VAR
```

```
TON1(IN:= Time_start AND NOT TON2.Q, PT:=T#300MS, Q=>send_start , ET=> );
TON2(IN:= send_start  , PT:=T#300MS , Q=> , ET=> );
send(
    Execute:= send_start,                //触发写指令
    CanID:= 5,
    SendBuff:=send_buffer_point ,
    SendDataLen:=8,
    Port:= 1,
    Protocl:= 1,
    Done=> ,
    Busy=> ,
    Error=> ,
    ErrorID=> );
Rcv(
        Enable:=RCV_start ,              //读指令条件导通
        CanID:=10 ,
        RecvBuff:= recv_buffer_point,
        Port:=1 ,
        Protocl:= 1,
        RecvDataLen=> ,
        Done=> ,
        Busy=> ,
        Error=> ,
        ErrorID=> );
```

2. XL5N-32T is used to configure the CAN port protocol and baud rate, and then write the CB_SEND and CB_RCV commands. The configuration corresponding to the XSLH-30A32 command is written with the corresponding remote communication ID number, communication address, communication byte count, communication CAN port and protocol. The sample program is shown in the following figure:




3. After configuring the instructions and parameters, download and trigger the instructions to receive and receive data.

# 8. File processing

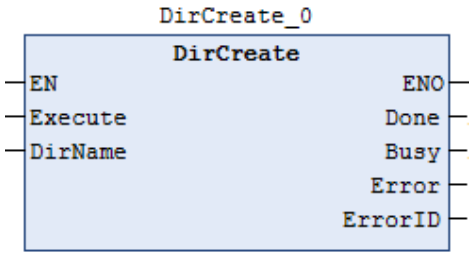
Due to the different file processing methods between Windows and Linux systems, there may be some differences in using the same file processing instruction on different operating systems.

## 8-1. Create directory DirCreate

(1) Instruction overview

Create a directory with the specified name in the SD card.

| Create directory [DirCreate] | | | |
|---|---|---|---|
| Execution condition | Edge triggering | Suitable model | XS, XSA |
| Firmware | - | Software | XS StudioV2.2.0 and above |
| Library | XJ_FunctionUtils | | |

| Instruction | Name | Graph | ST |
|---|---|---|---|
| DirCreate | Create directory |  | DirCreate_0(<br>Execute:= ,<br>DirName:= ,<br>Done=> ,<br>Busy=> ,<br>Error=> ,<br>ErrorID=> ); |

(2) Related variables

| VAR_INPUT | Name | Data type | Effective range | Initial value | Description |
|---|---|---|---|---|---|
| Execute | Command enable | BOOL | FALSE/TRUE | FALSE | Input high-level rising edge to start command |
| DirName | Specify directory name | STRING | Maximum 1985 characters | - | The directory name to be created |
| VAR_OUTPUT | Name | Data type | Effective range | Initial value | Description |
| Done | Completed flag | BOOL | FALSE/TRUE | FALSE | Instruction completion signal |
| Busy | Executing flag | BOOL | FALSE/TRUE | FALSE | Instruction execution status |
| Error | Error flag | BOOL | FALSE/TRUE | FALSE | Instruction error status |
| ErrorID | Error code | WORD | As the data type | - | Instruction error ID |

(3) Function

- Generate a directory with the specified name in "DirName" in the SD card.

(4) Note

- Even if the value of "Execute" becomes FALSE or the instruction execution time exceeds the task cycle, this instruction will continue to be executed until the end. Whether the processing has ended normally can be

confirmed by whether the value of "Done" has changed to true.

- When the SD storage card is unplugged while the file is open, the file will remain open. But when reinstalling the SD memory card, it will not be possible to read or write files. To read and write the file, please reopen the file.
- The directory name size specified in "DirName", with a maximum support of 255 characters for Linux systems with prefixes added, will be truncated if exceeded; The Windows system supports a maximum of 260 characters with prefix added, and an error will be reported if it exceeds this limit.
- An exception will occur in the following situations. "Error" is true.
  - When the SD card is not in a usable state.
  - The SD card is write protected.
  - When the remaining space of the SD card is insufficient.
  - When the number of directories that can be created exceeds.
  - When the directory specified in "DirName" already exists.
  - When the value of "DirName" is an illegal directory name.
  - When accessing the SD card and unable to access due to an exception.
- Notes on setting the directory name specified in "DirName":

  The following is a simple example. For more details, please refer to the path displayed in the PLC directory on the right side of the "File" interface when inserting SD card in XS models or inserting USB drive in XSA models:

  - XSF: There are two setting formats, including ① $SDDev$/part0/+custom file name(e.g $SDDev$/part0/AA) ② ./SDDev/part0/+Custom file name(e.g ../SDDev/part0/AA).

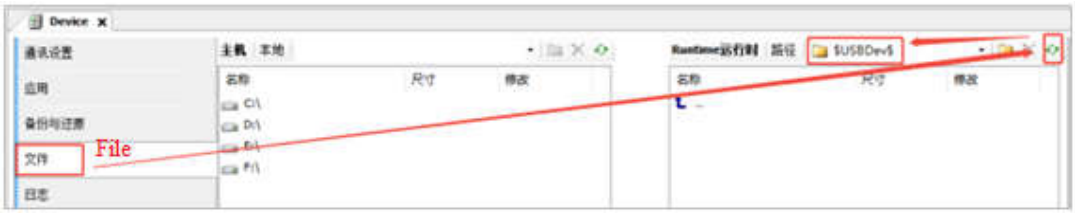
```
PROGRAM PLC_PRG
VAR
    A:XJ_FunctionUtils.DirCreate;
END_VAR
```

```
A(Execute:= ,
    DirName:= '$$SDDev$$/part0/AA',
    Done=> ,
    Busy=> ,
    Error=> ,
    ErrorID=> );
```

```
PROGRAM PLC_PRG
VAR
    A:XJ_FunctionUtils.DirCreate;
END_VAR
```

```
A(Execute:= ,
    DirName:= '../SDDev/part0/AA',
    Done=> ,
    Busy=> ,
    Error=> ,
    ErrorID=> );
```

- XSA-W: Set to ../$$USBDev$$/+Custom file name (e.g ../$$USBDev$$/AA).

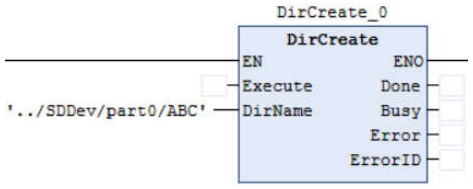
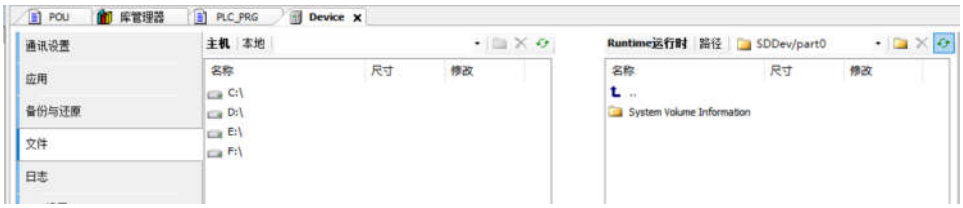


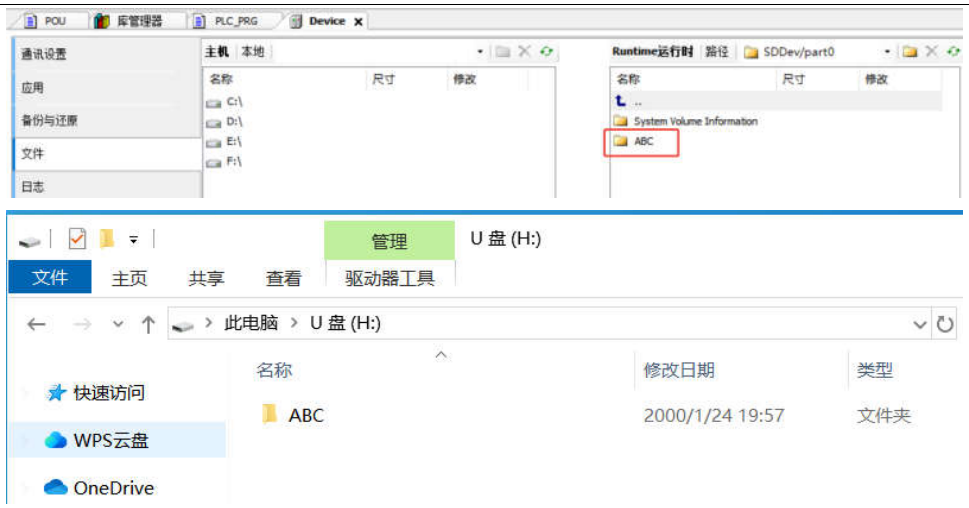

```
PROGRAM PLC_PRG
VAR
    A:XJ_FunctionUtils.DirCreate;
END_VAR
```

```
A(Execute:= ,
    DirName:= '../$$USBDev$$/AA',
    Done=> ,
    Busy=> ,
    Error=> ,
    ErrorID=> );
```

(5) Sample program

Taking the operation of inserting an SD card in the XSF5 as an example:

| DEMO | LD | ST |
|---|---|---|
| Define the variables | VAR<br>DirCreate_0:DirCreate;<br>END_VAR | |
| Sample program |  | DirCreate_0(Execute:= ,<br>DirName:= '../SDDev/part0/ABC',<br>Done=> ,<br>Busy=> ,<br>Error=> ,<br>ErrorID=> ); |
| Operation result | <br>Before instruction execution:<br>  <br>After instruction execution: | |

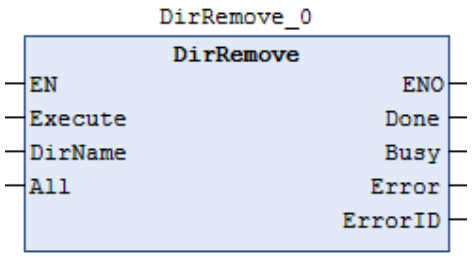

POU
库管理器
PLC_PRG
Device
通讯设置
应用
备份与还原
文件
日志
主机
本地
名称
尺寸
修改
C:\
D:\
E:\
F:\
Runtime运行时
路径
SDDev/part0
System Volume Information
ABC
管理
U 盘 (H:)
文件
主页
共享
查看
驱动器工具
此电脑
U 盘 (H:)
快速访问
WPS云盘
OneDrive
修改日期
类型
2000/1/24 19:57
文件夹

## 8-2. Delete directory DirRemove

(1) Instuction overview

Delete the specified directory in the SD card.

| Delete directory [DirRemove] | | | |
|---|---|---|---|
| Execution condition | Edge triggering | Suitable model | XS, XSA |
| Firmware | - | Software | XS StudioV2.2.0 and above |
| Library | XJ_FunctionUtils | | |

| Instruction | Name | Graph | ST |
|---|---|---|---|
| DirRemove | Delete directory |  | DirRemove_0(<br>Execute:= ,<br>DirName:= ,<br>All:= ,<br>Done=> ,<br>Busy=> ,<br>Error=> ,<br>ErrorID=> ); |

(2) Related variables

| VAR_INPUT | Name | Data type | Effective range | Initial value | Description |
|---|---|---|---|---|---|
| Execute | Instruction enable | BOOL | FALSE/TRUE | FALSE | Input high-level rising edge to start command |
| DirName | Delete directory name | STRING | Maximum 1985 characters | - | Delete directory name |
| All | Select all | BOOL | FALSE/TRUE | FALSE | Operation when there are files/subdirectories in the directory<br>True: Delete<br>FALSE: Do not delete |
| VAR_OUTPUT | Name | Data type | Effective range | Initial value | Description |
| Done | Completed flag | BOOL | FALSE/TRUE | FALSE | Instruction completion signal |
| Busy | Executing flag | BOOL | FALSE/TRUE | FALSE | Instruction execution status |
| Error | Error flag | BOOL | FALSE/TRUE | FALSE | Instruction error status |
| ErrorID | Error code | WORD | As the data type | - | Instruction error ID |

(3) Function

- Delete the specified directory in the SD storage card specified in the directory name "DirName".
- If there are files or subdirectories in the directory specified by "DirName", depending on the value of "All" selected, the process is as follows.

| “All” value | Operation |
|---|---|
| TRUE | Delete specified directory, including files and entire subdirectories |

| | |
|---|---|
| FALSE | Not deleting specified directory, exception occurred |

(4) Note

- Even if the value of "Execute" becomes FALSE or the instruction execution time exceeds the task cycle, this instruction will continue to be executed until the end. Whether the processing has ended normally can be confirmed by whether the value of "Done" has changed to true.
- When the SD storage card is unplugged while the file is open, the file will remain open. But when reinstalling the SD memory card, it will not be possible to read or write files. To read and write the file, please reopen the file.
- An exception will occur in the following situations. "Error" is true.
  - When the SD memory card is not in a usable state.
  - The SD storage card is write protected.
  - When the value of "All" is true and the directory specified in "DirName" is being accessed through other instructions.
  - When the value of "All" is FALSE and there are files or directories in the directory specified in "DirName".
  - When the directory specified in "DirName" is read-only.
  - When the file specified in "DirName" does not exist.
  - When accessing the SD memory card and unable to access due to an exception.

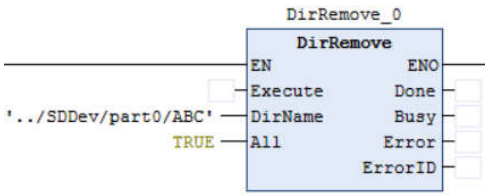
(5) Sample program

Taking the operation of inserting an SD card in the XSF5 as an example:

<table>
<tr><th>DEMO</th><th>LD</th><th>ST</th></tr>
<tr><td>Define the variables</td><td colspan="2">

```
VAR
    DirRemove_0:DirRemove;
END_VAR
```

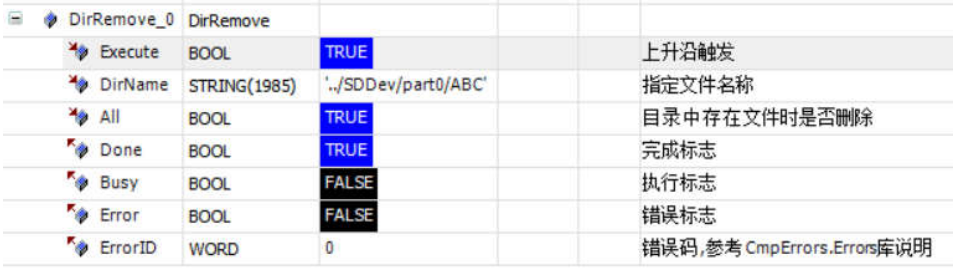
</td></tr>
<tr><td>Sample program</td><td>




</td><td>

```
DirRemove_0(
    Execute:= ,
    DirName:= '../SDDev/part0/ABC',
    All:= TRUE,
    Done=> ,
    Busy=> ,
    Error=> ,
    ErrorID=> );
```

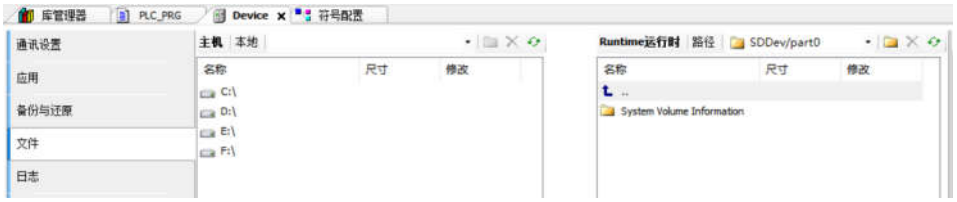
</td></tr>
<tr><td>Operation result</td><td colspan="2">



| | | | |
|---|---|---|---|
| DirRemove_0 | DirRemove | | |
| Execute | BOOL | TRUE | 上升沿触发 |
| DirName | STRING(1985) | '../SDDev/part0/ABC' | 指定文件名称 |
| All | BOOL | TRUE | 目录中存在文件时是否删除 |
| Done | BOOL | TRUE | 完成标志 |
| Busy | BOOL | FALSE | 执行标志 |
| Error | BOOL | FALSE | 错误标志 |
| ErrorID | WORD | 0 | 错误码,参考 CmpErrors.Errors库说明 |

Before execution:

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

| | |
|---|---|
| | After execution:<br>  <br>The specified directory has been deleted. |

## 8-3. Open file FileOpen

(1) Instruction overview

Open the specified file in the SD storage card.

| Open file [FileOpen] | | | |
|---|---|---|---|
| Execution condition | Edge triggering | Suitable model | XS, XSA |
| Firmware | - | Software | XS StudioV2.2.0 and above |
| Library | XJ_FunctionUtils | | |

| Instruction | Name | Graph | ST |
|---|---|---|---|
| FileOpen | Open file | FileOpen_0<br>FileOpen<br>EN<br>Execute<br>FileName<br>Mode<br>ENO<br>Done<br>Busy<br>Error<br>ErrorID<br>FileID | FileOpen_0(<br>Execute:= ,<br>FileName:= ,<br>Mode:= ,<br>Done=> ,<br>Busy=> ,<br>Error=> ,<br>ErrorID=> ,<br>FileID=> ); |

(2) Related variables

| VAR_INPUT | Name | Data type | Effective range | Initial value | Description |
|---|---|---|---|---|---|
| Execute | Instruction enable | BOOL | FALSE/TRUE | FALSE | Input high-level rising edge to start command |
| FileName | Specify a filename | STRING | Maximum 1985 characters | - | The file name to be opened |
| Mode | Open mode | ACCESS_MODE | As the data type | - | Open file mode |
| VAR_OUTPUT | Name | Data type | Effective range | Initial value | Description |
| Done | Completed flag | BOOL | FALSE/TRUE | FALSE | Instruction completion signal |
| Busy | Executing flag | BOOL | FALSE/TRUE | FALSE | Instruction execution status |
| Error | Error flag | BOOL | FALSE/TRUE | FALSE | Instruction error status |
| ErrorID | Error code | WORD | As the data type | - | Instruction error ID |

| | | | | | |
|---|---|---|---|---|---|
| FileID | File ID | DWORD | As the data type | - | The file ID of the opened file |

(3) Function

- Open the file in the SD storage card specified in "FileName" using the mode specified in "Mode".
- As the result of opening the file, the file ID "FileID" will be output. "FileID" can be used in the FileRead and FileWrite instructions to specify a file.
- The data type of "Mode" is enumeration type ACCESS-MODE. The meaning of enumeration values is as follows:

| Enum | Meaning |
|---|---|
| AM_READ | If the target path file exists, open the existing file as read-only. If it does not exist, return an error value |
| AM_WRITE | If the target path file exists, open the file by rewriting it. If it does not exist, a new file with write permission will be automatically created |
| AM_APPEND | If the target path file exists, open the existing file in a write only manner and append the content to the end of the file. If it does not exist, return an error value |
| AM_READ_PLUS | If the target path file exists, open the existing file for reading and writing. If it does not exist, return an error value |
| AM_WRITE_PLUS | If the target path file exists, open the file as read-write. If it does not exist, a new file with read/write permissions will be automatically created |
| AM_APPEND_PLUS | If the target path file exists, open the existing file as read-write and append the content to the end of the file. If it does not exist, a new file with read/write permissions will be automatically created |

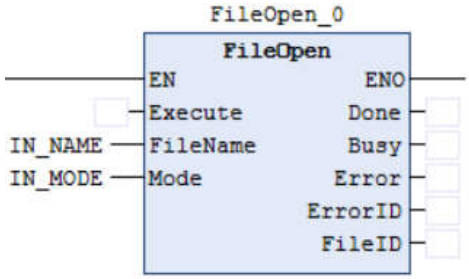
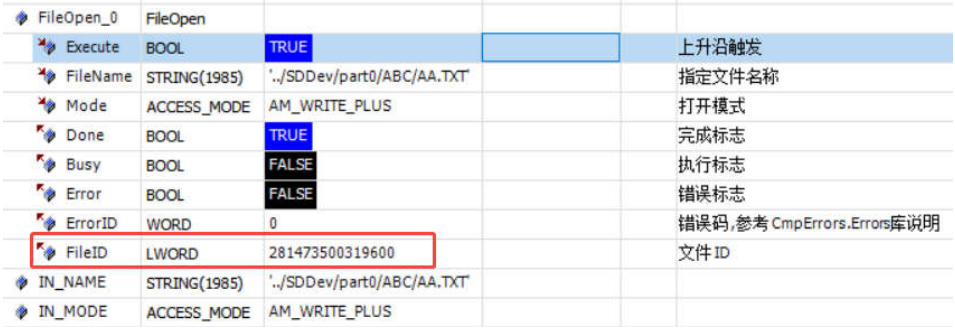
(4) Note

- Even if the value of "Execute" becomes FALSE or the instruction execution time exceeds the task cycle, this instruction will continue to be executed until the end. Whether the processing has ended normally can be confirmed by whether the value of "Done" has changed to true.
- This instruction needs to be executed before the FileSeek instruction, FileWrite instruction, FileRead instruction, FilePuts instruction, and FileGets instruction.
- For files opened with this command, please execute the FileClose command to close the file.
- When this instruction is completed, save the value in "FileID". When the value of "Done" changes from FALSE to True.
- When the SD storage card is unplugged while the file is open, the file will remain open. But when reinstalling the SD memory card, it will not be possible to read or write files. To read and write the file,
- A file opened in the form of AM_WRITE, when using SysFileRead in the Linus system to obtain character length, will display an error message with an error ID of 1; In the Windows system, using SysFileRead to obtain character length will display an error message with an error ID of 2.
- A file opened in AM_APPEND format, when using SysFileRead in the Linus system to obtain character length, will display an error message with an error ID of 1; Using SysFileRead to obtain character length in Windows system can execute normally.
- Files opened through SysFileOpen can be deleted under the RUNTIME path in the Linus system, but cannot be deleted under the RUNTIME path in the Windows system due to permission issues.
- An exception will occur in the following situations. "Error" is true.
  - When the SD memory card is not in a usable state.
  - The SD storage card is write protected.

- When the value of "Mode" is AM_READ, AM_APPEND or AM_READ_PLUS, and the file specified in "FileName" does not exist.
- When the value of "FileName" is an illegal file name.
- When the number of files and directories that can be created exceeds.
- When accessing the file specified in "FileName".
- When the file specified in "FileName" is read-only.
- When accessing the SD memory card and unable to access due to an exception.

(5) Sample program

Taking the operation of inserting an SD card in the XSF5 as an example:

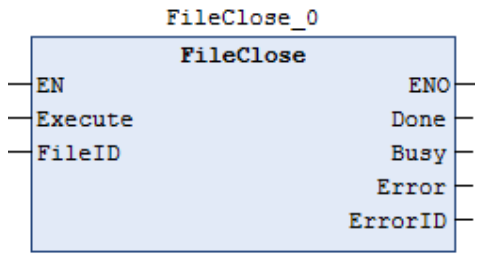
<table>
<tr><th>DEMO</th><th>LD</th><th>ST</th></tr>
<tr><td>Save the AA.txt in the SD card</td><td colspan="2">
</td></tr>
<tr><td>Define the variables</td><td colspan="2">VAR<br>FileOpen_0:FileOpen;<br>IN_NAME:STRING(1985):='../SDDev/part0/ABC/AA.TXT';<br>IN_MODE:ACCESS_MODE;<br>END_VAR</td></tr>
<tr><td>Sample program</td><td> 
</td><td>FileOpen_0(<br>Execute:= ,<br>FileName:= IN_NAME,<br>Mode:= IN_MODE,<br>Done=> ,<br>Busy=> ,<br>Error=> ,<br>ErrorID=> ,<br>FileID=> );</td></tr>
<tr><td>Operation result</td><td colspan="2"> 
</td></tr>
</table>

## 8-4. Close the file FileClose

(1) Instruction overview

Close the specified file in the SD storage card.

| Close the file [FileClose] | | | |
|---|---|---|---|
| Execution condition | Edge triggering | Suitable model | XS, XSA |
| Firmware | - | Software | XS StudioV2.2.0 and above |
| Library | XJ_FunctionUtils | | |

| Instruction | Name | Graph | ST |
|---|---|---|---|
| FileClose | Close the file |  | FileClose_0(<br>Execute:= ,<br>FileID:= ,<br>Done=> ,<br>Busy=> ,<br>Error=> ,<br>ErrorID=> ); |

(2) Related variables

| VAR_INPUT | Name | Data type | Effective range | Initial value | Description |
|---|---|---|---|---|---|
| Execute | Instruction enable | BOOL | FALSE/TRUE | FALSE | Input high-level rising edge to start command |
| FileID | File ID | _XWORD(DWORD) | As the data type | - | The file ID of the file to be closed |
| VAR_OUTPUT | Name | Data type | Effective range | Initial value | Description |
| Done | Completed flag | BOOL | FALSE/TRUE | FALSE | Instruction completion signal |
| Busy | Executing flag | BOOL | FALSE/TRUE | FALSE | Instruction execution status |
| Error | Error flag | BOOL | FALSE/TRUE | FALSE | Instruction error status |
| ErrorID | Error code | WORD | As the data type | - | Instruction error ID |

(3) Function

Close the files in the SD storage card specified in "FileID".

(4) Notes

- It is prohibited to execute this command multiple times for the same file, otherwise it may cause abnormal operation of the PLC and even crash.
- Even if the value of "Execute" becomes FALSE or the instruction execution time exceeds the task cycle, this instruction will continue to be executed until the end. Whether the processing has ended normally can be confirmed by whether the value of "Done" has changed to true.
- "FileID" needs to be obtained by executing the FileOpen command in advance.
- For files opened with the FileOpen command, be sure to execute this command and close the file after use,

otherwise unexpected results may occur.

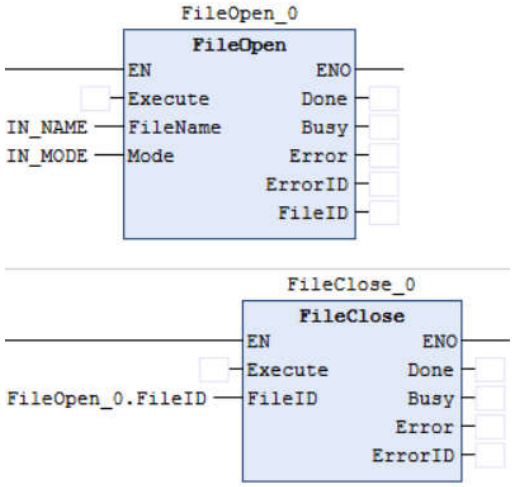
- When the SD storage card is unplugged while the file is open, the file will remain open. But when reinstalling the SD memory card, it will not be possible to read or write files. To read and write the file, please reopen the file.
- An exception will occur in the following situations. "Error" is true.
  - When the SD memory card is not in a usable state.
  - When accessing the file specified in "FileID".
  - When the file specified in "FileID" does not exist.
  - When accessing the SD memory card and unable to access due to an exception.

(5) Sample program

<table>
<tr><th>DEMO</th><th>LD</th><th>ST</th></tr>
<tr><td>Define the variables</td><td colspan="2">

```
VAR
    FileOpen_0:FileOpen;
    IN_NAME:STRING(1985):='../SDDev/part0/ABC/AA.TXT';
    IN_MODE:ACCESS_MODE;
    FileClose_0:FileClose;
END_VAR
```

</td></tr>
<tr><td>Sample program</td><td>




</td><td>

```
FileOpen_0(
    Execute:= ,
    FileName:= IN_NAME,
    Mode:= IN_MODE,
    Done=> ,
    Busy=> ,
    Error=> ,
    ErrorID=> ,
    FileID=> );
FileClose_0(
    Execute:= ,
    FileID:= FileOpen_0.FileID,
    Done=> ,
    Busy=> ,
    Error=> ,
    ErrorID=> );
```

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

| | | | |
|---|---|---|---|
| FileOpen_0 | FileOpen | | |
| Execute | BOOL | FALSE | 上升沿触发 |
| FileName | STRING(1985) | '../SDDev/part0/ABC/AA.TXT' | 指定文件名称 |
| Mode | ACCESS_MODE | AM_WRITE_PLUS | 打开模式 |
| Done | BOOL | FALSE | 完成标志 |
| Busy | BOOL | FALSE | 执行标志 |
| Error | BOOL | FALSE | 错误标志 |
| ErrorID | WORD | 0 | 错误码,参考 CmpErrors.Errors库说明 |
| FileID | LWORD | 281473500319600 | 文件 ID |
| IN_NAME | STRING(1985) | '../SDDev/part0/ABC/AA.TXT' | |
| IN_MODE | ACCESS_MODE | AM_WRITE_PLUS | |
| FileClose_0 | FileClose | | |
| Execute | BOOL | TRUE | 上升沿触发 |
| FileID | LWORD | 281473500319600 | 文件 ID |
| Done | BOOL | TRUE | 完成标志 |
| Busy | BOOL | FALSE | 执行标志 |
| Error | BOOL | FALSE | 错误标志 |
| ErrorID | WORD | 0 | 错误码,参考 CmpErrors.Errors库说明 |

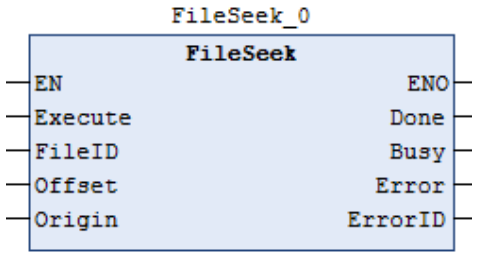
</td></tr>
</table>

## 8-5. Find the file FileSeek

(1) Instruction overview

Set the file location indicator in the specified file on the SD storage card.

| Find the file [FileSeek] | | | |
|---|---|---|---|
| Execution condition | Edge triggering | Suitable model | XS, XSA |
| Firmware | - | Software | XS StudioV2.2.0 and above |
| Library | XJ_FunctionUtils | | |

| Instruction | Name | Graph | ST |
|---|---|---|---|
| FileSeek | Find the file |  | FileSeek_0(<br>Execute:= ,<br>FileID:= ,<br>Offset:= ,<br>Origin:= ,<br>Done=> ,<br>Busy=> ,<br>Error=> ,<br>ErrorID=> ); |

(2) Related variables

| VAR_INPUT | Name | Data type | Effective range | Initial value | Description |
|---|---|---|---|---|---|
| Execute | Instruction enable | BOOL | FALSE/TRUE | FALSE | Input high-level rising edge to start command |
| FileID | File ID | _XWORD(DWORD) | As the data type | - | The file ID of the file where the file location indicator is to be set |
| Offset | Offset | DINT | As the data type | - | The offset position of the file location indicator relative to "Origin" |
| Origin | Refrence location | _eFSEEK_ORIGIN | As the data type | _eFSEEK_ORIGIN._SEEK_SET | The reference position of the file location indicator |
| VAR_OUTPUT | Name | Data type | Effective range | Initial value | Description |
| Done | Completed flag | BOOL | FALSE/TRUE | FALSE | Instruction completion signal |
| Busy | Executing flag | BOOL | FALSE/TRUE | FALSE | Instruction execution status |
| Error | Error flag | BOOL | FALSE/TRUE | FALSE | Instruction error status |
| ErrorID | Error code | WORD | As the data type | - | Instruction error ID |

(3) Function

- Set the file location indicator in the SD storage card file specified in the file ID "FileID". The file location indicator refers to the location in the file where reading or writing begins when executing file data read and

write instructions. For example, when reading from the beginning of a file, the FileSeek command should be used to set the file location indicator at the beginning of the file, and then the FileRead command should be executed.

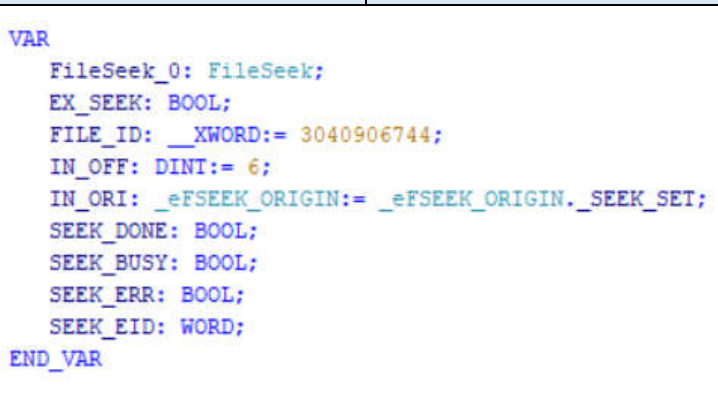
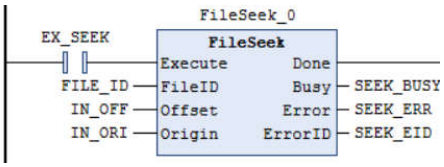
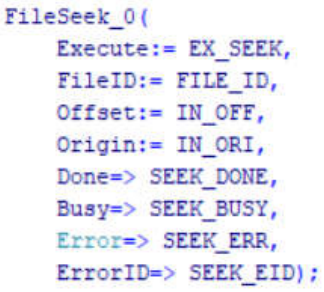
- Set the position of the file location indicator as the reference position "Origin" plus the offset "Offset".
- The data type of "Origin" is enumeration _eFSEEK_ORIGIN. The meaning of enumeration values is as follows:

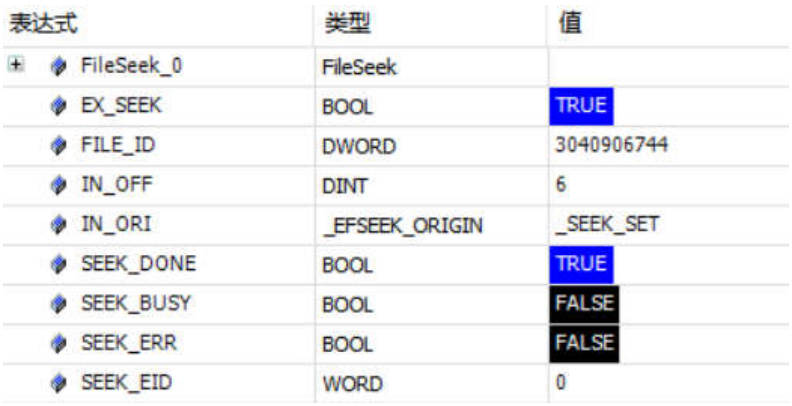
| Enum | Meaning |
|---|---|
| _SEEK_SET | At the beginning of the file |
| _SEEK_CUR | The location of the current file location indicator |
| _SEEK_END | At the end of the file |

(4) Notes

- Even if the value of "Execute" becomes FALSE or the instruction execution time exceeds the task cycle, this instruction will continue to be executed until the end. Whether the processing has ended normally can be confirmed by whether the value of "Done" has changed to true.
- "FileID" needs to be obtained by executing the FileOpen command in advance.
- An exception will occur in the following situations. "Error" is true.
  - When the SD memory card is not in a usable state.
  - When accessing the file specified in "FileID".
  - When the file specified in "FileID" does not exist.
  - When accessing the SD memory card and unable to access due to an exception.

(5) Sample program

| DEMO | LD | ST |
|---|---|---|
| Define the variables |  | |
| Sample program |  |  |

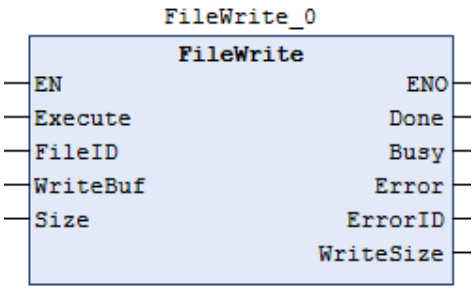
| | |
|---|---|
| Operation result |  |

## 8-6. Write in the file FileWrite

(1) Instruction overview

Write the data to the specified file on the SD storage card.

| Write in the file[FileWrite] | | | |
|---|---|---|---|
| Execution condition | Edge triggering | Suitable model | XS, XSA |
| Firmware | - | Software | XS StudioV2.2.0 and above |
| Library | XJ_FunctionUtils | | |

| Instruction | Name | Graph | ST |
|---|---|---|---|
| FileWrite | Write in the file |  | FileWrite_0(<br>Execute:= ,<br>FileID:= ,<br>WriteBuf:= ,<br>Size:= ,<br>Done=> ,<br>Busy=> ,<br>Error=> ,<br>ErrorID=> ,<br>WriteSize=> ); |

(2) Related variables

| VAR_INPUT | Name | Data type | Effective range | Initial value | Description |
|---|---|---|---|---|---|
| Execute | Instruction enable | BOOL | FALSE/TRUE | FALSE | Input high-level rising edge to start command |
| FileID | File ID | _XWORD(DWORD) | As the data type | - | The file ID to write data to |
| Size | Write in element numbers | UINT | As the data type | - | Number of elements to be written |
| WriteBuf | Write cache array | BOOL/ BYTE/WORD/DWORD/LWORD / USINT/UINT/UDINT/ULINT/ SINT/INT/DINT/LINT/ REAL/LREAL/ TIME/DATE/TOD/DT/STRING | As the data type | - | The write object of the written data |
| VAR_OUTPUT | Name | Data type | Effective range | Initial value | Description |
| Done | Completed flag | BOOL | FALSE/TRUE | FALSE | Instruction completion signal |
| Busy | Executing flag | BOOL | FALSE/TRUE | FALSE | Instruction execution status |
| Error | Error flag | BOOL | FALSE/TRUE | FALSE | Instruction error status |
| ErrorID | Error code | WORD | As the data type | - | Instruction error ID |

| | | | | | |
|---|---|---|---|---|---|
| WriteSize | Actual number of elements written | UINT | As the data type | - | Actual number of elements written |

(3) Function

- For the SD storage card file specified in the file ID "FileID", write the data written to the cache WriteBuf[ ] to the location indicated by the file location indicator.
- The file location indicator should be set to any location in advance through the FileSeek command.
- The amount of data written is (WriteBuf [ ] data type size) × "Size".
- The actual number of elements written is saved in "WriteSize". Usually, the value of "Size" is consistent with the value of "ReadSize".

(4) Note

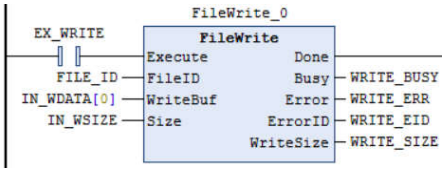
- Even if the value of "Execute" becomes FALSE or the instruction execution time exceeds the task cycle, this instruction will continue to be executed until the end. Whether the processing has ended normally can be confirmed by whether the value of "Done" has changed to true.
- When the value of "Size" exceeds the array area of WriteBuf[], it is necessary to avoid PLC exceptions.
- "FileID" needs to be obtained by executing the FileOpen command in advance.
- An exception will occur in the following situations. "Error" is true.
  - When the SD memory card is not in a usable state.
  - When accessing the file specified in "FileID".
  - When the file specified in "FileID" does not exist.
  - When accessing the SD memory card and unable to access due to an exception.
  - When the file specified in "FileID" is not opened in writable mode.
  - ReadBuf[] passes in the entire array, and the ErrorID displays error code "16#20".

(5) Sample program

| DEMO | LD | ST |
|---|---|---|
| Define the variables | | |

```
VAR
   FileWrite_0: FileWrite;
   EX_WRITE: BOOL;
   FILE_ID: __XWORD:= 3040906744;
   IN_WDATA: ARRAY[0..2] OF BYTE:= [16,32,64];
   IN_WSIZE: UINT:= 3;
   WRITE_DONE: BOOL;
   WRITE_BUSY: BOOL;
   WRITE_ERR: BOOL;
   WRITE_EID: WORD;
   WRITE_SIZE: UINT;
END_VAR
```

Sample program




```
FileWrite_0(
    Execute:= EX_WRITE,
    FileID:= FILE_ID,
    WriteBuf:= IN_WDATA[0],
    Size:= IN_WSIZE,
    Done=> WRITE_DONE,
    Busy=> WRITE_BUSY,
    Error=> WRITE_ERR,
    ErrorID=> WRITE_EID,
    WriteSize=> WRITE_SIZE);
```

Operation result

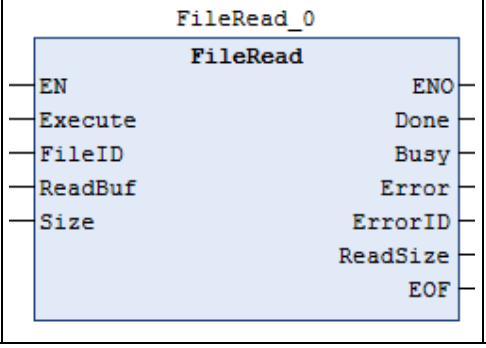
| 表达式 | 类型 | 值 |
|---|---|---|
| FileWrite_0 | FileWrite | |
| EX_WRITE | BOOL | TRUE |
| FILE_ID | DWORD | 3040906744 |
| IN_WDATA | ARRAY [0..2] OF BYTE | |
| IN_WDATA[0] | BYTE | 16 |
| IN_WDATA[1] | BYTE | 32 |
| IN_WDATA[2] | BYTE | 64 |
| IN_WSIZE | UINT | 3 |
| WRITE_DONE | BOOL | TRUE |
| WRITE_BUSY | BOOL | FALSE |
| WRITE_ERR | BOOL | FALSE |
| WRITE_EID | WORD | 0 |
| WRITE_SIZE | UINT | 3 |

## 8-7. Read the file FileRead

(1) Instruction overview

Read the data of the specified file from the SD storage card.

| Read the file [FileRead] | | | |
|---|---|---|---|
| Execution condition | Edge triggering | Suitable model | XS, XSA |
| Firmware | - | Software | XS StudioV2.2.0 and above |
| Library | XJ_FunctionUtils | | |

| Instruction | Name | Graph | ST |
|---|---|---|---|
| FileRead | Read the file |  | FileRead_0(<br>Execute:= ,<br>FileID:= ,<br>ReadBuf:= ,<br>Size:= ,<br>Done=> ,<br>Busy=> ,<br>Error=> ,<br>ErrorID=> ,<br>ReadSize=> ,<br>EOF=> ); |

(2) Related variables

| VAR_INPUT | Name | Data type | Effective range | Initial value | Description |
|---|---|---|---|---|---|
| Execute | Instruction enable | BOOL | FALSE/TRUE | FALSE | Input high-level rising edge to start command |
| FileID | File ID | _XWORD(DWORD) | As the data type | - | The file ID to read data from |
| Size | Read number of elements | UINT | As the data type | - | Number of elements to be read |
| ReadBuf | Read cache array | BOOL/ BYTE/WORD/DWORD/LWORD/ USINT/UINT/UDINT/ULINT/ SINT/INT/DINT/LINT/ REAL/LREAL/ TIME/DATE/TOD/DT/STRING | As the data type | - | The write object of the read data |
| VAR_OUTPUT | Name | Data type | Effective range | Initial value | Description |
| Done | Completed flag | BOOL | FALSE/TRUE | FALSE | Instruction completion signal |
| Busy | Executing flag | BOOL | FALSE/TRUE | FALSE | Instruction execution status |
| Error | Error flag | BOOL | FALSE/TRUE | FALSE | Instruction error status |
| ErrorID | Error code | WORD | As the data type | - | Instruction error ID |

| | | | | | |
|---|---|---|---|---|---|
| ReadSize | Actual read number of elements | UINT | As the data type | - | Actual read number of elements |
| EOF | File end flag | BOOL | FALSE/TRUE | FALSE | Have you reached the end of the file<br>True: Arrived<br>FALE: Not arrived |

(3) Function

- For the SD storage card file specified in the file ID "FileID", start reading data from the location indicated by the file location indicator and save it to the read cache ReadBuf[].
- The file location indicator should be set to any location in advance through the FileSeek command.
- The amount of data read is (ReadBuf[] data type size) × "Size".
- The actual number of elements read is saved in "ReadSize". Usually, the value of "Size" is consistent with the value of "ReadSize". If the amount of data from the location of the file location indicator to the end of the file is less than "Size", no exception will occur, and the data up to the end of the file will be saved to ReadBuf[]. At this point, the value of "ReadSize" becomes smaller than the value of "Size".
- In addition, when reading to the end of the file, the value of "EOF" at the end of the file becomes true. If not, the value of "EOF" is FALSE.

(4) Note

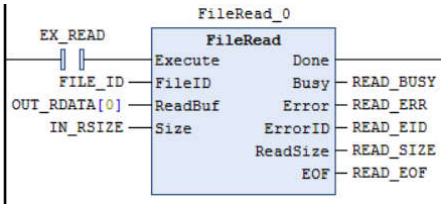
- Even if the value of "Execute" becomes FALSE or the instruction execution time exceeds the task cycle, this instruction will continue to be executed until the end. Whether the processing has ended normally can be confirmed by whether the value of "Done" has changed to true.
- When reading data up to the end of the file and the data size cannot be fully divided by the data type size of ReadBuf[], data with less than the data size of ReadBuf[] will also be processed. The file location indicator will come to the end of the file, and the value of "EOF" will become true.
- When the value of "Size" exceeds the array area of ReadBuf[], it is necessary to avoid PLC exceptions. The element after "Size" in ReadBuf[] (which will not be overwritten by reading) remains the value before executing this instruction.
- "FileID" needs to be obtained by executing the FileOpen command in advance.
- When this instruction is completed, save the value in "EOF". When the value of "Done" changes from FALSE to True.
- An exception will occur in the following situations. "Error" is true.
  - When the SD memory card is not in a usable state.
  - When accessing the file specified in "FileID".
  - When the file specified in "FileID" does not exist.
  - When accessing the SD memory card and unable to access due to an exception.
  - When the file specified in "FileID" is not opened in readable mode.
  - ReadBuf[] passes in the entire array, and the ErrorID displays error code "16#20".

(5) Sample program

<table>
<tr><th>DEMO</th><th>LD</th><th>ST</th></tr>
<tr><td>Define the variables</td><td colspan="2">

```
VAR
    FileRead_0: FileRead;
    EX_READ: BOOL;
    FILE_ID: __XWORD:= 3040906744;
    OUT_RDATA: ARRAY[0..2] OF BYTE;
    IN_RSIZE: UINT:= 3;
    READ_DONE: BOOL;
    READ_BUSY: BOOL;
    READ_ERR: BOOL;
    READ_EID: WORD;
    READ_SIZE: UINT;
    READ_EOF: BOOL;
END_VAR
```

</td></tr>
<tr><td>Sample program</td><td>




</td><td>

```
FileRead_0(
    Execute:= EX_READ,
    FileID:= FILE_ID,
    ReadBuf:= OUT_RDATA,
    Size:= IN_RSIZE,
    Done=> READ_DONE,
    Busy=> READ_BUSY,
    Error=> READ_ERR,
    ErrorID=> READ_EID,
    ReadSize=> READ_SIZE,
    EOF=> READ_EOF);
```

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

| 表达式 | 类型 | 值 |
|---|---|---|
| FileRead_0 | FileRead | |
| EX_READ | BOOL | TRUE |
| FILE_ID | DWORD | 3040906744 |
| OUT_RDATA | ARRAY [0..2] OF BYTE | |
| OUT_RDATA[0] | BYTE | 16 |
| OUT_RDATA[1] | BYTE | 32 |
| OUT_RDATA[2] | BYTE | 64 |
| IN_RSIZE | UINT | 3 |
| READ_DONE | BOOL | TRUE |
| READ_BUSY | BOOL | FALSE |
| READ_ERR | BOOL | FALSE |
| READ_EID | WORD | 0 |
| READ_SIZE | UINT | 3 |
| READ_EOF | BOOL | FALSE |

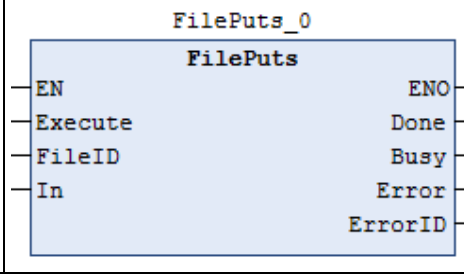
</td></tr>
</table>

## 8-8. Write string FilePuts

(1) Instruction overview

Write the string into the specified file on the SD storage card.

| Write string [FilePuts] | | | |
|---|---|---|---|
| Execution condition | Edge triggering | Suitable model | XS, XSA |
| Firmware | - | Software | XS StudioV2.2.0 and above |
| Library | XJ_FunctionUtils | | |

| Instruction | Name | Graph | ST |
|---|---|---|---|
| FilePuts | Write string |  | FilePuts_0(<br>Execute:= ,<br>FileID:= ,<br>In:= ,<br>Done=> ,<br>Busy=> ,<br>Error=> ,<br>ErrorID=> ); |

(2) Related variables

| VAR_INPUT | Name | Data type | Effective range | Initial value | Description |
|---|---|---|---|---|---|
| Execute | Instruction enable | BOOL | FALSE/TRUE | FALSE | Input high-level rising edge to start command |
| FileID | File ID | _XWORD(DWORD) | As the data type | - | The file ID to write data to |
| In | The string to be written | STRING | As the data type | - | The string to be written |
| VAR_OUTPUT | Name | Data type | Effective range | Initial value | Description |
| Done | Completed flag | BOOL | FALSE/TRUE | FALSE | Instruction completion signal |
| Busy | Executing flag | BOOL | FALSE/TRUE | FALSE | Instruction execution status |
| Error | Error flag | BOOL | FALSE/TRUE | FALSE | Instruction error status |
| ErrorID | Error code | WORD | As the data type | - | Instruction error ID |

(3) Function

- Write the string "In" to the location indicated by the file location indicator for the SD storage card specified in the file ID "FileID".
- The file location indicator should be set to any location in advance through the FileSeek command.

(4) Note

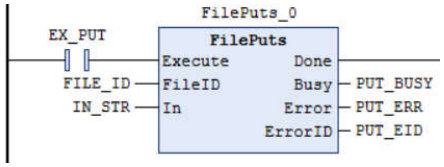
- Even if the value of "Execute" becomes FALSE or the instruction execution time exceeds the task cycle, this instruction will continue to be executed until the end. Whether the processing has ended normally can be confirmed by whether the value of "Done" has changed to true.
- "FileID" needs to be obtained by executing the FileOpen command in advance.
- An exception will occur in the following situations. "Error" is true.

- When the SD memory card is not in a usable state.
- The SD storage card is write protected.
- When the remaining space of the SD storage card is insufficient.
- When accessing the file specified in "FileID".
- When the file specified in "FileID" does not exist.
- When accessing the SD memory card and unable to access due to an exception.
- When the file specified in "FileID" is not opened in writable mode.

(5) Sample program

<table>
<tr><th>DEMO</th><th>LD</th><th>ST</th></tr>
<tr><td>Define the variables</td><td colspan="2">

```
VAR
    FilePuts_0: FilePuts;
    EX_PUT: BOOL;
    FILE_ID: __XWORD:= 3040906744;
    IN_STR: STRING(1985):= '603416';
    PUT_DONE: BOOL;
    PUT_BUSY: BOOL;
    PUT_ERR: BOOL;
    PUT_EID: WORD;
END_VAR
```

</td></tr>
<tr><td>Sample program</td><td> 
</td><td>

```
FilePuts_0(
    Execute:= EX_PUT,
    FileID:= FILE_ID,
    In:= IN_STR,
    Done=> PUT_DONE,
    Busy=> PUT_BUSY,
    Error=> PUT_ERR,
    ErrorID=> PUT_EID);
```

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

| 表达式 | 类型 | 值 |
|---|---|---|
| FilePuts_0 | FilePuts | |
| EX_PUT | BOOL | TRUE |
| FILE_ID | DWORD | 3040906744 |
| IN_STR | STRING(1985) | '603416' |
| PUT_DONE | BOOL | TRUE |
| PUT_BUSY | BOOL | FALSE |
| PUT_ERR | BOOL | FALSE |
| PUT_EID | WORD | 0 |

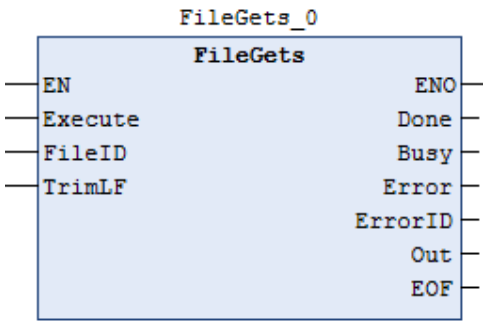
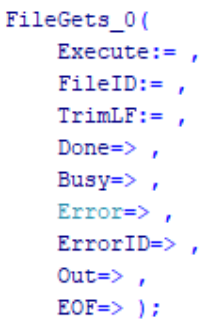
</td></tr>
</table>

## 8-9. Read string FileGets

(1) Instruction overview

Read 1 line of string from the specified file on the SD storage card.

| Read string [FileGets] | | | |
|---|---|---|---|
| Execution condition | Edge triggering | Suitable model | XS, XSA |
| Firmware | - | Software | XS StudioV2.2.0 and above |
| Library | XJ_FunctionUtils | | |

| Instruction | Name | Graph | ST |
|---|---|---|---|
| FileGets | Read string |  |  |



```
FileGets_0(
    Execute:= ,
    FileID:= ,
    TrimLF:= ,
    Done=> ,
    Busy=> ,
    Error=> ,
    ErrorID=> ,
    Out=> ,
    EOF=> );
```

(2) Related variables

| VAR_INPUT | Name | Data type | Effective range | Initial value | Description |
|---|---|---|---|---|---|
| Execute | Instruction enable | BOOL | FALSE/TRUE | FALSE | Input high-level rising edge to start command |
| FileID | File ID | _XWORD(DWORD) | As the data type | - | The file ID to read data from |
| TrimLF | Specify deletion Line break code | BOOL | FALSE/TRUE | FALSE | Specify deletion line break code TRUE: delete FALSE: not delete |
| VAR_OUTPUT | Name | Data type | Effective range | Initial value | Description |
| Done | Completed flag | BOOL | FALSE/TRUE | FALSE | Instruction completion signal |
| Busy | Executing flag | BOOL | FALSE/TRUE | FALSE | Instruction execution status |
| Error | Error flag | BOOL | FALSE/TRUE | FALSE | Instruction error status |
| ErrorID | Error code | WORD | As the data type | - | Instruction error ID |
| Out | Read string | STRING | As the data type | - | Read string |

| | | | | | |
|---|---|---|---|---|---|
| EOF | File end flag | BOOL | FALSE/TRUE | FALSE | Reached the end of the file<br>True: Arrived<br>FALE: Not arrived |

(3) Function

- Read one line of string from the location of the file location indicator for the SD storage card file specified in the file ID "FileID".
- The file location indicator should be set to any location in advance through the FileSeek command.
- The separation between rows is recognized based on the newline code. Output the read string to the read string "Out".
- Line breaks are automatically recognized from three types of CR, LF, and CR+LF. When the value of TrimLF, which specifies the deletion of the line break code, is set to true, it will be written to "Out" after deleting the line break code in the string.
- When reading to the end of the file, the value of the "EOF" at the end of the file flag becomes true.

(4) Note

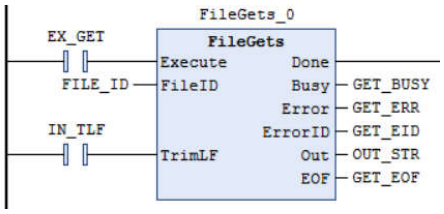
- Even if the value of "Execute" becomes FALSE or the instruction execution time exceeds the task cycle, this instruction will continue to be executed until the end. Whether the processing has ended normally can be confirmed by whether the value of "Done" has changed to true.
- When the length of a string exceeds 1985 bytes, the 1985 byte and its preceding string, as well as the last NULL character, will be saved to "Out".
- "FileID" needs to be obtained by executing the FileOpen command in advance.
- An exception will occur in the following situations. "Error" is true.
  - When the SD memory card is not in a usable state.
  - When accessing the file specified in "FileID".
  - When the file specified in "FileID" does not exist.
  - When accessing the SD memory card and unable to access due to an exception.
  - When the file specified in "FileID" is not opened in readable mode.

(5) Sample program

| DEMO | LD | ST |
|---|---|---|
| Define the variables | | |

```
VAR
   FileGets_0: FileGets;
   EX_GET: BOOL;
   FILE_ID: __XWORD:= 3040906744;
   IN_TLF: BOOL;
   GET_DONE: BOOL;
   GET_BUSY: BOOL;
   GET_ERR: BOOL;
   GET_EID: WORD;
   OUT_STR: STRING(1985);
   GET_EOF: BOOL;
END_VAR
```

Sample program




```
FileGets_0(
    Execute:= EX_GET,
    FileID:= FILE_ID,
    TrimLF:= IN_TLF,
    Done=> GET_DONE,
    Busy=> GET_BUSY,
    Error=> GET_ERR,
    ErrorID=> GET_EID,
    Out=> OUT_STR,
    EOF=> GET_EOF);
```

Operation result

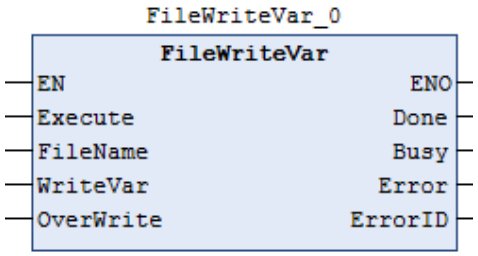
| 表达式 | 类型 | 值 |
|---|---|---|
| FileGets_0 | FileGets | |
| EX_GET | BOOL | TRUE |
| FILE_ID | DWORD | 3040906744 |
| IN_TLF | BOOL | FALSE |
| GET_DONE | BOOL | TRUE |
| GET_BUSY | BOOL | FALSE |
| GET_ERR | BOOL | FALSE |
| GET_EID | WORD | 0 |
| OUT_STR | STRING(1985) | '416603416603416' |
| GET_EOF | BOOL | TRUE |

## 8-10. Write variable file FileWriteVar

(1) Instruction overview

Write the value of one variable in binary form into the specified file on the SD storage card.

| Write variable file [FileWriteVar] | | | |
|---|---|---|---|
| Execution condition | Edge triggering | Suitable model | XS, XSA |
| Firmware | - | Software | XS StudioV2.2.0 and above |
| Library | XJ_FunctionUtils | | |

| Instruction | Name | Graph | ST |
|---|---|---|---|
| FileWriteVar | Write variable file |  | FileWriteVar_0(<br>Execute:= ,<br>FileName:= ,<br>WriteVar:= ,<br>OverWrite:= ,<br>Done=> ,<br>Busy=> ,<br>Error=> ,<br>ErrorID=> ); |

(2) Related variables

| VAR_INPUT | Name | Data type | Effective range | Initial value | Description |
|---|---|---|---|---|---|
| Execute | Instruction enable | BOOL | FALSE/TRUE | FALSE | Input high-level rising edge to start command |
| FileName | Specified file name | STRING | Maximum 1985 characters | - | The file name of the file to write variable data to |
| WriteVar | Specify the variables to be written | BOOL/ BYTE/WORD/DWORD/LWORD / USINT/UINT/UDINT/ULINT/ SINT/INT/DINT/LINT/ REAL/LREAL/ TIME/DATE/TOD/DT/STRING | As the data type | - | Variable to be written |
| OverWrite | Overlay selection | BOOL | FALSE/TRUE | FALSE | True: Allow overwrite FALSE: Do not overwrite |
| VAR_OUTPUT | Name | Data type | Effective range | Initial value | Description |
| Done | Completed flag | BOOL | FALSE/TRUE | FALSE | Instruction completion signal |
| Busy | Executing flag | BOOL | FALSE/TRUE | FALSE | Instruction execution status |

| | | | | | |
|---|---|---|---|---|---|
| Error | Error flag | BOOL | FALSE/TRUE | FALSE | Instruction error status |
| ErrorID | Error code | WORD | As the data type | - | Instruction error ID |

(3) Function

- Write a variable "WriteVar" in binary form to the file of the SD storage card specified in the file name "FileName".
- When there is no file with the same name as "FileName" in the SD storage card, a new file will be created.
- If a file with the same name as "FileName" already exists in the SD storage card, depending on the value allowed to overwrite "OverWrite", the following steps will be taken:

| "OverWrite" value | Operation |
|---|---|
| TRUE | Overwrite this file |
| FALSE | Not overwriting the file, an exception occurred |

(4) Note

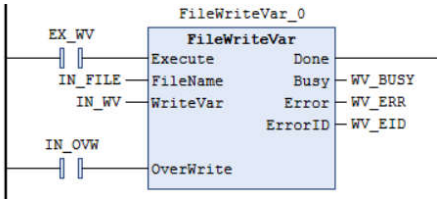
- Even if the value of "Execute" becomes FALSE or the instruction execution time exceeds the task cycle, this instruction will continue to be executed until the end. Whether the processing has ended normally can be confirmed by whether the value of "Done" has changed to true.
- "FileID" needs to be obtained by executing the FileOpen command in advance.
- An exception will occur in the following situations. "Error" is true.
  - When the SD memory card is not in a usable state.
  - The SD storage card is write protected.
  - When the remaining space of the SD storage card is insufficient.
  - When accessing the file specified in "FileID".
  - When the value of "FileName" is an illegal file name.
  - When the number of files and directories that can be created exceeds.
  - When a file with the same name as "FileName" already exists and is being accessed.
  - When a file with the same name as "FileName" already exists and the value of "OverWrite" is FALSE.
  - When a file with the same name as "FileName" already exists and is read-only.
  - When accessing the SD memory card and unable to access due to an exception.

(5) Sample program

| DEMO | LD | ST |
|---|---|---|
| Define the variables | | |

```
VAR
    FileWriteVar_0: FileWriteVar;
    EX_WV: BOOL;
    IN_FILE: STRING(1985):= 'DEMO/TESTFILE/VERSION.TXT';
    IN_WV: ARRAY[0..2] OF BYTE:= [64,32,16];
    IN_OVW: BOOL;
    WV_DONE: BOOL;
    WV_BUSY: BOOL;
    WV_ERR: BOOL;
    WV_EID: WORD;
END_VAR
```

Sample program




```
FileWriteVar_0(
    Execute:= EX_WV,
    FileName:= IN_FILE,
    WriteVar:= IN_WV,
    OverWrite:= IN_OVW,
    Done=> WV_DONE,
    Busy=> WV_BUSY,
    Error=> WV_ERR,
    ErrorID=> WV_EID);
```

Operation result

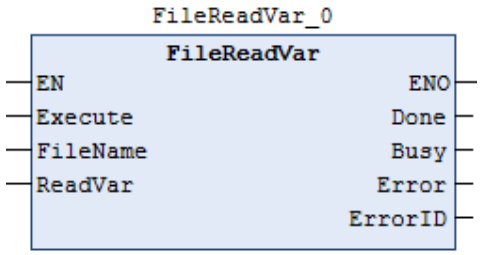
| 表达式 | 类型 | 值 |
|---|---|---|
| FileWriteVar_0 | FileWriteVar | |
| EX_WV | BOOL | TRUE |
| IN_FILE | STRING(1985) | 'DEMO/TESTFILE/VERSION.TXT' |
| IN_WV | ARRAY [0..2] OF BYTE | |
| IN_WV[0] | BYTE | 64 |
| IN_WV[1] | BYTE | 32 |
| IN_WV[2] | BYTE | 16 |
| IN_OVW | BOOL | TRUE |
| WV_DONE | BOOL | TRUE |
| WV_BUSY | BOOL | FALSE |
| WV_ERR | BOOL | FALSE |
| WV_EID | WORD | 0 |

## 8-11. Read variable file FileReadVar

(1) Instruction overview

Read the value of the specified file from the SD memory card in binary form and write it into a variable.

| Read variable file [FileReadVar] | | | |
|---|---|---|---|
| Execution condition | Edge triggering | Suitable model | XS, XSA |
| Firmware | - | Software | XS StudioV2.2.0 and above |
| Library | XJ_FunctionUtils | | |

| Instruction | Name | Graph | ST |
|---|---|---|---|
| FileReadVar | Read variable file |  | FileReadVar_0(<br>Execute:= ,<br>FileName:= ,<br>ReadVar:= ,<br>Done=> ,<br>Busy=> ,<br>Error=> ,<br>ErrorID=> ); |

(2) Related variables

| VAR_INPUT | Name | Data type | Effective range | Initial value | Description |
|---|---|---|---|---|---|
| Execute | Instruction enable | BOOL | FALSE/TRUE | FALSE | Input high-level rising edge to start command |
| FileName | The specified file name | STRING | Maximum 1985 characters | - | The file name of the file to read variable data from |
| ReadVar | Specify reading variables | BOOL/ BYTE/WORD/DWORD/LWORD/ USINT/UINT/UDINT/ULINT/ SINT/INT/DINT/LINT/ REAL/LREAL/ TIME/DATE/TOD/DT/STRING | As the data type | - | Save read data variables |
| VAR_OUTPUT | Name | Data type | Effective range | Initial value | Description |
| Done | Completed flag | BOOL | FALSE/TRUE | FALSE | Instruction completion signal |
| Busy | Executing flag | BOOL | FALSE/TRUE | FALSE | Instruction execution status |
| Error | Error flag | BOOL | FALSE/TRUE | FALSE | Instruction error status |
| ErrorID | Error code | WORD | As the data type | - | Instruction error ID |

(3) Function

- Read variable data in binary form for the SD storage card file specified in the file name "FileName", and substitute the read value into the read location variable "ReadVar".

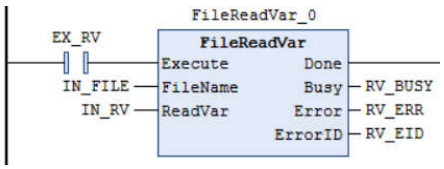
(4) Note

- Even if the value of "Execute" becomes FALSE or the instruction execution time exceeds the task cycle, this instruction will continue to be executed until the end. Whether the processing has ended normally can be confirmed by whether the value of "Done" has changed to true.
- When the size of the specified file is greater than the size of "ReadVar", no exception will occur, and only the data corresponding to the size of "ReadVar" will be read.
- When the size of the specified file is less than the size of "ReadVar", no exception will occur, and only the data corresponding to the size of the specified file will be read. The remaining area of "ReadVar" will maintain its value before executing this instruction.
- When "ReadVar" is the entire structure, adjustment areas may be inserted between members depending on the specific structure.
- An exception will occur in the following situations. "Error" is true.
  - When the SD memory card is not in a usable state.
  - When the value of "FileName" is an illegal file name.
  - When the file specified in "FileName" does not exist.
  - When accessing the file specified in "FileName".
  - When accessing the SD memory card and unable to access due to an exception.

(5) Sample program

<table>
<tr><th>DEMO</th><th>LD</th><th>ST</th></tr>
<tr><td>Define the variables</td><td colspan="2">

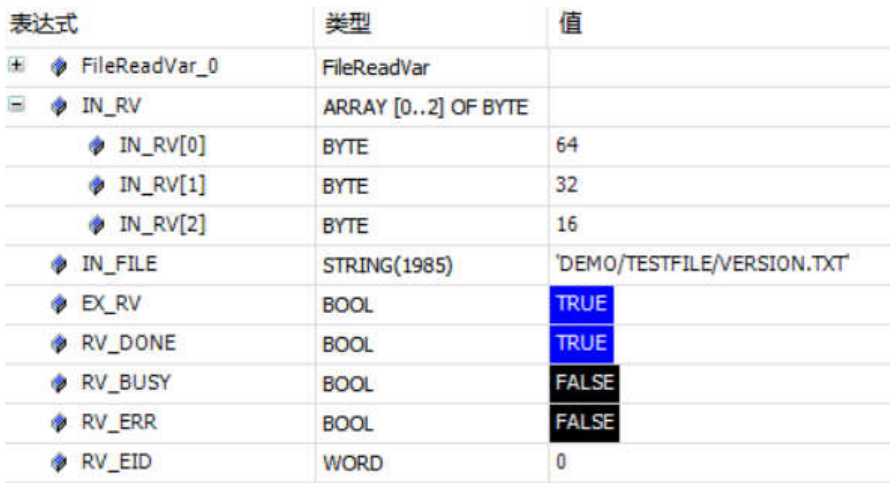
```
VAR
    FileReadVar_0: FileReadVar;
    IN_RV: ARRAY[0..2] OF BYTE;
    IN_FILE: STRING(1985):= 'DEMO/TESTFILE/VERSION.TXT';
    EX_RV: BOOL;
    RV_DONE: BOOL;
    RV_BUSY: BOOL;
    RV_ERR: BOOL;
    RV_EID: WORD;
    BCV:BOOL;
END_VAR
```

</td></tr>
<tr><td>Sample program</td><td>


</td><td>

```
FileReadVar_0(
    Execute:= EX_RV,
    FileName:= IN_FILE,
    ReadVar:= IN_RV,
    Done=> RV_DONE,
    Busy=> RV_BUSY,
    Error=> RV_ERR,
    ErrorID=> RV_EID);
```

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

| | |
|---|---|
| Operation result |  |

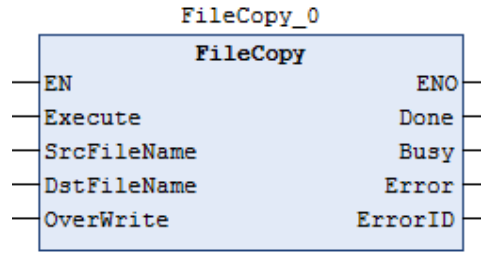
| 表达式 | 类型 | 值 |
|---|---|---|
| FileReadVar_0 | FileReadVar | |
| IN_RV | ARRAY [0..2] OF BYTE | |
| IN_RV[0] | BYTE | 64 |
| IN_RV[1] | BYTE | 32 |
| IN_RV[2] | BYTE | 16 |
| IN_FILE | STRING(1985) | 'DEMO/TESTFILE/VERSION.TXT' |
| EX_RV | BOOL | TRUE |
| RV_DONE | BOOL | TRUE |
| RV_BUSY | BOOL | FALSE |
| RV_ERR | BOOL | FALSE |
| RV_EID | WORD | 0 |

## 8-12. Copy file FileCopy

(1) Instruction overview

Copy the specified file from the SD storage card.

| Copy file [FileCopy] | | | |
|---|---|---|---|
| Execution condition | Edge triggering | Suitable model | XS, XSA |
| Firmware | - | Software | XS StudioV2.2.0 and above |
| Library | XJ_FunctionUtils | | |

| Instruction | Name | Graph | ST |
|---|---|---|---|
| FileCopy | Copy files |  | FileCopy_0(<br>Execute:= ,<br>SrcFileName:= ,<br>DstFileName:= ,<br>OverWrite:= ,<br>Done=> ,<br>Busy=> ,<br>Error=> ,<br>ErrorID=> ); |

(2) Related variables

| VAR_INPUT | Name | Data type | Effective range | Initial value | Description |
|---|---|---|---|---|---|
| Execute | Instruction enable | BOOL | FALSE/TRUE | FALSE | Input high-level rising edge to start command |
| StcFileName | Copy source file name | STRING | Maximum 1985 characters | - | Copy source file name |
| DstFileName | The target file name for pasting | STRING | Maximum 1985 characters | - | The target file name for pasting |
| OverWrite | Overlay selection | BOOL | FALSE/TRUE | FALSE | True: Allow overwrite<br>FALSE: Do not overwrite |
| VAR_OUTPUT | Name | Data type | Effective range | Initial value | Description |
| Done | Completed flag | BOOL | FALSE/TRUE | FALSE | Instruction completion signal |
| Busy | Executing flag | BOOL | FALSE/TRUE | FALSE | Instruction execution status |
| Error | Error flag | BOOL | FALSE/TRUE | FALSE | Instruction error status |
| ErrorID | Error code | WORD | As the data type | - | Instruction error ID |

(3) Function

- Copy the file from the SD storage card specified in the source file name "SrcFileName" and paste it into the destination file name "DstFileName".
- If there is no file with the same name as "DstFileName" in the SD storage card, a new file will be created.
- If a file with the same name as "DstFileName" already exists in the SD storage card, the following steps will be taken based on the allowed value of "OverWrite" to be overwritten:

| “OverWrite” value | Operation |
|---|---|
| TRUE | Overwrite this file |

| | |
|---|---|
| FALSE | Not overwriting file, exception occurred |

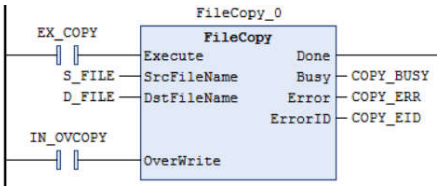
(4) Note

- Even if the value of "Execute" becomes FALSE or the instruction execution time exceeds the task cycle, this instruction will continue to be executed until the end. Whether the processing has ended normally can be confirmed by whether the value of "Done" has changed to true.
- When the SD storage card is unplugged while the file is open, the file will remain open. But when reinstalling the SD memory card, it will not be possible to read or write files. To read and write the file, please reopen the file.
- An exception will occur in the following situations. "Error" is true.
  - When the SD memory card is not in a usable state.
  - The SD storage card is write protected.
  - When the remaining space of the SD storage card is insufficient.
  - When the file specified in "SrcFileName" does not exist.
  - When the value of "SrcFileName" is an illegal file name.
  - When the value of "DstFileName" is an illegal file name.
  - When the number of files and directories that can be created exceeds.
  - When a file with the same name as "DstFileName" already exists and is being accessed.
  - When a file with the same name as "DstFileName" already exists and the value of "OverWrite" is FALSE.
  - When a file with the same name as "DstFileName" already exists and is read-only.
  - When accessing the SD memory card and unable to access due to an exception.

(5) Sample program

<table>
<tr><th>DEMO</th><th>LD</th><th>ST</th></tr>
<tr><td>Define the variables</td><td colspan="2">

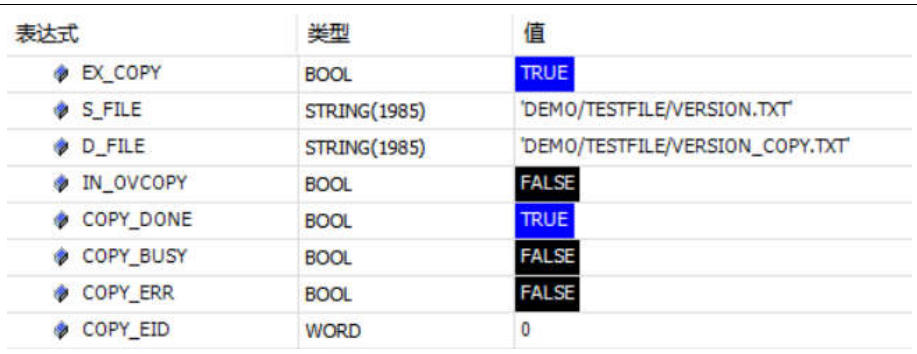
```
VAR
   FileCopy_0: FileCopy;
   EX_COPY: BOOL;
   S_FILE: STRING(1985):= 'DEMO/TESTFILE/VERSION.TXT';
   D_FILE: STRING(1985):= 'DEMO/TESTFILE/VERSION_COPY.TXT';
   IN_OVCOPY: BOOL;
   COPY_DONE: BOOL;
   COPY_BUSY: BOOL;
   COPY_ERR: BOOL;
   COPY_EID: WORD;
END_VAR
```

</td></tr>
<tr><td>Sample program</td><td>


</td><td>

```
FileCopy_0(
    Execute:= EX_COPY,
    SrcFileName:= S_FILE,
    DstFileName:= D_FILE,
    OverWrite:= IN_OVCOPY,
    Done=> COPY_DONE,
    Busy=> COPY_BUSY,
    Error=> COPY_ERR,
    ErrorID=> COPY_EID);
```

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

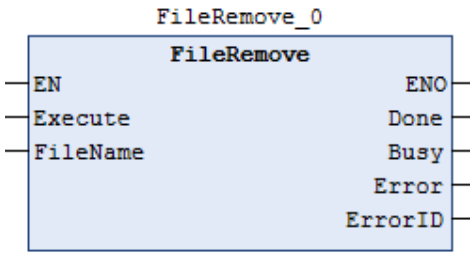
| | |
|---|---|
| Operation result |  |

## 8-13. Delete file FileRemove

(1) Instruction overview

Delete the specified file from the SD storage card.

| Delete file [FileRemove] | | | |
|---|---|---|---|
| Execution condition | Edge triggering | Suitable model | XS, XSA |
| Firmware | - | Software | XS StudioV2.2.0 and above |
| Library | XJ_FunctionUtils | | |

| Instruction | Name | Graph | ST |
|---|---|---|---|
| FileRemove | Delete file |  | FileRemove_0(<br>Execute:= ,<br>FileName:= ,<br>Done=> ,<br>Busy=> ,<br>Error=> ,<br>ErrorID=> ); |

(2) Related variables

| VAR_INPUT | Name | Data type | Effective range | Initial value | Description |
|---|---|---|---|---|---|
| Execute | Instruction enable | BOOL | FALSE/TRUE | FALSE | Input high-level rising edge to start command |
| FileName | The specified file name | STRING | Maximum 1985 characters | - | The file name to be deleted |
| VAR_OUTPUT | Name | Data type | Effective range | Initial value | Description |
| Done | Completed flag | BOOL | FALSE/TRUE | FALSE | Instruction completion signal |
| Busy | Executing flag | BOOL | FALSE/TRUE | FALSE | Instruction execution status |
| Error | Error flag | BOOL | FALSE/TRUE | FALSE | Instruction error status |
| ErrorID | Error code | WORD | As the data type | - | Instruction error ID |

(3) Function

- Delete the file from the SD storage card specified in the specified file name "FileName".

(4) Note

- Even if the value of "Execute" becomes FALSE or the instruction execution time exceeds the task cycle, this instruction will continue to be executed until the end. Whether the processing has ended normally can be confirmed by whether the value of "Done" has changed to true.
- When the SD storage card is unplugged while the file is open, the file will remain open. But when reinstalling the SD memory card, it will not be possible to read or write files. To read and write the file, please reopen the file.
- An exception will occur in the following situations. "Error" is true.
  - When the SD memory card is not in a usable state.
  - The SD storage card is write protected.
  - When the file specified in "FileName" does not exist.

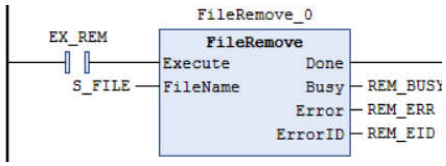
- When accessing the file specified in "FileName".
- When a file with the same name as "FileName" already exists and is read-only.
- When accessing the SD memory card and unable to access due to an exception.

(5) Sample program

<table>
<tr><th>DEMO</th><th>LD</th><th>ST</th></tr>
<tr><td>Define the variables</td><td colspan="2">

```
VAR
    FileRemove_0: FileRemove;
    EX_REM: BOOL;
    S_FILE: STRING(1985):= 'DEMO/TESTFILE/VERSION.TXT';
    REM_DONE: BOOL;
    REM_BUSY: BOOL;
    REM_ERR: BOOL;
    REM_EID: WORD;
END_VAR
```

</td></tr>
<tr><td>Sample program</td><td>




</td><td>

```
FileRemove_0(
    Execute:= EX_REM,
    FileName:= S_FILE,
    Done=> REM_DONE,
    Busy=> REM_BUSY,
    Error=> REM_ERR,
    ErrorID=> REM_EID);
```

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

| 表达式 | 类型 | 值 |
|---|---|---|
| FileRemove_0 | FileRemove | |
| EX_REM | BOOL | TRUE |
| S_FILE | STRING(1985) | 'DEMO/TESTFILE/VERSION.TXT' |
| REM_DONE | BOOL | TRUE |
| REM_BUSY | BOOL | FALSE |
| REM_ERR | BOOL | FALSE |
| REM_EID | WORD | 0 |

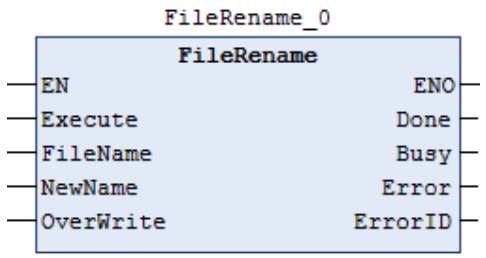
</td></tr>
</table>

## 8-14. Rename file FileRename

(1) Instruction overview

Change the name of the specified file or directory in the SD storage card.

| Rename file [FileRename] | | | |
|---|---|---|---|
| Execution condition | Edge triggering | Suitable model | XS, XSA |
| Firmware | - | Software | XS StudioV2.2.0 and above |
| Library | XJ_FunctionUtils | | |

| Instruction | Name | Graph | ST |
|---|---|---|---|
| FileRename | Rename file |  | FileRename_0(<br>Execute:= ,<br>FileName:= ,<br>NewName:= ,<br>OverWrite:= ,<br>Done=> ,<br>Busy=> ,<br>Error=> ,<br>ErrorID=> ); |

(2) Related variables

| VAR_INPUT | Name | Data type | Effective range | Initial value | Description |
|---|---|---|---|---|---|
| Execute | Instruction enable | BOOL | FALSE/TRUE | FALSE | Input high-level rising edge to start command |
| FileName | Original file name | STRING | Maximum 1985 characters | - | Original file name |
| NewName | The changed file name | STRING | Maximum 1985 characters | - | The changed file name |
| OverWrite | Overlay selection | BOOL | FALSE/TRUE | FALSE | True: Allow overwrite<br>FALSE: Do not overwrite |
| VAR_OUTPUT | Name | Data type | Effective range | Initial value | Description |
| Done | Completed flag | BOOL | FALSE/TRUE | FALSE | Instruction completion signal |
| Busy | Executing flag | BOOL | FALSE/TRUE | FALSE | Instruction execution status |
| Error | Error flag | BOOL | FALSE/TRUE | FALSE | Instruction error status |
| ErrorID | Error code | WORD | As the data type | - | Instruction error ID |

(3) Function

- Change the name of the file or directory specified in the original file name "FileName" to the updated file name "NewName" on the SD storage card.
- If there is no file with the same name as "NewName" in the SD storage card, a new file will be created.
- If a file with the same name as "NewName" already exists in the SD storage card, the following steps will be taken based on the allowed value of "OverWrite" to be overwritten:

| “OverWrite” value | Operation |
|---|---|
| TRUE | Overwrite the file or directory |
| FALSE | Not overwriting the file or directory, an |

| | |
|---|---|
| | exception occurred |

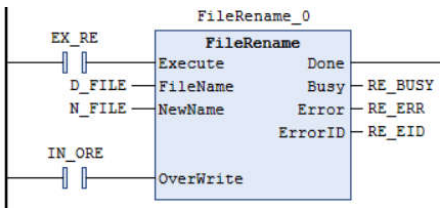
(4) Note

- Even if the value of "Execute" becomes FALSE or the instruction execution time exceeds the task cycle, this instruction will continue to be executed until the end. Whether the processing has ended normally can be confirmed by whether the value of "Done" has changed to true.
- If the directories of "FileName" and "NewName" are different, the file will be moved to the directory specified by "NewName".
- When the SD storage card is unplugged while the file is open, the file will remain open. But when reinstalling the SD memory card, it will not be possible to read or write files. To read and write the file, please reopen the file.
- An exception will occur in the following situations. "Error" is true.
  - When the SD memory card is not in a usable state.
  - The SD storage card is write protected.
  - When the file specified in "FileName" does not exist.
  - When the value of "FileName" or "FileName" is an illegal file name.
  - When accessing the file specified in "FileName".
  - When the number of files and directories that can be created exceeds.
  - When a file with the same name as "FileName" already exists and the value of "OverWrite" is FALSE.
  - When a file with the same name as "DstFileName" already exists and the value of "OverWrite" is true and the file is read-only.
  - When accessing the SD memory card and unable to access due to an exception.

(5) Sample program

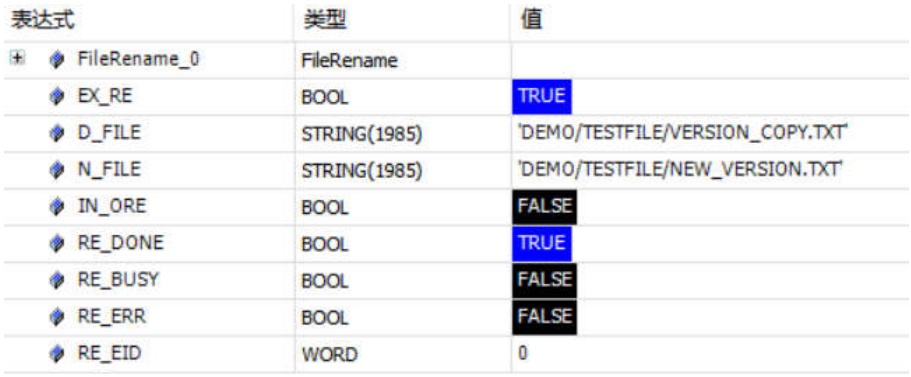
<table>
<tr><th>DEMO</th><th>LD</th><th>ST</th></tr>
<tr><td>Define the variables</td><td colspan="2">

```
VAR
   FileRename_0: FileRename;
   EX_RE: BOOL;
   D_FILE: STRING(1985):= 'DEMO/TESTFILE/VERSION_COPY.TXT';
   N_FILE: STRING(1985):= 'DEMO/TESTFILE/NEW_VERSION.TXT';
   IN_ORE: BOOL;
   RE_DONE: BOOL;
   RE_BUSY: BOOL;
   RE_ERR: BOOL;
   RE_EID: WORD;
END_VAR
```

</td></tr>
<tr><td>Sample program</td><td> 
</td><td>

```
FileRename_0(
    Execute:= EX_RE,
    FileName:= D_FILE,
    NewName:= N_FILE,
    OverWrite:= IN_ORE,
    Done=> RE_DONE,
    Busy=> RE_BUSY,
    Error=> RE_ERR,
    ErrorID=> RE_EID);
```

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

| | |
|---|---|
| Operation result |  |



## 8-15. Command error code

| Error name | Value | Meaning |
|---|---|---|
| ERR_OK | 16#0 | No error |
| ERR_FAILED | 16#1 | General Error - For Internal Errors Only |
| ERR_PARAMETER | 16#2 | The parameters for this operation are invalid |
| ERR_NOTINITIALIZED | 16#3 | The function block cannot be executed because the component has not been initialized yet. However, it may take effect in the future. |
| ERR_VERSION | 16#4 | Version Conflict |
| ERR_TIMEOUT | 16#5 | Operation timeout |
| ERR_NOBUFFER | 16#6 | Insufficient memory, cannot execute request |
| ERR_PENDING | 16#A | For asynchronous calls: not yet completed |
| ERR_NUMPENDING | 16#B | Too many pending requests, try again later |
| ERR_NOTIMPLEMENTED | 16#C | This function is not implemented |
| ERR_INVALIDID | 16#D | The object with the provided ID was not found |
| ERR_OVERFLOW | 16#E | Integer Overflow |
| ERR_BUFFERSIZE | 16#F | The size of the buffer is too small or invalid |
| ERR_NO_OBJECT | 16#10 | No object with this specified name is available |
| ERR_NOMEMORY | 16#11 | No available heap memory |
| ERR_DUPLICATE | 16#12 | Objects with the same name are still available |
| ERR_MEMORY_OVERWRITE | 16#13 | Heap memory write out of bounds! Memory overwrite error |
| ERR_INVALID_HANDLE | 16#14 | Invalid handle for object |
| ERR_END_OF_OBJECT | 16#15 | Arrival object end |
| ERR_NO_CHANGE | 16#16 | No change |
| ERR_INVALID_INTERFACE | 16#17 | Invalid or unknown interface |
| ERR_NOT_SUPPORTED | 16#18 | Not support function |
| ERR_NO_ACCESS_RIGHTS | 16#19 | This operation does not have access permission |
| ERR_OUT_OF_LIMITS | 16#1A | Exceeding the specified limit of resources |
| ERR_ENTRIES_REMAINING | 16#1B | Remaining entries that cannot be transferred due to buffer limitations |
| ERR_INVALID_SESSION_ID | 16#1C | Invalid online session ID |

| Error name | Value | Meaning |
|---|---|---|
| ERR_EXCEPTION | 16#1D | Exception occurred |
| ERR_SIGNATURE_MISMATCH | 16#1E | Signature mismatch of API function |
| ERR_VERSION_MISMATCH | 16#1F | Version mismatch |
| ERR_TYPE_MISMATCH | 16#20 | Type mismatch |
| ERR_ID_MISMATCH | 16#21 | ID mismatch |
| ERR_NO_CONSISTENCY | 16#22 | Consistency Error |
| ERR_NO_COMM_CYCLE | 16#23 | No need for communication loop |
| ERR_DONT_SUSPEND_TASK | 16#24 | Do not suspend tasks after an exception occurs |
| ERR_MEMORY_LOCK_FAILED | 16#25 | Memory cannot be locked during THIS operation |
| ERR_LICENSE_MISSING | 16#26 | License missing at runtime |
| ERR_OPERATION_DENIED | 16#27 | Operation rejected |
| ERR_DEVICE | 16#28 | Equipment error |
| ERR_DISK_FULL | 16#29 | Disk full |
| ERR_CRC_FAILED | 16#30 | Used internally within the runtime |
| ERR_MEDIA_ERASE | 16#31 | Used internally within the runtime |
| ERR_FILE_ERROR | 16#32 | File error. For example, a file cannot be opened for writing as it may be write protected |
| ERR_NO_RETAIN_MEMORY | 16#33 | No available reserved memory |
| ERR_OUT_OF_LIMITS_MIN | 16#34 | Exceeding the minimum limit specified by the resource |
| ERR_OUT_OF_LIMITS_MAX | 16#35 | Exceeding the minimum limit specified by the resource |
| ERR_SIZE_MISMATCH | 16#36 | Size mismatch |
| ERR_CALL_AGAIN | 16#37 | The operation is not yet completed. Please call the function again to continue |
| ERR_NOTHING_TO_DO | 16#38 | It is unrelated to the operation. Not executed. |
| ERR_SECURITY_CHECKS_FAILED | 16#39 | Some security checks have failed. This is a generic error code used to report this error on the public channel. In this case, the error code did not provide a detailed reason for the error. |
| ERR_INVALID_REFERENCE | 16#3B | Failed to dereference IEC reference in IecVarAccess due to invalid destination address, such as NULL. |
| ERR_CONVERSION_INCOMPLETE | 16#3C | The conversion of string encoding is not lossless. |
| ERR_SOCK_NOTINITIALIZED | 16#201 | Socket not initialized |
| ERR_SOCK_NOTSOCKET | 16#202 | The socket handle provided is invalid |
| ERR_SOCK_AFUNSUPPORTED | 16#203 | Unsupported address family |
| ERR_SOCK_PROTOUNSUPPORTED | 16#204 | Protocol not supported |
| ERR_SOCK_NOBUFFER | 16#205 | There is not enough buffer to process the request |
| ERR_SOCK_WOULDBLOCK | 16#206 | The socket is in non blocking mode, but the THIS calling will block it |
| ERR_SOCK_ADDRINUSE | 16#207 | The provided address has been used |
| ERR_SOCK_ADDRNOTAVAILABLE | 16#208 | The provided address is not available on this computer |

| Error name | Value | Meaning |
|---|---|---|
| ERR_SOCK_CONNREFUSED | 16#209 | The remote host has refused the connection |
| ERR_SOCK_TIMEDOUT | 16#20A | Operation timeout |
| ERR_SOCK_HOSTNOTFOUND | 16#20B | Host not found |
| ERR_SOCK_HOSTUNREACHABLE | 16#20C | Host Unreachable |
| ERR_SOCK_ISCONNECTED | 16#20D | The socket has been connected |
| ERR_SOCK_NOTCONNECTED | 16#20E | Socket is not connected |
| ERR_SOCK_SHUTDOWN | 16#20F | Called shutdown on socket |
| ERR_SOCK_MSGSIZE | 16#210 | Suitable for DGRAM type sockets. The packet to be sent exceeds the maximum packet size |
| ERR_SOCK_CLOSED | 16#211 | The socket has been closed normally. No longer allowed to send/receive |
| ERR_L7_TAG_MISSING | 16#300 | Missing label in online communication buffer |
| ERR_L7_UNKNOWNCMDGROUP | 16#301 | Unknown command group |
| ERR_L7_UNKNOWNCMD | 16#302 | Unknown command (in valid command group) |
| ERR_L7_INCOMPLETE | 16#303 | Level 7 service not completed |
| ERR_CERT_UNABLE_TO_GET_ISSUER_CERT | 16#701 | Illegal error (for uninitialized values, to avoid ERR_CERT-OK) |
| ERR_CERT_SUBJECT_ISSUER_MISMATCH | 16#71C | When searching for issuer certificates, these are 'informational' |
| ERR_REDU_TOO_TAGS | 16#B00 | Visual redundancy error |

# 9. FTP communication

File Transfer Protocol (FTP) is a widely used standard protocol for file transfer over a network. As a fundamental tool in network communication, FTP allows users to interact with servers via client software to perform file uploads, downloads, and other file operations. FTP operates at the application layer of the OSI model and typically uses TCP as its transport protocol to ensure the reliability and orderly delivery of data.

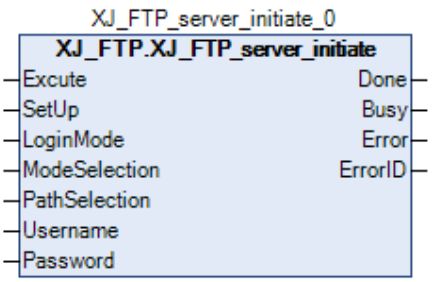
## 9-1. XSF as a server instruction

### 9-1-1. Server startupXJ_FTP_server_initiate

(1) Instruction overview

Startup the server

| Server startup [XJ_FTP_server_initiate] | | | |
|---|---|---|---|
| Execution condition | Edge triggering | Suitable model | XSF series |
| Firmware | V2.4.1 and above | Software | XS StudioV2.3.1 and above |
| Library | XJ_FTP | | |

| Instruction | Name | Graphic representation | ST language |
|---|---|---|---|
| XJ_FTP_server_initiate | Startup the server |  | XJ_FTP_server_initiate_0(<br>Excute:= ,<br>SetUp:= ,<br>LoginMode:= ,<br>ModeSelection:= ,<br>PathSelection:= ,<br>Username:= ,<br>Password:= ,<br>Done=> ,<br>Busy=> ,<br>Error=> ,<br>ErrorID=> ); |

(2) Related variables

| VAR_INPUT | Name | Data type | Effective range | Initial value | Description |
|---|---|---|---|---|---|
| Excute | Execute | BOOL | FALSE/TRUE | FALSE | Execute the function block on the rising edge |
| SetUp | Setup | BYTE | - | 0 | 0: not start<br>1: start |
| LoginMode | Login mode | BYTE | - | 0 | Login mode selection<br>0: Anonymous login<br>1: Virtual user login (reserved but not currently supported) |
| Username | User name | STRING | 1~30 | xinje | User login username<br>(Reserved but not currently supported) |
| Password | Password | STRING | 1~30 | xinje | User Login User Password<br>(Reserved but not currently supported) |

| VAR_INPUT | Name | Data type | Effective range | Initial value | Description |
|---|---|---|---|---|---|
| ModeSelection | Mode selection | BYTE | - | 0 | Mode selection<br>0: Passive mode<br>1: Active mode |
| PathSelection | Path selection | BYTE | - | 0 | Path selection<br>0: /FTP/Share_files_anonymity<br>1: /FTP/Temporary_file_storage |
| VAR_OUTPUT | Name | Data type | Effective range | Initial value | Description |
| Done | Execution completed | BOOL | FALSE/TRUE | FALSE | Execution completed |
| Busy | Executing | BOOL | FALSE/TRUE | FALSE | Executing |
| Error | Error flag | BOOL | FALSE/TRUE | FALSE | Error flag |
| ErrorID | Error code | SYS_XJ_ERR | - | - | Error code |

(3) Function description

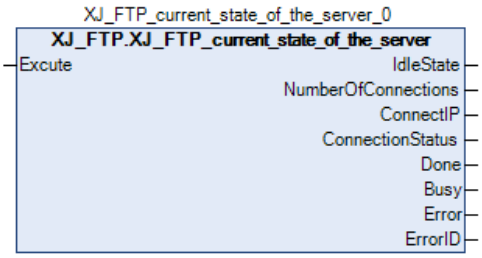
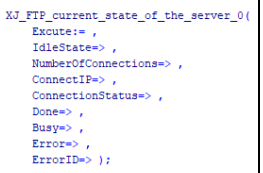
- At this time, the PLC is used as an FTP server, which can be connected to by an FTP client and supports up to 16 simultaneous client connections. The PLC can only trigger one XJ_FTP_server_initiate instruction to create a server connection.
- When the server login mode is anonymous, there is no need to enter username and password parameters in the instruction.
  - When the client is an Xinje PLC, the client needs to use XJ_FTP_login_Server instruction. When the Xinje PLC logs in to the FTP server as a client, the server must be enabled and support anonymous login. During anonymous login, there is no need to fill in a username and password in the client instruction.
  - When the client is a non-Xinje PLC accessing our XSF server, the client side needs to configure a username and password.
- Passive mode: Allows multiple clients to connect and exchange data at the same time. Ports used: random (data connection), 21 (control connection).
- Active mode: Does not allow multiple clients to connect and exchange data simultaneously. Ports used: 20 (data connection), 21 (control connection).

## 9-1-2. Server current state XJ_FTP_current_state_of_the_server

(1) Instruction overview

Get the current status of the server

| Get the current status of the server [XJ_FTP_current_state_of_the_server] | | | |
|---|---|---|---|
| Execution condition | Edge triggering | Suitable model | XSF series |
| Firmware | V2.4.1 and above | Software | XS StudioV2.3.1 and above |
| Library | XJ_FTP | | |

| Instruction | Name | Graphic representation | ST language |
|---|---|---|---|
| XJ_FTP_current_state_of_the_server | Server current state |  |  |

(2) Realted variables

| VAR_INPUT | Name | Data type | Effective range | Initial value | Description |
|---|---|---|---|---|---|
| Excute | Execute | BOOL | FALSE/TRUE | FALSE | Execute the function block on the rising edge |
| **VAR_OUTPUT** | **Name** | **Data type** | **Effective range** | **Initial value** | **Description** |
| IdleState | Idle state | BOOL | FALSE/TRUE | FALSE | Is the server idle TRUE: Busy False: Idle |
| NumberOfConnections | Connection numbers | INT | - | 0 | Connection numbers |
| ConnectIP | Connected IP | ARRAY[0..32] OF STRING | - | 0 | Connected client IP |
| ConnectionStatus | Connection status | ARRAY[0..32] OF BYTE | - | - | Reserved |
| Busy | Executing | BOOL | FALSE/TRUE | FALSE | Executing |
| Error | Error flag | ARRAY[0..32] OFBOOL | FALSE/TRUE | FALSE | Error flag |
| ErrorID | Error code | ARRAY[0..32] OF SYS_XJ_ERR | - | 0 | Error code |

## 9-2. XSF as a client instruction

### 9-2-1. Connect the server XJ_FTP_login_Server

(1) Instruction overview

Connect to the client and log in to the server.

| Connect and login the server [XJ_FTP_login_Server] | | | |
|---|---|---|---|
| Execution condition | Edge triggering | Suitable model | XSF series |
| Firmware | V2.4.1 and above | Software | XS StudioV2.3.1 and above |
| Library | XJ_FTP | | |

| Instruction | Name | Graphic representation | ST language |
|---|---|---|---|
| XJ_FTP_login_Server | Connect the server | XJ_FTP_login_Server_0<br>XJ_FTP.XJ_FTP_login_Server<br>Excute, SetUp, LoginMode, Username, Password, IPAddress, ModeSelection<br>Done, xActive, hConnection, Busy, Error, ErrorID | XJ_FTP_login_Server_0(<br>Excute:= ,<br>SetUp:= ,<br>LoginMode:= ,<br>Username:= ,<br>Password:= ,<br>IPAddress:= ,<br>ModeSelection:= ,<br>Done=> ,<br>xActive=> ,<br>hConnection=> ,<br>Busy=> ,<br>Error=> ,<br>ErrorID=> ); |

(2) Related variables

| VAR_INPUT | Name | Data type | Effective range | Initial value | Description |
|---|---|---|---|---|---|
| Excute | Execute | BOOL | FALSE/TRUE | FALSE | Execute the function block on the rising edge |
| SetUp | Connect | BYTE | - | 0 | 0: Disconnect<br>1: Connect |
| ModeSelection | Login mode | BYTE | - | 0 | Login mode selection<br>0: Anonymous login<br>No need for username and user password<br>1: User login<br>Use username and password |
| Username | User name | STRING | 1~30 | - | User name |
| Password | Password | STRING | 1~30 | - | Password |
| IPAddress | Target server IP address | STRING | - | 0 | Target IP address<br>IP_V4_Address |
| Mode_selection | Mode selection | BYTE | | 0 | Mode selection<br>0: Passive mode<br>1: Active mode |

| VAR_OUTPUT | Name | Data type | Effective range | Initial value | Description |
|---|---|---|---|---|---|
| Done | Execution completed | BOOL | FALSE/TRUE | FALSE | Execution completed |
| xActive | Connection flag | BOOL | FALSE/TRUE | FALSE | Server side detects remote client connection handle flag |
| hConnection | connection handle | _XWORD | - | - | The connection handle for establishing communication between the server and the client |
| Busy | Executing | BOOL | FALSE/TRUE | FALSE | Executing |
| Error | Error flag | BOOL | FALSE/TRUE | FALSE | Error flag |
| ErrorID | Error code | SYS_XJ_ERR | - | - | Error code |

(3) Function description

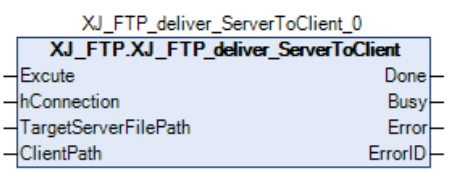
- At this time, the PLC is used as an FTP client, which can connect to an FTP server and supports up to 16 simultaneous server connections.

## 9-2-2. Client read file XJ_FTP_deliver_ServerToClient

(1) Instruction overview

The FTP client (PLC) reads files from the FTP server.

| Client read file [XJ_FTP_deliver_ServerToClient] | | | |
|---|---|---|---|
| Execution condition | Edge triggering | Suitable model | XSF series |
| Firmware | V2.4.1 and above | Software | XS StudioV2.3.1 and above |
| Library | XJ_FTP | | |

| Instruction | Name | Graphic representation | ST language |
|---|---|---|---|
| XJ_FTP_deliver_ServerToClient | Client read file |  | XJ_FTP_deliver_ServerToClient_0(<br>Excute:= ,<br>hConnection:= ,<br>TargetServerFilePath:= ,<br>ClientPath:= ,<br>Done=> ,<br>Busy=> ,<br>Error=> ,<br>ErrorID=> ); |

(2) Related variables

| VAR_INPUT | Name | Data type | Effective range | Initial value | Description |
|---|---|---|---|---|---|
| Excute | Execute | BOOL | FALSE/TRUE | FALSE | Execute the function block on the rising edge |
| hConnection | connection | _XWORD | - | - | The connection handle for |

| | | | | | |
|---|---|---|---|---|---|
| | handle | | | | establishing communication between the server and the client |
| TargetServer FileName | Target server file path/file name | WSTRING | Max 1985 characters | 0 | The file path/file name of the server (the relative address already includes /FTP/, just fill in the subsequent file path) |
| ClientPath | Client path | WSTRING | Max 1985 characters | 0 | Path for storing files on the client side |
| VAR_OUTPUT | Name | Data type | Effective range | Initial value | Description |
| Done | Execution completed | BOOL | FALSE/TRUE | FALSE | Execution completed |
| Busy | Executing | BOOL | FALSE/TRUE | FALSE | Executing |
| VAR_OUTPUT | Name | Data type | Effective range | Initial value | Description |
| Error | Error flag | BOOL | FALSE/TRUE | FALSE | Error flag |
| ErrorID | Error code | SYS_XJ_ERR | - | FALSE | Error code |

(3) Function description

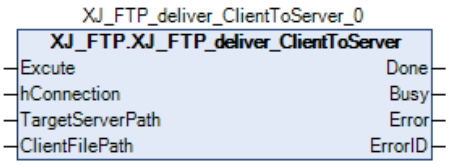
- The files transferred to the FTP client can be stored in the client's `/FTP/Share_files_anonymity/`, `/FTP/Temporary_file_storage/`, or a user-created folder in the root directory. It is generally recommended to choose `/FTP/Share_files_anonymity/` or `/FTP/Temporary_file_storage/`.

## 9-2-3. Client transfer files XJ_FTP_deliver_ClientToServer

(1) Instruction overview

Transfer a file from an FTP client (PLC) to an FTP server.

| Client transfer files [XJ_FTP_deliver_ClientToServer] | | | |
|---|---|---|---|
| Execution condition | Edge triggering | Suitable model | XSF series |
| Firmware | V2.4.1 and above | Software | XS StudioV2.3.1 and above |
| Library | XJ_FTP | | |

| Instruction | Name | Graphic representation | ST language |
|---|---|---|---|
| XJ_FTP_deliver_ClientToServer | Client transfer files |  | XJ_FTP_deliver_ClientToServer_0(<br>Excute:= ,<br>hConnection:= ,<br>TargetServerPath:= ,<br>ClientFilePath:= ,<br>Done=> ,<br>Busy=> ,<br>Error=> ,<br>ErrorID=> ); |

(2) Ralted variables

| VAR_INPUT | Name | Data type | Effective range | Initial value | Description |
|---|---|---|---|---|---|
| Excute | Execute | BOOL | FALSE/TRUE | FALSE | Start the execution of function block at the rising edge |
| hConnection | Connection handle | _XWORD | - | - | Establishing the connection handle for communication between the server and the client. |
| TargetServerName | Target server path | WSTRING | Maximum 1985 characters | 0 | The path where the server stores files (the relative address already includes/FTP/, just fill in the subsequent file path) |
| ClientFilePath | Client file path/file name | WSTRING | Maximum 1985 characters | 0 | Client file path/file name |
| VAR_OUTPUT | Name | Data type | Effective range | Initial value | Description |
| Done | Execution completed | BOOL | FALSE/TRUE | FALSE | Execution completed |
| Busy | Executing | BOOL | FALSE/TRUE | FALSE | Executing |
| Error | Error flag | BOOL | FALSE/TRUE | FALSE | Error flag |
| ErrorID | Error ID | SYS_XJ_ERR | - | FALSE | Error ID |

(3) Function description

ClientFilePath can be written as '/FTP/Share_files_anonymity/' or '/FTP/Temporary_file _storage/' or any file in a

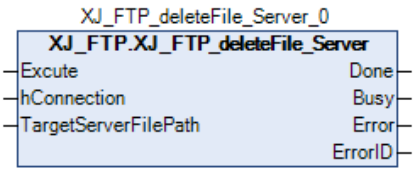
folder created by the user in the root directory. It is generally recommended to choose any file under '/FTP/Share_files_anonymity/' or '/FTP/Temporary_file _storage/'.

## 9-2-4. Delete server files XJ_FTP_deleteFile_Server

(1) Instruction overview

PLC serves as the client to delete the server's designated file.

| Delete server files [XJ_FTP_deleteFile_Server] | | | |
|---|---|---|---|
| Execution condition | Edge triggering | Suitable model | XSF series |
| Firmware | V2.4.1 and above | Software | XS StudioV2.3.1 and above |
| Library | XJ_FTP | | |

| Instruction | Name | Graphic representation | ST language |
|---|---|---|---|
| XJ_FTP_deleteFile_Server | Delete server files |  | XJ_FTP_deleteFile_Server_0(<br>Excute:= ,<br>hConnection:= ,<br>TargetServerFilePath:= ,<br>Done=> ,<br>Busy=> ,<br>Error=> ,<br>ErrorID=> ); |

(2) Related variables

| VAR_INPUT | Name | Data type | Effective range | Initial value | Description |
|---|---|---|---|---|---|
| Excute | Execute | BOOL | FALSE/TRUE | FALSE | Start the execution of function block at the rising edge |
| hConnection | Connection handle | _XWORD | - | - | Establishing the connection handle for communication between the server and the client. |
| TargetServerFileName | Target server file name | WSTRING | Maximum 1985 characters | 0 | Server file path/file name |
| VAR_OUTPUT | Name | Data type | Effective range | Initial value | Description |
| Done | Execution completed | BOOL | FALSE/TRUE | FALSE | Execution completed |
| Busy | Executing | BOOL | FALSE/TRUE | FALSE | Executing |
| Error | Error flag | BOOL | FALSE/TRUE | FALSE | Error flag |
| ErrorID | Error ID | SYS_XJ_ERR | - | FALSE | Error ID |

(3) Function description

The file path or storage path for the FTP server can only be under the /FTP/ directory, depending on the value chosen for PathSelection when starting the server.

Specifically, when the path selection is 0: /FTP/Share_files_anonymity/

when the path selection is 1: /FTP/Temporary_file_storage/.

## 9-3. Example of FTP communication command program

### 9-3-1. Two XSF5 devices establish FTP communication

The IP address for the client PLC is 192.168.6.7, and the IP address for the server PLC is 192.168.6.6.

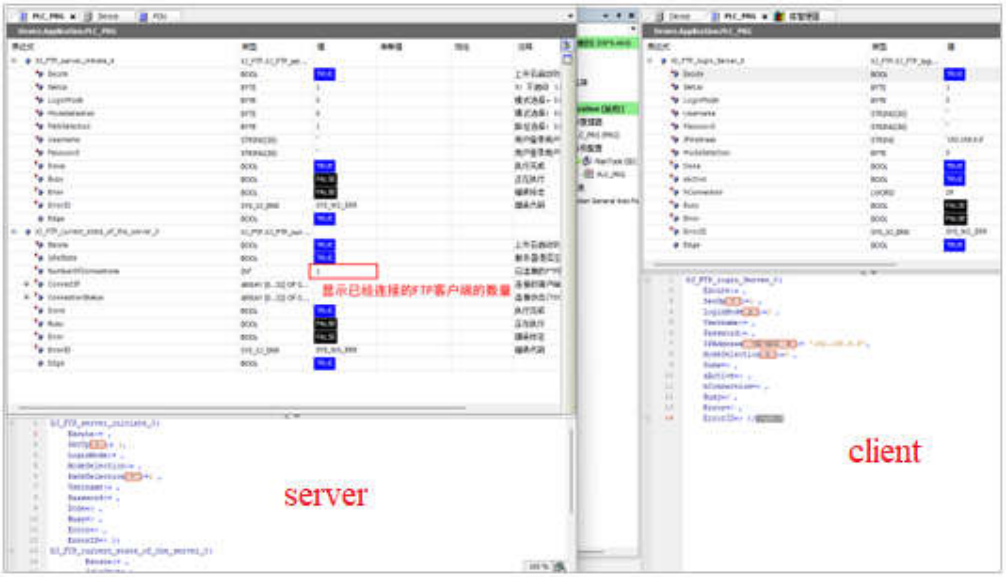
1. Establish an FTP connection between two PLCs. In the client command XJ_FTP_login_Server to connect to the server, select "1: Connect" for the connection status, set the target IP address to the IP address of the server you are connecting to, choose "Anonymous Login" for the login mode, and select passive mode for the mode.

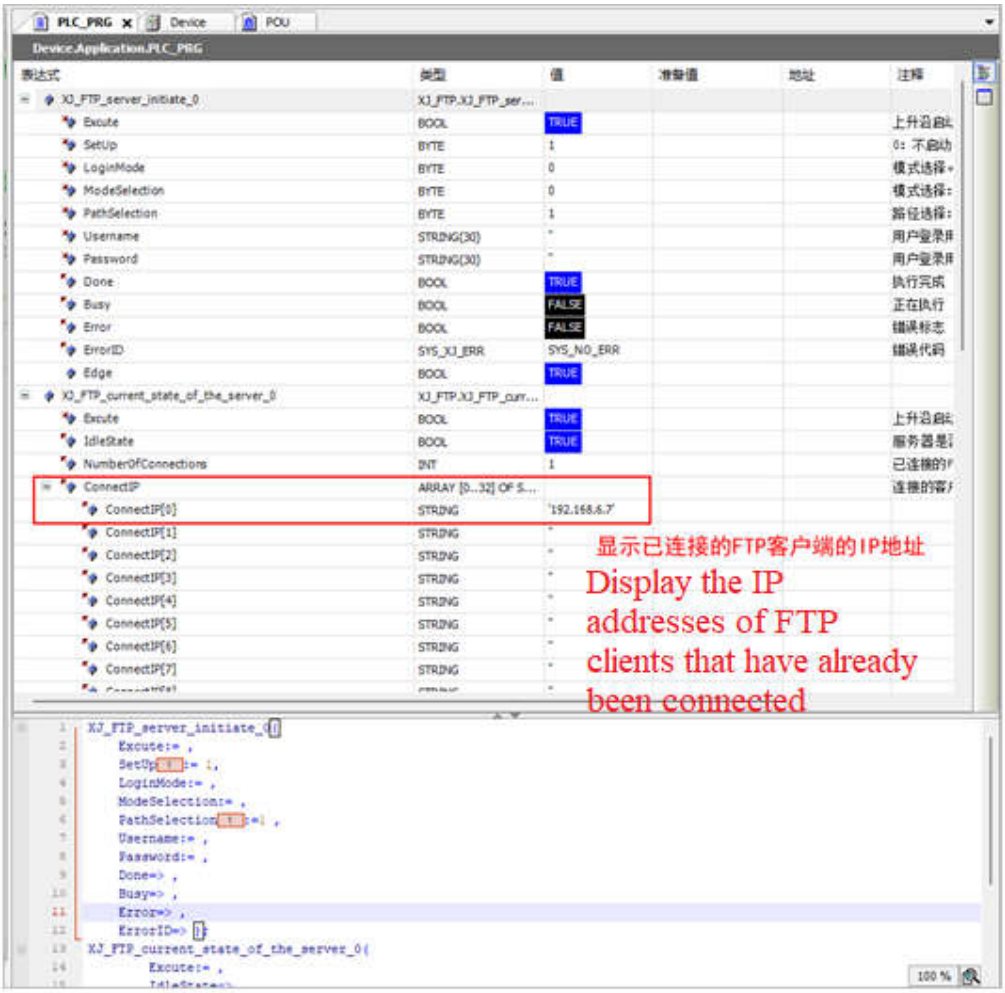
In the command XJ_FTP_server_initiate for the server to start, set the command to "1: Start," choose "Anonymous Login" for the login mode, select "0: File Directory (/FTP/Share_files_anonymity)" for the path selection, and select passive mode for the mode. Additionally, create a command to monitor server connection status to keep track of the connection state.

```
PROGRAM PLC_PRG
VAR
    XJ_FTP_server_initiate_0: XJ_FTP.XJ_FTP_server_initiate;
    XJ_FTP_current_state_of_the_server_0: XJ_FTP.XJ_FTP_current_state_of_the_server;
END_VAR
```

```
//启动服务器
XJ_FTP_server_initiate_0(
    Excute:= ,
    SetUp:=1 ,
    LoginMode:=0 ,
    ModeSelection:=0,
    PathSelection:=0 ,
    Username:= ,
    Password:= ,
    Done=> ,
    Busy=> ,
    Error=> ,
    ErrorID=> );
//监控服务器状态
XJ_FTP_current_state_of_the_server_0(
    Excute:= ,
    IdleState=> ,
    NumberOfConnections=> ,
    ConnectIP=> ,
    ConnectionStatus=> ,
    Done=> ,
    Busy=> ,
    Error=> ,
    ErrorID=> );
```

The current status of the monitoring server shows that it has been connected:






2. Implement the functionality to transfer files from the FTP server to the FTP client. On the client side, create the command XJ_FTP_deliver_ServerToClient, where the connection handle is the hConnection pin from XJ_FTP_login_Server.

```
PROGRAM PLC_PRG
VAR
    XJ_FTP_login_Server_0: XJ_FTP.XJ_FTP_login_Server;
    XJ_FTP_deliver_ServerToClient_0: XJ_FTP.XJ_FTP_deliver_ServerToClient;
    XJ_FTP_deliver_ClientToServer_0: XJ_FTP.XJ_FTP_deliver_ClientToServer;
    XJ_FTP_deleteFile_Server_0: XJ_FTP.XJ_FTP_deleteFile_Server;

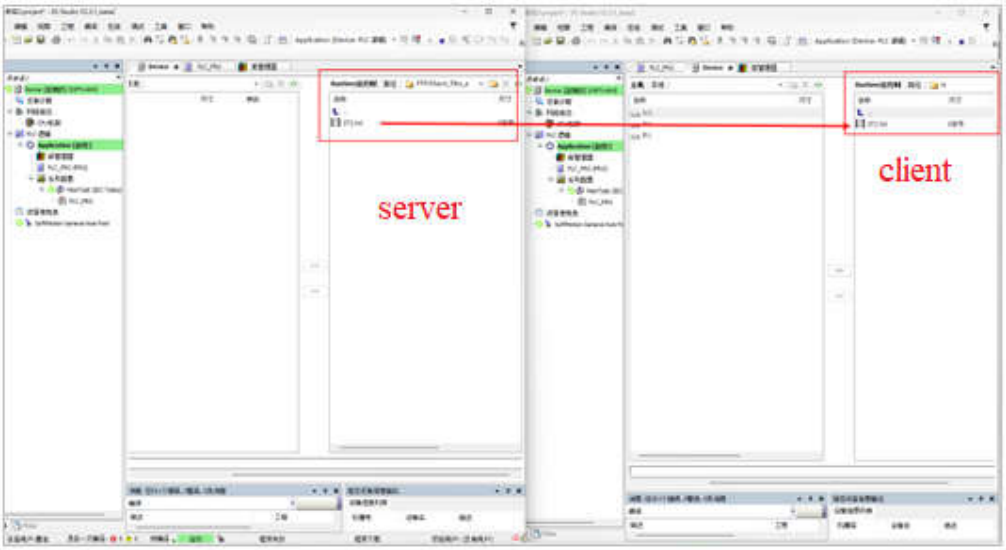
END_VAR
```

```
    Excute:= ,
    SetUp:= 1,
    LoginMode:= 0,
    Username:= ,
    Password:= ,
    IPAddress:= '192.168.6.6',
    ModeSelection:=0 ,
    Done=> ,
    xActive=> ,
    hConnection=> ,
    Busy=> ,
    Error=> ,
    ErrorID=> );
//将文件从FTP服务器传送到FTP客户端
XJ_FTP_deliver_ServerToClient_0(
    Excute:= ,
    hConnection:= XJ_FTP_login_Server_0.hConnection,
    TargetServerFilePath:= ,
    ClientPath:= ,
    Done=> ,
    Busy=> ,
    Error=> ,
    ErrorID=> );
```

Transfer the ST2.txt file from the server's /FTP/Share_files_anonymity path to the /H/ folder in the client's root directory.

Device.Application.PLC_PRG

| 表达式 | 类型 | 值 | 准备值 | 地址 | 注释 |
|---|---|---|---|---|---|
| XJ_FTP_login_Server_0 | XJ_FTP.XJ_FTP... | | | | |
| Excute | BOOL | TRUE | | | 上升沿启动功能块 |
| SetUp | BYTE | 1 | | | 0：断开 1：连接 |
| LoginMode | BYTE | 0 | | | 模式选择。0：匿名登录（无需用户名和用... |
| Username | STRING(30) | '' | | | 登录用户名。 |
| Password | STRING(30) | '' | | | 登录用户密码。 |
| IPAddress | STRING | '192.168.6.6' | | | 目标IP地址 IP_V4_Address |
| ModeSelection | BYTE | 0 | | | 模式选择：0：被动模式 1：主动模式 |
| Done | BOOL | TRUE | | | 执行完成 |
| xActive | BOOL | TRUE | | | 服务器端监听到客户端连接的标志 |
| hConnection | LWORD | 29 | | | 服务器与客户端建立通信的连接句柄 |
| Busy | BOOL | FALSE | | | 正在执行 |
| Error | BOOL | FALSE | | | 错误标志 |
| ErrorID | SYS_XJ_ERR | SYS_NO_ERR | | | 错误代码 |
| Edge | BOOL | TRUE | | | |
| XJ_FTP_deliver_ServerToClient_0 | XJ_FTP.XJ_FTP... | | | | |
| Excute | BOOL | TRUE | | | 上升沿启动功能块 |
| hConnection | LWORD | 29 | | | 服务器与客户端建立通信的连接句柄 |
| TargetServerFilePath | WSTRING(1985) | "/Share_files_anonymity/ST2.txt" | | | 服务器的文件路径/文件名(相对路径) |
| ClientPath | WSTRING(1985) | "/H/" | | | 客户端存放文件的路径。 |
| Done | BOOL | TRUE | | | 执行完成 |
| Busy | BOOL | FALSE | | | 正在执行 |
| Error | BOOL | FALSE | | | 错误标志 |
| ErrorID | SYS_XJ_ERR | SYS_NO_ERR | | | 错误代码 |
| Edge | BOOL | TRUE | | | |




4. Implement the transfer of a file from an FTP client to an FTP server by creating the XJ_FTP_deliver_ClientToServer command on the client side. The connection handle in the command is the hConnection pin from the XJ_FTP_login_Server.

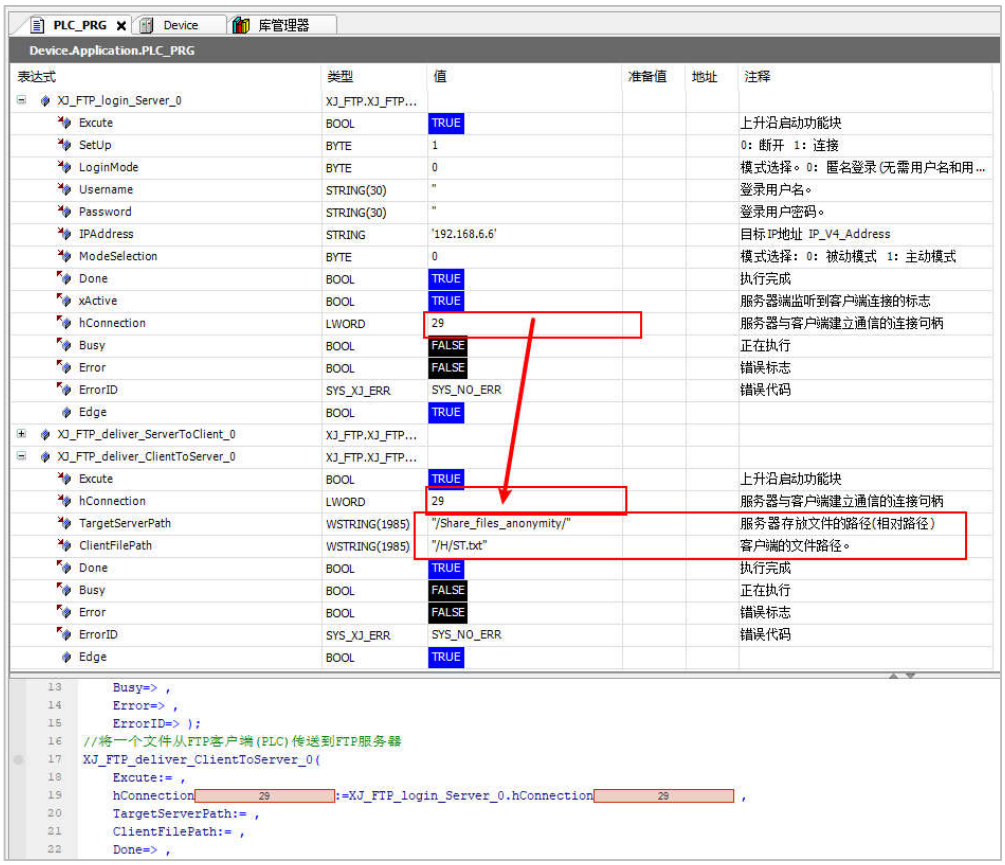
PLC_PRG | Device | 库管理器

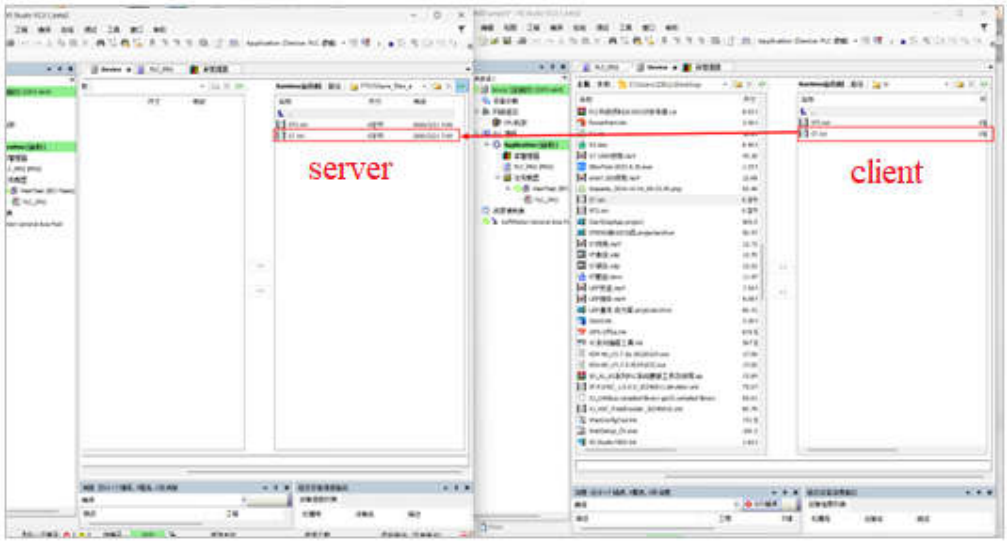
```
PROGRAM PLC_PRG
VAR
    XJ_FTP_login_Server_0: XJ_FTP.XJ_FTP_login_Server;
    XJ_FTP_deliver_ServerToClient_0: XJ_FTP.XJ_FTP_deliver_ServerToClient;
    XJ_FTP_deliver_ClientToServer_0: XJ_FTP.XJ_FTP_deliver_ClientToServer;
    XJ_FTP_deleteFile_Server_0: XJ_FTP.XJ_FTP_deleteFile_Server;

END_VAR
```

```
//连接服务器
XJ_FTP_login_Server_0(
    Excute:= ,
    SetUp:= 1,
    LoginMode:= 0,
    Username:= ,
    Password:= ,
    IPAddress:= '192.168.6.6',
    ModeSelection:=0 ,
    Done=> ,
    xActive=> ,
    hConnection=> ,
    Busy=> ,
    Error=> ,
    ErrorID=> );
//将一个文件从FTP客户端(PLC)传送到FTP服务器
XJ_FTP_deliver_ClientToServer_0(
    Excute:= ,
    hConnection:=XJ_FTP_login_Server_0.hConnection ,
    TargetServerPath:= ,
    ClientFilePath:= ,
    Done=> ,
    Busy=> ,
    Error=> ,
    ErrorID=> );
```

Transfer the ST.txt file from the client's/ H/ path to the server-side's /Share_files_anonymity directory.




5. Implement the PLC as a client to delete a specified file on the server. Create the XJ_FTP_deleteFile_Server command on the client side, and the connection handle in the command is the hConnection pin from the XJ_FTP_login_Server.

```
PROGRAM PLC_PRG
VAR
    XJ_FTP_login_Server_0: XJ_FTP.XJ_FTP_login_Server;
    XJ_FTP_deliver_ServerToClient_0: XJ_FTP.XJ_FTP_deliver_ServerToClient;
    XJ_FTP_deliver_ClientToServer_0: XJ_FTP.XJ_FTP_deliver_ClientToServer;
    XJ_FTP_deleteFile_Server_0: XJ_FTP.XJ_FTP_deleteFile_Server;

END_VAR
```

```
//连接服务器
XJ_FTP_login_Server_0(
    Excute:= ,
    SetUp:= 1,
    LoginMode:= 0,
    Username:= ,
    Password:= ,
    IPAddress:= '192.168.6.6',
    ModeSelection:=0 ,
    Done=> ,
    xActive=> ,
    hConnection=> ,
    Busy=> ,
    Error=> ,
    ErrorID=> );
//PLC作为客户端删除服务器的制定文件
XJ_FTP_deleteFile_Server_0(
    Excute:= ,
    hConnection:=XJ_FTP_login_Server_0.hConnection ,
    TargetServerFilePath:= ,
    Done=> ,
    Busy=> ,
    Error=> ,
    ErrorID=> );
```

The client deletes the ST.txt file in the /Share_files_anonymity directory of the server.

Device.Application.PLC_PRG

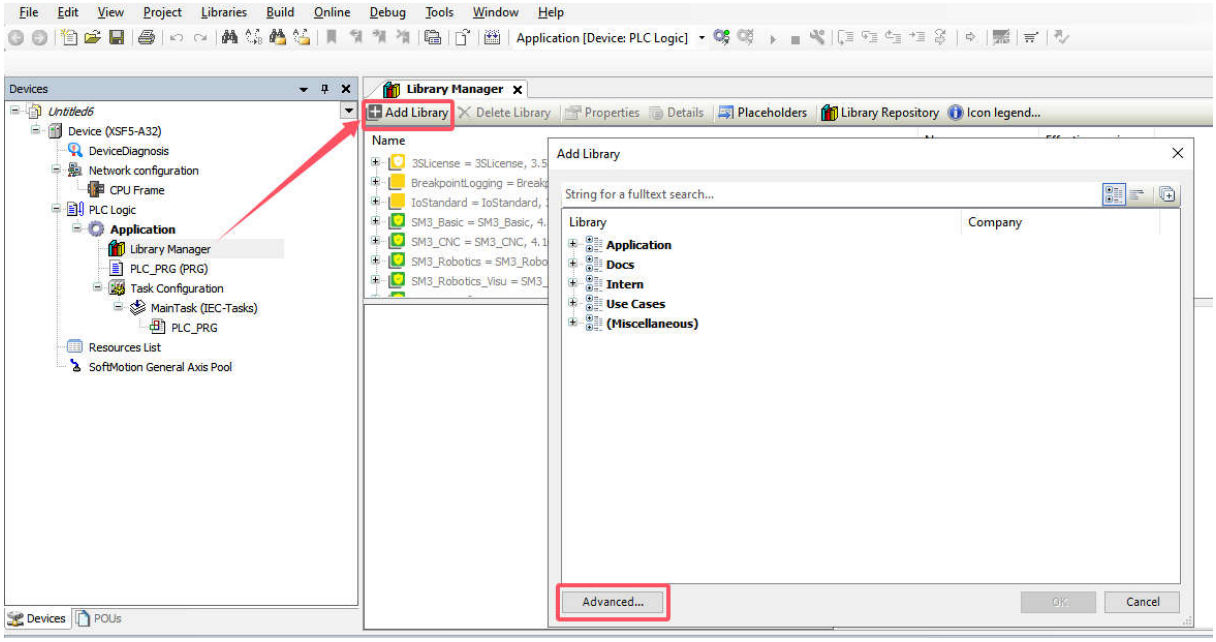
| 表达式 | 类型 | 值 | 准备值 | 地址 | 注释 |
|---|---|---|---|---|---|
| XJ_FTP_login_Server_0 | XJ_FTP.XJ_FTP... | | | | |
| Excute | BOOL | TRUE | | | 上升沿启动功能块 |
| SetUp | BYTE | 1 | | | 0：断开 1：连接 |
| LoginMode | BYTE | 0 | | | 模式选择。0：匿名登录(无需用户名和用... |
| Username | STRING(30) | '' | | | 登录用户名。 |
| Password | STRING(30) | '' | | | 登录用户密码。 |
| IPAddress | STRING | '192.168.6.6' | | | 目标IP地址 IP_V4_Address |
| ModeSelection | BYTE | 0 | | | 模式选择：0：被动模式 1：主动模式 |
| Done | BOOL | TRUE | | | 执行完成 |
| xActive | BOOL | TRUE | | | 服务器端监听到客户端连接的标志 |
| hConnection | LWORD | 29 | | | 服务器与客户端建立通信的连接句柄 |
| Busy | BOOL | FALSE | | | 正在执行 |
| Error | BOOL | FALSE | | | 错误标志 |
| ErrorID | SYS_XJ_ERR | SYS_NO_ERR | | | 错误代码 |
| Edge | BOOL | TRUE | | | |
| XJ_FTP_deliver_ServerToClient_0 | XJ_FTP.XJ_FTP... | | | | |
| XJ_FTP_deliver_ClientToServer_0 | XJ_FTP.XJ_FTP... | | | | |
| XJ_FTP_deleteFile_Server_0 | XJ_FTP.XJ_FTP... | | | | |
| Excute | BOOL | TRUE | | | 上升沿启动功能块 |
| hConnection | LWORD | 29 | | | 服务器与客户端建立通信的连接句柄 |
| TargetServerFilePath | WSTRING(1985) | "/Share_files_anonymity/ST.txt" | | | 服务器的文件路径/文件名(相对路径) |
| Done | BOOL | TRUE | | | 执行完成 |
| Busy | BOOL | FALSE | | | 正在执行 |
| Error | BOOL | FALSE | | | 错误标志 |
| ErrorID | SYS_XJ_ERR | SYS_NO_ERR | | | 错误代码 |
| Edge | BOOL | TRUE | | | |

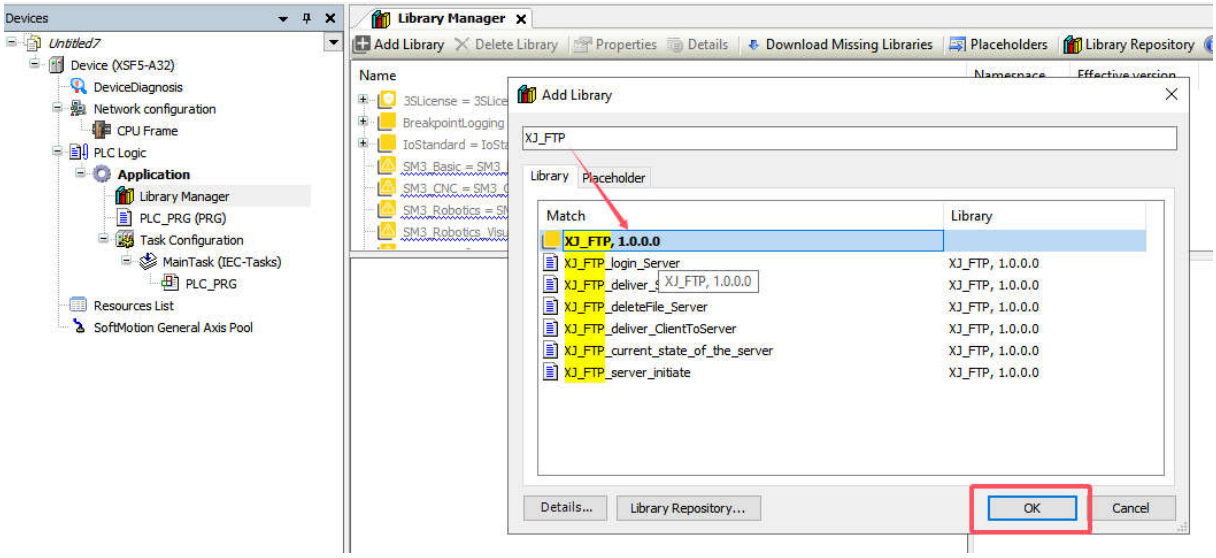
## 9-3-2. Establishing FTP Communication between XSF5 and FileZilla Software

- XSF5 serves as an FTP server

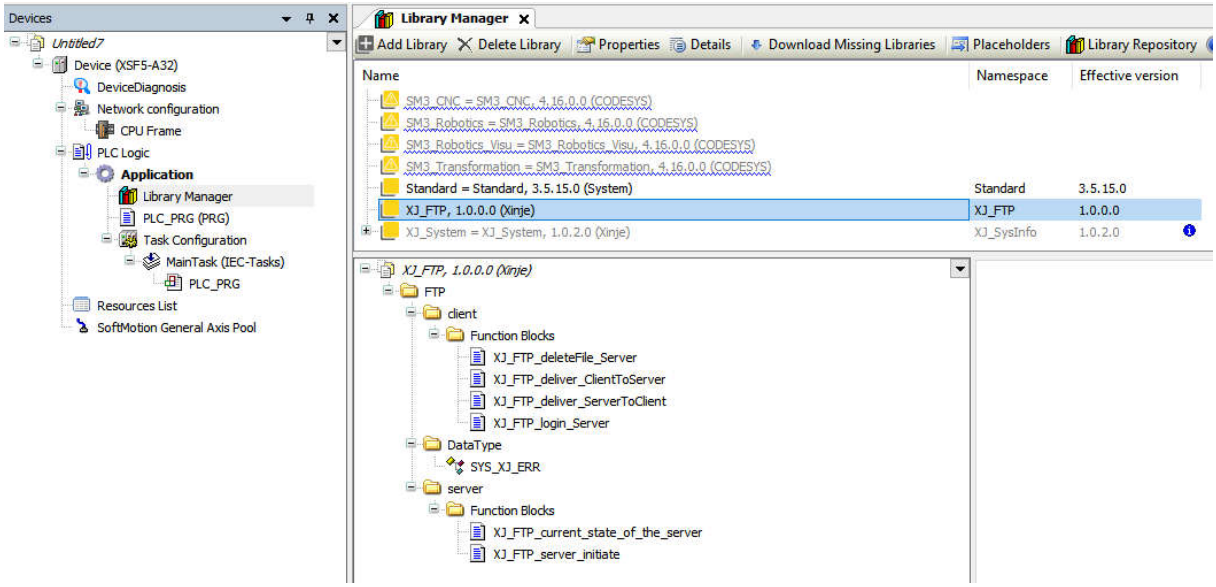
Use the FTP client software FileZillaClient to connect to XSF5.

1. Add FTP library.

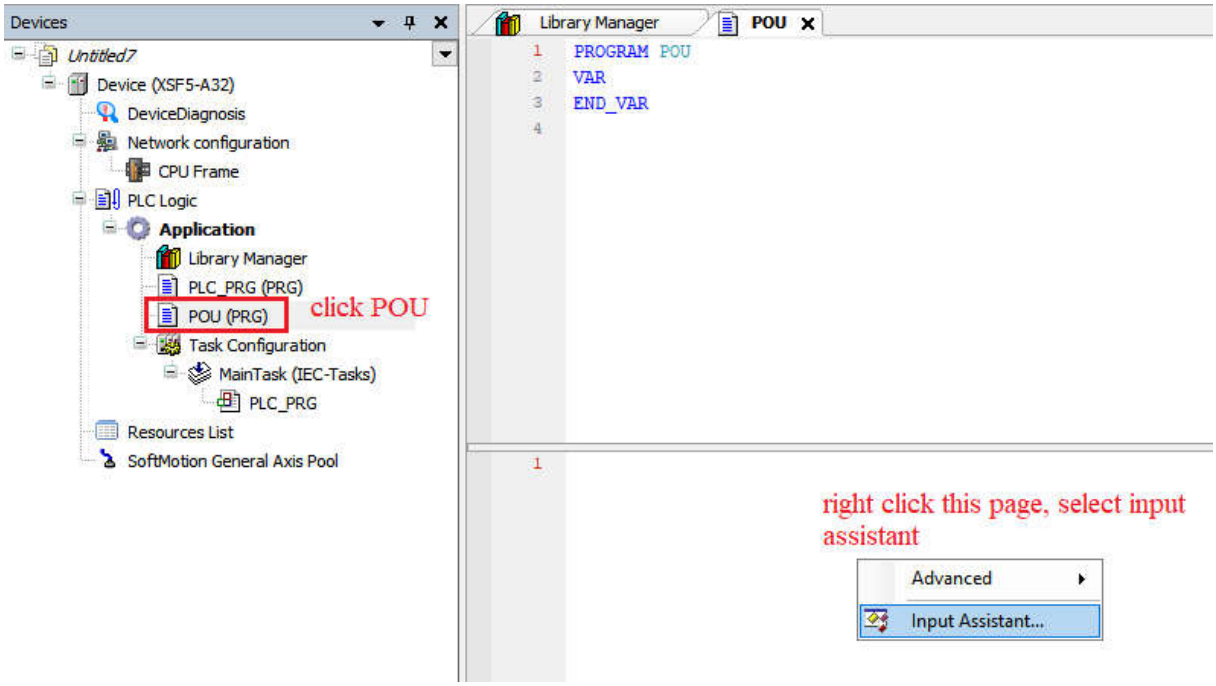


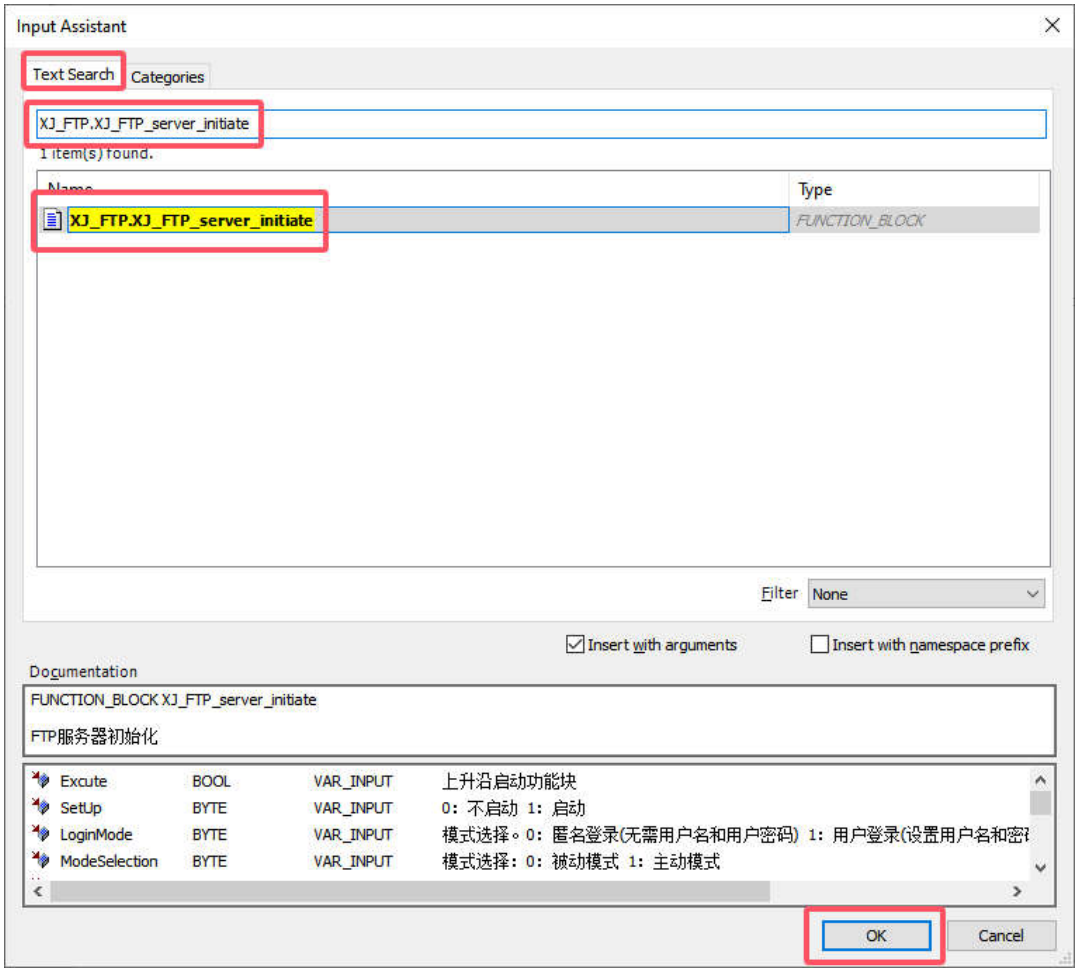


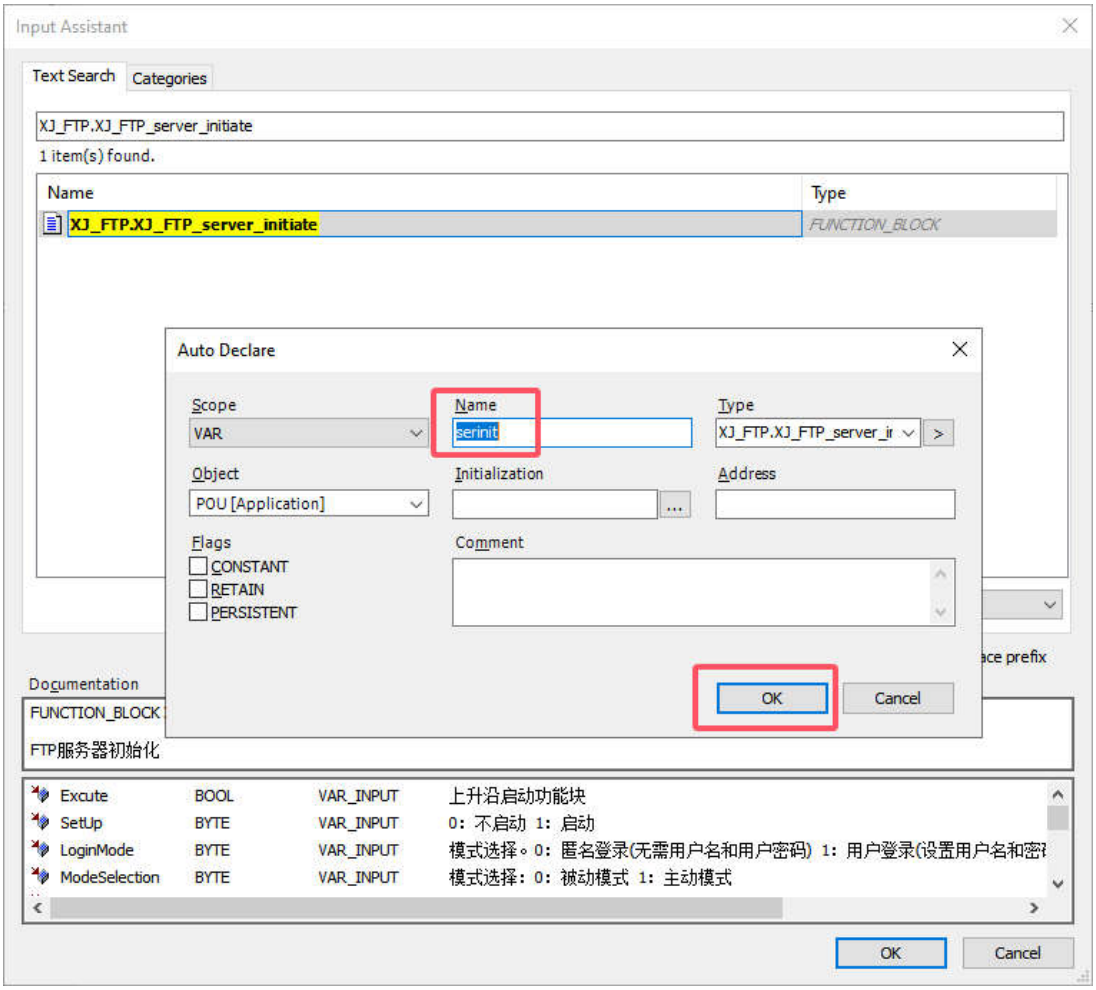

After the addition is complete, you can see in the library manager that the XJ_FTP library has been added. You can click on the library to view the specific function blocks, error types, and other information in the library:



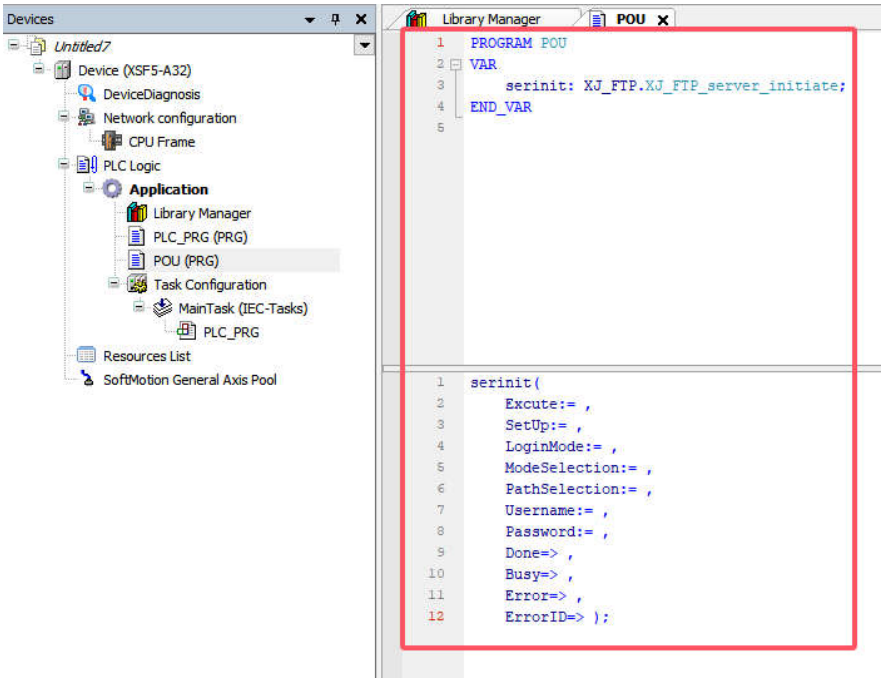

2. Add FTP function block.


Devices
Untitled7
Device (XSF5-A32)
DeviceDiagnosis
Network configuration
CPU Frame
PLC Logic
Application
Library Manager
PLC_PRG (PRG)
POU (PRG)
click POU
Task Configuration
MainTask (IEC-Tasks)
PLC_PRG
Resources List
SoftMotion General Axis Pool
Library Manager
POU
1 PROGRAM POU
2 VAR
3 END_VAR
4
1
right click this page, select input assistant
Advanced
Input Assistant...


Input Assistant
Text Search
Categories
XJ_FTP.XJ_FTP_server_initiate
1 item(s) found.
Name
Type
XJ_FTP.XJ_FTP_server_initiate
FUNCTION_BLOCK
Filter
None
Insert with arguments
Insert with namespace prefix
Documentation
FUNCTION_BLOCK XJ_FTP_server_initiate
FTP服务器初始化
Excute BOOL VAR_INPUT 上升沿启动功能块
SetUp BYTE VAR_INPUT 0: 不启动 1: 启动
LoginMode BYTE VAR_INPUT 模式选择。0: 匿名登录(无需用户名和用户密码) 1: 用户登录(设置用户名和密
ModeSelection BYTE VAR_INPUT 模式选择: 0: 被动模式 1: 主动模式
OK
Cancel

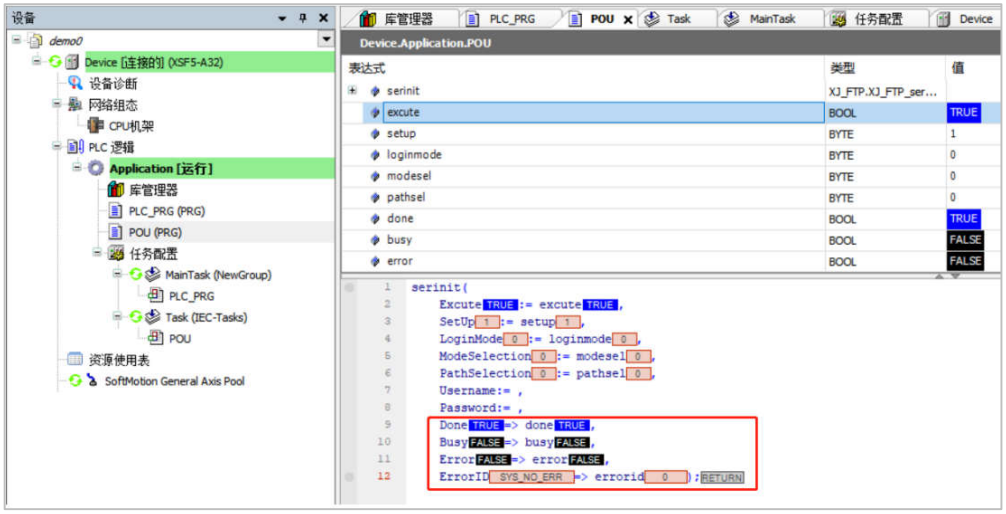



The added FTP server startup instance will appear in the POU:

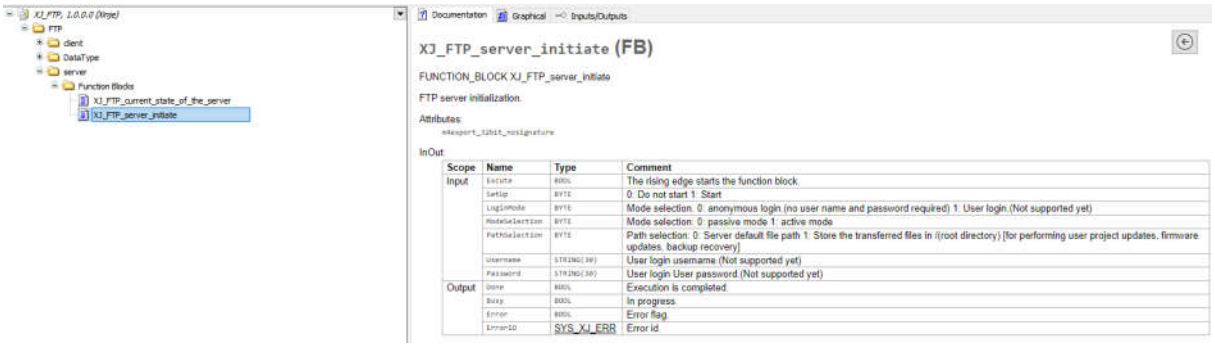


3. Start the FTP server on XSF5.

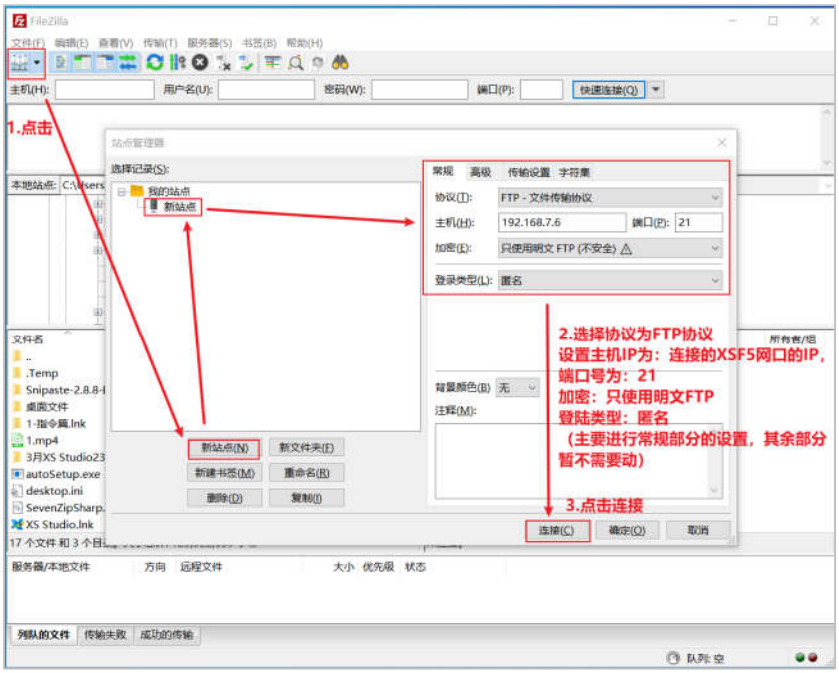
- **Input**
  - Excute: start function block at the rising edge
  - SetUp: 0: not start 1: start
  - LoginMode: mode selection
    0: Anonymous login (no username or user password required); 1: User login (currently not supported)
  - ModeSelection: mode selection
    0: Passive mode; 1: Active mode
  - PathSelection: path selection
    0: server defaulted file path (/FTP/Share_files_anonymity); 1: Store the transferred files in /(root directory) [for performing user program updates, firmware updates, backup and recovery] (/FTP/Temporary_file_storage)
  - Username: User login username (currently not supported)
  - Password: User login password (currently not supported)
- **Output**
  - Done: Execution completed
  - Busy: Executing
  - Error: Error flag
  - Error ID: Error code
- **Successfully enabled FTP server**




Parameter explanations can also be found in the library manager ->click on FTP library ->select the function block to view the specific parameter meanings in the function block:




4. Start FileZillaClient to connect FTP server.




Choose FTP as the protocol, set the host IP to the IP of the XSF5 network port connected,

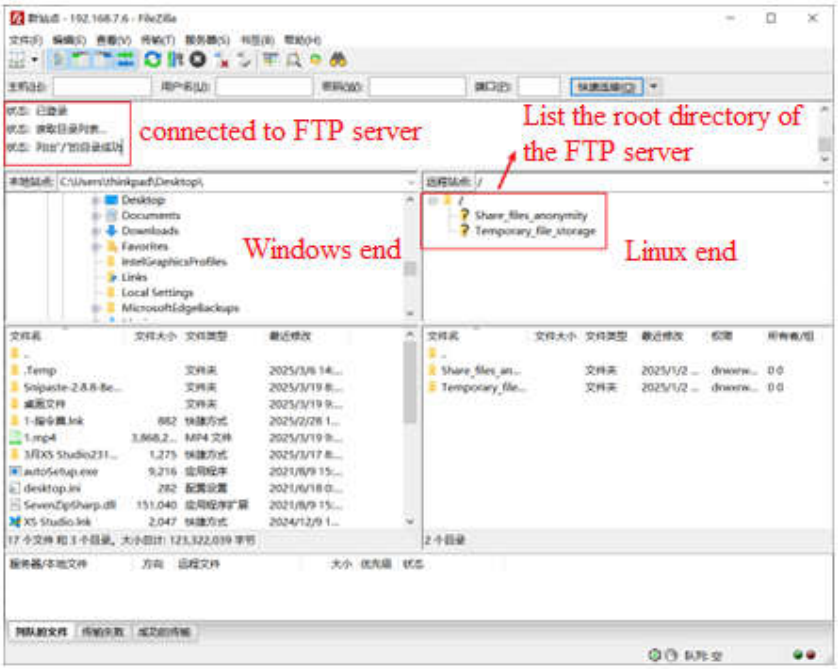
The port number is: 21

Encryption: Only use plaintext FTP

Login type: Anonymous

(Mainly responsible for setting up the regular parts, the rest of the parts no need to set).

Then click connect.




5. File transfer with FTP server.

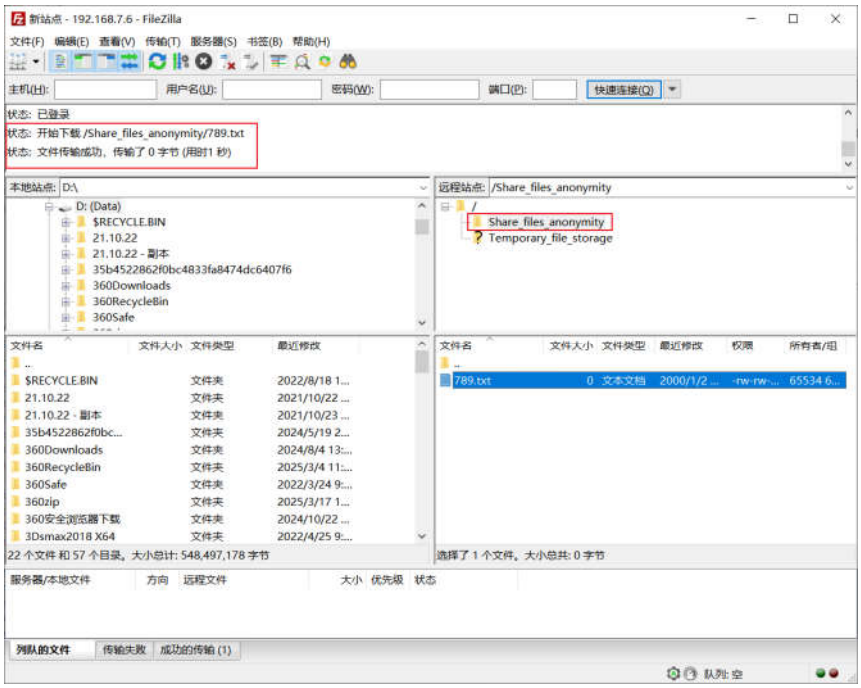
The selection of the directory is determined by the PathSelection value when starting the FTP server.

PathSelection:

0: Default file path on the server (/FTP/Share_files_anonymity);

1: Store the transferred files in /(root directory) [for performing user program updates, firmware updates, backup and recovery] (/FTP/Temporary_file_storage)

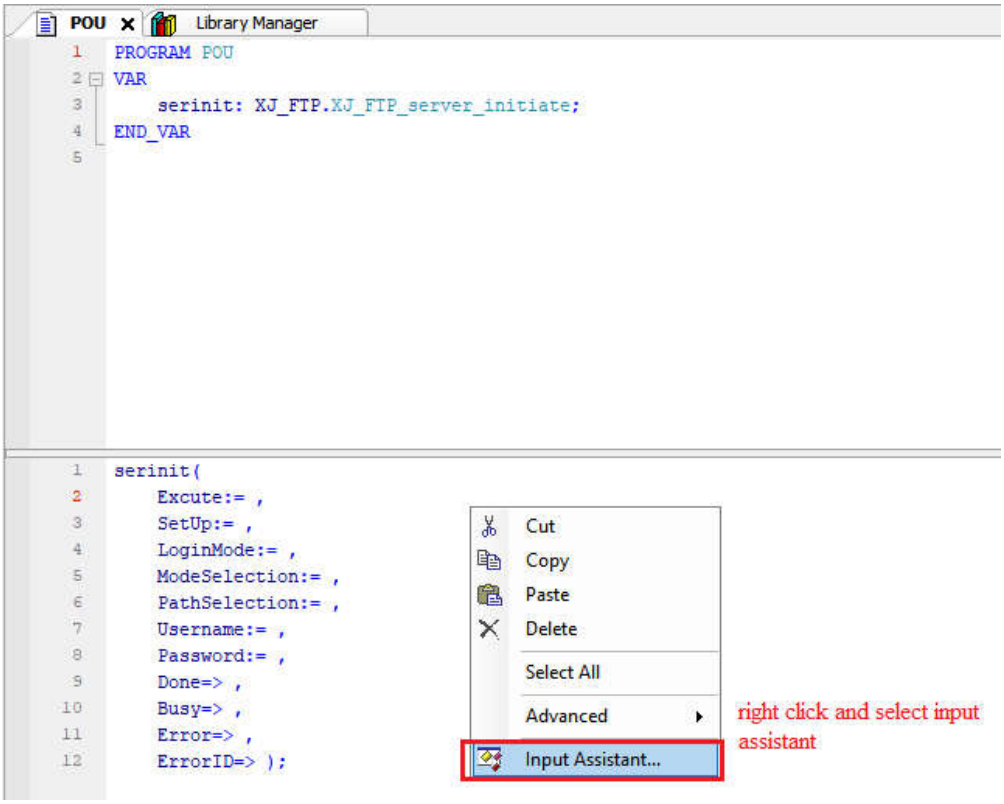
When clicking on the corresponding directory to transfer files, you can directly drag and drop files for uploading or downloading.



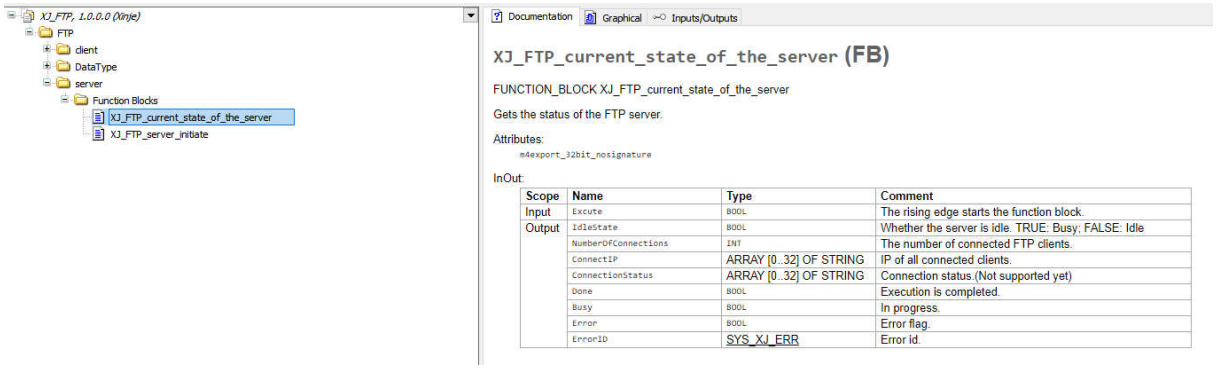
6. Check the current status of the FTP server.

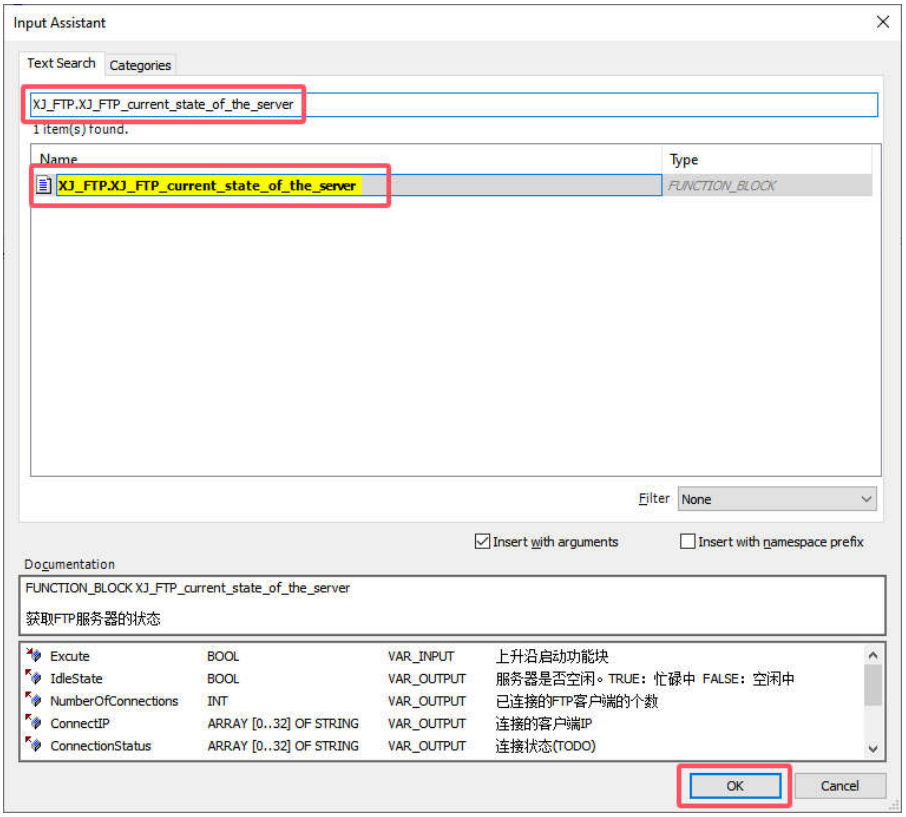
- **Input**
  - Excute: Rising edge start function block
  - IdleState: whether the server is idle
  - True: Busy; False: Idle
  - NumberOfConnections: The number of FTP clients that have been connected
  - ConnectIP: The IP address of the connected client
  - Connection Status: Connection Status (not currently supported)
- **Output**
  - Done: Execution completed
  - Busy: Executing
  - Error: Error flag
  - Error ID: Error code

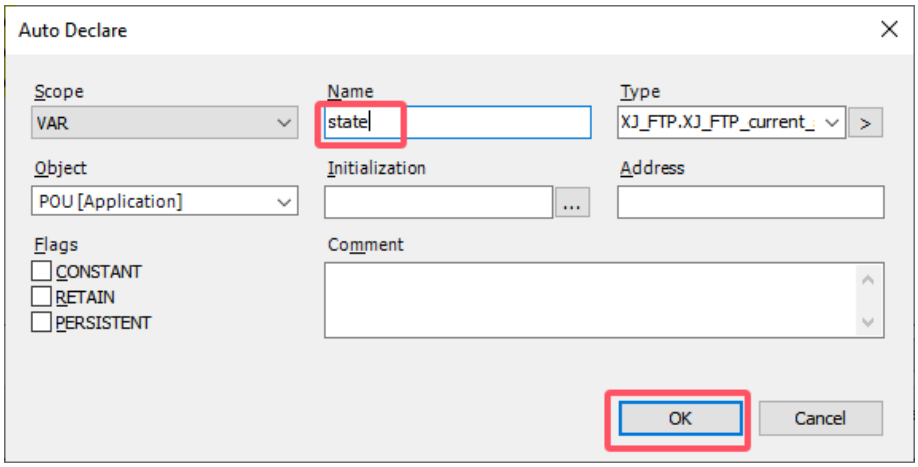
- **Add FTP server status function block**

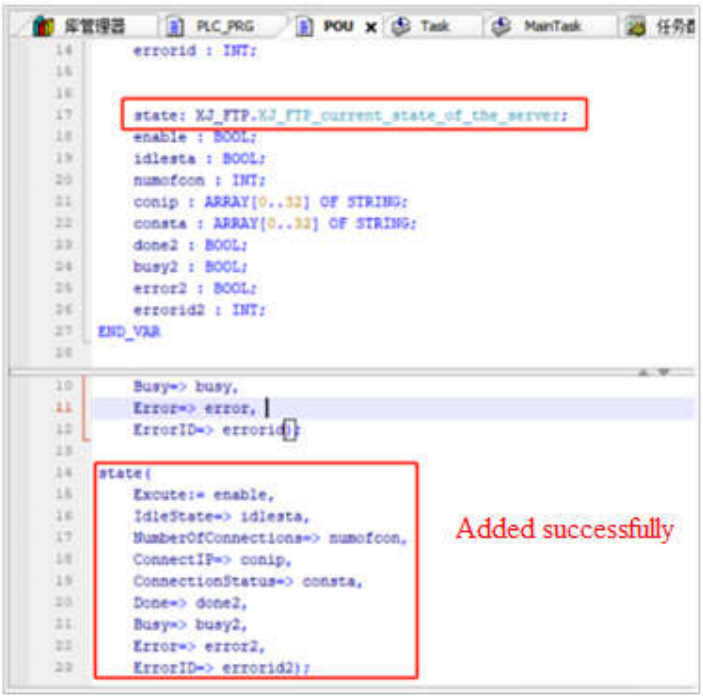



Parameter explanations can also be found in the library manager ->click on FTP library ->select the function block to view the specific parameter meanings in the function block:

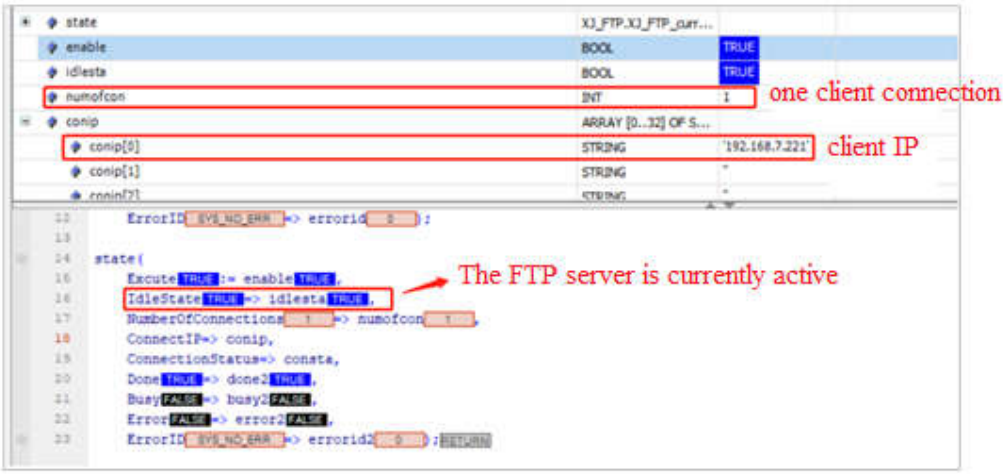

XJ_FTP, 1.0.0.0 (xinje)
FTP
client
DataType
server
Function Blocks
XJ_FTP_current_state_of_the_server
XJ_FTP_server_initiate
Documentation
Graphical
Inputs/Outputs
XJ_FTP_current_state_of_the_server (FB)
FUNCTION_BLOCK XJ_FTP_current_state_of_the_server
Gets the status of the FTP server.
Attributes:
m4export_32bit_nosignature
InOut:
Scope Name Type Comment
Input Excute BOOL The rising edge starts the function block.
Output IdleState BOOL Whether the server is idle. TRUE: Busy; FALSE: Idle
NumberOfConnections INT The number of connected FTP clients.
ConnectIP ARRAY [0..32] OF STRING IP of all connected clients.
ConnectionStatus ARRAY [0..32] OF STRING Connection status.(Not supported yet)
Done BOOL Execution is completed.
Busy BOOL In progress.
Error BOOL Error flag.
ErrorID SYS_XJ_ERR Error id.


Input Assistant
Text Search
Categories
XJ_FTP.XJ_FTP_current_state_of_the_server
1 item(s) found.
Name
Type
XJ_FTP.XJ_FTP_current_state_of_the_server
FUNCTION_BLOCK
Filter
None
Insert with arguments
Insert with namespace prefix
Documentation
FUNCTION_BLOCK XJ_FTP_current_state_of_the_server
获取FTP服务器的状态
Excute BOOL VAR_INPUT 上升沿启动功能块
IdleState BOOL VAR_OUTPUT 服务器是否空闲。TRUE：忙碌中 FALSE：空闲中
NumberOfConnections INT VAR_OUTPUT 已连接的FTP客户端的个数
ConnectIP ARRAY [0..32] OF STRING VAR_OUTPUT 连接的客户端IP
ConnectionStatus ARRAY [0..32] OF STRING VAR_OUTPUT 连接状态(TODO)
OK
Cancel

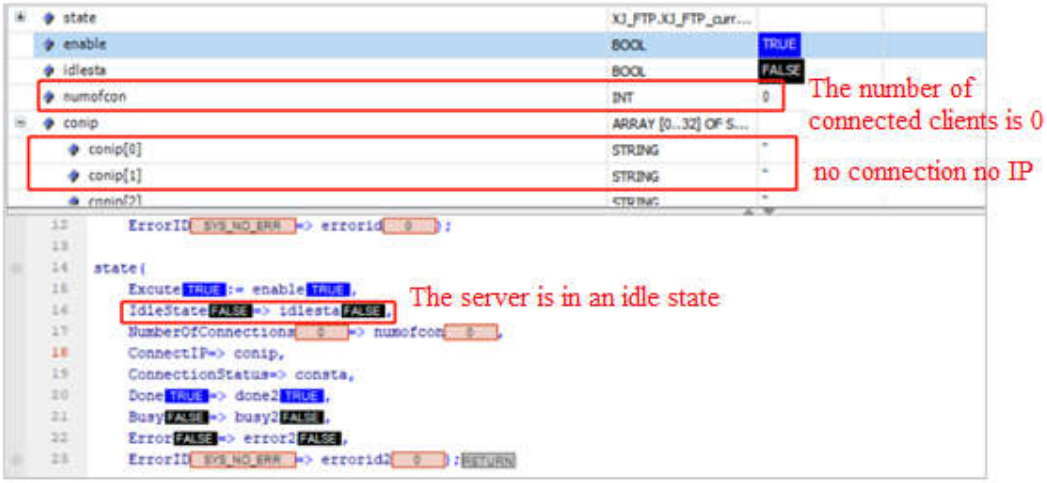

Auto Declare
Scope
VAR
Name
state
Type
XJ_FTP.XJ_FTP_current_
Object
POU [Application]
Initialization
Address
Flags
CONSTANT
RETAIN
PERSISTENT
Comment
OK
Cancel




Check the current status of the FTP server:




After disconnecting, retrieve the current status of the FTP server:

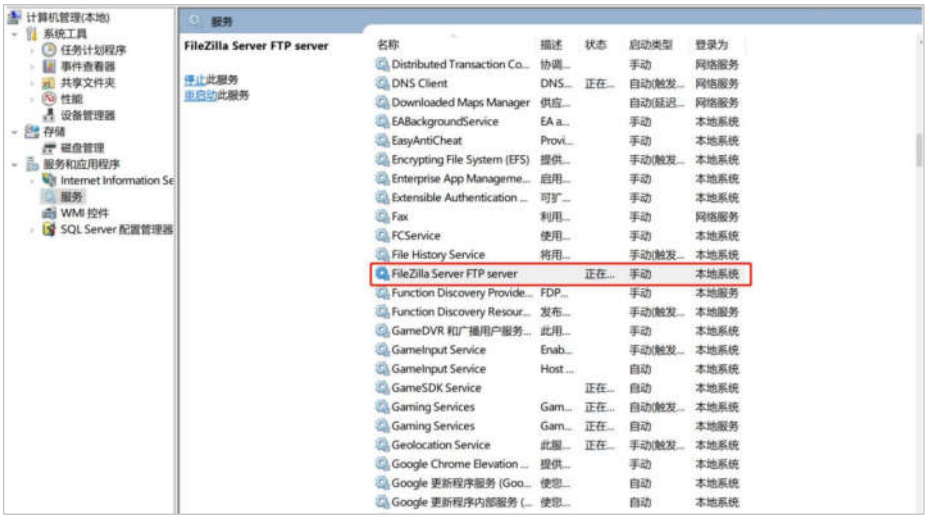


- Using XSF5 as an FTP client

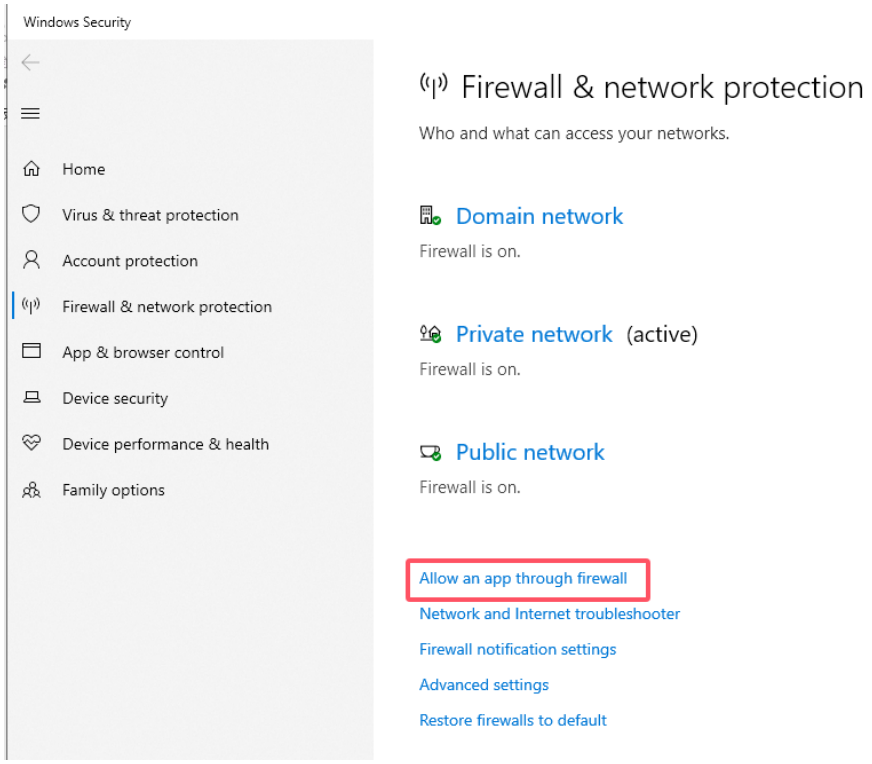
Use XSF as an FTP client to connect to the FTP server software FileZillaServer.

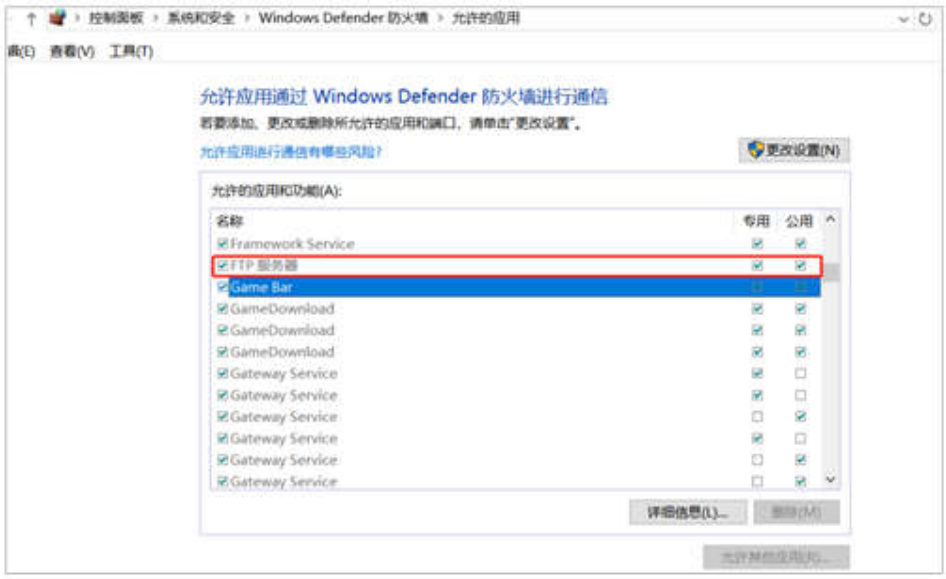
**Using FileZillaServer**

1. Before using FileZillaServer, right-click on this computer ->Manage ->Services and Applications ->Services, find and start the FTP server service:

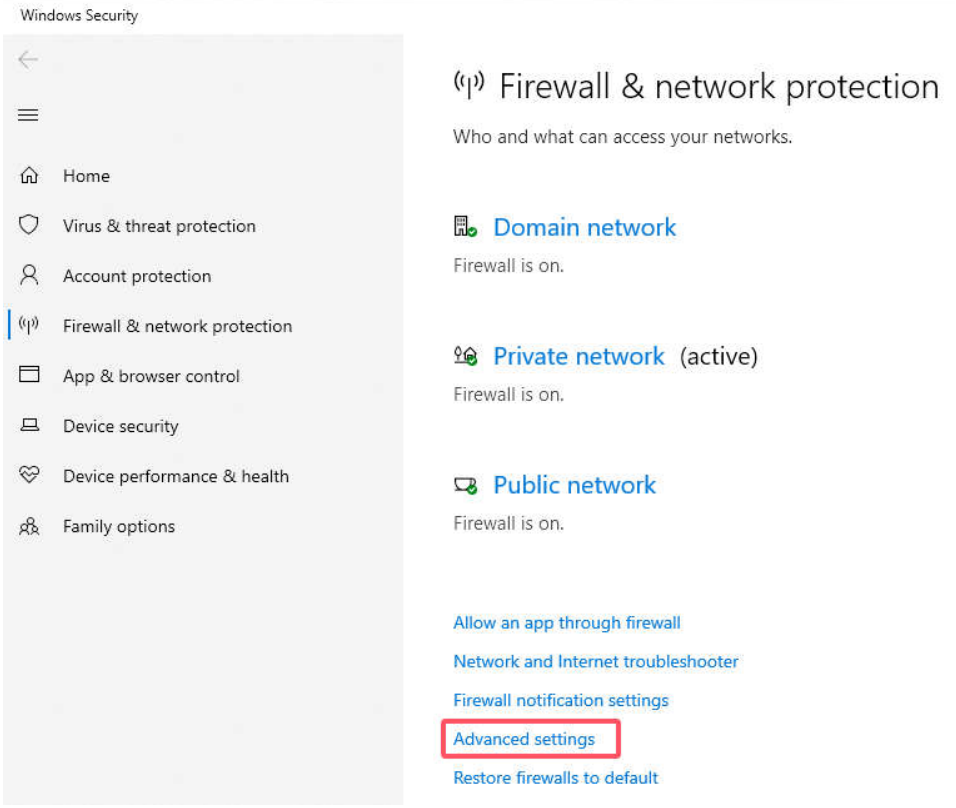


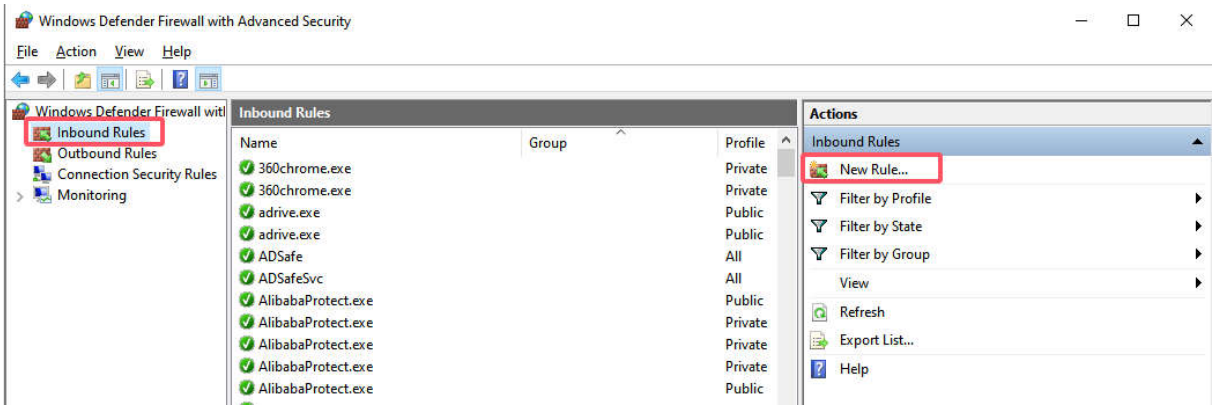

2. Allow FTP servers to communicate through firewalls (or disable firewalls):

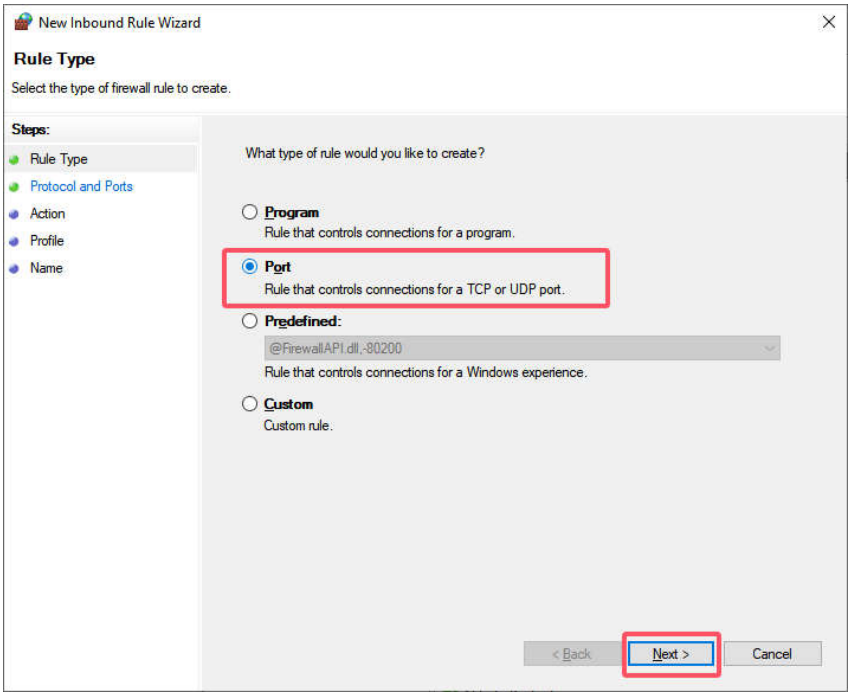


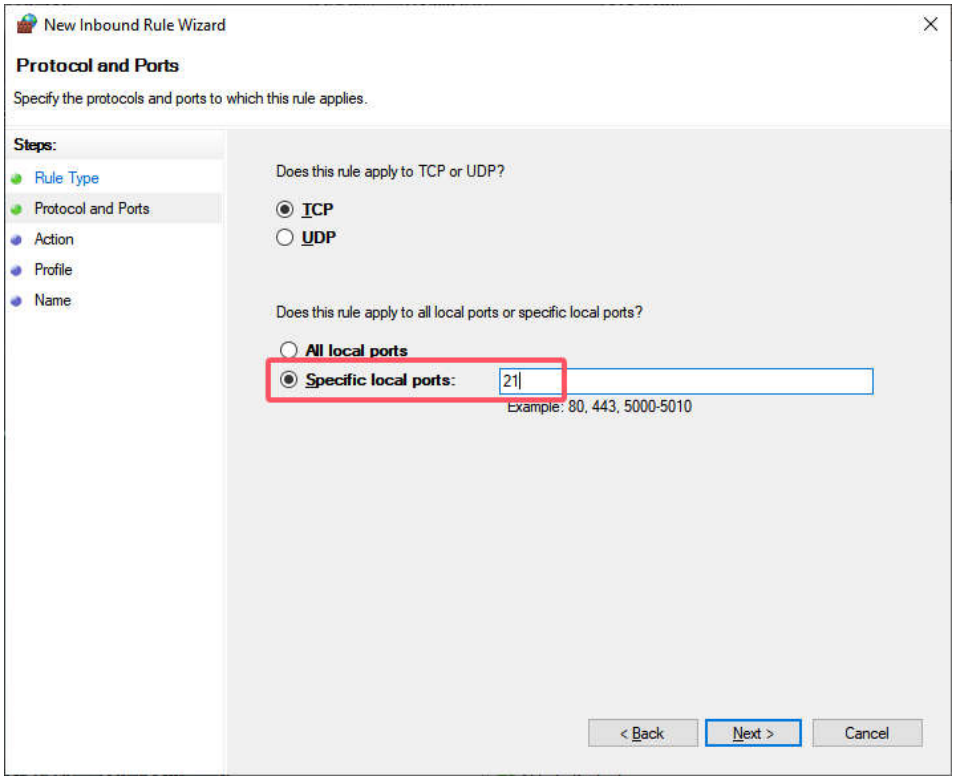



3. Configure firewall to release port 21, click Advanced Settings ->Inbound Rules ->New Rule:

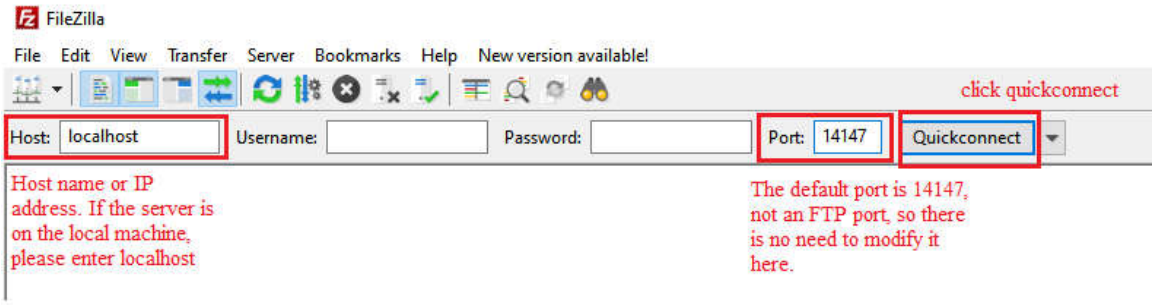


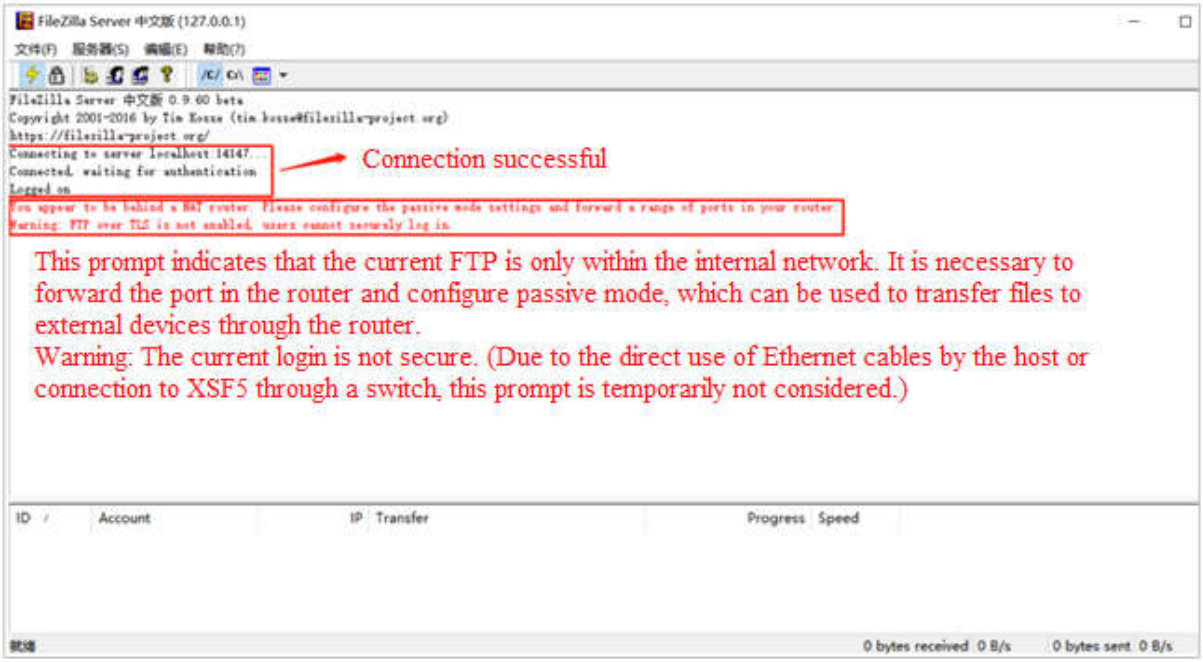

Windows Defender Firewall with Advanced Security
File Action View Help
Windows Defender Firewall witl
Inbound Rules
Outbound Rules
Connection Security Rules
Monitoring
Inbound Rules
Name
Group
Profile
360chrome.exe Private
360chrome.exe Private
adrive.exe Public
adrive.exe Public
ADSafe All
ADSafeSvc All
AlibabaProtect.exe Public
AlibabaProtect.exe Private
AlibabaProtect.exe Private
AlibabaProtect.exe Private
AlibabaProtect.exe Public
Actions
Inbound Rules
New Rule...
Filter by Profile
Filter by State
Filter by Group
View
Refresh
Export List...
Help


New Inbound Rule Wizard
Rule Type
Select the type of firewall rule to create.
Steps:
Rule Type
Protocol and Ports
Action
Profile
Name
What type of rule would you like to create?
Program
Rule that controls connections for a program.
Port
Rule that controls connections for a TCP or UDP port.
Predefined:
@FirewallAPI.dll,-80200
Rule that controls connections for a Windows experience.
Custom
Custom rule.
< Back
Next >
Cancel

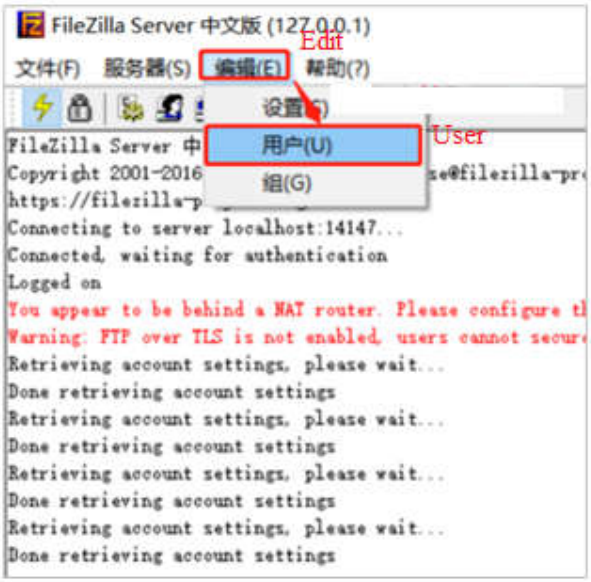



4. Click on the FileZillaServer icon to open it, and a connection interface will pop up directly. If the connection is successful, the software interface will prompt:

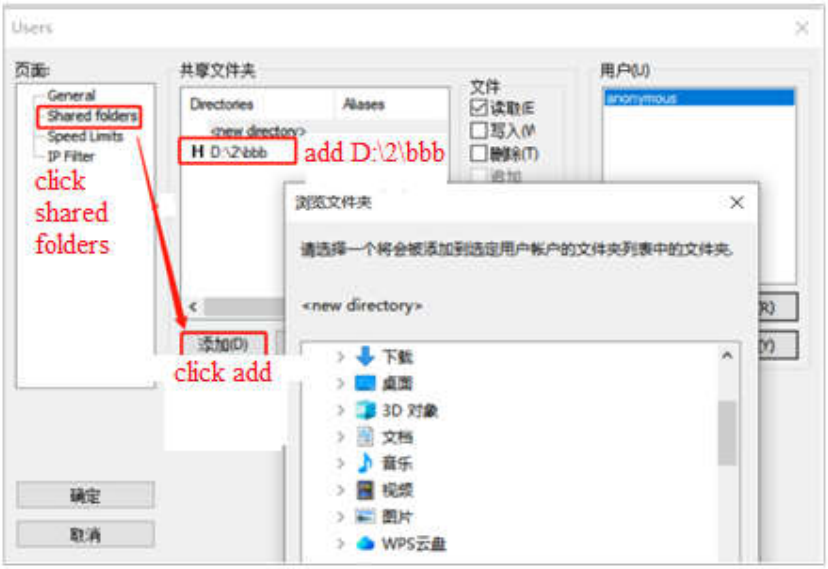





5. Click Edit ->User. Usually, user groups are added first, which makes it easier to control the permissions of users on the server. Due to the current support of only anonymous user login, users can be added directly by setting username: anonymous and password: anonymous

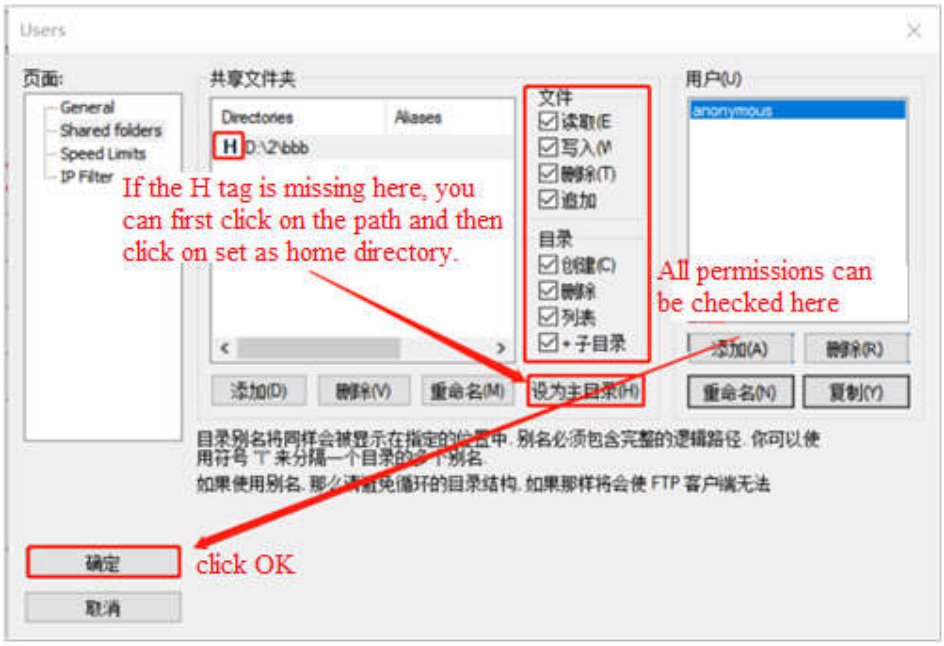


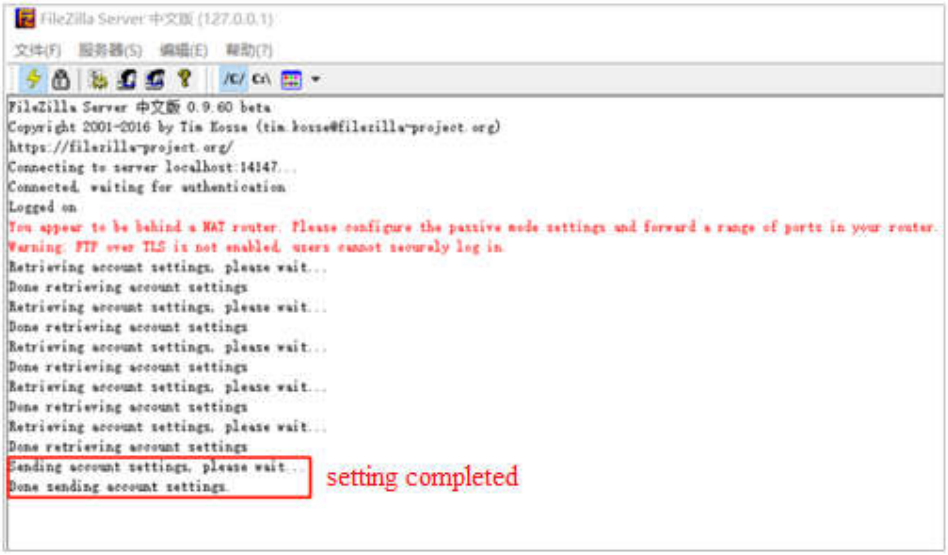
6. Configure shared folders. Here is an example of the D:\2\bbb that I created myself:



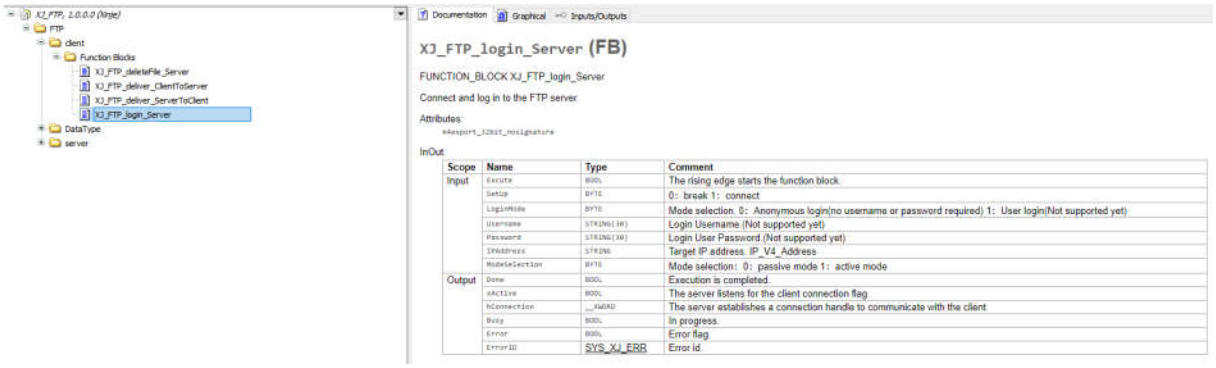
The file path set here is the root file directory of the FTP server. Do not transfer files in this directory. You need to create a folder in this directory, place the files that need to be transferred in the created folder, and then perform file transfer. For example, on bbb/123.

Click on the added shared folder to set the client's permissions for operating shared folders/files:

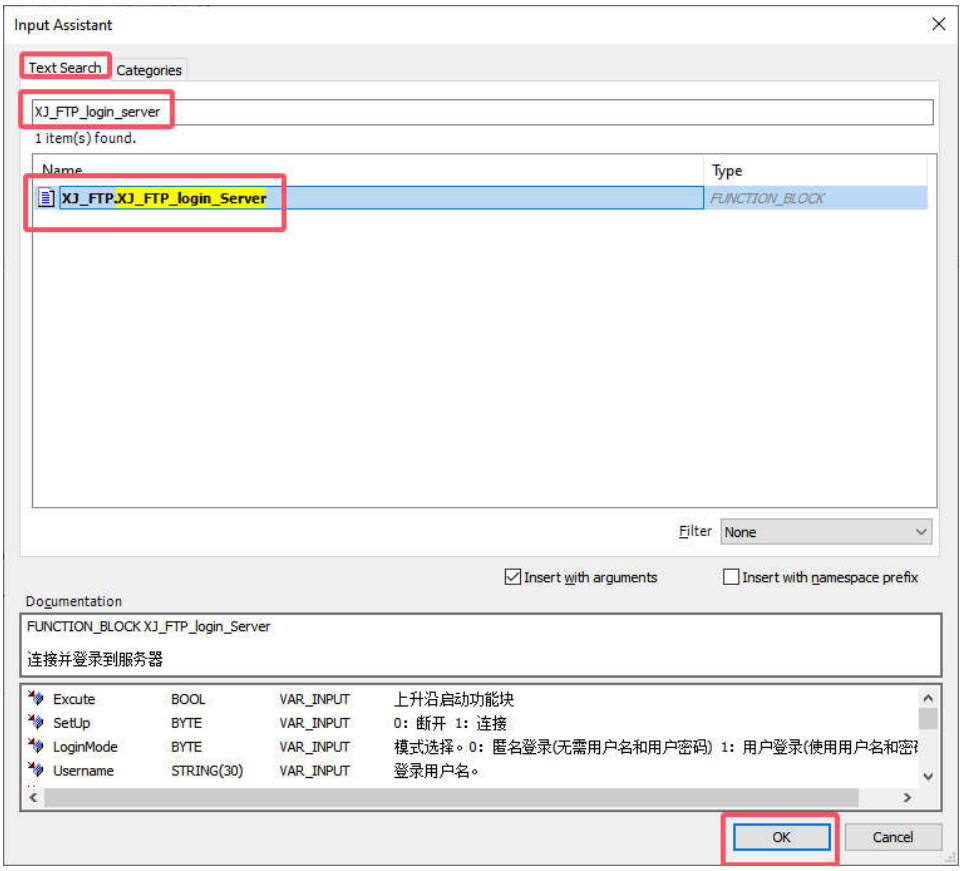


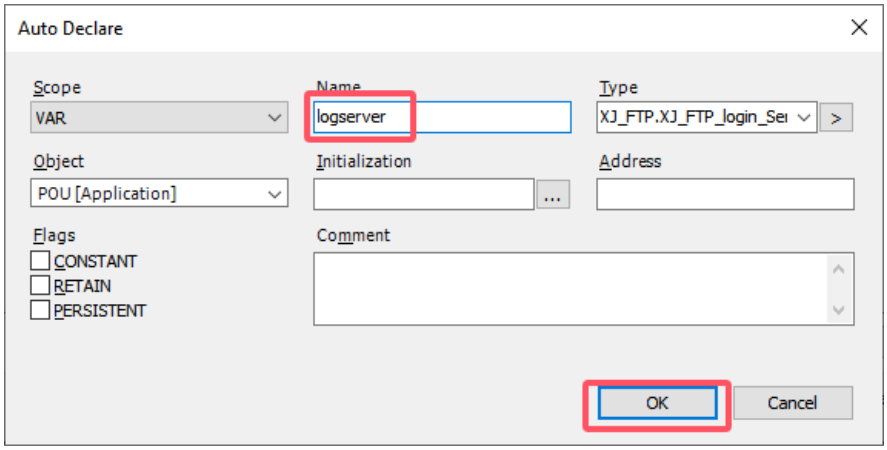



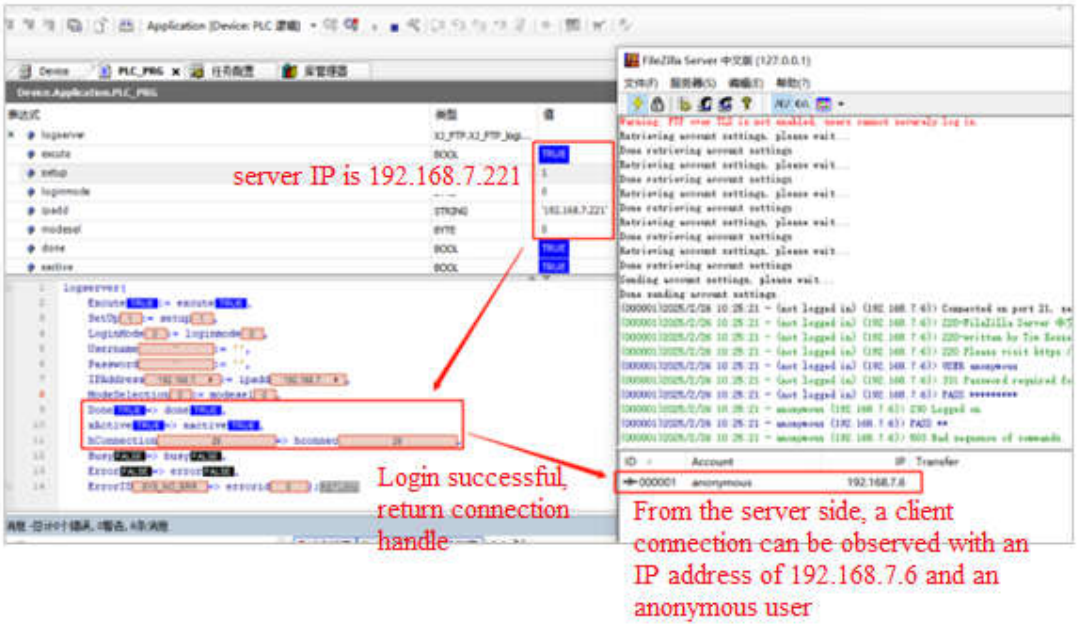
### XSF5 starts FTP client to connect to FTP server




Right click and select "Input Assistant". In the pop-up interface, enter "XN_FTP_login_Server":





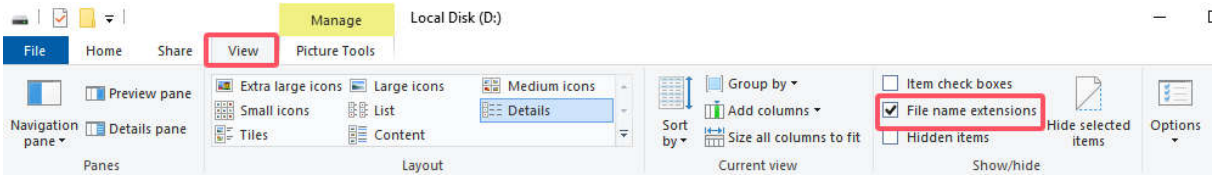

When not in operation for a period of time, the FTP server's own mechanism will disconnect the connection with the client:

```
(000001)2025/2/26 10:25:21 - anonymous (192.168.7.6)> 503 Bad sequence of commands.
(000001)2025/2/26 10:27:21 - anonymous (192.168.7.6)> 421 Connection timed out.
(000001)2025/2/26 10:27:21 - anonymous (192.168.7.6)> disconnected.
```

At this point, it is necessary to close the last opened connection handle, that is, first set setup to 0, and then change excute to trigger a rising edge and close the connection handle. Then set the setup to 1, fill in the server-side IP and other information, and trigger a rising edge to re-establish the connection.

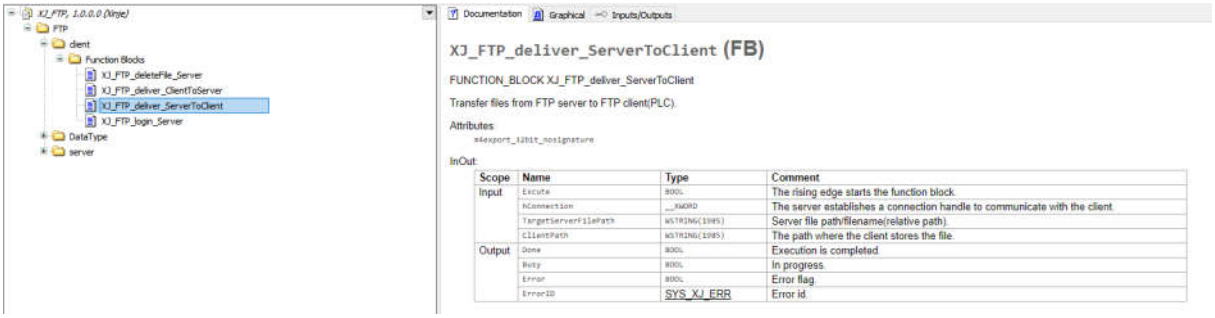
**XSF5 is used as a client to download files from the server**

Before transferring files, it is necessary to enable the file name extensions on the Windows side. In the Windows 10 system, the file extensions can be enabled by opening any folder, clicking "View" in the upper left corner of the page, and selecting the file name extensions:

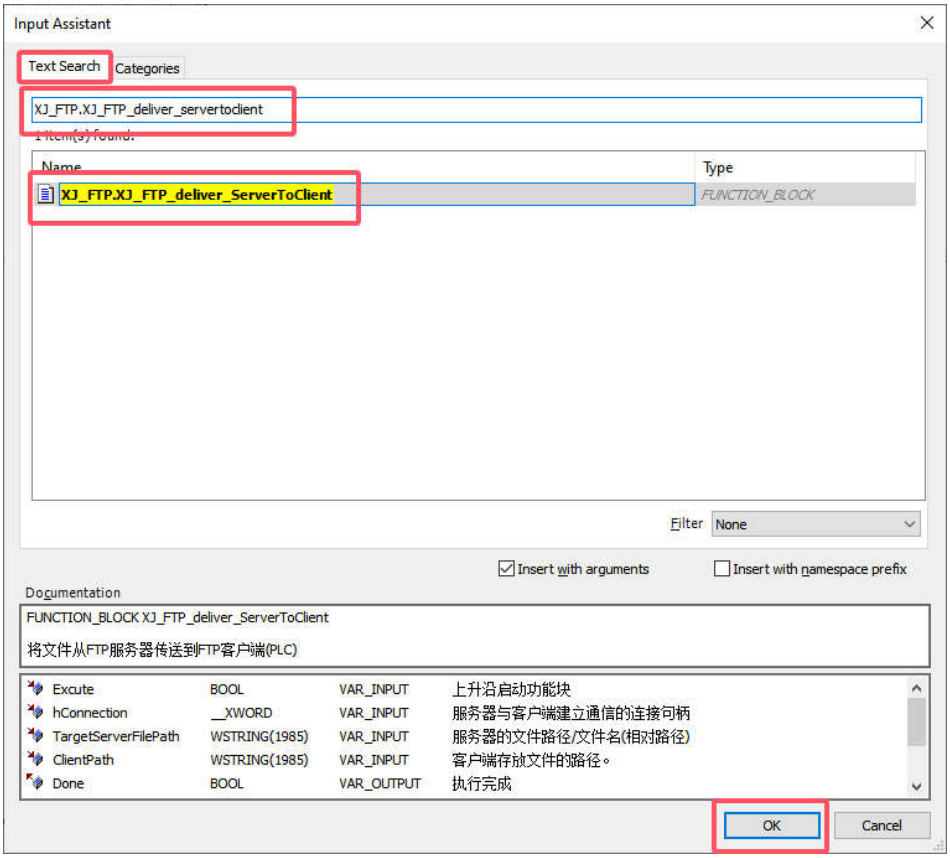


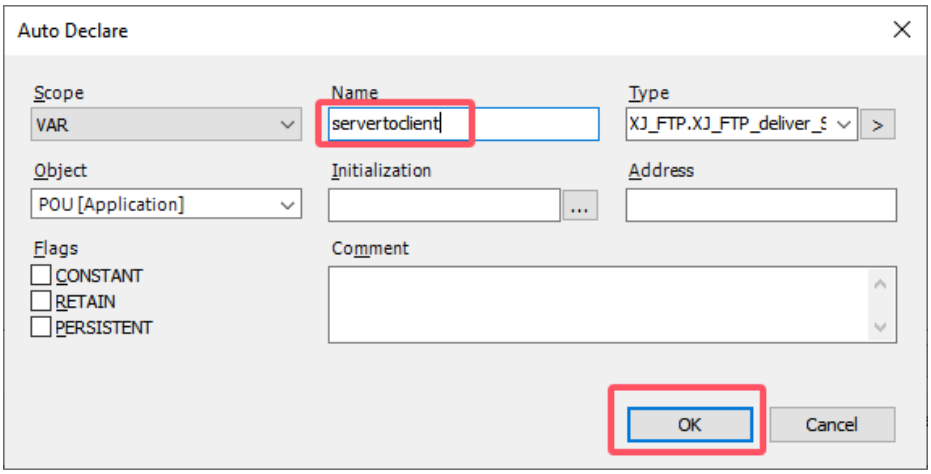

In the Windows 11 system, the file extensions activation method is to open any folder, click View ->Display ->Check the file name extensions.

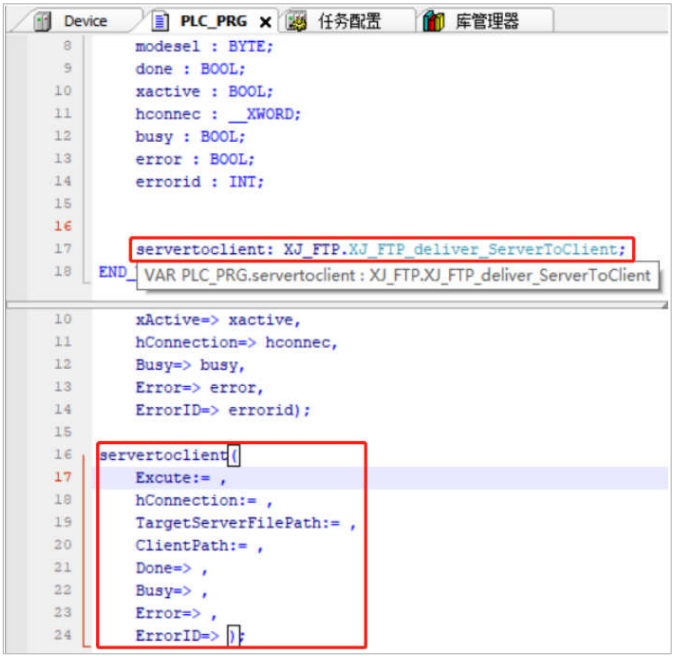
XJ_FTP_deliver_ServerToClient function block parameter explanation:



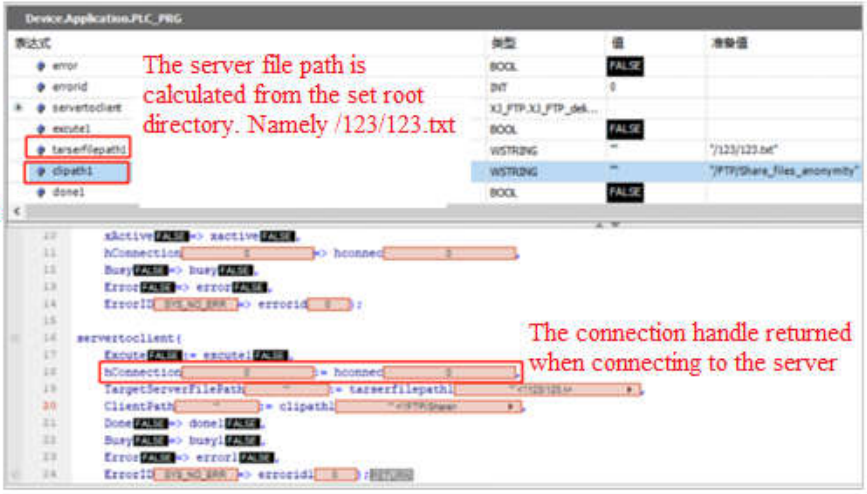

Add the function block:

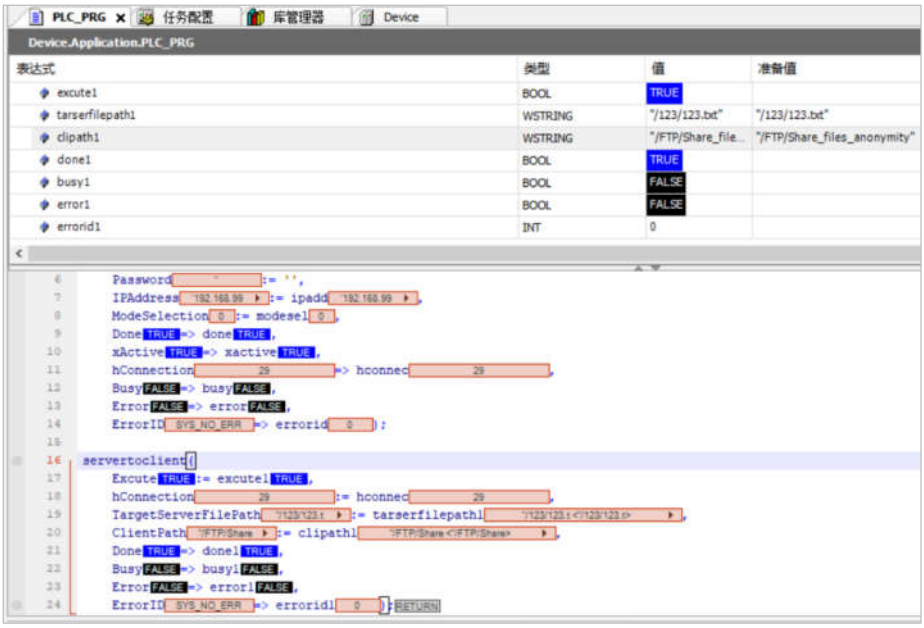


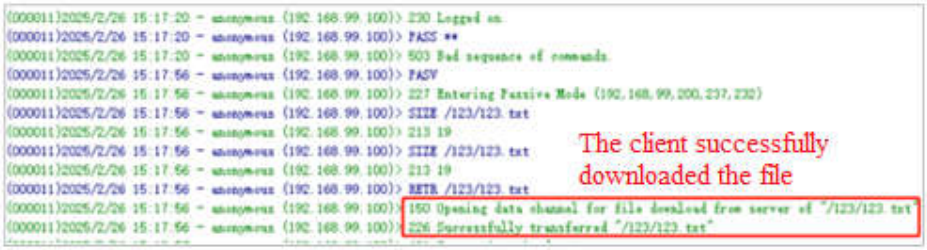


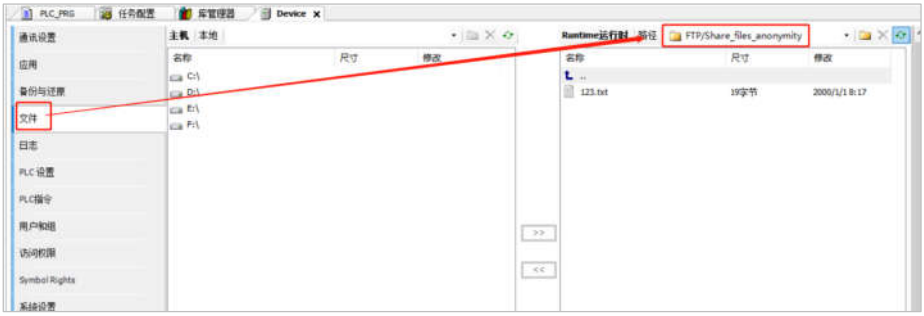

Client storage path: when XSF5 is a client, usually stores in the /FTP/share_files_anonymity folder





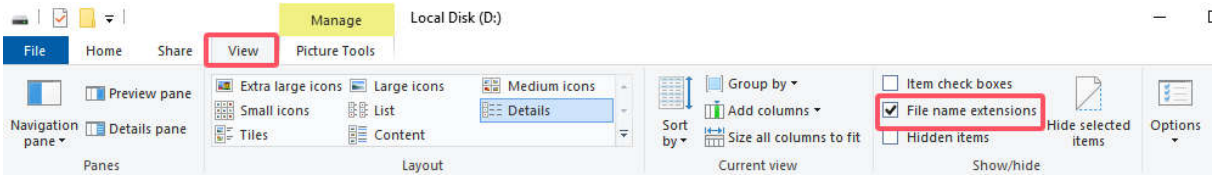




Click Device->Files->/FTP/Share_files_anonymity, file 123.txt already existed.



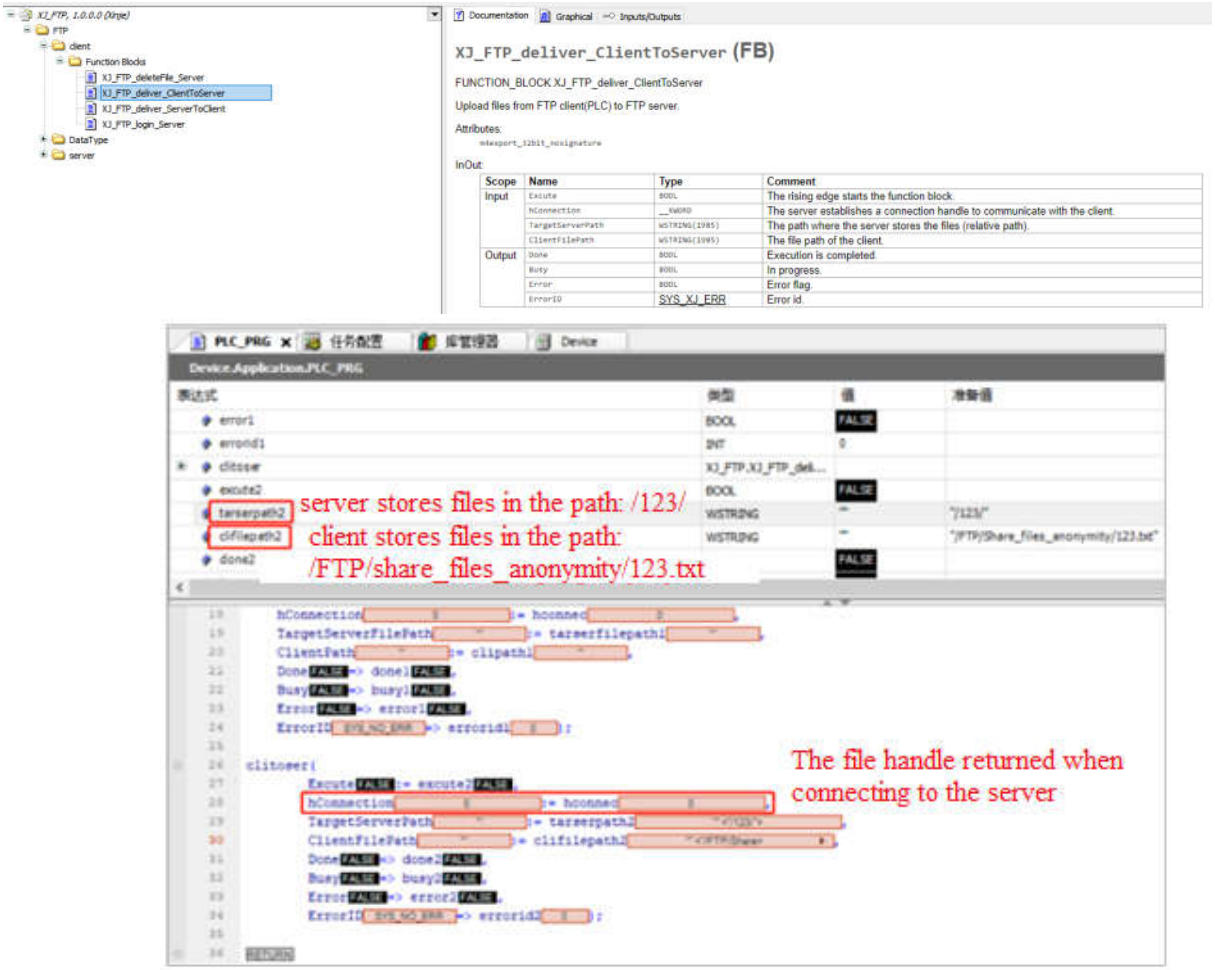
### XSF5 uploads files from the client to the server

Before transferring files, it is necessary to enable the file name extensions on the Windows side. In the Windows 10 system, the file extensions can be enabled by opening any folder, clicking "View" in the upper left corner of the page, and selecting the file name extensions:




In the Windows 11 system, the file extensions activation method is to open any folder, click View ->Display ->Check the file name extensions.

XJ_FTP_deliver_ClientToServer function block parameter explanation:

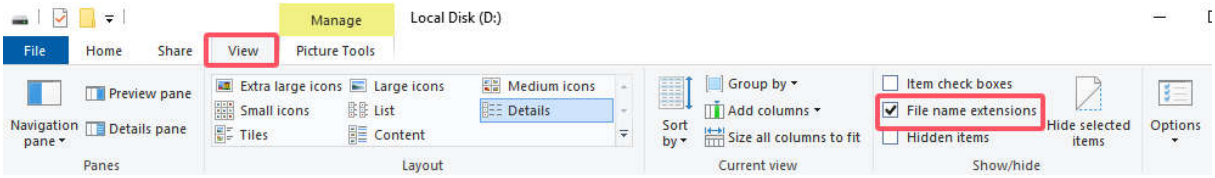


```
(000012)2025/2/26 16:09:12 - anonymous (192.168.99.100)> CWD /123/
(000012)2025/2/26 16:09:12 - anonymous (192.168.99.100)> 250 CWD successful. "/123" is current directory.
(000012)2025/2/26 16:09:12 - anonymous (192.168.99.100)> STOR /123/123.txt
(000012)2025/2/26 16:09:12 - anonymous (192.168.99.100)> 150 Opening data channel for file upload to server of "/123/123.txt"
(000012)2025/2/26 16:09:12 - anonymous (192.168.99.100)> 226 Successfully transferred "/123/123.txt"
```

Upon opening the directory D:\2\bbb\123, it was discovered that the 123.txt file already existed.

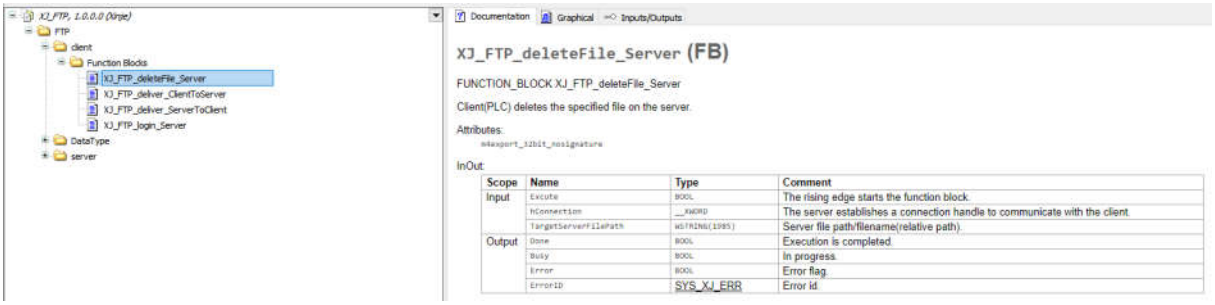
**To delete server files with the XSF5 client**

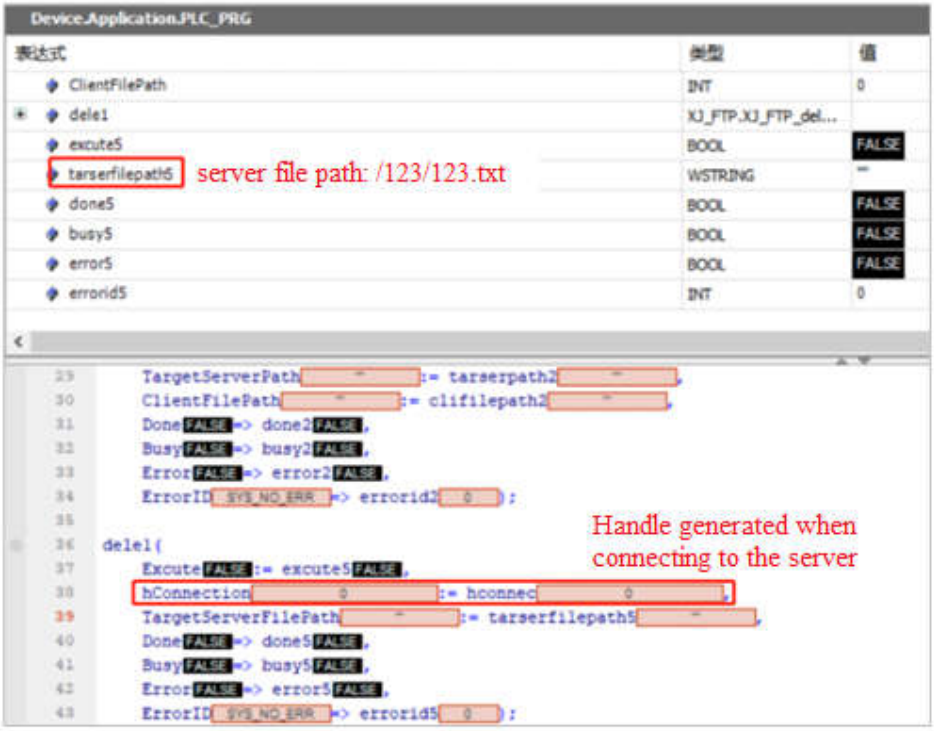
Before transferring files, it is necessary to enable the file name extensions on the Windows side. In the Windows 10 system, the file extensions can be enabled by opening any folder, clicking "View" in the upper left corner of the page, and selecting the file name extensions:

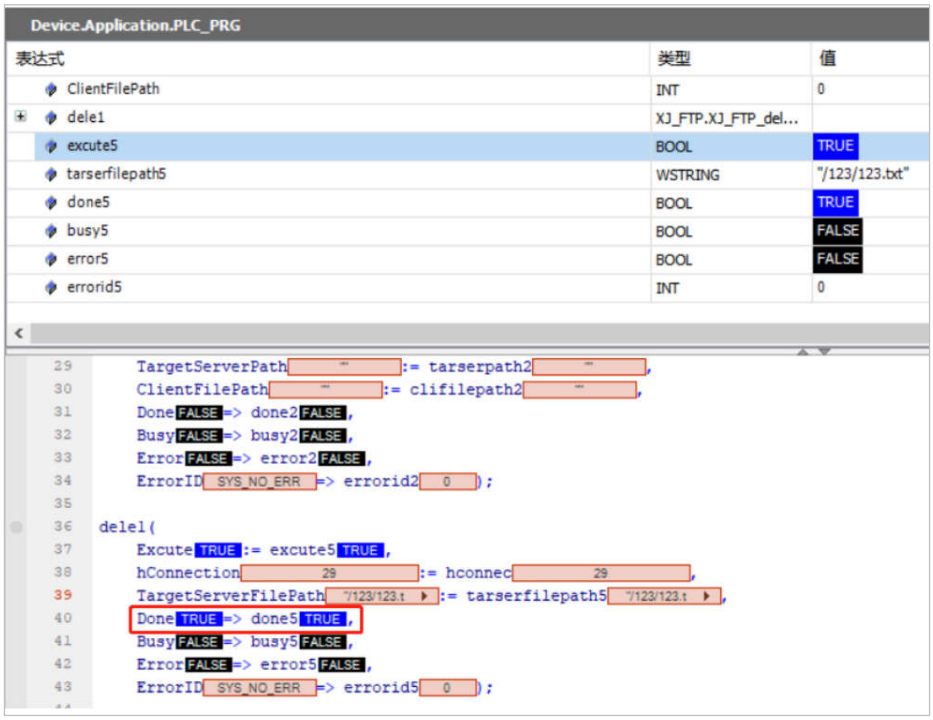


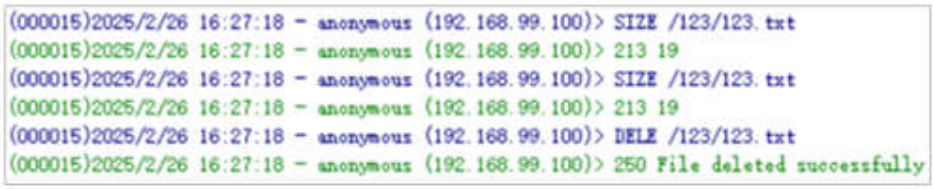

In the Windows 11 system, the file extensions activation method is to open any folder, click View ->Display ->Check the file name extensions.

XJ_FTP_deleteFile_Server function block parameter explanation:




Device.Application.PLC_PRG
表达式
类型
值
ClientFilePath
INT
0
dele1
XJ_FTP.XJ_FTP_del...
excute5
BOOL
TRUE
tarserfilepath5
WSTRING
"/123/123.txt"
done5
BOOL
TRUE
busy5
BOOL
FALSE
error5
BOOL
FALSE
errorid5
INT
0
TargetServerPath := tarserpath2,
ClientFilePath := clifilepath2,
Done => done2,
Busy => busy2,
Error => error2,
ErrorID => errorid2 );
dele1(
Excute := excute5,
hConnection := hconnec,
TargetServerFilePath := tarserfilepath5,
Done => done5,
Busy => busy5,
Error => error5,
ErrorID => errorid5 );


(000015)2025/2/26 16:27:18 - anonymous (192.168.99.100)> SIZE /123/123.txt
(000015)2025/2/26 16:27:18 - anonymous (192.168.99.100)> 213 19
(000015)2025/2/26 16:27:18 - anonymous (192.168.99.100)> SIZE /123/123.txt
(000015)2025/2/26 16:27:18 - anonymous (192.168.99.100)> 213 19
(000015)2025/2/26 16:27:18 - anonymous (192.168.99.100)> DELE /123/123.txt
(000015)2025/2/26 16:27:18 - anonymous (192.168.99.100)> 250 File deleted successfully

9-4. FTP communication error code

| Error ID | Error code | Description |
|---|---|---|
| SYS_NO_ERR | 0 | No error |
| SYS_FTP_SERVER_INIT_FAILED | 1 | FTP server initialization failed |
| SYS_USERNAME_OR_PASSWORD_INVALID | 2 | Invalid username or password |
| SYS_FTP_SERVER_NOT_ENABLED | 3 | FTP server not enabled |
| SYS_FTP_SERVER_NUMBER_REACH_MAX | 4 | The maximum number of FTP servers has been reached |
| SYS_FTP_CLIENTS_CONNECT_SERVER_NUMBER_REACH_MAX | 5 | The maximum number of clients connected to the FTP server has been reached |
| SYS_FTP_LOGIN_SERVER_FAILED | 6 | Failed to connect to FTP server |
| SYS_USERNAME_OR_PASSWORD_ERROR | 7 | Login error. Incorrect username or password |
| SYS_TARGET_FILE_NOT_EXIST | 8 | The target file does not exist |
| SYS_TRANSMISSION_PATH_NOT_EXIST | 9 | Path does not exist |
| SYS_FTP_SERVER_CLOSE_FAILED | 10 | FTP server shutdown failed |
| SYS_FTP_SERVER_STATE_FAILED | 11 | Failed to retrieve FTP server status |
| SYS_PARAMETER_ERROR | 12 | Input parameter error |
| SYS_FTP_CLIENT_NOT_EXIST | 13 | The client does not exist |
| SYS_FILE_TRANSFER_FAILURE | 14 | File transfer failed |
| SYS_CONNECTION_HANDLE_INVALID | 15 | Connection handle failure |
| SYS_SEND_OR_RECEIVE_INFO_FAILURE | 16 | Sending or receiving information failed |
| SYS_DELETE_FILE_FAILURE | 17 | Failed to delete file |

WUXI XINJE ELECTRIC CO., LTD.

Address: 816 Jianshe West Road, Binhu District, Wuxi City, Jiangsu Province, China

Tel: 0510-85134136 Fax: 0510-85111290

Website: www.xinje.com Email: sales@xinje.com